Errata

Title & Document Type: 16520A/21A Programming Reference

Manual Part Number: 16520-90908

Revision Date: September 1988

HP References in this Manual

This manual may contain references to HP or Hewlett-Packard. Please note that Hewlett-Packard's former test and measurement, semiconductor products and chemical analysis businesses are now part of Agilent Technologies. We have made no changes to this manual copy. The HP XXXX referred to in this document is now the Agilent XXXX. For example, model number HP8648A is now model number Agilent 8648A.

About this Manual

We've added this manual to the Agilent website in an effort to help you support your product. This manual provides the best information we could find. It may be incomplete or contain dated information, and the scan quality may not be ideal. If we find a better copy in the future, we will add it to the Agilent website.

Support for Your Product

Agilent no longer sells or supports this product. You will find any other available product information on the Agilent Test & Measurement website:

www.tm.agilent.com

Search for the model number of this product, and the resulting product page will guide you to any available information. Our service centers may be able to perform calibration if no repair parts are needed, but no other support from Agilent is available.

Agilent Technologies

Programming Reference

HP 16520A/16521A
Pattern Generator Module

for the HP 16500A Logic Analysis System

Manual Part Number 16520-90908

Printed in U.S.A. September 1988

Product Warranty

This Hewlett-Packard product has a warranty against defects in material and workmanship for a period of three years from date of shipment. During warranty period, Hewlett-Packard Company will, at its option, either repair or replace products that prove to be defective.

For warranty service or repair, this product must be returned to a service facility designated by Hewlett-Packard. However, warranty service for products installed by Hewlett-Packard and certain other products designated by Hewlett-Packard will be performed at the Buyer's facility at no charge within the Hewlett-Packard service travel area. Outside Hewlett-Packard service travel areas, warranty service will be performed at the Buyer's facility only upon Hewlett-Packard's prior agreement and the Buyer shall pay Hewlett-Packard's round trip travel expenses.

For products returned to Hewlett-Packard for warranty service, the Buyer shall prepay shipping charges to Hewlett-Packard and Hewlett-Packard shall pay shipping charges to return the product to the Buyer. However, the Buyer shall pay all shipping charges, duties, and taxes for products returned to Hewlett-Packard from another country.

Hewlett-Packard warrants that its software and firmware designated by Hewlett-Packard for use with an instrument will execute its programming instructions when properly installed on that instrument. Hewlett-Packard does not warrant that the operation of the instrument software, or firmware will be uninterrupted or error free.

Limitation of Warranty

The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by the Buyer, Buyer-supplied software or interfacing, unauthorized modification or misuse, operation outside of the environmental specifications for the product, or improper site preparation or maintenance.

NO OTHER WARRANTY IS EXPRESSED OR IMPLIED. HEWLETT-PACKARD SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTIES OR MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Exclusive Remedies THE REMEDIES PROVIDED HEREIN ARE THE BUYER'S SOLE AND EXCLUSIVE REMEDIES. HEWLETT-PACKARD SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER BASED ON CONTRACT, TORT, OR ANY OTHER LEGAL THEORY.

Assistance Product maintenance agreements and other customer assistance agreements are available for Hewlett-Packard products.

For any assistance, contact your nearest Hewlett-Packard Sales and Service Office.

Certification Hewlett-Packard Company certifies that this product met its published specifications at the time of shipment from the factory. Hewlett-Packard further certifies that its calibration measurements are traceable to the United States National Bureau of Standards, to the extent allowed by the Bureau's calibration facility, and to the calibration facilities of other International Standards Organization members.

Safety This product has been designed and tested according to International Safety Requirements. To ensure safe operation and to keep the product safe, the information, cautions, and warnings in this manual must be heeded.

Table of Contents

Chapter 3

Chapter 4

Chapter 5

Chapter 6

Appendix A

Programming the HP 16520A/16521A

1

Introduction

This manual combined with the *HP 16500A Programming Reference* manual provides you with the information needed to program the HP 16520A/HP 16521A pattern generator module. Each module has its own manual to supplement the mainframe manual since not all mainframes will be configured with the same modules.

About This Manual

This manual is organized into six chapters. The first chapter contains:

- General information and instructions to help you get started
- Mainframe system commands that are frequently used with the pattern generator module
- HP 16520A/HP 16521A Pattern Generator command tree
- Alphabetic command-to-subsystem directory

Chapter two contains the module level commands. Chapters three through six contain the subsystem commands for the pattern generator.

Chapter six contains the Symbol Subsystem.

Error messages for the HP 16520A are included in generic system error messages and are in the *HP 16500A Programming Reference* manual.

Appendix A contains information on the SYSTem:DATA and SYSTem:SETup commands for this module.

Programming the HP 16520A Pattern Generator

This section introduces you to the basic command structure used to program the pattern generator. Also included is an example program that uses the two pods of the master card.

Selecting the Module

Before you can program the pattern generator, you must first "select" it, otherwise, there is no way to direct your commands to the pattern generator.

To select the module, use the system command :SELect , followed by the numeric reference for the slot location of the pattern generator(1...5 refers to slot A...E respectively). For example, if the pattern generator master card is in slot E, then the command:

```
:SELect 5
```

would select this module. For more information on the select command, refer to the *HP 16500A Programming Reference Manual*.

Programming the Pattern Generator

A typical pattern generator program includes the following tasks:

- select the appropriate module
- set program parameters
- define a pattern generator program
- run the pattern generator program

The following example program generates a pattern using the two output pods of the master card:

```
10 OUTPUT XXX;":SELECT 1"
20 OUTPUT XXX;":FORMAT:REMOVE ALL"
30 OUTPUT XXX;":FORMAT:LABEL 1 'A',POSITIVE,127,0"
40 OUTPUT XXX;":FORMAT:LABEL 'B',POSITIVE,0,255"
50 OUTPUT XXX;":LIST:REMOVE ALL"
60 OUTPUT XXX;":LIST:PROG 1,NOOP,'#H7F','#HFF'"
70 OUTPUT XXX;":RMODE REPETITIVE"
80 OUTPUT XXX;":START"
90 END
```

Note

The three Xs (XXX) after the OUTPUT statement in the above example refer to the device address required for programming over either HP-IB or RS-232-C. Refer to your controller manual and programming language reference manual for information on initializing the interface.

Program Comments

Line 10 selects the pattern generator in slot A
Line 20 removes all labels previously assigned
Line 30 assigns label 'A', the output polarity and defines active channels of pod A3
Line 40 assigns label 'B' and defines active channels of pod A2
Line 50 removes all program lines
Line 60 lists the first line of the pattern generation program. Channel data may be specified in binary, octal, decimal, hexadecimal.
Line 70 Sets the RMODE to repetitive. If the program is to be run only once, select the :RMODE SINGLE command.
Line 80 Starts the program.

For more information on the specific pattern generator commands, refer to chapters two through six of this manual.

Mainframe Commands

These commands are part of the HP 16500A mainframe system and are mentioned here only for reference. For more information on these commands, refer to the *HP 16500A Programming Reference* manual.

CARDcage? Query

The CARDcage query returns a string which identifies the modules that are installed in the mainframe. The returned string is in two parts. The first five two-digit numbers identify the card type. The identification number for the HP 16520A pattern generator is 21 and the HP 16521A identification number is 22. A "-1" in the first part of the string indicates no card is installed in the slot.

The five single-digit numbers in the second part of the string indicate in which slots cards are installed and where the master card is located.

Example: `11,12,-1,-1,31,1,1,0,0,5`

A returned string of 11,12,-1,-1,31,1,1,0,0,5 means that an oscilloscope timebase card (ID number 11) is loaded in slot A and the oscilloscope acquisition card (ID number 12) is loaded in slot B. The next two slots (C and D) are empty (-1). Slot E contains a logic analyzer module (ID number 31).

The next group of numbers (1,1,0,0,5) indicate that a two card module is installed in slots A and B with the master card in slot A. The "0" indicates an empty slot or the module software is not recognized or not loaded. The last digit (5) in this group indicates a single module card is loaded in slot E. Complete information for the CARDcage query is in the *HP 16500A Programming Reference* manual.

MENU Command/query

The MENU command selects a new displayed menu. The first parameter (X) specifies the desired module. The optional second parameter specifies the desired menu in the module (defaults to 0 if not specified). The query returns the currently selected (and displayed) menu.

For the HP 16520A/HP 16521A Pattern generator:

- X,0 - Format Menu
- X,1 - Listing Menu
- X,2 - Macro 1 Menu
- X,3 - Macro 2 Menu
- X,4 - Macro 3 Menu
- X,5 - Macro 4 Menu

X = slot number that contains the pattern generator master card.

SELect Command/query

The SELect command selects which module or intermodule will have parser control. SELect 0 selects the intermodule, SELect 1 through 5 selects modules A through E respectively. Parameters -1 and -2 select software options 1 and 2. The SELect query returns the currently selected module.

STARt Command

The STARt command starts the specified module or intermodule. If the specified module is configured for intermodule, STARt will start all modules configured for intermodule.

