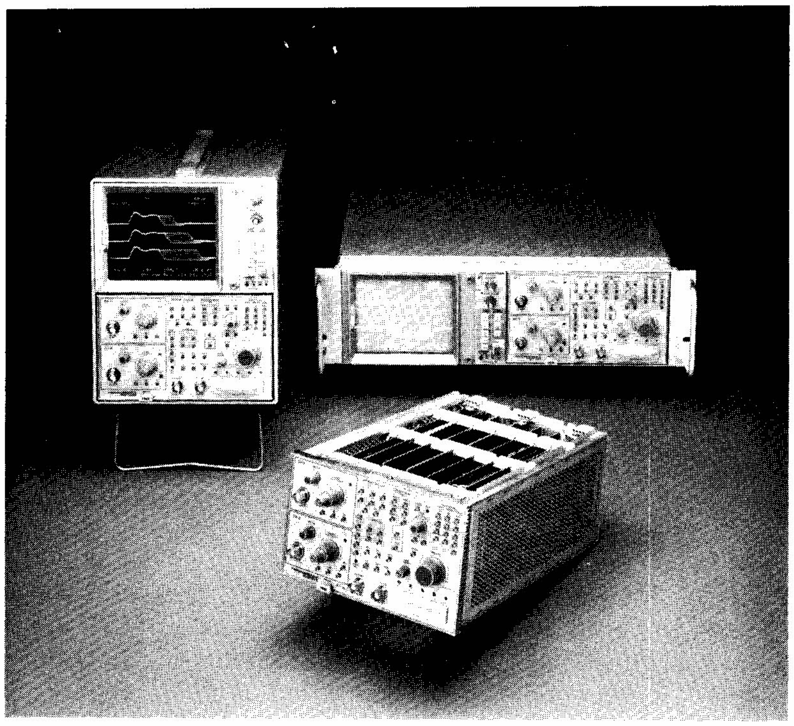
7D20 Instrument Interfacing Guide



1728-01

This Interfacing Guide is designed to help you get started using the 7D20 Programmable Digitizer on the GPIB as quickly and easily as possible. This IIG provides information on setting up the 7D20 for GPIB operation as well as some program examples that illustrate the communication with the 7D20. This guide is not intended to take the place of the 7D20 Operators Manual or the manuals for your controller.

CONNECTING TO THE SYSTEM

Attach the 7D20 to the GPIB using a standard GPIB cable. The 7D20 can be connected to the bus from either the front-panel GPIB connector or when installed in a R7603/ option 20 mainframe, from the rear-panel GPIB connector. Note, when using the 7D20 with the R7603 Option 20, be sure the jumper cable (Part number 175-7151-00) is installed in the 7D20.

For additional copies, order
070-1728-00
First Printing APR 1983

Tektronix®
COMMITTED TO EXCELLENCE

The GPIB system can be cabled in two general configurations: star or linear (Fig. 1). While the star configuration is the recommended configuration, the two configurations can be mixed as long as the total cable length does not exceed 20 meters and the instruments are distributed on the bus according to a few rules.

First, no more than 15 total devices (including the controller) can be included on a single bus. In addition, to maintain the bus electrical characteristics, one device load must be connected for every two meters of cable (generally each instrument represents one device load to the bus). The 7D20 represents one device load.

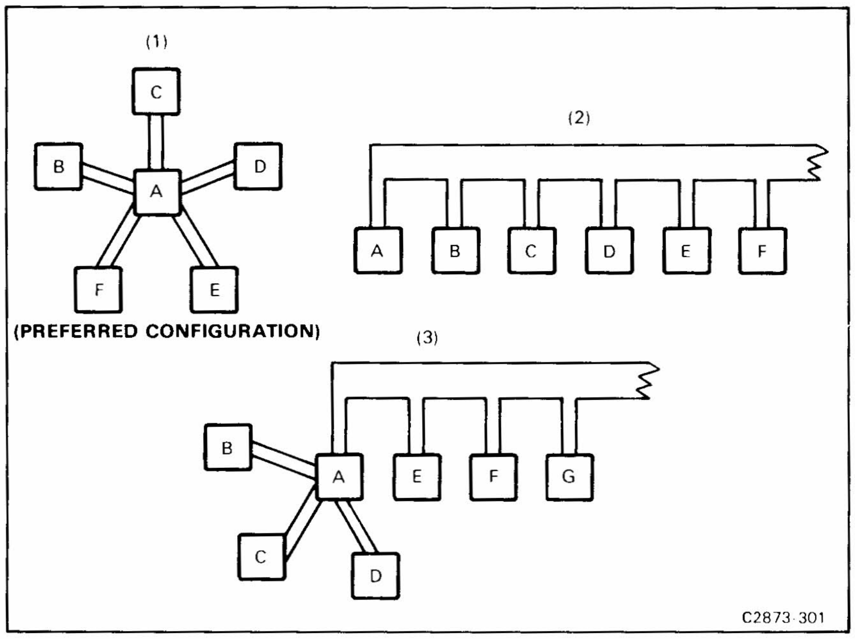
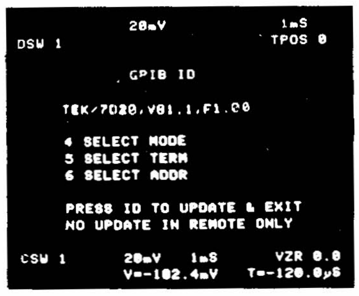


Fig. 1. *GPIB system configurations.*

Instrument Configuration (Selection and Verification)

The 7D20 provides for the selection of the mode, terminator, and bus address by means of the front-panel numeric keypad (MEMORY DISPLAY KEYS 1-6), in conjunction with the ID menu displayed on the CRT as shown in Fig. 2.



```
            20mV                 1mS
DSW 1                          TPOS 0

              GPIB ID

      TEK/7D20,V81.1,F1.00

      4 SELECT MODE
      5 SELECT TERM
      6 SELECT ADDR

      PRESS ID TO UPDATE & EXIT
      NO UPDATE IN REMOTE ONLY

CSW 1       20mV    1mS       VZR 0.0
            V=-102.4mV    T=-120.0µS
```

Fig. 2. *CRT display of the ID selection menu.*

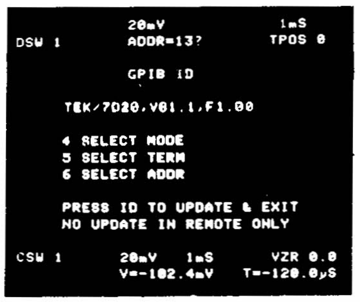
The selection process is initiated by pressing the front-panel ID key, which changes the numeric keypad from waveform selection to function selection, and produces the selection menu on the CRT. The selection process is terminated by pressing the front-panel ID key again. All selections made are implemented at this time unless the 7D20 is in the REMOTE ONLY state. If so, the prompt message 'UPDATE IGNORED' will appear on the screen in the prompt field. The 7D20 will enter the REMOTE ONLY state if the controller sends the REMOTE WITH LOCKOUT (RWLS) interface command and addresses the 7D20. In this state, all front-panel functions are disabled with the exception of the "USER" SRQ functions , Variable Volts/Div, and Horizontal Position controls. The Variable Volts/Div and Horizontal Position can be disabled by placing them into their 'cal' position and the USER SRQ functions can be disabled (masked) with GPIB commands.

The GPIB operating parameters are stored in non-volatile memory enabling the 7D20 to retain its GPIB configuration when the power is removed.

Address Selection

Each instrument connected to the bus must have a unique address. This address is used by the controller to direct the flow of data to and from each device. To select the 7D20's GPIB address, press front-panel key #6 (MEMORY DISPLAY). The prompt field will display the current GPIB address as 'ADDR= <addr>?'. The question mark and key #6 will both blink. Repeatedly pressing key #6 increments the address number, beginning with the current address. The address will increment thru each address until reaching 30, then reset to 0 and begin the count again.

Fig. 3 shows the display when the GPIB address is 13.



```
            20mV              1mS
DSW 1       ADDR=13?          TPOS 0

            GPIB ID

     TEK/7D20,V01.1,F1.00

     4 SELECT MODE
     5 SELECT TERM
     6 SELECT ADDR

     PRESS ID TO UPDATE & EXIT
     NO UPDATE IN REMOTE ONLY

CSW 1      20mV    1mS        VZR 0.0
           V=-102.4mV   T=-120.0μS
```

3857-303

Fig. 3. *CRT display of ADDRESS selection prompt.*

When you are choosing a bus address for the instrument, keep a few things in mind. First, the address must be unique—no other device on the same bus can have the same address. Second, remember that some controllers reserve an address for themselves. The Tektronix 4050-Series Graphic Computing Systems, for example, reserves address zero and the 4041 System Controller uses address 30 by default. The 4041's address can be changed under program controll. If your controller reserves an address, the instrument cannot be set to that address.

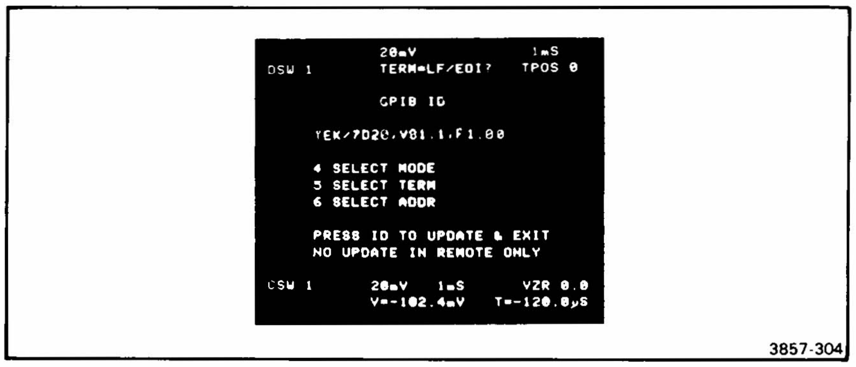
The 7D20 uses primary addressing only. Sending a secondary address will have no effect.

Terminator Selection

The terminator is the signal or character which is used to indicate the end of a message transfer. The two most common terminators are the EOI (End or Identify) signal line and the character LF (Line Feed). If EOI is selected, the 7D20 will assert the EOI line simultaneously with the last data byte when sending a message and will recognize the EOI line as the terminator when receiving a message. If LF is selected, a CR (Carriage Return) and LF (Line Feed) are sent following the last data byte. The EOI line is asserted simultaneously with the LF. When inputting a message, the 7D20 will terminate the message upon receiving either the LF character or the EOI line being asserted.

The 7D20 allows for the selection of either terminator by pressing key #5. The prompt field will display the current status of the GPIB terminator as 'TERM=<term>?'. Key #5 and the question mark will blink. Repeatedly pressing the key alternates the terminator between EOI and LF/EOI. Refer to your controller's manual for proper terminator selection.

Fig. 4 illustrates the display when the LF/EOI terminator is selected.



```
              20mV              1mS
DSW 1         TERM=LF/EOI?     TPOS 0

              GPIB ID

       TEK/7D20,V81.1,F1.00

       4 SELECT MODE
       5 SELECT TERM
       6 SELECT ADDR

       PRESS ID TO UPDATE & EXIT
       NO UPDATE IN REMOTE ONLY

CSW 1        20mV     1mS        VZR 0.0
             V=-102.4mV    T=-120.0µS
```

Fig. 4. *CRT display of TERMINATOR selection prompt.*

Mode Selection

The 7D20 mode determines the 7D20's communication capabilities. Each mode has its own characteristics and purpose. The four modes are: Off-Line, Talk-Only, Listen-Only, Talk/Listen.

OFF-LINE:

Off-Line is used to effectively remove the 7D20 from the bus, prohibiting sending or receiving any messages. The 7D20 still represents one device load.

TALK-ONLY:

Talk-Only is primarily used in GPIB systems where there is no bus controller. The 7D20 assumes that it has received its talk address and will send data when directed to from the front-panel function keys. Talk-Only is used to send the current cursor waveform (CSW) to a listening device with the utility functions 'SEND CSW ASCII' or 'SEND CSW BINARY'. The format and content of the waveform transfer is detailed in the Waveform Transfer section.

The 7D20 will not accept any messages while in the Talk-Only mode. The current status of the 7D20 can, however, be read with a serial poll sequence.

LISTEN-ONLY:

Listen-Only is also for use in systems which do not contain a system controller. In Listen-Only the 7D20 assumes its listen address has been received and it will listen to all messages sent over the bus. The Listen-Only mode is useful for loading front-panel settings or waveforms from a storage device without a controller. In Listen-Only mode, no status information or service requests are available to the operator.

Prior to setting the 7D20 to the Listen-Only state, make sure all SRQ's have been cleared. The 7D20 cannot respond to a serial poll sequence when in Listen-Only.

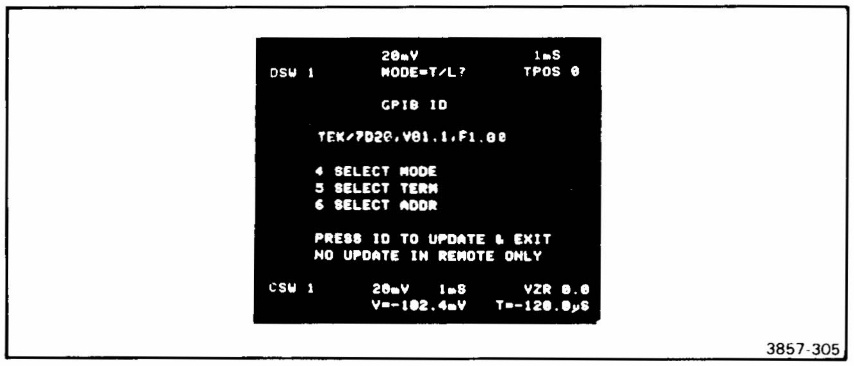
The Debug feature covered later in this IIG will function in the Listen-Only mode.

TALK/LISTEN:

Talk/Listen is the normal mode for the 7D20 and allows full bi-directional communication. In the Talk/Listen mode the 7D20 must be addressed by the system controller prior to any data transfers.

To select the GPIB mode, press key #4. The prompt field will display the current status of the GPIB mode as 'MODE= <mode>?'. Key #4 and the question mark will blink. Repeatedly pressing the key will sequence the mode selection through Off-Line,Talk-Only,Listen-Only, and Talk/Listen.

Fig. 5 illustrates the display when the Talk/Listen mode is selected.



```
            20mV                 1mS
DSW 1       MODE=T/L?            TPOS 0

            GPIB ID

      TEK/7D20,V81.1,F1.00

      4 SELECT MODE
      5 SELECT TERM
      6 SELECT ADDR

      PRESS ID TO UPDATE & EXIT
      NO UPDATE IN REMOTE ONLY

CSW 1       20mV    1mS          VZR 0.0
            V=-102.4mV    T=-120.0μS
```

3857-305

Fig. 5. *CRT display of MODE selection prompt.*

PROGRAMMING THE 7D20

POWER UP SELF-TEST AND SRQ

Immediately after power is applied to the 7D20, an extensive self-test is initiated. Following the completion of the self-test, the 7D20 will assert the SRQ signal line on the GPIB. Note: If the 7D20 is set to OFF-LINE or LISTEN-ONLY no SRQ will be generated. The instrument's status byte is set to reflect either a normal power-up (65) or internal error (99), indicating the self-test detected a problem.

If the controller's SRQ interrupt is disabled, the power-up SRQ can be ignored. With the Tektronix 4051 or 4052 without the R14 ROM pack, the message "NO SRQ ON UNIT" will appear if an SRQ is present and there is no SRQ handler routine.

