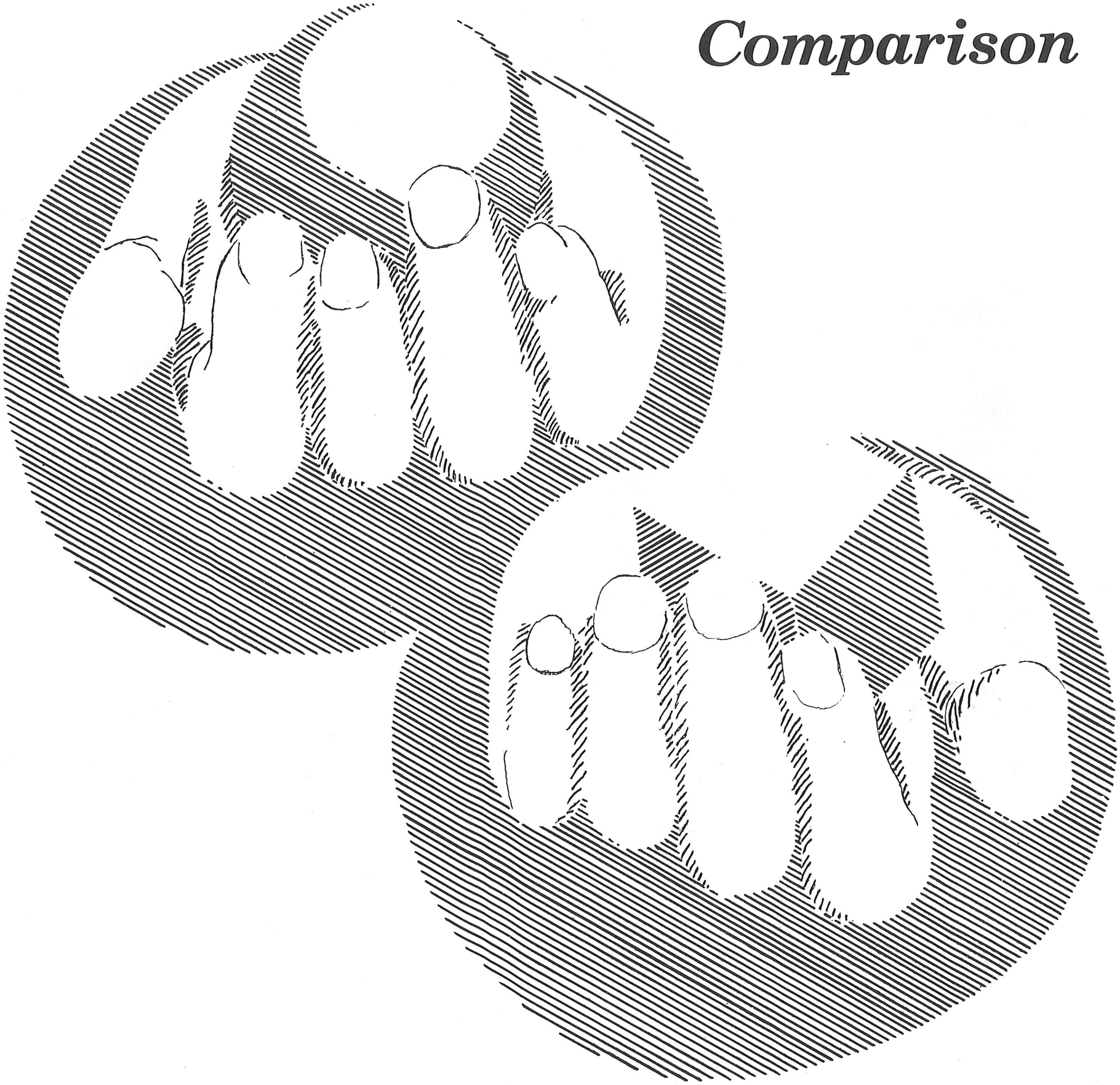
Vol. 4 No. 4 FALL/WINTER 1979

Tektronix®
COMMITTED TO EXCELLENCE

HANDSHAKE

Newsletter of the Signal Processing Systems Users Group

Measurements by Comparison



Tektronix® COMMITTED TO EXCELLENCE

HANDSHAKE

Newsletter of the Signal Processing Systems Users Group

Table of contents

Managing Editor: Bob Ramirez
Edited by: Bob Ramirez
Graphics by: Bernard Chalumeau and Joann Crook

HANDSHAKE is published quarterly by the SPS Documentation Group. Permission to reprint material in this publication may be obtained by writing to:

HANDSHAKE Editor
Group 157 (94-384)
Tektronix, Inc.
P.O. Box 500
Beaverton, Oregon 97077

HANDSHAKE is provided free of charge by Tektronix, Inc., as a forum for people interested in digital signal processing. As a free forum, statements and opinions of the authors should be taken as just that--statements and opinions of the individual authors. Material published in **HANDSHAKE** should not be taken as or interpreted as statement of Tektronix policy or opinion unless specifically stated to be such.

Also, neither **HANDSHAKE** nor Tektronix, Inc., can be held responsible for errors in **HANDSHAKE** or the effects of these errors. The material in **HANDSHAKE** comes from a wide variety of sources, and, although the material is edited and believed to be correct, the accuracy of the source material cannot be fully guaranteed. Nor can **HANDSHAKE** or Tektronix, Inc., make guarantees against typographical or human errors, and accordingly no responsbility is assumed to any person using the material published in **HANDSHAKE**.

Programming poster

In the center of this issue of HANDSHAKE, you'll find a removable wall chart. It is designed to assist you in programming. One side contains IEEE 488 (GPIB) and ASCII codes as well as octal, hexadecimal, and decimal equivalents. The other side contains programming information for the TEKTRONIX 7912AD Programmable Digitizer.

Additional programming charts can be obtained by ordering extra copies of the FALL/WINTER 1979 HANDSHAKE via the HANDSHAKE Reply Card.

Dropped bits

On page 14 of the last issue of HANDSHAKE (Spring/Summer '79) an error occurred in line 140 of the histogram routine. Seems a bit got lost in our automated typesetter and a "+" became an "M". The corrected program line should read:

```
140 LET B(X(I))=B(X(I))+1
```

Reader survey

Did you return the survey card included in the last issue of HANDSHAKE? We're still interested in hearing from you because your answers help make HANDSHAKE a better publication.

To those of you who have already responded, our thanks.

The Editor

Printed in U.S.A.