STOP Command

The STOP command stops the specified module or intermodule. If the specified module is configured for intermodule, STOP will stop all modules configured for intermodule.

RMODe Command/query

The RMODe command specifies the run mode (single or repetitive) for a module or intermodule. If the selected module is configured for intermodule, the intermodule run mode will be set by this command. The RMODe query returns the current setting.

SYSTem:ERRor? Query The SYSTem:ERRor query returns the oldest error in the error queue. In order to return all the errors in the error queue, a simple FOR/NEXT loop can be written to query the queue until all errors are returned. Once all errors are returned, the queue will return zeros.

SYSTem:PRINt Command/query The SYSTem:PRINt command initiates a print of the screen or listing buffer over the current printer communication interface. The SYSTem:PRINt query sends the screen or listing buffer data over the current controller communication interface.

MMEMory Subsystem The MMEMory Subsystem provides access to both internal disc drives for loading and storing configurations.

INTermodule Subsystem The INTermodule Subsystem commands are used to specify intermodule arming between multiple modules.

Command Set Organization

The command set for the HP 16520A is divided into four separate subsystems. The subsystems are: FORMat, LISTing, MACRo, and the SYMBol subsystem. Each of the subsystems commands are covered in their individual chapters starting with chapter 2.

Each of these chapters contains a description of the subsystem, syntax diagrams and the commands in alphabetical order. The commands are shown in longform and shortform using upper and lowercase letters. For example, FORMat indicates that the longform of the command is FORMAT and the shortform is FORM. Each of the commands contain a description of the command and its arguments, the command syntax, and a programming example.

Figure 1-1 is the command tree for the HP 16520A pattern generator module.

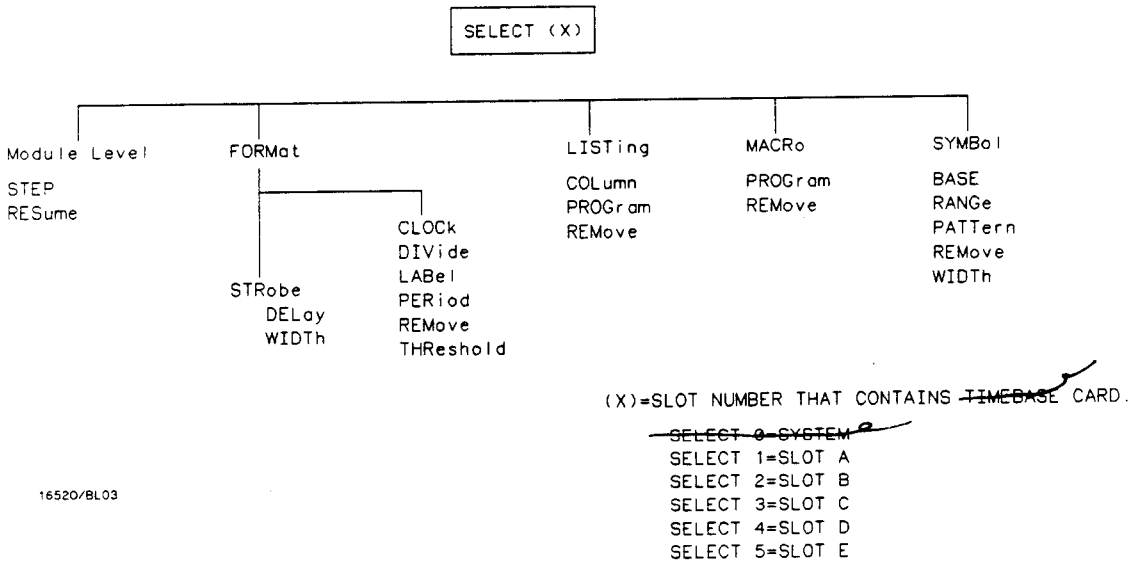


Figure 1-1. HP 16520A/HP 16521A Command Tree

Table 1-1. Alphabetical Command to Subsystem Directory.

COMMAND	WHERE USED
BASE	SYMBol
CLOCk	FORMat
COLumn	LISTing
DELay	STRobe
DIVide	FORMat
LABel	FORMat
PATTern	SYMBol
PERiod	FORMat
PROGram	LISTing
	MACRo
RANGe	SYMBol
REMove	FORMat
	LISTing
	MACRo
	SYMBol
RESume	Module Level
STEP	Module Level
THReshold	STRobe
WIDth	STRobe
	SYMBol

16520/BL02

Module Status Reporting

Each module reports its status to the Module Event Status Register (MESR<N>) which in turn reports to the Combined Event Status Register (CESR) in the HP 16500A mainframe (see *HP 16500A Programming Reference* manual). The Module Event Status Register is enabled by the Module Event Status Enable Register (MESE<N>).

The following descriptions of the MESE<N> and MESR<N> commands provide the module specific information needed to enable and interpret the contents of the registers

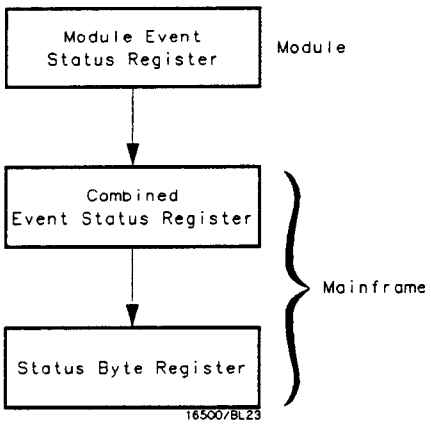


Figure 1-2. Module Status Reporting

MESE<N> command/query

The MESE<N> command sets the Module Event Status Enable register bits. The MESE register contains a mask value for the bits enabled in the MESR register. A one in the MESE will enable the corresponding bit in the MESR register; a zero will disable the bit.

The first parameter after the command specifies the module (<N> = 1...5 refers to the module in slot A...E). The second parameter specifies the enable value.

The MESE query returns the current setting.

Refer to table 1-2 for information about the Module Event Status register bits, bit weights, and what each bit masks in the module. Complete information for status reporting is in the *HP 16520A Programming Reference* manual.

Command Syntax: :MESE<N> <enable_mask>

where:

<N> ::= {1|2|3|4|5} number of slot in which the module resides
<enable_mask> ::= integer 0 to 255

Example: OUTPUT XXX;":MESE5 2"

Query Syntax: :MESE<N>?

Returned Format: [MESE] <enable_mask><NL>

Example:
```
10 OUTPUT XXX;":MESE2?"
20 ENTER XXX; Mes
30 PRINT Mes
40 END
```

Table 1-2. Module Event Status Enable Register

Module Event Status Enable Register (A "1" enables the MESR bit)		
BIT	WEIGHT	ENABLES
7	128	NOT USED
6	64	NOT USED
5	32	NOT USED
4	16	NOT USED
3	8	NOT USED
2	4	NOT USED
1	2	NOT USED
0	1	RUN COMPLETE

16520/BL04

The Module Event Status Enable Register contains a mask value for the bits to be enabled in the Module Event Status Register (MESR). A one in the MESE enables the corresponding bit in the MESR, a zero disables the bit.

MESR<N> query

The MESR<N> query returns the contents of the Module Event Status register.

Note

Reading the register clears the Module Event Status Register.

Table 1-3 shows each bit in the Module Event Status Register and their bit weights for this module. When you read the MESR, the value returned is the total bit weights of all bits that are high at the time the register is read.

The parameter 1...5 refers to the module in slot A...E respectively.

Query Syntax: :MESR<N>?

Returned Format: [MESR]<status><NL>

where:

<N> ::= {1|2|3|4|5} number of slot in which the module resides
<status> ::= 0 to 255

Example:
```
10 OUTPUT XXX;":MESR2?"
20 ENTER XXX; Mer
30 PRINT Mer
40 END
```

Table 1-3. Module Event Status Register

Module Event Status Register		
BIT	WEIGHT	CONDITION
7	128	NOT USED
6	64	NOT USED
5	32	NOT USED
4	16	NOT USED
3	8	NOT USED
2	4	NOT USED
1	2	NOT USED
0	1	1=RUN COMPLETE 0=RUN NOT COMPLETE

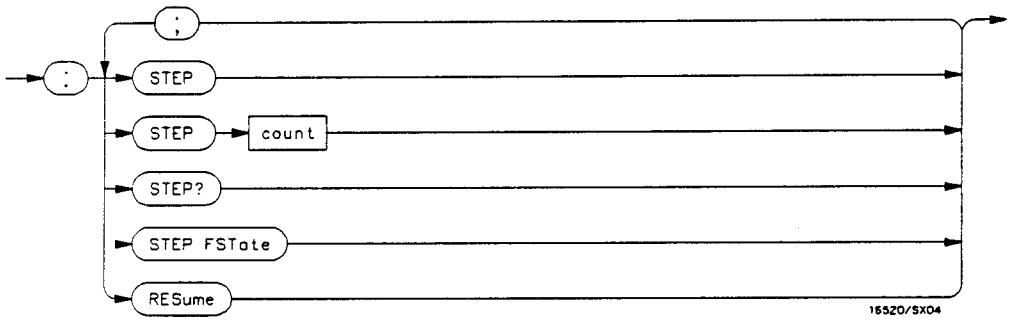
16520/BL05

Note

The MESR bit will be set at the end of the program or if a BREAK instruction is encountered within the program.