To clear the SRQ from the 7D20 following a power-up sequence, two serial polls must be executed. The first poll will clear the "Power Up" SRQ and the second will clear the "Operation Complete" SRQ which was generated at the completion of the self-test. Fig. 6 shows a simple SRQ handler using the Tektronix 4041 controller. Further details on instrument status is covered in the SERVICE REQUEST section of this IIG.

```
Tektronix 4041 BASIC

   100  ON SRQ THEN GOSUB SRQHDLR
   110  ENABLE SRQ
   120  .
        .
        .
   1000 SRQHDLR: POLL STB,DEV
   1010 PRINT "STATUS ";STB;" REPORTED FROM INSTRUMENT ";DEV
   1020 RESUME

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

                    PROGRAM EXPLANATION:

    100  Sets up location of the routine to handle the SRQ
    110  Enables the controllers to respond to SRQ's
120-999  Normal program
   1000  Start of interrupt handler:
         Executes a serial poll of all addresses until
         a device responds with a status byte. STB is set
         to the value returned by the instrument and DEV
         is set to the primary address of the responding
         device.
   1010  Prints the message in quotation marks along
         with the values of STB and DEV on the console
         device of the 4041.
   1020  Return program control to the line executing
         at the time of the SRQ interrupt.
```

1728-02

Fig. 6. *Example of a basic SRQ handler using the 4041 Controller.*

Power Up Default Settings

When the 7D20 powers up, some of the front-panel control settings are not restored and are set to a default condition. Table 1 shows the default settings at power up. Controls not listed in this table retain their last settings.

Sending Commands to the 7D20

The Command set is the basic vocabulary of the 7D20 and directs an action to be taken when a command is received. A command consists of a group or string of ASCII characters . Most controllers will send an ASCII string by enclosing the desired characters with quotes (""). Fig. 7 shows how a string is sent to a GPIB device from a variety of controllers.

CONTROLLER	OUTPUT COMMAND
TEK 4050 SERIES BASIC	PRINT @10:"string"
TEK 4041 BASIC	PRINT #10:"string"
TEK SPS BASIC	PUT "string" INTO #10
FLUKE 1720A BASIC	PRINT @10%:"string"
HP-85 BASIC	OUTPUT 710;"string"
HP 9826A BASIC	OUTPUT 710;"string"
HP 9825/26A HPL	wrt 710,"string"

1728-03

Fig. 7. *Sending an ASCII string with several different controllers.*

For example, to send the command INIT, the statement would look like:

4041 BASIC PRINT #10:"INIT"

Note: the device address for the above examples is "10".

Table 1
POWER-UP DEFAULT FRONT PANEL SETTINGS

VERTICAL	
VARIABLE GAIN:	ACTIVE[a]
POSITION:	ACTIVE[a]
TRIGGERING	
TRIGGER LEVEL:	ACTIVE[a]
HORIZONTAL	
HORIZONTAL POSITION:	ACTIVE[a]
OTHER	
RQS,RQS#:	OFF
MENU:	OFF
ID:	OFF
HOLD:	OFF
f:	OFF
DISPLAY AND MEASUREMENT FUNCTIONS	
DISPLAY 3-6:	OFF
COPY:	OFF
CSW:	If CSW was 3-6 it is reset to CH1 if AQR MODE is CH1, BOTH, or ADD; to CH2 if AQR MODE is CH2.
VXPD,VCMP:	OFF
VPUP,VPDN:	OFF (Except separation for HMAG or VS)
AVE, AVEN:	OFF
ENV,ENVN:	OFF

[a]Set to current knob position.

Commands are structured in a Header/Label/Value format ensuring maximum flexibility and ease of use. A space is used to separate the Header from the Label, and a colon to seperate the Label from the Value. If the command has only a Header and Value, a space is used to separate them. Multiple functions within a group can be programmed in a single command by separating each Label/Value with a comma. For example, to set the trigger source to 'MODE' and the trigger coupling to 'AC' a single command string can be sent.

"TRIGGER SOURCE:MODE,COUPLING:AC"

Multiple command groups can also be programmed within a single message by separating each Header/Label/Value with a semi-colon. To set the trigger source to 'MODE' and turn on waveform memory #3, the following string can be used.

"TRIGGER SOURCE:MODE;DISPLAY 3:ON"

The commands can be spelled out completely for development or abbreviated for increased speed. For example, the short form of the above command would be:

"TR SO:MO;DI 3:ON"

Refer to the Command List section of this IIG, Operators Manual or Reference Guide for complete command listings.

Query Commands

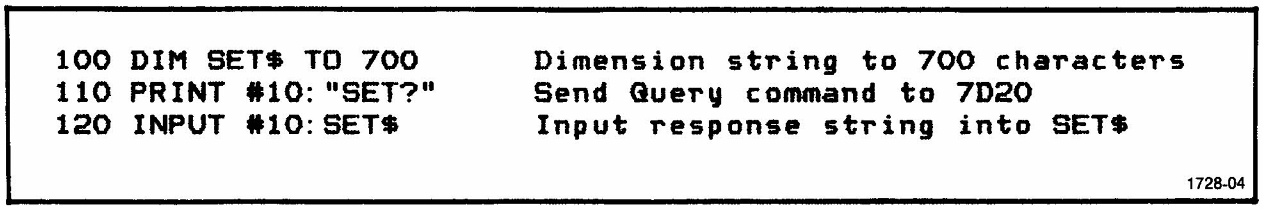
Queries are used to solicit information from the 7D20. Queries consist of either a header followed by a '?' or a header followed by a '?', and one or more labels. This format is similar to the instrument commands. A header followed only by a '?' queries the entire group. For example, the query TRIGGER? will respond with the current setting of all the trigger functions. A header followed by a '?' and a label addresses only a single function within the group. For example, the query TRIGGER? MODE will respond with only the current setting of the trigger function MODE. Query responses are formatted using the same syntax as the corresponding command. In this manner, the response to a query can be sent back to the 7D20 as a command without any processing.

Responses to a query are sent as a series of ASCII characters. Most controllers have single statements to input strings from the GPIB. Check with the controller manual for details on dimensioning strings. To input a query response, first dimension the desired string to a size large enough to hold all the expected characters. Then input the query response. Fig. 8 shows how a query response is input from several different controllers.

CONTROLLER	INPUT COMMAND
TEK 4050 SERIES BASIC	INPUT @10:S$
TEK 4041 BASIC	INPUT #10:S$
TEK SPS BASIC	GET S$ FROM #10
FLUKE 1720A BASIC	INPUT @10%:S$
HP-85 BASIC	ENTER 710 ;S$
HP 9826A BASIC	INPUT 710;S$
HP 9825/26A HPL	red 710,S$

Fig. 8. *Inputting a query response using a variety of controllers.*

For example, Fig 9 shows how to acquire the complete current 7D20 settings using a Tektronix 4041 controller.



```
100 DIM SET$ TO 700        Dimension string to 700 characters
110 PRINT #10: "SET?"      Send Query command to 7D20
120 INPUT #10: SET$        Input response string into SET$
```

Fig. 9. *Getting the 7D20 settings with a 4041 controller.*

Line 110 sends the command 'SET?' to the 7D20 which in this case is set to address 10. Line 120 inputs the response into the predefined string named SET$. The 7D20 will send the complete command names as the default condition. Due to the extensive number of programmable settings, the SET? response string will be approximately 700 ASCII bytes. The LONGFORM OFF command will direct the 7D20 to send only the abbreviated portion of each command when responding to a query. When the 7D20 is returning the short form the entire setting string will be returned in less than 400 ASCII bytes.

Waveform Transfers

The 7D20 has six waveform locations and two formats for waveform transfers. The 7D20 uses the 'DATA MEMORY' command to establish the source and destination for subsequent waveform transfers. The 'DATA ENCDG' (Data Encoding) command is used to determine if the transfer is to be in ASCII or BINARY format.

For example, to set the source and destination Memory to location #4, the following command would be used.

Using the Tektronix 4041 controller

```
100 PRINT #10: "DATA MEMORY: 4"
```

To find out which memory is currently selected, simply send the 'DATA? MEMORY' query and input the response string.

```
100 PRINT #10: "DATA? MEMORY"
110 INPUT #10: RESP$
```

For this example, the response would be:

```
RESP$= DATA MEMORY: 4
```

The DATA ENCDG is set to either ASCII or BINARY in the same manner as the DATA MEMORY command.

For ASCII:

```
100 PRINT #10: "DATA ENCDG: ASCII"
```

For BINARY:

```
100 PRINT #10: "DATA ENCDG: BINARY"
```

To determine the current encoding send the 'DATA? ENCDG' query and input the response string.

```
100 PRINT #10: "DATA? ENCDG"
110 INPUT #10: RESP$
```

The response would be:

```
RESP$= DATA ENCDG: ASCII
```

or

```
RESP$= DATA ENCDG: BINARY
```

The number of waveform points transfered will be either 1024 or 820 points depending on the Time/Division setting. In the Time/Div range of 200uS to 2uS the number of points transfered is 820. For all other settings the number is 1024. If desired, the 'DATA INTERPOLATE:ON' command can be used, directing the 7D20 to interpolate the 820 points to 1024 prior for transfer. The number of points to be transfered is contained in the waveform preamble.

To verify the waveform transfer setup, a single query will return all the necessary information.

```
100 PRINT #10: "DATA?"
110 INPUT #10: SETUP$
```

With the above setups the response would be:

```
DATA ENCDG: ASCII, INTERPOLATE: ON, MEMORY: 4
```

Waveform Formats

ASCII ENCODING:

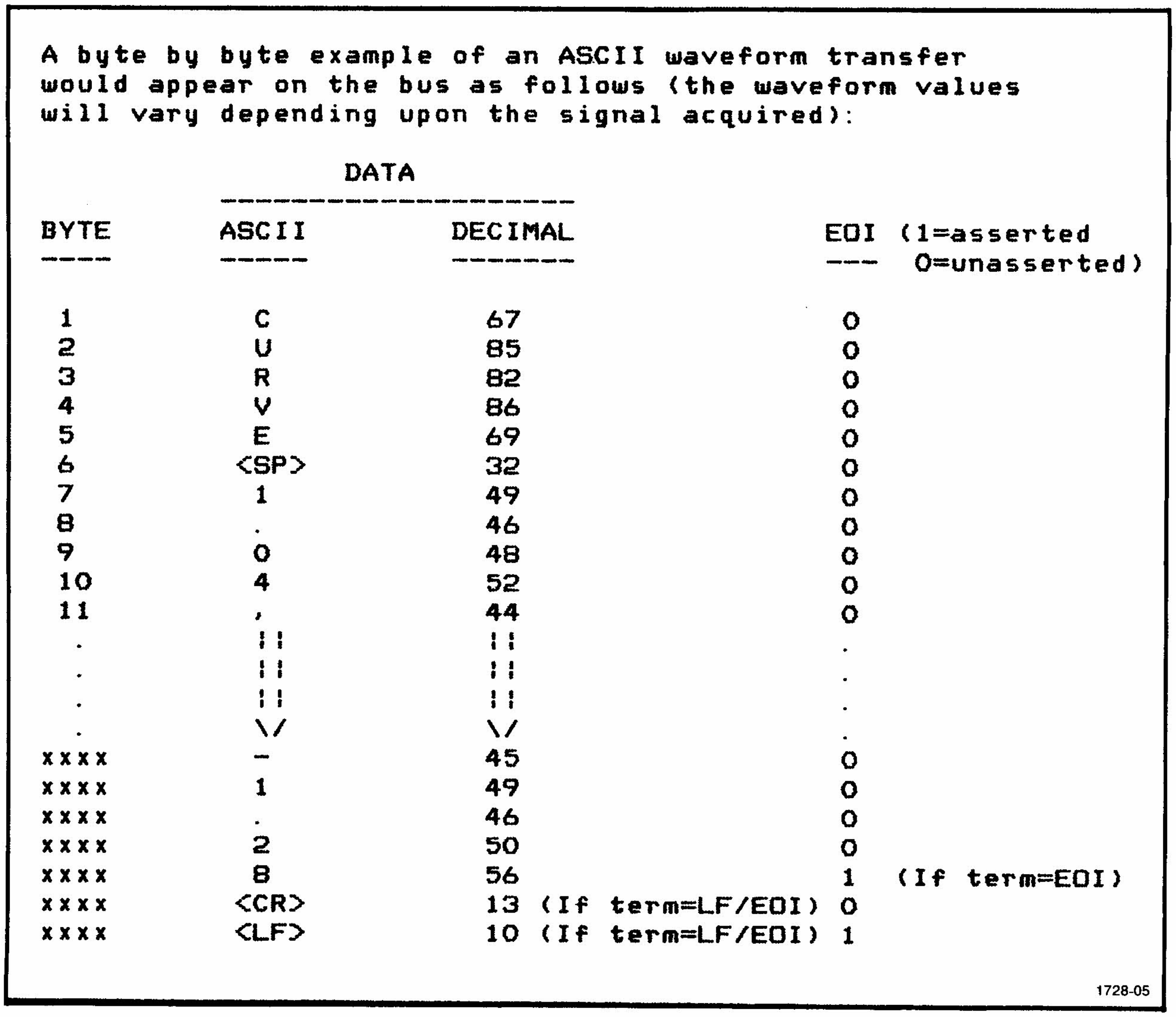
In ASCII encoding, each waveform point is expressed in CRT graticule divisions relative to center screen with a range of ±5 divisions. For example, if the display showed a straight line positioned one division above center screen, the data read by the controller would be 1.0 for each waveform point.

In ASCII format the curve data transfer is represented as follows:

```
CURVE<space>data, data, data, . . . , data<terminator>
```

The actual number of characters transfered will vary depending on the data values. The resolution of the 7D20 is 0.04 divisions with trailing zeros omitted. If the data value was 1.00, the 7D20 would send a value of 1.0 leaving off the final zero. A negative number will have a minus sign (−) preceding the digits. The value zero is sent as 0.0.

Fig. 10 shows an example of the data sent over the bus during an ASCII waveform transfer.



```
A byte by byte example of an ASCII waveform transfer
would appear on the bus as follows (the waveform values
will vary depending upon the signal acquired):

                     DATA
            ----------------------
BYTE        ASCII        DECIMAL                EOI (1=asserted
----        -----        -------                ---  0=unasserted)

 1            C            67                    0
 2            U            85                    0
 3            R            82                    0
 4            V            86                    0
 5            E            69                    0
 6          <SP>           32                    0
 7            1            49                    0
 8            .            46                    0
 9            0            48                    0
 10           4            52                    0
 11           ,            44                    0
  .          ! !           ! !                   .
  .          ! !           ! !                   .
  .          ! !           ! !                   .
  .          \/            \/                    .
xxxx          -            45                    0
xxxx          1            49                    0
xxxx          .            46                    0
xxxx          2            50                    0
xxxx          8            56                    1  (If term=EOI)
xxxx        <CR>           13 (If term=LF/EOI)   0
xxxx        <LF>           10 (If term=LF/EOI)   1

                                                              1728-05
```

Fig. 10. *Typical bus traffic during an ASCII waveform transfer.*

BINARY ENCODING:

In Binary Encoding, each waveform point is represented by one byte of data since the vertical resolution of the 7D20 is 8 bits. The range of the data is 0 to 255, where 0 represents the -5.12 division level, 127 is center screen, and 255 represents the 5.08 division level.

