REMOTE CONTROL AND PROGRAMMING REFERENCE

for the FLUKE 99

ScopeMeter Test Tool

==========

This file contains remote control and programming information for the FLUKE 99 with use of the PM9080/001 Isolated Optical to RS232 Adapter/Cable.

It consists of the following chapters:

APPENDIXES

1. INSTALLATION OF THE PM9080/001

- Connect the PM9080/001 to the RS232 port of the computer as indicated in the PM9080/001 Instruction Manual. If necessary, use the 9-pin to 25-pin adapter and the 25-pin gender changer included in the PM9080/001 shipment.

- Hook the PM9080/001 cable to the ScopeMeter test tool as indicated in the PM9080/001 Instruction Manual.

- Turn on the computer and the ScopeMeter test tool.

- Make sure that the communication settings match for the RS232 port of the computer and the ScopeMeter test tool.

  After power-on, the default settings of the ScopeMeter test tool are as follows:

  1200 baud, No parity, 8 data bits, 1 stop bit

  You can modify these settings with the PC (Program Communication) command. See chapter 3 COMMAND REFERENCE.

  You can modify the computer RS232 port settings to match the above ScopeMeter test tool settings with the following DOS command:

  MODE COM1:1200,N,8,1

  This command assumes that COM1 is the RS232 port used on the computer. Replace COM1 in the above command with COM2, COM3, or COM4 if one of these ports is used. You can place this command in the computer startup file AUTOEXEC.BAT so that the default settings for the computer are the same as for the ScopeMeter test tool. If you want to use a higher data transfer speed (baud rate), let your QBASIC program change the settings for both the computer and the ScopeMeter test tool. See the example under the PC (Program Communication) command in chapter 3 COMMAND REFERENCE.

2. INTRODUCTION TO PROGRAMMING

** Basic Programming Information **

When you have installed the PM9080/001 as described in the previous chapter, you can control the ScopeMeter test tool from the computer with simple communication facilities, such as GWBASIC, QuickBASIC and QBASIC (programming languages from Microsoft Corporation).

All examples given in this manual are in the QBASIC language but will also run in QuickBASIC. QuickBASIC allows you to make executable files from programs so you can start such programs directly from DOS.
It is assumed that you have knowledge of these programming languages. QBASIC is supplied with Microsoft Operating System MS-DOS 5.0 and higher, and has an 'on-line' Help function.

Features of the syntax and protocol for the ScopeMeter test tool are as follows:

- Easy input format with a 'forgiving' syntax:
  All commands consist of two characters that can be UPPER or lower case.
  Parameters that sometimes follow the command may be separated from it by one or more separation characters.

- Strict and consistent output format:
  Alpha character responses are always in UPPERCASE.
  Parameters are always separated by a comma ("," = ASCII 44, see Appendix D).
  Responses always end with the carriage return code (ASCII 13). Because the carriage return code is a non-visible character (not visible on the screen or on paper), this character is represented as <cr> in the command syntax.
- Synchronization between input and output:
  After receipt of every command, the ScopeMeter test tool returns an acknowledge character (digit) followed by the carrige return code (ASCII 13). This indicates that the command has been successfully received and executed.
  The computer program must always read this acknowledge response before sending the next command to the ScopeMeter test tool.

** Commands sent to the ScopeMeter test tool **

All commands for the ScopeMeter test tool consist of a header made up of two alpha characters sometimes followed by parameters. Example:

RI	This is the Reset Instrument command. It resets the ScopeMeter test tool.

Some of the commands are followed by one or more parameters to give the ScopeMeter test tool more information. Example:

SS101	This is the Save Setup command. It saves the present acquisition settings in memory. The SS header is followed by the parameter 101 to indicate where to store the settings. The meaning of this parameter is described in Chapter 3 COMMAND REFERENCE.

Some commands require several parameters. Example:

PC2400,N,8,1	This is the Program Communication command. This command requires four parameters. The parameters are separated by a comma, which is called the Program Data Separator. You may also use spaces (SP = ASCII 32) or tabs (HT = ASCII 9) to separate parameters. For example, the previous example may also be written as follows: PC 2400 N 8 1

You may use several spaces or tabs to separate parameters but if a comma is used for separation, you may use only one comma between the parameters. Also refer to the section 'Data Separators'.

A code at the end of each command tells the ScopeMeter test tool that the command is ended. This is the carriage return code (ASCII 13) and is called the Program Message Terminator. This code is needed to indicate to the ScopeMeter test tool that the command is completed so it can start executing the command. Also refer to the section 'Command and Response Terminators'.

** Responses received from the ScopeMeter test tool **

After each command sent to the ScopeMeter test tool, there is an automatic response from it, indicated as <acknowledge> (which you MUST input), to let the computer know whether or not the received command has been successfully executed. Refer to the 'Acknowledge' section below.

There are several commands that ask the ScopeMeter test tool for response data. Such commands are called Queries. Example:

ID — This is the IDentification query, which asks for the model number and the software version of the ScopeMeter test tool.

When the ScopeMeter test tool has received a query, it sends the <acknowledge> reply as it does after any command, but now it is followed by the queried response data.

The format of the response data depends upon which query is sent. When a response consists of different response data portions, these are separated with commas (ASCII code 44). Also refer to the section 'Data Separators'.

All response data, <acknowledge> as well as following (queried) response data are terminated with the carriage return code (<cr> = ASCII 13). Also refer to the section 'Command and Response Terminators'.

** Acknowledge **

After receiving of a command, the ScopeMeter test tool automatically returns the <acknowledge> response to let the computer know whether or not the received command has been successfully executed.
This response is a one-digit number followed by <cr> as response terminator. If <acknowledge> is 0, it indicates that the ScopeMeter test tool has successfully executed the command. If the command was a query, the <acknowledge><cr> response is immediately followed by the queried response data terminated with <cr>.
If <acknowledge> is 1 or higher, it indicates that the ScopeMeter test tool has not executed the command successfully. In that case, if the command was a query, the <acknowledge><cr> response is NOT followed by any further response data.
There can be several reasons for a non-zero <acknowledge> response. For more information see Appendix A.
In case of an error you can obtain more detailed status information by using the ST (STATUS) query.

Note: YOU MUST ALWAYS INPUT <acknowledge>, EVEN WHEN THE COMMAND WAS NOT A QUERY.

** Data Separators **

Data Separators are used between parameters sent to the ScopeMeter test tool and between values and strings received from the ScopeMeter test tool. The following list shows the data separators that can be used:

- Program Data Separator

Name	Character	ASCII Value Decimal	Comments
comma	,	44	Single comma allowed
space	SP	32	Multiple spaces allowed
tab	HT	9	Multiple tabs allowed

- Response Data Separator

Name	Character	ASCII Value Decimal	Comments
comma	,	44	

** Command and Response Terminators **
(Message Terminators)

- Command (Program Message) Terminators

  A code is needed at the end of each command to tell the ScopeMeter test tool that the command is ended, and that it can start executing the command. This code is called the Program Message Terminator. The code needed for the test tool is carriage return (ASCII code 13 decimal).
  Notes:

  1. The carriage return code is a non-visible ASCII character. Therefore this code is represented as <cr> in the Command Syntax and Response Syntax lines given for each command.
  2. The QBASIC programming language, which is used for all program examples, automatically adds a carriage return to the end of the command output. (In the QBASIC language, this is the PRINT #.... statement.)

  After <cr> is recognized by the ScopeMeter test tool, the entered command is executed. After EACH command the ScopeMeter test tool returns <acknowledge><cr> to the computer to signal the end of the command processing (also see the section 'Acknowledge'.)

- Response (Message) Terminators

  The response from the ScopeMeter test tool ends with a carriage return (ASCII 13). This is indicated as <cr> in the Response Syntax for each command.

** Typical program sequence **
An example

A typical program sequence consists of the following user actions:

1. Set the communication parameters for the RS232 port of the computer to match the ScopeMeter test tool settings.

2. Output a command or query to the test tool.

3. Input the acknowledge response from the test tool.

   If the response value is zero, go to step 4.

   If the response value is non-zero, the test tool did not execute the previous command. Read the error message from the following acknowledge subroutine, recover the error, and repeat the command or query. (This is not shown in the following program example.)

4. If a query was output to the test tool, input its response.

5. The sequence of points 2, 3, and 4 may be repeated for different commands or queries.

6. Close the communication channel.

Refer to the program example on the next page.

```
'Example of a typical program sequence:

'*****************  Begin example program  ***************

OPEN "COM1:1200,N,8,1,CS,DS,RB2048" FOR RANDOM AS #1

     'This QBASIC program line sets the parameters for the
     'RS232 port (COM1 on the Computer) to match the test
     'tool power-on default settings. It also opens a
     'communication channel (assigned #1) for input or output
     'through the COM1 port. Your test tool must be connected
     'to this port. "RB2048" sets the size of the computer
     'receive buffer to 2048 bytes to prevent buffer overflow
     'during communication with the ScopeMeter test tool.

PRINT #1, "ID"

     'Outputs the IDENTITY command (query) to the test tool.

GOSUB Acknowledge

     'This subroutine inputs the acknowledge response from
     'the test tool and displays an error message if the
     'acknowledge value is non-zero.

INPUT #1, Response$

     'This inputs the response data from the IDENTITY query.

PRINT Response$

     'Displays the queried data.

CLOSE #1

     'This closes the communication channel.

END

     'This ends the program.

'
```

```
'*****************  Acknowledge subroutine  ******************
'Use this subroutine after each command or query sent to the
'ScopeMeter test tool. This routine inputs the acknowledge
'response from the test tool. If the response is non-zero,
'the previous command was not correct or was not correctly
'received by the test tool. Then an error message is
'displayed and the program is aborted.

Acknowledge:
INPUT #1, ACK           'Reads acknowledge from test tool.
IF ACK <> 0 THEN
   PRINT "Error "; ACK; ": ";
   SELECT CASE ACK
     CASE 1
        PRINT "Syntax Error"
     CASE 2
        PRINT "Execution Error"
     CASE 3
        PRINT "Synchronization Error"
     CASE 4
        PRINT "Communication Error"
     CASE IS < 1
        PRINT "Unknown Acknowledge"
     CASE IS > 4
        PRINT "Unknown Acknowledge"
   END SELECT
   PRINT "Program aborted."
   END
END IF
RETURN

'******************  End example program  *****************
```

3. COMMAND REFERENCE

CONVENTIONS

** Page layout used for each command **

- Header

  Each command description starts on a new page with a header for quickly finding the command. This header indicates the command name and the two-character header used for the command syntax. Example:

  ```
  ============================================================
               AUTO SETUP                      AS
  ------------------------------------------------------------
  ```

  Where AUTO SETUP is the name of the command (no syntax),

  and AS are the first two characters used for the command syntax (not the complete syntax).

- Purpose:
  Explains what the command does or what it is used for.

- Command Syntax:
  Shows the syntax for the command. Parameters are separated by commas. Commands are terminated by <cr> (carriage return).

- Response Syntax:
  Shows the format of the response from the ScopeMeter test tool. Responses are terminated by <cr> (carriage return). Each Response Syntax starts with the <acknowledge> response, followed by the query response if the syntax relates to a query.

- Example:
  This is an example QBASIC program which shows how you can use the command. The example may also include some other commands to show the relation with these commands.
  The following two comment lines (start with ') successively indicate the beginning and the end of an example program.

```
'******************  Begin example program  ****************

'******************  End example program  ******************
```

Use an MS-DOS Editor and copy the complete program between these two lines to a file name with the .BAS extension.
Start QBASIC and open this file from the FILE menu.
Long programs (longer than 55 lines) include page breaks. Such page breaks are preceded by the ' (remark) character to prevent the QBASIC interpreter from interpreting them as an incorrect statement.
When you have connected the ScopeMeter test tool as indicated in the PM9080 Instruction Manual, you can start the program from the RUN menu.

** Syntax conventions **

The Command Syntax and the Response Syntax may contain the following characters:

UPPERCASE	These characters are part of the syntax. For commands, lower case is also allowed.
<...>	An expression between these brackets is a code, such as <cr> (carriage return) that can not be expressed in a printable character, or it is a parameter that is further specified. Do not insert the brackets in the command!
[...]	The item between these brackets is optional. This means that you may omit it. Do not insert the brackets in the command!
|	This is a separator between selectable items. This means that you must choose only one of the items (exclusive or).