Module Level Commands 2

The Module Level Commands control the operation of pattern generator programs. The two Module Level Commands are STEP and RESume. Refer to figure 2-1 for the Module Level Syntax Diagram.



count = *an integer from 1 to 999, specifying number of program lines*

Figure 2-1. Module Level Syntax Diagram

STEP

Command/Query

The STEP command consists of four parts: the STEP command, the STEP Count command, the STEP query , and the STEP FSTate command .

The STEP command causes the pattern generator to step through the number of program lines specified by the STEP Count command. The valid program line range for the STEP Count command is from 1 to 999. The STEP count query returns the current count.

The STEP FSTate (step first state) command allows you to return to program line 0 of the current program.

Command Syntax:

For the STEP command :STEP

Example: OUTPUT XXX;":STEP"

Command Syntax:

For STEP Count command STEP <count>

where:

<count> ::= an integer from 1 to 999, specifying the number of program lines.

Example: OUTPUT XXX;":STEP 20"

Query Syntax: :STEP?

Returned Format: [STEP] <count>

Example:
```
10 DIM Sc$[100]
20 OUTPUT XXX;":STEP?"
30 ENTER XXX;Sc$
40 PRINT Sc$
50 END
```

Command Syntax:

For STEP FSTate command: :STEP FSTate

Example: OUTPUT XXX;":STEP FSTATE"

RESume Command

When the pattern generator encounters a BREAK instruction, program execution is halted. The RESume command allows the program to continue until another BREAK instruction is encountered, or until the end of the program is reached.

Command Syntax: :RESume

Example: OUTPUT XXX;":RESUME"

FORMat Subsystem 3

Introduction

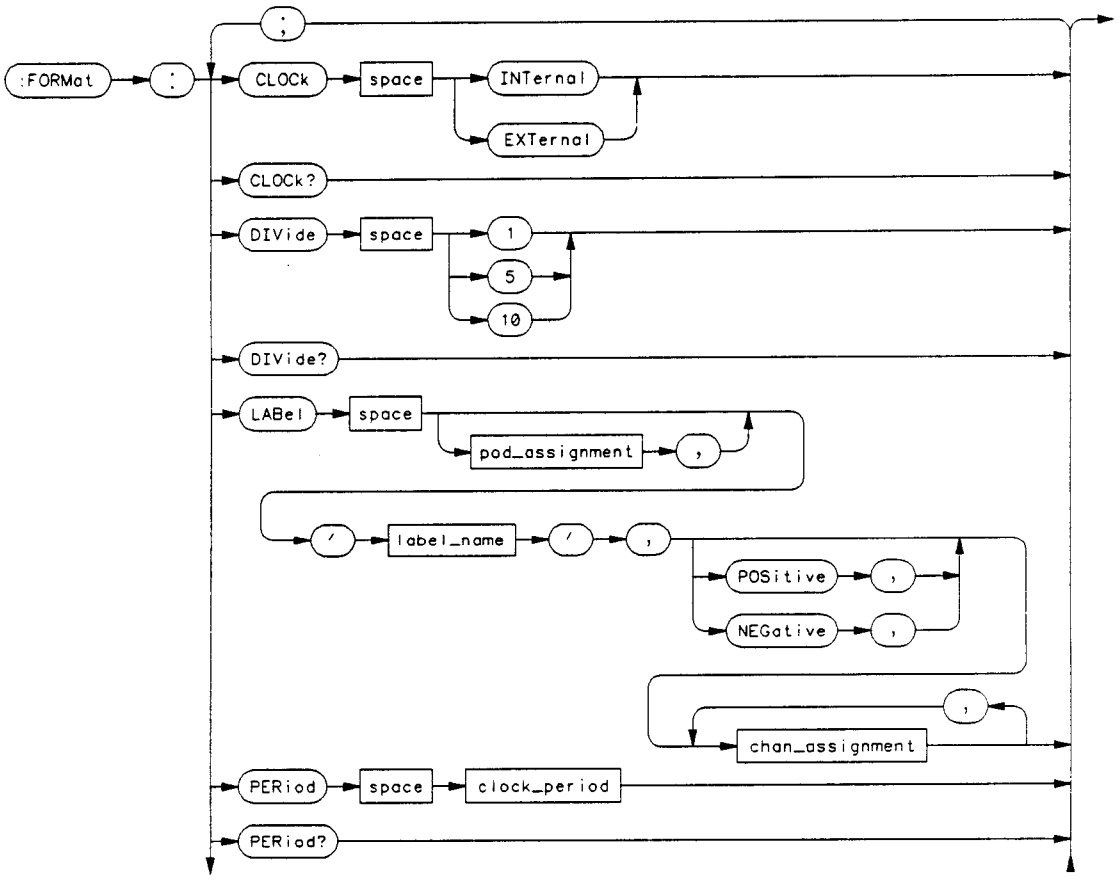
The commands of the Format subsystem control the pattern generator values such as data output rate, strobe width and delay, and the channels that you want to be active. The Format subsystem also lets you specify the clock source and allows you to group channels together under a common, user - defined name. Refer to Figure 3-1 for the Format subsystem syntax diagrams.

Strobe Sublevel Set

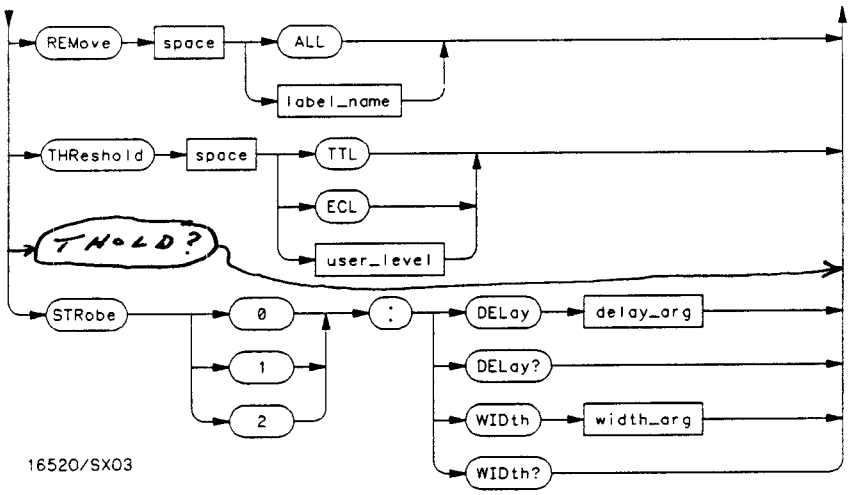
The commands of the Strobe sublevel are part of the Format subsystem and are used to set the strobe delay and strobe width.

Each pattern generator master card has three strobe outputs. The strobe outputs are data channels with selectable pulse width and pulse delay. While standard data channels can change state only at the start of an output clock cycle, strobes can change state after the clock transition or can change state in the middle of a clock cycle.

If the polarity of a strobe channel needs to be changed, use the LABEL command. To specify the polarity of strobe channels individually, rename each strobe under a different label.



P/O Figure 3-1. Format Subsystem Syntax Diagram



label name = *a string of up to 6 alphanumeric characters*
chan_assignment = *an integer from 0 to 255*
clock period = *a real number from 20 ns through 200 us*
user_level = *a voltage level from -9.9 V to + 9.9 V*
delay_arg = *a real number specifying strobe delay time*
width_arg = *a real number specifying strobe width*

P/O Figure 3-1. Format Subsystem Syntax Diagram

CLOCk

Command/Query

The CLOCk command specifies the clock source for the pattern generator. The choices are INTernal or EXTernal. The clock specified by this command is the output data clock. Each time a new clock period starts, the pattern generator outputs go to their next state, as defined by the program listing. The internal clock pulse period may be varied using the PERIOD command. The maximum external clock rate is 50 MHz. The query returns the current clock choice.

Command Syntax: :FORMat:CLOCk {INTernal|EXTernal}

Example: OUTPUT XXX;":FORMat:CLOCK INTERNAL"

Query Syntax: :FORMat:CLOCk?

Returned Format: [:FORMat:CLOCk] {INTernal|EXTernal}<NL>

Example:
```
10 DIM CI$[100]
20 OUTPUT XXX;":FORMAT:CLOCK?"
30 ENTER XXX;CI$
40 PRINT CI$
50 END
```

DIVide — Command/Query

The DIVide command allows you to divide the external clock frequency by 1, 5 or 10. When divide by 1 is chosen, the output strobes are not available. The divide by 5 and divide by 10 division parameters determine the resolution by which the strobe width and delay parameters can be set. In divide by 5, the width and delay may be set in 1/5 increments of the clock period. In divide by 10, the width and delay may be set in 1/10 increments of the clock period. The query returns the current division ratio.