In BINARY format the curve data transfer is represented as follows:

CURVE<space>%<Binary Count><Data...>
<Checksum><Terminator>

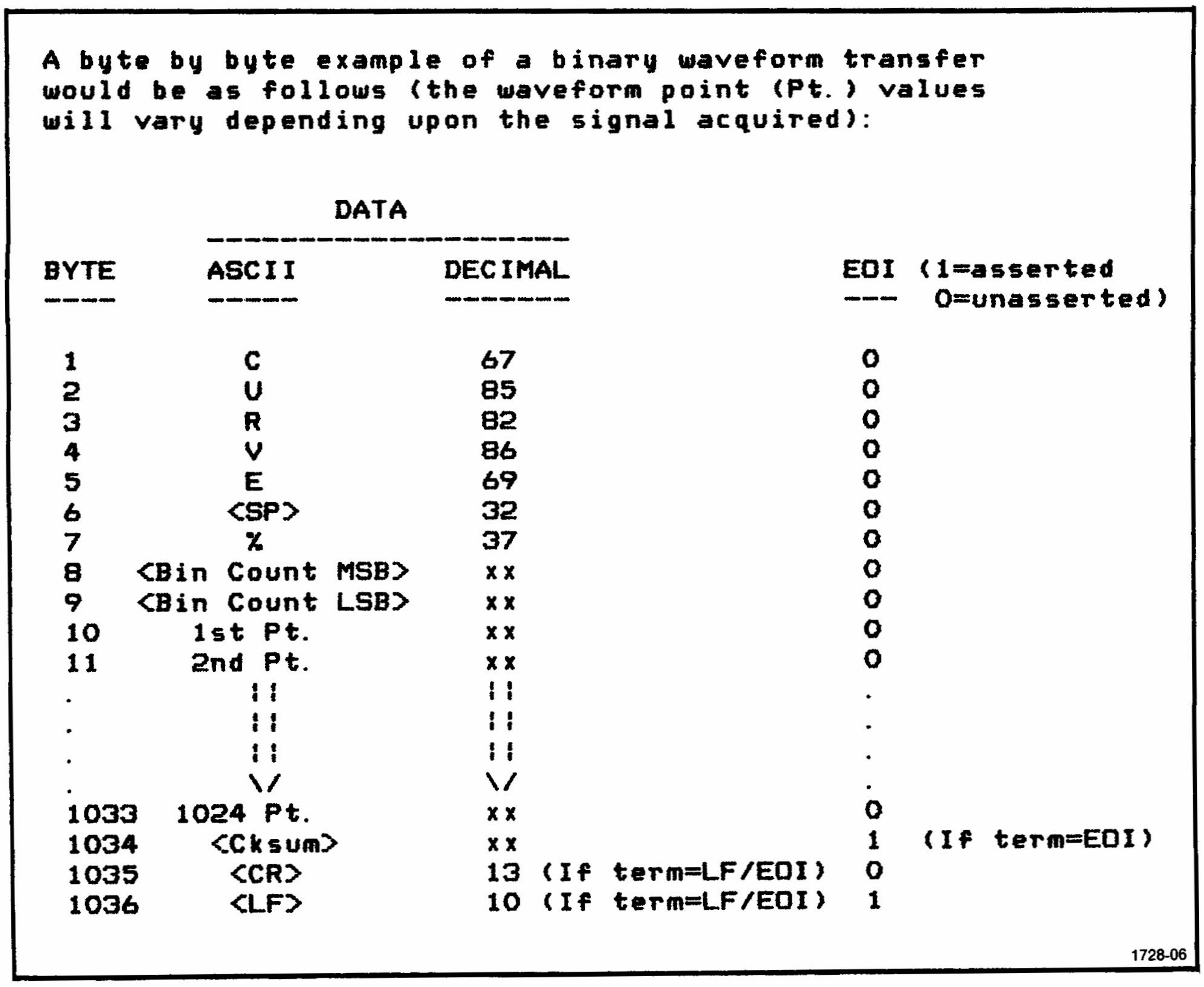
% is used as a header character to synchronize the start of the transfer.

<Binary Count MSB> is the most significant byte of the two byte Binary Count used to indicate the number of data bytes to follow plus 1 for the checksum byte.

<Binary Count LSB> is the least significant byte of the Binary Count.

<checksum> is the two's complement of the modulo 256 sum of the preceding data bytes and the binary count. The checksum is a good way for the controller to verify that all data values have been received correctly.

Fig. 11 shows an example of the data sent over the bus during a BINARY waveform transfer. The waveform point (Pt.) values will vary depending upon the signal acquired.



```
A byte by byte example of a binary waveform transfer
would be as follows (the waveform point (Pt.) values
will vary depending upon the signal acquired):


                  DATA
         ----------------------
BYTE      ASCII         DECIMAL                  EOI (1=asserted
----      -----         -------                  ---  0=unasserted)

 1          C             67                      0
 2          U             85                      0
 3          R             82                      0
 4          V             86                      0
 5          E             69                      0
 6        <SP>            32                      0
 7          %             37                      0
 8    <Bin Count MSB>     xx                      0
 9    <Bin Count LSB>     xx                      0
 10       1st Pt.         xx                      0
 11       2nd Pt.         xx                      0
 .          ::            ::                      .
 .          ::            ::                      .
 .          ::            ::                      .
 .          \/            \/                      .
 1033   1024 Pt.          xx                      0
 1034     <Cksum>         xx                      1   (If term=EOI)
 1035      <CR>           13 (If term=LF/EOI)     0
 1036      <LF>           10 (If term=LF/EOI)     1
```

1728-06

Fig. 11. *Typical bus traffic during a BINARY waveform transfer.*

Waveform Preamble

The waveform preamble provides the waveform attributes,such as number of points per waveform, the trigger, scale factors, offset, horizontal increment, scaling units, and data encoding. The preamble information is sent as an ASCII string of less than 200 characters with the actual length depending on the characteristics of the waveform.

A typical response to a preamble query (WFMPRE?) would be:

ASCII FORMAT:

```
WFMPRE WFID:W 1,ENCDG:ASCII,NR.PT:1024,
PT.FMT:Y, XINCR:1.0E-5,PT.OFF:1.2E+1,
XZERO:0, XUNIT:S,YMULT:1.0,YZERO:0,YUNIT:V
```

BINARY FORMAT:

```
WFMPRE WFID:W 1,ENCDG:BINARY,NR.PT:1024,
PT.FMT:Y, XINCR:1.0E-5,PT.OFF:1.2E+1,
XZERO:0, XUNIT:S, YMULT:1.0, YZERO:0,
YUNIT:V, BYT/NR:1, BN.FMT:LF,BIT/NR:8,
CRVCHK:CHKSMO
```

Transferring Waveforms

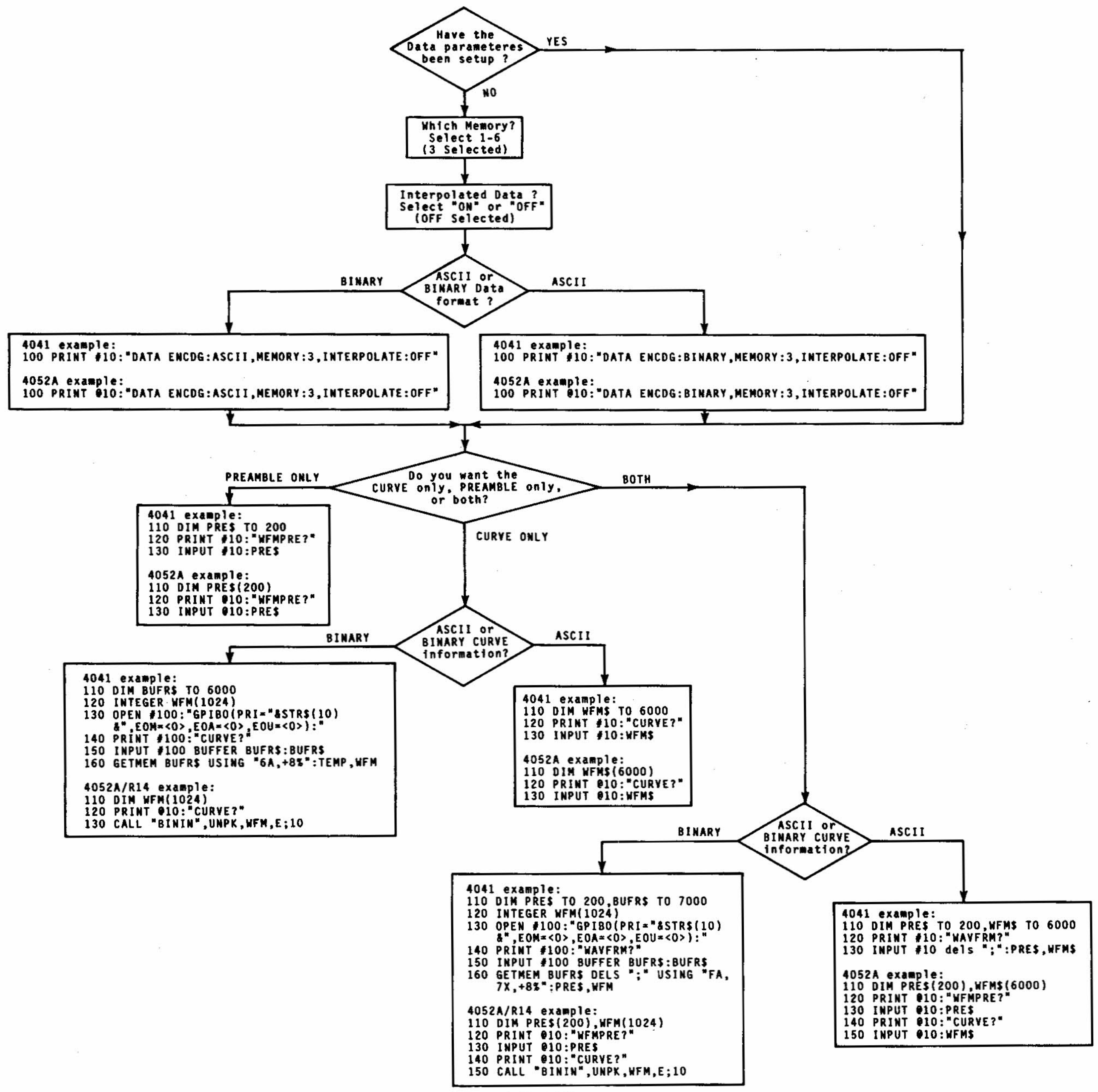
The 7D20 can respond with either the Preamble only, Curve only, or both Preamble and Curve together.

Command:	**Response:**
"CURVE?"	Curve Data Only
"WFMPRE?"	Preamble Only
"WAVFRM?"	Preamble and Curve data

When responding to the "WAVFRM?" query the preamble is separated from the curve data with a ";".

Preamble;Curve Data

The following diagram illustrates the decisions and procedures required to transfer a waveform between the 7D20 and controller via the GPIB.


Have the Data parameteres been setup ?
YES
NO
Which Memory? Select 1-6 (3 Selected)
Interpolated Data ? Select "ON" or "OFF" (OFF Selected)
BINARY
ASCII or BINARY Data format ?
ASCII
PREAMBLE ONLY
Do you want the CURVE only, PREAMBLE only, or both?
BOTH
CURVE ONLY
BINARY
ASCII or BINARY CURVE information?
ASCII
BINARY
ASCII or BINARY CURVE information?
ASCII

```
4041 example:
100 PRINT #10:"DATA ENCDG:ASCII,MEMORY:3,INTERPOLATE:OFF"

4052A example:
100 PRINT @10:"DATA ENCDG:ASCII,MEMORY:3,INTERPOLATE:OFF"
```

```
4041 example:
100 PRINT #10:"DATA ENCDG:BINARY,MEMORY:3,INTERPOLATE:OFF"

4052A example:
100 PRINT @10:"DATA ENCDG:BINARY,MEMORY:3,INTERPOLATE:OFF"
```

```
4041 example:
110 DIM PRE$ TO 200
120 PRINT #10:"WFMPRE?"
130 INPUT #10:PRE$

4052A example:
110 DIM PRE$(200)
120 PRINT @10:"WFMPRE?"
130 INPUT @10:PRE$
```

```
4041 example:
110 DIM BUFR$ TO 6000
120 INTEGER WFM(1024)
130 OPEN #100:"GPIB0(PRI="&STR$(10)
    &",EOM=<0>,EOA=<0>,EOU=<0>):"
140 PRINT #100:"CURVE?"
150 INPUT #100 BUFFER BUFR$:BUFR$
160 GETMEM BUFR$ USING "6A,+8%":TEMP,WFM

4052A/R14 example:
110 DIM WFM(1024)
120 PRINT @10:"CURVE?"
130 CALL "BININ",UNPK,WFM,E;10
```

```
4041 example:
110 DIM WFM$ TO 6000
120 PRINT #10:"CURVE?"
130 INPUT #10:WFM$

4052A example:
110 DIM WFM$(6000)
120 PRINT @10:"CURVE?"
130 INPUT @10:WFM$
```

```
4041 example:
110 DIM PRE$ TO 200,BUFR$ TO 7000
120 INTEGER WFM(1024)
130 OPEN #100:"GPIB0(PRI="&STR$(10)
    &",EOM=<0>,EOA=<0>,EOU=<0>):"
140 PRINT #100:"WAVFRM?"
150 INPUT #100 BUFFER BUFR$:BUFR$
160 GETMEM BUFR$ DELS ";" USING "FA,
    7X,+8%":PRE$,WFM

4052A/R14 example:
110 DIM PRE$(200),WFM(1024)
120 PRINT @10:"WFMPRE?"
130 INPUT @10:PRE$
140 PRINT @10:"CURVE?"
150 CALL "BININ",UNPK,WFM,E;10
```

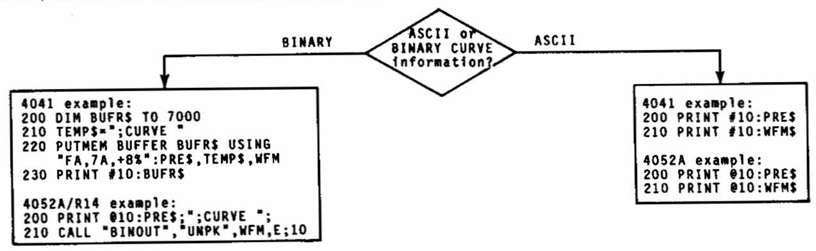
```
4041 example:
110 DIM PRE$ TO 200,WFM$ TO 6000
120 PRINT #10:"WAVFRM?"
130 INPUT #10 dels ";":PRE$,WFM$

4052A example:
110 DIM PRE$(200),WFM$(6000)
120 PRINT @10:"WFMPRE?"
130 INPUT @10:PRE$
140 PRINT @10:"CURVE?"
150 INPUT @10:WFM$
```

In the examples above using the ASCII format, the string variable WFM$ contains all of the ASCII characters sent as illustrated in Fig. 10 with the exception of the carriage return <CR> and line feed <LF>.

For the BINARY examples, the array WFM contains the positive integers ranging from 0 to 255 representing only the waveform data points as shown in Fig. 11.

Transferring Waveforms Back to the 7D20

Requires previous use of extraction routines to define variables.



```
4041 example:
200 DIM BUFR$ TO 7000
210 TEMP$=";CURVE "
220 PUTMEM BUFFER BUFR$ USING
    "FA,7A,+8%":PRE$,TEMP$,WFM
230 PRINT #10:BUFR$

4052A/R14 example:
200 PRINT @10:PRE$;";CURVE ";
210 CALL "BINOUT","UNPK",WFM,E;10
```

```
4041 example:
200 PRINT #10:PRE$
210 PRINT #10:WFM$

4052A example:
200 PRINT @10:PRE$
210 PRINT @10:WFM$
```

PROGRAMMING AIDS

The 7D20 has two commands which can be very useful in the learning of 7D20 operation and debugging 7D20 programs.

