APPLICATION NOTE

Application Note #1 (Remote Operation)

Using the IEEE-488 bus on the IBM-PC for communication with the DATA 6000.

ANALOGIC
DATA PRECISION
16 Electronics Avenue Danvers, MA 01923 U.S.A.

INTRODUCTION:

The first step in communications with the DATA 6000 is a simple program that allows the user to enter commands and retrieve the result, if there is one. This application note describes such a program that was written on an IBM-XT running Microsoft Basic Version D2.00. The I/O card for the IBM is the Capital Equipment Corp. (CEC) PC-488 card.

There are two types of commands that one can send the DATA 6000. One is a command that produces no output to the IEEE bus. Examples of these are commands that put menus on the screen, such as DISP, MARK, DIR and INP. The other type of command is one which does produce a response to the bus. Examples of this type are RMS, PKPK, NPTS and MAX. The response consists of value of the operation defined by the command (e.g., RMS, PKPK) or the current value of a variable (NPTS).

The IBM-PC program, included herein handles both types of responses and can be used to transfer entire buffers over the IEEE-bus if desired. The following set-ups are required on the DATA 6000 in order to run this program successfully.

1.) CMD DEV = GPIB: (press DIR-PROG at same time)
2.) BUS ADR = 15 (press hit I/O, move cursor to GPIB configuration, hit I/O again)

These set-ups are default values, but double-check before running the program. Once the program is run and the ENTER COMMAND statement appears on the screen, any DATA 6000 command may be entered. If the ENTER COMMAND statement does not appear, it means that communications were not established and probably indicates a faulty IEEE cable or an addressing error.

Assuming successful communications, one can now try some simple commands. The labels on any of the keys, if sent as a command, will evoke the same response as does pushing that key manually. Once that is done, it is suggested that one try a command like RMS which will send back the RMS value of the waveform in the primary trace. Now if one sends the command BUF.A1 the contents of BUF.A1 will be returned. Note that the line length default value is 30 and that the number of characters per line is limited by this

line length. This may be increased or decreased according to future programming needs. When transferring buffers it is recommended that FLD DLM (press I/O key, move cursor to Field Format, press I/O again) be set to FLD COL. This results in output formatted in columns for readability.

In addition to commands, one may send a program line to the DATA 6000 by typing the line number followed by a space and the desired command. For example, to enter a program line to calculate the max value of a waveform one would type:

```
10 DSPL MAX BUF.A1
```

followed by a carriage return. If you now type PROG, you should see this line in the program. This then becomes a very effective way to enter programs.

The attached program is set up so that lines 20-80 initialize the bus, and lines 200-270 output and enter the data to and from the DATA 6000. Line 100 sends the PROMPT=3 command to the DATA 6000 and turns the prompt on. This is not necessary, however, it gives the operator assurance that the DATA 6000 is ready to receive another command (note that when the prompt is on, it is sent after completion of every command). Note that line 250 checks the result from the DATA 6000 to see if it is the prompt and if so, returns to ask for another command. Lines 160-180 simply enter that new command into the PC.

D6000 COMMAND PROGRAM FOR IBM PC

USING CEC GPIB BUS & SUBROUTINES

```
10 'GPIB Command Program for the DATA 6000
20 CLS : A$=SPACE$(80) :B$=SPACE(80)
30 INIT=0 : OUTPUT = 9 : ENTER=21
40 D6000%=15 : ADDR%=21 : SYSCONT%=0
50 DEF SEG = &HC000
60 CALL INIT (ADDR%,SYSCONT%)
70 A$="PROMPT=3" : GOSUB 140
80 '
90 PRINT
100 PRINT "ENTER COMMAND"
110 INPUT A$
120 GOSUB 140 : GOTO 90
130 '
140 CALL OUTPUT (D6000%,A$,STATUS%)
150 CALL ENTER (B$,LENGTH%,D6000%,STATUS)
160 IF MID$(B$,1,1)=">" THEN RETURN
170 PRINT B$
180 GOTO 150
190 '
200 END
```

Application Note

Application Note #2 (Remote Operation)

Sending Waveforms to the D6000 from the IBM-PC via RS232 lines.

ANALOGIC
DATA PRECISION
16 Electronics Avenue Danvers, MA 01923 U.S.A.

Introduction

The D6000 may be used as a data processing tool whether the waveforms of interest are created by digitizing through one of the plug in modules or, they are created by another system and transferred in for analysis. The latter case is the subject of this application note. Included is a program listing for an IBM-PC XT and a brief description of the program used to accomplish this transfer on a Rev. 4 D6000. Additional notes follow in order to cover some of the more detailed points of the transfer operation.

Program Description

The enclosed program (RSDATA.BAS) was written on an IBM-PC XT using PCDOS 2.0 and IBM Personal Computer Basic Version D2.00. Its purpose is to create a waveform in the IBM-PC and transfer it via RS232 lines to the D6000. The program is commented however some discussion is applicable here.

Lines 110 through 160 indicate the required set-up for the D6000. These paramenters must be set before the communication lines can be opened. Line 180 allocates memory in the PC for the waveform. Lines 210-220 clear the screen and open the communications lines for 9600 BAUD transfers. Lines 250-300 compute a decaying sine wave in the PC and store it in the array called A. Lines 330-340 initialize a record (also named A) in the Data 6000 with the correct scale factors. This record is given an initial value of 0.0 for all 1024 pts. Lines 360-380 place the record in the top trace of the display. Lines 410-500 send the data to the D6000 in blocks of 16 values. Since the transfer is done in the ASCII mode, the program creates a string variable for the first block consisting of the characters

A 0 15 = VALUE 1 Value 2 ..... VALUE 16

Where VALUE 1 = the ASCII equivalent of the first number in the A array.

VALUE 2 = the ASCII equivalent of the second number second number in the A array.

etc.

This string is then sent to the D6000 and the next string constructed for the next 16 values. This process continues until the entire record is sent. Lines 520-530 ask if the transfer is to be done again. If so, A is reinitialized and the complete record sent again. Lines 560-640 send the string out and wait for a reply, if any.

To run the program, set the D6000 parameters up according to lines 110-160. Load the program into the BASIC workspace in the PC and type RUN ↵ . You should see the message.

COMPUTING THE WAVEFORM - PLEASE WAIT

After a several second delay, the message

DONE COMPUTING - NOW SENDING INITIAL WAVEFORM TO D6000

will appear. This is followed shortly thereafter by the message

NOW SENDING DISPLAY MESSAGES TO D6000

At this point you should see the screen of the D6000 blink and A should appear as a straight line in the top trace of the screen.

The next message to appear is:

SENDING DATA TO D-6000 - LOOK AT THE SCREEN

Now you should see A being filled with the decaying sine wave in chunks of 16 values. Since A is 1024 points long, you will only see the first half of it being transferred. Wait for the message:

AGAIN?

to appear on the PC. The entire waveform may now be viewed by pressing the X key and adjusting X OFFSET and X SCALE appropriatly. If you want to see

the transfer again, type Y (capital Y) followed by a carriage return.

This program is very useful if your application does not demand high speed data transfers and may be used to study any waveform created by a remote computer. Once in the D6000, the full power of the signal processing and scaler math capabilities is available to analyze the waveform.

NOTES:

1) Use a straight through RS232 connector from the serø port (lower RS232 port labeled TERMINAL/DCE) to the COM1 port (DTE) of the IBM PC.

2) A waveform must be initialized in the D6000 before values may be assigned. If not done an error message will appear when the waveform is used in a statement or operation. This procedure is what lines 330-340 accomplish

3) The prompt is turned off in this program because all commands and statements sent to the D6000 require no response. The prompt would only add unnecessary transmissions to this program.

4) The record is sent in chunks of 16 values because of the string structure of the IBM-PC XT. In this computer the maximum string length (default mode) is 256 characters. 16 values in the form of the equation presented at the bottom of page 1 occupy about half of the available space in a string variable. Obviously other string lengths may be used as long as the 256 character limit is not exceeded.

```
10 'RSDATA.BAS
20 'TO CREATE A RECORD ON THE IBM AND TRANSFER TO THE D6000 IN ASCII.
30 '        D. ESTRICH @DATA PRECISION   7/25/85
40 '
50 'USES HIGH SPEED (9600 BAUD) COMMUNICATIONS.
60 '
70 'CONNECTIONS - USE THE COM1 PORT ON THE IBM AND THE SER0 PORT ON THE
80 '              D6000 (LOWER OF THE TWO RS232 PORTS) AND A PIN FOR PIN
90 '              RS232 CABLE.
100 '
110 'SET THE D6000 CMD DEV to SER0: (DIR AND PROG AT SAME TIME)
120 'SET I/O FIELD FORMAT PORT AND RS-232 CONFIG PORT TO SER0
130 'SET THE D6000 BAUD RATE, PARITY, STOP BITS, AND DUPLEX
140 'to 9600, NONE, 1, and HALF respectively
150 'DISARM THE PLUG-IN
160 'TURN THE PROMPT OFF (I/O KEY LINE FORMAT)
170 '
180 DIM A(1024)
190 '
200 ' OPEN COMMUNICATIONS PORT ON THE IBM
210 CLS
220 OPEN "COM1:9600,N,8,1,CS,DS,CD" AS #1
230 '
240 'CREATE THE WAVEFORM IN THE PC
250 PRINT "COMPUTING THE WAVEFORM - PLEASE WAIT"
260 SCALE=3.14159/15!
270 FOR I=1 TO 1024
280 A(I)=(1024-I)*SIN(I*SCALE)
290 NEXT I
300 PRINT "DONE COMPUTING - NOW SENDING INITIAL WAVEFORM TO D6000"
310 '
320 'CREATE THE WAVEFORM IN THE D6000 WITH INITIAL VALUES OF 0.0
330 B$="A @W 1024 V S 0.0 5E-5 0.0 2048.=0.0"
340 NWAIT=2800:GOSUB 560
350 '
360 PRINT "NOW SENDING DISPLAY MESSAGES TO D6000"
370 B$="TRCSRC =A ;DISP"
380 NWAIT=2800:GOSUB 560
390 '
400 'SEND DATA TO D6000
410 PRINT "SENDING DATA TO D6000 - LOOK AT THE SCREEN"
420 'FIRST CONVERT THE DATA TO ASCII CHARACTERS
430 NWAIT=100
440 FOR J=0 TO 1023 STEP 16
450 B$="A "+STR$(J)+" "+STR$(J+15)+"="
460 FOR I=1 TO 16
470 B$=B$+STR$(A(I+J))+" "
480 NEXT I
490 GOSUB 560
500 NEXT J
510 '
520 INPUT "AGAIN ?  ",ANS$
530 IF ANS$="Y" THEN 330
540 END
550 '
560 PRINT #1,B$
570 '
580 'WAIT FOR COMMUNICATION TO END - GET ANSWER IF ANY
590 FOR I=1 TO NWAIT:NEXT I
600 V=LOC(1)
610 IF V<1 THEN RETURN
620 A$=INPUT$(V,#1)
630 IF A$<>"" THEN PRINT A$
640 RETURN
```

Application Note

Application Note #3 (Remote Operation)

Remote Operation of D6000 from IBM-PC XT using RS232 lines

ANALOGIC
DATA PRECISION
16 Electronics Avenue Danvers, MA 01923 U.S.A.

Introduction

Remote operation of the D6000 provides the flexibility to control data acquisition, processing, and transfers through programmatic control. This application note discusses a fundamental remote control program written on an IBM-PC XT with IBM Personal Computer Basic (Version D2.00), that runs with a revision 4 D6000. Included are the program listing and a discussion of some considerations when using RS232 communications.

Discussion

This program is set up to communicate using ASCII characters. This means that all characters and numbers must be in ASCII form before being sent to the D6000. It also means, because a number may contain five significant digits plus a decimal point, that six ASCII characters are used to represent a number. In addition, the RS232 lines consist of just 2 wires and are capable of transmitting only 1 bit at a time. Therefore each number transmitted may consist of up to 48 sequintial bits (6 characters X 8 bits/character). The point of this is that RS232 ASCII communications are used when transfer speeds are not a prime consideration and/or one is interested in transmitting data over large distances. Other options are available if either of these conditions exist, and they are RS232 binary transfers and IEEE-488 ASCII and binary transfers.

For purposes of this program, commands sent to the D6000 may be grouped into two categories, those that involve a response and those that don't. This is important when considering use of the prompt which, in the default mode is turned on (press I/O, move cursor to line format, press I/O again, change PORT to serø, prompt appears in feild 5). The prompt (if on) will be returned to the remote following every command. This allows the remote computer to determine when a command is done being processed, which is impossible without the prompt for commands that invoke no response. If

the nature of your control program is such that only commands that invoke no response or only commands that do invoke a response are used, then the prompt is not necessary.

Line 250 of the program is a wait loop and is necessary because the hand-shaking lines are not used in this program. Since different commands take different periods of time to execute, the basic program needs to delay for a "long" period of time before it executes line 280 (this line checks the length of the message in the input buffer). If line 280 was executed before the message was sent, obviously it would miss the message.

One limitation on the use of this program is that the message length is limited to 256 bytes or 256 ASCII characters. This means that a maximum of approximately 32 numbers of a record may be transferred in one transmission. This is an IBM-Basic limitation and may be worked around by increasing the random file size in the Basic command (see IBM Basic Reference Manual) and including multiple read statements to read the input buffer. Another limitation is the speed of data transfers however this may be overcome on RS232 lines with binary transfers as discussed previously.

In summary, note that this program contains the elements necessary to construct a more complicated control program which can perform many data acquisition and processing tasks. Some ideas to consider are:

1) A program to write programs on the D6000.
   -this is a simple extension of the RSTRNS.BAS program where the command consists of a program statement line.
2) A program to control the acquisition of a signal by sending required set-up configuration and ARM-DARM commands.
3) A program to transfer data in and out of D6000.
4) A program to control processing of records in the D6000 and retrieve results.
5) A program to load files from disk for processing.
6) A program to load different control files and D6000 programs from disk for a changing test configuration.
7) A device driver program that integrates the D6000 into a system and is accessed by function calls. In this way the D6000 acts as a signal processing/math library which may be accessed by many users without detailed knowledge of the D6000.

```
10 'RSTRNS.BAS
20 'ENTER COMMAND PROGRAM FOR THE IBM PC XT.
30 '        D. ESTRICH @DATA PRECISION   7/25/85
40 '
50 'USES HIGH SPEED (9600 BAUD) COMMUNICATIONS.
60 '
70 'CONNECTIONS - USE THE COM1 PORT ON THE IBM AND THE SER0 PORT ON THE
80 '              D6000 (LOWER OF THE TWO RS232 PORTS) AND A PIN FOR PIN
90 '              RS232 CABLE.
100 '
110 'SET THE D6000 CMD DEV to SER0:
120 'SET I/O FIELD FORMAT PORT AND RS-232 CONFIG PORT TO SER0
130 'SET THE D6000 BAUD RATE, PARITY, STOP BITS, AND DUPLEX
140 'to 9600, NONE, 1, and HALF respectively
150 '
160 ' OPEN COMMUNICATIONS PORT ON THE IBM
170 CLS
180 OPEN "COM1:9600,N,8,1,CS,DS,CD" AS #1
190 '
200 ' ENTER THE DESIRED COMMAND AND SEND TO THE 6000
210 INPUT "ENTER COMMAND  ",B$
220 PRINT #1,B$
230 '
240 'WAIT FOR ANSWER
250 FOR I=1 TO 800:NEXT I
260 '
270 ' INPUT THE ANSWER IF THERE IS ONE AND PRINT OUT ON THE SCREEN
280 V=LOC(1)
290 IF V<1 THEN 210
300 A$=INPUT$(V,#1):IF A$ <> "" THEN PRINT A$;
310 GOTO 210
```

Application Note

Application Note #4 (Remote Operation)

Maximizing Data Transfer Rates from the D6000 to the IBM-PC

ANALOGIC
DATA PRECISION
16 Electronics Avenue Danvers, MA 01923 U.S.A.

Introduction

For applications requiring fast data transfer rates, the binary transfer mode may be used in conjunction with the DMA capability of the IBM. This application note describes a program, written on a IBM-PC XT in IBM Personal Computer Basic Version D2.00, that was used to measure the transfer times for various record sizes. Included are the results of the test and a listing of the program used. Note that the CEC-PC-488 card was used in this case and that these results are repeatable with that card.

Program Description

In general, one can assume that longer records will be transferred at higher overall rates than shorter ones. This is because the overhead time associated with each transfer is fixed (regardless of record length) and becomes a smaller percentage of the total transfer time, as the record length becomes large. The overhead in this case is the time spent requesting the data and excuting the interperative basic statements of the program. With this overhead included, the maximum rate for transferring a single buffer multiple times was 150 K bytes/sec for a 16384 point record. This rate could be increased by using a compiled language or assembler, and by eliminating the header portion of the record.

