Tektronix®
COMMITTED TO EXCELLENCE

TEK SPS BASIC
V02/V02XM
7912AD Commands Package

CP57006/CP57506

INSTRUCTION MANUAL

Tektronix, Inc.
P.O. Box 500
Beaverton, Oregon 97077

Serial Number _____

First Printing MAY 1979 PRODUCED BY SPS DOCUMENTATION GROUP

Scans
By
Artek Media

Artek Media
1042 Plummer Cir. SW
Rochester, MN 55902

www.artekmedia.com

"High resolution scans of obsolete technical manuals"

If your looking for a quality scanned technical manual in PDF format please visit our WEB site at www.artekmedia.com or drop us an email at manuals@artekmedia.com and we will be happy to email you a current list of the manuals we have available.

If you don't see the manual you need on the list drop us a line anyway we may still be able to point you to other sources. If you have an existing manual you would like scanned please write for details, This can often be done very reasonably in consideration for adding your manual to our library.

Typically the scans in our manuals are done as follows;

1) Typed text pages are typically scanned in black and white at 300 dpi.
2) Photo pages are typically scanned in gray scale mode at 600 dpi
3) Schematic diagram pages are typically scanned in black and white at 600 dpi unless the original manual had colored high lighting (as is the case for some 70's vintage Tektronix manuals).

If you purchased this manual from us (typically through our Ebay name of ArtekMedia) thank you very much. If you received this from a well-meaning "friend" for free we would appreciate your treating this much like you would "share ware". By that we mean a donation of at least $5-10 per manual is appreciated in recognition of the time (a manual can take as much as 40 hours to reproduce, book, link etc.), energy and quality of effort that went into preserving this manual. Donations via PayPal go to: manuals@artekmedia.com or can be mailed to us the address above.

Thanks

Dave & Lynn Henderson
Artek Media

SOFTWARE SUPPORT POLICY

Unless otherwise provided, Tektronix, Inc., agrees that during the one (1) year period following installation, if the customer encounters a problem with this software which the customer's diagnosis indicates is caused by a software defect, the customer may submit a Software Performance Report to Tektronix, Inc. For problems occurring in current, unaltered releases of software, Tektronix, Inc., will respond to Software Performance Reports via a software maintenance periodical. The software maintenance periodical will be provided at no cost to the customer for one year following installation and will contain information for correcting or bypassing verified problems where possible, and will give notice of availability of corrected software.

Any software updates released by Tektronix, Inc., to correct problems during the one (1) year period will be provided to the customer at no charge on the standard distribution media specified in the software documentation. If media other than the standard distribution media is requested, the customer will only be charged for the current cost of the optional media.

SOFTWARE LICENSE

This software product, including subsequent improvements or updates, is furnished under a license for use on a single controller. It may only be copied, in whole or in part (with the proper inclusion of the Tektronix, Inc., copyright notice on the software), for use on that specific controller.

Specification and price change privileges are reserved.

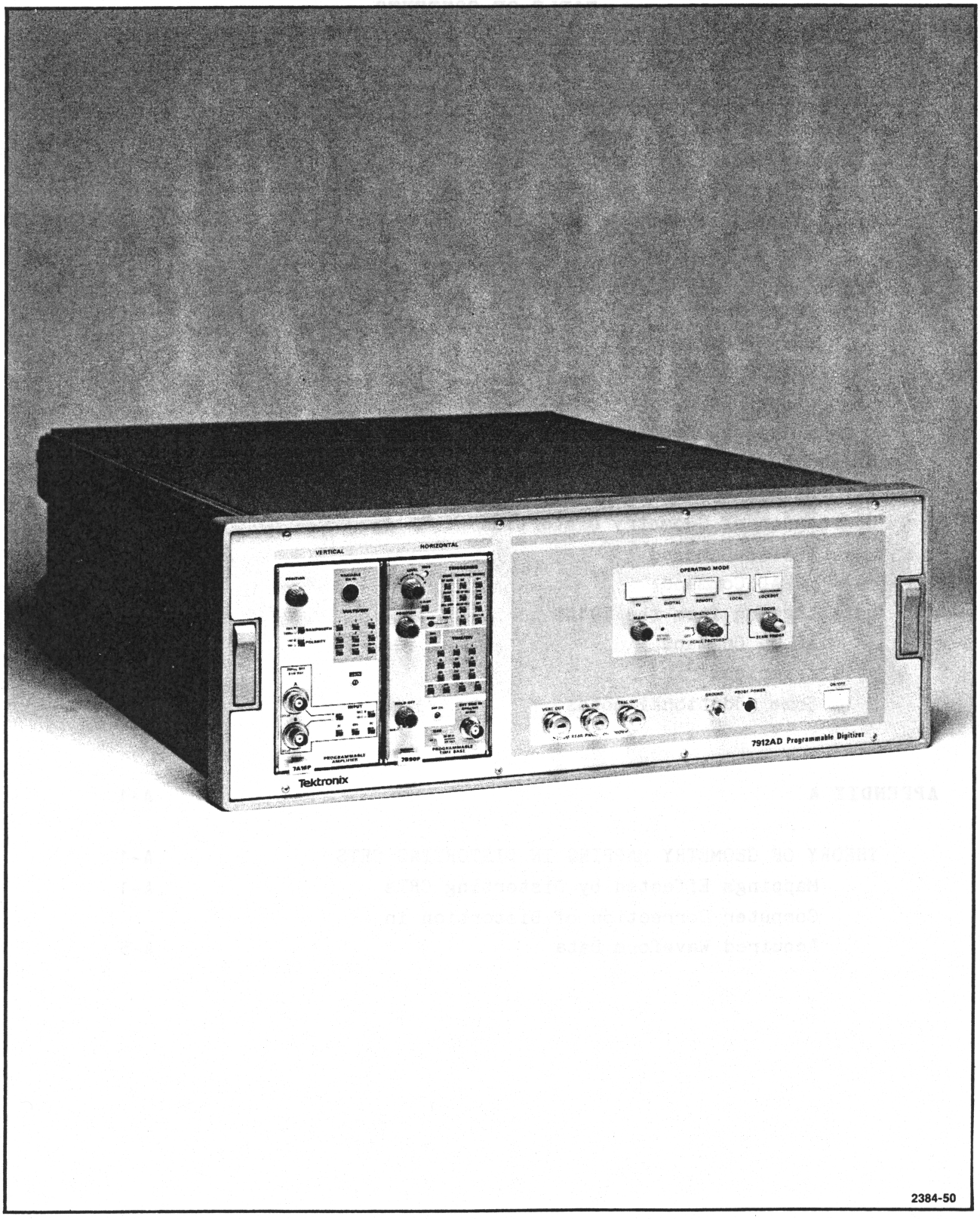
Although the material in this manual has been thoroughly edited and checked for accuracy, Tektronix, Inc., makes no guarantees against typographical or human errors. Also, Tektronix, Inc., assumes no responsibility or liability, consequential or otherwise, of any kind arising from misinterpretation or misuse of the material in this manual. The contents of this manual are subject to change without notice.

PREFACE

This manual describes the 7912AD Nonresident Commands Package for TEK SPS BASIC VØ2 and VØ2XM software. Any exception to an option or a capability of a command in this package being supported by a specific release of the software is noted where appropriate.

The prerequisite software for executing the commands in this package is the corresponding version of the TEK SPS BASIC System Software. The VØ2 package (CP57006) requires the VØ2 System Software (CP57000); the VØ2XM package (CP57506) requires the VØ2XM System Software (CP57500).

TABLE OF CONTENTS



The TEKTRONIX 7912AD Programmable Digitizer with programmable plug-ins.

7912AD NONRESIDENT COMMANDS

Introduction

This manual describes the TEK SPS BASIC VØ2 or VØ2XM commands that support the TEKTRONIX 7912AD Programmable Digitizer. This software package does not contain an instrument specific driver. Instead, the data from the 7912AD is read into the controller via a TEKTRONIX CP4100/IEEE Interface or a CP1100/IEEE 488 Interface. The driver used in this manual is the IEEE 488 Interface driver (GPI.SPS), which is part of the TEK SPS BASIC VØ2 and VØ2XM System Software. This interface driver and its commands are discussed in Section 6 of the System Software manual. Another driver for the interface, the high-level IEEE 488 Instrument driver (INS.SPS), is available in an additional package. This second driver and its accompanying commands offer easier control of IEEE 488 instruments that conform to the Tektronix codes and formats, such as the 7912AD.

While the IEEE 488 Interface driver (GPI.SPS) is very useful for communicating with the 7912AD, it leaves the acquired data in a relatively primitive form. Some enhancement capability is built into the 7912AD itself, including removal of target defects from data and edge processing. Further enhancement is provided in some special nonresident commands for TEK SPS BASIC. These commands allow: convenient graphing of raw 7912AD data without preprocessing (ADPLOT), reduction of raw data to edge data (EDGEAD, algorithm the same as used in the 7912AD), computation of a zero-reference value (ZREF), rejection of target defects (REJECT, same algorithm as in the 7912AD), geometry correction for CRT-induced distortion (INSTAD and MAPAD) and finally, full reduction of the acquired waveform to a calibrated array consisting of one floating-point vertical value for every horizontal address (NORMAD).

An additional command (ADLOG) allows you to digitize and acquire a series of waveforms at high speed. The waveforms are stored in a disk file for later processing.

Each command (except ADLOG), whether performed in the 7912AD or in the controller, is a step in the process of reducing 7912AD waveform data to a more useful form. The end result is the normalized array. Each controller command is discussed in detail in this manual. Instrument commands that perform the same function as a related controller command (e.g. EDGEAD and READ EDGE) are discussed briefly. More detailed information on specific instrument commands and command formats can be found in Section 3 of the 7912AD Operators manual.

Loading Nonresident Commands

Since each 7912AD command is a separate nonresident module, each must be loaded individually. This can be done in two ways. The first method is to load the desired command with the LOAD command. For example, the following command would load the ADPLOT command from the system device:

```
LOAD "ADPLOT"
```

Similarly,

```
LOAD DX: "EDGEAD", "NORMAD"
```

would load the EDGEAD and NORMAD commands from the left-hand drive of the CP112 or CP115 Floppy Disk drive. Commands can be loaded automatically, as well. To autoload a command, simply enter the command name (and accompanying arguments) on the terminal or reference it in a program line. The command so referenced will be autoloaded from the system device.

Commands that have been deliberately LOADed remain in memory until they are deliberately RELEASEd. However, autoloaded commands are automatically released from memory as required to make room for new commands to be loaded. In cases where processing speed is important, the 7912AD commands should be LOADed before running a program. This minimizes the time required for program execution since the required commands are already in memory. In cases where conservation of controller memory is important, it is probably best to autoload the commands as they are needed and allow them to be released when they are no longer required.

Graphing 7912AD Data in Standard Format

ADPLOT Command. 7912AD data acquired in standard format (verticals and pointer array) may be plotted on the terminal screen with the ADPLOT command. Such a plot allows a quick preview of 7912AD data before processing by other commands. From such a plot you can check for such things as correct beam intensity and the presence of target defects.

The format of the ADPLOT command is:

Syntax Form:

[line no.] **ADPLOT** integer array, { integer array / integer waveform }

[,expression]

Descriptive Form:

[line no.] **ADPLOT** vertical value data, pointer table
[,vertical scale factor]

The first argument following the command name specifies the vertical-data array. It contains the upper and lower vertical data acquired in standard format. Only positive values (including zeroes) are displayed.

The second argument specifies the pointer array or waveform. Any portion of the pointer array may be specified by zoning, but care must be taken to use the corresponding portion of the vertical-data array. If the second argument is a waveform, the associated array must be integer. Vertical and horizontal units are displayed if the strings are present. Also, the data sampling interval is applied to the horizontal axis of the graph if present (otherwise the data sampling interval is set to a default value of 1).

The optional third argument contains the vertical scale factor in units per division. Data from the vertical-data array is multiplied by this vertical scale factor and divided by 64 (the number of points per vertical division) to achieve vertical scaling. If the argument is not present, a default scale factor of 64 is applied.

Like the GRAPH command, ADPLOT responds to all non-default values set by the SETGR command with one exception: the number of major tic intervals defaults to 1Ø on the x-axis and 8 on the y-axis unless overridden by the SETGR command.

