HEWLETT PACKARD

HP 54600A
and
HP 54601A
Oscilloscopes

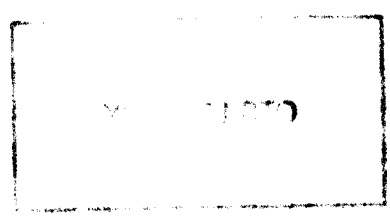
Programmer's Guide for QuickC and" QuickBASIC

HAS

Programmer's Guide for QuickC and QuickBASIC

Publication Number 54600-90912
First Edition July 1991

HP 54600A and HP 54601A Oscilloscopes



Contents

Introduction

Today's research and manufacturing engineers are being pressured to improve productivity while reducing costs. To remain competitive in the fast-paced electronics marketplace, engineers must reduce product design cycles. Testing the finished product must be accomplished in the shortest possible time and in the most repeatable manner.

The low cost of the personal computer, teamed with the power of custom software, allow engineers to use oscilloscopes to speed the development and analysis of their circuitry.

About This Book

This guide describes how an HP Vectra or an IBM PC/XT/AT personal computer is programmed to converse with the HP 54600A series of oscilloscopes via the HP-IB IEEE 488.2 interface. A sample program is provided to get you started and to overcome some of the hurdles a first-time programmer faces. The guide is not intended to be all inclusive. That is, it is not a tutorial on HP-IB, programming a personal computer, or the full HP-IB capability of the oscilloscope.

The HP-IB interface card used is the HP 82335. The earlier version card (HP 82990A) can be used if the most recent HP 82335 drivers are installed. If the most recent drivers are not used, some commands like the IOENTERAB, will not function.

A sample program is written in both Microsoft QuickC (version 2.5) and Microsoft QuickBASIC (version 4.5) . Each program is modular, with explanations about each module.

The computer configuration must be set up as specified in both the HP-IB manual ("Using the HP-IB Interface and Command Library') and the Microsoft Language manuals. All drivers and libraries must be installed as specified in those manuals.

If you are familiar with programming oscilloscopes over the IEEE 488 bus, you may wish to skip to the "Getting Started" Section. The next two sections discuss introductory programming material and the three fundamental parts of a program.

References

This book provides more information than the standard documentation for the HP 54600A series digitizing oscilloscopes. It is not all-inclusive and is meant to be used with knowledge contained in the following references:

HP-IB Reference Manuals --*Tutorial Description of the Hewlett Packard Interface Bus HP* Part Number 5021-1927 and *Using the HP-B Interface and Command Library HP* Part , Number 82335-90001.

HP 54600A and HP 54601A Digitizing Oscilloscope *Programmer's* Guide HP Part Number 54600-90909.

Microsoft QuickC Manuals: C *for Yourself* (version 2.5), *Up and Running* (version 2.5), and *Tool Kit* (version 2.5).

Microsoft QuickBASIC Manuals for IBM Personal Computers and Compatibles: *Learning to Use Microsoft QuickBASIC* (version 4.5), *Programming in BASIC* (version 4.5), and *Microsoft QuickBASIC BASIC Language Reference* (version 4.5) .

Programming the Oscilloscope

Typically, there are several basic operations that an engineer will want a computer and oscilloscope to do in a programming environment:

- set up the oscilloscope,
- make measurements,
- get data (measurements, waveform data, configurations) from the oscilloscope for analysis by the computer ,and
- send information (waveform data or configurations) to the oscilloscope.

Very complicated tasks can be accomplished using the flexibility of the computer and the combination of these four basic functions.

To be successful in merging the oscilloscope and computer, you first need to make the measurement using the front panel. The process over the bus will be much the same as that from the front panel.

Many front panel and programming operations are similar:

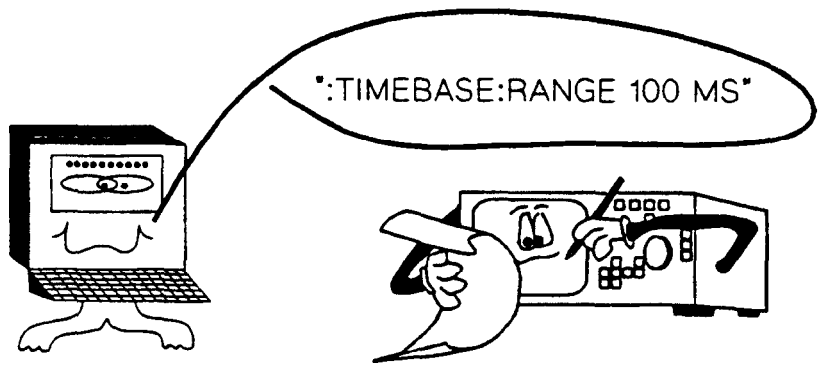
Front Panel:	push a key, use the menus
Programming:	send a command, use the subsystems

Several front panel and programming operations are different:

Front Panel:	set V/div, see waveform
Programming:	send the full scale range, get data that can be plotted on computer screen or converted to volts/time

Computers communicate with the oscilloscope by sending and receiving messages over the bus using input/output (I/O) statements provided in the programming language.

COMMUNICATING WITH THE SCOPE



Messages for programming the oscilloscope appear as ASCII character strings embedded inside I/O program statements. Each message consists of a device address, a program message, and a terminator. The address uniquely identifies one of the instruments on the bus while the program message specifies the action to be taken by that instrument.

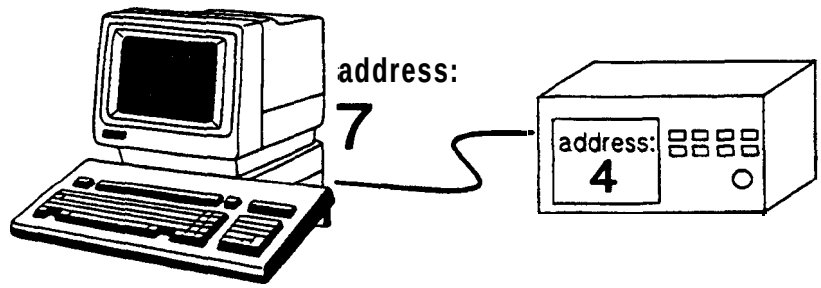
Device Address

The address is composed of two parts:

- Interface Select Code (ISC)
- instrument address

Each HP-IB interface card has its own ISC address set on the card. This address is used to direct commands and communications to the proper interface. Each instrument on the HP-IB bus must have a unique address that is between decimal 00 and 30. The instrument address must be set in the HP-IB menu from the front panel, as described in "Getting Started".

ADDRESSING



DEVICE ADDRESS = (Interface * 100) + instrument

= (7 * 100) + 4 = 704

QuickBASIC:	QuickC:
Isc& = 7	**DEFINE Isc 7L**
Scope& = 704	**DEFINE SCOPE** 704L

(must be long variables)

Message Basics: Command/Query

There are two basic forms of messages: commands and queries.

The format of the command and query are the same, but the query has a question mark (?) attached to the command message.

The function of the command and query are vastly different.

- Commands are used to set up the oscilloscope or to specify that an action be taken.
- Queries are used to find out how the oscilloscope is set up or to get results from the oscilloscope.

A query causes the oscilloscope to put data into an output buffer of the oscilloscope. This data must be read from the oscilloscope into the computer using an input statement. If the data is **not** read from the oscilloscope, and a new command or query is sent to the oscilloscope, the error message "QUERY INTERRUPTED" is displayed.

Examples of Commands:

```
":TIMEBASE:RANGE 5E-6"
```

Sets the timebase range to 5 us (500 ms/div) .

```
":AUTOSCALE"
```

Automatically sets up the oscilloscope to display a signal.

Examples of Queries:

```
":TIMEBASE:RANGE?"
```

Queries the oscilloscope for its current time base range. The oscilloscope puts the result into the output buffer, and the result must be read from the oscilloscope using an input statement.

```
":MEASURE:VPP?"
```

Queries the oscilloscope for a volt peak-to-peak measurement. The oscilloscope puts the result into the output buffer, and the result must be read from the oscilloscope using an input statement.