** Overview of commands for the ScopeMeter test tool **

AUTO SETUP AS

Purpose:

Invokes an automatic setup for the active mode. The result of this command is the same as pressing the AUTO SET key on the ScopeMeter test tool.

Note: You can select the items that are affected by the AUTO SET procedure via the USER OPTIONS key and F1 (MORE OPTIONS) softkey. Select AUTO SET MENU.

Command Syntax:

AS<cr>

Response Syntax:

<acknowledge><cr>

Example:

The following example program sends an AUTO SETUP command to the ScopeMeter test tool. Connect a repetitive signal on INPUT A to see the effect of AUTO SETUP. Run the program for both the METER and the SCOPE mode.

```
'*****************  Begin example program  *****************

CLS                       'Clears the PC screen.
OPEN "COM1:1200,N,8,1,CS,DS,RB2048" FOR RANDOM AS #1
PRINT #1, "AS"            'Sends AUTO SETUP command.
GOSUB Acknowledge         'Input acknowledge from test tool.
CLOSE #1
END


'*****************  Acknowledge subroutine  ******************
'Use this subroutine after each command or query sent to the
'ScopeMeter test tool. This routine inputs the acknowledge
'response from the test tool. If the response is non-zero,
'the previous command was not correct or was not correctly
'received by the test tool. Then an error message is
'displayed and the program is aborted.

Acknowledge:
INPUT #1, ACK             'Reads acknowledge from test tool.
IF ACK <> 0 THEN
   PRINT "Error "; ACK; ": ";
   SELECT CASE ACK
     CASE 1
        PRINT "Syntax Error"
     CASE 2
        PRINT "Execution Error"
     CASE 3
        PRINT "Synchronization Error"
     CASE 4
        PRINT "Communication Error"
     CASE IS < 1
        PRINT "Unknown Acknowledge"
     CASE IS > 4
        PRINT "Unknown Acknowledge"
   END SELECT
   PRINT "Program aborted."
   END
END IF
RETURN

'******************  End example program  ******************
```

ARM TRIGGER AT

Purpose:

Resets and arms the trigger system for a new acquisition trigger when in the SCOPE mode. In the METER mode the acknowledge from this command equals the value 2 (=Execution error). This command is used for single shot measurements in the SCOPE mode.

Command Syntax:

AT<cr>

Response Syntax:

<acknowledge><cr>

Example:

The following example program arms the trigger system of the ScopeMeter test tool with the AT command.
This means that after this command the test tool starts an acquisition when a trigger occurs from the signal (when exceeding the trigger level) or from a TA (Trigger Acquisition) command.

To see the effect of the AT command used in the following program example, do the following:

- Connect a repetitive signal to INPUT A.
- Set the ScopeMeter test tool in SCOPE mode.
- Press AUTO SET key. The waveform appears on the display.
- Select the SINGLE trigger and the 'wait for trigger' modes:
    - press the TRIGGER key
    - press F1 (MORE TRIGGER)
    - press the down key to select TRACE REPEAT, press F5.
        - press the up/down key to select 'single', press F5.
    - press the down key to select TRACE START, press F5.
        - press the up/down key to select 'wait for trigger', press F5.
- Press F1 (CLOSE) to close the window.
  The waveform is now frozen.
- Press the TIME ns key to increase the time base speed. The waveform image does not change (still frozen).
- Run the following program, which sends a AT (ARM TRIGGER) command to the ScopeMeter test tool.

The waveform will then change in accordance with the new time base setting.

After the AT command it is assumed that the signal amplitude is sufficient to trigger the acquisition. If it is not, you can use the TA (TRIGGER ACQUISITION) command to force the acquisition to be triggered. But this is not useful if you want the acquisition to be started on a signal edge for synchronization purposes.

Also see the example program for the IS command, which also uses the AT command for a single shot application.

```
'*****************  Begin example program  *****************

OPEN "COM1:1200,N,8,1,CS,DS,RB2048" FOR RANDOM AS #1
PRINT #1, "AT"          'Sends the ARM TRIGGER command.
GOSUB Acknowledge       'Input acknowledge from test tool.
CLOSE #1
END


'****************  Acknowledge subroutine  ******************
'Use this subroutine after each command or query sent to the
'ScopeMeter test tool. This routine inputs the acknowledge
'response from the test tool. If the response is non-zero,
'the previous command was not correct or was not correctly
'received by the test tool. Then an error message is
'displayed and the program is aborted.

Acknowledge:
INPUT #1, ACK           'Reads acknowledge from test tool.
IF ACK <> 0 THEN
   PRINT "Error "; ACK; ": ";
   SELECT CASE ACK
     CASE 1
        PRINT "Syntax Error"
     CASE 2
        PRINT "Execution Error"
     CASE 3
        PRINT "Synchronization Error"
     CASE 4
        PRINT "Communication Error"
     CASE IS < 1
        PRINT "Unknown Acknowledge"
     CASE IS > 4
        PRINT "Unknown Acknowledge"
   END SELECT
   PRINT "Program aborted."
   END
END IF
RETURN
```

```
'******************  End example program  ******************
```

CPL VERSION QUERY CV

Purpose:

Queries the CPL interface version.

Command Syntax:

CV<cr>

Response Syntax:

<acknowledge><cr><version><cr>

where
<version> is an ASCII string representing the year this version has been created.

Example:

```
'*****************  Begin example program  *****************

OPEN "COM1:1200,N,8,1,CS,DS,RB2048" FOR RANDOM AS #1
PRINT #1,"CV"             'Sends CPL VERSION query.
GOSUB Acknowledge         'Input acknowledge from test tool.
INPUT #1,VERSION$         'Inputs queried data.
PRINT "CPL Version "; VERSION$    'Displays version data.
END


'*****************  Acknowledge subroutine  ******************
'Use this subroutine after each command or query sent to the
'ScopeMeter test tool. This routine inputs the acknowledge
'response from the test tool. If the response is non-zero,
'the previous command was not correct or was not correctly
'received by the test tool. Then an error message is
'displayed and the program is aborted.

Acknowledge:
INPUT #1, ACK           'Reads acknowledge from test tool.
IF ACK <> 0 THEN
   PRINT "Error "; ACK; ": ";
   SELECT CASE ACK
     CASE 1
        PRINT "Syntax Error"
     CASE 2
        PRINT "Execution Error"
     CASE 3
        PRINT "Synchronization Error"
     CASE 4
        PRINT "Communication Error"
     CASE IS < 1
        PRINT "Unknown Acknowledge"
     CASE IS > 4
        PRINT "Unknown Acknowledge"
   END SELECT
   PRINT "Program aborted."
   END
END IF
RETURN

'*****************  End example program  *****************
```

DEFAULT SETUP DS

Purpose:

Performs a Master Reset. This sets the ScopeMeter test tool to the default settings (factory settings). Refer to the Users Manual for more information.

Command Syntax:

DS<cr>

Response Syntax:

<acknowledge><cr>

Note: Wait for at least 2 seconds after the <acknowledge> reply has been received, to let the test tool settle itself before you send the next command.

Example:

```
'*****************  Begin example program  *****************

OPEN "COM1:1200,N,8,1,CS,DS,RB2048" FOR RANDOM AS #1
CLS
PRINT #1, "DS"          'Sends DEFAULT SETUP command.
GOSUB Acknowledge       'Input acknowledge from test tool.
SLEEP 2                 'Delay (2 s) necessary after "DS".
PRINT #1, "ID"          'Sends the IDENTIFICATION query.
GOSUB Acknowledge       'Input acknowledge from test tool.
INPUT #1, ID$           'Inputs identity data from test tool.
PRINT ID$               'Displays identity data.
CLOSE #1
END


'****************  Acknowledge subroutine  ******************
'Use this subroutine after each command or query sent to the
'ScopeMeter test tool. This routine inputs the acknowledge
'response from the test tool. If the response is non-zero,
'the previous command was not correct or was not correctly
'received by the test tool. Then an error message is
'displayed and the program is aborted.

Acknowledge:
INPUT #1, ACK           'Reads acknowledge from test tool.
IF ACK <> 0 THEN
   PRINT "Error "; ACK; ": ";
   SELECT CASE ACK
     CASE 1
        PRINT "Syntax Error"
     CASE 2
        PRINT "Execution Error"
     CASE 3
        PRINT "Synchronization Error"
     CASE 4
        PRINT "Communication Error"
     CASE IS < 1
        PRINT "Unknown Acknowledge"
     CASE IS > 4
        PRINT "Unknown Acknowledge"
   END SELECT
   PRINT "Program aborted."
   END
END IF
RETURN

'*****************  End example program  *****************
```

GO TO LOCAL GL

Purpose:

Sets the ScopeMeter test tool in the local operation mode so the keypad is enabled. This command has the same effect as pressing the AUTO SET key on the keypad, provided that the keypad has not been disabled by the LL (Local Lockout) command. Also refer to the GR (Go to Remote) command and the LL (Local Lockout) command.

Command Syntax:

GL<cr>

Response Syntax:

<acknowledge><cr>

Example:

The following example uses the GR (GO TO REMOTE) command (refer to the description for this command) to set the ScopeMeter test tool in the REMOTE state so that the keypad is disabled (except for the AUTO SET key). After that, the GL (GO TO LOCAL) command is sent so that the keypad is enabled again.

```
'*****************  Begin example program  *****************

CLS                      'Clears the PC screen.
OPEN "COM1:1200,N,8,1,CS,DS,RB2048" FOR RANDOM AS #1
PRINT #1, "GR"           'Sends GO TO REMOTE command.
GOSUB Acknowledge        'Input acknowledge from test tool.
PRINT "All test tool keys (except AUTO SET key) are now"
PRINT "disabled by the GR (GO TO REMOTE) command."
PRINT "Check this."
PRINT "The remote state is indicated as REMOTE on the bottom"
PRINT "right of the display."
PRINT
PRINT "Press any key on the PC keyboard to continue."
SLEEP
PRINT
PRINT #1, "GL"           'Sends GO TO LOCAL command.
GOSUB Acknowledge        'Input acknowledge from test tool.
PRINT "The test tool keys are now enabled again by the "
PRINT "GL (GO TO LOCAL) command."
PRINT "Check this."
CLOSE #1
END

'
```

```
'*****************  Acknowledge subroutine  ******************
'Use this subroutine after each command or query sent to the
'ScopeMeter test tool. This routine inputs the acknowledge
'response from the test tool. If the response is non-zero,
'the previous command was not correct or was not correctly
'received by the test tool. Then an error message is
'displayed and the program is aborted.

Acknowledge:
INPUT #1, ACK           'Reads acknowledge from test tool.
IF ACK <> 0 THEN
   PRINT "Error "; ACK; ": ";
   SELECT CASE ACK
     CASE 1
        PRINT "Syntax Error"
     CASE 2
        PRINT "Execution Error"
     CASE 3
        PRINT "Synchronization Error"
     CASE 4
        PRINT "Communication Error"
     CASE IS < 1
        PRINT "Unknown Acknowledge"
     CASE IS > 4
        PRINT "Unknown Acknowledge"
   END SELECT
   PRINT "Program aborted."
   END
END IF
RETURN

'******************  End example program  ******************
```

GO TO REMOTE GR

Purpose:

Sets the ScopeMeter test tool in the remote operation mode so that the keypad is disabled (except for the AUTO SET key). You can use one of the following methods to return to the local operation mode so that the keypad is enabled:

1. Sending the GL (Go to Local) command.
2. Pressing the AUTO SET key on the keypad. This is only possible if the keypad is not disabled currently by a LL (Local Lockout) command.