As an example of a routine which acquires and plots a raw-data trace, consider the following program. This and all other example programs in this manual assume the IEEE 488 Interface driver ("GPI.SPS") has been LOADed, the 7912AD commands are on the system disk to allow them

to be auto-loaded, the interface number is Ø and the 7912AD is in a powered-up condition with a primary address of Ø and a secondary address of Ø. This results in a bus listen address of 32, a bus talk address of 64, and a bus secondary address of 96 for the 7912AD. Note that 32, 64, or 96 must be added to the primary bus address to get listen, talk, and secondary addresses, respectively. (The addition is shown for clarity in program lines 5Ø, 6Ø, and 7Ø.)

Some of the example programs use the PAGE and GRAPH commands. If you do not have the TEK SPS BASIC VØ2 Graphics Package, the programs should be modified to remove these graphics commands.

```
1Ø REM
2Ø REM   PROGRAM TO ACQUIRE AND DISPLAY 7912AD RAW DATA
3Ø REM
4Ø SIFTO @Ø,3ØØØ
5Ø LA=Ø+32
6Ø TA=Ø+64
7Ø SA=Ø+96
8Ø PUT "MODE TV" INTO @Ø,LA,SA
9Ø PRINT "SET UP 7912AD TO ACQUIRE DATA"
1ØØ PRINT "PRESS ANY KEY WHEN READY"
11Ø SIFCOM @Ø,LA,SA,"GTL"
12Ø WAIT
13Ø PUT "DIG DAT" INTO @Ø,LA,SA
14Ø PUT "READ PTR,VER" INTO @Ø,LA,SA
15Ø GOSUB 2ØØØ
16Ø INTEGER P(511)
17Ø P=QQ
18Ø GOSUB 2ØØØ
19Ø PUT "MODE TV" INTO @Ø,LA,SA
2ØØ PAGE
21Ø ADPLOT QQ,P
22Ø STOP
2ØØØ REM
2Ø1Ø REM   SUBROUTINE TO READ DATA ARRAY
2Ø2Ø REM
2Ø3Ø IFDTM @Ø,"UNP"
2Ø4Ø GET X FROM @Ø,TA,SA
2Ø5Ø IF CHR(X)<>"%" THEN STOP
2Ø6Ø IFDTM @Ø,"PAK","HBF"
2Ø7Ø GET CW FROM @Ø,TA,SA
```

```
208Ø CW=(CW-1)/2-1
209Ø DELETE QQ
21ØØ INTEGER QQ(CW)
211Ø GET QQ FROM @Ø,TA,SA
212Ø IFDTM @Ø,"UNP"
213Ø GET X FROM @Ø,TA,SA
214Ø GET X FROM @Ø,TA,SA
215Ø IF CHR(X)<>";" THEN STOP
216Ø RETURN
```

The REM command identifies the program. At line 4Ø, the interface time-out value is set to 3ØØØ milliseconds. This allows the 7912AD enough time to switch from TV to Digital mode. Listen, Talk, and Secondary addresses are then assigned to variables in lines 5Ø through 7Ø. The 7912AD is put into TV mode at line 8Ø. Then a message is displayed requesting the user set up a waveform to digitize. The instrument is instructed to go to local mode to facilitate the set-up. When any terminal key is pressed, the WAIT command is exited. The instrument is instructed to digitize data and to prepare for reading of pointer and vertical data arrays. At line 15Ø, a subroutine is invoked to actually acquire the array data from the instrument.

The acquisition subroutine starts at line 2ØØØ with remarks that identify the subroutine. The IFDTM command instructs the interface to transmit data "unpacked", or one byte per word. One character is fetched using the GET command. Since the first character of every 7912AD data block must be a "%", the subroutine stops if this character is not correct. The interface is then told to transmit data 'packed high byte first', or two bytes per word at line 2Ø6Ø. The byte count of the data to be received is fetched and is converted to an integer array dimension. The destination array, QQ, is deleted in case its new size is not the same as its old size. QQ is redimensioned to the new size and is then filled by the GET command at line 211Ø. The interface is told to transmit data in "unpacked" mode again at line 212Ø. A checksum byte is fetched and ignored at line 213Ø, and a data terminator character is fetched. Since 7912AD array data is always terminated by a ";", the subroutine stops if the character is not correct. The subroutine returns to the calling program with the execution of the RETURN command at line 216Ø.

This subroutine is used in the examples throughout the rest of this manual. The line numbers may change from program to program, however.

The main program continues at line 16Ø, dimensioning a pointer data array, P, with 512 integer elements. The pointer array is set equal to the array QQ as returned from the subroutine. The subroutine is invoked again to read the vertical data array. The data is left in array QQ. The instrument is returned to 'TV mode' at line 19Ø. The terminal screen is paged and the data is plotted. The results are shown in Fig. 1. In this case, the vertical axis is labeled in data levels and the horizontal axis is labeled in array element numbers.

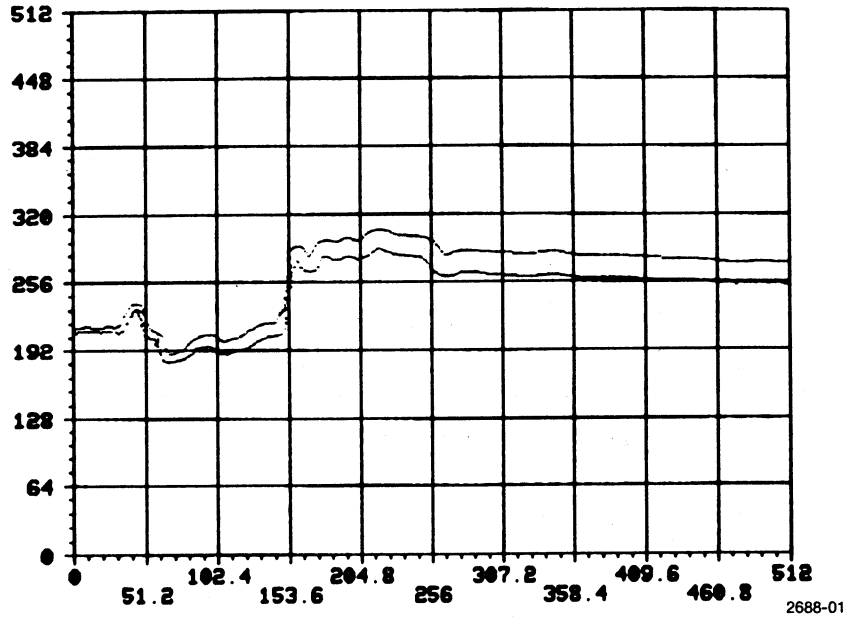


Fig. 1. Graph of raw 7912AD data.

A little additional code acquires scale-factor information from the plug-in units and stores it in string variables. The example program shown below is similar to the previous one, with the added or modified code shown in bold print. This convention is used in subsequent examples, as well. Each program to be discussed is simply an extension of the last, with the added features highlighted.

```
1Ø REM
2Ø REM   PROGRAM TO ACQUIRE AND DISPLAY 7912AD RAW DATA
3Ø REM      WITH SCALE FACTOR (WAVEFORM) INFORMATION
4Ø REM
5Ø SIFTO @Ø,3ØØØ
6Ø LA=Ø+32
```

```
70 TA=0+64
80 SA=0+96
90 PUT "MODE TV" INTO @0,LA,SA
100 PRINT "SET UP 7912AD TO ACQUIRE DATA"
110 PRINT "PRESS ANY KEY WHEN READY"
120 SIFCOM @0,LA,SA,"GTL"
130 WAIT
140 PUT "DIG DAT" INTO @0,LA,SA
150 PUT "READ PTR,VER" INTO @0,LA,SA
160 GOSUB 2000
170 INTEGER P(511)
180 WAVEFORM WP IS P,SP,HP$,VP$
190 P=QQ
200 GOSUB 2000
210 PUT "MODE TV" INTO @0,LA,SA
220 PUT "HS1?" INTO @0,LA,SA
230 GET A$ FROM @0,TA,SA
240 SP=VAL(SEG(A$,5,LEN(A$)-1))
250 SP=SP/51.2
260 PUT "VS1?" INTO @0,LA,SA
270 GET A$ FROM @0,TA,SA
280 VS=VAL(SEG(A$,5,LEN(A$)-1))
290 PUT "HU1?" INTO @0,LA,SA
300 GET A$ FROM @0,TA,SA
310 HP$=SEG(A$,5,LEN(A$)-1)
320 PUT "VU1?" INTO @0,LA,SA
330 GET A$ FROM @0,TA,SA
340 VP$=SEG(A$,5,LEN(A$)-1)
350 PAGE
360 ADPLOT QQ,WP,VS
370 STOP
2000 REM
2010 REM   SUBROUTINE TO READ DATA ARRAY
2020 REM
2030 IFDTM @0,"UNP"
2040 GET X FROM @0,TA,SA
2050 IF CHR(X)<>"%" THEN STOP
2060 IFDTM @0,"PAK","HBF"
2070 GET CW FROM @0,TA,SA
2080 CW=(CW-1)/2-1
2090 DELETE QQ
2100 INTEGER QQ(CW)
```

```
211Ø GET QQ FROM @Ø,TA,SA
212Ø IFDTM @Ø,"UNP"
213Ø GET X FROM @Ø,TA,SA
214Ø GET X FROM @Ø,TA,SA
215Ø IF CHR(X)<>";" THEN STOP
216Ø RETURN
```

The boldfaced portions of this example are additions to our previous program. They allow acquisition of scale factor and units information which is used to define a waveform. This example has been renumbered, so the line numbers won't correspond exactly with the first example.

The WAVEFORM command at line 18Ø associates the pointer array, P, a data sampling interval variable, SP, and the horizontal and vertical units string variables HP$ and VP$, to a waveform variable, WP.

At line 2ØØ, the 7912AD is asked for the horizontal scale factor for channel 1. This value is converted from units per horizontal division (1Ø divisions full screen) to units per horizontal point (512 points full screen or 51.2 points per division) at line 25Ø. The vertical scale factor, in units per vertical division, is fetched and saved in variable VS. Strings associated with the horizontal and vertical units are fetched and assigned to the waveform string variables.

The waveform data is displayed by the ADPLOT command at line 36Ø. In this case, the vertical axis is labeled in data levels that have been adjusted by the vertical scale factor in variable VS. The horizontal axis is labeled in data sampling units. The results are shown in Fig. 2.

Producing a Normalized Trace

After the 7912AD data has been acquired in standard format (vertical vs. pointers arrays), two or three more steps are necessary for production of a normalized trace. (In this sense, "normalized" means one calibrated, floating-point vertical value for each horizontal address.)

The first step is performed either by the controller with the EDGE command or by the 7912AD itself with the READ EDGE command; it generates a set of data in which there are typically two integer values, representing the top and bottom of the trace, for each horizontal location. Generally, the EDGEAD (or READ EDGE) routine is performed twice; once for the waveform trace and once for the zero-reference trace.

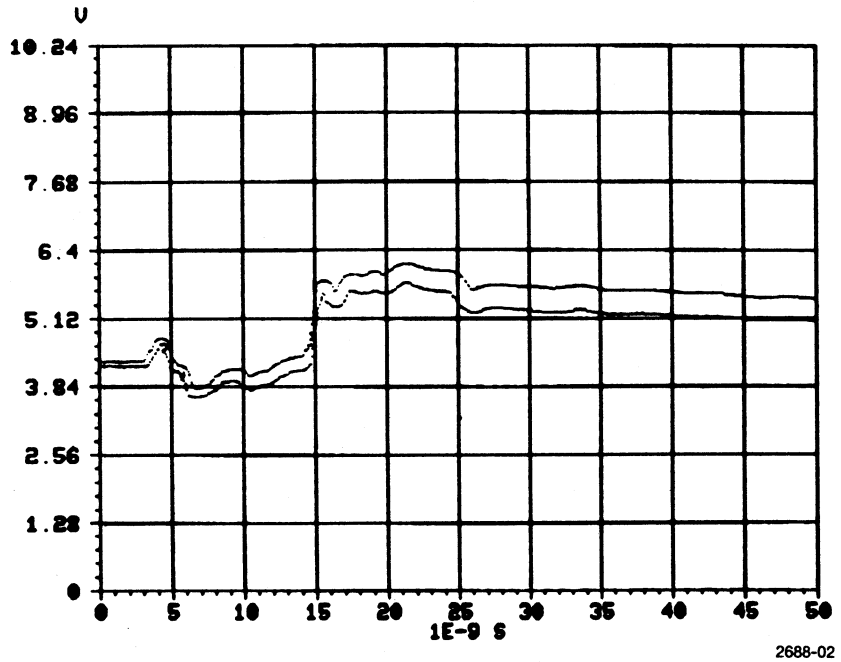


Fig. 2. Raw 7912AD data modified by scale factors.