Literature available from Tektronix

Back issues of HANDSHAKE available

Each issue of HANDSHAKE is overprinted by several thousand copies. Still, numerous requests for back issues have depleted much of the shelf stock. So, if we have been unable to completely fill your request for back issues, we apologize.

Recently, however, a number of past issues have been reprinted to meet continued strong interest in the topics covered by them. The issues currently in good supply are listed below and can be ordered via the HANDSHAKE reply card bound into this issue of HANDSHAKE.

Spring 1976 No. 3

Devoted to the fast Fourier transform, this issue contains a short tutorial on the FFT, two articles dealing with FFT leakage, and a detailed article on mechanical signature analysis.

Summer 1976 No. 4

This issue deals with metals analysis with one feature article on laser spectroscopy, another on ultrasonic testing of die-cast parts, and still another feature-length article on fatigue testing.

Summer 1977 Vol. 2, No. 4

"Focus on Transient Analysis" is the theme of this issue. EMP testing, fluorescence decay analysis, and time-of-flight analysis are the major topics.

Fall 1977 Vol. 3, No. 1

ATE (automatic test equipment) is the major topic, with both tutorial and instrument interfacing specifics provided.

Winter 77-78 Vol. 3, No. 2

If you need some basic information on GPIB (IEEE 488) operation, the first article in this issue should be of interest. Another major article discusses automation of swept RF measurements.

Spring 1978 Vol. 3, No. 3

Interested in spectrum analysis? Spectrum analyzer basics and methods of automating spectrum analyzer measurements are presented. Also, a short article covers EMI measurements with a spectrum analyzer and a signal processing system.

Summer 1978 Vol. 3, No. 4

Optical fiber evaluation is the featured topic of this issue.

Winter 78-79 Vol. 4, No. 2

This issue, with a hefty 10-page tutorial on signal processing basics, is directed toward newcomers to the field of digital signal processing.

Spring-Summer 1979 Vol. 4, No. 3

If you have to measure pulse parameters very often, better get a copy of this issue for your files. There are a full 12 pages of information on basic techniques of pulse analysis.

Article reprint available

"Digitizing-oscilloscope systems simplify transmission measurements," which appeared in **EDN** June 5, 1979, is now available in reprint form. For your copy, check the appropriate box on the HANDSHAKE reply card.

New Technical Note ready

"A Review of Error Reduction Methods for Optical Fiber Transmission Measurements" is now ready for distribution. Send in the HANDSHAKE reply card for your copy of this new Technical Note.

Limit testing, linearity, aberrations,

and other measurements by comparison

Making measurements by comparison can often be the most efficient approach to test and evaluation. You have a known item or quantity. It's typical of what you want or expect—a standard. Other quantities or items can be compared quickly and easily to this chosen standard to see how well they match. Then, depending upon requirements, the comparison can provide anything from reams of research data to a simple GO/NO-GO decision for production testing.

This isn't a startlingly new concept. But a digital signal processing system can add some startlingly new twists to it...twists that can make comparison testing easier, faster, and more revealing.

For example, with a digital system, you only have to acquire your standard waveform once and store it. From then on, you can call it up from memory as you need it for comparisons. That's easier and faster than having to set up or generate the standard each time you make a test. Beyond that, the comparisons can be made in greater detail. Instead of trying to visually resolve small differences on a CRT, the test waveform can be subtracted from the standard to give you just the difference as a waveform. This difference waveform can then be analyzed further for specific details. In other cases, you can generate your own standard with software, an ideal pulse for example. Another possibility is developing limit waveforms and checking the test waveforms to see if they violate the limits. And finally, with digital signal processing, there is the opportunity to routinely apply techniques not readily available with analog instrumentation. Comparison by correlation is just one example of the latter.

The possibilities are virtually limitless. However, a few examples can provide most of the basics as well as demonstrate some of the new twists offered by digital signal processing.

Signal processing makes the difference

Difference testing is simply subtracting one waveform from another. Usually, one of the waveforms is the standard, the item you want to compare against. The other waveform is the one from which you hope to derive relative information about the waveform itself or something that it represents. The fact that there is a difference between waveforms, or the amount of difference, or some attribute of the difference waveform can reveal important information—information like pulse aberrations, linearity, device quality.

In analog testing, difference waveforms can be obtained by using a differential amplifier or a dual-channel scope with INVERT and ADD functions. However, there are a number of constraints that limit strictly analog implementations:

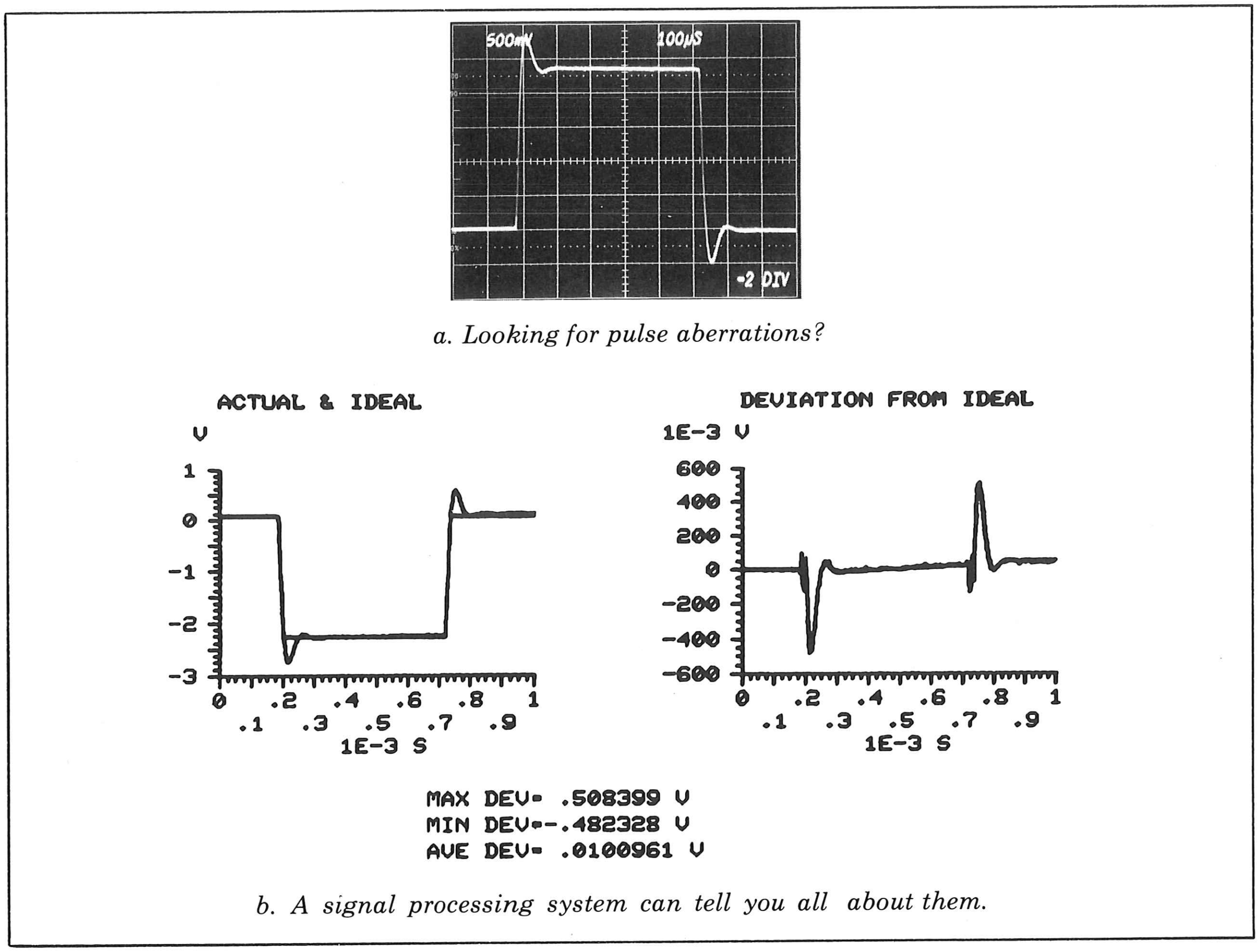
- The two waveforms have to be acquired simultaneously.
- What happens if you don't have a standard waveform?
- Often, the difference waveform must be analyzed further to get the information of interest.
- The signal environment—noise, trigger jitter, delay or phase shift—can often render the difference waveform useless.