Help Command

The HELP command allows the user to obtain a list of all valid command headers used by the 7D20 as shown in Fig. 12.

By simply adding a "?" to any one of the command headers, the user can get the detailed format for any of the commands. In this manner the user can learn to program the 7D20 without any other documentation. If the command LONGFORM OFF is sent, all subsequent query responses will show the minimum characters required for each command.

```
HELP query (HELP?) will return a list of all valid
command headers:

CH1, CH2, TRIGGER, HORIZONTAL, DISPLAY, COPY, CSW, AQR,
CURSOR, STORE, RECALL, DT, INIT, TEST, CAL, RQS, CER, EXR,
INR, EXW, OPC, USER, PID, SRQ, WFMPRE, CURVE, DATA, WAVFRM,
TEXT, DEBUG, RECORDING, LONGFORM
```
1728-07

Fig. 12. *List of all valid command headers sent by the 7D20 in response to an HELP? query.*

For example, Fig. 13 shows the a sample program to input the DISPLAY command settings. The command header DISPLAY was used with only a '?' being added. The 7D20's response is with LONGFORM ON.

```
4041 controller:

  100 DIM DISP$ to 84
  110 PRINT #10: "DISPLAY?"
  120 INPUT #10: DISP$

DISP$ now contains all of the DISPLAY information, and if
we print the contents of DISP$ we would see:

DISPLAY CSW: 2, 1: ON, 2: OFF, 3: OFF, 4: OFF, 5: ON, 6: OFF, VECTOR: ON,
REFERENCE: OFF, RDOUT: ON
```
1728-08

Fig. 13. *Inputting the DISPLAY settings from the 7D20.*

As described previously in the command section, we could use the information in DISP$ to control any function of the DISPLAY group. To turn on memory displays 2 and 6, the command would be:

```
100 PRINT #10: "DISPLAY 2: ON, 6: ON"
```

Remember multiple command labels need only be separated with a ",".

Debug Command

A technique useful in debugging a controller program is to view the data being sent to each device. The 7D20 cannot display the data going to other devices, but all commands being sent to the 7D20 can be viewed. Sending the command DEBUG ON will cause all commands received by the 7D20 to be displayed on the CRT. Any errors detected by the instrument are displayed following the point of error. The error codes displayed on the CRT are the same as sent over the GPIB in response to an EVENT? query.

Long data strings will produce a scrolling display on the CRT. The scroll rate is set slow enough to allow the viewer time to view the data. If an error is detected within a long string, the scrolling will pause for approximately 2 seconds to allow the viewer to identify the error code. Following the pause, scrolling will resume with no loss of data.

Upon receipt of the message terminator the 7D20 will display the ! character; indicating either the EOI line or the LF character was received depending on the terminator selection . This allows the user to determine if the complete message is being received by the 7D20. Any characters received which are not recognized by the 7D20 are displayed as @ characters.

To disable the DEBUG feature, simply send the command DEBUG OFF.

7D20 I/O Characteristics

The 7D20's GPIB architecture is designed to allow the user to program the instrument without any concern for the hardware. Advanced programmers should note the following hardware considerations:

1. The input buffer is 128 characters deep. When the input buffer space is full, the bus transfer is suspended while the 7D20 processes the data. As buffer space becomes available, the transfer is continued. This process is transparent to the controller unless the controller has a handshake timeout feature that is set for a short time interval.

2. The output buffer is also 128 characters deep. When 128 characters are loaded into this buffer it is considered full.

3. If a query is sent to the 7D20 that fills the output buffer before the instrument has been addressed to talk and the input buffer is full, the output buffer is cleared and an execution error status byte is reported. An EVENT? query at this time would evoke an event code of 203 indicating "I/O buffers full, output dumped". It is possible to send a sequence such as: Query, command, command... and fill the input buffer before addressing to talk the 7D20. This will create an I/O deadlock. To avoid loss of data, the controller should talk address the instrument with each query sent; otherwise, an I/O deadlock may occur.

INTERFACE MESSAGES

Interface messages are commands defined by the IEEE 488 standard and are used to control the interfacing functions of the instrument rather than the 7D20 functions. The IEEE 488 standard specifies these messages so that they are the same for all devices. Each instrument is allowed to implement any one or all of the Interface messages. Refer to the instrument's manual for details. Interface messages are distinguished from device dependent messages by the controller asserting the ATN (Attention) line during the interface message.

For details on sending interface messages, consult the controller's manual.

The 7D20 does not respond to the following interface messages:

*PPC—Parallel Poll Configure
*PPU—Parallel Poll Unconfigure
*TCT—Take Control

The 7D20 responds to the following interface messages:

Serial Poll Enable/Disable (SPE/SPD)

The general operating status is read from the 7D20 by way of a Serial Poll. When the device receives the Serial Poll Enable (SPE) command, it prepares to send the status byte. When it receives its talk address, the 7D20 will send one byte to the controller. To exit the serial poll state, the controller must send a Serial Poll Disable (SPD) command. For more details, refer to Service Request section.

Group Execute Trigger (GET)

The GET is an addressed command which causes the device to execute a previously defined function. The device which is to receive the GET must be listen addressed prior to receiving the command. The 7D20 uses the GET in combination with the HOLD, HOLDNEXT, AVE, AVEN, ENV, ENVN, and NORMAL instrument commands. When any of these commands are sent to the 7D20 in a 'DT' (Device Trigger) command, the 7D20 will wait until a GET is received before implementing the command. Only one deferred command can be triggered from each GET command.

Local Lockout (LLO)

The Local Lockout command is used to disable front-panel operation. When the 7D20 receives the LLO command, the instrument enters the REMOTE ONLY state. The front-panel REMOTE ONLY light will light indicating front-panel operaton is suspended. Unless previously disabled by command, the user RQS functions are still operational.

To exit the LLO state, the controller must either set the Remote Enable (REN) line false or send the Go To Local (GTL) command. If the GTL command is sent, the 7D20 will exit the REMOTE ONLY state but, will reenter the REMOTE ONLY state if readdressed. Some controllers only assert the REN line while a program is executing or for the duration of an immediate mode statement. Anytime the REN line is false all instruments MUST go to their local state.

Go To Local (GTL)

The Go To Local command is an addressed command used to return the device to its local state. The front-panel keys are functional by the user only when the instrument is not in the REMOTE ONLY state.

Device Clear (DCL/SDC)

The controller issues the Device Clear message (DCL) or (SDC) to initialize internal GPIB functions of devices on the bus. DCL is a universal command that applies to all devices. Its effect on each instrument, however, is decided by the designer. When the 7D20 receives the DCL command, the following occurs:

1. All bus communications in progress are terminated without untalking or unlistening.

2. All pending SRQ flags except POWER ON are cleared.

3. All pending event query responses except POWER ON are cleared.

4. Input and output buffers are cleared.

If the 7D20 is listen addressed (MLA), it will respond to a Selected Device Clear (SDC) in the same manner as described for Device Clear.

Interface Clear (IFC)

The Interface Clear (IFC) message interrupts any data input or output, and places the 7D20 in the unaddressed state. If the 7D20 is talking when IFC is asserted, it will continue from the point of interruption when again addressed as a talker. If listening when interrupted, the 7D20 will resume receiving data when again addressed as a listener.

SERVICE REQUEST

Service requests are used by the 7D20 to inform the controller of its operating status. A two step operation is available using the serial poll response for general status and a corresponding EVENT? response for detailed status reporting. When an operation occurs within the instrument which warrants attention, the SRQ signal line is asserted. The controller should execute a Serial Poll sequence followed by an EVENT?. Shown in Table 2 are the various status bytes generated by the 7D20. Each time an event occurs which generates a status byte and SRQ, a corresponding event code is put into a queue. A listing of all EVENT codes and their meaning can be found in either the Operators Manual or Programmers Reference Guide.

Each of the general status bytes shown in Table 2 can be selectively masked so that no SRQ will be generated if their respective operation occurs. The two exceptions to this rule are the Power On and Fatal Error status bytes—they cannot be masked.

Power On indicates the 7D20 has been turned on and is 'On-Line'. This status must be read in order to clear it from the system.

Fatal Error indicates the 7D20 has a hardware failure and is inoperative. No further communications are possible until the problem is located and corrected.

Two examples are shown to read the status byte from the 7D20 or any instrument which implements the serial poll function.

LOW LEVEL 4041 CODE:

```
100 WBYTE SPE
110 WBYTE ATN(64+devadd)
120 WBYTE ATN(MLA)
130 RBYTE status
140 WBYTE SPD
150 WBYTE ATN(UNT,UNL)
```

HIGH LEVEL 4041 CODE:

```
100 POLL status,device;devadd
```

Note: In the above example, the "devadd" parameter can be omitted from the poll statement and the controller will sequence through all of the bus addresses. If you know which device to poll, then set 'devadd' equal to the instrument address. This will cause only the desired device to be polled, reducing the time for the command to execute.

Table 2
7D20 STATUS BYTES

Title	Binary	Decimal	
		not busy	**busy**
(Device Dependant:)			
Fatal Error	1R1X 0011	227	243
(System:)			
Power On	010X 0001	65	81
Command Error	0R1X 0001	97	113
User Request	010X 0011	67	83
Execution Error	0R1X 0010	98	114
Internal Error	0R1X 0011	99	115
Execution Warning	0R1X 0101	101	117
Internal Warning	0R1X 0110	102	118
Operation Complete	0R0X 0010	66	82
No Status	000X 0000	0	16

R is asserted with the GPIB command: RQS ON. If the 7D20 is in the RQS OFF mode and polled, the status byte is sent without DIO7 asserted.

X is the busy bit, and is asserted only when the 7D20 is in AVE, AVEN, ENV, ENVN, or HOLDNEXT at the time of the SRQ. The first column of decimal values is without the busy bit set and the last column is with the busy bit set. Both columns assume RQS ON.

Event Codes

Event codes provide detailed operating status of the 7D20. Event codes are defined for functions ranging from self-test and command errors to the completion of asynchronous events such as a HOLDNEXT complete.

To receive the Event code information from the 7D20, send an 'EVENT?' (Event query) command; then input the ASCII string response.

4041 controller:

```
100 PRINT #10: "EVENT?"
110 INPUT #10:E$
```

*E$ can be changed to a numeric variable to input only the numeric portion of the response.

The 7D20 will also respond to the ERR? query command used by many Tektronix instruments. The response format is the same as with the EVENT? query.

Refer to EVENT QUERIES section of the Operators Manual for the code definitions. The 7D20 will store up to 40 event codes, but only 2 status bytes. For this reason an EVENT? should be executed each time a serial poll of the 7D20 is done.

Recommended 4041 routine to handle 7D20 SRQ's:

```
100 POLL status, device; 10                ! Execute Serial Poll
110 INPUT #10 PROMPT "EVENT?":EVTCODE ! Input Event? Response
```

Utility Software

Utility software is available as an optional accesory to the 7D20. The software consists of a series of subroutines and subprograms that perform common 7D20 GPIB functions such as data acquisition,data transfer,front-panel set-up, etc. Software is available on appropriate media for a variety of controllers, including Tektronix 4052A and 4041.

Additional Resources

Operators manual	070-3857-01
Programmers reference guide.	070-3205-00

TABLE 3
7D20 Command Set

CHANNEL 1 GROUP

Command/Query	Description
CH1 VOLTS: <volts/div>	Sets the channel 1 volts/div to the valid setting nearest <volts/div>.

<volts/div>	volts/div
<3.5E-3	5 mV[a]
3.5E-3 to <7.5E-3	5 mV
7.5E-3 to <15E-3	10 mV
15E-3 to <35E-3	20 mV
35E-3 to <75E-3	50 mV
75E-3 to <150E-3	100 mV
150E-3 to <350E-3	200 mV
350E-3 to <750E-3	500 mV
750E-3 to <1.5	1 V
1.5 to <3.5	2 V
3.5 to <7.5	5 V
>7.5	5 V[a]

Command/Query	Description
CH1? VOLTS	Responds with the channel 1 volts/div setting: **CH1 VOLTS:<volts/div>**
CH1 POSITION: <position>	Sets the channel 1 position to the valid setting nearest <position>.

<position>	position (div)
<−10.25	−10.24[a]
−10.24 to +10.22	<position>
>+10.23	+10.22[a]

Command/Query	Description
CH1? POSITION	Responds with the channel 1 position setting: **CH1 POSITION:<position>**
CH1 COUPLING: AC | GND | DC	Sets the channel 1 coupling to either AC, GND or DC.
CH1? COUPLING	Responds with channel 1's coupling setting: **CH1 COUPLING:<coupling>**
CH1 VARIABLE: ON | OFF	Enables/Disables CH 1 variable gain control.
CH1? VARIABLE	Responds with the channel 1 variable gain control status: **CH1 VARIABLE:<onoff>**
CH1 PROBE: <prbcode>	Command is ignored by the 7D20.
CH1? PROBE	Responds with probe attenuation encoding for channel 1: **CH1 PROBE:<prbcode>**

[a]execution error warning

Command/Query	Description
CH1?	Responds with all channel 1 settings: **CH1 VOLTS:<volts/div>, POSITION:<position>, COUPLING:<coupling>, VARIABLE:<onoff>, PROBE:<prbcode>** <volts/div> ::= <NR3> <position> ::= <NR2> <coupling> ::= AC | GND | DC <onoff> ::= ON | OFF <prbcode> ::= 1 | 10 | 100

CHANNEL 2 GROUP

Command/Query	Description
CH2 VOLTS: <volts/div>	Sets the channel 2 volts/div to the valid setting nearest <volts/div>.

<volts/div>	volts/div
<3.5E-3	5 mV[a]
3.5E-3 to <7.5E-3	5 mV
7.5E-3 to <15E-3	10 mV
15E-3 to <35E-3	30 mV
35E-3 to <75E-3	50 mV
75E-3 to <150E-3	100 mV
150E-3 to <350E-3	200 mV
350E-3 to <750E-3	500 mV
750E-3 to <1.5	1 V
1.5 to <3.5	2 V
3.5 to <7.5	5 V
≥7.5	5 V[a]

Command/Query	Description
CH2? VOLTS	Responds with the channel 2 volts/div setting: **CH2 VOLTS:<volts/div>**
CH2 POSITION: <position>	Sets the channel 2 vertical position to the valid setting nearest <position>.