The heart of the program is lines 220-320. Line 220 requests the record BUF.A1. The first DMA statement (line 240) calls the DMA routine and inputs the header information. The second DMA call (line 270) inputs the binary data. A counter is then incremented and the same procedure executed again. The BEEP statements are used to time the loop after it has executed 1000 times. Results of running this test for several record sizes are presented in Table 1. Note the increasing transfer rates for larger records.

TABLE 1 - DMA Transfer Rates for Various Record Sizes

NUMBER OF POINTS	DATA TRANSFER RATES (INCLUDING OVERHEAD TIMES)
512	14 K Bytes/sec
1024	26 K Bytes/sec
2048	44 K Bytes/sec
4096	74 K Bytes/sec
8192	110 K Bytes/sec
16384	150 K Bytes/sec

Notes:

1.) DAT in lines 230 and 260 indicate the length of the transmission in bytes. The program listing indicates a 36 byte header and a 1024 byte record (512 pts.). To test for different record sized DAT was changed appropriatly in line 260.

2.) The data in these transfers were stored in the same memory location for each transfer thereby overwritting old data. This was done for ease of programming. If overwrites are not desireable, the offset address in memory must be changed each time through the loop. DDEST is the offset address and is defined in line 230 for the header and 260 for the data.

```
10 'QUICK.BAS
20 '
30 'TEST OF MINIMUM TRANSFER TIMES FROM D6000 TO
40 'THE IBM PC WITH THE CEC CONTROLLER BOARD
50 '  USES PC DOS 2.0 AND IBM PERSONAL COMPUTER BASIC VERSION D2.00
60 '  FOR REVISION 4.00 D6000
70 '
80 'TRANSFERS A FILE FROM D6000 TO PC MEMORY USING DMA
90 'THEN DOES IT AGAIN AS FAST AS POSSIBLE.
100 '
110 'WRITTEN BY DOUG ESTRICH DATA PRECISION MAY 14, 1985
120 '
130 CLS:DEFINT A-Z:DEF SEG=&HC000
140 INIT=0:ADDR=21:LEVEL=0
150 TRANSMIT=3:REC=27:TRA=24:DMA=42
160 CALL INIT (ADDR,LEVEL)
170 T$="FORMAT=BINARY;OMODE=4;FASTIO=ON"
180 GOSUB 350
190 ICOUNT=0
200 BEEP
210 '
220 T$="BUF.A1":GOSUB 350:GOSUB 370
230 DAT=36:DDEST=&H3000:MODE=&H105
240 CALL DMA(DDEST,DAT,MODE,STATUS)
250 GOSUB 390
260 DAT=1024:DDEST=&H3003:MODE=&H105
270 CALL DMA(DDEST,DAT,MODE,STATUS)
280 ICOUNT=ICOUNT+1
290 IF ICOUNT=1000 THEN BEEP
300 GOSUB 390
310 '
320 GOTO 220
330 END
340 '
350 TS$="UNL UNT MTA LISTEN 15 DATA '"+T$+"' 10 END "
360 CALL TRANSMIT (TS$,STATUS%):RETURN
370 TR$="UNL MLA TALK 15"
380 CALL TRANSMIT (TR$,STATUS%):RETURN
390 A=INP(8)
400 IF (A AND 2) =2 THEN RETURN ELSE 390
```

APPLICATION NOTE

Application Note #5 (Remote Operation)

Mnemonics for the D1000 Pre-Amplifier

ANALOGIC®
DATA PRECISION®
16 Electronics Avenue Danvers, MA 01923 U.S.A.

INTRODUCTION:

Remote control of the D1000 Pre-Amplifier may be used to change the input range and coupling settings of each of the two channels. The same mnemonic code is used for each channel. When the code is sent it operates on the channel defined by the screen display under the INP key. This channel may be changed by sending the sequence:

INPSEL = 1 for CH 1
or
INPSEL = 2 for CH 2

The mnemonic and codes are listed below:

COUPL2 = 1 = DC
= 2 = AC

RANGE2 = 1 = ±50mV
2 = ±100mV
3 = ±200mV
4 = ±500mV
5 = ±1.0V
6 = ±2.0V
7 = ±5.0V
8 = ±10.0V
9 = ±20.0V

In addition, the input configuration of the 620 plug-in may be changed by using the mnemonic INPCON

INPCON = 1 = 620 ONLY
INPCON = 2 = 620, D1000

APPLICATION NOTE

APPLICATION NOTE #6

OPERATING THE D6000 USING RS232 SERIAL INTERFACE (GENERAL)

ANALOGIC
DATA PRECISION
16 Electronics Avenue Danvers, MA 01923 U.S.A.

SPECIFICATIONS

The D6000 serial interfaces provide both DCE (for use with terminal or printer) and DTE (for use with host computer or modem) configurations. Upon power-up, both interfaces default to a "three-wire" serial interface; requiring no handshake signals to be present. Although this facilitates connection, data transfer to a device whose bit rate (baud rate) may be set correctly but whose frame rate is slow (as in a printer, unbuffered) will be impossible. The 'HNDSHK' command will permit implementation of DSR/DTR handshaking, giving control of the transmission frame rate to the slower device. Normal voltages for 'HNDSHK=2':

	CMD DEV	2	3	4	5	6	8	20
(DCE)	SERO:	IN	-10	0	+10	+10	+10	IN
(DTE)	SER1:	-10	IN	0	IN	0	0	+10

Pin 1 is shield, Pin 7 is signal ground '0' indicates an open connection, 'IN' indicates an input (nominally -0.6 volts DC, if measured open).

For example, a three wire interface to a dumb terminal would be wired as follows:

TERMINAL (DTE)		D6000 (SERO:) (DCE)
Pin 7	...to...	Pin 7
Pin 2	...to...	Pin 2
Pin 3	...to...	Pin 3

A very fast computer interface would be wired:

COMPUTER (DCE)		D6000 (SER1:) (DTE)
Pin 7	...to...	Pin 7
Pin 2	...to...	Pin 2
Pin 3	...to...	Pin 3

A slow computer or unbuffered printer interface: (HNDSHK=2)

PTR(DTE)/COMPUTER (DCE)			(SER0: (SER 1:) D6000 (DCE)/(DTE)
	Pin 7	...to...	Pin 7
(Computer Only)	Pin 2	...to...	Pin 2
	Pin 3	...to...	Pin 3
(Computer Only)	Pin 5	...to...	Pin 5
	Pin 20	...to...	Pin 20

(NOTE that since a printer is a 'Receive-Only' device, it is not necessary to use a control for the printer transmit pin, nor the transmit pin (2) itself.)

Depending upon what device (computer, printer, terminal, etc.) is to be used, however, some thought must be given to what the device must "SEE" at the interface in order to operate. An Epson printer, for example, must "SEE" a high (+10V) at it's Pin 6 in order to begin printing; an IBM PC (with asynchronous communications adapter) must "SEE" (if full handshake has been implemented) Pins 5, 6 and 8 high (+10V) before transmitting (although, by default, a 'three-wire' interface will work).

The D6000 does not support the so-called 'software controls' (control of the transmission via certain characters in the ASCII code, (Xon, Xoff)).

Data sent to or from the D6000 is handled in three different ways; line, field and block. To visualize these different 'Formats' consider these examples:

1. Sending a simple command to the D6000.

   - "RESET" <Carriage Return>

   will cause the D6000 to reset to the default state. This is a 'LINE'. The 'Line Delimiter' (selected by either 'LINDLM = 1' (Remote Command) or by pressing 'I/O', selecting 'Line Format', then 'I/O' again is the 'Carriage Return' character (ASCII code 13 (decimal)) in this case.

2. Receiving Data Points.

   - "BUF.A1 (0,28)" <CR>

   will cause the first 29 data values in the "BUF.A1" record to be sent as 'Fields', groups of ASCII numbers (0-9 with '-', '.', and 'E' as necessary) representing the actual voltage values of the first 29 data points:

   ```
   -0.00801,....0.1877,.....,0.9909
    0............10...........29
   ```

   In this case, the command 'BUF.A1 (0,28)' <CR> was a 'Line' using 'Line Delimiter' <CR>, however, the response (the data points) represents several discrete values, each of which must be 'Delimited' by a character; in this case, a 'Comma'. The D6000, of course, must indicate that it has ended it's transmission. The 'Line End' character(s), by default, is <CR><LF>; a combination that will cause the terminal's cursor to move down one line and to move to the start of that line.

By default, the 'Prompt Character' (a character familiar to CP/M and MS-DOS users) is an ASCII 62 (decimal) the 'Right Caret' character, indicating that the D6000 is ready for the next transmission.

3. Block Transfer of Binary Data

"BUF.A1#" <CR>

will cause the D6000 to transmit n X 2 bytes (number of point times 2 characters), each work (two-byte pair) representing the vertical position of the data point. Since, in many record every ASCII character is present, using an ASCII character to indicate the end of transmission will usually cause a premature action by the receiving device (often the user's application program). Therefore, this kind of data must be treated as a 'Block;' some fixed number of bytes which, when all received and counted, constitute the entire message. Binary data is always sent in block format and the user's application must account for this.

CONFIGURATIONS

As stated earlier, the 'Baud Rate' (or 'bits-per-second') of the remote device must match that of the D6000 (version 3.02 users: note that your remote device should operate at 4800 baud or greater) in order to receive data correctly. (Incorrect, unexpected data is called, appropriately, 'Garbage'). The D6000 supports numbers of different RS232 'Parameters' including baud rate, parity, stop bits and duplex. 'Parity' and 'Stop Bits' (put simply) determine the number of actual bits used to represent characters (or data). Be cautious of computers and peripherals that default to a 7 bit character length if you intend to transfer binary data.

In any case, the parameters of your remote device and that of the D6000 must be identical. Check this by selecting 'RS232 Parameters' from the 'I/O' menu.

By the way, a word about 'Message Format.' A 'Message' might be a character, a 'Line' or a group of 'Fields'. The character selected by the 'MSE DLM' will be interpreted as 'EOM' by the D6000. Message End characters should generally be avoided unless there is a specific need not addressed by 'Line' or 'Field' terminators.

As stated earlier, the D6000 provides both DTE ('Data Terminal Equipment') and DCE ('Data Communications Equipment').

As a rule-of-thumb, note these differences:

DTE	DCE
Male Connector	Female Connector
Pin 2 is TXD (Transmits)	Pin 2 (still called 'TXD') receives
Used on terminals/printers	Used on computers/modems
SER 1: on D6000	SER 0: on D6000

If a cable has continuity between 1-1, 2-2, 3-3, etc., connect 'DTE' to 'DCE'. If cross connections are seen (2-3, 6-20, 8-20, etc.), connect DTE - DTE, DCE - DCE. A cross-wired cable is called a 'Null Modem'. Generally, however, it is best to custom-wire an RS232 cable for the application (and clearly label it!).

DATA FORMATS

The following messages will cause record 'BUF.A1' to be sent over an interface in signed, free format ASCII numeric fields (eg. -2.67E8, 0.002, 3.783-5, -2.77). Note that 'Format' as seen in the 'IO' 'Field Format' menu, will determine whether or what unit will accompany the numerics. (Refer to 'Format' command.)

-BUF.A1

returns all point values with 'Field Delimiter' between them and the line end character(s) for every n fields approaching the selected line length.

-SER0: = BUF.A1 OR SER1: = BUF.A1

SAME AS ABOVE

-SUM, SUB, SQ, RCP, (ETC.)

same format as above, but the values are the result of math operating on Trace 1 (or Trace 1 and 2).

-BUF.A1 (68)

causes only the value for the 69th point in BUF.A1 to be sent.

-BUF.A1 (27,108)

causes points 27-108 (inclusive) to be sent as free-format fields as in 'BUF.A1' above.

-BUF.A1?

causes the 'Descriptor' (data type, size, scaling factors and offsets) to be sent as fields in ASCII free format.

-BUF.A1#

block binary transfer, 16 bit signed integer (note that this structure (#) supersedes the ASCII format).

-GREEN 'a' key, type 'SER0' (or SER1) with key pad, then blue "F" key (shifted alpha), 'Copy' for colon (:), the 'a' to clear (top line reads 'SER0: = BUF.A1' (or any record in Trace (1)) and will send that record to the port.

-BLUE 'F', then 'XFER' (shifted 'Copy') will call a menu permitting manual section of source and destination records or interfaces (a good way to "Force" data out to test the link and baud rate).

Application Note

APPLICATION NOTE #7 (REMOTE OPERATION)

GPIB BINARY DATA TRANSFER FOR THE IBM PC WITH THE CEC CONTROLLER BOARD

ANALOGIC®
DATA PRECISION®
16 Electronics Avenue Danvers, MA 01923 U.S.A.

The three attached programs demonstrate GPIB bus communication and data transfer between the IBM PC and the D6000. The D6000 requires the AOS (Advanced Operating System) for the binary transfer programs (Program 1 & 2). The CEC board is the new PC 488.

The first program transfers data (BUF.A1) from the D6000 to the IBM PC memory using DMA transfer. The data is then transferred from the IBM PC memory back to the D6000. The new record in the D6000 is called NEW.A1. Approximately three acquisitions and two way transfers can take place each second.

The second program is similar to the first. The difference is that DMA and FASTIO are not used; just binary transfer.

The third program is a command program which can be used to test and enter D6000 programs.

```
10  'DMA1
20 'DEMONSTRATION PROGRAM FOR GPIB DMA TRAMSFER USING
30 'THE IBM PC WITH THE CEC CONTROLLER BOARD
40 '
50 'TRANSFERS A FILE FROM D6000 TO PC MEMORY USING DMA
60 'THEN TRANSFERS THE FILE FROM PC MEMORY TO D6000 USING DMA
70 '
80 'WRITTEN BY FRED ABORN, DATA PRECISION MARCH 20, 1985
90 '
100 CLS:DEFINT A-Z:DEF SEG=&HC000
110 INIT=0:ADDR=21:LEVEL=0
120 TRANSMIT=3:REC=27:TRA=24:DMA=42
130 CALL INIT (ADDR,LEVEL)
140 T$="FORMAT=BINARY;OMODE=4;FASTIO=ON"
150 GOSUB 350
160 '
170 T$="BUF.A1":GOSUB 350:GOSUB 370
180 DAT=36:DDEST=&H3000:MODE=&H105
190 CALL DMA(DDEST,DAT,MODE,STATUS)
200 GOSUB 390
210 DAT=1024:DDEST=&H3003:MODE=&H105
220 CALL DMA(DDEST,DAT,MODE,STATUS)
230 GOSUB 390
240 '
250 T$="NEW.A1=":GOSUB 350
260 DAT=36:DDEST=&H3000:MODE=&H109
270 CALL DMA(DDEST,DAT,MODE,STATUS)
280 GOSUB 390
290 DAT=1024:DDEST=&H3003:MODE=&H109
300 CALL DMA(DDEST,DAT,MODE,STATUS)
310 GOSUB 390
320 '
330 GOTO 170
340 '
350 TS$="UNL UNT MTA LISTEN 15 DATA '"+T$+"' 10 END "
360 CALL TRANSMIT (TS$,STATUS%):RETURN
370 TR$="UNL MLA TALK 15"
380 CALL TRANSMIT (TR$,STATUS%):RETURN
390 A=INP(8)
400 IF (A AND 2) =2 THEN RETURN ELSE 390
410 END
```

```
10  'IEEE0
20 'DEMONSTRATION PROGRAM FOR GPIB BINARY TRANSFER USING
30 'THE IBM PC WITH THE CEC CONTROLLER BOARD
40 '
50 'TRANSFERS A FILE FROM D6000 TO PC MEMORY USING BINARY TRANSFER
60 'THEN TRANSFERS THE FILE FROM PC MEMORY TO D6000 USING BINARY TRANSFER
70 '
80 'WRITTEN BY FRED ABORN, DATA PRECISION MARCH 15, 1985
90 '
100 DEFINT A-Z:DEF SEG=&HC000:INIT=0:ADDR=21
110 LEVEL=0:TRANSMIT=3:REC=27:TRA=24
120 CALL INIT (ADDR,LEVEL)
130 T$="FORMAT=BINARY;OMODE=4":GOSUB 290
140 '
150 T$="BUF.A1":GOSUB 290:GOSUB 330              'INPUT ROUTINE
160 HEADER=36:HDEST=&H3000
170 CALL REC(HDEST,HEADER,LENGTH,STATUS)          'TRANSFER THE 36 BYTE HEADER
180 DAT=1024:DDEST=&H3003                         'BLOCK TO ADDRESS 30000
190 CALL REC(DDEST,DAT,LENGTH,STATUS)             'TRANSFER THE 1024 BYTE DATA
200 '                                             'BLOCK TO ADDRESS 30030
210 T$="NEW.A1=":GOSUB 310                        'OUTPUT ROUTINE
220 HEADER=36:HDEST=&H3000
230 CALL TRA(HDEST,HEADER,STATUS)                 'TRANSMIT HEADER TO D6000
240 DAT=1024:DDEST=&H3003
250 CALL TRA(DDEST,DAT,STATUS)                    'TRANSMIT 1024 BYTE DATA
260 '                                             'BLOCK TO D6000
270 GOTO 150
280 '
290 TS$="UNL UNT MTA LISTEN 15 DATA '"+T$+"' 10 END "    'NORMAL TRANSMIT
300 CALL TRANSMIT (TS$,STATUS%):RETURN
310 TS$="UNL UNT MTA LISTEN 15 DATA '"+T$+"' 10  "       'TRANSMIT WITHOUT EOI
320 CALL TRANSMIT (TS$,STATUS%):RETURN
330 TR$="UNL MLA TALK 15"                         'INSTRUCT D6000 TO TALK
340 CALL TRANSMIT (TR$,STATUS%):RETURN
```

D6000 COMMAND PROGRAM FOR IBM PC

USING CEC GPIB BUS & SUBROUTINES

```
10 'GPIB Command Program for the DATA 6000
20 CLS : A$=SPACE$(80) :B$=SPACE(80)
30 INIT=0 : OUTPUT = 9 : ENTER=21
40 D6000%=15 : ADDR%=21 : SYSCONT%=0
50 DEF SEG = &HC000
60 CALL INIT (ADDR%,SYSCONT%)
70 A$="PROMPT=3" : GOSUB 140
80 '
90 PRINT
100 PRINT "ENTER COMMAND"
110 INPUT A$
120 GOSUB 140 : GOTO 90
130 '
140 CALL OUTPUT (D6000%,A$,STATUS%)
150 CALL ENTER (B$,LENGTH%,D6000%,STATUS)
160 IF MID$(B$,1,1)=">" THEN RETURN
170 PRINT B$
180 GOTO 150
190 '
200 END
```

APPLICATION NOTE

Application Note #8

Using the DATA 6000 with Lotus

ANALOGIC®
DATA PRECISION®
16 Electronics Avenue Danvers, MA 01923 U.S.A.