Digital signal processing techniques shrink these common hurdles into things that can easily be taken in stride.

Look at the problem of not having a standard waveform for instance. This is common in studying pulse aberrations (Fig. 1a). Where do you get an ideal pulse for comparison? More often than not, the standard is a mental image, and you are forced to pick information from an oscilloscope display by eye. When the pulse you wish to study is nearly ideal itself, this becomes particularly difficult—unless you employ digital techniques.

An ideal solution for pulses. The major steps in investigating pulse aberrations with a signal processing system are blocked out in Fig. 2.

Going from the top of the diagram, the pulse is acquired and digitized first. Generally, this is accompanied by signal averaging to improve signal-to-noise ratio (SNR) and minimize the



a. *Looking for pulse aberrations?*

```
MAX DEV= .508399 V
MIN DEV=-.482328 V
AVE DEV= .0100961 V
```

b. *A signal processing system can tell you all about them.*

Fig. 1. *One of the advantages of digital signal processing is that you can generate ideal waveforms as standards for comparison. In the case of measuring aberrations, the square pulse in question is digitized and then analyzed for its basic parameters—top, base, and 10% and 90% points. An ideal square pulse is then generated with these parameters. The difference between this ideal and the pulse under test is an aberrations waveform.*

effects of random jitter. This digitized and "cleaned-up" pulse is stored in memory. Then analysis routines are applied to extract the basic pulse parameters. These are the 0% and 100% levels and the locations of the 10% and 90% points on the pulse transitions. (Techniques and software routines for extracting this type of pulse information were discussed in the Spring-Summer 1979 HANDSHAKE. A copy of that issue can be obtained by checking the appropriate box on the HANDSHAKE Reply Card provided with this issue.)

With the pulse's basic parameters known, an ideal version of the pulse can be generated by software. In the case of Fig. 1b, the ideal was taken to be a square pulse. It was generated by connecting straight-line segments through the 0% and 100% levels for the pulse base and top and through the 10% and 90% points on the transitions.

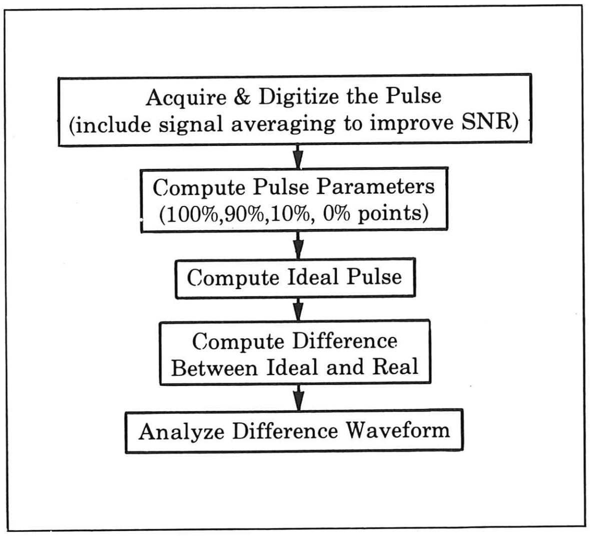

Acquire & Digitize the Pulse
(include signal averaging to improve SNR)
Compute Pulse Parameters
(100%,90%,10%, 0% points)
Compute Ideal Pulse
Compute Difference
Between Ideal and Real
Analyze Difference Waveform

Fig. 2. *Block diagram of the process used to get the information in Fig. 1b.*

A routine for doing this is provided as a separate part of this article. It generates an ideal square pulse; however, the ideal pulse could just as well have been generated as a Nyquist or Gaussian pulse, or whatever else you need. It's merely a matter of writing the appropriate routine.

Once the ideal pulse is generated, a simple subtraction of waveform arrays provides the difference waveform (Fig. 1b). This can be analyzed for further specifics (maximum, minimum, average deviations) as indicated in Fig. 1b.

Notice that the above process requires acquisition of only one signal—the pulse to be analyzed. There's no need for additional calibration signals or lengthy calibration procedures. The standard for comparison is generated by software. And it's ideal—no noise, no jitter, no delay to worry about.

Looking at linearity. Measuring linearity is similar to measuring pulse aberrations in some basic respects. The process uses a standard or reference waveform and computes the difference between it and the waveform in question. That's the basic idea. There are some variations, however. For one, you can measure either the linearity of a ramp or the linearity of a device. For another, you can choose to generate an ideal ramp as the standard, or the standard can be an actual ramp signal that is acquired, digitized, and stored in the signal processing system.