<position>	position (div)
<−10.25	−10.24[a]
−10.24 to +10.22	<position>
>+10.23	+10.22[a]

Command/Query	Description
CH2? POSITION	Responds with the channel 2 position setting: **CH2 POSITION:<position>**
CH2 COUPLING: AC | GND | DC	Sets the channel 2 coupling to either AC, GND or DC.
CH2? COUPLING:	Responds with channel 2 coupling setting: **CH2 COUPLING:<coupling>**

[a]**execution error warning**

Command/Query	Description
CH2 VARIABLE: **ON** | **OFF**	Enables/Disables CH 2 variable gain control.
CH2? VARIABLE	Responds with the channel 2 variable gain control status: **CH2 VARIABLE:<onoff>**
CH2 INVERT: **ON** | **OFF**	Turns inversion of the channel 2 signal either ON or OFF
CH2? INVERT	Responds with the channel 2 inversion setting: **CH2 INVERT:<onoff>**
CH2? PROBE	Responds with probe encoding for channel 2: **CH2 PROBE:<prbcode>**
CH2?	Responds with all channel 2 settings: **CH2 VOLTS:<volts/div>,POSITION <position>, COUPLING:<coupling>,VARIABLE:<onoff>, INVERT:<onoff>,PROBE:<prbcode>** <volts/div> ::= <NR3> <position> ::= <NR2> <coupling> ::= AC | GND | DC <onoff> ::= ON | OFF <prbcode> ::= 1 | 10 | 100

TRIGGERING GROUP

Command/Query	Description
TRIGGER MODE: **P-P** | **AUTO** | **NORMAL]**	Sets trigger mode to either PEAK to PEAK, AUTO, or NORMAL.
TRIGGER? MODE	Responds with the triggering mode setting: **TRIGGER MODE:<mode>**
TRIGGER HOLDNEXT: **ON** | **OFF**	Turns HOLDNEXT either ON or OFF. Retains the same trigger level as previously set by P-P, AUTO, or NORM. Upon entering HOLD, an operation complete SRQ is generated if OPC:ON.
TRIGGER? HOLDNEXT	Responds with the hold next setting: **TRIGGER HOLDNEXT:<onoff>**
TRIGGER COUPLING: **AC** | **ACLFREJ** | **ACHFREJ** | **DCHFREJ** | **DC**	Sets trigger coupling to either AC, AC low-frequency reject, AC high-frequency reject, DC high-frequency reject, or DC.
TRIGGER? COUPLING	Responds with the triggering coupling: **TRIGGER COUPLING:<coupling>**
TRIGGER SOURCE: **MODE** | **CH1** | **CH2** | **LINE** | **EXT** | **EXT/10**	Sets trigger source to either MODE, CH1, CH2, LINE, EXT, or EXT/10. If MODE is selected, trigger source is CH1 when acquire mode is CH1, ADD, or BOTH, and CH 2 when Acquire mode is CH 2.

[a]execution error warning

Command/Query	Description
TRIGGER? SOURCE	Responds with the triggering source: **TRIGGER SOURCE:<source>**
TRIGGER SLOPE: PLUS I MINUS	Sets trigger slope to either PLUS or MINUS.
TRIGGER? SLOPE	Responds with the triggering slope: **TRIGGER SLOPE:<slope>**
TRIGGER LEVEL: <level>	Sets trigger level to the valid setting nearest <level>.

<level>	trigger level
<−6.425	−6.4[a]
−6.4 to +6.35	<level>
>+6.375	+6.35[a]

In P-P, +6.35 and −6.4 are scaled to correlate with the maximum and minimum values of the waveform.

Command/Query	Description
TRIGGER? LEVEL	Responds with the triggering level setting: **TRIGGER LEVEL:<level>**
TRIGGER POSITION: <division>	Sets position for displaying trigger point to the valid setting nearest <division>.

<division>	trigger point
<−1500.5	−1500[a] div
−1500.5 to <10.5	<division>
10.5 or greater	10[a] div

Restriction: Maximum position is 10th graticule division; minimum is 1500 divisions to left of graticule line zero. In ROLL or EXT digitizing modes, the minimum setting with ENV or AVE is 0.

Command/Query	Description
TRIGGER? POSITION	Responds with the trigger display position setting: **TRIGGER POSITION:<position>**
TRIGGER?	Responds with all triggering settings:

TRIGGER MODE:<mode>,HOLDNEXT: <onoff>,
COUPLING:<coupling>,SOURCE:<source>,
SLOPE:<slope>,LEVEL:<level>,
POSITION:<position>

```
<mode> ::= P-P I AUTO I NORM
<coupling> ::= AC I ACLFREJ I ACHFREJ I DCHFREJ I DC
<source> ::= MODE I CH1 I CH2 I LINE I EXT I EXT/10
<slope> ::= PLUS I MINUS
<level> ::= <NR3>
<position> ::= <NR1>
```

[a]execution error warning

Command/Query	Description

TIME/DIV GROUP

Command/Query	Display
HORIZONTAL **TIME**: <time/div>	Sets time/div to the valid setting nearest <time/div>.

<time/div>	time/div
<35E-9	50 nS[a]
35E-9 to <75E-9	50 nS
75E-9 to <150E-9	100 nS
150E-9 to <350E-9	200 nS
350E-9 to <750E-9	500 nS
750E-9 to <1.5E-6	1 μS
1.5E-6 to <3.5E-6	2 μS
3.5E-6 to <7.5E-6	5 μS
7.5E-6 to <15E-6	10 μS
15E-6 to <35E-6	20 μS
35E-6 to <75E-6	50 μS
75E-6 to <150E-6	100 μS
150E-6 to <350E-6	200 μS
350E-6 to <750E-6	500 μS
750E-6 to <1.5E-3	1 mS
1.5E-3 to <3.5E-3	2 mS
3.5E-3 to <7.5E-3	5 mS
7.5E-3 to <15E-3	10 mS
15E-3 to <35E-3	20 mS
35E-3 to <75E-3	50 mS
75E-3 to <150E-3	100 mS
150E-3 to <350E-3	200 mS
350E-3 to <750E-3	500 mS
750E-3 to <1.5	1 S
1.5 to <3.5	2 S
3.5 to <7.5	5 S
7.5 to <15	10 S
15 to <35	20 S
35 or greater	20 S[a]

HORIZONTAL**? TIME**	Responds to the time/div setting: **HORIZONTAL TIME:<time/div>**
HORIZONTAL **POSIT**ION: **ON | OFF**	Turns enable for horizontal position knob ON or OFF. Sets horizontal position to current knob setting or calibrated setting.
HORIZONTAL**? POSIT**ION	Responds with the horizontal position knob enable state: **HORIZONTAL POSITION:<onoff>**
HORIZONTAL **CLOCK**: **INT**ERNAL **| EXTP | EXTN**	Selects internal clock mode with time/div determined by <time/div> (INT) or external clock mode with positive (EXTP) or negative (EXTN) edge sensitivity, respectively.
HORIZONTAL**? CLOCK**	Responds with horizontal clock source: **HORIZONTAL CLOCK:<hclock>**

[a]**execution error warning**

Command/Query	Description
HORIZONTAL?	Responds with all the horizontal setting: **HORIZONTAL TIME:<time/div>, POSITION:<onoff>,CLOCK:<hclock>** <time/div> ::= <NR3> <hclock> ::= INTERNAL I EXTP I EXTN

DISPLAY GROUP

Command/Query	Description
DISPLAY **1:** **ON I OFF**	Turns the waveform 1 display either ON or OFF.
DISPLAY**? 1**	Responds with the display status of waveform 1: **DISPLAY 1:<onoff>**
DISPLAY **2:** **ON I OFF**	Turns the waveform 2 display either ON or OFF.
DISPLAY**? 2**	Responds with the display status of waveform 2: **DISPLAY 2:<onoff>**
DISPLAY **3:** **ON I OFF**	Turns the waveform 3 display either ON or OFF.
DISPLAY**? 3**	Responds with the display status of waveform 3: **DISPLAY 3:<onoff>**
DISPLAY **4:** **ON I OFF**	Turns the waveform 4 display either ON or OFF.
DISPLAY**? 4**	Responds with the display status of waveform 4: **DISPLAY 4:<onoff>**
DISPLAY **5:** **ON I OFF**	Turns the waveform 5 display either ON or OFF.
DISPLAY**? 5**	Responds with the display status of waveform 5: **DISPLAY 5:<onoff>**
DISPLAY **6:** **ON I OFF**	Turns the waveform 6 display either ON or OFF.
DISPLAY**? 6**	Responds with the display status of waveform 6: **DISPLAY 6:<onoff>**
DISPLAY **CSW:** <wfm #>	Selects waveform <wfm #> as the cursor waveform.
DISPLAY**? CSW**	Responds with the cursor waveform number: **DISPLAY CSW:<wfm #>**
DISPLAY **VE**CTOR: **ON I OFF**	Turns vector display either ON or OFF.

[a]execution error warning

Command/Query	**Description**
DISPLAY? VECTOR	Responds with the vector status: **DISPLAY VECTOR:<onoff>**
DISPLAY REFERENCE: ON I OFF	Turns the cursor waveform reference display display ON or OFF. Restriction: Reference waveform can only be turned on if the cursor waveform is in HMAG or VS mode.
DISPLAY? REFERENCE	Responds with the reference cursor waveform status: **DISPLAY REFERENCE:<onoff>**
DISPLAY RDOUT: ON I OFF	Turns display text lines 1, 2, 15, and 16 on or off. DISP RDOUT may be selected by means of the Utilities menu.
DISPLAY? RDOUT	Responds with readout display status: **DISPLAY RDOUT:<onoff>**
DISPLAY?	Responds with the waveform display information: **DISPLAY CSW:<csw>,1:<onoff>,2:<onoff>, 3:<onoff>,4:<onoff>,5:<onoff>,6:<onoff>, VECTOR:<onoff>,REFERENCE:<onoff>, RDOUT:<onoff>**
COPY 1: <wfm #>	Copies waveform 1 to waveform <wfm #>.
COPY 2: <wfm #>	Copies waveform 2 to waveform <wfm #>.
COPY 3: <wfm #>	Copies waveform 3 to waveform <wfm #>.
COPY 4: <wfm #>	Copies waveform 4 to waveform <wfm #>.
COPY 5: <wfm #>	Copies waveform 5 to waveform <wfm #>.
COPY 6: <wfm #>	Copies waveform 6 to waveform <wfm #>.

<onoff> ::= ON I OFF
<wfm #> ::= 1 I 2 I 3 I 4 I 5 I 6

[a]execution error warning

Command/Query	Description

CURSOR WAVEFORM GROUP

Command/Query	Description
CSW VOLTS: <volts/div>	This command is ignored by the 7D20.
CSW? VOLTS	Responds with the cursor waveform's display volts/div: **CSW VOLTS:<volts/div>**
CSW VXPD: <setting>	Vertically expands or compresses the cursor waveform by selecting the display volts/div for the cursor waveform.

<setting>	display volts/div
<−2.5	2 steps lower than acquired[a]
−2.5 to <−1.5	2 steps lower than acquired
−1.5 to <−0.5	1 step lower than acquired
−0.5 to <0.5	same volts/div as acquired
0.5 to <1.5	1 step higher than acquired
1.5 to <2.5	2 steps higher than acquired
2.5 or greater	2 steps higher than acquired[a]

Restriction: Waveform memory displays may be compressed or expanded no more than two steps from the acquired scale factor (in a 1-2-5 sequence) without incurring error. VXPD does not affect active waveforms except in HMAG, or Y vs X modes when the REFerence waveform is displayed.

CSW? VXPD	Responds with the cursor waveform's displayed volts/div expansion factor: **CSW VXPD:<setting>**
CSW POSITION: <position>	Sets cursor waveform's displayed Vertical position.

<position>	position offset
<−5.14	−5.12[a]
−5.12 to 5.08	<position>
+5.10 or greater	+5.08[a]

The position offset is added to the original acquired position value. Vertical expansion will also affect the actual displayed position. Position does not affect active waveforms except in HMAG, or Y vs X modes when the REF waveform is displayed.

CSW? POSITION	Responds with the cursor waveform's display position: **CSW POSITION:<position>**
CSW HMAG: **ON | OFF | ALLON | ALLOFF**	ON and OFF turns horizontal magnification on or off, respectively, for the CSW waveform. ALLON and ALLOFF affects all displayed waveforms.
CSW? HMAG	Responds with the cursor waveform's hmag setting: **CSW HMAG:<hmag>** An OFF response indicates the cursor waveform is not magnified, but other waveforms may be.

[a]execution error warning

Command/Query	Description
CSW VS: <wfm #> |	Displays the cursor waveform versus waveform <wfm #>.
0	Turns cursor waveform VS display OFF.
CSW? VS	Responds with the cursor waveform's versus status: **CSW VS:<versus>** To interrogate the VS status of waveforms other than the cursor waveform, the cursor waveform must be reselected.
CSW?	Responds with the display information for cursor waveform: **CSW VOLTS:<volts/div>,VXPD:<setting>, POSITION:<position>, HMAG<hmag>,VS:<versus>** <volts/div> ::= <NR3> <setting> ::= -2 | -1 | 0 | 1 | 2 <position> ::= <NR2> <hmag> ::= ON | OFF | ALLON | ALLOFF <versus> ::= <wfm #> | 0 <wfm #> ::= 1 | 2 | 3 | 4 | 5 | 6

ACQUISITION GROUP

Command/Query	Description
AQR MODE: **CH1 | BOTH | ADD | CH2**	Sets acquire mode to either CH1, BOTH, ADD, or CH2.
AQR? MODE	Responds with the acquire mode setting: **AQR MODE:<mode>**
AQR HOLD: **ON | OFF**	Turns acquire hold either ON or OFF.
AQR? HOLD	Responds with the acquire hold status: **AQR HOLD:<hold>**
AQR SET: <n>	Sets N to <n>. N is the number of averages or envelopes, as appropriate.

<n>	N
<6	8[a]
6 to <12	8
12 to <24	16
24 to <48	32
48 to <96	64
96 to <192	128
192 to <384	256
384 or greater	256[a]

Command/Query	Description
AQR? SET	responds with <n>: **AQR SET:<n>**

[a]execution error warning

Command/Query	Description
AQR TYPE: NORMAL	Acquires waveform(s) normally. Terminates HOLD condition.
AVE	Averages waveform(s) continuously. Terminates HOLD. Averages Wfm 1 or Wfm 2 as determined by AQR Mode control. A GPIB AVE command received during AVE is ignored. AVE terminates ENV, ENVN, SETN, and AVEN.
AVEN	Averages waveform(s) N times. If AVEN is received during AVE then AVE is terminated and AVEN executes. Upon completion of the AVEN operation, an operation complete SRQ is generated when the 7D20 enters HOLD.
ENV	Envelopes waveform(s) continuously. Terminates HOLD. Wfm 1 or Wfm 2, as determined by AQR MODE, acquire envelope processed data. GPIB ENV commands received during execution are ignored. ENV terminates AVE, AVEN, SETN, and ENVN.
ENVN	Envelopes waveform(s) N times. Upon completion of the ENVN operation, an operation complete SRQ is generated when the 7D20 enters HOLD.
AQR? TYPE	responds with the acquire type status: **AQR TYPE:<type>**
AQR?	returns the current acquisition status: **AQR MODE:<mode>,TYPE:<type>, HOLD:<hold>,SET:<n>** <mode> ::= CH1 | BOTH | ADD | CH2 <hold> ::= ON | OFF <n> ::= 8 | 16 | 32 | 64 | 128 | 256 <type> ::= NORMAL | AVE | AVEN | ENV | ENVN

CURSOR GROUP

Command/Query	Description
CURSOR MODE: INDEP | ALIGN	Sets cursor mode to INDEP or ALIGN.
CURSOR? MODE	Responds with the cursor mode: **CURSOR MODE:<mode>**
CURSOR DELTA: ON | OFF	Turns cursor 2 either ON or OFF.
CURSOR? DELTA	Responds with the cursor delta mode: **CURSOR DELTA:<onoff>**

[a]execution error warning

Command/Query	**Description**
CURSOR 1: <point #>	Sets cursor 1 to the valid position nearest <point#>.

	<point #>	point #
p/w = 820:	<−0.5	0[a]
	−0.5 to <819.5	<point #>
	>819.5	819[a]
p/w = 1024:	<−0.5	0[a]
	0.5 to <1023.5	<point #>
	⩾1023.5	1023[a]

CURSOR? 1

Responds with cursor 1's point #:

CURSOR 1:<point #>

CURSOR 2: <point #>

Sets cursor 2 to the valid position nearest <point #>.