The DATA 6000 Waveform Analyzer offers a nearly ideal architecture for use with popular spreadsheet/graphics programs running on IBM PC's, XT's, AT's, and compatibles. Because the default I/O format of captured data is ASCII numbers, the D6000 is well suited for a simple BASIC program that can write a file for IMPORT to a spreadsheet.

Lotus' 1-2-3 package, for example, uses a simple menu (FILE-IMPORT-NUMBERS) selection to convert an ASCII data file to the worksheet file format. 1-2-3 offers powerful editing and graphics options to the user who must manipulate data for reporting and presentation of data while additional processing may be performed off-line using macros thereby increasing total throughput.

When the 'import' facility is selected it 'looks for' an ASCII file of type .PRN. If numeric, this file will consist of ASCII numbers (-2.6702, 3.01 E -03, -0.002, etc.). All numbers separated by commas will be assigned to cells in a row where numbers separated by carriage returns are assigned to the first column. A combination of delimiters produces a matrix of cells.

Example:

	COLUMNS					
	A	B	C	D	E	
ROWS						
1	1.00,	2.00,	3.00,	2.00,	1.03	<CR>
2	1.01,	2.02,	3.05,	2.07,	2.08	<CR>
3	1.05,	2.09,	3.18,	2.15,	2.11	<CR>

Since 1-2-3 permits 8,000 rows x 256 columns, DATA 6000 records should be columnwize (assuming the default size of 512 points or more). This means that, for a particular record, the point values written to the .PRN file must be separated by carriage returns and not commas.

The basic program listed here performs three tasks:

1. Extracts the waveform data from the D6000

2. Rewrites the values to a 1-2-3 compatible ASCII file

3. Draws the waveform on screen for verification

Once the file is written, it may be archived and/or used as input to the import facility. Should several datasets be needed in one worksheet, the 'combine' facility permits combination of records.

To use the program, press the 'DIR-PROG' key pair to select 'SERØ': as the 'command device' (CMD DEV). Using the 'IO' key, select SERØ: field delimiter as 'comma' (FIELD FORMAT menu), prompt as 'none' (line format menu) and baud rate = 600, half duplex (RS232 parameters menu).

Be sure that you have a 'three-wire' interface (pin 2 to 2, 3 to 3, and 7 to 7). Run the program. You can monitor the progress of the program by observing the graphics screen as it plots the data. When complete, a copy of the waveform will be on disk as <FILENAME>.PRN, the filetype needed for import.

```
10 REM ************************************************************
20 REM  LOTUS 1-2-3 ((R) LOTUS CORP) FORMATTER FOR DATA 6000 RECORDS ****
30 REM *****  (C) 1986 - BILL GARDNER / DATA PRECISION CORP.  **********
40 REM ************************************************************
50 REM
60 REM              ********* IMPORTANT!!! **********
70 REM   In order to operate properly, SER0: must be pre-set to the
80 REM   following values:      RS-232 Menu: BAUD RATE=600, DUPLEX=HALF
90 REM                       Line Format Menu: PROMPT=NONE
100 REM                     Field Format Menu: FLD DLM=COMMA
110 REM
120 DIM ITEM(10) 'Target variable for descriptor items like #points
130 CLS
140 SCREEN 2 'High resolution screen
150 LOCATE 12,30:INPUT "BUFFER NAME";BUFNAM$
160 CLS
170 LOCATE 12,30:INPUT "OUTPUT NAME";FILNAM$
180 CLS
190 OPEN "COM1:600,N,8,1" AS #1 'Adjust D6000 rate via 'IO',RS232
200 OPEN "C:"+FILNAM$+".PRN" FOR OUTPUT AS #2 'Select a filename
210 PRINT #1,BUFNAM$;"?" 'Transfer the buffer descriptor
220 FOR DESC=1 TO 9      'This reads the descriptor.  The
230 INPUT #1,ITEM(DESC) '#points and full scale are saved
240 NEXT                 'in ITEM(3) and ITEM(9) for later
250 PRINT #1,BUFNAM$  'Initiate the waveform transfer
260 FOR PTS=1 TO ITEM(3) 'Sets #points to saved descriptor value
270 INPUT #1,DATUM  'Read a value
280 PSET (PTS*(640/ITEM(3)),-DATUM*(100/ITEM(9))+100) 'Draw the value
290 PRINT #2,DATUM  'Write value to output file (.PRN type for 1-2-3)
300 NEXT
310 CLOSE 1:CLOSE 2
320 END
```

Application Note

Application Note #9

Transferring FFT results to the IBM-PC

ANALOGIC®
DATA PRECISION®
16 Electronics Avenue Danvers, MA 01923 U.S.A.

Spectral data from the D6000 may be transferred to the IBM-PC for further analysis or archiving. A program is included which shows how to do this with the default FFt conditions. Some notes on the program follow:

1) The D6000 is reset in order to place the system in a known state. After acquiring a data set (automatic with default auto trigger) the system is disarmed by sending the ASCII string DARM.

2) The next command sets the line length (for the D6000 output data) to 72 characters and the field delimiter to FLD COL (FLDDLM=1). This will line the data up into neat columns so that it is easy to decode.

3) Next the FFT is performed and the results returned by sending the command FFT.

4) Results are stored in the PC-Basic memory under the name FFT. At this point the data may be further manipulated or archived to disk.

5) This procedure applies to other types of functions on the D6000 such as CONV (for convolution), and Corr (for correlation). Note that the default conditions for the desired operation may be changed prior to the operation. To implement this send the ASCII string for the desired change before performing the operation. For example, if we send the string FFTWDW=3 before sending FFT then the spectrum would be calculated with a HAMMING window instead of the default HANNING window. Similarly the number of points, sampling rate, coupling, etc. may also be changed to optimize the acquisition conditions for a given signal.

```
10 'OUTFFT.BAS
20 '
30 'TO PERFORM THE FFT ON AN AQUIRED D6000 BUFFER AND SEND TO THE PC
40 '
50 'D. ESTRICH    © DATA PRECISION    6-13-86
60 '
70 'REQUIREMENTS:          1) IBM-PC XT  W/PC-DOS 2.00 OR HIGHER
80 '                       2) CEC PC-488 COMMUNICATIONS CARD
90 '                       3) DATA 6000  REV. 4.0 OR HIGHER
100 '
110 '*****  SET UP THE 6000  ******************************************
120 CLS
130 PRINT "RESET THE D6000 AND CONNECT A SIGNAL SOURCE"
140 PRINT ""
150 INPUT "PRESS THE RETURN KEY WHEN READY ",ANS$
160 '
170 '*****  INITIALIZE THE COMMUNICATIONS WITH THE D6000  ***********
180 GOSUB 460
190 '
200 '*****   AQUIRE THE RECORD IN THE D6000 AND DISARM   ***********
210 DEFSNG F:DIM FFT(256)
220 A$="DARM":GOSUB 390
230 'SET UP THE OUTPUT CHARACTERISTICS OF THE 6000
240 A$="LINLEN=72;FLDDLM=1":GOSUB 390
250 'TELL THE 6000 TO PERFORM AN FFT WITH DEFAULT CONDITIONS
260 '  AND RETURN THE RESULT
270 A$="FFT":GOSUB 540
280 '
290 '***** OUTPUT THE RESULTS TO THE SCREEN ************************
300 CLS
310 PRINT "HERE ARE THE FFT RESULTS (in Volts)"
320 FOR I=1 TO 256
330 PRINT I,FFT(I)
340 NEXT I
350 '
360 END
370 '
380 '
390 CALL OUTPUT (D6000,A$,STATUS)
400 CALL ENTER (B$,LENGTH,D6000,STATUS)
410 IF MID$(B$,1,1)=">" THEN RETURN
420 PRINT B$
430 B$=SPACE$(80)
440 GOTO 400
450 '
460 DEFINT A-Z:CLS:A$=SPACE$(80):B$=SPACE$(80)
470 INIT=0:OUTPUT=9:ENTER=21
480 D6000=15:ADDR=21:SYSCONT=0
490 DEF SEG =&HC000
500 CALL INIT (ADDR,SYSCONT)
510 A$="PROMPT=3":GOSUB 390
520 RETURN
530 '
540 CALL OUTPUT (D6000,A$,STATUS)
550 J=0
560 CALL ENTER (B$,LENGTH,D6000,STATUS)
570 IF MID$(B$,1,1)=">" THEN RETURN
580 FOR I=1 TO 8
590 FFT(I+8*J)=VAL(MID$(B$,9*(I-1)+1,9))
600 NEXT I
610 J=J+1
620 B$=SPACE$(80)
630 GOTO 560
```

APPLICATION NOTE

Application Note #100 (Programming)

Using the KEY numbering system on the DATA 6000

ANALOGIC
DATA PRECISION
16 Electronics Avenue Danvers, MA 01923 U.S.A.

INTRODUCTION:

Each key and valid key combination on the DATA 6000 can be accessed using the key codes identified in the operators manual on page F6-89 (and on page 7-26 in the Quick-Look Manual). These codes supplement the flexible programming and remote operation of the instrument by the mnemonic commands listed on pages F6-1 through F6-147 of the operators manual (page 7-15 through 7-25 of Quick Look Manual). The program presented in this Application Note provides retrieval of files from the disk and stores them into system memory.

Example:

This program will work with any plug-in module. It reloads sequentially ordered files from disk into system memory (for post acquisition processing for example). The complete program follows:

```
10 KEY = 2005          (Presses F-Key)
20 KEY = 2045          (Presses XFER-key)
30 KEY = 1006          (Presses two diagonal keys - field 1 - increments
                        system file)
40 KEY = 1066          (Presses ENTER - key - initiates transfer)
50 KEY = 1056          (Presses lower right key - field 4 - increments
                        file name)
```

It requires setting up the transfer menu as follows:

```
press F followed by XFER
  SYS FILE   = BUF.A1
  DIRECTION = ←
  VOLUME     = A:
  FILE       = first file to be transferred
```

To run this program one can use the acquisition step simply to trigger the running of the program. If no external signal is available, the TRIG-MODE may be set to AUTO to insure an acquisition. Upon running this, one will see the Transfer menu and the file names being incremented. This program will also continue until DARM is pressed or until memory is full.

NOTES:

1) In this program, a "new file name" error message may appear on the screen. This is simply a status report and it does not indicate an error.

2) A good habit to develop is to write programs with the plug-in disarmed. The reason for this is that the DATA 6000 powers up with AUTO MODE (for trigger) and AQU DONE (for execute on). This means that the program you are writing may execute prematurely every time a signal is acquired. If the auto trigger is enabled this may happen in the middle of a line and cause unpredictable things to happen. Avoid this difficulty by disarming until ready to run the program.

3) In using the 1000 group keys (soft keys), the key command corresponds to the current definition of that soft-key. That is, line 30 of the program presses two diagonal keys in field 1 which are defined by the transfer menu to be the SYS FILE.

Application Note

Application Note #101 (Programming)

Programming and recording of scalar measurements on a waveform.

ANALOGIC
DATA PRECISION
16 Electronics Avenue Danvers, MA 01923 U.S.A.

GENERAL:

In order to fully utilize the capabilities of the DATA 6000, utility, programs may be written to perform a wide variety of data analysis tasks. Such a program could, for example, perform a series of data reduction operations and store just the necessary results. This application note presents a simple but powerful program for reducing data that measures the frequency and RMS magnitude of an input signal. These results are then stored in another pair of records along with previous results. The program listing and a description of its use is presented.

Example:

The program described here can be easily entered via the front panel of the DATA 6000 while in the DISP mode. The listing below shows the keystroke sequence (assuming DISP has been pressed) and the resulting program line. Since BUF.A1 appears default as the source for trace 1, it will be used in this example.

KEYSTROKE SEQUENCE	PROGRAM LINE
R/S and f simultaneously	-
DARM	-
EX: RMS	10 DSPL RMS (BUF.A1)
EX: FREQ	20 DSPL FREQ (BUF.A1)
EX: TRND RMS	30 TR.RMS = TRND (RMS(BUF.A1),,512,0,0,)
EX: TRND FREQ	40 TR.FREQ = TRND (FREQ(BUF.A1),,512,0,0)
DISP	-

To run this program, press the ARM button. Since programs execute on acquisition done (default value) the DATA 6000 will acquire a sample of 512 points at the default period and then execute the program. The results of the RMS and FREQ operations will be displayed on the top of the screen in real time. In addition, the results will be loaded into two new records named TR.RMS and TR.FREQ. These records may be viewed in the upper and lower traces by selecting them as the sources (under the DISP key - soft fields 2 and 4).

Using this method, the results of any of the scalar measurements performed by the DATA 6000 may be stored in a record and saved to disk, or plotted out. This provides a permanent record of important results of the data reduction scheme implemented.

NOTES:

1) It is suggested that the signal into channel 1 of the plug-in be a sine wave of near full scale amplitude to start. Errors may occur if the frequency drops to low to measure, therefore, start with at least two full cycles on the screen. If an error does occur (probably ERROR #79 - Value out of range), increase the frequency and press CE to clear the top lines.

2) If TR.RMS and TR.FREQ are displayed in the top and bottom traces, one can change the magnitude and frequency of the signal and view the corresponding changes in the trend records. Note that BUF.A1 need not be displayed to perform the programmed operations.

3) After 512 acquisitions TR.RMS and TR.FREQ will start to fill in a FIFO (first in first out) fashion. This is because of the TRND mode used in lines 30 and 40. The other option is to stop after N values have been accumulated. These options may be selected under the TRND key before writing the program line.

Application Note

Application Note No. 200 (Scalar Math Examples)

Measuring the time of a Projected Baseline Crossing

ANALOGIC
DATA PRECISION
16 Electronics Avenue Danvers, MA 01923 U.S.A.

This application first occurred during test of semi-conductor devices. At this time, we do not know of other uses for the solution given here, but it demonstrates the DATA 6000 math flexibility and ease of use for interpolating values and projecting lines. If you uncover a related application, please forward it to the Data Precision Application Engineering Staff.

Figure 1 illustrates the signal. As presented by the user, the baseline went through Ø V, but we have drawn it at +400mV to make the program more general. Note that the rising edge of interest does not cross the baseline. However, the user wants this edge to be located automatically. Then, an imaginary straight line should be "drawn" through the 10% and 90% points of this edge. This imaginary line should be continued until it "intersects" the baseline, and the time of "intersection" (T2) should be noted. Finally, the interval from the negative-going baseline crossing (T1) to T2 should be displayed.