Whatever the variation, digital signal processing techniques can handle it. To give you an idea of how, let's look first at the most basic approach—testing a ramp for its linearity.

The ramp in question could be from anything, the output of some mechanical device, a sawtooth used as horizontal drive for a cathode ray tube, anything. Determining its linearity requires two waveforms, the ramp itself and an ideal ramp to compare against. What is computed from the two sets of ramp data is normalized point-to-point linearity, and it can be defined as follows:

$v_i(t)$ = the reference ramp against which linearity is computed

V_i = the estimated peak-to-peak excursion of the reference ramp

$v_o(t)$ = the ramp whose linearity is being computed

V_o = the estimated peak-to-peak excursion of the ramp being analyzed

Using these terms, linearity in terms of percent of the full-scale value for $v_o(t)$ can be computed as follows:

$$\mathrm{Lin}(t) = 100\left(\frac{v_o(t)}{V_o} - \frac{v_i(t)}{V_i}\right)$$

$$= A_0 + A_1 t + A_2 t^2 + A_3 t^3 + \ldots + A_n t^n$$

A_0 and A_1 are the offset and linear terms. They are zero in the following computations. The other terms—$A_2, A_3, \ldots, A_n$—are the nonlinear terms and are nonzero when either $v_o(t)$ or $v_i(t)$ is nonlinear.

A TEK SPS BASIC program for generating an ideal ramp and making the normalized point-to-point linearity computation with the above equation is listed in Fig. 3a. The first segment of the program—lines 10 through 75—comprises the ramp acquisition and instrument control statements. The form and order of these statements depend upon the type of waveform digitizer used; however, the routine should always be written so that the ramp is acquired into TEK SPS BASIC waveform WB.

With the ramp in question acquired and stored in array B of waveform WB, the linearity analysis portion of the program can be called into action. This exists as lines 100 through 170 in Fig. 3a. Lines 105 and 110 define two more waveform storage areas, WA and WC, for the ideal ramp and the linearity results. The ideal ramp is generated in lines 120 and 125. Note that it doesn't need to have the same slope as the ramp being tested. It just has to be linear. Following generation of the ideal ramp, both ramps have their mean values removed (line 135). This removes offset (sets the A_0 term in the defining equation to zero). The next step, line 140, is to compute the peak-to-peak value of each ramp. Then, in line 145, normalized point-to-point linearity is computed. The remaining lines of the program deal with handling waveform units and graphing the linearity error waveform.

A test example is shown in Fig. 3b. The oscilloscope display there shows a ramp signal that appears to be quite linear. It was acquired with signal averaging to reduce additive random noise and stored in the signal processing system. Expanded segments of this stored ramp and the ideal ramp are shown in Fig. 3c. This detailed view (1/50th of the ramp duration) does reveal some minor variations in the acquired ramp. The actual linearity error for the entire ramp is shown in Fig. 3d. Note that the vertical scaling is in millipercent, so the linearity error is really not large.

continued on page 8

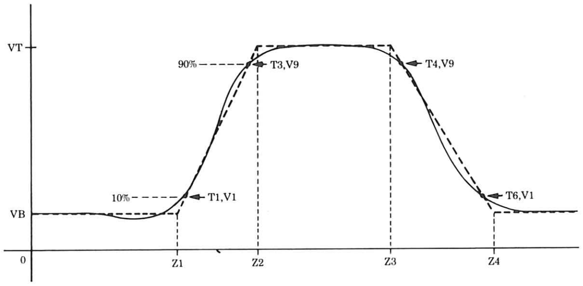
Generating the ideal pulse

To measure a pulse's aberrations or deviations from ideal, you need to have an ideal for comparison. The following TEK SPS BASIC V02-02 program generates such an ideal. Its ideal square pulse passes through specified top and base levels and specified 10% and 90% points on the transitions. These specifications, indicated in the accompanying figure, are determined by a pulse analysis program such as discussed in an article previously published in HANDSHAKE ("Some useful approaches to pulse analysis," Spring-Summer 1979 HANDSHAKE). It is also assumed in the following program that the acquired pulse resides in TEK SPS BASIC waveform WA—WAVEFORM WA IS A(511),HA,HA$,VA$, where A is the data array, HA is the digital sampling interval, HA$ contains the horizontal units, and VA$ contains vertical units. The ideal pulse is generated in waveform WB—WAVEFORM WB IS B(511),HB,HB$,VB$.

```
600 REM COMPUTE DEVIATION FROM IDEAL SQUARE PULSE
605 REM COMPUTE TRANSITION END POINTS
610 LET MR=(V9-V1)/(T3-T1)
615 LET MF=(V9-V1)/(T4-T6)
620 LET Z1=(VB-V1+T1*MR)/(MR*HA)\LET Z1=ITP(Z1+.5)
625 LET Z2=(VT-V1+T1*MR)/(MR*HA)\LET Z2=ITP(Z2+.5)
630 LET Z3=(VT-V1+T6*MF)/(MF*HA)\LET Z3=ITP(Z3+.5)
635 LET Z4=(VB-V1+T6*MF)/(MF*HA)\LET Z4=ITP(Z4+.5)
640 REM SET TOP AND BASE ZONES
645 LET B(0:Z1)=VB\LET B(Z4:511)=VB
650 LET B(Z2:Z3)=VT
655 REM GENERATE TRANSITIONS
660 FOR I=(Z1+1) TO Z2
665 LET B(I)=B(I-1)+(VT-VB)/(Z2-Z1)
670 NEXT I
675 FOR I=(Z3+1) TO Z4
680 LET B(I)=B(I-1)-(VT-VB)/(Z4-Z3)
685 NEXT I
690 REM COMPUTE & DISPLAY DEVIATIONS
695 LET HB=HA\LET HB$=HA$\LET VB$=VA$
700 WAVEFORM WC IS C(511),HC,HC$,VC$
705 LET WC=WA-WB\PAGE
710 VIEWPORT 100,400,500,700\SETGR GRAT 2,2,VIEW
715 GRAPH WA,WB
720 VIEWPORT 600,900,500,700\SETGR GRAT 2,2,VIEW
725 GRAPH WC
730 SMOVE 100,760\PRINT "ACTUAL & IDEAL"
735 SMOVE 600,760\PRINT "DEVIATION FROM IDEAL"
740 LET S$=" "
745 SMOVE 300,375\PRINT "MAX DEV=";MAX(C);S$&VC$
750 SMOVE 300,350\PRINT "MIN DEV=";MIN(C);S$&VC$
755 SMOVE 300,325\PRINT "AVE DEV=";MEA(C);S$&VC$
760 END
```