	<point #>	point #
p/w = 820:	<−0.5	0[a]
	−0.5 to <819.5	<point #>
	>819.5	819[a]
p/w = 1024:	<−0.5	0[a]
	0.5 to <1023.5	<point #>
	⩾1023.5	1023[a]

Restriction: Cursor 2 may not be set to a position less than Cursor 1.

CURSOR? 2

Responds with cursor 2's status:

CURSOR 2:<point #>

CURSOR?

Responds with the cursor status:

CURSOR MODE:<mode>,DELTA:<onoff>,
1:<point #>,2:<point #>

<mode> ::= INDEP | ALIGN
<point #> ::= <NR1>
<onoff> ::= ON | OFF

COORD? VCRD

Responds with the displayed vertical cursor coordinate ([delta]V value if cursor delta mode is on):

COORD VCRD:<vert coord>

COORD? HCRD

Responds with the displayed horizontal cursor coordinate ([delta]t value if cursor delta mode is on):

COORD HCRD:<horz coord>

COORD?

Responds with the displayed cursor coordinates:

COORD VCRD:<vert coord>,HCRD:
<horz coord>
<vert coord> ::= <NR3>
<horz coord> ::= <NR3>

[a]execution error warning

STORED SETTINGS GROUP

Command/Query	Description
STORE <set #>	Stores front panel settings in memory location <set #>.
RECALL <set #>	Recalls front panel settings from memory location <set #>. <set #> ::= 1 | 2 | 3 | 4 | 5 | 6

DEVICE TRIGGER

NOTE

Device Trigger is Bus-unique; that is, no front-panel equivalent exists in the 7D20.

Command/Query	Description
DT OFF |	Disables 7D20 response to GET.
HOLD | **HOLDNEXT** | **AVE** | **AVEN** | **ENV** | **ENVN** | **NORMAL** |	Executes deferred command on GET interface message. Only one deferred command may be triggered from each GET. Clears previous pending deferred commands without executing them.
DT?	Responds with deferred command to be executed on next GET interface message. **DT <deferred command>** <deferred command> ::= OFF | HOLD | HOLDNEXT | AVE | AVEN | ENV | ENVN | NORMAL

INITIALIZATION GROUP

NOTE

The Initialization Group is Bus-unique.

Command/Query	Description
INIT	Full initialization equivalent to an INIT PANEL, WAVEFORM, GPIB command.
INIT PANEL	Initializes front panel settings as described in front-panel section of operator's manual.
INIT WAVFRM	Initializes waveform memories 1-6 by setting the vertical data and vertical zero reference (VZR) values to zero and display attributes (VPUP, VPDN, VXPD, VCMP, HMAG, VS, and REF) to the off condition.

[a]**execution error warning**

Command/Query	Description
INIT GPIB	Initializes the status of the following functions, which affect GPIB operation:

Sets:

DT	OFF
RQS	ON
CER	ON
EXR	OFF
INR	ON
SYS	ON
EXW	ON
OPC	ON
USER	ON
PID	OFF
DEBUG	OFF

LONGFORM ON
RECORDING OFF
DATA MEMORY:1
DATA ENCODING:BINARY
DATA INTERPOLATE:OFF

Clears:
Event Buffer

SELFTEST

The 7D20 performs a selftest routine that tests 63 circuit modules. If a fault condition is detected, an SRQ is generated with an Internal Error status byte. An event query may be used to determine the specific module at fault. When the selftest is complete, an SRQ and an Operation Complete status byte are generated. Upon completion or exit from the selftest routine, the front panel settings are restored to the status prior to selftest and the waveform memory is initialized.

Command/Query	Description
TEST START |	Initiates selftest routine.
CONTINUE |	Continues selftest routine having halted on fault condition.
EXIT	Exits selftest routine that has halted on fault condition. Will not terminate an executing selftest.
TEST	Same as TEST START.

CALIBRATION GROUP

Command/Query	Description
CAL DISPLAY |	Turns front-panel display calibration pattern on.
DDAC |	Initiates vertical display DAC calibration routine.
POSOFF |	Initiates vertical position offset calibration routine.
POSGAIN |	Initiates vertical position gain calibration routine.
RAMP |	Initiates fast ramp calibration routine.
OFF |	Turns off selected calibration routine.
CAL?	Responds with calibration selection status: **CAL DISPLAY | DDAC | VERTICAL | POSOFF | POSGAIN | RAMP | OFF**
RAMP?	Responds with fast ramp calibration routine offset gain and count values: **RAMP GAIN:<gain>,COUNT:<count>** <gain> ::= <NR2> <count> ::= <NR1>

SERVICE REQUEST GROUP

NOTE

These Commands and Queries are all bus-unique; that is, no front-panel equivalents exist.

Command/Query	Description
RQS ON | OFF	Enables/disables all service request functions except power-on.
RQS?	Responds with the RQS service request status: **RQS<onoff>**
CER ON | OFF	Enables/disables command-error service request.
CER?	Responds with the CER service request status: **CER<onoff>**
EXR ON | OFF	Enables/disables execution-error service request.
EXR?	Responds with the EXR service request status: **EXR<onoff>**
INR ON | OFF	Enables/disables internal-error service request.
INR?	Responds with the INR service request status: **INR<onoff>**
EXW ON | OFF	Enables/disables execution-warning service request.
EXW?	Responds with the EXW service request status: **EXW<onoff>**
OPC ON | OFF	Enables/disables operation-complete service request.
OPC?	Responds with the OPC service request status: **OPC<onoff>**
USER ON | OFF	Enables/disables front panel generated RQS Key service request.
USER?	Responds with the USER service request status: **USER<onoff>**
PID ON | OFF	Enables/disables probe-ID service request.
PID?	Responds with the PID service request status: **PID<onoff>**
SRQ?	Responds with all service request settings: **RQS<onoff>;CER<onoff>;EXR<onoff>; INR<onoff>;EXW<onoff>;OPC<onoff>; USER<onoff>;PID<onoff>** <onoff> ::= ON | OFF

[a]execution error warning

WAVEFORM PREAMBLE GROUP

NOTE

The Waveform Preamble Group is Bus-unique.

NOTE

WFMPRE commands that are not used by the 7D20 will be accepted as part of the waveform preamble or individually, but will be ignored. No error or warning will be issued. This permits preamble outputs to be input without generating unnecessary service requests.

Command/Query	**Description**
WFMPRE **WFID:**	Command is not used by 7D20. Destination waveform is set with the DATA command.
WFMPRE? **WFID**	Responds with the current source waveform's number and indicates if interpolation will be applied to curve data: **WFMPRE WFID:<wfmid>**
WFMPRE **ENC**DG: **ASCII** | **BIN**ARY	Command is not used by 7D20. Data encoding is set with the DATA command.
WFMPRE? **ENC**DG	Responds with the current curve encoding: **WFMPRE ENCDG:<encdg>**
WFMPRE **NR**.PT: 820 | 1024	Sets number of points to input to 820 | 1024.
WFMPRE? **NR**.PT	Responds with the current source waveform's points/waveform: **WFMPRE NR.PT:<p/w>**
WFMPRE **PT.**FMT: **Y**	Command is not used by 7D20. Point format is fixed as Y format.
WFMPRE? **PT.**FMT	Responds with a point format of Y: **WFMPRE PT.FMT:Y**
WFMPRE **XI**NCR: <x increment>	Sets time between points to <x increment>. <x increment> ::= time/div / 80 points/division for 820 point waveforms. <x increment> ::= time/div / 100 points/division for 1024 point waveforms
WFMPRE? **XI**NCR	Responds with the time between points of the current source waveform: **WFMPRE XINCR:<x increment>**
WFMPRE **PT.O**FF: <point #>	Selects the waveform point at which the cursor reads time zero. Range is −150,000 ≤point # ≤1023, where point 0 is the left-most displayed point on the waveform. An execution error warning is given if an out of range condition occurs.
WFMPRE? **PT.O**FF	Responds with the trigger point #: **WFMPRE PT.OFF:<point #>**

[a]**execution error warning**

Command/Query	**Description**
WFMPRE **XZ**ERO: 0	Command is not used by 7D20. XZERO is fixed as 0.
WFMPRE? XZERO	Responds with a XZERO of 0: **WFMPRE XZERO:0**
WFMPRE **XU**NIT: S	Sets displayed horizontal scale factor unit for waveforms input from the curve command.
WFMPRE? XUNIT: <xunit>	Responds with XUNIT, **WFM**PRE **XU**NIT:**<xunit>**
WFMPRE **YMU**LT: <y multiplier>	Sets vertical scale (volts/div) to <y multiplier>.
WFMPRE? YMULT	Responds with the current source waveforms vertical scale factor: **WFMPRE YMULT:<y multiplier>**
WFMPRE **YZ**ERO: <y zero>	Sets Y zero to <y zero>. <y zero> ::= -(volts/div [x] VZR (vertical zero reference)) Where VZR is prior to vertical expansion, or positioning.
WFMPRE? YZERO	Responds with the current source waveform vertical zero: **WFMPRE YZERO:<y zero>**
WFMPRE **YU**NIT: V	Sets displayed vertical scale factor unit for waveforms input from the CURVE command.
WFMPRE **YU**NIT: <y unit>	Responds with YUNIT, **WFM**PRE **YU**NIT: <y unit>
WFMPRE **BY**T/NR: 1	Command is not used by 7D20. Binary data field width is fixed as one byte.
WFMPRE? BYT/NR	Responds with a field width of 1 byte: **WFMPRE BYT/NR:1**
WFMPRE **BN**.FMT: **LF**	Command is not used by 7D20. Binary number format is fixed as binary-fraction.
WFMPRE? BN.FMT	Responds with a number format of binary fraction: **WFMPRE BN.FMT:LF**
WFMPRE **BI**T/NR: 8	Command is not used by 7D20. Binary data precision is fixed as eight bits.
WFMPRE? BIT/NR	Responds with a precision of 8 binary its: **WFMPRE BIT/NR:8**
WFMPRE? **CRVC**HK	Responds with the form of curve error check used (checksum): **WFMPRE CRVCHK: CHKSM0**

[a]execution error warning

Command/Query	Description
WFMPRE?	Responds with the current source waveform's preamble: If ASCII: **WFMPRE WFID:<wfmid>,ENCDG:ASCII, NR.PT:< p/w>,PT.FMT:Y,XINCR:<x increment>,PT.OFF:<point #>,XZERO:0, XUNIT:<xunit>,YMULT:<y multiplier>, YZERO:<y zero>,YUNIT:<y unit>,** If binary: **WFMPRE WFID:<wfmid>,ENCDG:BINARY, NR.PT:<p/w>,PT.FMT:Y,XINCR:<x increment>,PT.OFF:<point #>,XZERO:0,XUNIT: <xunit>,YMULT:<y multiplier>,YZERO: <y zero>,YUNIT:<y unit>, BYT I NR:1,BN.FMT:LF, BIT I NR:8,CRVCHK:CHKSM0,** <wfm #> ::= 1 I 2 I 3 I 4 I 5 I 6 <encdg> ::= ASCII I BINARY <x increment> ::= <NR3> = time/div / points/div <point #> ::= <NR1> <x multiplier> ::= <NR3> = vertical scale factor <y zero> ::= <NR3> = -(VZR * vertical scale factor) <wfmid> ::= W <wfm #> I I W <wfm #>, where I indicates interpolated data. <x unit> ::= Capability to be set to any one of the the 7D20 single character set, except for (,), (;), or ESC (escape). <y unit> ::= Same as above. If x unit or y unit is set to "&", the 7D20 will interpret this as a blank unit or space.

WAVEFORM CURVE

NOTE
Waveform Curve is Bus-unique.

Command/Query	Description
CURVE <ASCII curve>	Loads destination waveform with <ASCII curve>.
<binary block curve>	Loads destination waveform with <binary block curve>. <ASCII curve> ::= <ASCII point>, <ASCII point>... <binary block curve> ::= %<binary count> <binary point>...<checksum>
CURVE?	Responds with the current source waveform's curve: If ASCII: **CURVE <ASCII point>,<ASCII point>...** If binary: **CURVE %<binary count><binary point>... <checksum>** <ASCII point> ::= <NR2> <binary count> ::= two bytes representing number of data points +1 <binary point> ::= 8-bit byte (00 FF) <checksum> ::= 2's complement of the modulo 256 sum of the preceding binary data bytes and the binary count but not the "%" preceding the binary byte count.

WAVEFORM PREAMBLE AND CURVE

NOTE

Waveform Preamble and Curve is Bus-Unique.

Command/Query	Description
WAVFRM?	Responds with <preamble>;<curve> <preamble> ::= response to **WFM**PRE? <curve> ::= response to **CURVE?**
DATA ENCDG: **ASC**II | **BIN**ARY	Sets encoding for transmission of waveform data from 7D20 as binary or ASCII.
DATA? ENCDG	Responds with curve data encoding for waveforms transmitted from 7D20: **DATA ENCDG:<encdg>**
DATA INTERPOLATE: **ON | OFF**	Sets 820 to 1024-point interpolation mode for all Extended Real-time Waveforms transmitted from 7D20.
DATA? INTERPOLATE	Responds with status of interpolation mode: **DATA INTERPOLATE:<onoff>**
DATA MEMORY: <wfm #>	Sets memory destination and source for all waveform data transmission to and from 7D20.
DATA? MEMORY	Responds with source/destination for waveform data transmission: **DATA MEMORY:<wfm #>**
DATA?	Responds with data command status: **DATA ENCDG:<encdg>,** **INTERPOLATE:ON | OFF,** **MEMORY:<wfm #>** <encdg> ::= ASCII | BINARY <wfm #> ::= 1 | 2 | 3 | 4 | 5 | 6

ªexecution error warning

READOUT/TEXT

NOTE

Readout/Text is Bus-Unique.

Command/Query	Description
TEXT "text"	Displays text in center 12 lines of CRT.
TEXT?	Responds with the center 12 lines of display text: **TEXT "<line 3><CR><line 4><CR><line 5><CR>... <line 12><CR><line 13><CR><line 14><CR>"** <line 3> ::= third line of display <line 4> ::= fourth line of display . . . <line 13> ::= thirteenth line of display <line 14> ::= fourteenth line of display

NOTE

A double quote (") within a line must be sent as two double quotes (" ").

Command/Query	Description
RDOUT?	Responds with the four lines of readout: **RDOUT "<line 1><CR><line 2><CR><line 15><CR><line 16><CR>"** <CR>::= carriage return

PROGRAMMING AIDS

NOTE

The following commands and queries are bus-unique.