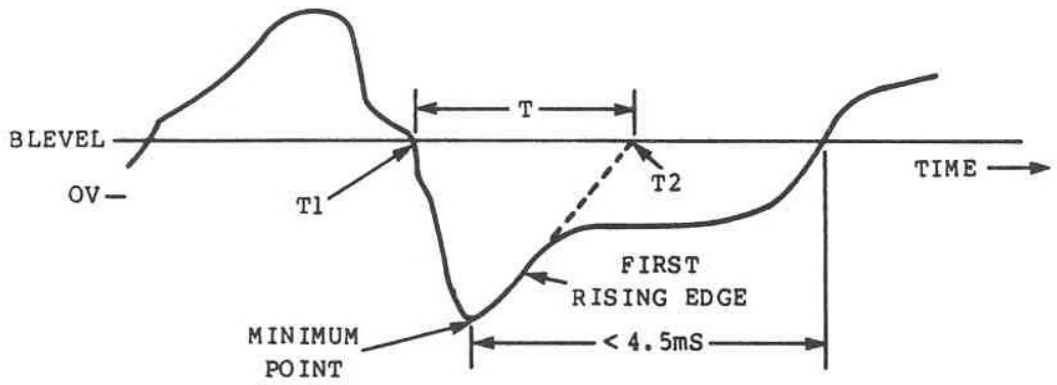


Figure 1. Typical Waveform. The level of the waveform in the region between the first rising edge and the second one is unknown and variable. The time from the minimum point to the next baseline crossing is less than 4.5mS in all test runs.

The objective is to measure the time interval, T. The Algorithm of the solution is:

1) Locate the first, negative going, baseline crossing. Store this time as T1.

2) Locate the rising edge of interest.

3) Find the 10% and 90% points on this edge.

4) Find the slope of the straight line that can be drawn between these two points.

5) Locate the baseline crossing of this imaginary line. Store this time as T2.

6) Subtract T1 from T2, and display the difference.

The enclosed program implements this Algorithm as follows:

Line 10	causes lines 30, 40, and 50 to apply to the waveform in the Top trace (BUF.A1).
Line 20	sets the cursor width to be at least as long as the time from the minimum point to the following baseline crossing (will vary for other applications).
Line 30	Finds and stores the time of the first baseline crossing in the negative direction.

This completes step 1 of the Algorithm.

To locate the first rising edge (Step 2) we will employ a useful trick that applies to many situations. Restricting our attention to the region between the waveform minimum and the next baseline crossing, we see that the rising edge ends at a point of minimum slope in the waveform. Thus, we will use the differential waveform as shown in Figure 2.

Lines 40 and 50	set the cursor between the minimum point (the start of the rise time) and the next crossing.
Line 60	simply stores a time period (two times the timebase sampling period) for later use.
Lines 70 - 90	create the "differentiated" waveform, place it onto Trace 2, and cause the next statements to apply to this trace.
Line 100	moves the start of the cursor forward by two points, to exclude the minimum point from consideration in the next step. (This amount may vary for other applications).

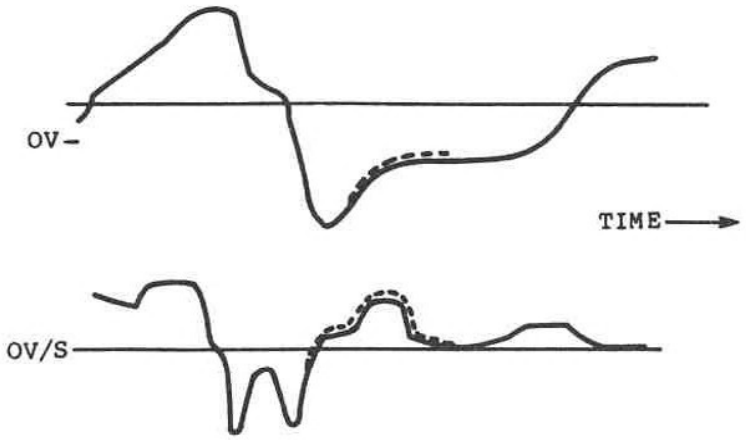


Figure 2. Measured Waveform (top), approximate differential waveform (bottom), and cursor set over first rising edge (dashed line).

Line 110 searches the "differentiated" record, in the region of interest only, for the point of minimum slope, and sets the end of the cursor there. That is, it automatically locates the time at which the rising edge of BUF.A1 plataeus, or stops rising.

Lines 120 and 130 adjust the cursor so that it starts at the same point as before Line 100, and ends at the same point as after line 110. That is, it covers the rising edge of interest.

This completes Step 2 of the Algorithm.

Line 140 directs the following lines to apply to the TRACE that has BUF.A1.

Line 150 sets the cursor to cover from the 10% to the 90% points on the edge found in step 2.

This completes Step 3 of the Algorithm.

Line 160 the slope of a straight line drawn between these points is simply the cursor height divided by the cursor width at this time.

This completes Step 4 of the Algorithm.

Now, to project the intersection of this imaginary line with Baseline, we will use the theory of similar triangles.

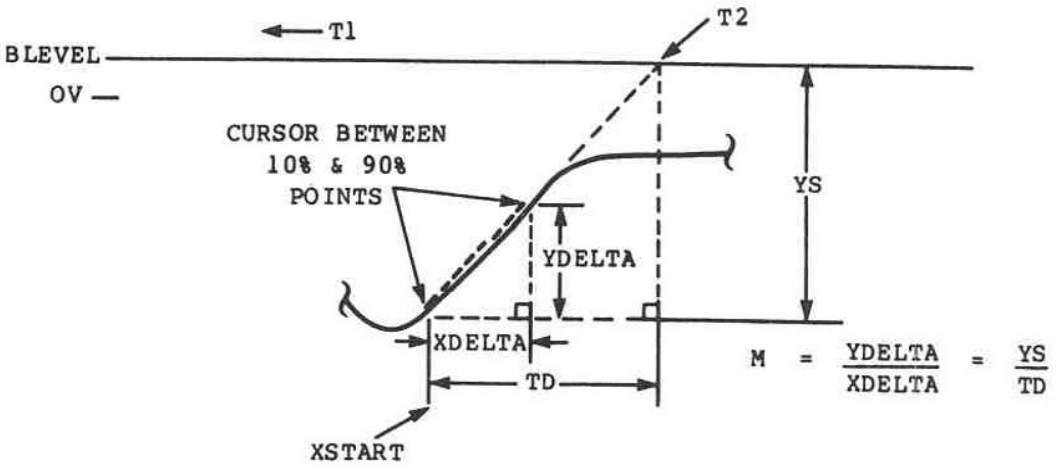


Figure 3. Using similar triangles to project the time of baseline "crossing".

Line 170 computes the absolute value of TRIANGLE side YS, given that the baseline is positive while the cursor starts in the negative region. (This may change in other applications).

Line 180 solves the equation:

$$\frac{YS}{TD} = \frac{YDELTA}{XDELTA} = M, \text{ or } TD = \frac{YS}{M},$$

Line 190 for the unknown triangle side. Dividing M (units of Kilovolts per second) into YS (units of seconds) yields a result in unit of points, which is stored as a temporary variable, Td. Line 190 converts this value to TD (units of seconds).

Line 200 adds TD to the starting time co-ordinate of the cursor, yielding T2, the baseline crossing time of the imaginary line.

Step 5 of the Algorithm is now complete.

Lines 210 and 220 compute and display the desired interval, T.

Step 6 of the Algorithm is now complete.

Program to Project a Baseline Crossing

```
                                        Comments

10 TRACE = 1                            10 Screen marks and modifiers will now
                                           apply to TRACE 1.

20 XDELTA - 4.5mS                       20 Cursor is set wide enough to guarantee
                                           finding the rising edge in Step 110.

30 T1 = BL:SC:CRSN(BUF.A1)              30 Stores the time of the first baseline
                                           crossing in the negative direction.
                                           (If entering this statement from the
                                           front panel, do not type the = symbol).

40 M = SC:MIN (BUF.A1)

50 CP = CR:BL:SE:CRSP(BUF.A1)

60 BU = 2*PERIOD                        60 Stores a "Back-Up" value which will be
                                           used later.  Adapts automatically to
                                           new timebase periods.

70 D=DIFF(BUF.A1)

80 TRCSRC(2) = D

90 TRACE = 2                            90 Screen marks and modifiers will now
                                           apply to TRACE 2.

100 XSTART = XSTART + BU

110 DSPL(CR:SE:MIN(D))

120 XSTART = XSTART - BU

130 XEND = XEND + 2*BU

140 TRACE = 1                           140 Screen marks and modifiers will now
                                            apply to TRACE 1.

150 DSPL (CR:SC:SE:RISE(BUF.A1))        150 Sets cursor between 10% and 90% points
                                            on the rising edge.

160 M = YDELTA/XDELTA

170 YS = 1* YSTART - BLEVEL             170 Use negation symbol (α-shift, unit)
                                            for the first minus sign.

180 Td = YS/M

190 TD = UNIT (Td, 1, Ø, 1, 1)
```

```
200 T2 = TD + XSTART
```

Note: Two math operations in a single statement are recommended only for the D6000 Advanced Operating System.

```
210 T = T2 - T1

220 DSPL (T)
```

Application Note

Application Note #300 (Advanced Math)

The Hilbert Transform on the Data 6000

ANALOGIC
DATA PRECISION
16 Electronics Avenue Danvers, MA 01923 U.S.A.

For sequences with certain properties, the Hilbert transform may be used to relate the real and imaginary parts of the Fourier transform. In addition, the Hilbert transform may be used to construct what is known as the analytic signal. This analytic signal is important when one is interested in obtaining the envelope or instantaneous frequency of a signal.

This write-up describes the Hilbert Transform and its implementation on the Data 6000. A program listing is included which gives the Hilbert transformed signal for any input signal.

INTRODUCTION:

The Hilbert transform of a time function is defined as,

$$H\,[a(t)] = \tilde{a}\,(t) = \frac{1}{\pi}\int_{-\infty}^{\infty} a\,(\tau)\quad\frac{1}{t-\tau}\;d\tau \qquad (1)$$

From this expression, it can be seen that this transform does not change the independent variable and therefore, the output is also a time dependent function. The application of Hilbert transforms is not limited to functions of time though, and one may equally well substitute frequency for time in (1) and perform a Hilbert transform on a frequency domain function. This discussion will center on time domain functions but will use frequency domain techniques to obtain the output represented by (1) above.

METHOD

If (1) is rewritten in the following form,

$$\tilde{a}(t) = \frac{1}{\pi} \; a(t) \; * \; \frac{1}{t} \qquad (2)$$

where * denotes convolution

then we can take the Fourier transform of (2) to obtain

$$\mathcal{F}[\tilde{a}(t)] = \frac{1}{\pi} [A(f)] \; \mathcal{F}\left[\frac{1}{t}\right] \qquad (3)$$

where $\mathcal{F}$ denotes the Fourier transform
$A(f) = \mathcal{F}[a(t)]$

since,

$$\mathcal{F}\left[\frac{1}{t}\right] = \int_{-\infty}^{\infty} \frac{1}{x} e^{-j2\pi f x} dx$$

$$= -i\pi \; \operatorname{sgn} f$$

where $\operatorname{sgn} f = -1, \; f < 0$
$\qquad\qquad\quad\;\; 1, \; f > 0$

then we have for (3)

$$\mathcal{F}[\tilde{a}(t)] = A(f) \; [-i \operatorname{sgn} f] \qquad (4)$$

In the frequency domain then, the desired result is obtained by multiplying the spectrum by i (+90°) for negative frequencies and -i (-90°) for positive frequencies. Performing an inverse Fourier transform then gives the time domain result.

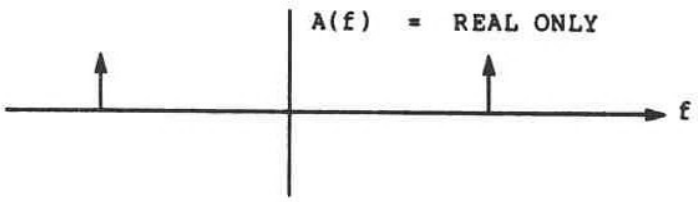
IMPLEMENTATION IN DATA 6000

A simple example is presented here to demonstrate the Hilbert transform.

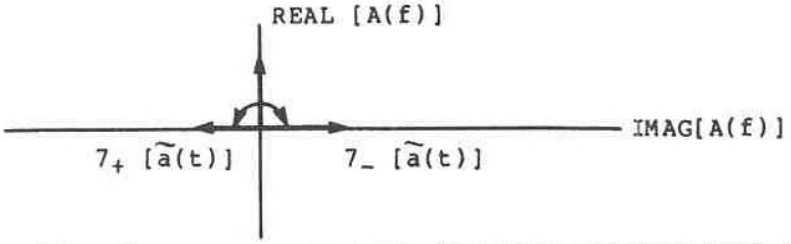
Consider a cosine wave input as follows:

$$a(t) = A \cos(2\pi f t)$$

From the symmetry properties of the Fourier transform we know that A(f) will be real and even because a(t) is real and even. A(f) is represented below:



If we look down the frequency axis, we can easily see the effect of the Hilbert transform on this function:



Here the negative frequency component has been rotated +90° (7_[a(t)]) and the positive frequency component -90° (7_+[a(t)]) to give a purely imaginary and odd spectrum. Again using the symmetry properties of Fourier transforms, this corresponds to a sine wave at the same frequency and with the same amplitude as the input signal. Note that the choice of f was arbitrary therefore, a wide band signal will have all its frequency components shifted by 90° when it is Hilbert transformed.

Since the D6000 uses an FFT algorithm to approximate the Fourier transform, the negative frequency components are represented in the last half of the FFT record. To apply the appropriate rotation for the Hilbert transform, a new record is created as follows:

$Real_H$ = Imag∘ Except that the last half of $Real_H$ is multiplied by -1.

$Imag_H$ = Real∘ Except that the first half if $Imag_H$ is multiplied by -1.

where: $Real_H$ = Real part of 7[a(t)]
$Imag_H$ = Imaginary part of 7[a(t)]
Real∘ = Real part of 7[a(t)]
Imag∘ = Imaginary part of 7[a(t)]

The forward FFT is done with no window and real and imaginary output. The inverse FFT is done with real and imaginary input, no window, and real and imaginary output.

PROGRAM FOR PERFORMING HILBERT TRANSFORM

The two programs below represent the power-on program and the operational program for performing Hilbert Transforms. They will run in any plug-i with a Revision 4 system. The power on program, when stored on disk by the name PWRON.PGM will run on either power up or reset (press R/S and f at the same time). If the disk drive is not available, one can run the second program independently but some set up is required. This set up consists of making a copy of BUF.A1, named F and performing an FFT of F as indicated in line 20 of PWRON.PGM. Then, copies of A may be made and named C and D. It is suggested that BUF.A1 be a sine or square wave of near full scale amplitude to insure A, B, C, and D are scaled properly for running the program. The HILBRT.PGM program may then be entered and run by pressing the R/S key (be sure the program is set to execute on RUN/STOP). If the disk drive is available, simply power on with the disk in drive A: and follow the directions on the bottom of the screen.

The program labelled HILBRT.PGM is set up to first remove the mean of the input and perform the FFT on the resulting waveform F. A and B represent the real and imaginary parts of that FFT. These records are then used to create the real and imaginary parts of the Hilbert transformed signals as described in this note. These records are called C and D. The inverse transform is then computed and the real part of this result is named REALC. This result and the input signal, F, are displayed as a final step in the program. The accompanying plot shows the results of performing Hilbert transform on a square wave.

NOTE:

1) To begin it is suggested that a sine wave of near full scale amplitude be input before loading PWRON.PGM. This will scale the records A, B, C, and D properly. This may be done by powering up the unit and adjusting the amplitude of the input waveform until it is near full scale (with the disk drive lever up). Then power up again with the disk drive lever down so PWRON.PGM is loaded.

2) A simple test case to solidify your understanding of the program is to input a "low" frequency sine wave. Since any plug-in may be used, "low" is defined to be a sine wave of 5 or 6 full cycles on the display. The result of the Hilbert Transform operation (REALC) should be a -cos wave. Note here that end effects will be a problem unless there is an integer number of cycles in BUF.A1.