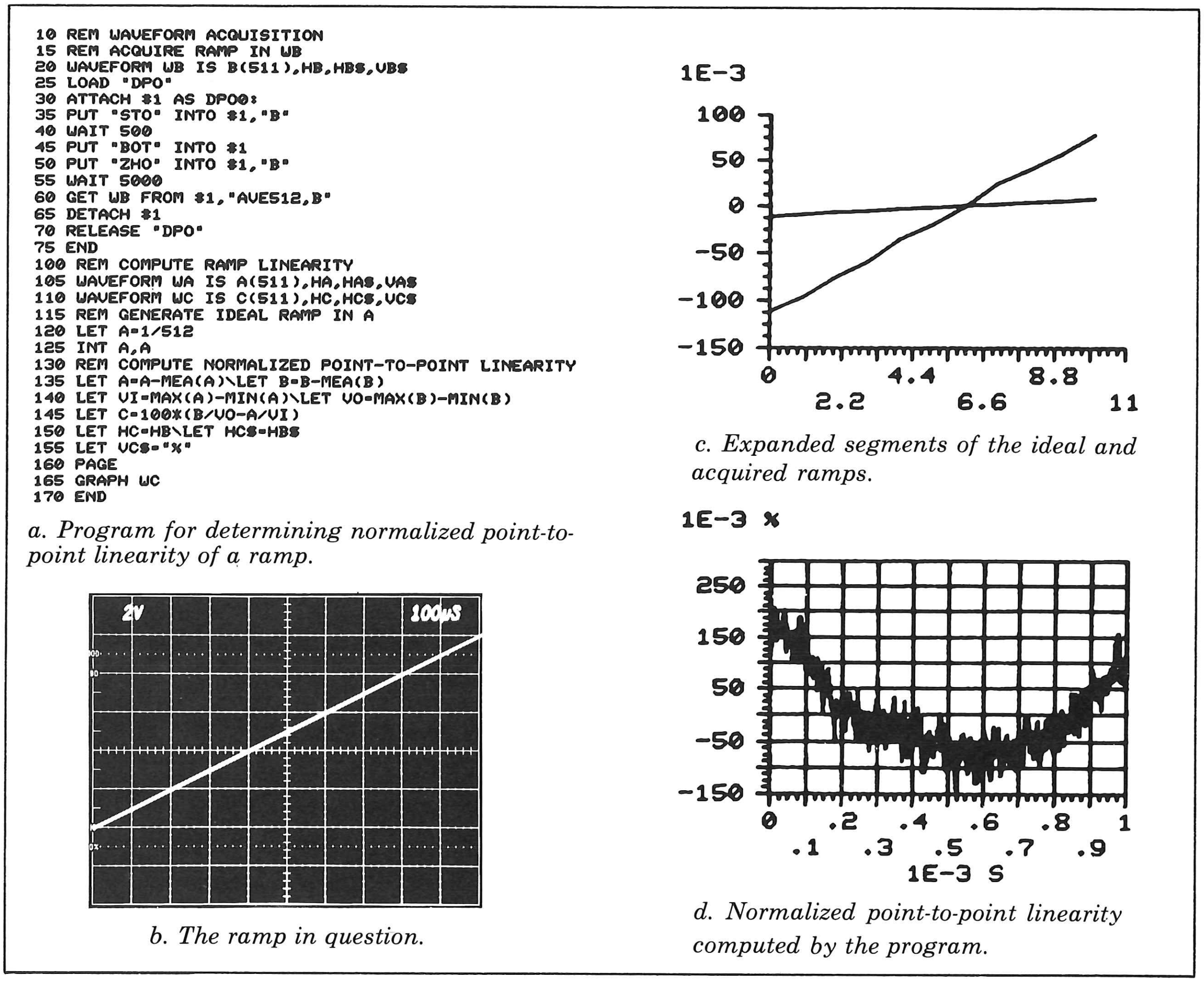
Generation of the ideal pulse begins with computing transition slopes in lines 610 and 615. These are used in straight-line equations (y=mx+b) to compute the transition end points, the points where the transitions intersect the top and base levels. These end points are designated Z1, Z2, Z3, and Z4 and are computed in lines 620 through 635. Note that the computations are floating-point and, therefore, must be converted to integers to coincide with the integer indexing of the array.

The transition end points, along with the beginning and ending points of the array, mark off five zones. Three of these zones are used in lines 645 and 650 to contain the top and base levels in array B. The two remaining zones contain the transitions which are generated by FOR loops in lines 660 through 685. This completes generation of the ideal pulse.



Specification of variables used in generating an ideal square pulse (dashed) to fit parameters of an acquired pulse (solid).

Comparison of the ideal pulse to the acquired pulse is done in lines 695 to 705. Lines 695 and 700 set up the arrays, and the actual comparison takes place in line 705 where the ideal is subtracted from the real. The result, deviations from ideal, is stored in waveform WC. Following this, lines 710 through 740 set up graphics for displaying the results in the manner of Fig. 1b of the main article, and some simple deviation parameters are computed and printed out by lines 745, 750, and 755.



```
10 REM WAVEFORM ACQUISITION
15 REM ACQUIRE RAMP IN WB
20 WAVEFORM WB IS B(511),HB,HB$,VB$
25 LOAD "DPO"
30 ATTACH $1 AS DPO0:
35 PUT "STO" INTO $1,"B"
40 WAIT 500
45 PUT "BOT" INTO $1
50 PUT "ZHO" INTO $1,"B"
55 WAIT 5000
60 GET WB FROM $1,"AVE512,B"
65 DETACH $1
70 RELEASE "DPO"
75 END
100 REM COMPUTE RAMP LINEARITY
105 WAVEFORM WA IS A(511),HA,HA$,VA$
110 WAVEFORM WC IS C(511),HC,HC$,VC$
115 REM GENERATE IDEAL RAMP IN A
120 LET A=1/512
125 INT A,A
130 REM COMPUTE NORMALIZED POINT-TO-POINT LINEARITY
135 LET A=A-MEA(A)\LET B=B-MEA(B)
140 LET VI=MAX(A)-MIN(A)\LET VO=MAX(B)-MIN(B)
145 LET C=100*(B/VO-A/VI)
150 LET HC=HB\LET HC$=HB$
155 LET VC$="%"
160 PAGE
165 GRAPH WC
170 END
```

a. Program for determining normalized point-to-point linearity of a ramp.

b. The ramp in question.

c. Expanded segments of the ideal and acquired ramps.

d. Normalized point-to-point linearity computed by the program.