The second step is performed by the ZREF command. It computes an average zero-reference value based on the edge data acquired from a zero-reference trace.

The third step is performed by the NORMAD command. It uses the waveform edge data from the waveform trace to produce a normalized array containing exactly one floating-point vertical for each horizontal. After execution of the NORMAD command, the data can be operated upon by any of the standard TEK SPS BASIC signal processing commands.

Edge Data

The 7912AD produces waveform information in two formats: raw data consisting of vertical data and a pointer array, and edge data. Edge data results from processing the raw vertical and pointer arrays to produce an ordered set of waveform data with two integer values representing the top and bottom edges of the trace for each horizontal location.

The actual processing can occur within the 7912AD itself (the READ EDGE command) or within the controller after raw data is acquired (the EDGEAD command). The algorithms used are identical; the only difference in the two operations is where they are performed. The 7912AD requires about 15 milliseconds to process raw data and generate edge data. Some

speed-sensitive applications may require very rapid acquisition of a series of waveforms. In these cases, the raw waveform data may be repetitively acquired and read from the 7912AD (with the REPEAT command) without any time loss for processing. Later, the EDGEAD command in the controller can be used to process the acquired data into edge arrays.

Letting the 7912AD process its own data with the READ EDGE command is the simplest to implement, and often preferable. The 7912AD REPEAT command does not produce edge data, however. If REPEAT mode operation is required, the EDGEAD command must be used. More information about REPEAT mode, the READ EDGE command, and the general format of 7912AD commands can be found in Section 3 of the 7912AD Operator's Manual. Only the EDGEAD command is discussed in detail here.

EDGEAD Command. The EDGEAD command uses the vertical array and pointer array as its inputs. It outputs two arrays of data representing the upper and lower edges of the 7912AD trace.

The format of EDGEAD is:

Syntax Form:

```
[line no.] EDGEAD integer array, {integer array   }
                                 {integer waveform},

           {simple numeric variable} {simple numeric variable}
           {integer array           },{integer array          }
           {integer waveform        } {integer waveform       }

           [,expression,expression]
```

Descriptive Form:

```
[line no.] EDGEAD vertical value data, pointer table,
                  target for upper-edge data,
                  target for lower-edge data
                  [,maximum trace width, rate of change]
```

The first and second arguments specify the source. The first is an integer array containing only verticals. The second argument is an integer array or integer waveform containing pointer values. Normally, these two arguments are those used to store acquired raw data (pointers and verticals) in a previous command.

The third and fourth arguments specify the destination. The third argument is the destination for the upper-edge data. If a simple numeric variable is specified, it is autodimensioned to the same size as the pointers array. If an integer array is specified, it must be the same size as the pointers array. If an integer waveform is specified, the component array must be the same size as the pointers array. The fourth argument is the destination for the lower-edge data. The rules for dimensioning are the same as for the third argument.

The last two arguments are optional; however, if you wish to use either one, both must be specified. These arguments define the maximum trace-width and rate-of-change parameters used in the edge determination algorithm.

With regard to the above arguments, if the pointers array is a waveform component and either of the edge arrays specified is a waveform component, then the data sampling interval, horizontal units and vertical units are copied from the pointers waveform to the edge waveforms.

The EDGEAD algorithm operates by assigning appropriate values in the vertical source array to corresponding positions in the two destination arrays. With regard to special conditions, the following rules apply:

PTW = previous trace width
CTW = current trace width
RT = rate of change variable
TW = trace-width parameter

1) Defects flagged by the DEF ON command are rejected. If no data exist or if all the data are flagged as defects, a value of -1 is stored for that horizontal location in both edge arrays. If a valid data point is found for only one edge because all other data are flagged, that point is stored in the corresponding array and a -1 is stored in the other array.

2) If two or more valid data exist, the maximum and minimum non-defect values are tested. First, the current trace width (CTW) is determined by subtracting the minimum from the maximum:

$$CTW = MAX - MIN \qquad (1)$$

For the maximum and minimum values to be accepted as data, CTW must not exceed two constraints. The first constraint is maximum trace width (TW) set by the TW command:

$$CTW \leq TW \qquad (2)$$

The second constraint is the maximum ratio of trace widths (RT) set by the RT command. The previous trace width is determined by taking the difference between the maximum and minimum values of the last pair of data points accepted by the edge operation. For the first scan and until the first pair of points is accepted, the initial value of PTW is set by:

$$PTW_i = \frac{TW}{RT} \qquad (3)$$

The ratio test is applied by comparing the ratio of the two trace widths to the parameter RT:

$$\frac{CTW}{PTW} \leq RT \qquad (4)$$

For speed, the edge algorithm actually combines the TW and RT tests into the single step given in equation 4 by limiting PTW to the initial value given in equation 3 for all vertical scans. Thus, CTW is not allowed outside the bounds set by TW, even though the two values are never directly compared.

As a result of these two tests, data outside a window set by the TW and RT parameters are rejected. This is a dynamic test that rejects noise or defects, but allows the trace width to vary as it does before, during, and after a fast transition.

If the current trace width passes both the trace width and ratio of trace width tests, the maximum value is stored in the upper-edge array and the minimum value in the lower-edge array. If the current trace width fails either of the tests, a -1 is stored in both arrays.

As a result of this processing, the edge data reside in two 512-point arrays, the upper-edge array and the lower-edge array. Each array is ordered from left-to-right of trace. Because the edge arrays reside in

the same processed memory space as the SA and ATC arrays, the pairs of arrays are mutually exclusive. The edge operation destroys the ATC and SA arrays. When read out, the edge data are ready for normalizing or geometry correction.

The power-up value of TW is 1ØØ; the power-up value of RT is 2 (although it is input or output by the 7912AD as a value multiplied by 32 for greater resolution -- see the RT command). These power-up values may not be optimum in all cases. Try larger values if a portion of the waveform is rejected during processing; try smaller values if noise or defects creep into the processed data. If the trace width is expected to stay within a given limit, for instance, set TW to that limit. Then set RT for best results.

ZREF Command. The ZREF command uses the edge data from a zero-reference trace to produce an average value representing the zero-reference level. This zero level can then be used by the NORMAD command in producing calibrated waveform data.

The format of the ZREF command is:

Syntax Form:

[line no.] **ZREF** {array / waveform} , {array / waveform} , floating-point variable

Descriptive Form:

[line no.] **ZREF** upper-edge data, lower-edge data, target for zero-reference value

The first two arguments, whether arrays or waveforms must have the same number of elements. The first argument contains upper-edge data, the second contains lower-edge data.

The third argument is the destination for the zero-value after it is computed.

The actual computation of the zero-value is done by computing mean verticals from the corresponding upper- and lower- edge data. This is done on a point-by-point basis. As each mean value is computed, it is

accumulated into a "summing" variable and a "counter" variable is incremented. When the last of the mean verticals has been accumulated, the value of the "summing" variable is divided by the value of the "counter" variable, and the result is stored as the zero-reference value.

Acquiring a zero-reference value is a three-step process. First, the ground trace is carefully positioned and acquired in standard format (verticals and pointer array). Then the EDGE (or EDGEAD) routine is executed to provide edge data. Finally, the ZREF command is used in the edge data to produce the zero value.

NOTE

The POSITION control on the 7912AD vertical plug-in should not be disturbed once a zero-reference value has been acquired. If you must reposition the waveform, be sure to reacquire a zero-reference value after all adjustments are made. Otherwise DC offset will appear in the normalized waveform.

NORMAD Command. The NORMAD command uses the edge data to produce a single-valued array of floating-point values representing the 7912AD trace.

The format of the NORMAD command is:

Syntax Form:

[line no.] **NORMAD** { array / waveform } , { array / waveform },

{ simple numeric variable / floating-point array / floating-point waveform } , expression,

expression [,floating-point variable]

Descriptive Form:

[line no.] **NORMAD** upper-edge data, lower-edge data,
target for normalized data, zero-reference value,
vertical scale factor [,target for number
of consecutive interpolated data values]

The first and second arguments specify the source arrays. The first argument is an array or waveform containing the upper-edge data. The second argument is an array or waveform containing the lower-edge data.

The third argument is the destination array for the normalized data. If a simple numeric variable name is specified, it is dimensioned as a floating-point array containing the same number of elements as each of the source arrays.

The fourth argument is an expression specifying the zero-reference value. It must evaluate to a value in the range of Ø to 511. Generally it will have been acquired with the ZREF command.

The fifth argument is an expression specifying the vertical scale factor in units per division. The specified value must not be zero.

The last argument is optional. If included, it is assigned a value indicating the maximum number of consecutive points which were interpolated during normalization.

If the upper-edge argument is a waveform and the destination argument is also a waveform, then the data-sampling interval, horizontal units, and vertical units are copied to the corresponding components of the destination waveform.

The first computation performed by the NORMAD command produces an average-to-center of trace array which is stored in the destination argument. The computation is done on a point-by-point basis for each horizontal location in which there are data; the mean of the corresponding upper- and lower-edge values is computed in extended precision. Following the computation of all 512 possible mean values, any missing values (-1's) are replaced by interpolation or extrapolation. Any missing values in the middle of the waveform are linearly interpolated using the nearest valid datum on each side of the missing datum. Any missing values occurring at the endpoints are extrapolated using the last two valid data.

At this time, the optional variable is assigned the maximum number of consecutive positions which have been filled in with interpolates (extrapolates are not counted). After command execution, this variable can be examined to determine if the edge data has missing gaps (intensity too low) or to set a bound on errors due to linear interpolation.

Finally, the result is calculated using the relation:

C = ((A+B)/2-ZR)*SF/64

In the above equation, the following notation is used:

C: destination array (3rd argument)
A: upper-edge array (1st argument)
B: lower-edge array (2nd argument)
ZR: zero-reference level (4th argument)
SF: vertical scale factor (5th argument)

If ZR is not in the range of Ø to 511, a fatal error is issued. Also, if invalid scale factor(s) are found, error is issued.

A Typical Acquisition Routine. The following program can be used to obtain a zero-corrected, normalized waveform. Following the acquisition, the waveform is graphed on the terminal screen, complete with scale factors and units. (This assumes that you have the optional TEK SPS BASIC VØ2 Graphics package.)

```
1Ø REM
2Ø REM   PROGRAM TO ACQUIRE A NORMALIZED 7912AD WAVEFORM
3Ø REM
4Ø INTEGER A(511),B(511)
5Ø WAVEFORM WA IS A,IA,HA$,VA$
6Ø WAVEFORM WB IS B,SB,HB$,VB$
7Ø WAVEFORM WC IS C(511),SC,HC$,VC$
8Ø SIFTO @Ø,3ØØØ
9Ø LA=Ø+32
1ØØ TA=Ø+64
11Ø SA=Ø+96
12Ø PUT "MODE TV" INTO @Ø,LA,SA
13Ø PRINT "TO ACQUIRE ZERO-REFERENCE TRACE,"
135 PRINT "GROUND VERTICAL PLUG-IN, ADJUST INTENSITY,"
```

```
140 PRINT "POSITION TRACE AND PRESS ANY KEY"
150 SIFCOM @0,LA,SA,"GTL"
160 WAIT
170 GOSUB 1000
180 EDGEAD QQ,P,A,B
190 ZREF A,B,ZR
200 PRINT "TO ACQUIRE WAVEFORM TRACE,"
205 PRINT "UNGROUND VERTICAL PLUG-IN, ESTABLISH WAVEFORM,"
210 PRINT "ADJUST INTENSITY AND PRESS ANY KEY"
220 SIFCOM @0,LA,SA,"GTL"
230 WAIT
240 GOSUB 1000
250 EDGEAD QQ,WP,WA,WB
260 NORMAD WA,WB,WC,ZR,VS
270 PAGE
280 GRAPH WC
290 STOP
1000 REM
1010 REM   SUBROUTINE TO ACQUIRE RAW WAVEFORM DATA
1020 REM
1030 INTEGER P(511)
1040 WAVEFORM WP IS P,SP,HP$,VP$
1050 PUT "DIG DAT" INTO @0,LA,SA
1060 PUT "READ PTR,VER" INTO @0,LA,SA
1070 GOSUB 2000
1080 P=QQ
1090 GOSUB 2000
1100 PUT "MODE TV" INTO @0,LA,SA
1110 PUT "HS1?" INTO @0,LA,SA
1120 GET A$ FROM @0,TA,SA
1130 SP=VAL(SEG(A$,5,LEN(A$)-1))
1140 SP=SP/51.2
1150 PUT "VS1?" INTO @0,LA,SA
1160 GET A$ FROM @0,TA,SA
1170 VS=VAL(SEG(A$,5,LEN(A$)-1))
1180 PUT "HU1?" INTO @0,LA,SA
1190 GET A$ FROM @0,TA,SA
1200 HP$=SEG(A$,5,LEN(A$)-1)
1210 PUT "VU1?" INTO @0,LA,SA
1220 GET A$ FROM @0,TA,SA
1230 VP$=SEG(A$,5,LEN(A$)-1)
1240 RETURN
```

```
2000 REM
2010 REM   SUBROUTINE TO READ DATA ARRAY
2020 REM
2030 DELETE QQ
2040 IFDTM @0,"UNP"
2050 GET X FROM @0,TA,SA
2060 IF CHR(X)<>"%" THEN STOP
2070 IFDTM @0,"PAK","HBF"
2080 GET CW FROM @0,TA,SA
2090 CW=(CW-1)/2-1
2100 INTEGER QQ(CW)
2110 GET QQ FROM @0,TA,SA
2120 IFDTM @0,"UNP"
2130 GET X FROM @0,TA,SA
2140 GET X FROM @0,TA,SA
2150 IF CHR(X)<>";" THEN STOP
2160 RETURN
```

For this example, we've made a subroutine from lines 140 through 340 of our previous example. This subroutine returns pointer data with waveform information in WP and vertical data in array QQ. The subroutine is numbered from line 1000 to 1240.