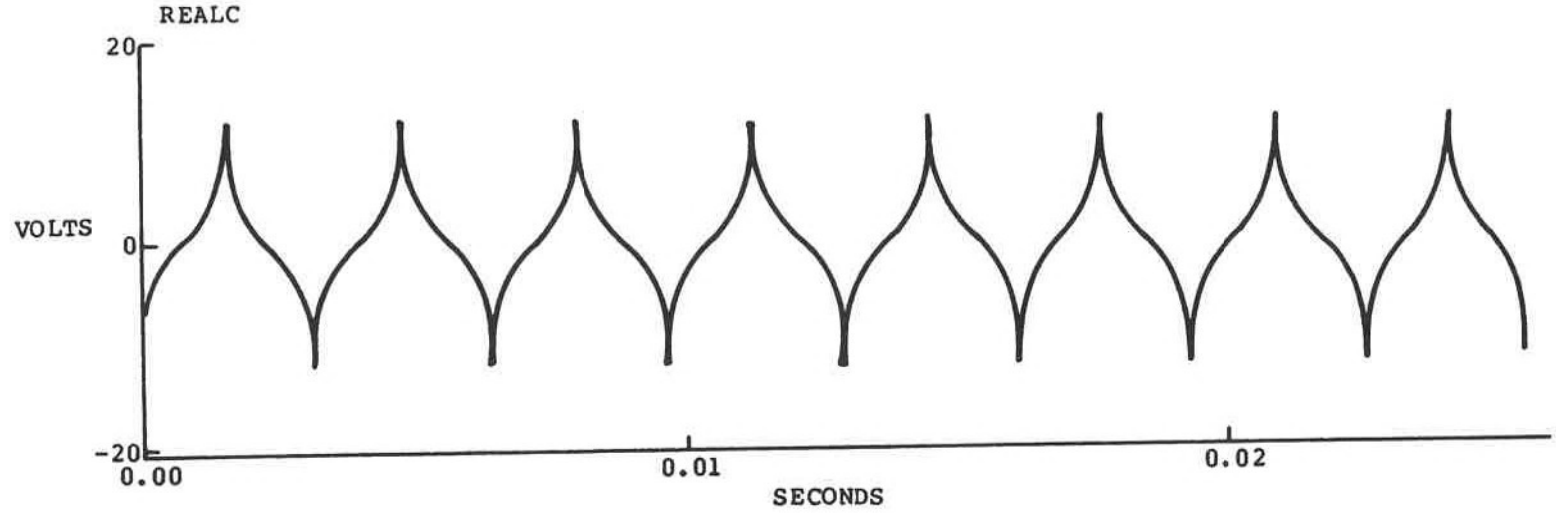
3) After trying a sine wave, it is suggested that triangle and square waves be input to see the results of transforming wide band signals.

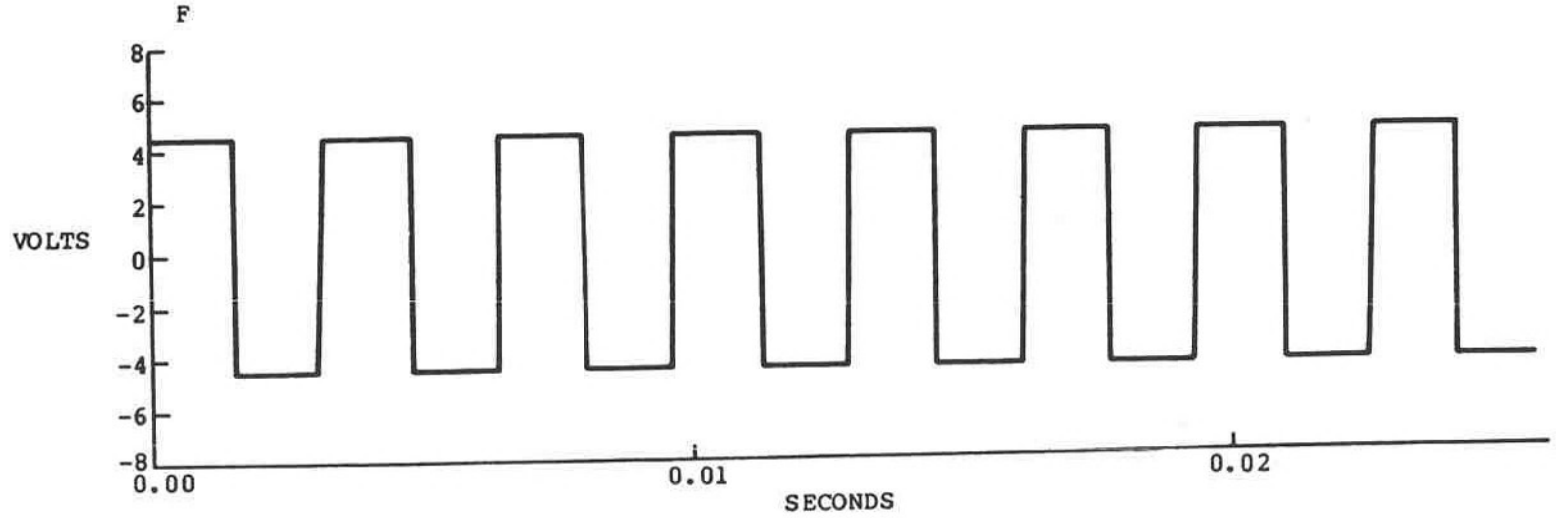
PWRON.PGM

```
 10  DARM
 20  A[B] = FFT (BUF.A1,,0,0,0,17)
 30  C = A
 40  D = A
 50  EXECON = 1
 60  PGMMOD = 2
 70  PROG
 80  DISP
 90  UKEYM = 2
100  LABEL (1,1) = "BUF.A1 DISPLAYED - PRESS ARM TO ACQUIRE SIGNAL"
110  LABEL (2,1) = "PRESS R/S TO RUN"
120  UKEY
130  LOAD ("A:HILBRT.PGM")
```

HILBRT.PGM

```
 10  F = BUF.A1 - MEAN (BUF.A1)
 20  A[B] = FFT (F,,0,0,0,17)
 30  TRCSRC(1) = A
 40  TRCSRC(2) = B
 50  DISP
 60  C(0,256) = B(0,256)
 70  D(0,256) = -1 * A(0,256)
 80  C(257,511) = -1 * B(257,511)
 90  D(257,511) = A(257,511)
100  TRCSRC(2) = D
110  DISP
120  REALC[IMAGC] = FFT(C,D,1,0,1,17)
130  TRCSRC(1) = REALC
140  TRCSRC(2) = F
150  DISP
```


REALC
20
VOLTS
0
-20
0.00
0.01
0.02
SECONDS


F
8
6
4
2
VOLTS
0
-2
-4
-6
-8
0.00
0.01
0.02
SECONDS

Application Note

Application Note #301 (Advanced Math)

Envelope Detection on the DATA 6000

ANALOGIC
DATA PRECISION
16 Electronics Avenue Danvers, MA 01923 U.S.A.

The envelope of a signal is a useful concept in demodulating an amplitude or frequency modulated signal. If a signal is given in the form:

$$a(t) = A(t) \cos \Psi(t)$$

then the following relations may be defined,

$$B(t) = \sqrt{a^2(t) + \tilde{a}^2(t)} \qquad = \text{envelope of } a(t)$$

$$\phi(t) = \arctan \left[\frac{\tilde{a}(t)}{a(t)}\right] \qquad = \text{phase angle of } B(t) \text{ in the complex plane}$$

and $\tilde{a}(t)$ is the Hilbert transform of $a(t)$ defined as follows,

$$a(t) = \frac{1}{\pi} \int_{-\infty}^{\infty} a(\tau) \frac{1}{t-\tau} d\tau$$

In this relationship, $\tilde{a}(t)$ is the harmonic conjugate of $a(t)$. To illustrate this consider Figure 1.

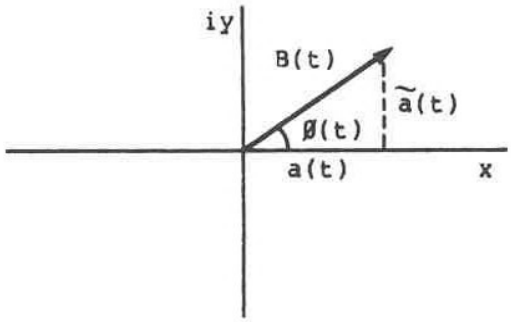


Figure 1.

Here it can be seen that B(t) is the magnitude and ø(t), the phase of the complex valued function a(t) + i$\tilde{a}$(t). The important properties of this representation are that,

1. At points where $\tilde{a}(t) = 0$, B(t) and a(t) have common tangents and,

2. B(t) = a(t) at these same points.

This means that the envelope determined using this method will have the same slope and magnitude of the original signal at or near its local maxima. The example presented below will clarify these ideas.

If $a(t) = \cos W_o t$

then $\tilde{a}(t) = \sin W_o t$ and is the harmonic conjugate of a(t)

and the magnitude of the envelope becomes,

$$B(t) = \sqrt{\cos^2 W_o t + \sin^2 W_o t} = 1$$

therefore, the envelope of a(t) is a straight line with magnitude equal to the magnitude of a(t) and slope equal to the slope of a(t) at its local maxima (i.e. slope = 0 at $W_o t = 0, 2\pi, 4\pi \ldots$).

Note also that $\phi(t) = W_o t$, and is equal to the angle of Figure 1. If we define the instantaneous frequency as

$$W(t) = \frac{d}{dt}\,\phi(t)$$

then it easily seen that for the example,

$$W(t) = W_o$$

The concept of instantaneous freq. is useful in determining the frequency of the modulating component of a frequency modulated signal.

It is noted here that this method is useful only if A(t) is slowly varying when compared to the carrier frequency.

IMPLEMENTATION IN DATA 6000

Two programs are included to demonstrate how an envelope detected signal is obtained on the Data 6000. These programs may be stored on disk and will run on power up provided the first is named PWRON.PGM. If no disk drive is available it is suggested that lines 20 through 40 of PWRON.PGM be executed manually to create the necessary files for the ENVLP.PGM.

ENVLP.PGM is set up to perform the following functions. First, lines 30-60 make a copy of whatever is in trace 1 and call the resulting record F. Lines 70-170 create the Hilbert transformed signal and call it REALC. Lines 210-230 create the envelope squared and call it ENVSQ. The remaining lines are primarily for display purposes and the final display consists of ENVSQ and F on the top and bottom traces respectively.

The attached plot shows the result of performing this envelope detection algorithm on a simulated echo return. This signal may represent a seisimic bottom return, radar reflection or any pulse-echo return. The first trace contains the reflected signal from a multi-level target as a result of a wide band interogating pulse. The second trace is the cross correlation of this echo with the outgoing pulse. The third is the Hilbert transform of trace 2, and the fourth is the envelope detected pulse of ECHCOR.

Several important features are evident here. The first is that trace 4 gives a clear indication of the location of the reflecting surfaces as a function of time. Secondly, noise has been drastically reduced by the correlation processing used. Third, the ENVSQ trace gives quantitative information on the magnitude of the original waveform (ECHO) as discussed in the body of this application note. And finally, note that the program used is simple and performed on a single acquisition and processing instrument.

NOTES:

1) This program will run on any plug-in and a Revision 4 system. The following set-up is suggested to familiarize yourself with the program operation.

   a) Turn the data 6000 on with the disk drive latch up. BUF.A1 will appear in the top trace.

   b) Input a sine wave of "intermediate" frequency and near full scale amplitude into Channel 1. Since the default timebase of the plug-ins varies, "intermediate" is defined as 20-25 cycles on the screen of the DATA 6000.

   c) Now with the disk drive latch down, perform a reset (press R/S and f simultaneously).

   d) Since the signal is already set-up one may simply press R/S and the ENVSQ record will be displayed along with F after the program is done executing.

   e) To run again, remember Trace source 1 must first be changed to BUF.A1 before pressing R/S.

2) For low frequency waves, end effects will be significant and the true envelope squared will not be obtained near those endpoints.

3) The magnitude of ENVSQ is related directly to the magnitude of the input signal as follows:

$$(ENVSQ)^{1/2} = Peak\ (BUF.A1)$$

PWRON.PGM

```
 10  DARM
 20  A[B] = FFT (BUF.A1,,0,0,0,17)
 30  C = A
 40  D = A
 50  EXECON = 1
 60  PGMMOD = 2
 70  PROG
 80  DISP
 90  UKEYM = 2
100  LABEL(1,1) = "BUF.A1 IS DISPLAYED"
110  LABEL(2,1) = "PRESS R/S TO RUN PROGRAM"
120  UKEY
130  LOAD ("A:ENVLP.PGM")
```

ENVLP.PGM

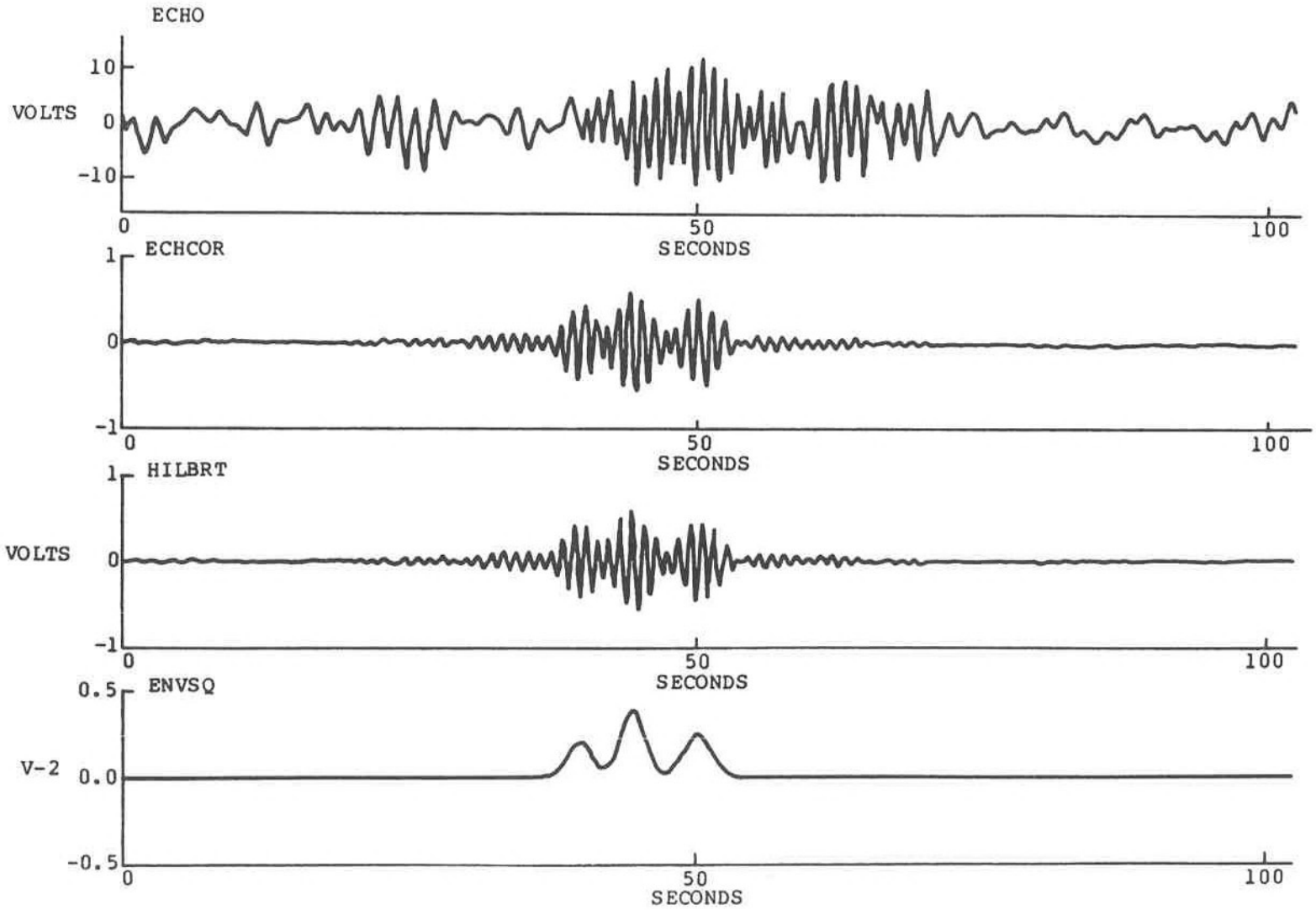
```
 10  MARK
 20  DISP
 30  KEY = 2004
 40  KEY = 2012
 50  KEY = 2004
 60  KEY = 2045
 70  A[B] = FFT (F,,0,0,0,17)
 80  TRCSRC(1) = A
 90  TRCSRC(2) = B
100  DISP
110  C(0,256) = B(0,256)
120  D(0,256) = -1 * A(0,256)
130  C(257,511) = -1 * B(257,511)
140  D(257,511) = A(257,511)
150  TRCSRC(2) = D
160  DISP
170  REALC[IMAGC] = FFT (C,D,1,0,1,17)
180  TRCSRC(1) = REALC
190  TRCSRC(2) = F
200  DISP
210  SQ.LC = SQ(REALC)
220  SQ.F = SQ(F)
230  ENVSQ = SQ.LC + SQ.F
240  TRCSRC(1) = ENVSQ
250  DISP
260  KEY = 2002
```

TRACE 1 = ECHO RETURN

TRACE 2 = CROSS CORRELATION

TRACE 3 = HILBERT TRANSFORM OF ECHCOR

TRACE 4 = ENVELOPE SQUARED OF ECHCOR



APPLICATION NOTE

Application Note #302 (Advanced Math)

Relationships between Signal Amplitudes and MAGC output of FFT

ANALOGIC
DATA PRECISION
16 Electronics Avenue Danvers, MA 01923 U.S.A.

INTRODUCTION:

Relating the results of an FFT operation to a reference voltage is an important step in many instrumentation systems. The output of the D6000 FFT may easily be related to the PEAK value of a waveform by using the MAGC output mode. The procedure is discussed here along with some of the implications involved in the process.