Command Syntax:

GR<cr>

Response Syntax:

<acknowledge><cr>

Example:

The following example uses the GR (GO TO REMOTE) command (refer to the description for this command) to set the ScopeMeter test tool in the REMOTE state so that the keypad is disabled (except for the AUTO SET key). After that, the GL (GO TO LOCAL) command is sent so that the keypad is enabled again.

```
'*****************  Begin example program  *****************

CLS                      'Clears the PC screen.
OPEN "COM1:1200,N,8,1,CS,DS,RB2048" FOR RANDOM AS #1
PRINT #1, "GR"           'Sends GO TO REMOTE command.
GOSUB Acknowledge        'Input acknowledge from test tool.
PRINT "All test tool keys (except AUTO SET key) are now"
PRINT "disabled by the GR (GO TO REMOTE) command."
PRINT "Check this."
PRINT "The remote state is indicated as REMOTE on the bottom"
PRINT "right of the display."
PRINT
PRINT "Press any key on the PC keyboard to continue."
SLEEP
PRINT
PRINT #1, "GL"           'Sends GO TO LOCAL command.
GOSUB Acknowledge        'Input acknowledge from test tool.
PRINT "The test tool keys are now enabled again by the "
PRINT "GL (GO TO LOCAL) command."
PRINT "Check this."
CLOSE #1
END

'
```

```
'*****************  Acknowledge subroutine  ******************
'Use this subroutine after each command or query sent to the
'ScopeMeter test tool. This routine inputs the acknowledge
'response from the test tool. If the response is non-zero,
'the previous command was not correct or was not correctly
'received by the test tool. Then an error message is
'displayed and the program is aborted.

Acknowledge:
INPUT #1, ACK           'Reads acknowledge from test tool.
IF ACK <> 0 THEN
   PRINT "Error "; ACK; ": ";
   SELECT CASE ACK
     CASE 1
        PRINT "Syntax Error"
     CASE 2
        PRINT "Execution Error"
     CASE 3
        PRINT "Synchronization Error"
     CASE 4
        PRINT "Communication Error"
     CASE IS < 1
        PRINT "Unknown Acknowledge"
     CASE IS > 4
        PRINT "Unknown Acknowledge"
   END SELECT
   PRINT "Program aborted."
   END
END IF
RETURN

'******************  End example program  ******************
```

IDENTIFICATION ID

Purpose:

Returns information about the model number of the ScopeMeter test tool and its firmware version.

Command Syntax:

ID<cr>

Response Syntax:

<acknowledge><cr><identity><cr>

where,
<identity> is an ASCII string

Example:

The following example program queries the identity data of the ScopeMeter test tool and displays this data on the PC screen.

```
'*****************  Begin example program  *****************

CLS                      'Clears the PC screen.
OPEN "COM1:1200,N,8,1,CS,DS,RB2048" FOR RANDOM AS #1
PRINT #1, "ID"           'Sends IDENTIFICATION query.
GOSUB Acknowledge        'Input acknowledge from test tool.
INPUT #1, IDENT$         'Inputs the queried data.
PRINT IDENT$             'Displays queried data.
CLOSE #1
END


'*****************  Acknowledge subroutine  ******************
'Use this subroutine after each command or query sent to the
'ScopeMeter test tool. This routine inputs the acknowledge
'response from the test tool. If the response is non-zero,
'the previous command was not correct or was not correctly
'received by the test tool. Then an error message is
'displayed and the program is aborted.

Acknowledge:
INPUT #1, ACK            'Reads acknowledge from test tool.
IF ACK <> 0 THEN
   PRINT "Error "; ACK; ": ";
   SELECT CASE ACK
     CASE 1
        PRINT "Syntax Error"
     CASE 2
        PRINT "Execution Error"
     CASE 3
        PRINT "Synchronization Error"
     CASE 4
        PRINT "Communication Error"
     CASE IS < 1
        PRINT "Unknown Acknowledge"
     CASE IS > 4
        PRINT "Unknown Acknowledge"
   END SELECT
   PRINT "Program aborted."
   END
END IF
RETURN

'*****************  End example program  *****************
```

INSTRUMENT STATUS IS

Purpose:

Queries the contents of the test tool's status register. The returned value reflects the present operational status of the test tool. This is a 16-bit word, presented as an integer value, where each bit represents the Boolean value of a related event.

Command Syntax:

IS<cr>

Response Syntax:

<acknowledge><cr>

If <acknowledge> value 0 is returned, this response is immediately continued with the following data:

<status><cr>

where,
<status> = integer value 0 to 1023

<status> value	Status Description	Valid for: *) SC	ME	DI	OM	EX	CO
1	Hardware settled	X	X	X	X	X	X
2	Acquisition armed	X	X	X	X	X	X
4	Acquisition triggered	X					
8	Acquisition busy	X	X	X	X	X	X
16	WAVEFORM A memory filled	X	X	X	X	X	X
32	WAVEFORM B memory filled	X					X
64	WAVEFORM A+/-B memory filled	X					
128	Math function ready	X	X	X	X	X	
256	Numeric results available	X	X	X	X	X	
512	Hold mode active	X	X	X	X	X	X

*) SC = SCOPE mode
ME = METER mode
DI = DIODE measurement
OM = OHM measurement
EX = EXT.mV (EXTernal mV input)
CO = COMPONENT TEST

Example:

Typical use of this query is for single shot measurements in the SCOPE mode.
The following program is an example for this application. The program uses the ScopeMeter in SINGLE acquisition mode and in the HOLD mode. The program sends an AT command and waits for the signal on INPUT A to trigger the acquisition. When the acquisition is finished, the waveform data is queried and input from the test tool. The waveform data are stored to file and the waveform is displayed on the PC screen.

Before you start a single shot measurement of a signal, you must know the settings wanted for the amplitude, time base, and trigger level for the signal. Make the appropriate settings for the signal.

Do the following:
- Set the test tool in the SCOPE mode.
- Press the SUB MENU key and select SINGLE shot mode (F2 key).
- If a window appears asking whether to set TRACE START to 'Wait For Trigger' (when 'Free run' is on), then press F1 (YES) key to let the test tool execute this setting.
- Use the HOLD/RUN key to set the ScopeMeter test tool in the HOLD mode.
- Start the following program.
- The program waits for a signal on INPUT A that is large enough to trigger the acquisition. Offer a signal that exceeds the trigger level setting.

```
'*****************  Begin example program  *****************

CLS                      'Clears the PC screen
OPEN "COM1:1200,N,8,1,CS,DS,RB2048" FOR RANDOM AS #1
PRINT #1, "AT"           'Sends the ARM TRIGGER command
GOSUB Acknowledge        'Input acknowledge from test tool
DO
  PRINT #1, "IS"            'Sends the INSTRUMENT STATUS command
  GOSUB Acknowledge         'Input acknowledge from test tool
  INPUT #1, Status$         'Input Instrument Status
  StatVal = VAL(Status$)    'Decimal value of Instrument Status
  PRINT "Instrument Status : "; StatVal
  IF StatVal = 1 THEN
     PRINT "Trigger not yet armed."
     ARM = 0
  ELSE
     ARM = 1
  END IF
  IF (StatVal AND 2) = 2 THEN
     PRINT "Trigger Armed; waiting for trigger."
  END IF
  IF (StatVal AND 4) = 4 THEN
     PRINT "Acquisition triggered."
  END IF
  IF (StatVal AND 8) = 8 THEN
     PRINT "Acquisition in progress."
  END IF
  IF (StatVal AND 16) = 16 THEN
     PRINT "WAVEFORM A memory filled."
  END IF
  IF (StatVal AND 512) = 512 THEN
     PRINT "Test tool back in HOLD mode."
  END IF
  SLEEP 1
LOOP WHILE ((StatVal > 1) AND (StatVal AND 512) = 0) OR ARM = 0
GOSUB Query.Wave.Form
END
'
```

```
'***************  Subroutine Query.Wave.Form  **************
'This program queries the waveform data measured on INPUT A
'and displays the waveform on the PC screen.
Query.Wave.Form:
CLS                        'Clears the PC screen.
DIM sample$(512)
PRINT #1, "QW101"          'Queries INPUT A waveform data.
GOSUB Acknowledge          'Input acknowledge from test tool.
PRINT "Reading waveform data from test tool...."
PRINT
INPUT #1, NAM$             'Inputs source name (Channel A).
INPUT #1, YUN$             'Inputs Y_unit (V).
INPUT #1, XUN$             'Inputs X_unit (s).
'The following FOR/NEXT loop inputs four values, namely:
'Y_zero, X_zero (offsets) and Y_resol, X_resol (resolutions)
FOR i = 1 TO 4
   INPUT #1, ADM(i)
NEXT i
INPUT #1, YRNG             'Inputs Y_range.
INPUT #1, XNR              'Inputs number of samples
SC% = 0
FOR i = 1 TO XNR
   sample$(i) = INPUT$(1, #1)   'Inputs sample
   SC% = (SC% + ASC(sample$(i))) AND 255    'Calculates checksum
NEXT i
ISC% = ASC(INPUT$(1, #1))   'Inputs trace checksum
CLOSE #1                    'Closes communication channel
IF ISC% <> SC% THEN
   PRINT "CHECKSUM ERROR;  PROGRAM TERMINATED"
   END
END IF
PRINT
filnam$ = "\single.wve"          'The filename to store the
                                 'waveform data.
PRINT "Copying waveform data to file '"; filnam$; "'....."
OPEN filnam$ FOR OUTPUT AS #2
FOR i = 1 TO 4
   PRINT #2, ADM(i)              'Stores X and Y offset and
                                 'resolution values to file
NEXT i
FOR i = 1 TO XNR
   PRINT #2, sample$(i);         'Stores sample value to file
NEXT i
PRINT #2, CHR$(SC%);             'Stores checksum to file
CLOSE #2                         'Closes file I/O
PRINT "Waveform copied to file 'trace.wve'."
SLEEP 1
GOSUB Display.trace
RETURN
'
```

```
'***************  Subroutine Display.trace  ***************
'This subroutine displays the queried waveform on the PC
'screen.
Display.trace:
CLS
SCREEN 2            'CGA, EGA, VGA or MCGA low res. monitor.
VIEW PRINT 2 TO 18
PRINT " Waveform filename: "; filnam$
PRINT TAB(44); "Range    : "; 25 * ADM(3); "V/DIV"
PRINT TAB(44); "Time base: "; 25000 * ADM(4); "ms/DIV"
PRINT
PRINT TAB(44); "Offsets:"
PRINT TAB(44); "    Vertical  : "; ADM(1); " V"
PRINT TAB(44); "    Horizontal: "; ADM(2); " s"
VIEW (10, 20)-(320, 160), , 1
WINDOW (0, -100)-(250, 100)
FOR mark = -100 TO 100 STEP 25
   LINE (0, mark)-(3, mark)          'Left tick marks.
NEXT mark
FOR mark = -100 TO 100 STEP 25
   LINE (247, mark)-(250, mark)      'Right tick marks.
NEXT mark
FOR mark = 0 TO 250 STEP 25
   LINE (mark, -4)-(mark, 4)         'Midscreen tick marks.
NEXT mark
FOR mark = 0 TO 250 STEP 25
   LINE (mark, -100)-(mark, -97)     'Top tick marks.
NEXT mark
FOR mark = 0 TO 250 STEP 25
   LINE (mark, 97)-(mark, 100)       'Bottom tick marks.
NEXT mark
Style% = &HFF00                      'Use to make dashed line.
LINE (250, 0)-(0, 0), , , Style%     'Draw the x-axis
FOR x = 1 TO 250
   trace.dot = ASC(sample$(x)) - 128   'Y-sample relative to
                                       'mid-screen (128).
   LINE -(x - 1, trace.dot)     'Draw a line from the last
                                'point to the new point.
NEXT x
RETURN
'
```

```
'*****************  Acknowledge subroutine  ******************
'Use this subroutine after each command or query sent to the
'ScopeMeter test tool. This routine inputs the acknowledge
'response from the test tool. If the response is non-zero,
'the previous command was not correct or was not correctly
'received by the test tool. Then an error message is
'displayed and the program is aborted.

Acknowledge:
INPUT #1, ACK           'Reads acknowledge from test tool.
IF ACK <> 0 THEN
   PRINT "Error "; ACK; ": ";
   SELECT CASE ACK
     CASE 1
        PRINT "Syntax Error"
     CASE 2
        PRINT "Execution Error"
     CASE 3
        PRINT "Synchronization Error"
     CASE 4
        PRINT "Communication Error"
     CASE IS < 1
        PRINT "Unknown Acknowledge"
     CASE IS > 4
        PRINT "Unknown Acknowledge"
   END SELECT
   PRINT "Program aborted."
   END
END IF
RETURN

'******************  End example program  ******************
```

LOCAL LOCKOUT LL

Purpose:

Inhibits return to the local operation mode by the AUTO SET key. The complete keypad, except the ON/OFF key, is then disabled. Return to local operation is possible by switching the ScopeMeter test tool OFF then ON, or by sending the following remote commands:

1. GL (Go to Local)
2. RI (Reset Instrument)

Command Syntax:

LL<cr>

Response Syntax:

<acknowledge><cr>

Example:

The following example program uses the GR (GO TO REMOTE) command to set the ScopeMeter test tool to the remote operation mode so that the keypad is disabled. Normally you press the AUTO SET key to enable the keypad again (return to the local operation mode). The AUTO SET key is disabled when you have previously sent the LL (LOCAL LOCKOUT) command as shown in the following example. A RI (RESET INSTRUMENT) command is used to return the test tool to the local operation mode (keypad enabled).

```
'*****************  Begin example program  *****************

CLS                     'Clears the PC screen.
OPEN "COM1:1200,N,8,1,CS,DS,RB2048" FOR RANDOM AS #1
PRINT #1, "GR"          'Sends GO TO REMOTE command.
GOSUB Acknowledge       'Input acknowledge from test tool.
PRINT "All test tool keys (except AUTO SET key) are now"
PRINT "disabled by the GR (GO TO REMOTE) command."
PRINT "Check this."
PRINT "The remote state is indicated as REMOTE on the bottom"
PRINT "right of the display."
PRINT
PRINT "Press any key on the PC keyboard to continue."
SLEEP
CLS
PRINT #1, "LL"          'Sends LOCAL LOCKOUT command.
GOSUB Acknowledge       'Input acknowledge from test tool.
PRINT "Now all test tool keys, including the AUTO SET key"
PRINT "are disabled by the LL (LOCAL LOCKOUT) command."
PRINT "Enabling them again is possible as follows:"
PRINT " *  By sending the GL (GO TO LOCAL) command."
PRINT " *  By sending the RI (RESET INSTRUMENT) command."
PRINT " *  By switching the test tool OFF then ON."
PRINT
PRINT "Press any key on the PC keyboard to continue."
SLEEP
CLS
PRINT #1, "RI"          'Sends RESET INSTRUMENT command.
GOSUB Acknowledge       'Input acknowledge from test tool.
PRINT "The test tool keys are now enabled again by the "
PRINT "RI (RESET INSTRUMENT) command."
PRINT "Check this."
CLOSE #1
END
'
```

```
'*****************  Acknowledge subroutine  ******************
'Use this subroutine after each command or query sent to the
'ScopeMeter test tool. This routine inputs the acknowledge
'response from the test tool. If the response is non-zero,
'the previous command was not correct or was not correctly
'received by the test tool. Then an error message is
'displayed and the program is aborted.

Acknowledge:
INPUT #1, ACK           'Reads acknowledge from test tool.
IF ACK <> 0 THEN
   PRINT "Error "; ACK; ": ";
   SELECT CASE ACK
     CASE 1
        PRINT "Syntax Error"
     CASE 2
        PRINT "Execution Error"
     CASE 3
        PRINT "Synchronization Error"
     CASE 4
        PRINT "Communication Error"
     CASE IS < 1
        PRINT "Unknown Acknowledge"
     CASE IS > 4
        PRINT "Unknown Acknowledge"
   END SELECT
   PRINT "Program aborted."
   END
END IF
RETURN

'******************  End example program  ******************
```

PROGRAM COMMUNICATION PC

Purpose:

Programs the following settings used for RS232 communication:

1. Baud rate
2. Parity
3. Number of data bits
4. Number of stop bits
5. Handshake method

Command Syntax:

PC <baudrate>,<parity>,<data bits>,<stop bits>
[,<handshake>]

where,

<baudrate>	=	75|110|150|300|600|1200|2400|4800|9600|19200|38400
<parity>	=	O|E|N (Odd, Even, or No parity)
<data bits>	=	7|8
<stop bits>	=	1
<handshake>	=	XONXOFF (software handshaking)

Default settings after power-on:

<baudrate>	=	1200
<parity>	=	N
<data bits>	=	8
<stop bits>	=	1
<handshake>	=	disabled (see the following notes)

Notes:

Option XONXOFF takes care off handshaking to prevent overflow of the receive buffer (length 254 bytes). The test tool sends XOFF (ASCII code 19 decimal) to tell the computer that it must postpone data transmission until it receives XON (ASCII code 17 decimal). This protocol will only work if the computer reacts on XON and XOFF codes. It is recommended to set the read buffer length for the computer to 2048 or higher as shown in several example programs, with the following QBASIC command:

```
OPEN "COM1:1200,N,8,1,CS,DS,RB2048" FOR RANDOM AS #1
```

When you use PROGRAM WAVEFORM (PW) or QUERY WAVEFORM (QW) commands, you MUST omit option XONXOFF. The PW and QW commands use binary data that may contain XON and XOFF codes.

Response Syntax:

<acknowledge><cr>

Example:

The following program assumes that the ScopeMeter test tool is connected to the COM1 port of the PC and that the power-on defaults are valid. The program switches over to baud rate 19200. This includes first programming the test tool and then the RS232 port of the PC to baud rate 19200. This higher baud rate considerably increases the data transfer speed. After the data transfer from the QS query, the baud rate is set back to 1200, the power-on default value.

```
'*****************  Begin example program  *****************

CLS
OPEN "COM1:1200,N,8,1,CS,DS,RB2048" FOR RANDOM AS #1
                           'Programs COM1 port parameters to
                           'match with the ScopeMeter power-on
                           'defaults.

PRINT #1, "PC19200,N,8,1" 'Programs ScopeMeter to a higher
                           'baud rate (other parameters not
                           'changed).

GOSUB Acknowledge          'Input acknowledge from test tool.

CLOSE #1
OPEN "COM1:19200,N,8,1,CS,DS,RB2048" FOR RANDOM AS #1
                           'Programs COM1 port parameters to
                           'match with the new ScopeMeter
                           'settings.

PRINT #1, "QS"             'Sends QUERY SETUP query.
GOSUB Acknowledge          'Input acknowledge from test tool.
INPUT #1, N, SETUP$        'Inputs queried data.
PRINT SETUP$

PRINT #1, "PC1200,N,8,1"   'Programs ScopeMeter back to the
                           'original communication settings.
GOSUB Acknowledge          'Input acknowledge from test tool.

CLOSE #1
END
'
```

```
'*****************  Acknowledge subroutine  ******************
'Use this subroutine after each command or query sent to the
'ScopeMeter test tool. This routine inputs the acknowledge
'response from the test tool. If the response is non-zero,
'the previous command was not correct or was not correctly
'received by the test tool. Then an error message is
'displayed and the program is aborted.

Acknowledge:
INPUT #1, ACK           'Reads acknowledge from test tool.
IF ACK <> 0 THEN
   PRINT "Error "; ACK; ": ";
   SELECT CASE ACK
     CASE 1
        PRINT "Syntax Error"
     CASE 2
        PRINT "Execution Error"
     CASE 3
        PRINT "Synchronization Error"
     CASE 4
        PRINT "Communication Error"
     CASE IS < 1
        PRINT "Unknown Acknowledge"
     CASE IS > 4
        PRINT "Unknown Acknowledge"
   END SELECT
   PRINT "Program aborted."
   END
END IF
RETURN

'******************  End example program  ******************
```

PROGRAM SETUP PS

Purpose:

Restores a complete setup, which was previously queried with the QS (Query Setup) command and saved in a string variable or to a file.

Command Syntax:

PS 1,<setup><cr>

where,
<setup> = string of hexadecimal characters, representing a completely new setup for the ScopeMeter test tool to be the actual.

Response Syntax:

<acknowledge><cr>

Note: Wait for at least two seconds after the <acknowledge> reply has been received, to let the test tool settle itself before you send the next command.

Remarks:

The ScopeMeter test tool sends the <acknowledge> reply after it has executed the setup from the PS command.
You must send the <setup> string as a whole, exactly as returned from the QS (Query Setup) command.
If you do not follow this rule, the ScopeMeter test tool may crash. A Master Reset may then be necessary to recover the test tool. (Refer to the test tool Users Manual.)

Example:

The following example program demonstrates the use of the QS (QUERY SETUP) and the PS (PROGRAM SETUP) commands.
The present setup is queried from test tool and saved to file. The program asks you to change the test tool settings.
Then the original setup is read from file and sent back to the test tool.

```
'*****************  Begin example program  *****************

OPEN "COM1:1200,N,8,1,CS,DS,RB2048" FOR RANDOM AS #1
CLS
GOSUB ClearPort          'Clears pending data from port.
PRINT #1, "QS"           'Sends QUERY SETUP query.
GOSUB Acknowledge        'Input acknowledge from test tool.
INPUT #1, N              'Reads <count> (=1).
INPUT #1, SURESP$        'Reads queried data.
OPEN "SETUP1" FOR OUTPUT AS #2
                         'Opens file SETUP1 for data storage.
PRINT #2, SURESP$        'Stores setup data to file SETUP1.
CLOSE #2                 'Closes file SETUP1.

PRINT "Present setup data are stored in the file SETUP1"
PRINT "This setup will now be retrieved from the file and"
PRINT "sent back to the test tool."
PRINT "To see if this works, change the present settings and"
PRINT "verify if the test tool returns to the previous"
PRINT "settings."
PRINT
PRINT "Press any key on the PC keyboard to continue."
SLEEP

OPEN "SETUP1" FOR INPUT AS #2
                         'Opens file SETUP1 for data retrieval.
INPUT #2, SETUP$         'Reads setup data from file and
                         'saves these into SETUP$.
CLOSE #2                 'Close file SETUP1.
PRINT #1, "PS 1," + SETUP$  'Programs ScopeMeter with the
                            'setup data stored in SETUP$.
GOSUB Acknowledge        'Input acknowledge from test tool.
END
'
```

```
'*****************  Acknowledge subroutine  ******************
'Use this subroutine after each command or query sent to the
'ScopeMeter test tool. This routine inputs the acknowledge
'response from the test tool. If the response is non-zero,
'the previous command was not correct or was not correctly
'received by the test tool. Then an error message is
'displayed and the program is aborted.

Acknowledge:
INPUT #1, ACK           'Reads acknowledge from test tool.
IF ACK <> 0 THEN
   PRINT "Error "; ACK; ": ";
   SELECT CASE ACK
     CASE 1
        PRINT "Syntax Error"
     CASE 2
        PRINT "Execution Error"
     CASE 3
        PRINT "Synchronization Error"
     CASE 4
        PRINT "Communication Error"
     CASE IS < 1
        PRINT "Unknown Acknowledge"
     CASE IS > 4
        PRINT "Unknown Acknowledge"
   END SELECT
   PRINT "Program aborted."
   END
END IF
RETURN


'*******  Clears pending data from the RS232 port *********
ClearPort:
  WHILE LOC(1) > 0
    Dummy$ = INPUT$(1, #1)
  WEND
RETURN

'******************  End example program  ******************
```

PROGRAM WAVEFORM PW

Purpose:

Programs a waveform or the setup data related to a waveform into the ScopeMeter test tool.

Command Syntax:

Consists of two parts. The first part indicates the destination, the second part sends the waveform or the setup data for a waveform. The ScopeMeter test tool gives an <acknowledge> reply after the first and after the second part. Both <acknowledge> replies must be 0 to successfully complete waveform programming.

Syntax part 1:

PW <trace_no>[,S]

where,

S Indicates that the setup data for a waveform will be given with part 2 of the command syntax. If S is omitted, the waveform data is expected to be given with part 2 of the command syntax.

<trace_no> = 101 to 123, assigned to the following trace destinations:

<trace_no>	TRACE DESTINATION
101	INPUT A
102	INPUT B
103	A +/- B
104 to 123	Stored waveform 1 to 20

Response from part 1:

<acknowledge><cr>

If the value 0 is returned for <acknowledge>, you can send the syntax part 2 as follows.

Syntax part 2:

If you have omitted option S or s in part 1 of the command syntax, you now must give the waveform data as follows:

<sample 1><sample 2>..<sample 512><checksum>

where each sample is an 8-bit byte (binary data), the decimal value of which indicates the vertical position on the display (positions from 0 to 255) of the ScopeMeter test tool. The sample number relates to the horizontal position on the display (positions from 0 to 511). For more detailed descriptions about the waveform structure, refer to Appendix C.
If you have given option S or s with the first syntax part, you now must give the setup data as follows:

1,<setup>

where,
<setup> = hex string format.

CAUTION: Use only original setup strings, read from the ScopeMeter test tool with the QW (Query Waveform) command with option S or s. If you send self-constructed or modified strings, the correct operation of the test tool is no longer guaranteed. Then you must give a Master Reset to recover the test tool.