Three Message Types

There are three basic message types defined by IEEE 488.2:

- common commands,
- root level commands, and
- subsystem commands.

Common Commands

The IEEE 488.2 Standard designates a set of commands that all instruments must have to guarantee that all instruments have a minimum set of capabilities. All of these commands are prefaced with an asterisk. The syntax of the common command is:

*** <message > <separator > < data > < terminator >**

Examples of Common Commands:

• *RST	**Resets oscilloscope to default configuration**
• *CLS	Clears status data structures
***OPC?**	**A query that requests status of operation complete register.**
• *ESE	**Sets event status enable register to 1.**

Root Level Commands

The root level commands control many of the basic operations of the oscilloscope. The syntax of the root level command is:

```
:<message><separator><data><terminator>
```

Examples of Root bevel Commands:

```
":AUTOSCALE"
```

Performs autoscale.

```
":DIGITIZE CHAN1"
```

Digitizes channel 1.

Subsystem Commands

Configuring the oscilloscope from the front panel usually means finding and setting the function in the appropriate menu. Analogous to the front panel menus are the corresponding HP-IB subsystems. Similar functions are grouped together under a subsystem. For example, the time base range, delay, and reference are all grouped together under the TIMEBASE subsystem.

The syntax of the subsystem command is:

```
:<subsystem>:<function><separator><data><terminator>
```

Examples of Subsystem Commands:

```
":CHAN1:RANGE 20MV"
```

Sets channel 1 sensitivity to 20 mv (2 mv/div) .

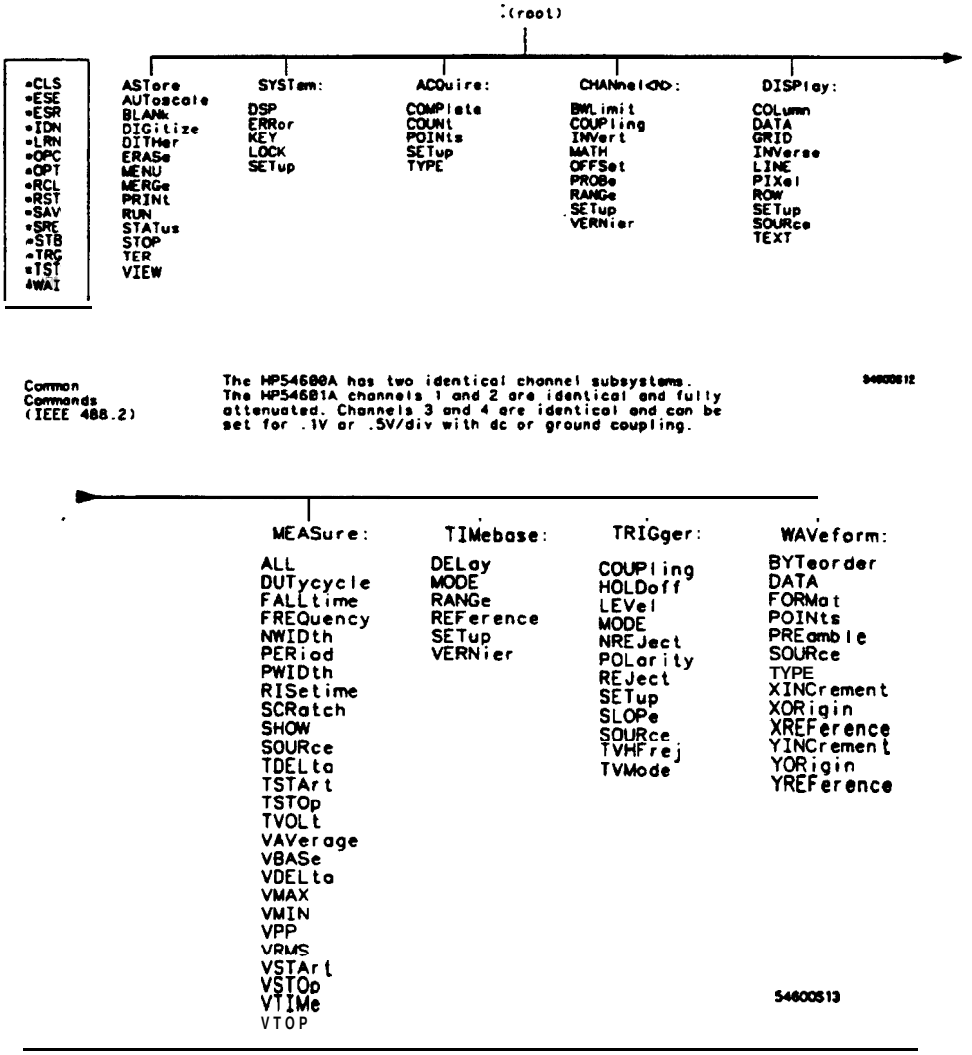
```
":TIMEBASE:RANGE 200US "
```

Sets timebase range to 200 us (20 us/div).

In the first example, the CHANNEL subsystem is specified with the vertical sensitivity being set. In the second example, the TIMEBASE subsystem is specified with the horizontal sensitivity being set. The RANGE function is common to both subsystems, but the oscilloscope will change the appropriate value based on the subsystem specified.

Command Tree and Tree Traversal Rules

The parser is the **software** inside the oscilloscope that analyzes each command. Understanding the use of the colon, subsystem **headers,** and terminators by the parser will make programming easier and more efficient. The organization of the oscilloscope commands is best explained through the command tree shown below.



The common commands are commands which are independent of the tree and do not affect the position of the parser within the tree. Root level commands are commands that reside at the root of the command tree. They are prefaced by a colon that puts the parser at the root level. Finally, the subsystem commands are commands that are grouped together under a common node of the tree, such as the TIMEBASE commands.

The following rules apply to traversing the command tree:

- a leading colon places the parser at the root of the tree.
- a program message terminator places the parser at the root of the tree.
- a subsystem command places the instrument in that subsystem until a leading colon or a terminator is found.

Format of Statements

For ease of explaining the sample programs, each statement sent to the oscilloscope will be a single command, written in long form. It is recommended that the short form of the command be used and that the commands be concatenated to improve throughput.

For example, instead of sending:

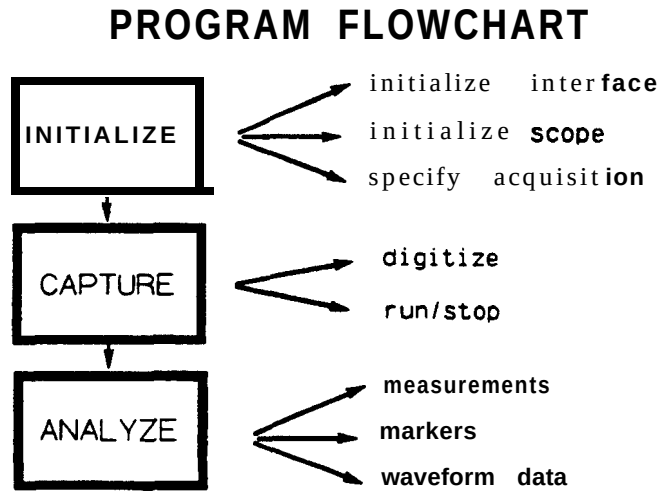
```
":TIMEBASE:RANGE 200MS"
":TIMEBASE:DELAY 100MS",
```

send:

```
":TIM:RANG 200MS;DEL 100MS".
```

Note that the leading colon places the parser at the root level. The command TIM enters the timebase subsystem and two timebase commands can be sent. When an end-of-line terminator or another colon is encountered, the parser is reset to the root level.

Fundamental Parts of a Program



Initialize

First, initialize the HP-IB interface and oscilloscope. This step ensures consistent, repeatable performance of the program. Without these initialization procedures, a program may run one time, but not function properly the next time. This may be due to some setting change made either from the front panel or HP-IB which drastically change the configuration of the oscilloscope.