Command Syntax: :FORMat:DIVide <divide by ratio>

where

<divide by value> ::= 1, 5, or 10

Example:
```
OUTPUT XXX;":FORM:DIV 1"
```

Query Syntax: :FORMat:DIVide?

Returned Format: [FORMat:DIVide] <divide by ratio><NL>

Example:
```
10 DIM Di$[100]
20 OUTPUT XXX;":FORMAT:DIVIDE?"
30 ENTER XXX;Di$
40 PRINT Di$
50 END
```

LABel

Command/Query

The LABel command inserts a new label or modifies the contents of an existing label. If more than 20 labels are specified, and an attempt is made to insert another new label, the last label (bottom label) will be modified.

Stimulus channels can be assigned to only one label at a time. If duplicate assignments are made, the last channel assignments take precedence.

The first parameter is optional and is used to specify the first pod that is to have channels assigned. If the first parameter choice is not made, then the STROBE/DATA pod of the master card is assumed. The pods are numbered in the same order as they appear in the format menu, with zero representing the STROBE/DATA pod of the master card. The second parameter sets the channel polarity. If the polarity is not specified, the last polarity assignment is used. The last parameters assign the active channels for each pod.

Each assignment parameter is a binary encoding of the channel assignments of the pod. A "1" in a bit position means that the associated channel in that pod is included in the label. A "0" in a bit position excludes the channel from the label. The minimum value for any pod specification is 0, the maximum value for all pods except the STROBE/DATA pod is 255. The maximum value for the STROBE/DATA pod is 127. A value of 255 includes all channels of a pod assignment. The query must specify a label name and returns the current pod assignments and channel polarity for that label.

Command Syntax: :FORMat:LABel [<pod assignment>,]<label name>[<polarity>,]<channel assignment>, ... <channel assignment>

where:

<pod assignment> ::= an integer from 0 to 26, depending on how many expansion cards are used.
<label name> ::= string of up to 6 alphanumeric characters
< polarity> ::= polarity of the channel outputs,NEGative or POSitive

<channel assignment> ::= a string in one of the following forms:
'B01...' for binary
'#Q01234567..'for octal
'#H0123456789ABCDEF... for hexadecimal
'0123456789...' for decimal.

Example: OUTPUT XXX;":FORMAT:LABEL 1,'A',POSITIVE,255,0"

Query Syntax: :FORMat:LABel? <label name>

Returned Format: [:FORMat:LABel] <label name>,<polarity>,<channel assignment>, ... <channel assignment><NL>

Example:
```
10 DIM La$[100]
20 OUTPUT XXX;":FORMAT:LABEL? 'A'"
30 ENTER XXX;La$
40 PRINT La$
50 END
```

PERiod — Command/Query

The PERiod command specifies the internal clock period. The range limits are from 20 ns to 200 us in a 1, 2, 5 sequence. The query returns the current clock period.

Command Syntax: :FORMat:PERiod <clock period>

where:

<clock period> ::= a real number from 20 ns to 200 us, in a 1, 2, 5 sequence

Example:
```
OUTPUT XXX;":FORMAT:PERIOD 1.0E-6"
```

Query Syntax: :FORMat:PERiod?

Returned Format: [:FORMAT:PERIOD] <clock period> <NL>

Example:
```
10 DIM Cp$[100]
20 OUTPUT XXX;":FORMAT:PERIOD?"
30 ENTER XXX;Cp$
40 PRINT Cp$
50 END
```

REMove Command

The REMove is used to delete a single label, or all labels from the format menu. If a label name is specified, it must match a label name currently active in the format menu.

Command Syntax: :FORMat:REMove {ALL| <label name>}

Example: OUTPUT XXX;":FORMAT:REMOVE ALL"

THReshold

Command/Query

The THReshold command sets the input threshold levels for the pattern generator input pod. The selection may be TTL, ECL or User Defined. The user defined input may range from -9.9 V to +9.9 V. The query returns the current setting.

Command Syntax: :FORmat:THReshold {TTL|ECL|<value>}

where:

<value> ::= voltage (real number) -9.9 to +9.9

Example OUTPUT XXX;":FORMAT:THRESHOLD 5.2V"

Query Syntax: :FORMAT:THRESHOLD?

Returned Format: [FORMAT:THRESHOLD] <value>

Example:

```
10 DIM Th$[100]
20 OUTPUT XXX;":FORMAT:THRESHOLD?"
30 ENTER XXX;Th$
40 PRINT Th$
50 END
```

STRobe Selector

The STRobe Selector is used as part of a compound header to set the output strobe parameters.

It always follows the FORMAT selector because it selects a branch below the Format level in the command tree. When setting strobe parameters, the strobe number must always be specified (strobe 0 through strobe 2).

Command Syntax: :FORMat:STRobe<strobe number>:<strobe parameter>

where:

<strobe number> ::= strobe number 0, 1, or 2
<strobe parameter> ::= strobe parameter may either be DELay or WIDTh command

Example: OUTPUT XXX;":FORMAT:STROBE1:DELAY 10E-9"

DELay

Command/Query

The DELay command sets the strobe delay value. The delay value set is with respect to the rising edge of the output clock. In other words, the delay value tells the pattern generator to delay the start of the strobe from the rising edge of the output clock. The output strobes are not available at the clock period of 20 ns or external clock divide by 1. Strobe delay time and strobe width are related to the clock period. The delay time plus width can not exceed the one clock period. The delay parameters may be set as shown in Table 3-1. The query reports the current delay setting.

INTERNAL CLOCK PERIOD	EXTERNAL CLOCK ÷	DELAY PARAMETER SETTING (MAXIMUM RESOLUTION)
20ns	÷1	OUTPUT STROBES ARE NOT AVAILABLE
50ns	÷5	DELAY may be set in 1/5 increments of output clock period.
100ns to 200µs	÷10	DELAY may be set in 1/10 increments of output clock period.

16520/BL10

Table 3-1. Strobe DELAY Parameter Setting

Command Syntax: :FORMat:STRobe < strobe number > :DELay < delay value >

where:

< strobe number > ::= strobe number 0, 1, or 2

< delay value > ::= a real number from 0 to the current output data rate if internally clocked
or
an integer between 0 and 9 if output is externally clocked

Example: OUTPUT XXX;":FORMAT:STROBE1:DELAY 1E-6"

Query Syntax: FORMat:STRobe < strobe number > :DELay?

Returned Format: [FORMAT:STROBE1 < strobe number > :DELAY] < delay value > < NL >

Example:
```
10 DIM Sd$[100]
20 OUTPUT XXX;":FORMAT:STROBE1:DELAY?"
30 ENTER XXX;Sd$
40 PRINT Sd$
50 END
```

WIDTh

Command/Query

The WIDTh command sets the strobe width value. The width parameter are set in the same manner as the strobe delay parameters. Refer to Table 3-2 for an explanation of the strobe width parameter settings. The query returns the current width setting.

INTERNAL CLOCK PERIOD	EXTERNAL CLOCK ÷	WIDTH PARAMETER SETTING (MAXIMUM RESOLUTION)
20ns	÷1	OUTPUT STROBES ARE NOT AVAILABLE
50ns	÷5	WIDTH may be set in 1/5 increments of clock period.
100ns to 200µs	÷10	WIDTH may be set in 1/10 increments of output clock period.

16520/BL09

Table 3-2. Strobe WIDTH Parameter Settings

Command Syntax: :FORMat:STRobe <strobe number>:WIDTh <width value>

where:

<strobe number> ::= strobe number 0, 1, 2

<width value> ::= a real number between 0 and the output data rate if internal clock is used
or
an integer between 0 and 10 if external clock is used

Example: OUTPUT XXX;":FORMAT:STROBE0:WIDTH 500ns"

Query Syntax: :FORMat:STRobe:WIDTh?