Lines 10 through 110 of the main program are initialization. The REM commands identify the program. Lines 40 through 70 define two integer waveforms, WA for upper-edge data and WB for lower-edge data. Waveform WC, the destination for the normalized waveform, is defined at line 80. The interface time-out value is set to 3000 milliseconds and the talk, listen, and secondary instrument addresses are initialized.

At line 120, the 7912AD is placed in "TV mode" so the ground reference trace can be seen on the monitor. Lines 130 and 140 print a message explaining how to set up the instrument to acquire a ground reference trace. The instrument is set to local mode. When a terminal key is pressed, the WAIT command at line 160 is exited. The subroutine at line 1000 is called to get raw waveform data. The EDGEAD command, ignoring the waveform information this time, converts the raw data to upper- and lower-edge data, and the zero-reference value is computed at line 190.

Lines 200 through 260 acquire and normalize a waveform. Lines 200 and 210 display a message explaining how to set up the instrument to acquire the waveform. The instrument is again instructed to go to local

mode. When the WAIT command at line 23Ø is exited, the subroutine at line 1ØØØ is again called to return raw waveform data. The EDGEAD command, this time using the waveform information, creates upper- and lower-edge waveforms. The normalized waveform, WC, is returned from the NORMAD command at line 26Ø.

The terminal screen is erased and the normalized waveform is GRAPHed at line 28Ø. The graph is shown in Fig. 3.

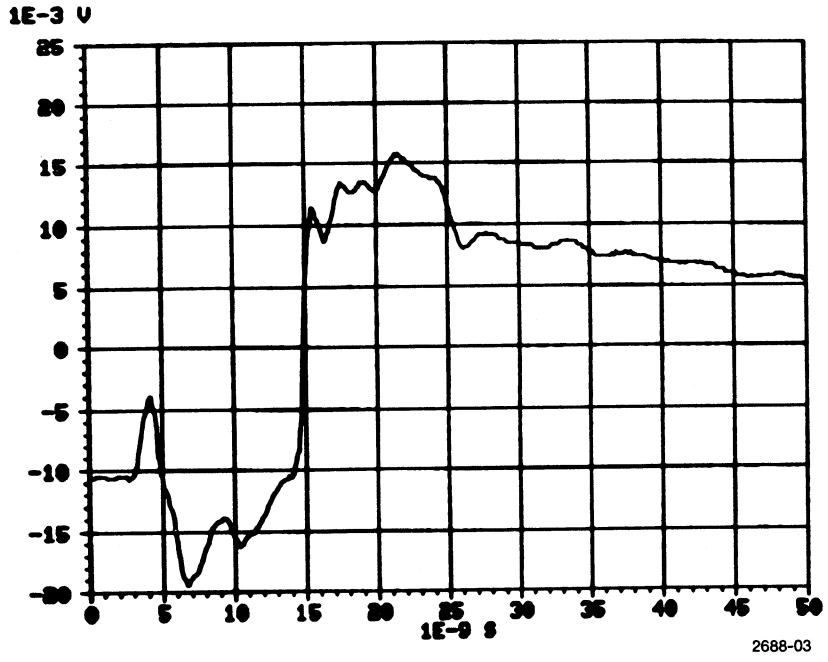


Fig. 3. Normalized 7912AD waveform.

Defect Rejection

The 7912AD scan converter target may have some slight defects resulting from burns or substrate irregularities. These defects appear as tiny bright spots that are unaffected by beam intensity adjustment. When waveform data are read from the target, these defects appear as added data and are read with the waveform. Later processing could be significantly affected by this extraneous data, so defect removal algorithms were included in the 7912AD design.

Defect removal consists of one or more 7912AD digitize cycles performed with the writing beam intensity at minimum (off). Any data read from the target under those conditions is considered to be defect

information. A special 7912AD command (DIG DEF) causes the 7912AD to read and store defect information in an internal memory. A second command (DEF ON) causes the 7912AD to automatically remove defect information from acquired waveforms. Both of these operations are internal to the 7912AD. Further information can be found in the 7912AD Operator's manual.

Defect removal can also be performed in the controller with the TEK SPS BASIC REJECT command. The REJECT command uses defect data read from the 7912AD memory (acquired with a READ DEF command) to flag defect elements in an array. Subsequent processing (with EDGEAD for example) removes the flagged defect data. The REJECT command is described in detail below.

REJECT Command. The REJECT command provides a way to flag the vertical data that have been identified as defects.

The format of REJECT is:

Syntax Form:

[line no.] **REJECT** integer array, { integer array / integer waveform }, integer array

Descriptive Form:

[line no.] **REJECT** vertical value data which is target for defect flags, pointer table, defect data

The first argument specifies the vertical data array in which the defects are to be flagged. The second argument specifies the corresponding pointers array or waveform. The third argument specifies the defect array, previously read from the instrument with the READ DEF command.

Data identified as a defect (i.e. corresponding to one of the defect array entries) is flagged by negating its value. When a subsequent EDGEAD routine is performed, the negative values are ignored. If desired, defects flagged by REJECT can be unflagged with the ABS function (which computes the absolute value of the array). This effectively reinstates the defects. Since the verticals and pointer arrays are rebuilt during

each acquisition, the REJECT command should be executed after each acquisition of the raw data in standard format.

The following program is similar to earlier examples, but defect removal has been added. The highlighted areas illustrate the additional code.

```
10 REM
20 REM   PROGRAM TO ACQUIRE A NORMALIZED 7912AD WAVEFORM
30 REM          WITH CRT TARGET DEFECT REJECTION
40 REM
50 INTEGER A(511),B(511)
60 WAVEFORM WA IS A,IA,HA$,VA$
70 WAVEFORM WB IS B,SB,HB$,VB$
80 WAVEFORM WC IS C(511),SC,HC$,VC$
90 SIFTO @0,3000
100 LA=0+32
110 TA=0+64
120 SA=0+96
130 PRINT "DIGITIZING CRT TARGET DEFECTS 64 TIMES"
140 PUT "DIG DEF,64" INTO @0,LA,SA
150 PUT "READ DEF" INTO @0,LA,SA
160 GOSUB 2000
165 FD=CW | IF FD= -1 THEN GOTO 200
170 DELETE DF
180 INTEGER DF(SIZ(QQ)-1)
190 DF=QQ
200 PUT "MODE TV" INTO @0,LA,SA
210 PRINT "TO ACQUIRE ZERO-REFERENCE TRACE, GROUND VERTICAL PLUG-IN"
220 PRINT "ADJUST INTENSITY, POSITION TRACE AND PRESS ANY KEY"
230 SIFCOM @0,LA,SA,"GTL"
240 WAIT
250 GOSUB 1000
260 EDGEAD QQ,P,A,B
270 ZREF A,B,ZR
280 PRINT "TO ACQUIRE WAVEFORM TRACE, UNGROUND VERTICAL PLUG-IN"
290 PRINT "ESTABLISH WAVEFORM, ADJUST INTENSITY AND PRESS ANY KEY"
300 SIFCOM @0,LA,SA,"GTL"
310 WAIT
320 GOSUB 1000
330 EDGEAD QQ,WP,WA,WB
340 NORMAD WA,WB,WC,ZR,VS
350 PAGE
```

```
360 GRAPH WC
370 STOP
1000 REM
1010 REM   SUBROUTINE TO ACQUIRE RAW WAVEFORM DATA
1020 REM
1030 INTEGER P(511)
1040 WAVEFORM WP IS P,SP,HP$,VP$
1050 PUT "DIG DAT" INTO @0,LA,SA
1060 PUT "READ PTR,VER" INTO @0,LA,SA
1070 GOSUB 2000
1080 P=QQ
1090 GOSUB 2000
1100 REJECT QQ,P,DF   IF FD<>-1 THEN REJECT QQ,P,DF
1110 PUT "MODE TV" INTO @0,LA,SA
1120 PUT "HS1?" INTO @0,LA,SA
1130 GET A$ FROM @0,TA,SA
1140 SP=VAL(SEG(A$,5,LEN(A$)-1))
1150 SP=SP/51.2
1160 PUT "VS1?" INTO @0,LA,SA
1170 GET A$ FROM @0,TA,SA
1180 VS=VAL(SEG(A$,5,LEN(A$)-1))
1190 PUT "HU1?" INTO @0,LA,SA
1200 GET A$ FROM @0,TA,SA
1210 HP$=SEG(A$,5,LEN(A$)-1)
1220 PUT "VU1?" INTO @0,LA,SA
1230 GET A$ FROM @0,TA,SA
1240 VP$=SEG(A$,5,LEN(A$)-1)
1250 RETURN
2000 REM
2010 REM   SUBROUTINE TO READ DATA ARRAY
2020 REM
2030 DELETE QQ
2040 IFDTM @0,"UNP"
2050 GET X FROM @0,TA,SA
2060 IF CHR(X)<>"%" THEN STOP
2070 IFDTM @0,"PAK","HBF"
2080 GET CW FROM @0,TA,SA
2090 CW=(CW-1)/2-1   IF CW=-1 THEN GOTO 2120
2095 (arrow to 2090)
2100 INTEGER QQ(CW)
2110 GET QQ FROM @0,TA,SA
2120 IFDTM @0,"UNP"
2130 GET X FROM @0,TA,SA
```

```
2140 GET X FROM @0,TA,SA
2150 IF CHR(X)<>";" THEN STOP
2160 RETURN
```

The boldfaced portions of this example were added to our previous example to show the steps necessary to detect and reject CRT target defects from 7912AD raw data. This example has been renumbered, so the line numbers do not match the previous example exactly.

Lines 130 and 140 display a message explaining that the potential defects are being digitized 64 times. In practice, the number may be as many or as few times as you require. Line 140 causes the main writing beam to be turned off and the defect digitizing to begin. The defects are then read into an integer defect array, DF.

Since defects should be rejected from the raw data array, the command that performs the rejection has been placed in the subroutine at line 1000 which acquire raw data. The REJECT command at line 1100 causes all defects in the vertical data array, QQ, to be flagged. These flagged values will later be ignored by the EDGEAD command when the edge arrays are computed.

Correction of Geometry Distortion

Just as optical lens systems suffer distortions due to aberrations of one sort or another, electrostatic and magnetic lens systems suffer similar distortions. These distortions fall into two main classes. The first class includes those effects which cause the electron beam to become defocussed or to have coma (comet shaped) distortion. The second class is image distortion, where the image, rather than the beam, becomes distorted. It is this second class with which geometry correction routines must deal.

There are several types of image distortion. Some of the more common types are described as "pincushion" distortion, "barrel" distortion, and "keystone" distortion. Although these effects are present to some degree in all lens systems (including conventional cathode ray tubes), the problem can be slightly more pronounced in the 7912AD scan-converter tube due to the presence of two electron guns -- one for the writing beam and one for the reading beam. Further, the applications of the 7912AD, particularly those requiring the digital signal processing capability of TEK SPS BASIC, often require a high degree of accuracy. To enhance the

overall performance of the 7912AD in these critical applications, geometry correction routines have been included in the software.