Second, configure the oscilloscope. The timebase, channel, and trigger systems are set for the desired measurement.

Finally, specify the ACQUIRE subsystem. From the front panel, the only setting that can be made is the mode (such as normal, average, or envelope). But over the bus, you can set the number of points to be acquired and the completion criteria -- both impact throughput.

Capture

Even though the oscilloscope has been properly initialized and set up, measurements cannot yet be made. It must be noted that when the oscilloscope is responding to HP-IB commands, it is in an interrupt mode, and the data acquisition process is being held off. Changes to the oscilloscope's configuration invalidate previously collected data. As a result, if a measurement is requested immediately, there may not be sufficient waveform data available to analyze. Therefore, it is recommended that the next program step is to capture the data using the DIGITIZE command.

The DIGITIZE command captures data as specified by you. It is a macro that first clears the waveform data buffers and then starts the acquisition process. Acquisition continues until the criteria, such as number of averages, completion criteria, and number of points, is satisfied. Once the criteria is satisfied, the acquisition process stops. The signal is displayed on the oscilloscope, and the captured data can be measured, stored in memory, or transferred to the computer for analysis. Additional commands are stored in a buffer of the oscilloscope until the DIGITIZE is completed.

An alternative to the DIGITIZE command is to let the oscilloscope run with a wait in the program before doing any measurements. This method is not recommended because the length of the wait is an indeterminant value. The DIGITIZE command gives confidence to the measurement by ensuring complete data capture. In addition, the DIGITIZE stops the acquisition process, so all measurements are made on the data being displayed, not on a continually changing set of data.

Analyze

Once the data has been captured, measurements can be made. The measurements can include the IEEE standard parametric measurements (such as Vpp, Frequency, + width) or the positioning and reading of voltage and time markers. The waveform data can be transferred from the oscilloscope into the computer for special analysis, if desired.

Getting Started

Before any communication can occur between the computer and oscilloscope, the oscilloscope must be properly configured to talk with the computer over the bus. This MUST be done from the front panel because the HP 54600A series of oscilloscopes are configured to talk to a printer and not to a computer.

To configure the HP 54600A series oscilloscope:

1 Prom the front panel, select the Print/Utility menu.

2 Select the HP-IB menu.

2 Set the mode to connect to "computer". Set the address as desired (7 is used in this guide).

Note: default setting is connect to "HP Print" and address 1.

4 Connect a signal to the oscilloscope:

This sample program assumes that a signal is available on channel 1. You should use the 10:1 probe that was shipped with the oscilloscope and connect the AC signal on the front panel to channel 1 input.

Links to HP-IB Drivers

To run programs in either QuickC or QuickBASIC, you must link to various libraries and include files. The HP-IB Command Library disks contain an INSTALL program that copies the C Library files or the BASIC Library files to your system for you. The location of these files will depend on your computer configuration and where you specify the files to be installed. Refer to your HP-IB Interface manual for details.

Also make sure QuickC or QuickBASIC is installed properly. Refer to your Microsoft programming manual for details.

Microsoft QuickBASIC Library

If writing in Microsoft QuickBASIC, you must link to the HP-IB library file qbhpib.qlb, with the path for your system.
For example:

```
qb /l c:\hpib\qbhpib.qlb
```

Microsoft QuickC Libraries

If writing in Microsoft QuickC, you must link to the HP-IB library file clhpib.lib, with the path for your system. To do this, you must first enter the Options menu and turn full menus on. Leave the Options menu and enter the Make menu. Choose Set Program List and link your source program to the HP-IB library.
For example, with the path for your system, link to:

```
c:\qc25\lib\clhpib.lib
```

For complete instructions on this linking process, see the Microsoft "Up and Running" Manual ("Compiling and Linking" section).

In the sample QuickC program, the graph function draws the oscilloscope's signal on the PC display. If you use this function, you must include GRAPHICS.LIB in the combined libraries while installing QuickC. If you did not include it, link explicitly to the graphics library GRAPHICS.LIB in the Make menu of the QuickC environment.

Another Microsoft QuickC setting that may need to be reviewed is the size of the stack. The stack size is set to a default value of 2048. If your program has a function that has a large local array, this default stack setting does not provide sufficient space, resulting in a stack overflow problem. To reset the size of the stack, first select the Options menu. Next enter the Make dialog box and there select the "Linker Flags". Set the stack size to handle your local variables. . For example, in the sample program, the integrate function declares a real array of 4000 for the integral values, hence the stack size was set at 36,000.

Sample Program Structure

The sample program in this guide is written in both QuickC and QuickBASIC. The sample program follows the format of initialize, capture and analyze as discussed in the "Three Fundamental Parts of a Program" section. The programs are written in modular form to allow detailed discussion of each function. You can write a custom program by selecting and modifying these modules. Also, refer to the disk supplied with the *HP 54600A Programmer's* Guide. The programs on the disk are very similar to the sample program in this guide.

The sample program first initializes the oscilloscope and HP-IB interface and then acquires a signal. The program then performs some parametric measurements, transfers the waveform data from the oscilloscope to the PC, and saves the waveform data to the PC disk. The program then retrieves the waveform data from the PC disk, draws the signal on the PC display, calculates the integral of the signal, and draws the integral on the PC display.

Both the QuickC and QuickBASIC sample programs contain the following fundamental modules:

main program:	Define global variables and constants; specify include files; call the various functions.
initialize:	Initialize HP-IB and oscilloscope; setup oscilloscope and acquire subsystem.
capture-waveform:	Digitize waveform to capture data.
measure:	Perform simple parametric measurement
get waveform:	Bring waveform data and voltage/timing information (preamble) into computer.
save-waveform:	Store the waveform data and preamble information to a PC disk file.
retrieve-waveform:	Retrieve waveform data and preamble information from a PC disk fii.
graph:	Plot the waveform on computer display with scalii information derived from preamble.
integrate:	**Convert waveform data to volts, summing to** calculate and plot the integral of the signal.

Programming with QuickC

```
#include "c:\qc25\chpib.h"      /*HPIB library consant declarations  */
#include "c:\qc25\cfunc.h"      /*HPIB library function              */
#include  <stdio.h>             /*used for printf()                  */
#include <graph.h>              /*used for pc graphics routines      */
float     preamble[10];         /*real array to hold volt/time info  */
unsigned char waveform[4000];   /*array to hold waveform data        */
#define  ISC   7L               /*HPIB card address; must be long    */
#define SCOPE ISC*100 + 7L      /*oscilloscope address; must be long */
#define  TRUE  1                /*constant used in graph function    */

main()
{
  initialize();           /*initialize interface,scope & acquisition */
  capture waveform();     /*digitize signal                          */
  measure();              /*do Vpp and frequency measurement         */
  get_waveform();         /*bring waveform & preamble into computer  */
  save_waveform();        /*store waveform & preamble to disk file   */
  retrieve_waveform();    /*retrieve waveform & preamble from disk   */
  graph();                /*plot waveform data & scaling info on PC  */
  integrate();            /*calculate and plot integral of signal    */
}
```

Four include statements start the program. The files, chpib.h and cfunc.h, define needed constants, an error string procedure, and function prototyping. The file cfunc.h is not mandatory for a large memory model program, its inclusion helps reveal variable-type problems. The path of these files must specify the disk drive and directory where these .h files reside. The stdio.h file is needed whenever input or output functions are used. The graph.h file is needed whenever graphics functions are used.

Two global arrays are defined for the waveform data and preamble information. These arrays are needed ONLY if the waveform data is to be read into the computer for further analysis or storage.

The preamble array must be a real array of size 10. The preamble information is used to convert the raw waveform data into voltage and time pairs.

The size and structure of the waveform data array may vary with the language, oscilloscope, and format of the data. For QuickC, an unsigned char array is used with the HP 54600A with a waveform format of byte. The waveform data formats and preamble information are covered in detail in the "Waveform Data and Preamble" section of this guide.