DISCUSSION:

To illustrate the ideas that follow, let's write a program on the D6000. It will be necessary to have a signal generator capable of putting out a sine wave of adjustable frequency in the range of the plug-in you are using. Note that this procedure is plug-in independent and assumes a REV. 4.0 system. Enter the program as follows:

1) press R/S and f simultaneously to reset

2) press DARM

3) press EX: followed by f then FFT

4) change the WINDOW in field 2 to NONE

5) press a key in field 5 (ENTER) to enter the equation to program memory

Note at this point the top line of the screen should read

```
10 MAGCA1 = FFT(BUF.A1,,Ø,Ø,Ø,4)
```

This program will now execute every time an acquisition is made and will calculate the MAGC output of the FFT. In order to see the result, run the program one time by pressing ARM then DARM. Press the DISP key and change the top trace to MAGCA1, the bottom to BUF.A1.

Now connect your signal source (if you haven't already) and set the frequency so that there are an integer number of cycles on the screen of the D6000 (don't forget to press ARM) as you are adjusting note that the resultant FFT record will change. When there is an integer number of cycles in BUF.A1, the MAGCA1 record will contain a single component at that frequency. If you DARM at this point and turn the cursor on at the peak of MAGCA1, you should find that the value read is equal to the peak value of the input waveform. This is an ideal case where the frequency of the signal and the center frequency of the corresponding FFT bin line up exactly. If they don't line up the results are interpreted as follows:

$$PEAK = [E_i \, V_i^2]^{1/2}$$

where PEAK = peak voltage of the input waveform
V_i = voltage of the i^{th} frequency component of MAGCA1
i = index for the summation over several frequency bins.

In words then, the peak value is the square root of the sum of squares of several components. In this case, the energy in the waveform has been spread over several bins in the FFT. To illustrate, ARM the D6000 and change the frequency of the input waveform so that there is an integer number of cycles plus 1/2 cycle on the screen. Note that MAGCA1 will now have a band of non-zero values. One could evaluate equation (1) manually, however, the D6000 may be used by pressing DARM then f followed by ENGY. The requested result appears on the screen but it needs to be modified to get the desired result. First divide by the resolution of MAGCA1 (i.e. the width of one frequency bin), then take the square root. The resultant number will again be the peak value of the input waveform. This process may also be programmed to give an automated result.

To summarize the key points of this note then:

1) MACG output yields the peak value of a waveform when no window is used.

2) If the frequency of the signal is not equal to the center frequency of one of the bins of the FFT, then the energy of the signal is distributed over several bins and the peak value may be found by evaluating equation (1).

APPLICATION NOTE

Application Note #303

Calculating a coherence function using the DATA 6000

ANALOGIC
DATA PRECISION
16 Electronics Avenue Danvers, MA 01923 U.S.A.

Introduction:

The coherence function can be interpreted as the fractional portion of the mean square value at the output which is contributed by the input at frequency f.[1] This function is then useful for determining the effects of uncorrelated noise sources in a linear system. The definition of coherence used here is taken from the reference cited in footnote 1.

$$\gamma_{xy}^2(f) = \frac{|G_{xy}(f)|^2}{G_x(f)G_y(f)}$$

where γ_{xy}^2 = coherence function
$G_{xy}(f)$ = cross-spectral density function
$G_x(f)$ = power spectral density function of the input x(t)
$G_y(f)$ = power spectral density function of the output y(t)

D6000 Implementation

The program used in the D6000 to calculate γ_{xy}^2 is included with this application note. Some assumptions are made regarding its use and they are described below.

1) The user has set up the D6000 to acquire two records simultaneously. The CH. 1 signal (BUF.A1) represents the input of the system and CH. 2 (BUF.A2) represents the output.

2) Copies of BUF.A1 and BUF.A2 are made and identified as IN and OUT respectively.

3) A value for NOISE (in line 110 of the program) has been determined according to the data used. This procedure will be described below.

---

1 Bendat & Piersol, Random Data, Analysis & Measurement Procedures; Wiley-Interscience, 1971

Details of the program are described here.

1) Lines 10-20 perform the FFT on the input and output of the system and use no window. The results are complex valued records containing both the real and imaginary parts.

2) Lines 30-50 create the numerator term $|Gxy(f)|^2$. Note that in line 30, the D6000 automatically conjugates the second term of a complex multiply. Line 40 is simply a "NO FFT" conversion from complex to MAG forms of the spectral data.

3) Lines 60-100 create the denominator terms Gx(f) and Gy(f) as well as the product Gx(f)Gy(f).

4) Line 110 is where noise is added to the denominator to avoid a zero divide situation. NOISE represents another record with the same units as DENOM and therefore may be created as follows. First run the program once before defining NOISE. This will cause an error at line 110 but we may then use DENOM to create NOISE. Press the DISP key and use DENOM as the source for trace 1. Then make a copy of it (called NOISE) by typing

    1) ∝ - shift NOISE ∝ - shift
    2) Press COPY.

    Now look in the DIR and expand DENOM to find the full scale range. Set noise equal to 2 or 3 LSB's by first dividing the full scale range by 2^{16} then type

    NOISE 0 511 = X
    where x is 2 or 3 LSB's

5) Line 120 calculates the final result.

Some hints on using the program.

Start out with a simple case of a square wave input and sine wave output (i.e. a low pass filter) and note the results. For this case there will be a peak at the frequency of the sine wave and very little energy anywhere else in the coherence function. The value of this peak should be very close to 1.00.

Set up the program to run on R/S so that it doesn't execute before you make the copies (IN and OUT). Note that the default condition is to execute programs on acquisition done.

Once you have run some simple cases (try also a square wave input and triangle wave output) use real data. In the case of real data you may want to include two lines at the beginning of the program to make the copies automatically. The program may then be set up to run on acquisition done and therefore run automatically.

```
10 XF = FFT(IN,,0,0,0,8)
20 YF = FFT(OUT,,0,0,0,8)
30 NUM = YF * XF
40 NUM = FFT(NUM,,4,0,2,3)
50 NUM2 = NUM * NUM
60 DENOMX = XF * XF
70 DENOMY = YF * YF
80 DENOMX = FFT(DENOMX,,4,0,2,3)
90 DENOMY = FFT(DENOMY,,4,0,2,3)
100 DENOM = DENOMX * DENOMY
110 DENOM = DENOM + NOISE
120 COHER = NUM2 / DENOM
```

APPLICATION NOTE

Application Note #304

FFT outputs on the Data 6000

ANALOGIC
DATA PRECISION
16 Electronics Avenue Danvers, MA 01923 U.S.A.

Introduction:

The Data 6000 has a variety of output forms for the FFT operation. These choices cover many standard presentations of spectral data. The type chosen is a function of the users application but it is important to keep in mind that all the different FFT output types are derived from a single output. This primary output type is REAL,IMAG and all others are calculated from it by multiplying by an appropriate scale factor. This application note shows the various scalings for each of the FFT output types.

Out put Selections

1.) REAL,IMAG

These are the real and imaginary components of the FFT (ie. the cosine and sine transforms). They are defined mathematically as follows:

$$\left.\begin{aligned} REAL_m &= \frac{1}{N}\sum_{n=0}^{N-1} x_n \cos\frac{2\pi nm}{N} \\ IMAG_m &= \frac{1}{N}\sum_{n=0}^{N-1} x_n \sin\frac{2\pi nm}{N} \end{aligned}\right\} m = 0,1,2\ldots N-1$$

where $REAL_m$ = m^{th} component of the cosine transform
$IMAG_m$ = m^{th} component of the sine transform
N = Number of points acquired (must be 2^n number)
X_n = n^{th} value of acquired record

Note here that the frequency of the m^{th} term of REAL or IMAG is determined by multiplying m by the resolution of the FFT. This resolution is found as follows:

$$\Delta f = \frac{1}{N\Delta t}$$

where Δf = resolution
Δt = sampling period

The frequency of any component then is $m * \Delta f$. With real only inputs, the frequencies above $\frac{1}{2\Delta t}$ will be the complex conjugate of the lower frequencies.

The frequency $\frac{1}{2\Delta t}$ is known as the Nyquist frequency. Half the energy of the input signal will be contained in the data above the Nyquist frequency. The mirror image side is necessary only for inverse transformations.

2.) MAG Magnitude

The magnitude FFT (MAG) is derived from the real and imaginary outputs of the FFT algorithm as follows:

$$MAG = (2)\sqrt{(REAL)^2 + (IMAG)^2}$$

The frequencies above ½/period are omitted (with real only inputs) since no FFT algoritm can determine frequencies above ½/sampling rate. Mathematically, however, those frequencies contain half the energy. Hence, the (2) in the above equation compensates for the FFT trunkation. Power and amplitudes of the input frequency components can determined from this magnitude FFT once the effects of the windowing and number of points are computed (See PWR SPCTRM and MAGC).

3.) MAG COEFF Magnitude Coefficient

The magnitude coefficient gives the peak value of a frequency component of the input waveform. For example: A 1V peak sinusoidal input will have an FFT with a magnitude coefficient of 1V. The magnitude coefficient compensates for the effects of windowing and the number of points used. The magnitude coefficient is related to the MAG (Magnitude) by the following equation:

$$MAGC = \sqrt{\frac{W2/N}{W1}}\ MAG$$

where W1 and W2 are the windowing factors givin in TABLE 1 below:

TABLE 1

WINDOW	(AVERAGE) W1	AVERAGE SQUARED W2
Non or boxcar	1	1
Hanning or Cosine	.5	.375
Hamming	.54	.3974
Cosine Taper	.875	.8438
Sawtooth or Triangle	.5	.333
Sine	.6366	.5
Sine cubed	.4244	.3125
Sine fourth or Hanning squared	.375	.2734

See Also Application Note #302 for a further discussion of MAGC.

4.) PWR SPCTRM

This is the output that corresponds to a normal power spectral denisty. The D6000 uses the same definition presented in Ref. 1 and that definition is:

$$\text{PWR SPCTRM}_m = \frac{MS}{\Delta f}$$

where PWR SPCTRM_m = m^{th} component of record
MS = mean square value of input signal in m^{th} component
Δf = resolution of FFT (see above)

The power of a waveform can be found by executing the PWR SPCTRM FFT and then pressing AREA (a function key on the math keyboard). A window of NONE should be used. Power calculated will be found to equal the power caluclated by squaring the RMS value obtained on the input waveform.

Applying the cursor to only a small frequency range and calculating the area under the cursor will give the power for the specified range of frequencies.

Integrating the PWR SPCTRM FFT will give a maximum amplitude which also equals the waveform power as calculated by pressing the AREA key.

5.) MAGS

This is related to the MAG record as follows:

$$MAGS = MAG \sqrt{\omega 2 * \Delta t}$$

where $\omega 2$ is defined in Table 1
Δt = sampling period

6.) PSD

This is not to be confused with PWR SPCTRM (which is the more standard form of a power spectral denisty) and is defined in terms of the MAG record as follows:

$$PSD = \frac{(MAG)^2}{2}$$

7.) ALOG

This output yeilds the power of a signal in a logarithmic form. The D6000 assumes that this voltage is applied over a 500Ω resistance. The expression used to calucate the ALOG record is as follows:

$$ALOG = 20 \log (MAGS) + 27 \text{ dB}$$

Note here that the 27 dB correction accounts for the 500Ω resistance.

Since the output is in dB re 1 watt rms/HZ (dB) or one milli-watt rms/HZ (dBm) @ 1Ω , the power re 1 watt rms (or 1 mW rms) may be found by adding 10*log (Δf). Here Δf corresponds to the resolution of the FFT.

8.) RLOG Relative Log

The relative log can be derived from the magnitude spectrum as follows:

$$RLOG = 20 \text{ LOG } (MAGS)$$

The maximum apmlitude is adjusted to be 0dB.

9.) REAL

For a forward transform this is the same as the REAL part of REAL,IMAG. For an inverse transform it is the time domain record resulting from the evaluation of:

$$x_m = \frac{1}{N} \sum_{n=0}^{N-1} X_n e^{j\frac{2\pi nm}{N}} \qquad m = 0, 1, \ldots N-1$$

where x_m = time domain output
X_n = Complex fourier coefficients (REAL + jIMAG)

10.) COMPLEX

The complex FFT contains the information in both the real and imaginary FFT. This facilitates mathematical manipulation. Multiplication, for example, will simultaneously multiply both the real and imaginary components. Division, addition, and subtraction similarly operate on both real and imaginary components.

Only the REAL part of the complex FFT is displayed.

11.) POLAR

The polar format is derived as follows:

$$\text{POLAR} = \sqrt{(\text{REAL})^2 + (\text{IMAG})^2} \quad \angle\theta$$

The angle θ has been computed to be the angle between the real and imaginary components. This angle is not calculated in POLAR output mode.

The other output options consists of the same spectral choices only with phase included. The phase records are calculated from the real and imag parts of the FFT as follows:

$$\text{PHASE}_m = \tan^{-1} \frac{\text{IMAG}_m}{\text{REAL}_m}$$

where PHASE_m = M^{th} component of phase record

The phase records are limited to ±180°.

References:

Ref. 1: Bendat + Piersal, Random Data: Analysis + Measurement Procedures, Wiley-Interscience, 1971.

APPLICATION NOTE #400

An example where the 650 plug-in was used in measuring high dynamic range and high frequency content signals in non-destructive testing.

ANALOGIC®
DATA PRECISION®
16 Electronics Avenue Danvers, MA 01923 U.S.A.

The difficulty in measuring signals is often compounded by the nature of the signal itself. This is particularly true when non-repetitive, large, one volt peak to peak signals contain important information in the microvolt region with frequency components of up to 200 KHz. Such signals are often encountered in seismic, acoustic, and non-destructive testing.

Data Precision's newest Model 650 Plug-In coupled with the D6000 Waveform Analyzer addresses such measurements. With its 16 bit converter, 64K points of memory and sampling rates of up to 1 MHz, signals in excess of 90dB of dynamic range with frequency components of up to 250 KHz can now be captured and analyzed on a single input range.

As an example, (See Figure 1), a 150 KHz, 1V p-p pulse was used to excite a non-homogeneous sample in order to observe its propagating and attenuation properties. The 650 plug-in was used to capture both the excitation and returns using only one of its five input ranges.

The example shows that the first Echo is attenuated by a factor of 17 after 70 usec followed by a second and third returns at 154 usec and 413 usec respectively with significantly greater attenuations.

Figure 2 shows Echo 3 signal with information content in the microvolt region. Using averaging capability of the D6000 and comparing the two traces parameters such as transducer noise, propagation delay and decay characteristics, and frequency can easily be determined.

As in the case of other D6000 plug-ins, the 650 simply attaches to the D6000 to make it a measurement instrument with unprecedented capabilities in dynamic range, frequency, and processing combinations.