Fig. 3. *Testing a ramp for linearity by comparing it to an ideal ramp generated by software.*

Another case of linearity testing is shown in Fig. 4. Here a ramp signal is used to test the linearity of a device. The input ramp swing is chosen to traverse the amplitude limits over which linearity is to be tested. Then, the resulting output ramp is compared to the input to determine if any nonlinearity has been injected by the device under test. If the device is linear, the output ramp is a duplicate of the input except for gain and offset factors.

Normally in analog testing, both the input and output ramps must be acquired simultaneously in a dual-channel, differential situation to see the difference between the two ramps. Signal delay, noise, different channel offsets, and so forth must be taken care of before the difference is formed. If they aren't, these noncommon terms will erroneously be included as part of the difference waveform.

With a signal processing system, simultaneous acquisition is not necessary as long as the input ramp is periodic: for example, a sawtooth waveform or the zero-crossing portion of a clipped sinusoid (Fig. 5). When this is the case, a signal processing system can be used to first acquire and store the input ramp and then the output ramp. Also, signal processing includes the benefit of signal averaging to improve the SNR of each

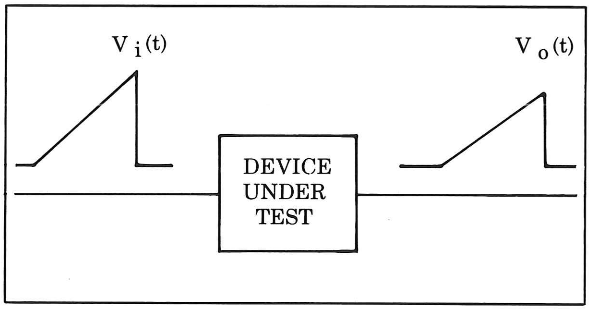


Fig. 4. *Testing device linearity. For a linear device, $v_o(t)=Gv_i(t)$, where G is a constant gain term.*

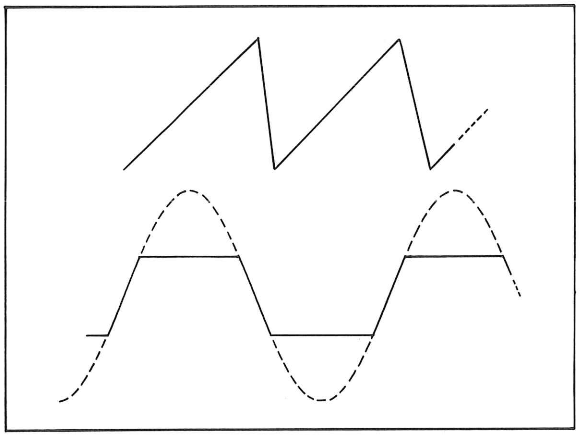


Fig. 5. *Repetitive ramps for linearity testing include sawtooth waves and clipped sinusoids.*

ramp. And, since acquisition is done through a single channel, any nonlinearity in that channel adds to each ramp in the same degree and will cancel in the differencing. It should also be noted that the input ramp need not be exactly linear, either, since its nonlinearities will also appear in the output ramp. The point-to-point differencing sets all of these common terms to zero.

A program for computing device linearity error can be written in much the same format as the program listed in Fig. 3a. The difference, of course, will be that two waveforms are acquired—the input waveform in WA as the reference and the output waveform in WB. And, since the reference is an acquired signal, there is no need to generate an ideal ramp (lines 120 and 125 can be deleted). Also, you'll probably want to compute device gain. This is given by:

$$G=20\ \mathrm{Log}_{10}(V_o/V_i)$$

and a program line for computing it can be inserted anywhere after line 140, where the peak-to-peak ramp swings are computed.

There is one other item that should be considered before computing device linearity. This is delay. Ideally, there should be no delay between the input and output ramp for correct point-to-point differencing. With many simple devices, this condition is fairly well met. To ensure the condition for the majority of cases, the test ramp duration should be kept very large relative to any possible delay that could be caused by the device under test.

There are cases, however, where significant delay cannot be avoided. Still, if it is allowed to remain in the ramp data, it can cause differencing errors such as indicated in Fig. 6. To avoid such errors, ramp data should be checked for delay if there is any possibility of it existing. And, if a delay exists, program steps should be taken to remove it from the ramp data before differencing is done.

A simple technique for detecting delay is available if ramp data is acquired so that the endpoints of both ramps are substantially within the acquisition window. An example of this is shown in Fig. 7. When ramps are acquired in this manner, simple cross-function searches for the location of the maximum of each ramp will reveal any difference. If a difference in locations exists, which is obvious in Fig. 7, then a delay exists. The amount of difference, which can be computed from the locations of the ramp maximums, is the amount of delay. This can be used in a FOR-loop computation to shift ramp data in one array for

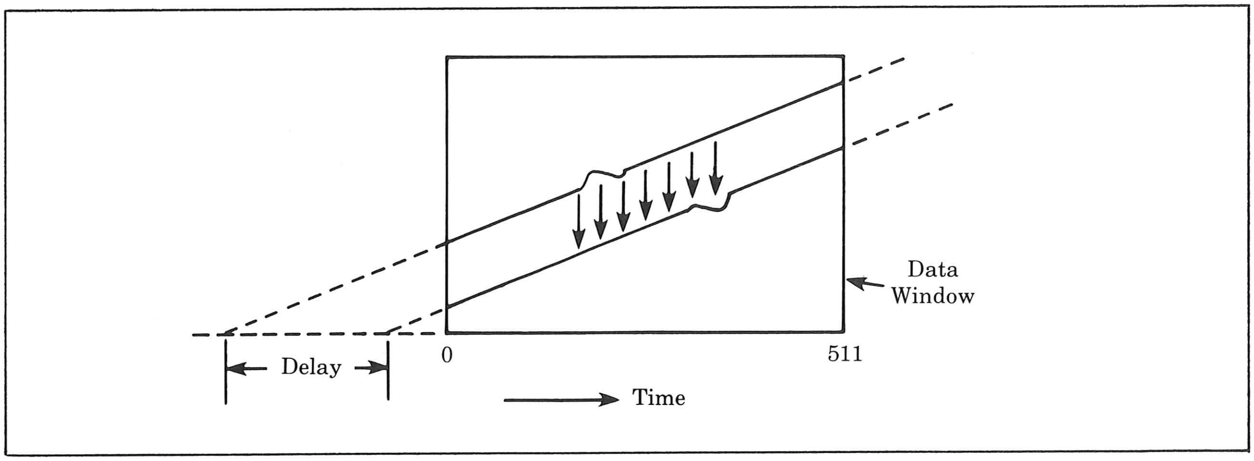


Fig. 6. *Time delays greater than zero may cause errors in linearity computations. As indicated here, ramp anomalies that normally cancel in zero-delay differencing become part of the difference waveform.*