The Interface Select Code (ISC) is defined as 7L. It must be a long as it is used as a parameter in the HP-IB library calls. The ISC address is set by a dip switch on the HP-IB card.

The oscilloscope address, SCOPE, 'is defined as ISC* 100 + 7L. The "7" is the address of the oscilloscope as set on the front panel in the HP-IB menu. This value must also be a long as it is used as a parameter in the HP-IB library calls.

The value of TRUE is defined as 1 and is used in the graph function.

Handling Errors

```
error_handler (error,routine)
int error;                    /*error number                          */
char *routine;                /*name of routine generating error      */
 {
 char ch;
 if (error != NOERR)
    {
      printf("Error in call to %s \n",routine);
      printf("%d %s \n",error,errstr(error));

      printf("press Enter to continue: ");
      printf("%c",ch = getchar());
    }
}                             /*end error_handler                     */
```

Errors may occur while communicating over the bus. To avoid hanging up the interface and computer, it is recommended that an error handling funtion be used.

In the function header, an error variable is declared to hold the error number. A string is also declared to hold the name of the command (or other descriptor) in which the error occurred.

If an error occurs, a *message is* **printed** telling you **where the error** occurred. The error number and error message string are printed. The **errstr** routine returns the error message string and is explained in the "C Error Handling" section of the HP-IB manual. To let you see the error message, the function pauses until the "Enter" key is pressed on the PC keyboard. Note that **Ctrl-C** will terminate the program.

Sending a Message Using String Output:

```
send (cmd)
char *cmd;              /*ASCII string to be sent to oscilloscope     */
{
   error_handler (IOOUTPUTS(SCOPE, cmd, strlen(cmd)),cmd);
}
```

To simplify the format of the sample program when sending messages to the oscilloscope, a generic output function has been written. The use of the "send" function simply makes the code easier to read. Instead of a call to the IOOUTPUTS function and the associated error checking, a single call is made to the function "send". The function then sends the ASCII string to the oscilloscope and the error handler routine prints any errors.

This generic string output function uses the HP-IB IOOUTPUTS function to output an ASCII string to the oscilloscope. The three parameters of IOOUTPUTS are the address of the instrument, the string to be sent, and the length of the string. If an error is encountered, the error handling routine prints the error.

Initializing the HP-IB and the Oscilloscope

```
initialize ()
{
 error_handler(IORESET(ISC),"IORESET");          /*reset interface    */
 error_handler(IOTIMEOUT(ISC,(double)10),"IOTIMEOUT");
                                                 /*set bus timeout    */
error_handler(IOCLEAR(ISC),"IOCLEAR");           /*clear interface    */

send ("*rst");  /*restore oscilloscope to default configuration     */
send (":autoscale");                 /*autoscale  oscilloscope      */
send (":channel1:probe x10");                    /**probe setting     */

/*set up acquisition subsystem: some are set by *rst, but are       */
/*                        included for completeness                 */
send (":acquire:type normal");                   /*normal acq mode    */
send (":waveform:points 4000");                  /*4000 data points   */

send (":acquire:complete 100");                  /*100% completion    */

_clearscreen(_GCLEARSCREEN);                     /*clear PC's screen */

}
```

Initializing the HP-IB Interface

IORESET sets the HP-IB interface to its start-up state (see the HP-IB manual for complete description).

IOTIMEOUT sets an interface timeout value in seconds to handle I/O operations that do not complete (for example, the HP-IB cable is not attached or the oscilloscope address is not correct).

IOCLEAR clears the interface in computer and all attached devices.

Initializing the Oscilloscope

The *RST is a common command that resets the oscilloscope to a known default configuration. Using this command will ensure that the oscilloscope is in a known state before configuring the oscilloscope. Using the *RST command will ensure very consistent, repeatable results. Without the *RST, a program may run one time, but the program may give different results the next time if the socpe is differently configured. For example, if the trigger mode is normally set to the edge trigger mode, the program may function properly, But if someone put the oscilloscope into the tv trigger mode from the front panel, the program may result in signals and measurements that are totally incorrect. The *RST defaults to a set confiiation, so the program may proceed from a given state each time.

The autoscale function will find and display all signals attached to the oscilloscope. You may or may not wish to have this done. It is probably advisable to program the oscilloscope's time base, channel, and trigger for the specific measurement to be made (just as would be done from the front panel). For example:

```
:chanl:range 8
```

sets vertical sensitivity to 1 V/div

```
:timebase:range 2
```

sets horizontal sensitivity to 200 ms/div

and use the other commands as needed to configure oscilloscope for desired measurement.

Specifying the Acquisition

From the front panel, the acquisition mode is set using the display menu and selecting the desired display mode such as normal or average. For the normal front panel run mode, this is the only control that needs to be made. For the average mode, the number of averages needs to be set.

When programming the oscilloscope over the bus and using the DIGITIZE command to ensure data capture, the ACQUIRE subsystem needs to be initialized. The most common ACQUIRE subsystem command is TYPE, where the aquisition mode of normal or average is set. If selecting the average mode, COUNT will control the number of averages to be taken.

In addition, for the DIGITIZE, you may indicate the number of data points to be acquired. The number of data points specified will depend upon time base setting and the acquisition type. In general, the fewer number of points acquired, the faster the acquisition will complete, but with a loss in timing resolution.

You may also specify a COMPLETE criteria between 0 and 100 percent. The DIGITIZE command will force the acquisition to run until the complete criteria is satisfied. All other commands to the oscilloscope are held off until the DIGITIZE is complete. Depending on the time base range and acquisition mode, setting the completion criteria to a value lower than 100% may significantly improve throughput while still providing sufficient data for very accurate measurements.

Capturing Data

```
capture_waveform()
{
 send (":digitize channel1");      /*capture data for channel 1    */
}
```

At this point, the oscilloscope and interface are completely configured. It is now suggested that the DIGITIZE operation be performed. DIGITIZE is a macro that clears the data buffers and acquires data for the requested channels to the specifications that you have set. When the data has been acquired, the acquisition process is stopped and the signal is displayed on the oscilloscope display. The data may then be measured using the parametric measurements or markers, or it may be stored in memory, or it may be transferred over the bus without the data changing.

The use of the DIGITIZE is highly recommended as it will ensure that sufficient data is available for measurement. Keep in mind that when the oscilloscope is running, communication with the computer interrupts data acquisition. Setting up the oscilloscope over the bus causes the data buffers to be cleared and the internal oscilloscope hardware to be reconfigured. If a measurement is immediately requested, there may not have been enough time for the data acquisition process to collect data and the results may be questionable. When operating from the front panel, the measurement may momentarily reflect 'not found", but is quickly updated, but over the bus only the error value of 9.9E+37 is returned if there is insufficient data.

In the HP 54600A, all channels that are currently on when the digitize command is sent are acquired. However, only those channels that are part of the command are acquired to the specified completion and acquire criteria. For this reason, it is suggested that the VIEW command be used to turn on channels that are to be acquired and the BLANK command be used to turn off undesired channels. This will improve the throughput of the DIGITIZE command.

Making Measurements

```
measure()
{
   float vpp_value, frequency;
   char ch;

 send (":measure:vpp?");          /*query oscilloscope for Vpp      */
 error handler(IOENTER(SCOPE,&vpp_value)); /*input Vpp             */
 send (":measure:frequency?");    /*query oscilloscope for freq     */
 error handler(IOENTER(SCOPE,&frequency)); /*inputfrequency        */
 printf ("Vpp = %f V \n",vpp_value);      /*print Vpp               */
 printf ("freq = %f Hz \n",frequency);    /*print frequency         */
 printf ("Press Enter to continue:");     /*pause for display       */
 ch = getch();                            /*read keypad to continue*/
}
```

After a digitize command, the signal is displayed on the screen and the acquisition process is stopped. Any of the parametric measurements can be made on this data using either the parametric measurements (such as Vpp, frequency, pwidth) or markers (such as VTIME, TVOLT, ESTART). All measurements use the IEEE standard for making measurements.