Geometry correction for the 7912AD is based on the instrument's internally generated electronic dot graticule. Normally the dot graticule is displayed as a 9 by 11 grid of dots which divide the usable area of the scan-converter target into 8 vertical divisions and 1Ø horizontal divisions.

Since the dot graticule, when digitized and acquired, is subject to the same two-dimensional distortion as the digitized waveforms, it provides a convenient means of determining the amount of distortion for any usable area of the target. When this is compared with the "software-generated" graticule (the line graticule displayed on the computer terminal during a GRAPH command), it is possible to assess the amount of distortion for any dot in the grid.

Also, by comparing the actual and ideal coordinates of each "dot" of the graticule, the magnitude of the horizontal and vertical corrections required can be computed for any data point. These corrections, when applied to all points of a waveform, produce a geometry-corrected waveform which is relatively free of distortion.

It is important to realize that the TEK SPS BASIC geometry correction routines do not, in themselves, serve as a complete autocalibration system. Specifically, the geometry correction routines correct for 7912AD geometry distortion as assessed by the dot-graticule generator. However, since the 7912AD dot generator operates independently of the 7912AD time-base and vertical plug-ins, the correction routines will not correct for nonlinearity that arises from the analog portion of the plug-ins. Therefore, if you are concerned about total accuracy of data, the linearity of the analog input circuitry should be checked. For example, you can apply known test signals to the plug-ins and observe the geometry-corrected waveform. This allows you to apply separate non-linearity corrections.

Geometry correction is accomplished by two separate TEK SPS BASIC commands. The first step is the installation of the geometry correction tables and is accomplished by the INSTAD command. During this process, the dot graticule data in array R is analyzed to determine the x and y coordinates of each dot of the graticule grid. Since there are several raw data points representing each of the dots in the graticule grid, an analysis is done to determine the "center" of each dot. Floating-point

values for the x and y coordinates of each dot are stored in corresponding elements of array arguments specified by the INSTAD command. These arrays become the "distorted standard" for geometry correction.

The actual correction of waveform data is accomplished by the MAPAD command, which corrects each edge array of waveform data output by the instrument or by the EDGEAD command. (The MAPAD command can also be used to correct a normalized waveform array, but the correction process is less accurate.) The MAPAD command uses the dot graticule "center-of-dot" coordinate values (set in the specified array arguments of INSTAD) and the corresponding ideal coordinates (corresponding to the software graticule intersections) to compute the necessary corrections on values in the edge arrays. After each edge array has been corrected by the MAPAD command, the NORMAD command is then used to return a single, geometry-corrected trace.

INSTAD Command. A preliminary step to enable geometry correction of waveforms is accomplished by the INSTAD command; it generates the horizontal and vertical correction tables used by the MAPAD command. (It does this by using the verticals and pointers arrays which are part of the 7912AD standard format.) Before executing the INSTAD instruction, the dot graticule must have been called up for display and adjusted for proper intensity. Subsequently, the graticule must be digitized and acquired into controller memory. A later example routine will illustrate the procedure.

The format of INSTAD is:

Syntax Form:

[line no.] **INSTAD** integer array, { integer array / integer waveform } ,

{ simple numeric variable / floating-point array } , { simple numeric variable / floating-point array }

Descriptive Form:

[line no.] **INSTAD** vertical value data, pointer table,
target for horizontal correction table,
target for vertical correction table

The first argument specifies the vertical data array and the second argument specifies the pointers array or waveform. These are source arguments.

The third argument specifies the destination for the horizontal correction table and the fourth argument specifies the destination for the vertical correction table. In either case, if an array is specified, it must have been dimensioned as a 99-element floating-point array. If a simple numeric variable is specified, it is autodimensioned as a 99-element floating-point array. At command completion, the third and fourth points in the 9 by 11 dot graticule grid.

NOTE

For the INSTAD routine to work optimally, the 7912AD should be well calibrated for digital mode. In some cases, this results in the top and bottom rows of graticule dots not being visible on the TV monitor. Also, the GRATICULE Intensity should be set so that the dot size (diameter) is within 5% to 15% of the center-to-center distance between dots.

The INSTAD algorithm begins with a software routine which places proper bounds around each of the dots in the 7912AD dot graticule grid. The size of the bounds is 64 raw data levels high and 51.2 raw data points wide. Figure 4 illustrates the boundaries placed by this routine.

Next, coordinates are computed for all interior dots (all dots except those lying on the outer boundary squares). In the ideal case where there is sufficient data for each dot present, the x and y coordinates are obtained by averaging the upper and lower verticals associated with each horizontal corresponding to a particular dot. Then the mean of the "average" verticals is computed to yield the final vertical coordinate y_n for that dot. To determine the horizontal coordinate x_n for the same dot, it is only necessary to compute the mean of each of the horizontal values associated with that particular dot. This is illustrated in Fig. 5.

As each of the horizontal and vertical dot coordinates are computed, the maximum height and width of each dot is determined. At the end of the grid search, these values are averaged to determine the mean height and

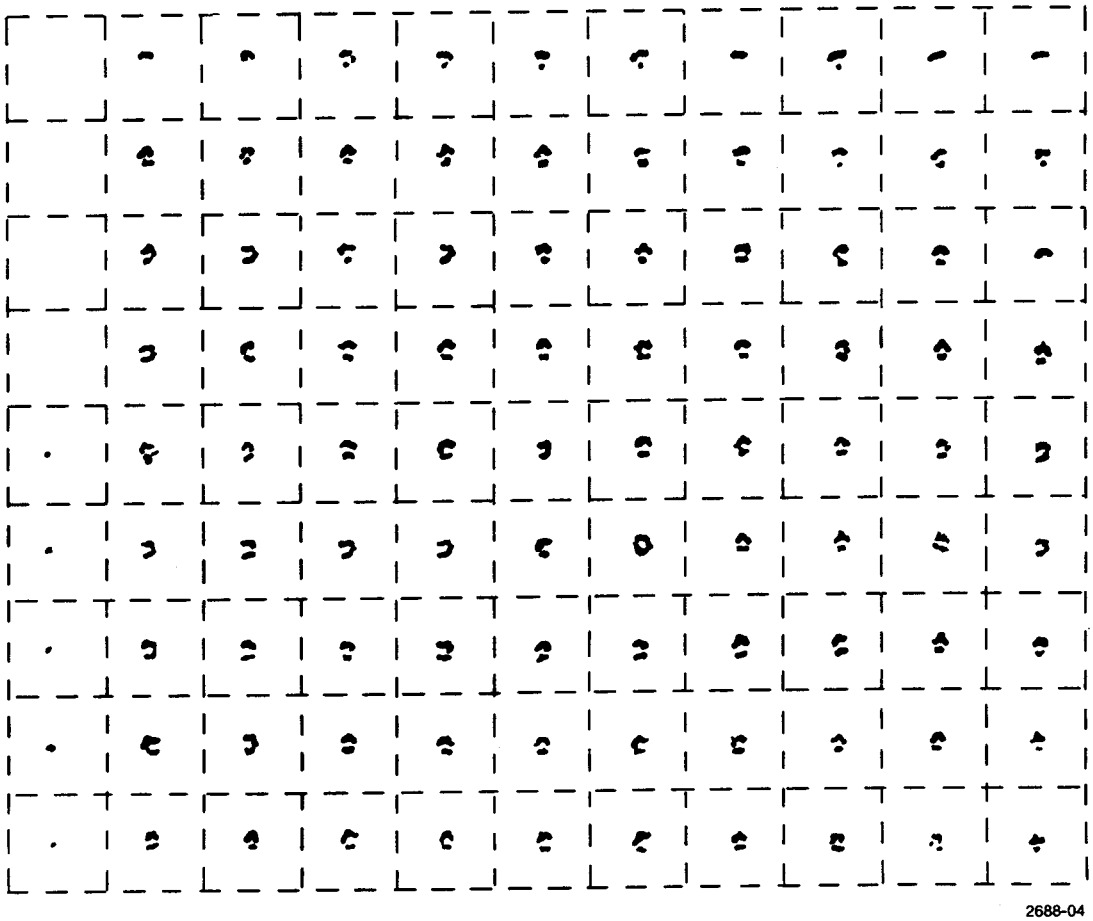


Fig. 4. Graticule dot boundaries.

width of all the dots. This value is used in determining coordinates of boundary dots that are partially out of bounds (explained shortly). If a search of the grid for any interior dots fails to produce data, a fatal error is issued. When such an error occurs, you should increase the dot graticule intensity, reacquire the dot graticule, and reexecute the INSTAD routine.

At this time, the dots lying in the boundary squares are processed. In this case, however, the coordinates of missing dots are filled in by extrapolation. Furthermore, the coordinates of boundary dots which are less than or equal to 2/3 of the average dot size (computed previously) are adjusted outward proportionately. This is done to estimate the center of the semicircular dots resulting from part of the boundary dots falling outside the active target area.

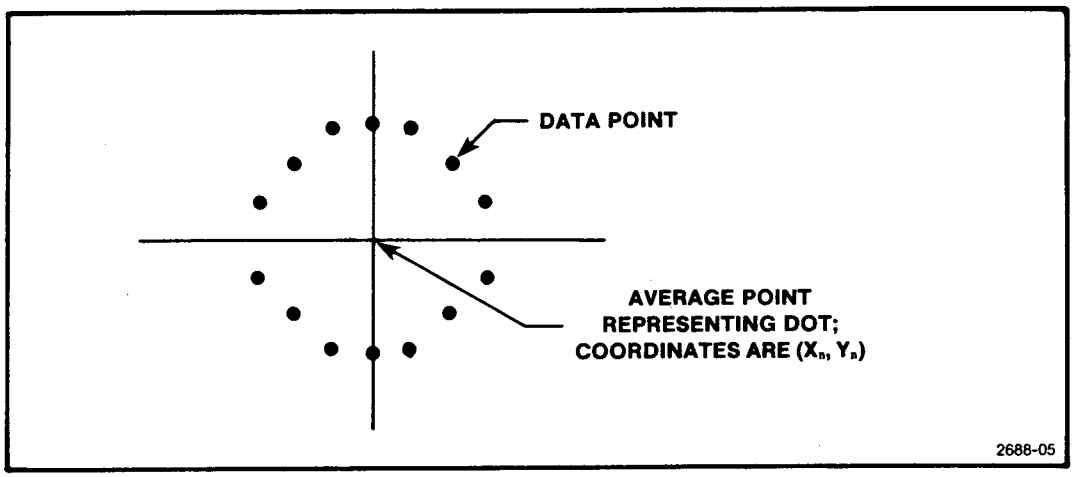


Fig. 5. Determining dot centers.

Assuming that the INSTAD routine has been executed successfully, the raw data coordinates representing the 99 dots are stored in arrays specified as arguments for INSTAD. The third array argument is used for the horizontal coordinates for each of the dots and the fourth array is used for the vertical coordinates. These coordinates are entered successively into the arrays, beginning with the lower-left dot and ending with the upper-right dot. The order of entry is from the bottom to the top of each column, proceeding from the left-most column to the right-most column.

Once all of the horizontal and vertical coordinates have been successfully stored in the specified arrays, the INSTAD routine is complete. At that time, you can examine the destination arrays to determine the adequacy of the correction process. A very effective way to examine them is to GRAPH them on the terminal. By noting the piecewise linearity of the graphed arrays, you can quickly assess the adequacy of the geometry correction. Fig. 6 shows a combined graph of arrays H and V for a typical correction process. (In this example, these arrays contain the horizontal and vertical correction tables.) Notice that the graph of the horizontal correction table is normally a step function while the graph of the vertical correction table is normally a sawtooth function. Any great departure from these general forms may be cause for concern.