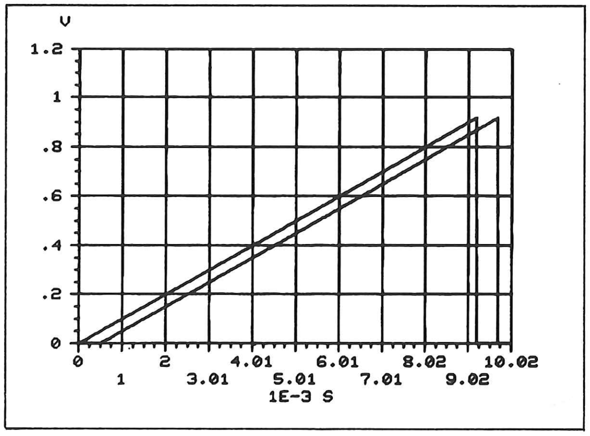


Fig. 7. *To ease detection, computation, and removal of delays, ramps should be acquired so that their endpoints are within the acquisition window, as shown here.*

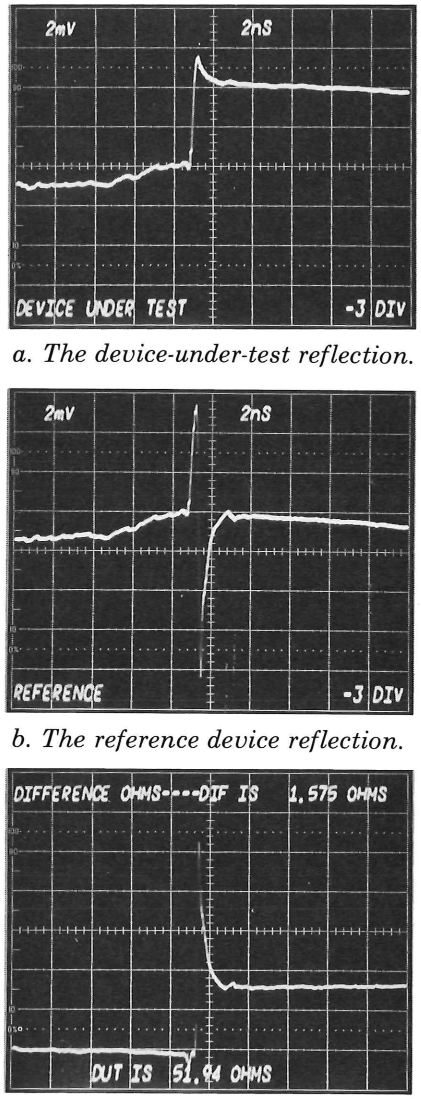
coincidence with ramp data in the other array. Or, the amount of delay can be used to define array zones over areas of desired coincidence which can then be used in zone-directed computations.

Again, signal processing has provided a variety of options and enhancements to the basic measurement concept. There's signal averaging to improve SNR, a choice of comparison to either an ideal or real reference signal, single-channel signal acquisition for simplicity as well as other processing benefits, and the choice of dealing with delay during or after acquisition, whichever is more convenient.

More complex cases. Beyond providing numerous acquisition benefits and simple waveform subtractions, signal processing also offers capabilities for more complex sequences of operations. A simple example of this occurs in time-domain reflectometry (TDR).

Briefly, TDR consists of sending a pulse down a transmission line and studying the pulse reflections for information about the line or the device terminating the line. Signal processing can augment these studies tremendously in a variety of areas. It is very useful in precisely determining incident and reflected voltage levels and is particularly useful in measuring small resistance differences, as might be encountered in testing terminations, connectors, or other wideband devices. Example results from such testing are shown in Fig. 8.

In Fig. 8a, the reflection from the unknown device is shown. A corresponding reflection from a known standard is shown in 8b. Both of these reflection waveforms are digitized and stored. Then one is subtracted from the other, and the difference waveform is analyzed to provide the resistance values shown in Fig. 8c. Details on the full analysis process are presented in Tektronix Application Note AX-3482, "TDR Difference Testing with TEKTRONIX Signal Processing Systems." Copies can be ordered via the HANDSHAKE reply card in this issue.



a. The device-under-test reflection.

b. The reference device reflection.

c. The device-under-test minus the reference.

Fig. 8. *Results of TDR difference testing with an unknown termination. Notice that aberrations common to both the device and reference waveforms cancel in the differencing, leaving a much smoother waveform for analysis.*

Testing to the limit

Sometimes, detailed information is not the desired end. It might be just a simple "yes" or "no" about quality, a pass-fail answer.

In the preceding examples, knowing that pulse aberrations, nonlinearity, or resistance difference are within a certain boundry could be enough. To get such simple GO/NO-GO answers, the basic testing still has to be done and some limits have to be chosen. Then it's mostly a matter of comparing against the limits—do the maximums and minimums of the difference waveforms exceed allowable values? Software MAX and MIN functions can quickly find the difference range for you, and standard IF-THEN statements can make the pass-fail decision under program control.

More graphic methods of limit testing have also been suggested. An example is shown in Fig. 9. In this case, a standard waveform is acquired or generated. Then, upper and lower boundary waveforms are generated by adding and subtracting percentages of the standard: for example, for ±5% limits, LET U=S+.05*S and LET L=S-.5*S. With the bounds generated and stored as individual waveforms, the waveform in question is then acquired and tested against these limit waveforms. Generally, the comparison is element-by-element with a FOR loop. However, it is also possible to subtract the test waveform from each limit waveform and then check the difference waveforms with the cross function for sign changes or other indications of excess.