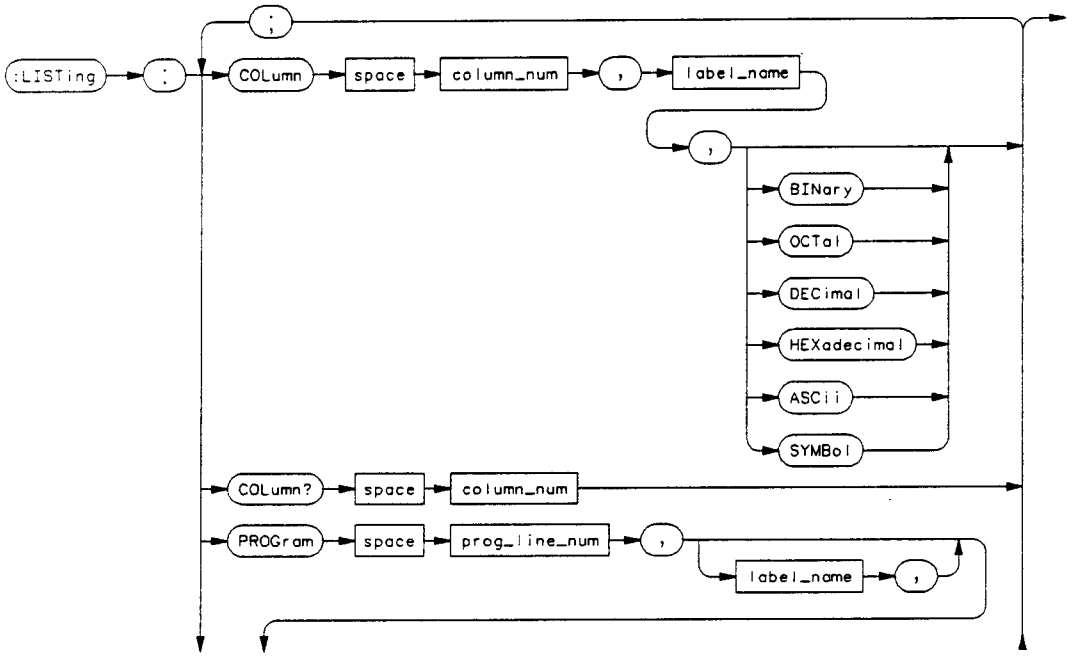
Returned Format: [FORMAT:STROBE<strobe number>:WIDTh]<width value><NL>

Example:
```
10 DIM Wv$[100]
20 OUTPUT XXX;":FORMAT:STROBE1:WIDTH 500ns"
30 ENTER XXX;Wv$
40 PRINT Wv$
50 END
```

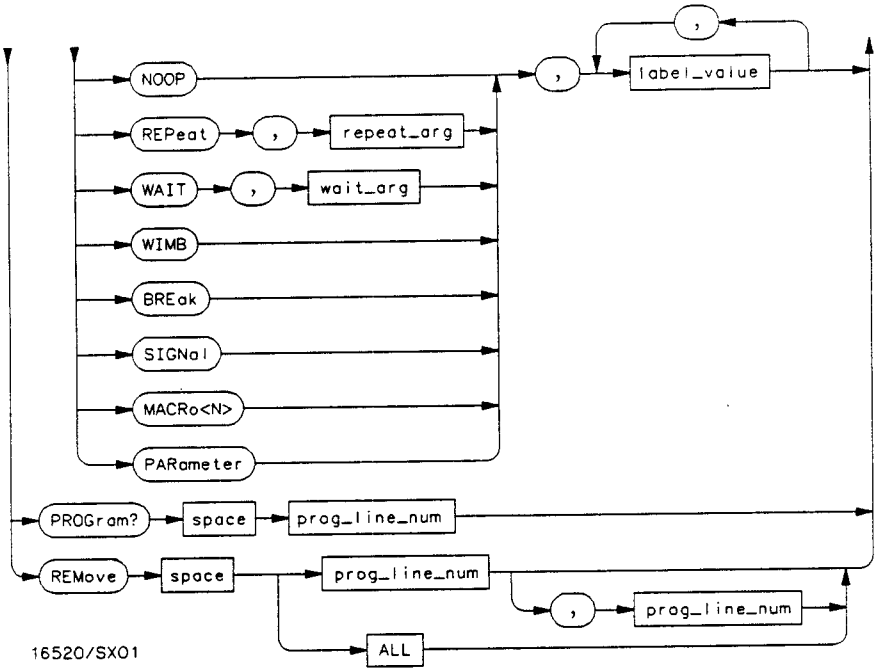
LISTing Subsystem 4

Introduction

The commands of the Listing subsystem allow you to write a pattern generator program using the parameters set in the Format subsystem.



P/O Figure 4-1. LISTing Subsystem Syntax Diagram



column_num = *an integer specifying the column that is to receive the* new *label*
label_name = *the label name that is to be removed*
prog_line_num = *an integer specifying the program line number*
label_value = *a string in one of the following forms:*
'#B01...' for binary
'Q01234567...' for octal
'H0123456789ABCDEF...' for hexadecimal
'0123456789...' for decimal
repeat_arg = *an integer from 1 through 256*
wait_arg = *an integer from 0 through 255*

P/O Figure 4-1. LISTing Subsystem Syntax Diagram

COLumn command/Query

The COLumn command allows you to reorder the labels in the listing menu and set the numerical base for each label. The order of the labels in the format menu is not changed when the COLUMN command is used.

The first parameter of the command specifies the column number, followed by a label name and an optional number base. If a number base is not specified, the current number base for the label is used.

The query must include a column number and returns the label in that column and its base.

Command Syntax :LISTing:COLumn <column number>,'<label name>'[,{BINary|OCTal|DECimal|HEXadecimal|ASCii|SYMBol}]

where:

<column number> ::= an integer specifying the column that is to receive the new label
<label name> ::= the label name that is to be moved

Note

To move the Instruct column, use INSTRUCT as the label name without quotation marks.

Example: OUTPUT XXX;":LIST:COL 1,'A',HEX"

COLumn

Query Syntax; :LISTing:COLumn? <column number>

Returned Format: [LISTING:COLUMN] <column number>,<label name>,
{BINary|OCTal|DECimal|HEXadecimal|ASCII|SYMBol}

Example:
```
10 DIM Co$[100]
20 OUTPUT XXX;":LIST:COL? 1"
30 ENTER XXX;Co$
40 PRINT Co$
50 END
```

PROGram

Command/Query

The PROGram command adds pattern generator program lines, or modifies an existing line. The first parameter is the program line number. If the line number specified is beyond the last program line currently entered, a new line is added to the program. If the line number reference is a line within the current program, the existing program line is modified. The valid range of line numbers is 0 to 4094.

The labels are programmed in the same order as they are specified in the format menu regardless of their order in the listing.

If macros are invoked in the main program, the PROGram command line numbers may not correspond with the line numbers shown on the listing menu. This is because the macro program is inserted in the main program list. The PROGRAM command however, compensates for this and allows contiguous line numbering.

The second parameter is an optional string parameter. It specifies the starting label for the pattern strings that follow. This parameter is useful when long program strings are to be separated into several commands.

The next parameter may be one of five instructions or a call to one of four user defined macros. The instructions that may be used in a program are: NOOP, REPeat, WAIT, WIMB (Wait IMB), BREak, SIGNal, and MACRO#.

NOOP The NOOP instruction places a no instruction into the program line.

REPeat The REPeat instruction allows you to repeat a program line up to 256 times.

WAIT Along with an external clock, there are three external input qualifiers available with each master card. The WAIT instruction causes the pattern generator to wait at the current program line until the three external inputs go to a predefined state. When the predefined state is met, the

program proceeds to the next program line.

When the wait parameter is represented in binary, each bit determines whether the associated state on the external inputs will be included or excluded from the wait condition. If all the wait bits are 1's, the pattern generator output is stopped, while all 0's allow the pattern generator to continue to the end of the program. A wait parameter of 01010101 is used in the example of Figure 4-2).

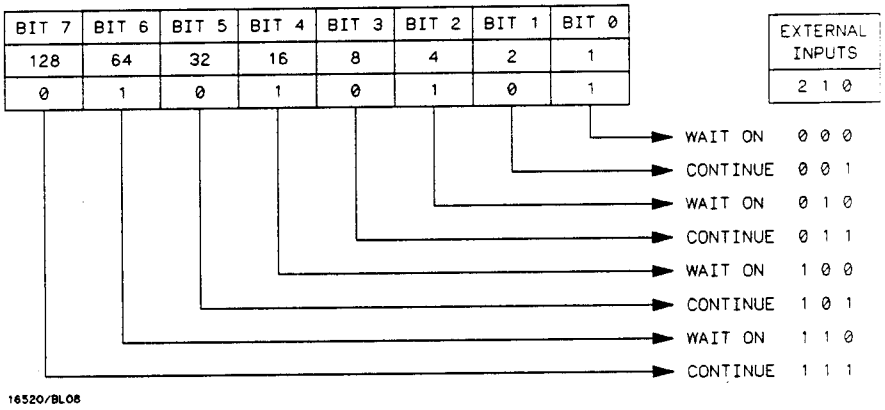


Figure 4-2. WAIT Condition Example

WIMB (Wait IMB) Any module in the HP 16500A can signal the other modules through the Intermodule Bus (IMB).

If the pattern generator encounters a WIMB instruction in the program, it will hold the data outputs at their current state, while the output data clock and the strobes continue to run. The pattern generator will not continue to the next program line until it sees a signal on the IMB. In other words, the pattern generator will wait until another module tells it to continue.

BREak When a BREak is encountered in a program line, the pattern generator will stop. To advance to the next program line, use the RESume command.

SIGNal (Signal IMB) The SIGNal instruction is the complement of the WIMB instruction. When the pattern generator program encounters a SIGNal instruction, it will output a signal to the Intermodule Bus (IMB). This signal is used to trigger other modules that are linked through the IMB, or the Port Out BNC.

MACRO<N> The macro instruction field lets you call one of four macros into a program. The macro programs are written in a similar manner as the main pattern generator programs. The MACRO subsystem section of this manual explains how to generate macros.