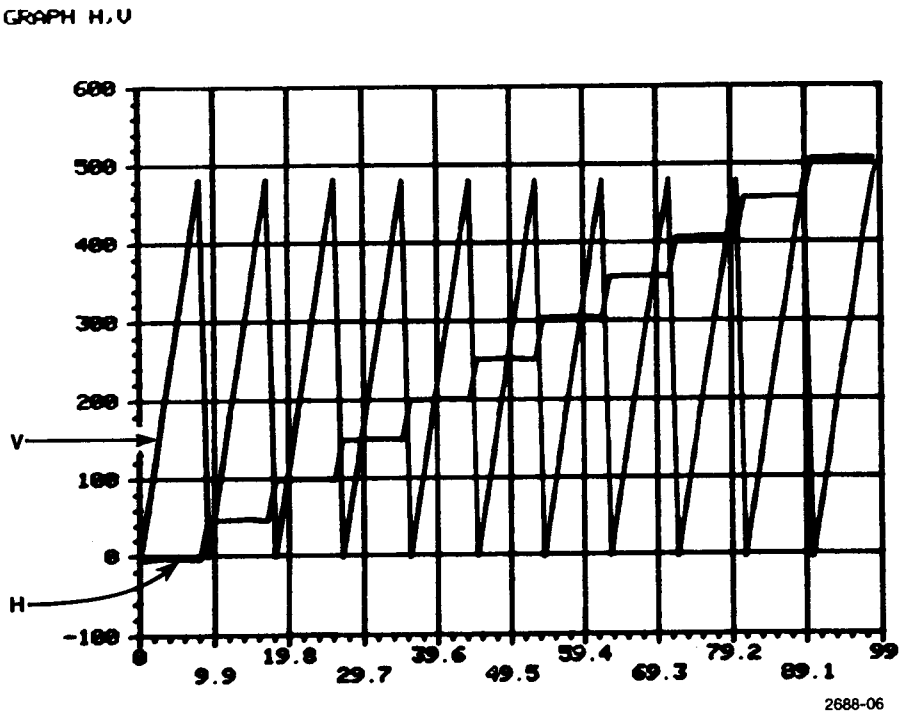


Fig. 6. Typical geometry correction arrays.

The edge data to be corrected is typically positive integer values. This is not critical except that any negative values are treated as missing data and values larger than 511 are set to 511. The missing data indicator is "-1". The MAPAD command interpolates missing interior data but does not extrapolate values which fall outside the "correction window" defined by the correction tables. As missing interior values are interpolated, a count is made of the number of consecutive interpolated values. The optional variable is then set to the maximum consecutive count obtained. This allows you to determine if the edge data has gaps (intensity too low) or to set a bound on the errors due to linear interpolation.

MAPAD Command. The MAPAD command performs geometry correction on 7912AD edge data (or data output by the EDGEAD instruction). It uses the "distorted" dot coordinates output by the INSTAD routine to determine the distortion for any area of the 7912AD scan-converter target. Then MAPAD applies appropriate horizontal and vertical corrections to the edge data.

The format of the MAPAD command is:

Syntax Form:

[line no.] **MAPAD** {array / waveform} , {array / waveform} ,floating-point array,

floating-point array [,floating-point variable]

Descriptive Form:

[line no.] **MAPAD** upper- or lower-edge data, target for geometry-corrected upper- or lower-edge data, horizontal correction table, vertical correction table [,target for maximum number of consecutive interpolated data values]

The first argument specifies a 512-element array or waveform which contains edge data to be corrected. (It may also contain normalized data although the correction process is not as accurate.) At the end of the command execution, the second argument specified is filled with corrected edge data (or corrected normalized data). It must also be a 512-element array or waveform. If both the source and destination are waveforms, then the data-sampling interval, horizontal units and vertical units are copied from the source to the destination.

The third and fourth arguments specify the horizontal and vertical correction tables respectively, as generated by the INSTAD command. These are both 99-element floating-point arrays and usually correspond to the same arguments used in the most recent INSTAD command.

The last argument, which is optional, is a floating-point variable. It is assigned the maximum number of consecutive interpolated data values in the destination array.

To aid your understanding of the MAPAD algorithm, an appendix is included at the end of this manual. It explains the mathematical foundations for geometry correction in cathode ray tubes. Using this appendix as a basis for discussion, the MAPAD command may be described as follows: First the MAPAD command and its arguments are checked for proper format and dimensions. Appropriate error messages are issued if the format or dimensions are incorrect.

Next, the boundaries of the correction window are established according to the layout of the 7912AD dot graticule. The boundaries are set as shown in Fig. 7. They are the upper and lower graticule row, respectively. Similarly, the left and right boundaries are set to the maximum and minimum coordinate values in the extreme left and extreme right graticule columns respectively. Any data residing outside the bounds of this correction window will be ignored by the MAPAD command, which sometimes results in missing data at either end or at the top and bottom of a MAPAD waveform.

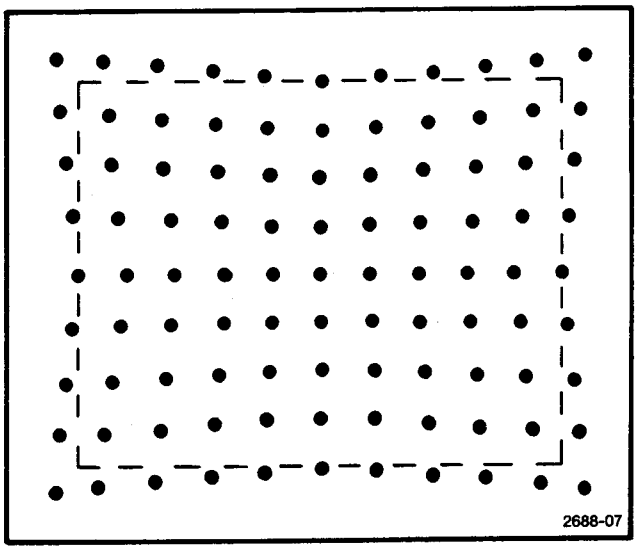


Fig. 7. Correction window boundaries.

At this time, each data point within the correction window is searched to find which "distorted" rectangle (quadrilateral) it resides within. For each data point, the search proceeds generally from left to right. The search is done in such a manner that the containing rectangle is located in the shortest possible time.

Once the containing rectangle is found, a test is made to determine whether the data point resides in the upper or lower triangle. When this determination has been made, the actual measurement and correction of geometry errors can begin. This correction is done in accordance with the equations listed in the appendix by substituting the values of the coordinates of the containing triangle and the coordinates of the data point. For each data point, a corrected horizontal address and its corrected vertical value is computed.

In a few cases, the corrected horizontal may be an integer value and thus the corrected value can be mapped directly into the new horizontal location. In most instances, however, the corrected horizontal is a fractional value. In these cases, it is necessary to interpolate a corrected vertical value for the nearest location that approximates the actual horizontal location.

During this interpolation phase of the MAPAD routine, any missing data on the inside of the correction window are filled in through linear interpolation. However, any mising data at the ends of the waveform is indicated by "-1" values but is not extrapolated. (If, however, you NORMAD the corrected edges, the resulting array is extrapolated.) As linear interpolation is occurring, the optional argument for MAPAD is set to the largest number of consecutive locations filled in through linear interpolation, as mentioned previously.

At the end of MAPAD command execution, the mapped edge data is output to the first array argument specified. The third and fourth array arguments still contain the 99-element horizontal and vertical correction tables.

NOTE

> The execution time of the MAPAD command is extremely data-dependent. In most cases, it requires at least 15 seconds on a CP1160 Controller.

The following program combines all the operations discussed so far. It can be used to acquire a waveform with scale-factors, perform defect removal, geometry correction, scale the waveform, and graph it on the terminal screen. Any new program segments are shown in bold print, as before.

```
1Ø REM
2Ø REM  PROGRAM TO ACQUIRE A NORMALIZED 7912AD WAVEFORM
3Ø REM          WITH CRT TARGET DEFECT REJECTION
4Ø REM          AND GEOMETRY CORRECTION
5Ø INTEGER A(511),B(511)
6Ø WAVEFORM WA IS A,IA,HA$,VA$
7Ø WAVEFORM WB IS B,SB,HB$,VB$
8Ø WAVEFORM WC IS C(511),SC,HC$,VC$
9Ø SIFTO @Ø,3ØØØ
```

```
100 LA=0+32
110 TA=0+64
120 SA=0+96
130 PRINT "DIGITIZING CRT TARGET DEFECTS 64 TIMES"
140 PUT "DIG DEF,64" INTO @0,LA,SA
150 PUT "READ DEF" INTO @0,LA,SA
160 GOSUB 2000
165 FD=CW\IF FD=-1 THEN GOTO 200
170 DELETE DF
180 INTEGER DF(SIZ(QQ)-1)
190 DF=QQ
200 PRINT "TO INSTALL GEOMETRY CORRECTION,"
205 PRINT "TURN BEAM INTENSITY DOWN,"
210 PRINT "ADJUST GRATICULE INTENSITY AND PRESS ANY KEY"
220 SIFCOM @0,LA,SA,"GTL"
230 WAIT
240 PUT "DIG GRAT" INTO @0 LA,SA
250 GOSUB 1000
260 DIM CH(98),CV(98)
270 INSTAD QQ,P,CH,CV
280 PUT "MODE TV" INTO @0,LA,SA
290 PRINT "TO ACQUIRE ZERO-REFERENCE TRACE, GROUND VERTICAL PLUG-IN"
300 PRINT "ADJUST INTENSITY, POSITION TRACE AND PRESS ANY KEY"
310 SIFCOM @0,LA,SA,"GTL"
320 WAIT
330 GOSUB 1000
340 EDGEAD QQ,P,A,B
350 DIM D(511)
360 MAPAD A,D,CH,CV
370 A=D
380 MAPAD B,C,CH,CV
390 B=D
400 ZREF A,B,ZR
410 PRINT "TO ACQUIRE WAVEFORM TRACE, UNGROUND VERTICAL PLUG-IN"
420 PRINT "ESTABLISH WAVEFORM, ADJUST INTENSITY AND PRESS ANY KEY"
430 SIFCOM @0,LA,SA,"GTL"
440 WAIT
450 GOSUB 1000
460 EDGEAD QQ,WP,WA,WB
470 MAPAD A,D,CH,CV
480 A=D
490 MAPAD B,D,CH,CV
500 B=D
510 DELETE D,QQ,P
```

```
52Ø NORMAD WA,WB,WC,ZR,VS
53Ø PAGE
54Ø GRAPH WC
55Ø STOP
1ØØØ REM
1Ø1Ø REM   SUBROUTINE TO ACQUIRE RAW WAVEFORM DATA
1Ø2Ø REM
1Ø3Ø INTEGER P(511)
1Ø4Ø WAVEFORM WP IS P,SP,HP$,VP$
1Ø5Ø PUT "DIG DAT" INTO @Ø,LA,SA
1Ø6Ø PUT "READ PTR,VER" INTO @Ø,LA,SA
1Ø7Ø GOSUB 2ØØØ
1Ø8Ø P=QQ
1Ø9Ø GOSUB 2ØØØ
11ØØ REJECT QQ,P,DF
111Ø PUT "MODE TV" INTO @Ø,LA,SA
112Ø PUT "HS1?" INTO @Ø,LA,SA
113Ø GET A$ FROM @Ø,TA,SA
114Ø SP=VAL(SEG(A$,5,LEN(A$)-1))
115Ø SP=SP/51.2
116Ø PUT "VS1?" INTO @Ø,LA,SA
117Ø GET A$ FROM @Ø,TA,SA
118Ø VS=VAL(SEG(A$,5,LEN(A$)-1))
119Ø PUT "HU1?" INTO @Ø,LA,SA
12ØØ GET A$ FROM @Ø,TA,SA
121Ø A$=SEG(A$,5,LEN(A$)-1)
122Ø PUT "VU1?" INTO @Ø,LA,SA
123Ø GET A$ FROM @Ø,TA,SA
124Ø VP$=SEG(A$,5,LEN(A$)-1)
125Ø RETURN
2ØØØ REM
2Ø1Ø REM   SUBROUTINE TO READ DATA ARRAY
2Ø2Ø REM
2Ø3Ø DELETE QQ
2Ø4Ø IFDTM @Ø,"UNP"
2Ø5Ø GET X FROM @Ø,TA,SA
2Ø6Ø IF CHR(X)<>"%" THEN STOP
2Ø7Ø IFDTM @Ø,"PAK","HBF"
2Ø8Ø GET CW FROM @Ø,TA,SA
2Ø9Ø CW=(CW-1)/2-1
21ØØ INTEGER QQ(CW)
211Ø GET QQ FROM @Ø,TA,SA
212Ø IFDTM @Ø,"UNP"
```

```
213Ø GET X FROM @Ø,TA,SA
214Ø GET X FROM @Ø,TA,SA
215Ø IF CHR(X)<>";" THEN STOP
216Ø RETURN
```

In this example, the boldfaced lines are added to provide correction of distortions in the geometry of the CRT. Again, the example has been renumbered and does not exactly match the previous example.