Measurement Query

When a measurement query is sent to the oscilloscope, the specified measurement is made and the result is stored in the output buffer of the oscilloscope. The next operation in the program is to read that measurement result into the computer. If another command or query is sent to the oscilloscope before the results are read, the error "QUERY INTERRUPTED" is displayed on the oscilloscope display.

Error in Measurements (9.9E + 37)

If the DIGITIZE command was not used to acquired data after configuring the oscilloscope, or if the oscilloscope was not given sufficient time to acquire data, there may not be enough waveform data available to make a valid measurement. In this case, the error value of 9.9E+37 is returned.

If the signal cannot be analyzed, the error value will also be returned. For example, if the signal is clipped, a Vpp measurement cannot be made. Or, if a whole period of the signal is not display, a frequency measurement cannot be made. In both cases, the error value of 9.9E +37 will be returned.

Transferring Waveform Data to the PC

```
get_waveform()
{
 int length;                          /*size of data transfers          */
 char msg[5];                         /*scratch buffer                  */
 send (":waveform:format byte");      /*54600 8 bit data format         */
 send (":waveform:preamble?");        /*query for volt/time info        */
 length = 10;
 error_handler(IOENTERA(SCOPE,preamble,&length),"preamble");
 send (":waveform:data?");            /*query for waveform data         */
 length = 4000;                       /*max size to be read             */
 error_handler(IOENTERAB(SCOPE,waveform,&length,1),"waveform");
 length = 1;                          /*input line feed character       */

 error_handler(IOENTERS(SCOPE,msg,&length),"read line feed");
}
```

After the DIGITIZE operation, the acquisition process is stopped. The dam that is displayed on the oscilloscope can be transferred to the computer, if desired, for storage to a file, for plotting to the computer display, or for further analysis.

The preamble information is an array of 10 real numbers that contains voltage and time scaling information that is used to convert the waveform data from a oscilloscope format to voltage and time. To transfer the preamble information, the formatted array transfer command, IOENTERA, is used.

The waveform data is a block of data consisting of a header (specifying the number of bytes of data that are to be transferred), the data in the selected format, and a terminating character (line feed). The arbitrary block data command, IOENTERAB, is used to transfer the data from the oscilloscope to the computer. All the data, except for the terminator, is read from the oscilloscope with the arbitrary block data command. The terminator is read using a string command, IOENTERS.

Before doing the arbitrary block data command, the value of length is set to 4000. This specifies the maximum number of bytes to be read. This value is determined by the number of points specified in the initialize function and the format requested.

In this example, the data is transferred out in the 8-bit format as QuickC has an unsigned char data structure. The data from the oscilloscope will be between 0 and 255.

NOTE: Both preamble and waveform data block are described in detail in the "Waveform Data and Preamble" section.

Storing Waveform Data and Information to the PC Disk

```
save_waveform()
{
  FILE *fp;
  fp = fopen("wave.dat", "wb");
  fwrite (preamble, sizeof(preamble[0]), 10, fp);
  fwrite (waveform, sizeof(waveform[0]), (int)preamble[2], fp);
  fclose (fp);
}
```

Once the waveform data and preamble information are in the computer, you may want to store this data to the disk for future reference.

In the program, a disk file is opened, the preamble array and waveform data are written to the file, and the file is closed.

Please note that for QuickC, the size of the waveform data record is the value in preamble[2] since C arrays start at zero, and the number of data points collected is in the third element of the preamble array.

Retrieving Waveform Data and Information from the PC Disk

```
retrieve_waveform()
{
  FILE *fp;
  fp = fopen("wave.dat", "rb");
  fread (preamble, sizeof(preamble[0]), 10, fp);
  fread (waveform, sizeof(waveform[0]), (int)preamble[2], fp);
  fclose (fp);
}
```

The preamble information and waveform information have been stored to a disk for retrieval when desired. To illustrate the retrieval process, the data will be read from disk by this function even though both the waveform data and preamble information arrays have valid data in them.

Drawing the Signal on the PC Display

```
graph()
{
   int    i;
   float  max, min;


 _setvideomode ( _ERESCOLOR);          /*set up EGA graphics          */
_rectangle ( _GBORDER, 0, 0, 639, 257);      /*draw border           */
 _setviewport (1, 1, 638, 256);       /*set viewport                 */
_setwindow (TRUE, 0, 0, 10, 8);       /*set mapping coordinates     */
 _setcolor(8);                        /*set color                    */
 for (i = 1; i < 10; i++)              /*plot graticule               */
  {
    _moveto_w(i,0);
    _lineto_w(i,8);
  }
 for (i = 1; i < 8; i++)
  {
    _moveto_w(0,i);
    _lineto_w(10,i);
  }
_setwindow(TRUE,1,0,preamble[2],255);/*set mapping coordinates     */
 _setcolor(10);
 for (i = 0; i < preamble[2]; i++)   /*plot waveform data           */
   _setpixel_w (i, waveform[i]);

_settextwindow(20, 1, 24, 80);       /*set text window              */
_settextposition (1, 1);             /*position cursor              */
 _settextcolor (15);                 /*set color: bright white      */
_printf ("V/div = %f V \n", 32 *preamble[7]);
 _settextposition (1,41);
_printf ("Offset=%V\n",(128-preamble[9])*preamble[7]+preamble[8]);
 _settextposition (2,1);
_printf ("S/div = %f S \n",preamble[2]*preamble[4]/10);
}
```

With graphics, you can add a visual dimension to the program, and often you will want to see the waveform displayed on the PC. To run graphics, your computer must have graphics capability, either **built** in or in the form of a graphics card such as the Color Graphics Adapter (CGA), Enhanced Graphics Adapter (EGA), or Video Graphics Array (VGA). You also need a video display (monochrome or color) that supports pixel-based graphics.

This sample program was written for an EGA adapter. When using the program, the _setvideomode, rectangle, setviewport, and setcolor commands may need to be altered to conform to your configuration.

The graphics library GRAPHICSUB contains functions for drawing lines, rectangles, and other shapes. The library can either be included in the combined libraries when installed, or can be explicitly linked in the MAKE menu.

Although the oscilloscope can be queried for its current settings of V/div, offset, and s/div, the preamble information will be used to calculate these settings. These scaling factors are displayed under the graticule. The calculations are explained in "Waveform Data and Preamble" section of this guide.

Calculating and Drawing the Integral of the Signal

```
integrate()
{
   float math[4001],min,max;  /*define array to hold math data      */
   int   i;
   math[0] = 0;                          /*start from zero           */
   for (i = 0;i < preamble[2];i++)/*calculate integral of signal    */
     math[i+1] = math[i] + (waveform[i]-preamble[9])*preamble[7]+
                 preamble[8];
   max = math[1];                        /*find max and min of integral */
   min = math[1];                        /*to set up graphics scaling   */
   for (i = 1; i <= preamble[2]; i++)
    {
     if (math[i] > max) max = math[i];
     if (math[i] < min) min = math[i];
    }
  _setwindow(TRUE,1, min, preamble[2],max);/*set up mapping        */
  _setcolor(12);

   for (i = 1; i <= preamble[2]; i++)
     _setpixel_w(i, math[i]);              /*plot integral          */
}
```

Another interesting exercise is to use the waveform data and the preamble information to calculate and plot the integral of the signal. The voltage of each data point is calculated using the preamble and this voltage value is summed to compute the integral.

These calculations are described in "Waveform Data and Preamble", the last section of this guide.