The program query returns the content of a program line and must include a program line number.

Command Syntax: :LISTing:PROGram <program line number>,[<'label name',]{NOOP|REPeat,<repeat arg>,|WAIT,<wait arg>,|WIMB|BREak|SIGNal|MACRO#|PARameter}<'label value'>[,<'label value>.....]

where:

<program line number> ::= an integer specifying the program line number
<label name> ::= string of up to six alphanumeric characters
<repeat arg> ::= an integer from 1 through 256
<wait arg> ::= an integer from 0 through 255
<label value> ::= a string in one of the following forms:
'#B01...' for binary
'#Q01234567...'for octal
'#H0123456789ABCDEF...'for hex
'0123456789...'for decimal

Examples:
```
OUTPUT XXX;":LIST:PROG 0,REPEAT,255,'#B01X10111'"
OUTPUT XXX;":LIST:PROG 1,NOOP,'0'"
OUTPUT XXX;":LIST:PROG 2,SIGNAL,'1234'"
OUTPUT XXX;":LIST:PROG 3,WAIT,#B01010101,'#H2XBC'"
OUTPUT XXX;":LIST:PROG 4,MACRO2,'#H3X45'"
OUTPUT XXX:":LIST:PROG 5,PARAMETER,'#B0101111100001111'"
```

Query Syntax: :LISTing:PROGram? <program line number>

Returned Format: [LISTING:PROGRAM] <program line number>,{NOOP|REPeat<repeat arg>|WAIT<wait arg>|WIMB|BREak|SIGNal|MACRO<N>|PARameter},<label value>[,<label value>.....]

Example:
```
10 DIM A$[100]
20 OUTPUT XXX;":LIST:PROG? 1"
30 ENTER XXX;A$
40 PRINT A$
50 END
```

REMove Command

The REMove command allows you to remove one or several lines from the main pattern generator program. If only one parameter number is given, that line number is deleted. If two numbers are given, the range of lines between those two values inclusive is deleted. The command REMove ALL deletes the entire program.

Command Syntax: LISTing:REMove{ <program line number[, <program line range> |ALL>}

where:

<program line number> ::= an integer specifying the program line to be removed

<program line range> ::= two integers separated by a comma, specifying the program line range to be removed.

Example: OUTPUT XXX;":LIST:REM 1,4"

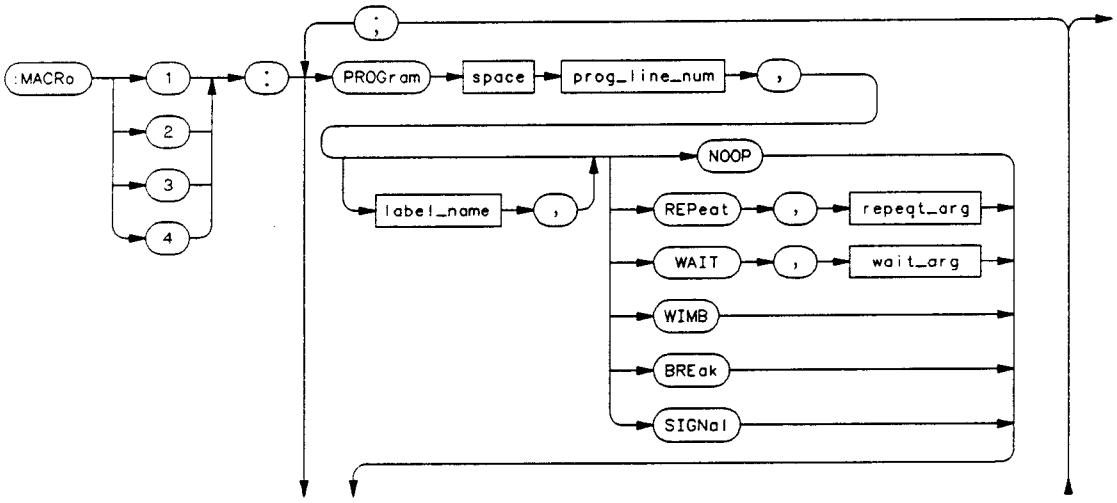
MACRo Subsystem 5

Introduction

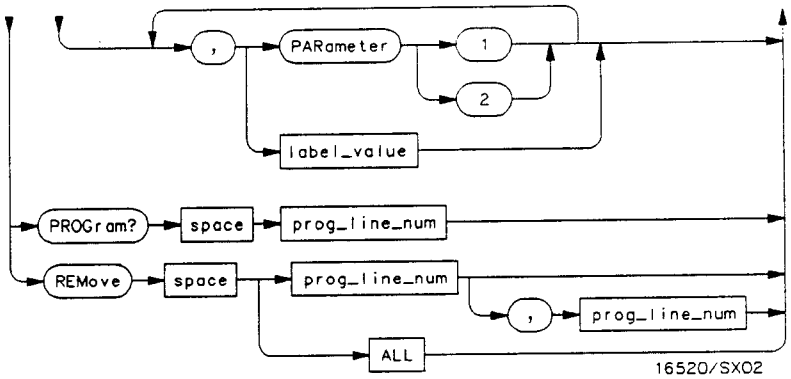
The commands of the MACRo subsystem allow you to write and edit macros for use in the main pattern generator program. Up to four macros may be called into the main listing program. The macros are labeled MACRO1 through MACRO4 and cannot be renamed over the interface bus.

The query returns the content of a program line and must include a program line number.

Refer to figure 5-1 for the MACRo subsystem syntax diagram.



PO/ Figure 5-1. MACRo Subsystem Syntax Diagram



prog_line_num = *an integer specifying the program line number*
repeat_arg = *an integer from 1 through 256*
wait_arg = *an integer from 0 through 255*
label_value = *a string in one of the following forms:*
'#B01...' for binary
'#Q01234567...' for octal
'#H012345679ABCDEF...' for hexadecimal
'0123456789...' for decimal

P/O Figure 5-1. MACRo Subsystem Syntax Diagram

PROGram

Command/Query

The PROGram command adds macro program lines, or modifies an existing line. This command is identical to the LISTing:PROGram command, with two exceptions:

- MACRo and PARameter are not included as choices for the instruction parameter because a macro cannot be invoked from another macro.
- The pattern generator allows you to pass parameters between the main listing program and the macros using the PARAM1 and PARAM2 key words. These key words may be substituted for any label value string

Setting Pass Parameters

There are two parameters available for each label in the macro list. They are labeled PARAM1 and PARAM2. In the example of figure 5-2, a macro call is made at line three of the main listing program.

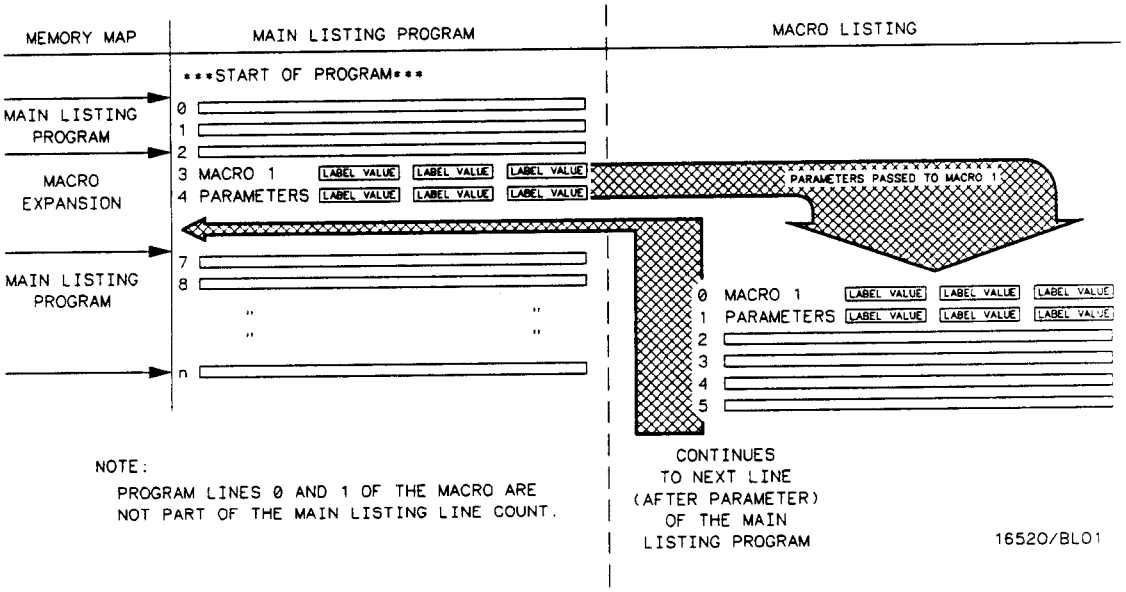


Figure 5-2. Setting Pass Parameters

The data of lines three and four are passed into the macro lines zero and one and are labeled PARAM1 and PARAM2. At the last line of the macro, the program continues to the next line in the main listing program. The main listing program line numbering is not consecutive. This is because the macro is placed in memory at the location of the macro command. Also note that lines zero and one of the macro are not part of the main listing line count.