Response Syntax:

<acknowledge><cr>

Example:

The following example program copies the waveform measured on INPUT A (WAVEFORM A) to waveform memory 1 (WAVEFORM 1). The waveform from INPUT A is queried (command "QW101") and written back to Memory 1 (command "PW104").
At program start, the subroutine 'ClearPort' reads all pending output data from the ScopeMeter test tool to prevent synchronization errors with following commands.
To test the following program, a waveform must be available on INPUT A, and waveform memory 1 must be displayed to show the copied waveform from INPUT A.

Reset the ScopeMeter (power F4 + ON/OFF).
Set the ScopeMeter in SCOPE mode.
To display WAVEFORM 1, there must have been a waveform saved in this memory. Save the WAVEFORM A to WAVEFORM 1 via SAVE and F1 (MORE SAVE) keys and use the RECALL key to set WAVEFORM 1 on the display. Use the UP/DOWN ASSIGN key (select 'move 1') and then press UP or DOWN to move the waveform to the bottom half of the display.
Change the waveform on INPUT A and start the following program. This waveform will then be copied to WAVEFORM 1 shown on the lower half of the display.

```
'*****************  Begin example program  *****************

OPEN "COM1:1200,N,8,1,CS,DS,RB2048" FOR RANDOM AS #1
CLS                        'Clears the PC screen.
GOSUB ClearPort            'Empties pending data from previous
                           'queries.
DIM WAVE%(512)
PRINT #1, "QW101"          'Queries INPUT A waveform data.
GOSUB Acknowledge          'Input acknowledge from test tool.
INPUT #1, NAM$             'Inputs source name (Channel A).
INPUT #1, YUN$             'Inputs Y_unit (V).
INPUT #1, XUN$             'Inputs X_unit (s).
'The following FOR/NEXT loop inputs four values, namely:
'Y_zero, X_zero (offsets) and Y_resol, X_resol (resolutions)
FOR I = 1 TO 4
   INPUT #1, ADM(I)
NEXT I
INPUT #1, YRNG             'Inputs Y_range.
INPUT #1, XNR              'Inputs number of samples.
PRINT "Vertical offset      : "+STR$(ADM(1))+YUN$
PRINT "Horizontal offset    : "+STR$(ADM(2))+XUN$
PRINT "Vertical resolution  : "+STR$(ADM(3))+YUN$
PRINT "Horizontal resolution: "+STR$(ADM(4))+XUN$
PRINT
PRINT "The display can distinguish "+STR$(YRNG)+" vertical levels"
PRINT
PRINT "The waveform from "+NAM$+" consisting of"
PRINT STR$(XNR)+" samples will now be read....."
SC% = 0
FOR I = 1 TO XNR
   sample$ = INPUT$(1, #1)      'Inputs sample.
   WAVE%(I) = ASC(sample$)      'Fills array with ASCII value
                                '(decimal) of each sample.
   SC% = (SC% + WAVE%(I)) AND 255    'Calculates checksum.
NEXT I
ISC% = ASC(INPUT$(1, #1))       'Inputs trace checksum.
IF ISC% <> SC% THEN PRINT "CHECKSUM ERROR"
PRINT "Waveform read and placed in array."
SC% = 0
PRINT #1, "PW104"           'Program waveform into Memory 1.
GOSUB Acknowledge           'Input acknowledge from test tool.
PRINT "Writing waveform to Memory 1 ...."
FOR I = 1 TO XNR
   SC% = (SC% + WAVE%(I)) AND 255
   sample$ = CHR$(WAVE%(I))
   PRINT #1, sample$;
NEXT I
PRINT #1, CHR$(SC% AND 255) 'Sends checksum.
GOSUB Acknowledge           'Input acknowledge from test tool.
PRINT "Waveform copied to Memory 1."
```

```
CLOSE #1
END
'
```

```
'*****************  Acknowledge subroutine  ******************
'Use this subroutine after each command or query sent to the
'ScopeMeter test tool. This routine inputs the acknowledge
'response from the test tool. If the response is non-zero,
'the previous command was not correct or was not correctly
'received by the test tool. Then an error message is
'displayed and the program is aborted.

Acknowledge:
INPUT #1, ACK           'Reads acknowledge from test tool.
IF ACK <> 0 THEN
   PRINT "Error "; ACK; ": ";
   SELECT CASE ACK
     CASE 1
        PRINT "Syntax Error"
     CASE 2
        PRINT "Execution Error"
     CASE 3
        PRINT "Synchronization Error"
     CASE 4
        PRINT "Communication Error"
     CASE IS < 1
        PRINT "Unknown Acknowledge"
     CASE IS > 4
        PRINT "Unknown Acknowledge"
   END SELECT
   PRINT "Program aborted."
   END
END IF
RETURN


'*******  Clears pending data from the RS232 port *********
ClearPort:
WHILE LOC(1) > 0
  Dummy$ = INPUT$(1, #1)
WEND
RETURN

'******************  End example program  ******************
```

QUERY MEASUREMENT QM

Purpose:

Queries a measurement result from the ScopeMeter test tool.

Command Syntax:

QM <field_no>[,V]<cr>

where,
<field_no> = 1 to 17 for the SCOPE mode
1 to 12 for the METER mode

SCOPE MODE

<field_no>	MEASUREMENT TYPE	DESCRIPTION
1	dV	Voltage between cursors
2	dt	Time between cursors
3	1/dt	Reciprocal of field no 2
4	TRIG to left	Trigger to cursor left
5	RMS	RMS value
6	MEAN	MEAN value
7	P-P	Peak to Peak voltage
8	MAX-P	Maximum peak voltage
9	MIN-P	Minimum peak voltage
10	FREQ	Signal frequency
11	RISE	Rise time (10% to 90%)
12	PHASE src>des1	Phase src to destination
13	PHASE src>des2	Phase src to destination
14	PHASE src>des3	Phase src to destination
15	VOLT at left	Voltage at cursor left
16	VOLT at right	Voltage at cursor right
17	TRIG to right	Trigger to cursor right

Remarks:

In the SCOPE MODE, measurements are available only with cursors set on as follows:
- press F1 key to set SCOPE MODE
- press SUB MENU key
- press F5 key to select CURSOR READING.

For the QM 12 or QM 13 or QM 14 commands, there must be two, three or four traces respectively on the display. Select this number as follows:
- press SUB MENU key
- press F1 key to select the MORE SCOPE menu.
- press F4 key to select NEXT PAGE.
- use UP/DOWN keys to highlight READINGS on DISPLAY and press F5 to select.
- Use UP/DOWN keys to highlight the wanted number of readings and press F5 to select.

Also refer to the ScopeMeter test tool Users Manual to understand the previous.

METER MODE

<field_no>	DESCRIPTION
1	First measurement result
2	Second measurement result
3	First calculated result
4	Maximum (record) result
5	Average (record) result
6	Minimum (record) result
7	Time stamp of last recorded maximum
8	Time stamp of last recorded average
9	Time stamp of last recorded minimum
10	Third measurement result
11	Fourth measurement result
12	Record MAX-MIN

Remark:

In the METER MODE, only results that are on the display are available.

Response Syntax:

<acknowledge><cr>

If <acknowledge> value 0 is returned, this response is immediately continued with the following data:

If option V or v is omitted:
<meas type>,<meas value>,<unit suffix><cr>

If option V or v is used:
<meas value><cr>

where,
<meas type> = description of the actual measurement
<meas value> = measurement value (floating point)
<unit suffix>= the unit related to <meas value>

Example for the METER mode:

```
'*****************  Begin example program  *****************

'This example program resets the test tool (RI command),
'programs the default setup (DS command) which sets the
'test tool in the METER mode and queries the first and
'the second measurement result (QM1 and QM2 commands).

CLS                     'Clears the PC screen.
OPEN "COM1:1200,N,8,1,CS,DS,RB2048" FOR RANDOM AS #1
PRINT #1, "RI"          'Sends the RESET INSTRUMENT command.
GOSUB Acknowledge       'Input acknowledge from test tool.
SLEEP 2                 'Delay (2 s) necessary after reset.
PRINT #1, "DS"          'Sends DEFAULT SETUP command.
                        'This sets test tool in METER mode.
GOSUB Acknowledge       'Input acknowledge from test tool.
SLEEP 2                 'Delay (2 s) necessary after "DS".
PRINT #1, "QM1"         'Queries first measurement result.
GOSUB Acknowledge       'Input acknowledge from test tool.
INPUT #1, type$, value$, unit$
PRINT type$, value$, unit$
PRINT #1, "QM2"         'Queries second measurement result.
GOSUB Acknowledge       'Input acknowledge from test tool.
INPUT #1, type$, value$, unit$
PRINT type$, value$, unit$
CLOSE #1
END
'
```

```
'*****************  Acknowledge subroutine  ******************
'Use this subroutine after each command or query sent to the
'ScopeMeter test tool. This routine inputs the acknowledge
'response from the test tool. If the response is non-zero,
'the previous command was not correct or was not correctly
'received by the test tool. Then an error message is
'displayed and the program is aborted.

Acknowledge:
INPUT #1, ACK           'Reads acknowledge from test tool.
IF ACK <> 0 THEN
   PRINT "Error "; ACK; ": ";
   SELECT CASE ACK
     CASE 1
        PRINT "Syntax Error"
     CASE 2
        PRINT "Execution Error"
     CASE 3
        PRINT "Synchronization Error"
     CASE 4
        PRINT "Communication Error"
     CASE IS < 1
        PRINT "Unknown Acknowledge"
     CASE IS > 4
        PRINT "Unknown Acknowledge"
   END SELECT
   PRINT "Program aborted."
   END
END IF
RETURN

'******************  End example program  ******************
```

Example for the SCOPE mode:

```
'*****************  Begin example program  *****************

CLS
PRINT "    Set the test tool in SCOPE mode"
PRINT "    Connect a signal to INPUT 1."
PRINT "    Turn test tool OFF then ON. This automatically"
PRINT "    performs AUTO SET."
PRINT "    Select CURSOR READING (via SUB MENU and F5 key)."
PRINT
PRINT "    Press any key on the PC keyboard to continue."
PRINT
SLEEP  'Wait for key press
CLS
OPEN "COM1:1200,N,8,1,CS,DS,RB2048" FOR RANDOM AS #1
PRINT #1, "QM1"          'Sends QUERY MEASUREMENT 1 command.
GOSUB Acknowledge        'Input acknowledge from test tool.
INPUT #1, type$, value$, unit$
PRINT "Voltage between cursors   : "; type$, value$, unit$
PRINT #1, "QM2"          'Sends QUERY MEASUREMENT 2 command.
GOSUB Acknowledge        'Input acknowledge from test tool.
INPUT #1, type$, value$, unit$
PRINT "Time between cursors      : "; type$, value$, unit$
PRINT #1, "QM5"          'Sends QUERY MEASUREMENT 5 command.
GOSUB Acknowledge        'Input acknowledge from test tool.
INPUT #1, type$, value$, unit$
PRINT "RMS value between cursors : "; type$, value$, unit$
CLOSE #1
END
'
```

```
'*****************  Acknowledge subroutine  ******************
'Use this subroutine after each command or query sent to the
'ScopeMeter test tool. This routine inputs the acknowledge
'response from the test tool. If the response is non-zero,
'the previous command was not correct or was not correctly
'received by the test tool. Then an error message is
'displayed and the program is aborted.

Acknowledge:
INPUT #1, ACK           'Reads acknowledge from test tool.
IF ACK <> 0 THEN
   PRINT "Error "; ACK; ": ";
   SELECT CASE ACK
     CASE 1
        PRINT "Syntax Error"
     CASE 2
        PRINT "Execution Error"
     CASE 3
        PRINT "Synchronization Error"
     CASE 4
        PRINT "Communication Error"
     CASE IS < 1
        PRINT "Unknown Acknowledge"
     CASE IS > 4
        PRINT "Unknown Acknowledge"
   END SELECT
   PRINT "Program aborted."
   END
END IF
RETURN

'******************  End example program  ******************
```

QUERY PRINT QP

Purpose:

Queries the print data of the ScopeMeter test tool in EPSON FX/LQ format, binary coded. This allows you to make a copy of the test tool screen on paper. Also see the VS (VIEW SCREEN) command.