Command/Query	Description
ID?	Responds with the 7D20's ID: **ID TEK/7D20,V81.1,<rom>.<patch>** <rom> ::= 2 character ROM version <patch> ::= 2 digit PATCH revision

***execution error warning**

Command/Query	Description
SET?	Responds with the front panel settings: **<ch1>;<ch2>;<horz>;<aqr>;<csw>;<disp>; <trig>;<curs>** <ch1> ::= response to CH1? <ch2> ::= response to CH2? <trig> ::= response to TRIG? <horz> ::= response to HORIZ? <disp> ::= response to DISP? <csw> ::= response to CSW? <aqr> ::= response to AQR? <curs> ::= response to CURS?
HELP?	Responds with a list of all valid command headers: **CH1, CH2, TRIGGER, HORIZONTAL, DISPLAY,COPY, CSW, AQR, CURSOR, STORE, RECALL,DT, INIT, TEST, CAL, RQS, CER, EXR, INR, EXW, OPC, USER, PID, SRQ, WFMPRE, CURVE, DATA, WAVFRM, TEXT, DEBUG, RECORDING, LONGFORM**
DEBUG **ON | OFF**	Turns on debug option. Commands are displayed in the order of occurrence on the screen area normally used by text commands and menus. A command or query that follows an EOI will clear the text area before it is displayed. If an error occurs, the erroneous item is displayed, followed by the 16-bit error code describing the type of error. Control characters are displayed as "@". Lower case characters are displayed as upper-case characters. An EOI is displayed as an exclamation mark (!). ASCII waveform transmissions are displayed, binary waveform transmissions are not.
DEBUG?	Responds with the debug status: **DEBUG <onoff>**
LONGFORM **ON | OFF**	Affects results when querying response to either receive longform or shortform of Command or Query.
LONGFORM?	Responds with the LONGFORM status: **LONGFORM ON or LO OF**

[a]execution error warning

Command/Query	Description
RECORDING ON | OFF	By using a controller and the RECORDING command, it is possible to get longer record lengths. The RECORDING mode works only from 100 mS/div to 20 S/div, including EXT clock mode. In the other time/div regions, RECORDING will have no effect. RECORDING may be used with any acquire mode, and with the normal, AVE and ENV, acquire types. RECORDING allows the user to read out points using a CURVE? command, 1k at a time. The waveforms appear to roll across the screen, just as in non-RECORDING mode, but every time 1024 new points have been acquired they are copied into a RAM buffer accessible via the CURVE? command. There is a separate buffer for each channel. The user knows it is time to do the next CURVE? command, when an OPC service request (66 or 82) is produced by the 7D20. If the user receives the service request before the 1024 points have been read out from the CURVE?, then an overrun condition has occurred, and either the controller needs to take less time to read in the curve, or you should use a slower time/div.
RECORDING?	Responds with RECORDING status: **RECORDING ON or RECORDING OFF**

[a]execution error warning

PROGRAMMING EXAMPLES

4041 BASIC

```
*+ ************************************************************+
* PUTWFM - Send waveform data (ASCII or BINARY) to the 7D20.
* January 30,1983
*
* Copyright [c] 1983, Tektronix, Inc. All rights reserved.
*
* REQUIRED EQUIPMENT:
*    None.
* PURPOSE:
*    Sends waveform data to the assigned 7D20 using the information
*    passed in the call statement. The waveform transfer mode is
*    determined from the waveform preamble.
* CALLING FORMAT:
*    CALL PUTWFM(addr,memnum,wfpre$,wfdata)
* INPUTS:
*    1. addr ---- GPIB primary address of the 7D20 (0 - 30).
*    2. memnum -- 7D20's Waveform memory to which data is to be
*                 read.
*    3. wfpre$ -- Preamble of the waveform to be sent.
*    4. wfdata -- Array of the waveform data to be sent. The values
*                 should be 0 - 255 for BINARY transfer and -5.12 -
*                 5.08 for ASCII transfer.
* OUTPUTS:
*    The 7D20 is directed to input waveform data into the selected
*    waveform memory, followed by the waveform preamble and data.
* VARIABLES USED:
*    addr ------- GPIB primary address of 7D20.
*    memnum ----- 7D20 display memory to which data is sent.
*    wfpre$ ----- String containing all attributes of the waveform
*                 data (WFDATA).
*    wfdata ----- Array containing data to be transferred. The transfer
*                 is in either Binary Block or ASCII string format
*                 depending on the encoding described in the preamble.
*    image$ ----- String used to construct the ASCII string to send. A
*                 comma is inserted between each data value (ie. -2.15,
*                 2.22,0.0,...).
*    bufr$ ------ String containing the entire waveform for transfer.
*                 used in both ASCII and BINARY transfers to increase
*                 the GPIB transfer time.
* ROUTINES CALLED:
*     None.
* POSSIBLE ERRORS:
*     1. Any of the input variables not defined.
*     2. WFPRE$ number of points does not match the length of WFDATA.
*_ ************************************************************ _

690 Sub putwfm(addr,memnum,wfpre$,wfdata) local bufr$,image$,wfpts
        ,put_bin,put_rtn
700     Integer wfpts
710     Init var bufr$
720     Print #addr:"da mem:"&str$(memnum)
730     Print #addr:wfpre$
740     If seg$(wfpre$,pos(wfpre$,":",pos(wfpre$,"enc",1))+1,3)=
        "bin" then goto put_bin
750     Dim bufr$ to 6000
760     Wfpts=valc(wfpre$,pos(wfpre$,"nr",1))
770     Image$="fa,x,512(fg,','),"&str$(wfpts-513)&"(fg,','),fg"
780     Putmem buffer bufr$ using image$:"curve",wfdata
790     Print #addr:bufr$
800     Goto put_rtn
810 Put_bin:     dim bufr$ to 1040
820     Print #addr:"curve ";
830     Putmem buffer bufr$ using "+8%":wfdata
840     Print #addr:bufr$
850 Put_rtn:     delete var bufr$,image$,wfpts
860     Compress
870     Return
880     End
```

```
* +**************************************************************+
* GETWFM - Get Waveform
* January 30, 1983
*
* Copyright [c] 1983, Tektronix, Inc. All rights reserved.
* REQUIRED EQUIPMENT:
*    None.
* PURPOSE:
*    To acquire waveform data and attributes from the desired
*    7D20. The memory display and transfer format information
*    is passed in the call to this routine. The routine sets-
*    up the 7D20 and inputs both the preamble and waveform
*    data.
*    The routine uses 200 as the I/O "logical unit number". This
*    number should not be used in any mainline section of code.
* CALLING FORMAT:
*    CALL GETWFM(addr,memnum,mode$,wfpre$,wfdata)
*
*    For example, to get the waveform from memory display #5 of
*    the 7D20 at address 3 in the BINARY format.
*      Call GETWFM(3,5,"bin",wfpre$,wfdata)
* INPUTS:
*    1. addr ----- GPIB primary address of the 7D20 (0 -30).
*    2. memnum --- Waveform memory number from which the data
*                  is to be transfered.
*    3. mode$ ---- "ASC" or "BIN" depending on the type of
*                  transfer to be performed.
* OUTPUTS:
*    1. Array (WFDATA) is returned with the waveform data.
*       WFDATA is auto-dimensioned according to the infor-
*       mation returned into WFPRE$. If the transfer mode
*       is set to "binary", the array WFDATA is defined to
*       be INTEGER; otherwise short floating point numeric.
*    2. The waveform attributes are returned in the string
*       WFPRE$.
* VARIABLES USED:
*    addr -------- GPIB primary address of the 7D20.
*    memnum ------ 7D20 memory display location to extract the
*                  waveform from.
*    mode$ ------- String used to select the transfer mode to
*                  be used (ASC or BIN).
*    wfpre$ ------ String which the waveform attributes are
*                  returned in (dimensioned to 200 characters).
*    wfdata ------ Array waveform data is returned in (INTEGER if
*                  MODE$="BIN" and FLOATING POINT NUMERIC if MODE$=
*                  "ASC").
*    bufr$ ------- String used to input the waveform array from the
*                  GPIB (temporary).
*    temp$ ------- String used to store the ascii characters preceeding
*                  the binary curve (temporary).
* ROUTINES CALLED:
*    None.
* ERRORS:
*    1. Possible invalid data will be received if the 7D20 is
*       not in "HOLD" and the memory display selected is cur-
*       rently being acquired.
* _**************************************************************_
```

```
890 Sub getwfm(addr,memnum,mode$ var wfpre$,wfdata) local bufr$,
        temp$,wfpts,get_asc,get_rtn
 900    Delete var bufr$,wfdata
 910    Init var bufr$,wfpre$,wfdata,wfpts
 920    Dim bufr$ to 6000,wfpre$ to 200
 930    Open #200:"gpib0(pri="&str$(addr)&",eom=<0>,eoa=<0>):"
 940    Print #200:"da mem:"&str$(memnum)&",encdg:"&mode$
 950    Input #200 prompt "wfmpre?":wfpre$
 960    Wfpts=valc(wfpre$,pos(wfpre$,"nr",1))
 970    Print #200:"curve?"
 980    If mode$="asc" then goto get_asc
 990    Integer wfdata(wfpts)
1000    Input #200 buffer bufr$:bufr$
1010    Getmem buffer bufr$ dels "%" using "FA,+8%":temp$,wfdata
1020    Goto get_rtn
1030 Get_asc:     dim wfdata(wfpts)
1040    Input #200 buffer bufr$:wfdata
1050 Get_rtn:     delete var bufr$,temp$,wfpts
1060    Compress
1070    Return
1080    End
```

```
*+ ***********************************************************+
* NORMALIZ - Normalize the waveform array to Volts/Seconds.
*
* Copyright [c] 1983, Tektronix, Inc. All rights reserved.
*
* REQUIRED EQUIPMENT:
*   None.
* PURPOSE:
*   Normalizes the waveform array into amplitude values. The
*   array is passed into the routine in either values of 0-
*   255 if in binary or -5.12 to 5.08 if in ascii mode. The
*   resultant array (normdata) will represent the actual voltage
*   level of the signal acquired with the ground reference
*   taken into account.
* CALLING FORMAT:
*   CALL NORMALIZ(wfdata,wfpre$,normdata)
* INPUTS:
*   1. wfdata ---- The input array to be normalized (from GETWFM).
*   2. wfpre$ ---- String containing the arrays attributes (from
*                  GETWFM).
* OUTPUTS:
*   1. normdata -- Normalized array.
* VARIABLES USED:
*   wfdata ------- Input array from GETWFM routine.
*   wfpre$ ------- Input array attributes from GETWFM routine.
*   normdata ----- Normalized array.
*   wfpts -------- temporary used to dimension array NORMDATA.
*   yzero -------- temporary containing the input arrays ground
*                  reference level contained in WFPRE$.
*   ymult -------- temporary containing the the input arrays Volts/
*                  division setting. Contained in WFPRE$.
*   mode$ -------- temporary containing the input arrays transfer
*                  mode (ASCII or BINARY). Contained in WFPRE$.
* ROUTINES CALLED:
*   None.
* POSSIBLE ERRORS:
*   1. WFDATA,WFPRE$ not defined.
* _***********************************************************_
```

```
750 Sub normaliz(wfdata,wfpre$ var normdata) local mode$,
        norm_asc,ymult,yzero,wfpts,mode$
760     Init var normdata
770     Wfpts=valc(wfpre$,pos(wfpre$,"nr",1))
780     Yzero=valc(wfpre$,pos(wfpre$,"yz",1))
790     Mode$=seg$(wfpre$,pos(wfpre$,":",pos(wfpre$,"enc",1))+1,3)
800     Ymult=valc(wfpre$,pos(wfpre$,"ym",1))
810     Dim normdata(wfpts)
820     If mode$="asc" then goto norm_asc
830     Normdata=(((wfdata-128)/25)*ymult)-yzero
840     Return
850 Norm_asc:     normdata=(wfdata*ymult)+yzero
860     Return
870     End
```