Programming withQuickBASIC

```
REM $INCLUDE: 'c:\hpib\qbsetup''QuickBasic header

DIM SHARED waveform%(4000)      'integer waveform data array
DIM SHARED preamble!(10)        'real array for volt/time info
DIM SHARED scope&,isc&          'long variable for addresses
isc&= 7                         'HP-IB Interface Select Code
scope& =isc&*100 +7             'scope address

CALL initialize          'initialize interface and oscilloscope
CALL capture.waveform    'digitize signal
CALL measure             'do VPPand Frequency measure
CALL get.waveform        'bring waveform & preamble into computer
CALL save.waveform       'store waveform & preamble to disk
CALL retrieve.waveform   'retrieve waveform & preamble from disk
CALL graph               'plot waveform data & scaling info
CALL integrate           'calculate and plot integral of signal
END
```

The header for the program is qbsetup . It sets up the necessary error status variables, declares some working variables, contains related COMMON statements and establishes an error handling routine.

Two global arrays (waveform and preamble) are defined for the waveform data and preamble information. These arrays are need ONLY if the waveform data is to be read into the computer for further analysis or storage.

The preamble array must be a real array of size 10. The preamble information is used to convert the raw waveform data into voltage and time pairs. The size and structure of the waveform data array may vary with the language, oscilloscope, and format of the data.

For QuickBASIC, an integer array is used for the HP 54600A with a waveform format of word. The waveform data formats and preamble information are covered in detail in the "Waveform Data and Preamble" section of this guide.

The Interface Select Code, isc&, is set equal to 7. This must be a long variable as it is used as a parameter in the HP-IB library calls. The ISC address is set by a dip switch *on* the HP-IB card.

The oscilloscope address, scope&, is set equal to isc& • 100+7. The "7" is the address of the oscilloscope as set on the front panel in the HP-IB menu. This must be a long variable as it is used as a parameter in the HP-IB library calls.

Handling Errors

Errors may occur while communicating over the bus. To avoid hanging up the interface and computer, it is recommended that an error handling procedure be used to report the errors.

An error handling routine is included with QBSETUP.BAS. If a subroutine detects an error, the error number is placed into the variable PCIB.ERR. A text error message is placed into the variable PCIB.ERR$.

After each subroutine call, if the value of the variable PCIB.ERR is 0, no error conditions have been detected. However, if its value is not 0, then the error message will be printed by transferring control, via BASIC's error trapping facilities, to an error handling routine in the QBSETUP.BAS file.

It is important to know that the variables PCIB.ERR, PCIB.BASERR, and NOERR are specified as COMMON in QBSETUP.BAS. This means that in the main program or any functions, these variables are recognized. However, if using subprograms (as in the sample program), they will not be part of the subprogram environment and must be specified as SHARED variables in any subprograms where they are used.

Sending a Message Using String Output

```
SUB send(cmd$)
SHARED PCIB.ERR, PCIB.BASERR, NOERR

 CALL iooutputs(scope&, cmd$, LEN(cmd$))
 IF PCIB.ERR<>NOERR THEN ERROR PCIB.BASERR
END SUB
```

To simplify the format of the sample program when sending messages to the oscilloscope, a generic output function has been written. The use of the "send" function simply makes the code easier to read. Instead of a call to the IOOUTPUTS function and the associated error checking, a single call is made to the function "send". The function then sends the ASCII string to the oscilloscope and calls the error handler routine if needed.

This generic string output function uses the HP-IB IOOUTPUTS function to output an ASCII string to the oscilloscope. The three parameters of IOOUTPUTS are the address of the instrument, the string to be sent, and the length of the string. If an error is encountered, the error handling routine will be invoked.

Initializing the HP-IB Interface and the Oscilloscope

```
SUB initialize
SHARED PCIB.ERR, PCIB.BASERR, NOERR
 CALL ioreset(isc&)                    'reset HP-IB interface
 if PCIB.ERR<>NOERR THEN ERROR PCIB.BASERR   'error handler
 timeout.val!=10
 CALL iotimeout (isc&,timeout.val!) 'set timeout to 10 seconds
 IF PCIB.ERR<>NOERR THEN ERROR PCIB.BASERR
 CALL ioclear(isc&)                    'clear interface and oscilloscope
 IF PCIB.ERR<>NOERR THEN ERROR PCIB.BASERR
 send ("*rst")                  'reset oscilloscope to default config
 send (":autoscale")             'autoscale oscilloscope
 send (":channel1:probe x10")  'probe setting
 'set up acquisition subsystem:  some are set by *rst, but are
 '                              included for completeness
 send (":acquire:type normal") 'normal acq mode
 send (":waveform:points 4000") '4000 data points
 send (":acquire:complete 100")  '100% completion
 CLS                            'clear screen of PC
END SUB
```

Initializing the **HP-IB** Interface

IORESET sets the HP-IB interface to its start-up **state** (see the **HP-IB** manual for complete description).

IOTIMEOUT sets an interface timeout value in seconds to handle I/O operations **that** do not complete (for example, the HP-IB cable is not attached or the oscilloscope address is not correct).

IOCLEAR clears the interface in the computer and all attached devices.

Initializing the Oscilloscope

The *RST is a common command that resets the oscilloscope to a known default configuration. Using this command will ensure that the oscilloscope is in a known state before configuring the oscilloscope. Using the *RST command will ensure very consistent, repeatable results. Without the *RST, a program may run one time, but the program may give different results the next time if the scope is differently configured. For example, if the trigger mode is normally set to the edge trigger mode, the program may function properly. But if someone put the oscilloscope into the tv trigger mode from the front panel, the program may result in signals and measurements that are totally incorrect. The *RST defaults to a set configuration, so the program may proceed from a given state each time.

The autoscale function will find and display all signals attached to the oscilloscope. You may or may not wish to have this done. It is probably advisable to program the oscilloscope's time base, channel, and trigger for the specific measurement to be made (just as would be done from the front panel). For example:

```
:chan1:range 8
```

sets vertical sensitivity to 1 V/div

```
:timebase:range 2
```

sets horizontal sensitivity to 200 ms/div

and use other commands as needed to configure oscilloscope for desired measurement.

Specifying the Acquisition:

From the front panel, the acquisition mode is set using the display menu and selecting the desired display mode such as normal or average. For the normal front panel run mode, this is the only control that needs to be made. For the average mode, the number of averages needs to be set.

When programming the oscilloscope over the bus and using the DIGITIZE command to ensure data capture, the ACQUIRE subsystem needs to be initialized. The most common ACQUIRE subsystem command is TYPE, where the aquisition mode of normal or average is set. If selecting the average mode, COUNT will control the number of averages to be taken.

In addition, for the DIGITIZE, you may indicate the number of data points to be acquired. The number of data points specified will depend upon time base setting and the acquisition type. In general, the fewer number of points acquired, the faster the acquisition will complete, but with a loss in timing resolution.

You may also specify a COMPLETE criteria between 0 and 100 percent. The DIGITIZE command will force the acquisition to run until the complete criteria is satisfied. All other commands to the oscilloscope are held off until the DIGITIZE is complete. Depending on the time base range and acquisition mode, setting the completion criteria to a value lower than 100% may significantly improve throughput while still providing sufficient date for very accurate measurements.

Capturing Data

```
SUB capture.waveform
  send ("*digitize channel1)    'capture data for channel 1
END SUB
```

At this point, the oscilloscope and interface are completely configured. It is now suggested that the DIGITIZE operation be performed. DIGITIZE is a macro that clears the data buffers, acquires data for the requested channels to the specifications that you have set, and finally, stops the acquisition process when complete, displaying the signal on the oscilloscope display. The data may then be measured using the parametric measurements or markers, or it may be stored in memory, or it may be transferred over the bus without the data changing.

The use of the DIGITIZE is highly recommended as it will ensure that sufficient data is available for measurement. Keep in mind that when the oscilloscope is running, communication with the computer interrupts data acquisition. Setting up the oscilloscope over the bus causes the data buffers to be cleared and the internal oscilloscope hardware to be reconfigured. If a measurement is immediately requested, there may not have been enough time for the data acquisition process to collect data and the results may be questionable. When operating from the front panel, the measurement may momentarily reflect "not found", but is quickly updated, but over the bus only the error value of 9.9E+37 is returned.