Command Syntax:

QP<cr>

Response Syntax:

<acknowledge><cr>

If <acknowledge> value 0 is returned, this response is immediately continued with the following data:

<count>,<print data><checksum>

where,

<count>	=	Decimal number of bytes in the <print data> block.
<print data>	=	Binary data bytes. The decimal value of each byte is in the range from 0 to 255.
<checksum>	=	One-byte checksum over all bytes in <print data>.

Example:

The following program reads the test tool screen (print) data and copies this data to the printer port LPT1.
The Read Buffer length for the PC is set to 7500 bytes to prevent buffer overflow during input from the test tool.
The print format is only suitable for an EPSON FX/LQ compatible printer.
The data transfer speed (baud rate) is set to 19200 and after the output it is set back to 1200 (default baud rate).

```
'*****************  Begin example program  *****************

CLS
OPEN "COM1:1200,N,8,1" FOR RANDOM AS #1
                          'Programs COM1 port parameters to
                          'match with the ScopeMeter power-on
                          'defaults.
PRINT #1, "PC19200,N,8,1" 'Programs ScopeMeter to a higher
                          'baud rate.
GOSUB Acknowledge         'Input acknowledge from test tool.
CLOSE #1
OPEN "COM1:19200,N,8,1,CS,DS,RB7500" FOR RANDOM AS #1
                          'Programs COM1 port parameters to
                          'match with the new ScopeMeter
                          'settings.
PRINT #1, "QP"            'Sends QUERY PRINT data command.
GOSUB Acknowledge         'Input acknowledge from test tool.
INPUT #1, Count           'Number of bytes that will follow.
PRINT STR$(Count + 1) + " Bytes will be input from test tool."
PRINT
PRINT "Busy reading print data !"
PRINT
GOSUB Response
PRINT #1, "PC1200,N,8,1"  'Programs ScopeMeter back to the
                          'default communication settings.
GOSUB Acknowledge         'Input acknowledge from test tool.

PRINT STR$(LEN(Resp$)) + " Bytes have been input from test tool"
PRINT

OPEN "LPT1" FOR OUTPUT AS #2
PRINT #2, LEFT$(Resp$, Count)   'Print the waveform data.
CLOSE                     'Close all files.
END
'
```

```
'****************  Acknowledge subroutine  ******************
'Use this subroutine after each command or query sent to the
'ScopeMeter test tool. This routine inputs the acknowledge
'response from the test tool. If the response is non-zero,
'the previous command was not correct or was not correctly
'received by the test tool. Then an error message is
'displayed and the program is aborted.

Acknowledge:
INPUT #1, ACK           'Reads acknowledge from test tool.
IF ACK <> 0 THEN
   PRINT "Error "; ACK; ": ";
   SELECT CASE ACK
     CASE 1
        PRINT "Syntax Error"
     CASE 2
        PRINT "Execution Error"
     CASE 3
        PRINT "Synchronization Error"
     CASE 4
        PRINT "Communication Error"
     CASE IS < 1
        PRINT "Unknown Acknowledge"
     CASE IS > 4
        PRINT "Unknown Acknowledge"
   END SELECT
   PRINT "Program aborted."
   END
END IF
RETURN
'
```

```
'****************** Response subroutine *********************
'This subroutine reads bytes from the RS232 buffer as long
'as they enter. When no bytes enter for 1 second, the program
'assumes that the test tool has terminated its response.
'All bytes that enter the buffer are appended to the string
'Resp$.

Response:
  start! = TIMER
  'Wait for bytes (maximum 1 s) to enter RS232 buffer
  WHILE ((TIMER < (start! + 1)) AND (LOC(1) = 0))
  WEND
  IF LOC(1) > 0 THEN       'If RS232 buffer contains bytes
       Resp$ = ""
       DO
           ' LOC(1) gives the number of bytes waiting:
           ScopeInput$ = INPUT$(LOC(1), #1)    'Input bytes
           Resp$ = Resp$ + ScopeInput$         'Append bytes
           start! = TIMER
           WHILE ((TIMER < (start! + 1)) AND (LOC(1) = 0))
           WEND
       LOOP WHILE LOC(1) > 0   'Repeat as long as bytes enter
  END IF
RETURN

'******************  End example program  ******************
```

QUERY SETUP QS

Purpose:

Queries the present acquisition setup data from the ScopeMeter test tool.

Command Syntax:

QS<cr>

Response Syntax:

<acknowledge><cr>1,<setup><cr>

where,
<setup> = string of hexadecimal characters, representing the present setup of the ScopeMeter test tool.

Example:

The following example program demonstrates the use of the QS (QUERY SETUP) and the PS (PROGRAM SETUP) commands. The present setup is queried from test tool and saved to file. The program asks you to change the test tool settings. Then the original setup is read from file and sent back to the test tool.

```
'*****************  Begin example program  *****************

OPEN "COM1:1200,N,8,1,CS,DS,RB2048" FOR RANDOM AS #1
CLS
GOSUB ClearPort          'Clears pending data from port.
PRINT #1, "QS"           'Sends QUERY SETUP query.
GOSUB Acknowledge        'Input acknowledge from test tool.
INPUT #1, N              'Reads <count> (=1).
INPUT #1, SURESP$        'Reads queried data.
OPEN "SETUP1" FOR OUTPUT AS #2
                         'Opens file SETUP1 for data storage.
PRINT #2, SURESP$        'Stores setup data to file SETUP1.
CLOSE #2                 'Closes file SETUP1.

PRINT "Present setup data are stored in the file SETUP1"
PRINT "This setup will now be retrieved from the file and"
PRINT "sent back to the test tool."
PRINT "To see if this works, change the present settings and"
PRINT "verify if the test tool returns to the previous"
PRINT "settings."
PRINT
PRINT "Press any key on the PC keyboard to continue."
SLEEP

OPEN "SETUP1" FOR INPUT AS #2
                         'Opens file SETUP1 for data retrieval.
INPUT #2, SETUP$         'Reads setup data from file and
                         'saves these into SETUP$.
CLOSE #2                 'Close file SETUP1.
PRINT #1, "PS 1," + SETUP$  'Programs ScopeMeter with the
                            'setup data stored in SETUP$.
GOSUB Acknowledge        'Input acknowledge from test tool.
END
'
```

```
'*****************  Acknowledge subroutine  ******************
'Use this subroutine after each command or query sent to the
'ScopeMeter test tool. This routine inputs the acknowledge
'response from the test tool. If the response is non-zero,
'the previous command was not correct or was not correctly
'received by the test tool. Then an error message is
'displayed and the program is aborted.

Acknowledge:
INPUT #1, ACK           'Reads acknowledge from test tool.
IF ACK <> 0 THEN
   PRINT "Error "; ACK; ": ";
   SELECT CASE ACK
     CASE 1
        PRINT "Syntax Error"
     CASE 2
        PRINT "Execution Error"
     CASE 3
        PRINT "Synchronization Error"
     CASE 4
        PRINT "Communication Error"
     CASE IS < 1
        PRINT "Unknown Acknowledge"
     CASE IS > 4
        PRINT "Unknown Acknowledge"
   END SELECT
   PRINT "Program aborted."
   END
END IF
RETURN


'*******  Clears pending data from the RS232 port *********
ClearPort:
   WHILE LOC(1) > 0
     Dummy$ = INPUT$(1, #1)
  WEND
RETURN

'******************  End example program  ******************
```

QUERY WAVEFORM QW

Purpose:

Queries the waveform data and/or the setup data related to the waveform from the ScopeMeter test tool.

Command Syntax:

QW <trace_no>[,V|S]

<trace_no> = 92 to 123, assigned to the following trace sources:

<trace_no>	TRACE SOURCE
92	Max A
93	Min A
94	Max B
95	Min B
96	Max Trend
97	Avg Trend
98	Min Trend
101	INPUT A
102	INPUT B
103	A +/- B
104 to 123	Stored waveform 1 to 20

V Trace values only
S Setup data only.

Response Syntax:

<acknowledge><cr>

If <acknowledge> value 0 is returned, this response is immediately continued with the following data:

If the optional parameter is omitted:

<admin>,<trace>

where <admin> and <trace> are part of the waveform. For detailed descriptions about the waveform structure, refer to Appendix C.

If option V or v (value only) is given:

<trace>

where <trace> is part of the waveform. For detailed descriptions about the waveform structure, refer to Appendix C.

If option S or s is given:

1,<setup><cr>

where,
<setup> = string of hexadecimal characters, representing the setup related to the given <trace no>.

Example:

The following example program copies the waveform measured on INPUT A (WAVEFORM A) to waveform memory 1 (WAVEFORM 1). The waveform from INPUT A is queried (command "QW101") and written back to Memory 1 (command "PW104").
At program start, the subroutine 'ClearPort' reads all pending output data from the ScopeMeter test tool to prevent synchronization errors with following commands.
To test the following program, a waveform must be available on INPUT A, and waveform memory 1 must be displayed to show the copied waveform from INPUT A.

Reset the ScopeMeter (power F4 + ON/OFF).
Set the ScopeMeter in SCOPE mode.
To display WAVEFORM 1, there must have been a waveform saved in this memory. Save the WAVEFORM A to WAVEFORM 1 via SAVE and F1 (MORE SAVE) keys and use the RECALL key to set WAVEFORM 1 on the display. Use the UP/DOWN ASSIGN key (select 'move 1') and then press UP or DOWN to move the waveform to the bottom half of the display.
Change the waveform on INPUT A and start the following program. This waveform will then be copied to WAVEFORM 1 shown on the lower half of the display.

```
'*****************  Begin example program  *****************

OPEN "COM1:1200,N,8,1,CS,DS,RB2048" FOR RANDOM AS #1
CLS                        'Clears the PC screen.
GOSUB ClearPort            'Empties pending data from previous
                           'queries.
DIM WAVE%(512)
PRINT #1, "QW101"          'Queries INPUT A waveform data.
GOSUB Acknowledge          'Input acknowledge from test tool.
INPUT #1, NAM$             'Inputs source name (Channel A).
INPUT #1, YUN$             'Inputs Y_unit (V).
INPUT #1, XUN$             'Inputs X_unit (s).
'The following FOR/NEXT loop inputs four values, namely:
'Y_zero, X_zero (offsets) and Y_resol, X_resol (resolutions)
FOR I = 1 TO 4
   INPUT #1, ADM(I)
NEXT I
INPUT #1, YRNG             'Inputs Y_range.
INPUT #1, XNR              'Inputs number of samples.
PRINT "Vertical offset      : "+STR$(ADM(1))+YUN$
PRINT "Horizontal offset    : "+STR$(ADM(2))+XUN$
PRINT "Vertical resolution  : "+STR$(ADM(3))+YUN$
PRINT "Horizontal resolution: "+STR$(ADM(4))+XUN$
PRINT
PRINT "The display can distinguish "+STR$(YRNG)+" vertical levels"
PRINT
PRINT "The waveform from "+NAM$+" consisting of"
PRINT STR$(XNR)+" samples will now be read....."
SC% = 0
FOR I = 1 TO XNR
   sample$ = INPUT$(1, #1)      'Inputs sample.
   WAVE%(I) = ASC(sample$)      'Fills array with ASCII value
                                '(decimal) of each sample.
   SC% = (SC% + WAVE%(I)) AND 255    'Calculates checksum.
NEXT I
ISC% = ASC(INPUT$(1, #1))       'Inputs trace checksum.
IF ISC% <> SC% THEN PRINT "CHECKSUM ERROR"
PRINT "Waveform read and placed in array."
SC% = 0
PRINT #1, "PW104"           'Program waveform into Memory 1.
GOSUB Acknowledge           'Input acknowledge from test tool.
PRINT "Writing waveform to Memory 1 ...."
FOR I = 1 TO XNR
   SC% = (SC% + WAVE%(I)) AND 255
   sample$ = CHR$(WAVE%(I))
   PRINT #1, sample$;
NEXT I
PRINT #1, CHR$(SC% AND 255) 'Sends checksum.
GOSUB Acknowledge           'Input acknowledge from test tool.
PRINT "Waveform copied to Memory 1."
```

```
CLOSE #1
END
'
```

```
'*****************  Acknowledge subroutine  ******************
'Use this subroutine after each command or query sent to the
'ScopeMeter test tool. This routine inputs the acknowledge
'response from the test tool. If the response is non-zero,
'the previous command was not correct or was not correctly
'received by the test tool. Then an error message is
'displayed and the program is aborted.

Acknowledge:
INPUT #1, ACK           'Reads acknowledge from test tool.
IF ACK <> 0 THEN
   PRINT "Error "; ACK; ": ";
   SELECT CASE ACK
     CASE 1
        PRINT "Syntax Error"
     CASE 2
        PRINT "Execution Error"
     CASE 3
        PRINT "Synchronization Error"
     CASE 4
        PRINT "Communication Error"
     CASE IS < 1
        PRINT "Unknown Acknowledge"
     CASE IS > 4
        PRINT "Unknown Acknowledge"
   END SELECT
   PRINT "Program aborted."
   END
END IF
RETURN


'*******  Clears pending data from the RS232 port *********
ClearPort:
WHILE LOC(1) > 0
  Dummy$ = INPUT$(1, #1)
WEND
RETURN

'******************  End example program  ******************
```

RESET INSTRUMENT RI

Purpose:

Performs the following actions:
- Resets the firmware of the ScopeMeter test tool.
- Clears input and output buffers.