4052A BASIC

```
*+ ***************************************************************** +
* TEKTRONIX INSTRUMENTATION SOFTWARE LIBRARY UTILITY ROUTINES
* GETWFM - Get Waveform Data from 7D20.
* February 1, 1983
*
* Copyright [c] 1983, Tektronix, Inc. All rights reserved.
*
* REQUIRED EQUIPMENT:
*    1.  R14A GPIB Enhancement ROM Pack
* PURPOSE:
*    To aquire waveform data and preamble information from the
*    7D20.  Encoding mode, waveform memory number, and address
*    of the 7D20 are passed in the call to this routine.  The
*    waveform data is passed back out in an array, WFDATA, which
*    is either unpacked Binary Block data (integer values between
*    0 and 255) or floating point data (values between -5.12 and
*    +5.08), depending on whether the encoding was ASCII or
*    Binary.
* CALLING FORMAT:
*    CALL Getwfm(Addr,_memnum,_mode$;Wfdata,Wfpre$)
* INPUTS:
*    Addr ------- GPIB address (1 - 30) of 7D20.
*    _memnum ---- 7D20's Waveform Memory from which data is
*                 to be read.
*    _mode$ ----- Data transfer mode desired.  Legal modes are
*                 "ASC" or "BIN".
* OUTPUTS:
*    Wfdata --- An array filled with the waveform data. If the
*               transfer mode is ASCII, the values are from
*               -5.12 to +5.08.  If the transfer mode is Binary,
*               the values of Wfdata will range from 0 to 255.
*               The size to which this array is diminsioned will
*               be either 820 or 1024, depending on the setting
*               of the 7D20's timebase.
*    Wfpre$ --- This is a long character string which contains
*               the Waveform Preamble.
* VARIABLES USED: (All Local)
*    Wfpts ----- Size of Wfdata array (number of elements).
*    Error ----- Local variable for receiving error code from
*                R14 ROM Pack BININ routine.
*    Temp ------ Local variables for intermediate holding of
*    Temp$ ----- values (junque).
*    Wfpts ----- Size (number of elements) to which Wfdata is
*                to be diminsioned.
* ROUTINES CALLED:
*    None.
* POSSIBLE ERRORS:
*    1.  Transfer error as reported by R14A BININ.
*- ***************************************************************** -
```

```
1000 SUB Getwfm(Addr,_memnum,_mode$;Wfdata,Wfpre$)
1010    LOCAL Error,Temp,Temp$,Wfpts
1020    DELETE Wfpre$,Wfdata
1030    DIM Wfpre$(200)
1040    PRINT @Addr:"DA MEM:";_memnum;",ENCDG:";_mode$
1050    PRINT @Addr:"WFMPRE?"
1060    INPUT @Addr:Wfpre$
1070    Temp=POS(Wfpre$,"NR.PT",1)
1080    Temp$=SEG(Wfpre$,Temp,10)
1090    Wfpts=VAL(Temp$)
1100    DIM Wfdata(Wfpts)
1110    PRINT @Addr:"CURVE?"
1120    IF _mode$="BIN" THEN
1130       CALL "BININ","UNPK",Wfdata,Error;Addr
1140       IF Error THEN
1150          PRINT "GETWFM - Error in Binary Block transfer: ";Error
1160       END IF
1170    ELSE
1180       INPUT @Addr:Wfdata
1190    END IF
1200 END SUB


*+ ************************************************************** +
* TEKTRONIX INSTRUMENTATION SOFTWARE LIBRARY UTILITY ROUTINES
* PUTWFM - Send Binary or ASCII Waveform Data to 7D20.
* February 1,1983
*
* Copyright [c] 1983, Tektronix, Inc. All rights reserved.
* REQUIRED EQUIPMENT:
*    1.  R14A GPIB Enhancement ROM Pack.
* PURPOSE:
*    Sends waveform data to targeted 7D20 memory as per values
*    passed in call to this routine.
*    Transfer may be either in Binary Block mode or ASCII.  Mode
*    is set by contents of _MODE$ in call to routine, but values
*    in WFDATA must be correct for the mode specified (values
*    from 0 to 255 for Binary Block and -5.12 to +5.08 for ASCII).
* CALLING FORMAT:
*    CALL Putwfm(Addr,_memnum,_mode$,Wfpre$,Wfdata)
* INPUTS:
*    Addr ----- Address (1 - 30) of targeted 7D20.
*    _memnum -- Waveform Memory number into which the data is
*               to be placed.
*    _mode$ --- "ASC" or "BIN", depending on type of transfer
*               to be performed.
*    Wfpre$ --- Preamble of the waveform to be transferred.
*    Wfdata --- Array of waveform data to be sent.  The values
*               should be 0 - 255 for Binary Block mode transfer
*               or -5.12 - +5.08 for ASCII mode.
* OUTPUTS:
*    1.  7D20 receives data into the desired waveform memory.
* VARIABLES USED: (All Local)
*    Error ---- Error codes set by R14A BINOUT routine.
*    Wfpts ---- Local variable containing the diminsioned size
*               of Wfdata for ASCII transfer.
*    Element -- Loop counter in ASCII data transfer routine.
*               One at a time, each element of Wfdata is put
*               into a string, a comma is appended to it, then
*               it is sent to the 7D20. The exception is the
*               last element of Wfdata which is sent without
*               a comma appended to it.
*    Wfdata$ -- String variable used in ASCII data transfer
*               loop to receive data pointed to by Element.
* ROUTINES CALLED:
*    None.
* POSSIBLE ERRORS:
*    1.  Transmit errors reported by BINOUT.
*    2.  Wfdata values are wrong for the desired transfer mode.
*- ************************************************************** -
```

```
1000 SUB Putwfm(Addr,_memnum,_mode$,Wfpre$,Wfdata)
1010    LOCAL Error,Wfpts,Element
1020    PRINT @Addr:"DA MEM:";_memnum;",ENCDG:";_mode$
1030    PRINT @Addr:Wfpre$
1040    CALL "LISTEN";Addr
1050    CALL "PRISTR","CURVE "
1060    IF _mode$="BIN" THEN
1070       CALL "BINOUT","UNPK",Wfdata,Error
1080       IF Error THEN
1090          PRINT "Error in Binary Block transfer - ";
1100          IF Error=10 THEN
1110             PRINT "Data in Source Array out of range."
1120          ELSE
1130             PRINT "Unknown error code returned. (";Error;")."
1140          END IF
1150       END IF
1160    ELSE
1170       Wfpts=UBOUND(Wfdata,1)
1180       FOR Element=1 TO Wfpts-1
1190          Wfdata$=STR(Wfdata(Element))&","
1200          CALL "PRISTR",Wfdata$
1210       NEXT Element
1220       Wfdata$=STR(Wfdata(Element))
1230       CALL "PRISTR",Wfdata$
1240       WBYTE -59
1250    END IF
1260    CALL "UNL"
1270 END SUB
```

```
*+ ************************************************************ +
* TEKTRONIX INSTRUMENTATION SOFTWARE LIBRARY UTILITY ROUTINE
* NORMWFM - Normalize waveform
* February 1, 1983
*
* Copyright [c] 1983, Tektronix, Inc. All rights reserved.
*
* REQUIRED EQUIPMENT:
*    None
* PURPOSE:
*    This routine converts waveform data (Wfdata) to normalized
*    data (_normdata) expressed in volts and adjusted for the
*    zero reference value from Wfpre$.  Wfdata is not actually
*    altered itself.
* CALLING FORMAT:
*    CALL _normwfm(Wfdata,Wfpre$;_normdata)
* INPUTS:
*    Wfdata ---- Waveform data array in either Binary or ASCII
*                aquision form.  It is intended that this array
*                is the output from the Getwfm routine.
*    Wfpre$ ---- Waveform preamble associated with Wfdata. This
*                is also intended to be the output from Getwfm.
* OUTPUTS:
*    _normdata ---- Normalized waveform data.  It is the result
*                   of calculating Wfdata against the vertical
*                   scale factor and zero offset extracted from
*                   Wfpre$.
* VARIABLES USED: (All Local)
*    Wfpts --------- Contains the size of Wfdata.
*     Mode$ -------- Contains the encoding mode from Wfpre$.
*    Yzero --------- Contains the zero reference from Wfpre$.
*    Ymult --------- Contains the Volts/Div from Wfpre$.
*    Temp$ --------- String variable for parsing Wfpre$.
* ROUTINES CALLED:
*   None
* POSSIBLE ERRORS:
*    1.  Not enough free memory to define _NORMDATA.
*- ************************************************************ -
```

```
1000 SUB _normwfm(Wfdata,Wfpre$;_normdata)
1010    LOCAL Temp$,Wfpts,Yzero,_mode$,Ymult
1020    DIM Temp$(LEN(Wfpre$))
1030    Temp$=SEG(Wfpre$,POS(Wfpre$,"NR",1),LEN(Wfpre$))
1040    Wfpts=VAL(Temp$)
1050    Temp$=SEG(Wfpre$,POS(Wfpre$,"YZ",1),LEN(Wfpre$))
1060    Yzero=VAL(Temp$)
1070    _mode$=SEG(Wfpre$,POS(Wfpre$,":",POS(Wfpre$,"ENC",1))+1,3)
1080    Temp$=SEG(Wfpre$,POS(Wfpre$,"YM",1),LEN(Wfpre$))
1090    Ymult=VAL(Temp$)
1100    DELETE _normdata
1110    DIM _normdata(Wfpts)
1120    _normdata=Wfdata
1130    IF _mode$="ASC" THEN 1160
1140    _normdata=_normdata-128
1150    _normdata=_normdata/25
1160    _normdata=_normdata*Ymult
1170    _normdata=_normdata+Yzero
1180 END SUB
```

ASCII & GPIB CODE CHART

B7 B6 B5	Ø Ø Ø	Ø Ø 1	Ø 1 Ø	Ø 1 1	1 Ø Ø	1 Ø 1	1 1 Ø	1 1 1
BITS	CONTROL		NUMBERS SYMBOLS		UPPER CASE		LOWER CASE	
B4 B3 B2 B1								
Ø Ø Ø Ø	0 **NUL** 0 0	20 **DLE** 10 16	40 0 **SP** 20 32	60 16 **0** 30 48	100 0 **@** 40 64	120 16 **P** 50 80	140 0 **`** 60 96	160 16 **p** 70 112
Ø Ø Ø 1	1 GTL **SOH** 1 1	21 LLO **DC1** 11 17	41 1 **!** 21 33	61 17 **1** 31 49	101 1 **A** 41 65	121 17 **Q** 51 81	141 1 **a** 61 97	161 17 **q** 71 113
Ø Ø 1 Ø	2 **STX** 2 2	22 **DC2** 12 18	42 2 **"** 22 34	62 18 **2** 32 50	102 2 **B** 42 66	122 18 **R** 52 82	142 2 **b** 62 98	162 18 **r** 72 114
Ø Ø 1 1	3 **ETX** 3 3	23 **DC3** 13 19	43 3 **#** 23 35	63 19 **3** 33 51	103 3 **C** 43 67	123 19 **S** 53 83	143 3 **c** 63 99	163 19 **s** 73 115
Ø 1 Ø Ø	4 SDC **EOT** 4 4	24 DCL **DC4** 14 20	44 4 **$** 24 36	64 20 **4** 34 52	104 4 **D** 44 68	124 20 **T** 54 84	144 4 **d** 64 100	164 20 **t** 74 116
Ø 1 Ø 1	5 PPC **ENQ** 5 5	25 PPU **NAK** 15 21	45 5 **%** 25 37	65 21 **5** 35 53	105 5 **E** 45 69	125 21 **U** 55 85	145 5 **e** 65 101	165 21 **u** 75 117
Ø 1 1 Ø	6 **ACK** 6 6	26 **SYN** 16 22	46 6 **&** 26 38	66 22 **6** 36 54	106 6 **F** 46 70	126 22 **V** 56 86	146 6 **f** 66 102	166 22 **v** 76 118
Ø 1 1 1	7 **BEL** 7 7	27 **ETB** 17 23	47 7 **'** 27 39	67 23 **7** 37 55	107 7 **G** 47 71	127 23 **W** 57 87	147 7 **g** 67 103	167 23 **w** 77 119
1 Ø Ø Ø	10 GET **BS** 8 8	30 SPE **CAN** 18 24	50 8 **(** 28 40	70 24 **8** 38 56	110 8 **H** 48 72	130 24 **X** 58 88	150 8 **h** 68 104	170 24 **x** 78 120
1 Ø Ø 1	11 TCT **HT** 9 9	31 SPD **EM** 19 25	51 9 **)** 29 41	71 25 **9** 39 57	111 9 **I** 49 73	131 25 **Y** 59 89	151 9 **i** 69 105	171 25 **y** 79 121
1 Ø 1 Ø	12 **LF** A 10	32 **SUB** 1A 26	52 10 ***** 2A 42	72 26 **:** 3A 58	112 10 **J** 4A 74	132 26 **Z** 5A 90	152 10 **j** 6A 106	172 26 **z** 7A 122
1 Ø 1 1	13 **VT** B 11	33 **ESC** 1B 27	53 11 **+** 2B 43	73 27 **;** 3B 59	113 11 **K** 4B 75	133 27 **[** 5B 91	153 11 **k** 6B 107	173 27 **{** 7B 123
1 1 Ø Ø	14 **FF** C 12	34 **FS** 1C 28	54 12 **,** 2C 44	74 28 **<** 3C 60	114 12 **L** 4C 76	134 28 \ 5C 92	154 12 **l** 6C 108	174 28 **|*** 7C 124
1 1 Ø 1	15 **CR** D 13	35 **GS** 1D 29	55 13 **−** 2D 45	75 29 **=** 3D 61	115 13 **M** 4D 77	135 29 **]** 5D 93	155 13 **m** 6D 109	175 29 **}** 7D 125
1 1 1 Ø	16 **SO** E 14	36 **RS** 1E 30	56 14 **.** 2E 46	76 30 **>** 3E 62	116 14 **N** 4E 78	136 30 **∧** 5E 94	156 14 **n** 6E 110	176 30 **~** 7E 126
1 1 1 1	17 **SI** F 15	37 **US** 1F 31	57 15 **/** 2F 47	77 UNL **?** 3F 63	117 15 **O** 4F 79	137 UNT **_** 5F 95	157 15 **o** 6F 111	177 **DEL (RUBOUT)** 7F 127
	ADDRESSED COMMANDS	UNIVERSAL COMMANDS	LISTEN ADDRESSES		TALK ADDRESSES		SECONDARY ADDRESSES OR COMMANDS (PPE)	(PPD)

*| on some keyboards or systems

KEY

octal	25 PPU	GPIB code
	NAK	ASCII character
hex	15 21	decimal

Tektronix®
COMMITTED TO EXCELLENCE

REF: ANSI STD X3. 4-1977
IEEE STD 488-1978
ISO STD 646-1973

TEKTRONIX INSTRUMENTATION SOFTWARE LIBRARY

Utility Software for 7D20

Utility Software is available from Tektronix, Inc. for the 7D20 Programmable Digitizer. This software consists of a set of subroutines and subprograms that perform common instrument functions over the GPIB such as data acquisition, front-panel set-up, etc. These routines are designed to be easily integrated into your application programs. And since they are small and well documented, the routines are easy to modify to suit your particular applications. Refer to the current Tektronix Instrumentation Software Library Catalog for instrument options, ROM packs, and other required equipment.

The following Utility Software was available when this Instrument Interfacing Guide was printed. Other software may be available; contact your local Tektronix Field Office for further information.

Description	Tektronix Part No.
7D20/4041 Utility Software (DC-100 tape)	062-6959-01
7D20/4052A Utility Software (DC-300 tape)	062-6961-01

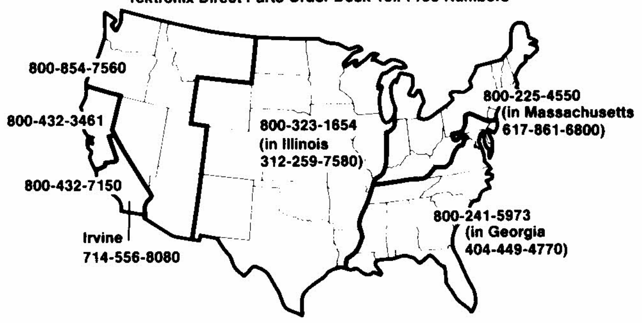
Ordering Utility Software (U.S. Only)

Your local Tektronix Field Office has the current prices for software available from the Tektronix Instrumentation Software Library.

Order Tektronix Instrumentation Software Library programs from Tektronix Central Parts Ordering by using the toll-free number serving your area. The following map identifies the geographical regions in the U.S. and the toll-free number serving each region.

Call the toll-free number serving your area and give the Customer Service Representative the Tektronix nine-digit part number and name of the software package you want to order. If you have any questions about the software, call your local Tektronix Field Office.

Tektronix Direct Parts Order Desk Toll-Free Numbers



Ordering Utility Software (Outside the U.S.)

Outside of the U.S., order Tektronix Instrumentation Software Library programs through your local Tektronix sales office or from the Tektronix Instrumentation Software Library order point serving your area. Refer to the following list for the applicable library order point.

Africa, Europe, Middle East

Contact local Tektronix sales office.

Australia

Tektronix Instrumentation Software Library
Tektronix Australia Pty. Limited
Sydney
80 Waterloo Road
North Ryde, N.S.W. 2113

Canada

Tektronix Instrumentation Software Library
Tektronix Canada Ltd.
P.O. Box 6500
Barrie, Ontario
Canada L4M 4V3

Caribbean, Latin America, and Far East (except Japan)

Tektronix Instrumentation Software Library
Export Marketing
Tektronix, Inc.
P.O. Box 500
Beaverton, OR 97077
U.S.A.

Japan

Tektronix Instrumentation Software Library
Sony/Tektronix Corporation
9-31 Kitashinagawa-5
Tokyo 141 Japan

Program Library

The Tektronix Instrumentation Software Library includes over 200 software programs for a variety of Tektronix programmable instruments and controllers. The Library Catalog provides abstracts of the available software. Programs are available as ready-to-load media or as listings (see Catalog). For a copy of the latest catalog, contact your local Tektronix Field Office or representative and ask for Tektronix Instrumentation Software Library Catalog #99W-5293.

Program Contributions

If you have a program which you would like to submit to the Tektronix Instrumentation Software Library, we will send you, in exchange, one software package of your choice from the Customer/User Software portion of the Program Library (see current library catalog). Submitted programs must use Tektronix programmable instruments and must meet certain coding and documentation standards.

To contribute a program, submit a copy of the program on media along with a listing and a Tektronix Instrument Software Library release form (see current library catalog). If the program was created as part of your employment, the release must be signed by an authorized representative of your employer. Acceptance of the program is subject to review of the Tektronix Instrumentation Software Library staff.

For further information on submitting a program or for information about coding and documentation standards, contact:

Tektronix Instrumentation Software Library
Tektronix, Inc.
Group 157, 54-016
P.O. Box 500
Beaverton, OR 97077

Software Warranty

Tektronix warrants that the media (tapes, disks, ROMs, etc.) on which Software Products are furnished and the encoding of the programs on the media will be free from defects in materials and workmanship for a period of three (3) months from the date of shipment. If any such medium or encoding proves defective during the warranty period, Tektronix will provide a replacement in exchange for the defective medium. Except as to the media on which Software Products are furnished, Software Products are provided "as is" without warranty of any kind, either expressed or implied. Tektronix does not warrant that the functions contained in the programs will meet Customer's requirements or that the operations of the programs will be uninterrupted or error-free.

In order to obtain service under this warranty, Customer must notify Tektronix of the defect before the expiration of the warranty period. If Tektronix is unable to provide a replacement that is free from defects in materials and workmanship within a reasonable time thereafter, Customer may terminate the license for the Software Product and return the Software Product with the associated materials for credit or refund.

Tektronix disclaims any implied warranties of merchantability or fitness for a particular purpose. Tektronix' responsibility to replace defective media or refund customer's payment is the sole and exclusive remedy provided to the customer for breach of this warranty. Tektronix will not be liable for any indirect, special, incidental, or consequential damages irrespective of whether Tektronix has advance notice of the possibility of such damages.

Tek. Lit. No. 45W5280