Command Syntax: :MACRo<macro number>:PROGram <program line number>,[<label_ name>,]
{NOOP|REPeat,<repeat arg>|WAIT,<wait arg>|WIMB|BREak|SIGNal|},
{PARameter<1|2>|<label value>}[,PARameter<1|2>|<label value>....]

where:

<macro number> ::= an integer from 1 through 4
<program line number> ::= an integer specifying the program line number
<label_name> ::= string of up to six alphanumeric characters
<repeat arg> ::= an integer from 1 through 256
<wait arg> ::= an integer from 0 through 255
<label value> ::= a string in one of the following forms:
'#B01...' for binary
'#Q01234567...' for octal
'#H123456789ABCDEF...' for hexadecimal
'0123456789...' for decimal

Examples:
```
OUTPUT XXX;":MACRo1:PROG 0,NOOP,'#B010110010','#B000100101'"
OUTPUT XXX;":MACRO1:PROG 1,REPEAT,127,PAR1,PAR2"
OUTPUT XXX;":MACRO1:PROG 2,NOOP,'#B01X10X10',PAR2"
```

Query Syntax: :MACRO<macro number>:PROGram? <program line number>

Returned Format: [:MACRo<macro number>:PROGram <program line number>],{NOOP|REPeat,<repeat arg>|WAIT,<repeat arg>|WIMB|BREak|SIGNal}|{PARameter<1|2>|<label value>}[,PARameter<1|2>|<label value>....]

Example:
```
10 DIM A$[100]
20 OUTPUT XXX;":MACRO1:PROGRAM? 1"
30 ENTER XXX;A$
40 PRINT A$
50 END
```

REMove Command

The REMove allows you to remove one or several lines from the macro. If only one parameter is given, only that line is deleted. If two numbers are specified, the range of lines between those values, inclusive, is deleted. The command REMove ALL deletes the entire program.

Command Syntax: :MACRo<macro number>:REMove <program line number>[,<program line range>|ALL

where:

<macro number>	::= an integer from 1 through 4
<program line>	::= an integer specifying the program line to be removed
<program line range>	::= two integers separated by a comma, specifying the program lines to be removed

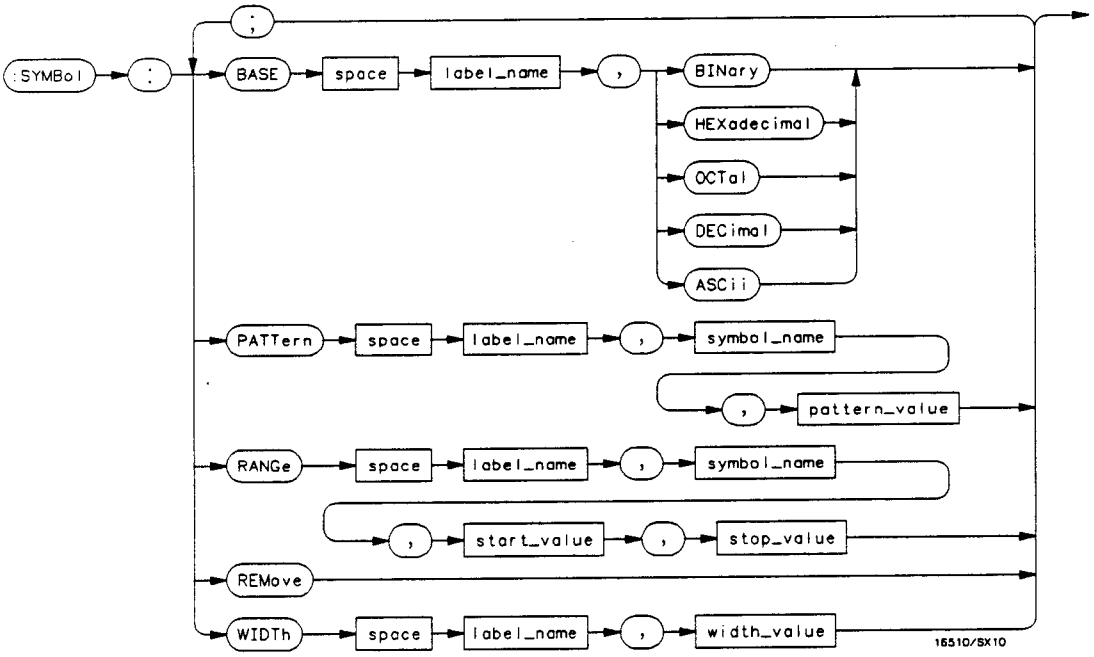
Example: OUTPUT XXX;":MACRO1:REM 1,3"

SYMBol Subsystem 6

Introduction

The SYMBol subsystem contains the commands that allow you to define symbols on the controller and download them to the HP 16520A/HP 16521A Pattern Generator module. The commands in this subsystem are:

- BASE
- PATTern
- RANGe
- REMove
- WIDTh



P/O Figure 6-1. SYMBol Subsystem Syntax Diagram

<label_name> = *string of up to 6 alphanumeric characters*
<symbol_name> = *string of up to 16 alphanumeric characters*
<pattern_value> = *string of one of the following forms:*
'#B01X...' for binary
'#Q01234567X..' for octal
'#H0123456789ABCDEFX...' for hexadecimal
'0123456789...' for decimal
<start_value> = *string of one of the following forms:*
'#B01...' for binary
'#Q01234567..' for octal
'#H0123456789ABCDEF...' for hexadecimal
'0123456789...' for decimal
<stop_value> = *string of one of the following forms:*
'#B01... for binary
'#Q01234567..' for octal
"#H0123456789ABCDEF..." for hexadecimal
'0123456789...' for decimal
<width_value> = *integer from 1 to 16*

P/O Figure 6-1. SYMBol Subsystem Syntax Diagram

BASE command

The BASE command sets the base in which symbols for the specified label will be displayed in the symbol menu. It also specifies the base in which the symbol offsets are displayed when symbols are used.

Note

BINary is not available for labels with more than 20 bits assigned. In this case the base will default to HEXadecimal.

Command Syntax: :SYMBol:BASE <label_name>,<base_value>

where:

<label_name> ::= string of up to 6 alphanumeric characters
<base_value> ::= {BINary | HEXadecimal | OCTal | DECimal | ASCii}

Example: OUTPUT XXX;":SYMBol:BASE 'DATA',HEXadecimal"

PATTern command

The PATTern command allows you to create a pattern symbol for the specified label. The pattern may contain "don't cares" in the form of XX...X's.

Command Syntax: :SYMBol:PATTern <label_name>,<symbol_name>,<pattern_value>

where:

<label_name> ::= string of up to 6 alphanumeric characters
<symbol_name> ::= string of up to 16 alphanumeric characters
<pattern_value> ::= string of one of the following forms:

'#B01X...' for binary
'#Q01234567X..' for octal
'#H0123456789ABCDEFX...' for hexadecimal
'0123456789...' for decimal

Example: OUTPUT XXX;":SYMBol:PATTern 'STAT', 'MEM_RD','#H01XX'"

RANGe command

The RANGe command allows you to create a range symbol containing a start value and a stop value for the specified label.

Note

Don't cares are not allowed in range symbols.

Command Syntax: :SYMBol:RANGe <label_name>,<symbol_name>,
<start_value>,<stop_value>

where:

<label_name> ::= string of up to 6 alphanumeric characters
<symbol_name> ::= string of up to 16 alphanumeric characters
<start_value> ::= string of one of the following forms:
"#B01..." for binary
'#Q01234567..' for octal
'#H0123456789ABCDEF...' for hexadecimal
'0123456789...' for decimal
<stop_value> ::= string of one of the following forms:
"#B01..." for binary
'#Q01234567..' for octal
'#H0123456789ABCDEF...' for hexadecimal
'0123456789...' for decimal

Example: OUTPUT XXX;":SYMBol:RANGe 'STAT', 'IO_ACCESS','#H0000','#H000F'"

REMove command

The REMove command deletes all symbols from the symbol menu.

Command Syntax: :SYMBol:REMove

Example: OUTPUT XXX;":SYMBol:REMove"

WIDTh command

The WIDTh command specifies the width (number of characters) in which the symbol names will be displayed when symbols are used.

Note

The WIDTh command does not affect the displayed length of the symbol offset value.

Command Syntax: :SYMBol:WIDTh <label_name>,<width_value>

where:

<label_name> ::= string of up to 6 alphanumeric characters
<width_value> ::= integer from 1 to 16

Example: OUTPUT XXX;":SYMBol:WIDTh 'DATA',9 "

Data and Setup Commands A

Introduction

The DATA and SETup commands are system commands that allow you to send and receive instrument configuration, setup and program data to and from a controller in block form. This is useful for saving block data for re-loading the pattern generator. This appendix explains how to use these commands.

The block data for the DATA command is broken into byte positions and descriptions. The SETup command block data is not described in detail. No changes should be made to the "config" section of the block data.