EXCITATION AND RETURN SIGNALS CAPTURED
BY THE 650 PLUG-IN ON 1V PP INPUT RANGE

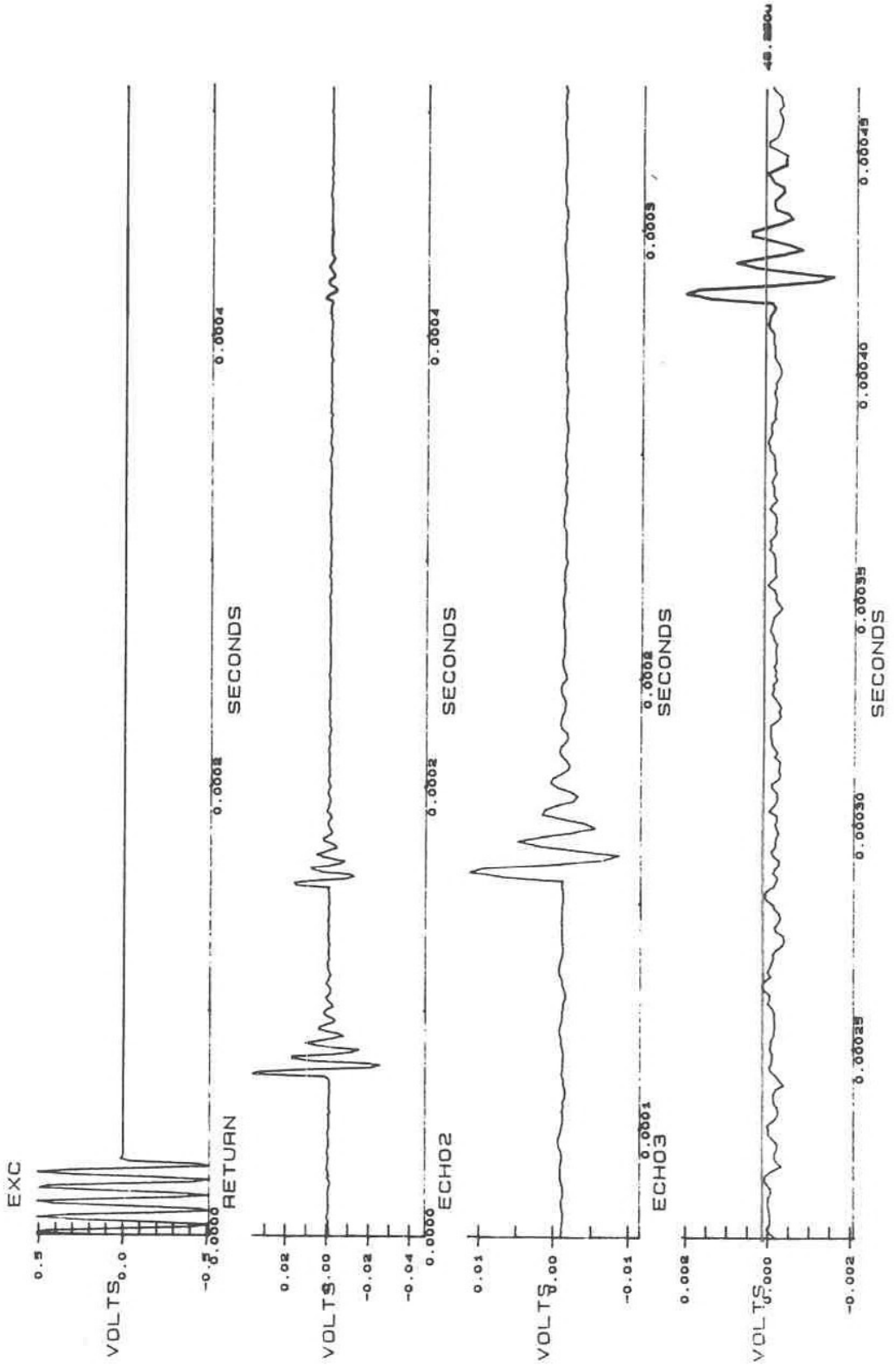
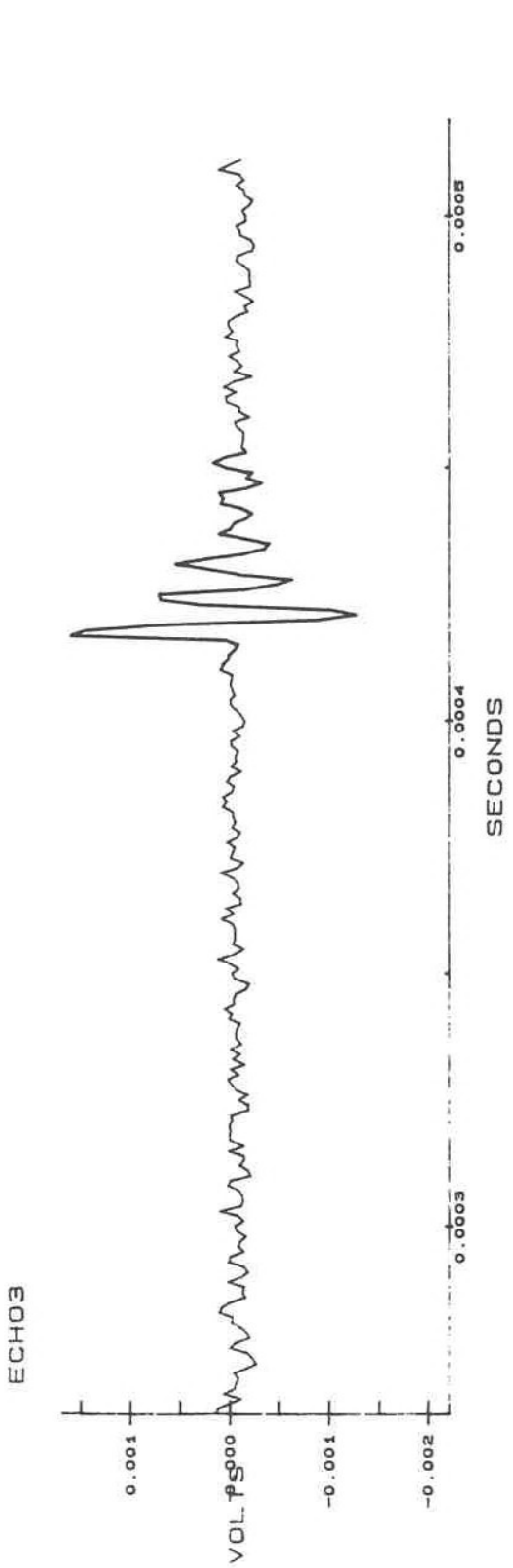


FIG 1


ECHO3
VOLTS
0.001
0.000
-0.001
-0.002
0.0003
0.0004
0.0005
SECONDS

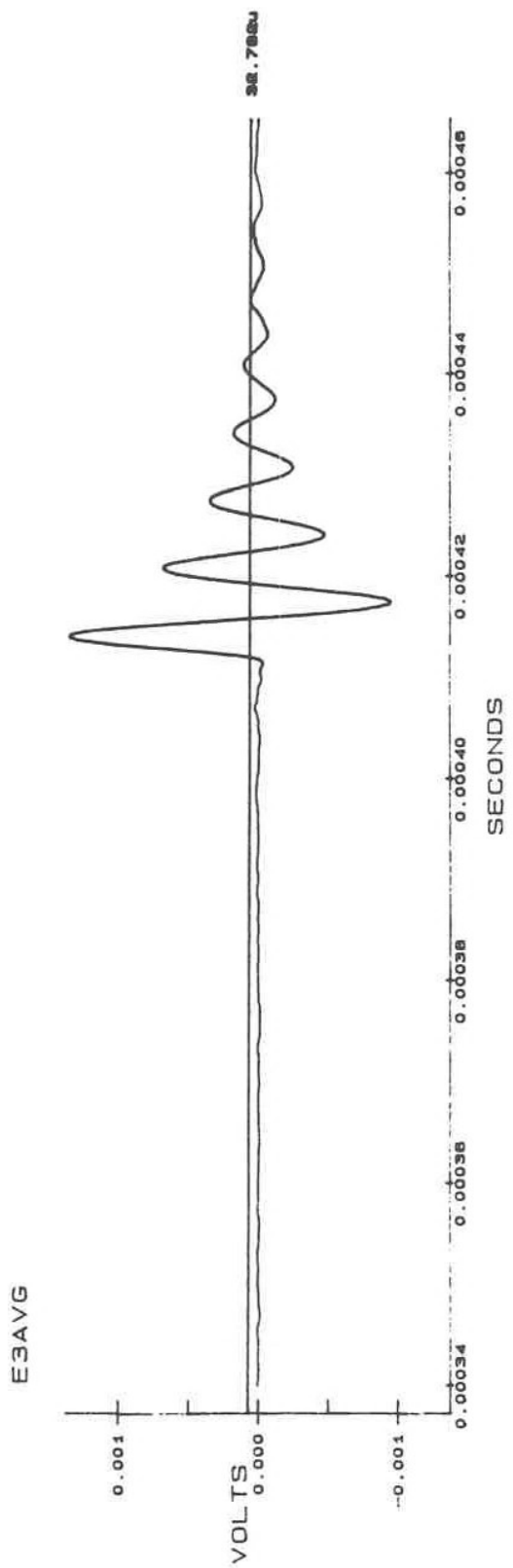

E3AVG
VOLTS
0.001
0.000
-0.001
0.00034
0.00036
0.00038
0.00040
0.00042
0.00044
0.00046
SECONDS

FIG 2

Application Note

Application Note #401

Procedure to turn on the calibration output from the Model 630, 36MHz digitizing, plug-in. (Rev. 4.01 630 in Rev. 4.00 D6000)

ANALOGIC®
DATA PRECISION®
16 Electronics Avenue Danvers, MA 01923 U.S.A.

Connect the "Cal"Output on the 630 front panel to the Channel 1 input connector. Then:

1. Reset (F-Shift and R/S).

2. Press INP and FLTR on the 630 both at the same time to get CAL signal menu.

3. In field 1 of this menu (CAL Out) press the bottom right button (CHOP,...).

4. Press TMB on PLug-In.

5. Press bottom right button in field 3 until period equals 500 microseconds. You should see a waveform that has 3 levels: minus full scale, 0V, plus full scale, 0V, minus full, 0V, etc. (You are still auto-triggering).

APPLICATION NOTE

Application Note #500 (User Tips)

Storage of Transient Waveforms in One Large Record

ANALOGIC
DATA PRECISION
16 Electronics Avenue Danvers, MA 01923 U.S.A.

INTRODUCTION:

Capturing and storing transient waveforms is a common application of the Data 6000. This application note describes a program to be run on the DATA 6000 that provides an efficient means of storing successively captured transients. This example is designed to show the user the following:

1) How to initialize a record in the DATA 6000.

2) How to transfer data points of one record into other records.

Example:

Consider the case of a 64 point acquisition of a transient waveform in BUF.A1 (it is assumed that the user can set up the TRIG and TB parameters to achieve this results). It is desired to create a new record which is 6400 points long where points 0-63 are the points of the first transient acquired, points 64-127 are the points of the second, etc. With this scheme then we will be able to acquire 100 transients and store them in a single record.

To perform this task, the program shown below should be entered and set up to execute on acquisition done (AQU DONE under the RUN mode of the PROG key). Prior to entering this program however, we must initialize the record BIG.A1 as well as the counters, A and B used in the program.

BIG.A1 is initialized by typing the following sequence:

```
BIG.A1 @W 6400  V S O  5E-5  0  10.24 = 0.0
```

This sequence has the following general form,

```
NAME  @W  NPTS  YUNITS  XUNITS  XOFF  X/POINT  YOFF  YFSCL=VALUE
```

```
Where  NAME     = name of new record
       @W       = waveform
       NPTS     = number of points in NAME
       YUNITS   = units of Y-axis
```

```
XUNITS    = units of x-axis
XOFF      = offset of x-axis
X/POINT   = number of XUNITS per point
YOFF      = offset of Y-axis
YFSCL     = full scale range of Y-axis
VALUE     = initial value of all points of NAME
```

Therefore, we have created a waveform named BIG.A1 with 6400 pts., 10.24V full scale, 50μ Sec. /point, and no X or Y offset.

To initialize the counters simply type,

```
A = 0
B = 63
```

The reason for using these values will be obvious after looking at the program. The program consists of the following three lines:

```
10  BIG.A1  A  B = BUF.A1
20  A = A + 64
30  B = B + 64
```

So for the first acquistion,

```
BIG.A1  0  63 = BUF.A1
```

and the second,

```
BIG.A1  64  127 = BUF.A1
```

and so on. After the 100th transient has been stored, the program will try to store the 101st transient, however, the record BIG.A1 is not large enough. This will produce an error message that halts program execution. To run it again this error must be cleared by pressing DARM then PROG and with MODE = RUN (2nd soft-key field) press CONT. A and B must also be re-initialized to 0 and 63 respectively. Then when ARM is pressed, BIG.A1 will be reassigned the results of the next 100 transient captures.

NOTES:

1) When assigning the values of one Buffer to the values of another, one must be careful of both the X and Y scales. If the X/point of the buffers is different, there will be no error indicated. Effectively then, the X/point of the source buffer is changed to that of the destination buffer. Y-values are transferred with a change of scale therefore, some loss of precision may result. For example, if BUF.A1 has a full scale range of 2.56V and BIG.A1 has a full scale range of 25.6V then the quantization steps in BIG.A1 are ten times the quantization steps in BUF.A1 and BUF.A1 will not be as accurately represented in BIG.A1. Note also that if a value in BUF.A1 exceeds the max value allowed for BIG.A1, an overflow will occur. Again, no error message appears for this error and it is the users responsibility to insure BIG.A1 has the appropriate scale factors.

2) The reverse of building a large buffer from many small ones (that is, splitting a large buffer up into several small ones) may most easily be accomplished by setting the cursor length to be equal to the length of a small section and executing a cursor copy (CR:COPY) on successive segments of the large buffer. This is often a useful post processing tool.

3) In order to see the program in process, it is suggested that BIG.A1 be placed in the bottom trace of a 2 trace display and an X scale of 1/4 be used (press X and change X Scale for trace 2). When the ARM key is pressed to begin program execution, you will see BIG.A1 being constructed as BUF.A1 is acquired. It is suggested that BIG.A1 be zeroed out before re-running the program so that the new data input is obvious. This may be done by typing in the following statement.

```
BIG.A1  0  6399 = 0.0
```

4) After mastering the material presented thus far, one additional enhancement to the procedure will add flexibility. That is, instead of typing in numbers for equation parameters, one may use variables. For example, lines 20 and 30 of the program presented above contain the number 64. This could be replaced by the variable N where N is defined to be 64 or 128 or whatever you want. Note that N must be defined prior to executing the statement. This applies to other statements also, such as the initialization sequence and the statement presented in Note 3 above (i.e. BIG.A1 0 N = 0.0).

Application Note

Application Note #501 (User Tips)

Maximizing Acquisition Lengths on the D6000 Plug-Ins

ANALOGIC
DATA PRECISION
16 Electronics Avenue Danvers, MA 01923 U.S.A.

INTRODUCTION:

Applications of the D6000 to acquisition of transient signals often require maximizing the length of the acquired record. The procedure for obtaining the maximum length record for each of the plug-in modules is the subject of this application note. It describes this procedure as well as the limits on the record length for each of the plug-ins. It is assumed that the extended memory is installed (Model 686) and that the software revision level is 4.0 or greater.

Background:

Data is digitized and stored in an intermediate memory segment called a frame before it is transferred to a buffer. A frame may not be accessed directly by the user and therefore is not available for computation and/or storage. Buffers are accessible by the user and represent a contiguous segment of data taken from the frame. Buffers appear in the directory (press DIR) when the system is turned on. A particular segment of the frame is chosen with the buffer offset and buffer length parameters. The following is a description of this procedure for each of the 4 plug-ins now available for the D6000.

610

1) Press BUFR and PROC simultaneously.
2) Set BUF LEN (2nd soft-field) to 512.
3) Press BUFR.
4) Set INPUT (3rd soft-field) to CH2 and MODE (4th soft-field) to OFF.
5) Press TMB. (see Note 2).
6) Set # POINTS to 30720.

At this point the plug-in should be armed and the arm light flashing slowly (about once every 2 sec for PERIOD = 50 uS). In addition, the signal on the screen will be displaying the first 512 pts of the acquired record in BUF.A1. To see the rest of the acquired record, press DARM then BUFR and PROC simultaneously. Change RECORD (field #1) to BUF.A1 and then increase the BUF OFFS (field #3) using the upper right key. The waveform will slide slowly to the left as the "window", through which the acquired record is

viewed, changes. Note that the BUF Len parameter may also be changed but because of memory limitations can only be 16384 maximum. This could be predicted (before changing BUF LEN) by looking at the contiguous memory available (press DIR and OPT simultaneously). Depending on the system configuration, this number should be approximately 17024.

611

1) Press BUFR and PROC simultaneously.
2) Set BUF LEN (2nd soft-field) to 512.
3) Press BUFR.
4a) Set BUFR INPUT to CH2 and MODE to OFF.
4b) Set BUFR INPUT to CH3 and MODE to OFF.
4c) Set BUFR INPUT to CH4 and MODE to OFF.
5) Press TMB (see Note 2).
6) Set # POINTS to 30720.

The description following the 610 setup applies here as well with the folowing exception. If contiguous memory is checked it will be about 14848 therefore the largest BUF LEN allowable in this set-up is 14336.

620

1) Press TMB.
2) Set # POINTS to 16000

at this point the plug-in arm light should be flashing slowly (about 1/sec). Note that we did not have to set the buffer length for the 620 (as we did for the 610 and 611). This is because the 620 has its own acquisition memory and does not use mainframe memory for this purpose. The 16k point limitation then is due to the size of this plug-in acquisition memory. We can still view windowed segments of this acquisition by pressing BUFR and PROC simultaneously and setting BUF LEN to 512. If BUF OFFS is then changed, the waveform will be seen sliding through the window.

Note here, that because of the plug-ins acquisition memory, two acquisitions may be made simultaneously, each one consisting of 16,000 data points. If one is clever then, a 32,000 pt acquisition may be made by assigning timebase A to channel 1 and timebase B to channel 1 (press BUFR key) and dialing in a delay to timebase B corresponding to the length of timebase A. For example, if the default period of 20nS is used, the delay would be:

DELAY = 16000 X 20nS
= 320 uS

This yields two records which represent a continuous sampling of the waveform over 32000 pts.

630

1) Press TMB
2) Set # POINTS to 8000.

Here, as with the 620, the acquisition memory of the plug-in allows the full acquisition to be viewed in the buffer. In addition the BUF LEN and BUF OFFS fields (press BUFR and PROC simultaneously) may be used to "slide" through the acquired record and look at specific segments of it.

As with the 620, one continuous acquisition may be made under two different timebases by inputing the appropriate delay in timebase B. For example, in the case of a 8000 pt timebase with the default period of 100nsec,

DELAY = 8000 (1000nS)
= 800 uS

The result is a 16000 pt waveform this is stored in two separate records.

Summary

Table 1 lists the results of the previous sections of this application note.