The RI command does not affect the following:
- Waveform, screen, and setup memories.
- Mode of operation (Scope, Meter, or Component Tester).
- RS232 communication settings (e.g. baud rate), to keep the communication alive.

Command Syntax:

RI<cr>

Response Syntax:

<acknowledge><cr>

Note: Wait for at least 2 seconds after the <acknowledge> reply has been received, to let the test tool settle itself before you send the next command.

Example:

The following example resets the test tool and waits for 2 seconds to let the test tool execute the reset and become ready for next commands.
The test tool is queried for the identification data; this data is input and displayed on the PC screen.

```
'*****************  Begin example program  *****************

CLS                       'Clears the PC screen.
OPEN "COM1:1200,N,8,1,CS,DS,RB2048" FOR RANDOM AS #1
PRINT #1, "RI"            'Sends the RESET INSTRUMENT command.
GOSUB Acknowledge         'Input acknowledge from test tool.
SLEEP 2                   'Delay (2 s) necessary after reset.
PRINT #1, "ID"            'Sends IDENTIFICATION query.
GOSUB Acknowledge         'Input acknowledge from test tool.
INPUT #1, IDENT$          'Inputs the queried data.
PRINT IDENT$              'Displays queried data.
CLOSE #1
END


'****************  Acknowledge subroutine  ******************
'Use this subroutine after each command or query sent to the
'ScopeMeter test tool. This routine inputs the acknowledge
'response from the test tool. If the response is non-zero,
'the previous command was not correct or was not correctly
'received by the test tool. Then an error message is
'displayed and the program is aborted.

Acknowledge:
INPUT #1, ACK             'Reads acknowledge from test tool.
IF ACK <> 0 THEN
   PRINT "Error "; ACK; ": ";
   SELECT CASE ACK
     CASE 1
        PRINT "Syntax Error"
     CASE 2
        PRINT "Execution Error"
     CASE 3
        PRINT "Synchronization Error"
     CASE 4
        PRINT "Communication Error"
     CASE IS < 1
        PRINT "Unknown Acknowledge"
     CASE IS > 4
        PRINT "Unknown Acknowledge"
   END SELECT
   PRINT "Program aborted."
   END
END IF
RETURN

'*****************  End example program  *****************
```

RECALL SETUP RS

Purpose:

Recalls an internally stored setup. This setup must have been stored in the ScopeMeter test tool manually or with the SS (Save Setup) command.

Command Syntax:

RS <setup reg><cr>

where,

<setup reg> =	1	to	40	for stored setups
	61	to	70	for stored screen setups
	92	to	99	for 'live traces' setups
	101	to	103	for 'live traces' setups
	104	to	123	for stored waveform setups

Response Syntax:

<acknowledge><cr>

Example:

The following example program saves the present setup in setup memory 8. You are requested to change the present settings. Then the original settings are recalled from setup memory 8 and made the actual setting.

```
'*****************  Begin example program  *****************

CLS                      'Clears the PC screen.
OPEN "COM1:1200,N,8,1,CS,DS,RB2048" FOR RANDOM AS #1
PRINT #1, "SS8"          'Sends SAVE SETUP command.
                         'Setup saved in setup memory 8.
GOSUB Acknowledge        'Input acknowledge from test tool
PRINT "The present setup data are stored in setup memory 8."
PRINT "The remainder of this program will restore these."
PRINT "To test if this works, change the present settings"
PRINT "and verify if the test tool returns to the original"
PRINT "settings after continuing the program."
PRINT
PRINT "Press any key on the PC keyboard to continue."
SLEEP
PRINT #1, "RS8"          'Sends RECALL SETUP command.
                         'Setup recalled from register 8.
GOSUB Acknowledge        'Input acknowledge from test tool.
PRINT
PRINT "Original settings restored"
CLOSE #1
END
'
```

```
'*****************  Acknowledge subroutine  ******************
'Use this subroutine after each command or query sent to the
'ScopeMeter test tool. This routine inputs the acknowledge
'response from the test tool. If the response is non-zero,
'the previous command was not correct or was not correctly
'received by the test tool. Then an error message is
'displayed and the program is aborted.

Acknowledge:
INPUT #1, ACK           'Reads acknowledge from test tool.
IF ACK <> 0 THEN
   PRINT "Error "; ACK; ": ";
   SELECT CASE ACK
     CASE 1
        PRINT "Syntax Error"
     CASE 2
        PRINT "Execution Error"
     CASE 3
        PRINT "Synchronization Error"
     CASE 4
        PRINT "Communication Error"
     CASE IS < 1
        PRINT "Unknown Acknowledge"
     CASE IS > 4
        PRINT "Unknown Acknowledge"
   END SELECT
   PRINT "Program aborted."
   END
END IF
RETURN

'******************  End example program  ******************
```

SAVE SETUP SS

Purpose:

Saves the present acquisition settings in one of the internal setup registers.

Command Syntax:

SS <setup reg><cr>

where,

<setup reg> =	1 to 40	for stored setups
	101 to 103	for 'live traces' setups
	104 to 123	for stored waveform setups

Response Syntax:

<acknowledge><cr>

Example:

The following example program saves the present setup in setup memory 8. You are requested to change the present settings. Then the original settings are recalled from setup memory 8 and made the actual.

```
'*****************  Begin example program  *****************

CLS                      'Clears the PC screen.
OPEN "COM1:1200,N,8,1,CS,DS,RB2048" FOR RANDOM AS #1
PRINT #1, "SS8"          'Sends SAVE SETUP command.
                         'Setup saved in setup memory 8.
GOSUB Acknowledge        'Input acknowledge from test tool.
PRINT "The present setup data are stored in setup memory 8."
PRINT "The remainder of this program will restore these."
PRINT "To test if this works, change the present settings"
PRINT "and verify if the test tool returns to the original"
PRINT "settings after continuing the program."
PRINT
PRINT "Press any key on the PC keyboard to continue."
SLEEP
PRINT #1, "RS8"          'Sends RECALL SETUP command.
                         'Setup recalled from register 8.
GOSUB Acknowledge        'Input acknowledge from test tool.
PRINT
PRINT "Original settings restored"
CLOSE #1
END
'
```

```
'*****************  Acknowledge subroutine  ******************
'Use this subroutine after each command or query sent to the
'ScopeMeter test tool. This routine inputs the acknowledge
'response from the test tool. If the response is non-zero,
'the previous command was not correct or was not correctly
'received by the test tool. Then an error message is
'displayed and the program is aborted.

Acknowledge:
INPUT #1, ACK           'Reads acknowledge from test tool.
IF ACK <> 0 THEN
   PRINT "Error "; ACK; ": ";
   SELECT CASE ACK
     CASE 1
        PRINT "Syntax Error"
     CASE 2
        PRINT "Execution Error"
     CASE 3
        PRINT "Synchronization Error"
     CASE 4
        PRINT "Communication Error"
     CASE IS < 1
        PRINT "Unknown Acknowledge"
     CASE IS > 4
        PRINT "Unknown Acknowledge"
   END SELECT
   PRINT "Program aborted."
   END
END IF
RETURN

'******************  End example program  ******************
```

STATUS QUERY ST

Purpose:

Queries the error status of the ScopeMeter test tool. This is a 16-bit word, presented as an integer value, where each bit represents the Boolean value of a related error event. After the reply or after a RI (Reset Instrument) command, the value is reset to zero.
A complete description of the status word is given in Appendix B.

Command Syntax:

ST<cr>

Response Syntax:

<acknowledge><cr>

If <acknowledge> value 0 is returned, this response is immediately continued with the following data:

<status><cr>

where,
<status> = integer value 0 to 32767

Example:

The following example program sends a wrong command to the test tool to test the Acknowledge subroutine and to check the status returned from the ST query.
The acknowledge subroutine contains a GOSUB Status.display to input the status data from the test tool when the acknowledge response is non-zero (ACK <> 0).

```
'*****************  Begin example program  *****************

CLS                          'Clears the PC screen.
OPEN "COM1:1200,N,8,1,CS,DS,RB2048" FOR RANDOM AS #1
PRINT #1, "PC12345,N,8,1"  'Sends a baud rate value that is
                         '   out of range for the test tool.
GOSUB Acknowledge.Status   'Input acknowledge from test tool
                           'and the status value if the
                           'acknowledge value is non-zero.
END


'*************  Acknowledge + Status subroutine  ***********
'This subroutine inputs the acknowledge value from the
'ScopeMeter test tool. If the acknowledge value is non-zero,
'the ST query is used to get further status information from
'the test tool with respect to the error.
'In case of an error the program is aborted.

Acknowledge.Status:
INPUT #1, ACK           'Reads acknowledge from test tool.
IF ACK <> 0 THEN
   PRINT "Error "; ACK; ": ";
   SELECT CASE ACK
     CASE 1
        PRINT "Syntax Error"
     CASE 2
        PRINT "Execution Error"
     CASE 3
        PRINT "Synchronization Error"
     CASE 4
        PRINT "Communication Error"
     CASE IS < 1
        PRINT "Unknown Acknowledge"
     CASE IS > 4
        PRINT "Unknown Acknowledge"
   END SELECT
   GOSUB Status.display         'Further specifies the error.
   PRINT "Program aborted."
   END
END IF
RETURN
'
```

```
'**************  Displays test tool status *****************

'This subroutine gives you further information if the
'acknowledge reply from the ScopeMeter test tool is non-zero.

Status.display:
PRINT #1, "ST"            'Sends the STATUS query.
GOSUB Acknowledge.Status  'Inputs acknowledge from test tool.
INPUT #1, STAT            'Inputs status value.
PRINT "Status " + STR$(STAT) + ": ";
SELECT CASE STAT
  CASE 0
     PRINT "No error"
  CASE 1
     PRINT "Unknown header"
  CASE 2
     PRINT "Data format of body is wrong"
  CASE 4
     PRINT "Data out of range"
  CASE 8
     PRINT "Invalid instruction in present mode"
  CASE 16
     PRINT "Called function not implemented"
  CASE 32
     PRINT "Invalid number of parameters"
  CASE 64
     PRINT "Wrong number of data bits"
  CASE 512
     PRINT "Conflicting instrument settings"
  CASE 16384
     PRINT "Checksum error"
END SELECT
RETURN

'******************  End example program  ******************
```

TRIGGER ACQUISITION TA

Purpose:

Triggers an acquisition. This command acts as a hardware trigger to start a new acquisition.
In SINGLE shot acquisition mode the trigger system must have been armed with the AT (Arm Trigger) command.
When the ScopeMeter test tool is in the METER Mode, <acknowledge> value 2 is returned to indicate Execution Error.

Command Syntax:

TA<cr>

Response Syntax:

<acknowledge><cr>

Example:

The following example program demonstrates the use of the TA command. The TA command is used when the computer program must synchronize acquisition timing with other computer controlled equipment. (This is not shown in this program.)