Definition of Block Data

Block data in the # format is made up of a block length specifier and a variable number of sections.

<block length specifier> <section 1>...<section N>

The block length specifier is defined as follows:

#8<length>

where:

<length> :: = the total length of all sections in byte format (must be represented with 8 digits)

Total bytes in 5 sections. Includes section headers — Excludes 10 bytes of "#8nnnn..."

For example, if the total length of the block (all sections) is 14506 bytes, the block length specifier would be "#800014560" since the length must be represented with 8 digits.

Sections consist of a section header followed by the section data as follows:

<section header> <section data>

where:

<section header> ::= 10 bytes for the section name
1 byte reserved (always 0)
1 byte for the module ID code (21 for pattern generator)
4 bytes for the length of the data in bytes

The section data format varies for each section and may be any length.

Note

The total length of a section is 16 (for the section header) plus the length of the section data. Thus, when calculating the length of a block of configuration data, don't forget to add the length of the headers.

HP-IB Example:

```
10 DIM Block$[3200]  !allocate enough memory for block data
20 DIM Specifier$[2]
30 OUTPUT XXX;"EOI ON"
40 OUTPUT XXX;"SYSTEM:HEAD OFF"
50 OUTPUT XXX;"SELECT 1"  !select module
60 OUTPUT XXX;"SYSTEM:DATA?  !send the data query
70 ENTER XXX USING"#,2A";Specifier$  !read in #8
80 ENTER XXX USING"#,8D",Blocklength  !read in block length
90 ENTER XXX USING"-K",Block$  !read in data
```

SYSTem:SETup

The SETup command for the pattern generator module is used to configure system parameters, such as the pod and bit assignment, input thresholds, strobe values, and clock rates.

The "CONFIG" section consists of 1128 bytes of information which fully describe the main parameters for the pattern generator. The total length of the section is 1144 bytes (recall that the section header is 16 bytes).

The data in this section of the block should not be changed to ensure proper pattern generator operation.

Command Syntax: :SYSTem:SETup <block data in # format>

Query Syntax: :SYSTem:SETup?

Returned Format: [:SYSTem:SETup] <block data in # format> <NL>

SYSTem:DATA

The DATA command is used to send and receive the pattern generator main program listings and the macro listings. The complete pattern generator data block consists of five sections not counting the SYMBOL section. The sections are:

Section 1 "MAINPROG"
Section 2 "MACRO1"
Section 3 "MACRO2"
Section 4 "MACRO3"
Section 5 "MACRO4"

Command Syntax: :SYSTem:DATA <block data in # format>

Query Syntax: :SYSTem:DATA?

Returned Format: [:SYSTem:DATA] <block data in # format> <NL>

Section 1 "MAINPROG" The Main Program section contains the program listing data. The length of this section depends on the length of the program listing and the number of expansion cards connected to the master card.

The data for this section is as follows:

1 **16 bytes** - section header "MAINPROG"

17 **2 bytes** - number of pods - The total number of pods for which the program is written. Valid values are 2 to 26 in increments of 6 because the master card has 2 pods and each expansion card has 6 pods.

1 10 bytes mainprog
10 1 bytes reserved
11 1 bytes ID
12 4 bytes Section size
w/16 byte header

19 **2 bytes** - total program length - The total length of the pattern generator program with macros expanded. Valid values are 1 to 4095.

21 **2 bytes** - edit line index - The index of the current editing line on screen. Valid values are 0 to the total program length - 1.

23 **6 bytes** - reserved - The values should be set to zero.

29 **2 bytes** - total program lines - The total number of program lines with macros not expanded. Valid values are 1 to total program length.

Note

Macro calls require two program lines. The first line contains the MACRO opcode and the values for PARAM1 for each label. The second line contains the PARAMETER opcode and the values for PARAM2 for each label.

31 **number of bytes = total program lines** (N) - opcode list - This block contains a list of the opcodes for the main program in order of ascending line numbers. The opcode for each main program line occupies one byte, with the opcode for line N preceding the opcode for line N + 1 in the structure. The valid opcodes are:

Note

A macro opcode must be followed by the appropriate macro parameter opcode.

0 - NOOP
1 - WAIT IMB
2 - WAIT EXTERNAL
3 - REPEAT
4 - SIGNAL IMB
8 - BREAK
16 - MACRO1
17 - MACRO2

18 - MACRO3
19 - MACRO4
20 - MACRO1 PARAMETER
21 - MACRO2 PARAMETER
22 - MACRO3 PARAMETER
23 - MACRO4 PARAMETER

Note

Byte position from here on varies with total program line length.

31 + N **number of bytes = total program lines (N)** - parameters - This block contains a list of the parameters for the main program in order of ascending line numbers. The parameter for each main program line occupies one byte, with the parameter for line N preceding the parameter for line N + 1 in the structure. These bytes will only be valid when they correspond to main program lines containing either a REPEAT or a WAIT instruction. The valid values for the WAIT instruction are 0 to 255, where 0 in a bit position means continue and a 1 in a bit position means wait. The WAIT bit positions are defined as follows:

WAIT PARAMETER BITS 2 1 0	BIT POSITION
0 0 0	0
0 0 1	4
0 1 0	2
0 1 1	6
1 0 0	1
1 0 1	5
1 1 0	3
1 1 1	7

18520/BL07

All other instructions parameters have no effect and should be zero.

31 + 2N **number of bytes = total program lines (N) * number of pods (P)** - data array - The data array block contains the 0/1 pattern information for each main program line and each pod. Program line number is the primary array index and the pod number is the secondary index, hence P bytes of pod data are sent for a given line before any data from the next line. For a given line, pod data is sent in order of descending pod numbers with master pod data before expansion pod data. When more than one expansion pod is installed, the data is sent in order of ascending slot numbers 1-5 or A-E.

With this organization, data will be sent out in the same order as if read from the LISTing menu as English text from left to right, then top to bottom. A "1" in the data array means generate a "1" on the corresponding channel of the output pod, assuming positive label polarity.

31 + 2N + PN **number of bytes = total program lines (N) * number of pods (P) -** auto-fill array - This array contains auto-fill/no auto-fill information for each main program line and each pod. This array is organized exactly as the data array described above, therefore bits map directly across from one to the other. A "1" in this array means output the last specified pattern for the corresponding bit from the data array when the auto-fill bit was 0.

Note

The easiest way to send a program is to indicate all the data in the data array and to send all 0's in the autofill array.

Section 2 "MACRO1" The "MACRO1" section contains all the program listing for MACRO1. The length of this section varies depending on the length of the macro listing and the number of expansion cards connected to the master card.

1 **16 bytes** - section header "MACRO1"

17 **1 byte** - number of pods - The total number of pods for which this macro is defined. Valid values are 2 to 26 in increments of 6 because the master card has 2 pods and each expansion card has 6 pods.

18 **1 byte** - length of macro - The total number of program lines in the macro. Valid values are 2 to 62.

19 **1 byte** - macro references - The total number of times the macro is referenced by the main program. Valid values are 0 to 127.

20 **7 bytes** - macro name - This is the name of the macro. The name may be up to 6 alphanumeric characters long. The last byte must be a null (0).

27 **1 byte** - macro number - This is the current macro number. Valid values are 0 through 3.

28 **280 bytes** - parameter names - This structure contains the parameter name for each parameter in the macro. The 280 bytes are organized as 7 bytes for each name * 20 labels for PARAM1 data, followed by 7 bytes for each name * 20 labels for PARAM2 data. The name may be up to 6 alphanumeric characters long and the seventh byte for each name must be null (0).

308 **6 bytes * macro lines (M) +2** - parameter values - This represents the parameter usages within the macros and should all be zeros.

Note

Byte position from here on varies with the macro program length.

310+6M **number of bytes = macro lines (M)** - opcode list - This is a list of the opcodes for the macro program. There should be one opcode for each line in the macro program. Refer to the "MAINPROG" opcode list for the description of opcodes.

310+7M **number of bytes = macro lines (M)** - parameters - This is a list of the parameters for the WAIT and REPEAT instructions used within the macro. Refer to "MAINPROG" parameters for a description of this structure.

310+8M **number of bytes = macro lines (M)** - data array - This is the 0/1 pattern information for each pod. A "1" in the data array means generate a "1" on the associated output line, subject to the polarity of that label. Refer to the "MAINPROG" data array for the description of this structure.

310+8M+PM **number of bytes = macro lines (M) * number of pods** auto-fill array - This represents the auto-fill/no auto-fill information. A "1" means output the last specified pattern for that bit when the auto-fill array was 0. Refer to the "MAINPROG" autofill array for the description of this structure.

Sections 3, 4, 5 "MACRO2", "MACRO3", "MACRO4"

The program listing for Macros 2 through Macros 4 are identical to Macro 1. The length of these sections vary with the length of the macro listing and the number of expansion cards connected to the master card. Refer to Section 2 of this appendix for details of the section definitions.

Index

P

Q

R

S

T

W