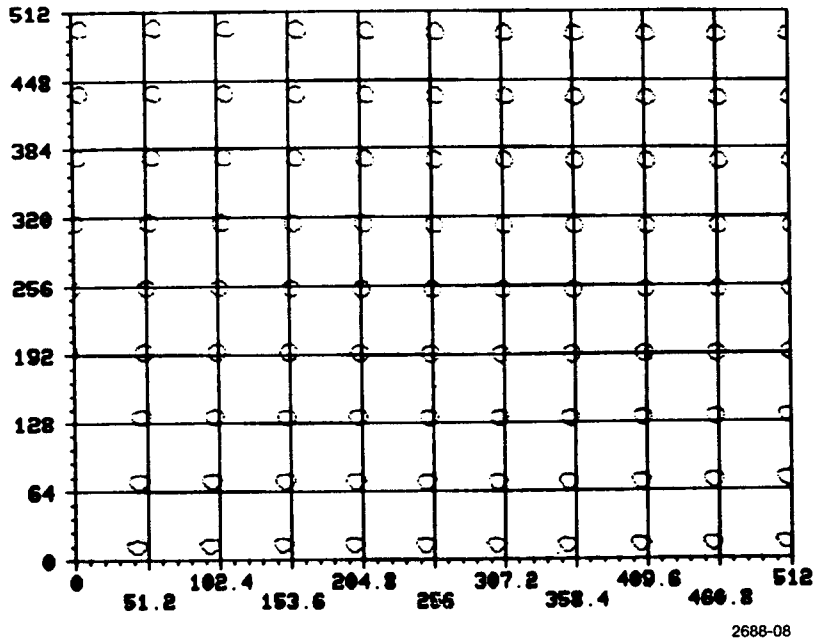


Fig. 8. 7912AD dot graticule.

Lines 2ØØ through 27Ø acquire a dot graticule (shown in Fig. 8) and build the geometry correction tables. Lines 2ØØ and 21Ø display a message explaining how to set up the instrument to acquire the dot graticule. The 7912AD is put into local mode to facilitate the adjustment. When the instrument is ready and a terminal key has been pressed, the WAIT command at line 23Ø is exited. The graticule is then digitized and is acquired as raw data using the subroutine starting at line 1ØØØ. Arrays for the horizontal and vertical portions of the correction table, CH and CV, are defined and the correction tables are calculated at line 27Ø.

Geometry correction is performed on the edge data of the zero-reference trace from lines 35Ø through 39Ø. Line 35Ø defines an array, D, which temporarily holds the output of the MAPAD command which performs the geometry correction. Lines 36Ø and 37Ø correct distortions in the upper-edge array; lines 38Ø and 39Ø correct the lower-edge array. The edge arrays are then ready to be used to determine the zero-reference value.

Correction of the waveform edge arrays is performed from line 47Ø through line 51Ø. Note that the waveform information available with the upper- and lower-edge arrays, WA and WB, need not be used with the MAPAD command, though waveform information could be passed through the command. The temporary edge array, D, and the raw data arrays, QQ and P, are deleted at line 51Ø since they are no longer necessary and take considerable free memory. A fully normalized, corrected waveform is shown in Fig. 9. Note the difference in this figure and Fig. 3.

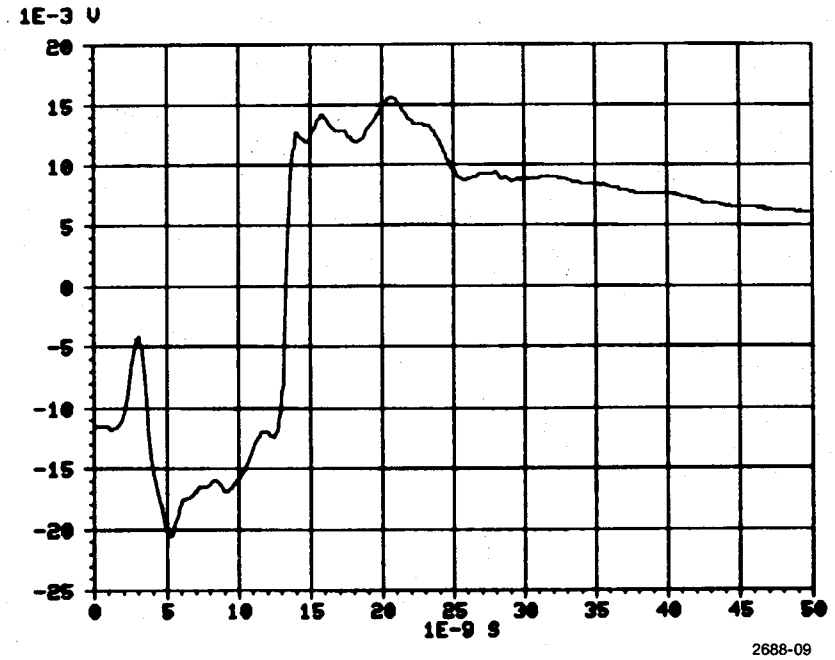


Fig. 9. Fully normalized, corrected waveform.

Data Logging from the 7912AD

ADLOG Command. The ADLOG command acquires raw waveform data from a 7912AD at very high speed. The acquisition routine uses the REPEAT mode of the 7912AD and the DMA (Direct Memory Access) capabilities of the IEEE 488 interface to achieve high throughput. Transfer rate varies with controller type, instrument settings, and transfer mode, but 2Ø-25 waveforms per second is typical. Output must be to a file previously OPENed FOR WRITE on a peripheral device that supports DMA data transfers (e.g., DK or DL).

The format of the ADLOG command is:

Syntax Form:

[line no.] **ADLOG** **#**expression **FROM @**expression,expression,

expression,expression[,**FAST**]

Descriptive Form:

[line no.] **ADLOG** **#**target plun **FROM @**IEEE 488 interface number,
talk address, secondary address,
number of times to digitize [,**FAST** mode switch]

The first argument specifies the peripheral logical unit number (PLUN) of the destination file (OPENED FOR WRITE). Because this command uses free memory for data transfers, opening this file with multiple buffers degrades logging performance.

The second argument is the number of the IEEE 488 interface through which the data is acquired.

The third argument is the instrument talk address. The address is derived from the instrument bus address (Ø-3Ø) plus 64. Thus, a primary talk address of Ø becomes talk address 64 for ADLOG.

The fourth argument is the instrument secondary address. This address is derived from the instrument seondary bus address (Ø-3Ø) plus 96. Thus, secondary address Ø becomes 96 for ADLOG.

The fifth argument is the number of digitize cycles required. Any integer number or expression that resolves to an integer can be used.

The optional keyword **FAST** instructs the internal disk driver to not check disk head positioning before writing to the disk. This option increases the logging rate by about 2Ø%, but at some risk. When using ths option, writing to a scratch or otherwise unused disk is recommended. Normally, no problems will be encountered using FAST, but if you get P18 errors from TEK SPS BASIC when reading the file, FAST could be the culprit.

Examples:

```
1ØØ OPEN #2 AS DK1:FN$ FOR WRITE
11Ø ADLOG #2 FROM @Ø,64,96,1Ø

2ØØ OPEN #DF AS DK1:DL$ FOR WRITE
21Ø ADLOG #DF FROM @2,TA,SA,N,FAST
```

The first example (line 11Ø) logs ten waveforms from a 7912AD at talk address Ø, secondary address Ø through interface number Ø to a file OPEN as PLUN 2. The second example (line 21Ø) logs N waveforms to a disk file OPEN as PLUN DF in FAST mode from a 7912AD at talk address TA, secondary address SA through interface number 2.

Some additional notes. ADLOG will accept waveforms from the 7912AD in all legal sweep modes, including single-sweep. Remember to set the interface time-out value (with SIFTO) to a value long enough to cover the time between sweeps when operating in singe-sweep mode.

If a time-out or other error should occur during logging, the ADLOG command will log all the data received prior to the error. A CLOSE or END statement will close the file normally. Note that, due to the error condition, only part of the last waveform logged may have been written to the disk. When reading that waveform (with READ) a P11 or P3 error will occur as the end-of-file is reached. All previous waveforms will be correct, however.

APPENDIX A

THEORY OF GEOMETRY MAPPING

IN DISTORTING CRTS

To provide a better understanding and appreciation for the algorithm used in the MAPAD command, a brief overview of the mathematical foundations for geometry correction is provided. First, we will talk about the mapping effected by a distorting CRT in yielding distorted data from an ideal coordinate system. Then we will discuss the inverse process that must be performed by the geometry correction routine -- namely that of producing ideal data from a distorted coordinate system.

Mappings Effected by Distorting CRTs. Geometry distortions in CRTs can be thought of as essentially transformations (continuous, differentiable ones) of a portion of a planar surface -- the target area. Mathematically then, they can be represented as vector valued functions of vector variables. In particular, they represent a function or mapping of one set of ordered pairs (representing horizontals and verticals) into another set of ordered pairs. This can be thought of as:

$$(x,y) \xrightarrow{\ f\ } (u,v)$$

or simply $(u,v) = f(x,y)$, where (x,y) is the ideal point and (u,v) is the actual point. A "perfect" CRT (if there is such a device) takes a planar surface (idealized target area) and maps it exactly into itself. That is, it does nothing more than perform the identity transformation:

$$(x,y) \longrightarrow (x,y)$$

Assuming that $f(x,y)$ is a correct mathematical model of the geometry distortions effected by the 7912AD scan converter, the errors or deviations from the ideal case are thus:

$$e(x,y) = (x,y) - f(x,y) \qquad (1)$$

In other words, the error term, $e(x,y)$ is simply the difference of the ideal coordinates (x,y) and the "distorting" function, $f(x,y)$.

Since any vector valued function can be expressed in terms of its real-valued component functions, we can express e(x,y) and f(x,y) as:

$$f(x,y) = (f_1(x,y) \ , \ f_2(x,y)) \qquad (2)$$

$$e(x,y) = (e_1(x,y) \ , \ e_2(x,y)) \qquad (3)$$

By adding the above functions and noting that (x,y) = f(x,y) + e(x,y) we have:

$$(x,y) = (f_1(x,y) + e_1(x,y) \ , \ f_x(x,y) + e_2(x,y)) \qquad (4)$$

By separating the x and y components and transposing terms, we get the x and y component error terms:

$$e_1(x,y) = x - f_1(x,y) \qquad (5)$$

$$e_2(x,y) = y - f_2(x,y) \qquad (6)$$

both of which are real-valued functions.

The x,y component distortion functions $f_1(x,y)$ and $f_2(x,y)$ can be measured with respect to a rectangular array of points (x_i,y_i) where $x_i = x_o + ih$ for $i = 0,1,2,3,...m$, and where $y_j = y_o + jk$ for $j = 0,1,2,3,...n$. The symbols h and k are horizontal and vertical spacing constants for a uniform array of points -- such as produced by the computer generated line graticule. With respect to the 7912AD, the points (x_i,y_j) could be the verticies of the computer generated line graticule and the points $(f_1(x_i,y_j),f_2(x_i,y_j))$ could be the major verticies of the 7912AD dot graticule. These data provide a way to estimate the error for any point within the 7912AD dot graticule, since we can interpolate the error factor at any point by knowing the errors at each major vertice of the 7912AD dot graticule.

Our problem has now been reduced to finding an interpolating polynomial for each of the rectangles, or preferably triangles, defined by the dot grid. Because of the simpler mathematics involved, we will divide the grid into triangles as shown in Fig. A-1.