To see the effect of the TA command used in the following program example, do the following:
- Connect a repetitive signal to INPUT A.
- Set the ScopeMeter test tool in SCOPE mode.
- Press the AUTO SET key. The waveform appears on the display.
- Press the TRIGGER key and select MORE TRIGGER (F1 key).
- Select TRACE START and set it to 'wait for trigger'.
- Select TRIGGER LEVEL, set it to 'manual'.
- Use the UP key to set the trigger level above the top of the signal level. Acquisition will not trigger anymore.
- Press the TIME ns key to increase the time base speed. The waveform image does not change because there is no trigger.
- Run the following program, which sends a TA (TRIGGER ACQUISITION) command to the ScopeMeter test tool. The waveform will then change in accordance with the new time base setting.

```
'*****************  Begin example program  *****************
CLS                        'Clears the PC screen.
OPEN "COM1:1200,N,8,1,CS,DS,RB2048" FOR RANDOM AS #1
PRINT #1, "TA"             'Sends TRIGGER ACQUISITION command.
GOSUB Acknowledge          'Input acknowledge from test tool.
END


'*****************  Acknowledge subroutine  ******************
'Use this subroutine after each command or query sent to the
'ScopeMeter test tool. This routine inputs the acknowledge
'response from the test tool. If the response is non-zero,
'the previous command was not correct or was not correctly
'received by the test tool. Then an error message is
'displayed and the program is aborted.

Acknowledge:
INPUT #1, ACK              'Reads acknowledge from test tool.
IF ACK <> 0 THEN
   PRINT "Error "; ACK; ": ";
   SELECT CASE ACK
     CASE 1
        PRINT "Syntax Error"
     CASE 2
        PRINT "Execution Error"
     CASE 3
        PRINT "Synchronization Error"
     CASE 4
        PRINT "Communication Error"
     CASE IS < 1
        PRINT "Unknown Acknowledge"
     CASE IS > 4
        PRINT "Unknown Acknowledge"
   END SELECT
   PRINT "Program aborted."
   END
END IF
RETURN

'******************  End example program  ******************
```

VIEW SCREEN VS

Purpose:

Restores a screen saved by keypad operation on the test tool display. You can use this command in combination with the QP (Query Print) command to transfer saved screens to a computer. The ScopeMeter test tool Users Manual gives information on how to save screens.

Command Syntax:

VS <view screen><cr>

where,
<view screen> = 0 to 10

0	Exit View Screen mode
1	View Screen 1
2	View Screen 2
3	View Screen 3
4	View Screen 4
5	View Screen 5
6	View Screen 6
7	View Screen 7
8	View Screen 8
9	View Screen 9
10	View Screen 10

The following commands will cause View Screen mode to exit: RI, DS, PS, RS, AT, and TA.
The following commands will use the settings related to the actual screen: QS, AS, and SS.
The QM command will always return the readings related to the actual screen.

Command Syntax:

<acknowledge><cr>

Example:

The following example program recalls the screen saved previously in screen memory 5.

```
'*****************  Begin example program  *****************

CLS                      'Clears the PC screen
OPEN "COM1:1200,N,8,1,CS,DS,RB2048" FOR RANDOM AS #1
PRINT #1,"VS 5"           'Sends VIEW SCREEN 5 command.
GOSUB Acknowledge         'Input acknowledge from test tool.
PRINT "The waveform from VIEW SCREEN 5 is now displayed."
PRINT
PRINT "You can use the example program for the QP command"
PRINT "to copy the screen image to an EPSON FX/LQ compatible"
PRINT "printer."
END


'*****************  Acknowledge subroutine  ******************
'Use this subroutine after each command or query sent to the
'ScopeMeter test tool. This routine inputs the acknowledge
'response from the test tool. If the response is non-zero,
'the previous command was not correct or was not correctly
'received by the test tool. Then an error message is
'displayed and the program is aborted.

Acknowledge:
INPUT #1, ACK            'Reads acknowledge from test tool.
IF ACK <> 0 THEN
   PRINT "Error "; ACK; ": ";
   SELECT CASE ACK
     CASE 1
        PRINT "Syntax Error"
     CASE 2
        PRINT "Execution Error"
     CASE 3
        PRINT "Synchronization Error"
     CASE 4
        PRINT "Communication Error"
     CASE IS < 1
        PRINT "Unknown Acknowledge"
     CASE IS > 4
        PRINT "Unknown Acknowledge"
   END SELECT
   PRINT "Program aborted."
   END
END IF
RETURN

'******************  End example program  ******************
```

APPENDIX A ACKNOWLEDGE DATA

The ScopeMeter test tool returns an <acknowledge> reply after each command or query. The value indicates correct or incorrect operation. You always must read this reply to check for the correct operation and to achieve synchronization between your program and the RS232 interface of the ScopeMeter test tool.

<acknowledge>

VALUE	MEANING
0	No Error
1	Syntax Error (see Note)
2	Execution Error (see Note)
3	Synchronization Error
4	Communication Error

Note: The ST query may give you additional information.

When the ScopeMeter test tool detects an error during the execution of a command, it sends the corresponding <acknowledge> reply, terminates further execution of the command and will be ready to accept a new command.

Syntax Error

Returned when the command is not understood by the ScopeMeter test tool for one of the following reasons :

- Unknown header
- Wrong instructions
- Data format of body is wrong, e.g. alpha characters when decimal data is needed.

Execution Error

Returned when internal processing is not possible because of one of the following reasons:

- Data out of range
- Conflicting instrument settings

Synchronization Error

Returned when the ScopeMeter test tool receives data while it does not expect any data. This can occur as follows:

- The ScopeMeter test tool receives a new command while a previous command or query is not yet completely executed. You can prevent this error by doing the following:

  1. Read the <acknowledge> reply after each command or query.
  2. If this <acknowledge> is zero and if a query was sent to the ScopeMeter test tool, read all available response data.

Communication Error

Any framing, parity or overrun error detected on the received data will cause Communication Error.

APPENDIX B STATUS DATA

The Status word returned from the ST query gives you extra information when you have received a non-zero <acknowledge> reply.
The Status word is a 16-bit binary word where each bit set true represents an error event with a decimal value determined by the bit position. (See the following table.)

When more than one bit is set true in the status word, the response from the ST query will be the sum of the decimal values of the individual bits.

Example:

<status> = 34 This equals 32 + 2
2 = Wrong parameter data format
32 = Invalid number of parameters

BIT	DECIMAL VALUE	EVENT DESCRIPTION	<acknowledge> VALUE
0	1	Illegal command	1
1	2	Wrong parameter data format	1
2	4	Parameter out of range	1 or 2
3	8	Instruction not valid in present state	1
4	16	Called function not implemented	2
5	32	Invalid number of parameters	2
6	64	Wrong number of data bits	2
9	512	Conflicting instrument settings	2
14	16384	Checksum error	2

Remarks:

1. A bit in the status word is set when the corresponding error event occurs.
2. Bits do not affect each other.
3. New error events will 'accumulate' in the status word. This means existing bits remain set.

The status word is cleared (all bits reset) as follows:

1. After the response (the status word) from the ST query has been read.
2. After the RI (Reset Instrument) command.

APPENDIX C WAVEFORM DATA

The waveform data that is received from the QW (Query Waveform) query, consists of the following data.

<admin>,<trace>

where,
<admin> = <trace name>,<Y-unit>,<X-unit>,<Y-zero>,<X-zero>,
<Y-resol>,<X-resol>,<Y-range>,<no_of_samples>

where,

<trace name>	= Name of the source from where the trace is read.
<Y_unit>	= Unit suffix for the Y value of the sample.
<X_unit>	= Unit suffix for the X value of the sample.
<Y_zero>	= Value of the vertical shift (set with the MOVE key).
<X_zero>	= Value of the TRIGGER delay setting in (menu TIME DELAY).
<Y_resol>	= Vertical resolution (unit = <Y_unit>).
<X_resol>	= Horizontal resolution (unit = <X_unit>).
<Y_range>	= Number of vertical levels possible (fixed to 255).
<no_of_samples>	= Number of X-samples (fixed to 512).

<trace> =
<sample 1><sample 2><sample 3>.....<sample 512><checksum>

where each sample is an 8-bit byte, the decimal value of which indicates the vertical position on the display (positions from 0 to 255 inside <Y_range>, see above) of the ScopeMeter test tool. The sample number relates to the horizontal position in the trace (positions from 0 to 511) of which about 240 will fit on the display.

APPENDIX D ASCII 7-BIT CODES

Hexadecimal value
| ASCII character
| | Decimal value
| | |

00	NUL	0	20	SP	32	40	@	64	60	`	96
01	SOH	1	21	!	33	41	A	65	61	a	97
02	STX	2	22	"	34	42	B	66	62	b	98
03	ETX	3	23	#	35	43	C	67	63	c	99
04	EOT	4	24	$	36	44	D	68	64	d	100
05	ENQ	5	25	%	37	45	E	69	65	e	101
06	ACK	6	26	&	38	46	F	70	66	f	102
07	BEL	7	27	'	39	47	G	71	67	g	103
08	BS	8	28	(	40	48	H	72	68	h	104
09	HT	9	29	)	41	49	I	73	69	i	105
0A	LF	10	2A	*	42	4A	J	74	6A	j	106
0B	VT	11	2B	+	43	4B	K	75	6B	k	107
0C	FF	12	2C	,	44	4C	L	76	6C	l	108
0D	CR	13	2D	-	45	4D	M	77	6D	m	109
0E	SO	14	2E	.	46	4E	N	78	6E	n	110
0F	SI	15	2F	/	47	4F	O	79	6F	o	111
10	DLE	16	30	0	48	50	P	80	70	p	112
11	XON	17	31	1	49	51	Q	81	71	q	113
12	DC2	18	32	2	50	52	R	82	72	r	114
13	XOF	19	33	3	51	53	S	83	73	s	115
14	DC4	20	34	4	52	54	T	84	74	t	116
15	NAK	21	35	5	53	55	U	85	75	u	117
16	SYN	22	36	6	54	56	V	86	76	v	118
17	ETB	23	37	7	55	57	W	87	77	w	119
18	CAN	24	38	8	56	58	X	88	78	x	120
19	EM	25	39	9	57	59	Y	89	79	y	121
1A	SUB	26	3A	:	58	5A	Z	90	7A	z	122
1B	ESC	27	3B	;	59	5B	[	91	7B	{	123
1C	FS	28	3C	<	60	5C	\	92	7C	|	124
1D	GS	29	3D	=	61	5D	]	93	7D	}	125
1E	RS	30	3E	>	62	5E	^	94	7E	~	126
1F	US	31	3F	?	63	5F	_	95	7F		127

Hexadecimal value
 | ASCII character
 | | Decimal value
 | | |

80	€	128	A0		160	C0	À	192	E0	à	224
81		129	A1	¡	161	C1	Á	193	E1	á	225
82	‚	130	A2	¢	162	C2	Â	194	E2	â	226
83	ƒ	131	A3	£	163	C3	Ã	195	E3	ã	227
84	„	132	A4	¤	164	C4	Ä	196	E4	ä	228
85	…	133	A5	¥	165	C5	Å	197	E5	å	229
86	†	134	A6	¦	166	C6	Æ	198	E6	æ	230
87	‡	135	A7	§	167	C7	Ç	199	E7	ç	231
88	ˆ	136	A8	¨	168	C8	È	200	E8	è	232
89	‰	137	A9	©	169	C9	É	201	E9	é	233
8A	Š	138	AA	ª	170	CA	Ê	202	EA	ê	234
8B	‹	139	AB	«	171	CB	Ë	203	EB	ë	235
8C	Œ	140	AC	¬	172	CC	Ì	204	EC	ì	236
8D		141	AD	-	173	CD	Í	205	ED	í	237
8E	Ž	142	AE	®	174	CE	Î	206	EE	î	238
8F		143	AF	¯	175	CF	Ï	207	EF	ï	239
90		144	B0	°	176	D0	Ð	208	F0	ð	240
91	‘	145	B1	±	177	D1	Ñ	209	F1	ñ	241
92	’	146	B2	²	178	D2	Ò	210	F2	ò	242
93	“	147	B3	³	179	D3	Ó	211	F3	ó	243
94	”	148	B4	´	180	D4	Ô	212	F4	ô	244
95	•	149	B5	µ	181	D5	Õ	213	F5	õ	245
96	–	150	B6	¶	182	D6	Ö	214	F6	ö	246
97	—	151	B7	·	183	D7	×	215	F7	÷	247
98	˜	152	B8	¸	184	D8	Ø	216	F8	ø	248
99	™	153	B9	¹	185	D9	Ø	217	F9	ù	249
9A	š	154	BA	º	186	DA	Ú	218	FA	ú	250
9B	›	155	BB	»	187	DB	Û	219	FB	û	251
9C	œ	156	BC	¼	188	DC	Ü	220	FC	ü	252
9D		157	BD	½	189	DD	Ý	221	FD	ý	253
9E	ž	158	BE	¾	190	DE	Þ	222	FE	þ	254
9F	Ÿ	159	BF	¿	191	DF	ß	223	FF		255