In the HP 54600A, all channels that are currently on when the digitize command is sent are acquired. However, only those channels that are part of the command are acquired to the specified completion and acquire criteria. For this reason, it is advised that the VIEW command be used to turn on channels that are to be acquired and the BLANK command be used to turn off undesired channels. This will improve the throughput of the DIGITIZE command.

Making Measurements

```
SUB measure
 SHARED PCIB.ERR, PCIB.BASERR, NOERR

   send (":measure:vpp?")                    'query oscilloscope for Vpp
   CALL IOENTER (scope&,vpp_value!)'input Vp
   IF  PCIB.ERR<>NOERR THEN ERROR PCIB.BASERR
   PRINT "Vpp = ";vpp_value!                 'print Vpp
   send (":measure:freq?)                     'queryscope for frequency
   CALL IOENTER (scope&, freq!)               'input frequency
   IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
   PRINT "frequency = ";frequency;" Hz"    'print frequency
   INPUT "Press Enter to continue",a$      'read keypad
END SUB
```

After a digitize command, the signal is displayed on the screen and the acquisition process is stopped. Any of the parametric measurements can be made on this data using either the parametric measurements (such as Vpp, frequency, pwidth) or markers (such as VTIME,TVOLT, ESTART). All measurements use the IEEE standard for making measurements.

Measurement Query

When a measurement query is made to the oscilloscope, the specified measurement is made and the result is stored in the output buffer of the oscilloscope. The next operation in the program is to read that measurement result into the computer. If another command or query is sent to the oscilloscope before the results are read, the error 'QUERY INTERRUPTED" will be displayed on the oscilloscope display.

Error in Measurements (9.9E + 37)

If the DIGITIZE command was not used to acquired data after configuring the oscilloscope, or if the oscilloscope was not given sufficient time to acquire data, there may not be enough waveform data available to make a valid measurement. In this case, the error value of 9.9E+37 will be returned.

If the signal cannot be analyzed, the error value is also returned. For example, if the signal is clipped, a Vpp measurement cannot be made. Or, if a whole period of the signal is not display, a frequency measurement cannot be made. In both cases, the error value of 9.9E + 37 is returned.

Transferring Waveform Data to the PC

```
SUB get.waveform
SHARED PCIB.ERR,NOERR,PCIB.BASERR
   send (":waveform:format word")          '16-bit format
   send (":waveform:preamble?)             'query for volt/time info
   max.len% = 10
   actual.len% = 0
   CALL ioentera (scope&, SEG preamble![1),max.len%,actual.len%)
   IF PCIB.ERR<>NOERR THEN ERROR PCIB.BASERR

   send (":waveform:data?")                'query for waveform data
   max.len% = 8000                         'max size of record
   actual.len%=0
   CALL ioenterab(scope&,SEG waveform%(1),max.len%,actual.len%,2)
   IF PCIB.ERR<>NOERR THEN ERROR PCIB.BASERR
   max.len% = 1                            'get line feed at end
   actual.len%=0
   msg$ = SPACE$(max.len%)
   CALL ioenters (oscilloscope&, msg$, max.len%,actual.len%)
   IF PCIB.ERR<>NOERR THEN ERROR PCIB.BASERR
END SUB
```

After the DIGITIZE operation, the acquisition process is stopped. The data that is displayed on the oscilloscope can be transferred to the computer, if desired, for storage to a file, for plotting to the computer display, or for further analysis.

The preamble information is an array of 10 real numbers that contains voltage and time scaling information that is used to convert the waveform data from a oscilloscope format to voltage and time. To transfer the preamble information, the formatted array transfer command, ioentera, is used.

The waveform data is a block of data consisting of a header (specifying the number of bytes of data that are to be transferred), the data in the selected format, and a terminating character (line feed). The arbitrary block data command, ioenterab, is used to transfer the data from the oscilloscope to the computer. All the data, except for the terminator, is read from the oscilloscope with the arbitrary block data command. The terminator is read using a string command, ioenters.

The value that the variable max.len% is set to before the arbitrary block command specifies the maximum number of bytes to be read, This value is determined by the number of points specified in the initialize function and the format requested.

In this example, the data is transferred out in the word format. This transfers 16 bits of data into an integer array. The data will be between 0 and 255.

NOTE: Both the preamble and waveform data block are described in detail in the "Waveform Data and Preamble" section of this guide.

Storing Waveform Data and Information to the PC Disk

```
SUB save.waveform
  OPEN "wave.dat" FOR BINARY AS #2
  FOR count% = 1 to 10
       PUT #2, , preamble!(count%)
  NEXT count%
  FOR count% = 1 TO preamble!(3)
       PUT #2, , waveform%(count%)
  NEXT count%
  CLOSE #2
END SUB
```

Once the waveform data and preamble information are in the computer, you may want to store this data to the disk for future reference.

In the program, a disk file is opened, the preamble array and waveform data are written to the file, and the file is closed.

In the QuickBASIC example, the size of the waveform data record is the value in preamble!(3) since BASIC arrays start at one, and the number of data points collected is in the third element of the preamble array.

Retrieving Waveform Data and Information from the PC Disk.

```
SUB retrieve.waveform
  OPEN "wave.dat" FOR BINARY AS #2
  FOR count% = 1 to 10
      GET #2, , preamble!(count%)
  NEXT count%
  FOR count% = 1 to preamble!(3)
      GET #2, , waveform%(count%)
  NEXT count%
  CLOSE #2
END SUB
```

The preamble information and waveform information have been stored to a disk for retrieval at any time. To illustrate the retrieval process, the data will be read from disk by this function even though both the waveform data and preamble information arrays have valid data in them.

Drawing the Signal on the PC Display

```
SUB graph
  SCREEN 9                              'set screen mode to EGA
                                        '640 x 200 x 2
  VIEW (1,1)-(638,256), ,15             'set viewport and draw border
  WINDOW (0,0)-(10,8)                   'prepare to draw the grid
  COLOR 8
  FOR i% = 1 TO 9
    LINE (i%,0)-(i%,8)
  NEXT i%
  FOR i% = 1 TO 7
    LINE (0,i%)-(10,i%)
  NEXT i%
  WINDOW (1,0)-(preamble!(3),255)'window on max data of 255
  COLOR 10
  FOR i% = 1 TO preamble!(3)
      PSET (i%, waveform%(i%))
  NEXT i%

  'use preamble information to calculate scaling information
  WDITH 80, 25
  VIEW PRINT 20 TO 24
  LOCATE 20, 1
  PRINT "V/DIV ="; (32*preamble!(8));" V"
  PRINT "Offset = ";(128-preamble!(10))*preamble!(8)
                  +preamble!(9);"V"
  LOCATE 20, 41
  PRINT "S/Div = "; preamble!(3)*preamble!(5)/10;" S"
END SUB
```

With graphics, you can add a visual dimension to the program, and often you will want to see the waveform displayed on the PC. To run graphics, your computer must have graphics capability, either built in or in the form of a graphics card such as the Color Graphics Adapter (CGA), Enhanced Graphics Adapter (EGA), or Video Graphics Array (VGA). You also need a video display (monochrome or color) that supports pixel- based graphics.

Although the oscilloscope can be queried for its current settings of V/div, offset, and s/div, the preamble information is used to calculate these settings. These scaling factors are displayed under the graticule. All of the calculations are described in "Waveform Data and Preamble", the last section of this guide.