For either the case of rectangles or triangles, the uniform grid implies the existence of a unique interpolating polynomial that is easily constructed. Several approaches are available for the determination of

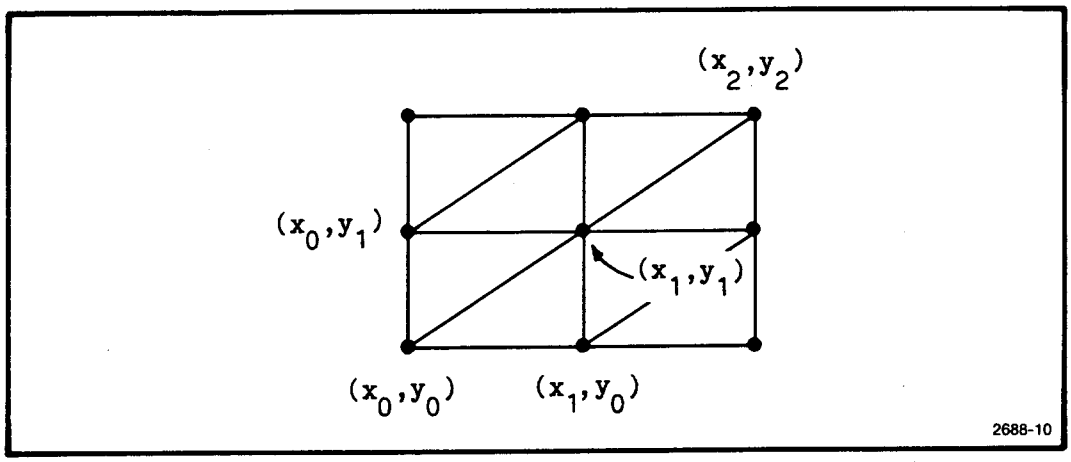


Fig. A-1. Ideal graticule coordinates.

such a polynomial, but the easiest is the direct approach.[1] in this approach, you assume the general form of the approximation and determine the coefficients of the polynomial by requiring that it be exact for a set of relevant functions. For example, consider the x-component error data:

$$e_1(x_0,y_0) = x_0 - f_1(x_0,y_0) \quad (7)$$

$$e_2(x_1,y_0) = x_1 - f_1(x_1,y_0) \quad (8)$$

$$e_3(x_1,y_1) = x_1 - f_1(x_1,y_1) \quad (9)$$

where for convenience the points (x_0,y_0), (x_1,y_0), and (x_1,y_1) have been selected. For errors $e_1(x,y)$, where (x,y) is on or within the triangle defined by the three given points, one logical approximation would involve a formula of the type:

$$e_1(x,y) = ae_1(x_0,y_0) + be_1(x_1,y_0) + ce_1(x_1,y_1) \quad (10)$$

where a, b, and c are in general functions of x and y. Requiring that the equation be exact if the error function is of the form 1, x, or y, leads to:

$$1 = a + b + c$$
$$x = ax_0 + bx_1 + cx_1$$
$$y = ay_0 + by_0 + cy_1$$

[1]R.W. Hamming. **Numerical Methods for Scientists and Engineers.** 2nd ed. McGraw-Hill, 1973, Chapter 15.

These equations can also be expressed by the matrix:

$$\begin{bmatrix} 1 \\ x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ x_0 & x_1 & x_1 \\ y_0 & y_0 & y_1 \end{bmatrix} * \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

While this system is not difficult to solve for a, b, and c, judicious selection of the (x_i,y_i) term makes it trivial. For example, let $x_o = y_o = 0$ and let $x_1 = y_1 = 1$. The system of equations is then:

$$1 = a + b + c$$
$$x = b + c$$
$$y = c$$

which leads to c = y, b = x-y, and a = 1-x, when we solve for a, b, and c simultaneously. (We can also arrive at these values by matrix algebra.)

By substituting the values of a, b, and c into equation 10, we arrive at the following equation:

$$e(x,y) = (1-x) * e_1(x_0,y_0) + (x-y) * e_1(x_1,y_0) + y * e_1(x_1,y_1) \quad (11)$$

This equation describes the horizontal error component associated with a point (x,y) located inside the triangle whose verticies are $(x_0,y_0) = (0,0)$, $(x_1,y_0) = (1,0)$, and $(x_1,y_1) = (1,1)$.

The vertical error component $e_2(x,y)$ is found by substituting the weight terms a, b, and c into the equation:

$$e_2(x,y) = ae_2(x_0,y_0) + be_2(x_1,y_0) + ce_2(x_1,y_1) \quad (12)$$

where:

$$e_2(x_0,y_0) = y_0 - f_2(x_0,y_0) \quad (13)$$
$$e_2(x_1,y_0) = y_0 - f_2(x_1,y_0) \quad (14)$$
$$e_2(x_1,y_1) = y_1 - f_2(x_1,y_1) \quad (15)$$

Computer Correction of Distortion in Acquired Waveform Data. While the above approach is useful for estimating CRT distortions, it is not immediately applicable to the correction of these distortions in acquired waveform data. Whereas before we were given an undistorted point and asked to match the distorted point f(x,y) or error e(x,y) at that point, we are now faced with the inverse problem. That is, we are presented with a distorted point (u,v) = f(x,y) and asked where the ideal (undistorted) point resides. Thus our problem is that of finding the inverse function $(x,y) = f^{-1}(u,v)$ or the inverse error function $e^{-1}(u,v)$.

One approach to the above problem is, after having modeled f(x,y) or e(x,y) and being given a distorted point (u,v), to find an x and y such that:

$$e(x,y) = (x,y) - (u,v)$$

However, this approach involves some tedious iterative techniques and thus a more direct approach is taken. This alternative approach is to take the measured, distorted coordinates of the 7912AD dot graticule, (u_i,v_i), as the independent variables and to model f^{-1} or e^{-1}. While it is true that a nonuniform grid does not always permit the determination of a unique interpolating polynomial, experience has shown that the coordinates of the 7912AD dot graticule are spaced uniformly enough so as to permit the construction of such a polynomial.

To determine the interpolating polynomial for the "distorted" grid, assume that grid points are (u_i,v_i), i=a, b, ...1, indexed in some reasonable fashion. Then the formula of the interpolating polynomial for a triangle proceeds in a similar manner as for the case of the nondistorted grid, described previously. For example, the polynomial associated with the horizontal error component for a point (u,v) in the lower right triangle of Fig. A-2 has the general form:

$$e_1(u,v) = ae_1(u_0,v_0) + be_1(u_1,v_1) + ce_1(u_2,v_2) \qquad (16)$$

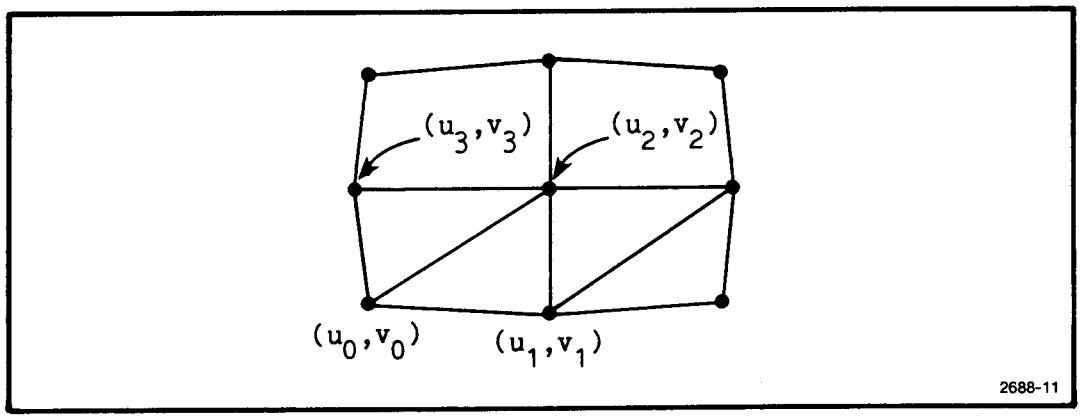


Fig. A-2. Distorted graticule coordinates.

Requiring the three point formula to be exact for the functions 1, u, and v leads to the equations:

$$1 = a + b + c$$
$$u = au_0 + bu_1 + cu_2$$
$$v = av_0 + bv_1 + cv_2$$

The matrix associated with these equations is:

$$\begin{bmatrix} 1 \\ u \\ v \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ u_0 & u_1 & u_2 \\ v_0 & v_1 & v_2 \end{bmatrix} * \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

To solve for the weight terms a, b, and c, we need to find the inverse matrix of the 3 by 3 matrix. After a fair amount of "mathematical gymnastics" this turns out to be:

$$\frac{1}{(u_1-u_0)(v_2-v_0)-(v_1-v_0)(u_2-u_0)} * \begin{bmatrix} u_1v_2-v_1u_2 & -(v_2-v_1) & (u_2-u_1) \\ -(u_0v_2-v_0u_2) & v_2-v_0 & -(u_2-u_0) \\ u_0v_1-v_0u_1 & -(v_1-v_0) & u_1-u_0 \end{bmatrix}$$

Using this inverse matrix leads to the following values for the weight terms:

$$a = \frac{u_1v_2 - v_1u_2 - u(v_2-v_1) + v(u_2-u_1)}{\Delta}$$

$$b = \frac{-(u_0v_2-v_0u_2) + u(v_2-v_0) - v(u_2-u_0)}{\Delta}$$

$$c = \frac{u_0v_1 - v_0u_1 - u(v_1-v_0) + v(u_1-u_0)}{\Delta}$$

where $\Delta = (u_1-u_0)(v_2-v_0) - (v_1-v_0)(u_2-u_0)$

Once we have solved for the weight terms a, b, and c, we can solve for the horizontal error component $e_1(u,v)$ by substituting these values into equation 16.

The vertical error component $e_2(u,v)$ is found by substituting the weight terms into the equation:

$$e_2(u,v) = ae_2(u_0,v_0) + be_2(u_1,v_1) + ce_2(u_2,v_2) \qquad (17)$$

where:

$$e_2(x_0,y_0) = y_0 - f_2(u_0,v_0) \qquad (18)$$
$$e_2(x_1,y_1) = y_1 - f_2(u_1,v_1) \qquad (19)$$
$$e_2(x_2,y_2) = y_2 - f_2(u_2,v_2) \qquad (20)$$

MANUAL CHANGE INFORMATION

At Tektronix, we continually strive to keep up with latest electronic developments by adding circuit and component improvements to our instruments as soon as they are developed and tested.

Sometimes, due to printing and shipping requirements, we can't get these changes immediately into printed manuals. Hence, your manual may contain new change information on following pages.

A single change may affect several sections. Since the change information sheets are carried in the manual until all changes are permanently entered, some duplication may occur. If no such change pages appear following this page, your manual is correct as printed.

Tektronix®
COMMITTED TO EXCELLENCE

MANUAL CHANGE INFORMATION

Change Reference: C1/581 Date: 4/27/81

Product: TEK SPS BASIC V02/V02XM 7912AD Commands Package CP57006/CP57506 070-2745-01

CHANGE	DESCRIPTION

TEK SPS BASIC VØ2 7912AD Commands

MANUAL CHANGE INFORMATION

... APRIL 17, 1981 ...

REV B, APR 1981

Page 21 --

Text Insertition: Add line 165 between 160 and 170.

```
165 FD=CW\IF FD=-1 THEN GOTO 2ØØ
```

Page 22 --

Text Correction: Correct line 11ØØ to read:

```
11ØØ IF FD<>-1 THEN REJECT QQ,P,DF
```

Text Addition: Insert line 2Ø95 between 2Ø9Ø and 21ØØ.

```
2Ø95 IF CW=-1 THEN GOTO 212Ø
```

Page 33 --

Text Addition: Insert line 165 between 16Ø and 17Ø.

```
165 FD=CW\IF FD=-1 THEN GOTO 2ØØ
```

Page 34 --

Text Correction: Correct line 11ØØ to read:

```
11ØØ IF FD<>-1 THEN REJECT QQ,P,DF
```

Text Addition: Insert line 2Ø95 between 2Ø9Ø and 21ØØ.

```
2Ø95 IF CW=-1 THEN GOTO 212Ø
```

Scans
By
Artek Media

Artek Media
1042 Plummer Cir. SW
Rochester, MN 55902

www.artekmedia.com

"High resolution scans of obsolete technical manuals"

If your looking for a quality scanned technical manual in PDF format please visit our WEB site at www.artekmedia.com or drop us an email at manuals@artekmedia.com and we will be happy to email you a current list of the manuals we have available.

If you don't see the manual you need on the list drop us a line anyway we may still be able to point you to other sources. If you have an existing manual you would like scanned please write for details, This can often be done very reasonably in consideration for adding your manual to our library.

Typically the scans in our manuals are done as follows;

1) Typed text pages are typically scanned in black and white at 300 dpi.
2) Photo pages are typically scanned in gray scale mode at 600 dpi
3) Schematic diagram pages are typically scanned in black and white at 600 dpi unless the original manual had colored high lighting (as is the case for some 70's vintage Tektronix manuals).

If you purchased this manual from us (typically through our Ebay name of ArtekMedia) thank you very much. If you received this from a well-meaning "friend" for free we would appreciate your treating this much like you would "share ware". By that we mean a donation of at least $5-10 per manual is appreciated in recognition of the time (a manual can take as much as 40 hours to reproduce, book, link etc.), energy and quality of effort that went into preserving this manual. Donations via PayPal go to: manuals@artekmedia.com or can be mailed to us the address above.

Thanks

Dave & Lynn Henderson
Artek Media