TABLE 1

PLUG-IN	MAXIMUM ACQUISITION LENGTH # POINTS
610	30720
611	30720
620 - 1 Timebase	16000
- 2 Timebases	32000
630 - 1 Timebase	8000
- 2 Timebases	16000

NOTES:

1) It is important not to confuse the BUF OFFS parameter with X OFF (press X key). Although the effects of changing these two parameters appears, in some cases, to be the same on the screen, their actual effects are very different. As explained in this note, BUF OFFS changes the segment of the frame that has been acquired. In contrast XOFF simply changes the segment of the buffer that is displayed on the screen.

2) For the 610 and 611 plug-ins with Revision 4.0 software, # POINTS should not be allowed to exceed 30720 pts because of system resets.

APPLICATION NOTE

Application Note #502 (User Tips)

Reading and Plotting Data from D6000 Disks on the IBM-PC

ANALOGIC
DATA PRECISION
16 Electronics Avenue Danvers, MA 01923 U.S.A.

INTRODUCTION:

Since D6000 disk formats are compatible with the IBM-PC systems, the D6000 data stored from an experiment may be pulled off that disk and analyzed using the IBM. This application note describes a program that reads a D6000 disk and converts the binary data to real numbers. In addition, the program contains a plot option so that the user can make a quick check of data integrity. It is written for use with PC-DOS 2.0, IBM Personal Computer Basic Version D2.00, and a disk written by a Revision 4, D6000. A program listing and a description of the disk file formats are included.

Program Description

Generally speaking, the programs purpose is to read binary data from the disk and convert them to real numbers. The added feature of plotting is strictly for illustrative purposes and may just as easily be replaced by a data processing subroutine specific to your application.

Since the program is commented, this discussion will center on features and "helpful hints" pertaining to reading disk files.

1) A disk file consists of a 36 byte header followed by the data. Since the data is represented by 16 bit integers, the length of the data area is NPTS X 2 bytes where NPTS is the number of points in the record.

2) The 36 byte header consists of important record information (see attached sheet on Formats of Binary Data) which is used to translate the binary data into meaningful numbers. The most important of these is probably the Y full scale range (YFSR). The information presented in this header is translated according to the data type. For example, the name is stored as ASCII data and may therefore be translated by reading each one of the bytes (9 total) corresponding to one character of the name and using the ASCII code to convert to the character.

If the first character of the name were A, then, using the PEEK command would give a value of 41. Looking at an ASCII Table, we see that 41 corresponds to the character A. In order to convert IEEE floating point numbers to real numbers note that lines 340-430 perform this operation for each of the four real values used in the header.

3) X units and Y units are defined as byte types and are numbered according to their position in the UNITS function. See the accompanying table for a summary of the codes for various units. In the program, the unit codes are read from memory in line 220. Note that this program only defines 3 codes in line 130. If other units are expected, they need to be defined here.

4) Response to the question of line 170 is A or B depending on where the disk is. Response to the question in line 180 is one of the .VAR file names listed in response to the previous question. It is typed exactly as it appears on the screen without the .VAR extension.

5) The IBM reads files from disk in blocks of 128 bytes. This is a default condition and may be changed by appending the BASIC command with an extension (see the IBM Basic Manual).

6) Header information is read in and translated in lines 210-430. Note that not all of the parameters of the header are needed or translated in this section. The most important number obtained here is FSR (line 410: FSR in program = YFSR in D6000). This sectio also dimensions two arrays, FARRAY and C, after determining the number of points in the record. FARRAY is the real valued array which will contain the translated data points. C is the array used for plotting.

7) Data is read and translated in lines 480 to 670. Note that some of the data has already been read in from executing the first GET statement (remember that 128 bytes were read but only 36 were header information). Lines 540-570 are the heart of the translation procedure and show a single conversion from the binary number to real number. After this loop is done the entire data record will have been stored in FARRAY. Note that contrary to the disk record which contained header and data, this array only contains data. The remaining information is stored under other variable names in the program.

8) This plot array is defined in lines 610 to 630. Note that M in line 610 was the integer value of the number before scaling. This is done to accommodate a large range of values. The maximum and minimum values of the record are also stored and displayed (after correcting for the scale factor) to give the operator an idea of scaling when the plot option is exercised.

9) The remainder of the program performs a listing of the data and/or a plot if desired. Plots are performed by the LINE statement in line 910. Note that the IBM is only capable of plotting 640 points horizontally. If desired, this part of the program could easily be modified to plot every N th point (to scale the X-axis) and/or add enhancements such as an axis, grid, etc.

```
 10  'DSKPLT.BAS
 20  '
 30  'CONVERTS .VAR FILES CREATED BY D6000 INTO REAL NO. FILES AND PLOTS ON
     SCREE
 40  '
 50  'DOUG ESTRICH 7/23/85 @ DATA PRECISION
 60  '
 70  'READS REV. 4.0 FILES FROM D6000
 80  'USES PCDOS 2.0 OR LATER AND IBM PERSONAL COMPUTER BASIC VERSION D2.00
 90  '
100  KEY OFF:KEY 10, "SYSTEM"+CHR$(13):PS$=" +##.######"
110  DIM XYNAME$(4):XYNAME$(1)="X PER POINT = ":XYNAME$(2)="X OFFSET  ="
120  XYNAME$(3)="Y FS RANGE  =  ":XYNAME$(4)="Y OFFSET  ="
130  DIM UNIT$(4):UNIT$(0)="  NONE":UNIT$(1)="  SEC":UNIT$(2)="  VOLTS"
140  DEFINT A-Z
150  DEFDBL M:DEFSNG F
160  '
170  INPUT "WHAT DRIVE ARE THE FILES IN";D$:FILES D$+":"
180  INPUT "WHAT .VAR FILE DO YOU WANT TO SEE ";FLNAME$
190  '
200  ' READ THE FIRST BLOCK OF 128 BYTES FROM DISK
210  OPEN D$+":"+FLNAME$+".VAR" AS #1
220  GET #1:FC=VARPTR(#1):XUNIT=PEEK(FC+202):YUNIT=PEEK(FC+203)
230  '
240  'CALCULATE FILE SIZE AND X AND Y UNITS - PRINT THEM OUT
250  FLSIZE=LOF(1)-36:IRMAX=FLSIZE/128
260  NPTS=FLSIZE/2:DIM C(NPTS):CMIN=32767:CMAX=-32768!
270  DIM FARRAY(NPTS)
280  MMIN=100:MMAX=-100
290  PRINT "TOTAL # OF DATA POINTS=";NPTS:IC=0
300  PRINT "X UNITS    =";XUNIT;:PRINT UNIT$(XUNIT)
310  PRINT "Y UNITS    =";YUNIT;:PRINT UNIT$(YUNIT)
320  '
330  'READ THE HEADER INFORMATION AND TRANSLATE IEEE FP TO REAL NUMBERS
340  FOR I=0 TO 3
350  X=I*4
360  A=PEEK(FC+204+X):B=PEEK(FC+205+X):C=PEEK(FC+206+X):D=PEEK(FC+207+X)
370  IF A=0 AND B=0 AND C=0 AND D=0 THEN FP=0:GOTO 420
380  S=A AND 128
390  E=(A AND 127)*2+(B AND 128)/128-127
400  M=(((B AND 127)+128)*2^16+C*2^8+D)/2^23
410  FP=(-1)^S*2^E*M:IF I=2 THEN FSR=FP
420  PRINT XYNAME$(I+1);FP
430  NEXT I
440  '
450  'READ DATA AND TRANSLATE TO REAL NUMBERS - LOAD INTEGERS INTO C FOR
     PLOT
460  '  AND INTO FARRAY FOR USE AS DESIRED
470  PRINT "GETTING THE DATA - PLEASE WAIT"
480  ICOUNT=0
490  FSF=FSR/2^16
500  FOR IR=1 TO IRMAX+1
510  IF IR=1 THEN IMIN=36:GOTO 530
```

```
520  GET #1,IR:FC=VARPTR(#1):IMIN=0
530  FOR I=IMIN TO 126 STEP 2
540  A=PEEK(FC+188+I):B=PEEK(FC+189+I)
550  M=A*256+B
560  IF M>32767 THEN M=M-65535!
570  MVAL=M*FSF
580  IF MVAL<MMIN THEN MMIN=MVAL
590  IF MVAL>MMAX THEN MMAX=MVAL
600  FARRAY(ICOUNT+1)=MVAL
610  C(ICOUNT+1)=100-INT(M/300)
620  IF C(ICOUNT+1)<CMIN THEN CMIN=C(ICOUNT+1)
630  IF C(ICOUNT+1)>CMAX THEN CMAX=C(ICOUNT+1)
640  ICOUNT=ICOUNT+1
650  IF ICOUNT>NPTS-1 THEN 680
660  NEXT I
670  NEXT IR
680  '
690  'PRINT DATA IF DESIRED
700  INPUT "DO YOU WANT A LISTING OF THE DATA? (Y OR N) ",A$
710  IF A$="N" OR A$="n" THEN 760
720  FOR I=1 TO NPTS
730  PRINT I,FARRAY(I)
740  NEXT I
750  '
760  'PLOT DATA IF DESIRED
770  PRINT ""
780  PRINT "THE IBM CAN PLOT ONLY 640 PTS. HORIZONTALLY THEREFORE"
790  PRINT "  ONLY THE FIRST 640 PTS. OF THE DATA RECORD WILL BE PLOTTED."
800  PRINT ""
810  INPUT "DO YOU WANT A PLOT OF THE DATA ? (Y OR N)  ";A$
820  IF A$="N" OR A$="n" THEN 970
830  SCREEN 2
840  KEY OFF:CLS:LOCATE 1,1:PRINT "                          ",FLNA
850  YMIN=CMIN*FSF:YMAX=CMAX*FSF
860  LOCATE 2,1:PRINT "                                      NPTS=";N
870  LOCATE 3,1:PRINT "                                      YMIN=";:
880  LOCATE 4,1:PRINT "                                      YMAX=";:
890  FOR I=2 TO NPTS
900  IF I>640 THEN 930
910  LINE (I-1,C(I-1))-(I,C(I))
920  NEXT
930  BEEP
940  CLOSE #1
950  LOCATE 23,1:PRINT "HIT ANY KEY TO EXIT-PROGRAM"
960  A$=INKEY$:IF A$="" THEN 960 ELSE 970
970  END
```

Format of Binary Data Stored on Disk or Transferred over GPIB or RS-232 Interface.

Data files consist of a 36 byte fixed header followed by a variable length number of data bytes. The list below gives a description of that information as a function of its position in the header. It identifies the type of data and gives a description of it.

Starting Position (Byte #)	Length (Bytes)	Type of Data	Description
0	2	byte	STX, SOH
2	9	ASCII	Name of record
11	1	byte	Data type
12	2	intg	Length in elements
14	1	byte	X axis units
15	1	byte	Y axis units
16	4	real	X per point
20	4	real	X offset
24	4	real	Y full scale range
28	4	real	Y offset
32	4	dp intg	Length of data area in bytes
36	variable		Actual data

The data types are as follows:

ASCII - ASCII exchange code data.
intg - signed 16 bit integer.
real - IEEE floating point.
dp intg - signed 32 bit integer.
byte - 8 bit integer.

Coding for UNITS in D6000 (Revision 4.0)

UNITS	CODE	UNITS	CODE	UNITS	CODE
S	1	dBW	37	Ft-lb	73
V	2	dBm	38	l	74
Hz	3	dBr	39	n	75
Pts	4	%	40		
V	5	Deg	41		
V 2	6	rad	42		
V 3	7	Cnts	43		
V 4	8	Events	44		
S	9	#	45		
S 2	10	W	46		
S 3	11	Wh	47		
S 4	12	A	48		
Trgs	13	Ah	49		
Clks	14	Ohm	50		
VS	15	H	51		
V S	16	F	52		
VS 2	17	degC	53		
VS 3	18	degF	54		
VS	19	degK	55		
V 25	20	G	56		
V 3S	21	J	57		
V 2S 2	22	Ft	58		
V/S	23	Ft/S	59		
V 2/S	24	Ft/S 2	60		
V 3/S	25	m	61		
V/ A	26	m/S	62		
V/S 2	27	m/S 2	63		
V/S 3	28	lb	64		
V 2/S 2	29	kg	65		
VHz	30	kgm/S 2	66		
V 2Hz	31	N	67		
V/Hz	32	Pa	68		
V/ Hz	33	N/m 2	69		
V 2/Hz	34	Psi	70		
dB	35	kg/cm 2	71		
dBV	36	FLTs	72		

Coding for UNITS in D6000 (Revision 4.0)

UNITS	CODE	UNITS	CODE	UNITS	CODE
S	1	dBW	37	Ft-lb	73
V	2	dBm	38	l	74
Hz	3	dBr	39	n	75
Pts	4	%	40		
^V	5	Deg	41		
V^2	6	rad	42		
V^3	7	Cnts	43		
V^4	8	Events	44		
S	9	#	45		
S^2	10	W	46		
S^3	11	Wh	47		
S^4	12	A	48		
Trgs	13	Ah	49		
Clks	14	Ohm	50		
VS	15	H	51		
V^S	16	F	52		
VS^2	17	degC	53		
VS^3	18	degF	54		
^VS	19	degK	55		
V^25	20	G	56		
V^3S	21	J	57		
V^2S^2	22	Ft	58		
V/S	23	Ft/S	59		
V^2/S	24	Ft/S^2	60		
V^3/S	25	m	61		
V/^A	26	m/S	62		
V/S^2	27	m/S^2	63		
V/S^3	28	lb	64		
V^2/S^2	29	kg	65		
VHz	30	kgm/S^2	66		
V^2Hz	31	N	67		
V/Hz	32	Pa	68		
V/^Hz	33	N/m^2	69		
V^2/Hz	34	Psi	70		
dB	35	kg/cm^2	71		
dBV	36	FLTs	72		

APPLICATION NOTE

Application Note #503

Automatic scaling of input waveforms on the Data 6000

ANALOGIC®
DATA PRECISION®
16 Electronics Avenue Danvers, MA 01923 U.S.A.

Introduction

In dealing with signals of widely varying amplitudes it is desireable to have a scaling feature for display purposes. For applications where the signal of interest is unknown at the start, this can be invaluable in helping to identify and characterize the device under test (DUT).

This application note presents a simple program that is used to autoscale the y-axis of the display so that the acquired signal is displayed with optimum range. The program will run with any plug-in module and was written for a Rev. 4.0 D6000 system.

Discussion

The object of the program is to measure the amplitudes of the waveform, acquired and stored in BUF.A1, and to adjust the Y-offset and Y-scale accordingly. The waveform will then appear as a near full scale range signal, centered in the display. The program listing follows:

```
10   A=MAX(BUF.A1)
20   B=PKPK(BUF.A1)
30   B=B/2
40   YOFF=A-B
50   C=B/5.12
60   YSCL=.95/C
```

In line 40, YOFF is the mnemonic for the Y-offset and is adjusted to center the waveform. In line 50, the number 5.12 is the full scale range (as indicated under the INP key) and will vary from one plug-in to another and one range to another. In line 60, YSCL is the mnemonic for Y-scale and is adjusted to present the waveform over 95% of the available display range.

If you now connect your DUT and ARM the D6000 you should see the signal display as indicated. Note that the program will run at the end of every acquisition (unless otherwise instructed) thereby changing the scales for every acquired waveform. Since only the display parameters are changed,

all calculations will be performed normally on BUF.A1 as though no scale factors have been applied. It is suggested that the Y key be pressed so you can see that Y-offset and Y-scale are actually being changed for each acquisition.

Mainframe Rev. 4.66
Contiguous memory 188416 } 660B
total 238208 }
Contig. 124928 } 611-1
total 153216 }

DATA 6100 USERS GUIDE

SECTION I
USER GROUP APPLICATION NOTES

This GUIDE is divided into alphabetically-designated sections, each one organized about one or more DATA 6100 performance functions. A separate section is used for the complete index, with references to the appropriate section, chapter, and page number.

The GUIDE Sections are:

Plug-ins 611-1 100kHz rev. 4.80 14bit
+ 660B 250MHz (but goes to 40ps!)
rev. 4.80

I

INTRODUCTION

The following application notes are taken from the 6100's precursor, the DATA 6000. Although the 6100 incorporates some major improvements over the 6000, the architecture is the same and the two machines are, to a large extent, interchangeable.

Since application notes are developed over a span of time and the 6100 is a new product, a number of notes exist for the 6000 but not yet for the 6100. However, 6000 notes can be applied to the 6100 and are equally useful for either machine. Included in this section are all DATA 6000 application notes as of this writing. More will be developed, as certainly will application notes for the 6100 specifically, utilizing its unique abilities.

Because of the wide variety of applications the 6000 and 6100 are designed and used for, application notes are often supplied by users. If you develop any application notes you would like to share with other users, please forward them to:

Applications
Data Precision
16 Electronics Ave.
Danvers, MA 01923