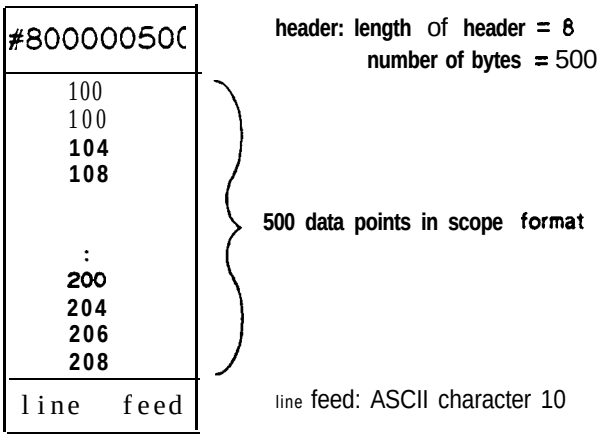
Calculating and Drawing the Integral of the Signal

```
SUB integrate
  SHARED PCIB.ERR, PCIB.BASERR, NOERR
  DIM math!(preamble!(3))          'define array for math data
  math!(0) = 0                     'start from zero
  FOR i% = 1 TO preamble!(3)       'calc and sum integral
  math!(i%) = math!(i%-1) + (waveform%(i%)-preamble!(10))
                *preamble!(8) + preamble!(9)
   NEXT i%
  max! = math!(1)                  'find min and max of integral
  min! = math!(1)                  'to set plot window
  FOR i% = 1 TO preamble!(3)
   IF math!(i%)  max! THEN max! = math!(i%)
   IF math!(i%)  min! THEN min! = math!(i%)
  NEXT i%
  WINDOW (1, min!)-(preamble!(3), max!) 'set up proper scale
  COLOR 12
  FOR i% = 1 TO preamble!(3)   'plot integral
    PSET (i%, math!(i%))
  NEXT i%
END SUB
```

Another interesting exercise is to use the waveform data and the preamble information to calculate and plot the integral of the signal. The voltage of each data point is calculated using the **peamble** and this voltage value is summed to compute the integral.

Waveform Data and Preamble

Sample data record with only **500** points of 8 bit data



The waveform data inside the oscilloscope can be transferred out in a variety of formats. The data is in a defined block format with a header specifying the length of the block. The block is terminated with a line feed character (ASCII character 10).

The data may be read from the oscilloscope in various formats as specified in the table below:

HP 54600A DATA FORMATS:

byte	**8** bits with values between 0 and 255
word	16 bits with **values** between 0 and 255 (hiih byte is zero **padded)**
ASCII	same as word format, separated by commas
no data (hole)	is represented by 0

The format used is based partly on the application, throughput considerations, and language data structures. For example, the O-bit format offers the best throughput as there is less data to be transferred across the bus. However, when writing in QuickBASIC, there is not a convenient S-bit data structure (like the QuickC unsigned char structure) available, hence a word format was selected for the sample QuickBASIC program.

The HP 54600A family offers 8-bit data, even when averaging is used, but the word format is offered for compatibility with computer systems and languages. The high byte of the word data is zero padded.

The HP 54600A sends the data in a high byte/low byte order in word format. The HP 54600A offers a waveform:byteorder command that allows the byte order to be controlled.

Although the HP82335 HP-ID interface card offers DMA transfer, the overhead associated with setting up the DMA offsets performance improvements.

The data is in a representation used by the oscilloscope and NOT in terms of volts. For this reason, the preamble array is also read into the computer. The preamble array contains vertical and horizontal scaling information which can be used to convert the waveform data into volts and the data number into a corresponding time.

Preamble Array of 10 Real Numbers

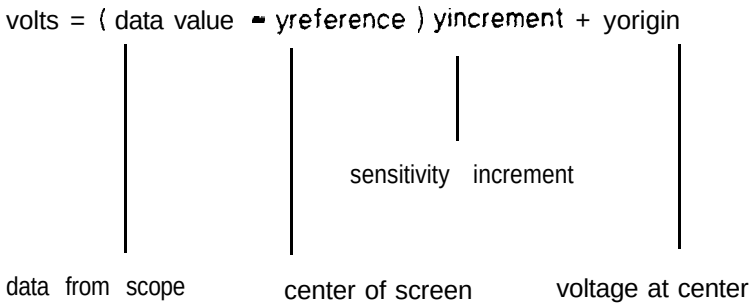
1 format:	ASCII, byte, word, etc.
2 acquisition type:	normal, average, etc.
3 number of points acquired:	dependent on oscilloscope
4 count:	not used: always = 1
5 xincrement:	time between consecutive points
6 xorigin:	time of xreference
7 xreference:	time reference point: always 1st point on screen
8 yincrement:	voltage between consecutive points
9 yorigin:	voltage at yreference (center screen)
10 yreference:	voltage reference point: always at center screen

When referencing the elements in the preamble array, take care that the subscript reference scheme is correct. For example, the format in a C program will be referenced by preamble[0], but will be referenced in a BASIC program by preamble(l). Likewise, the yreference in a C program will be referenced by preamble[9], but it will be referenced in a BASIC program by preamble(10).

To convert the waveform data to volt/time pairs, two conversion formulas are necessary: **one** for conversion to volts and one for conversion to time. The preamble information as specified on the previous page provides the conversion **values:**

PREAMBLE INFORMATION

use preamble to convert waveform data to volts:



Example of Conversion of Oscilloscope Data to Voltage

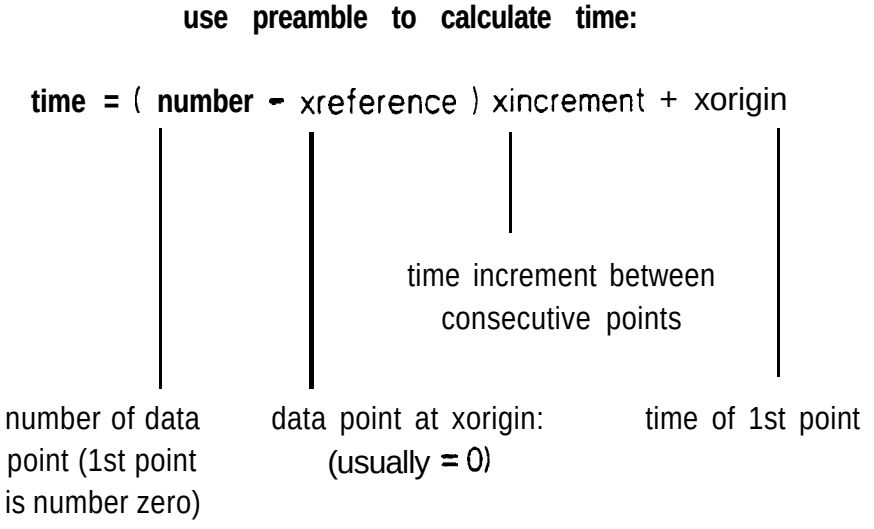
In reference to the formula above, the following calculates the voltage of the **first** data point in the waveform array:

in QuickC

```
voltage = (waveform[0] - preamble[9])*preamble[7] + preamble[8]
```

in QuickBASIC

```
voltage = (waveform%(1) - preamble !(10))*preamble!(8) + preamble!(9)
```



Example Calculation of Data Point Time

In refemce to the above formula, use the following to calculate the time of the first data point in the waveform array:

in QuickC

```
time = (0 - preamble[6])*preamble[4]) + preamble[5]
```

in QuickBASIC

```
time = (0 - preamble!(7))*preamble!(5) + preamble!(6)
```

Calculation of V/div, Offset, and s/div

There are three calculations in the graph function of the sample program that are dependent on the conversion formulas.

To calculate the vertical sensitivity (V/div), the increment value is multiplied by 32. The 32 is derived from the fact that there are 256 (O-255) vertical values spread across 8 divisions (256/8 = 32).

To calculate the offset, the center of the screen is defined as 128. The formula for conversion of oscilloscope data value to voltage is used with the data value being the center of the screen.

To calculate the horizontal sensitivity (s/div), multiply the time between consecutive points (xincrement) by the total number of points (number of points acquired), and divide by 10 for the ten divisions across the screen.

The formulas are different for QuickC and QuickBASIC. QuickC arrays are referenced from zero while QuickBASIC arrays are referenced from one.

HEWLETT
PACKARD

Reorder No. or
Manual Part No.
54600-90912-E0791

54600-90912