The best match

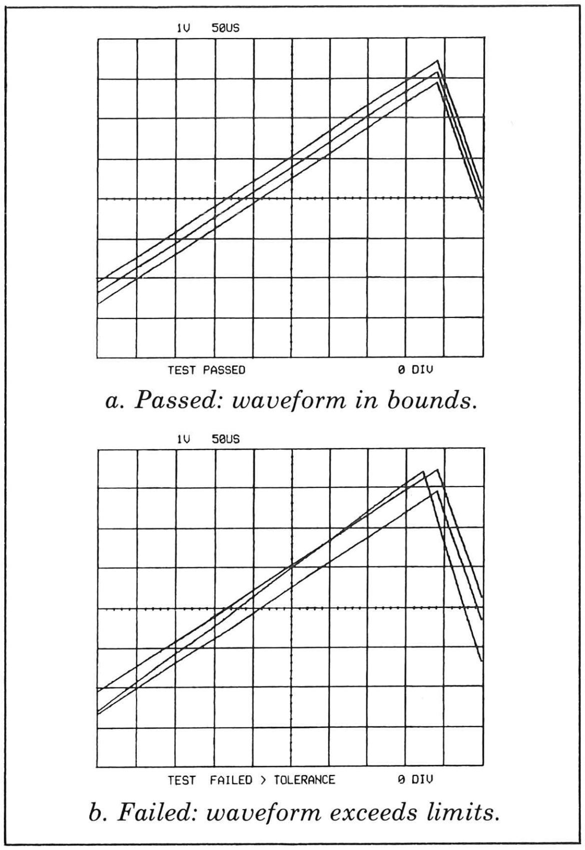
Correlation is another method of getting yes/no answers about waveforms. Specifically, it is an analytic method of testing the degree of match or correlation between two waveforms, or any other set of data for that matter.

In comparison testing, one of the waveforms for correlation is the standard. It might represent the characteristic of a known good device, the servo tracking pattern for head positioning in a computer disk storage system for example. The other waveform is from a similar unknown or untested source. By cross correlating it with the standard, a number indicating the quality or degree of match can be obtained.

Mathematically, the operation of cross correlation is defined by the following relationship:

$$R_{xy}(\tau) = \lim_{T\to\infty} \frac{1}{T} \int_{T/2}^{T/2} fx(t)fy(t+\tau)dt$$

R_{xy} is the correlation function formed by cross correlating the two waveforms, f_x and f_y which are functions of time, t. The correlation function,



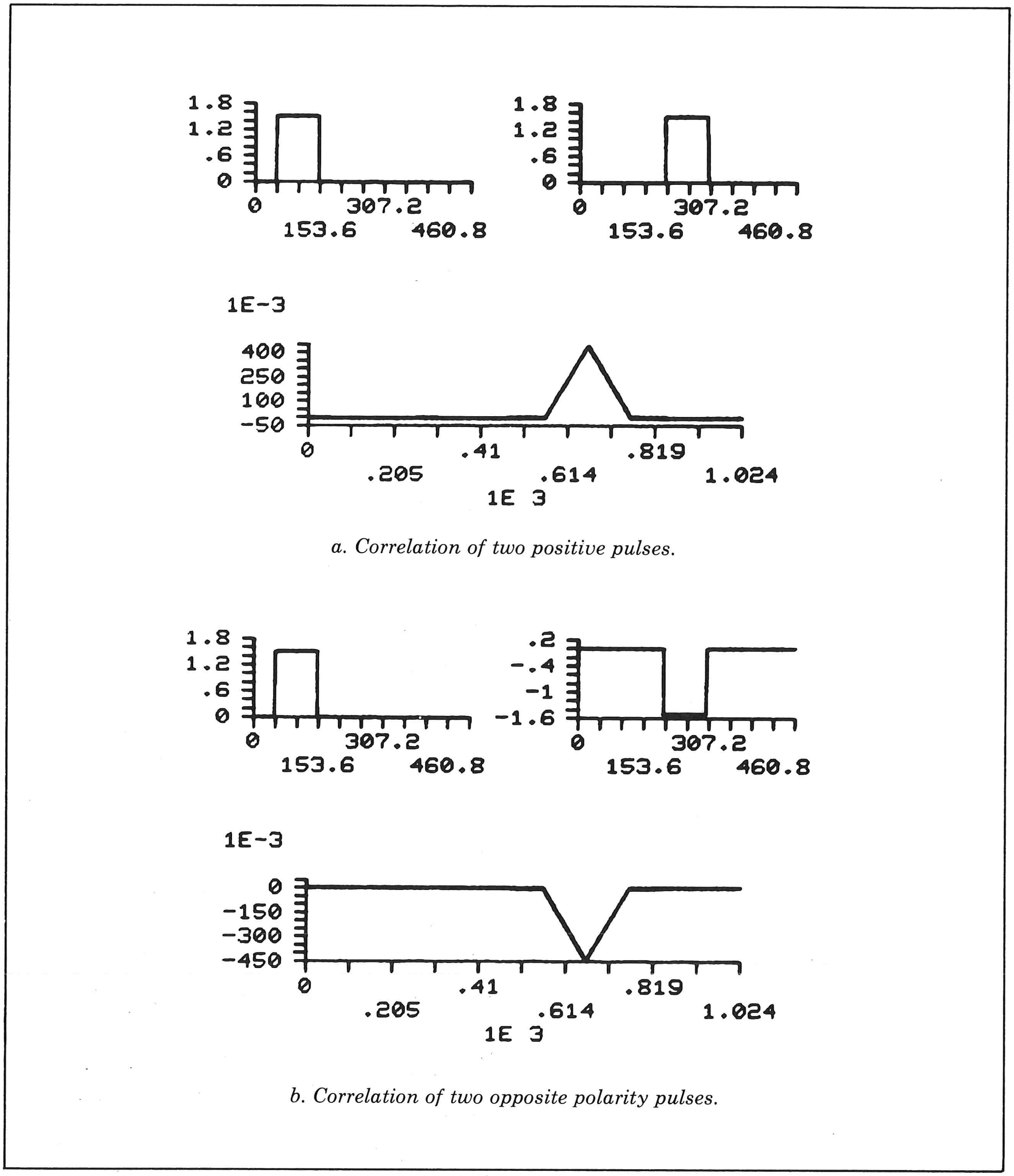
a. Passed: waveform in bounds.

b. Failed: waveform exceeds limits.

Fig. 9. *Simple GO/NO-GO results can also be augmented with graphics. This may be particularly desirable in a "fail" situation, both as a second check and as a means of indicating the reason for failure.*

R_{xy} is also a function of time, but time lag between f_x and f_y instead of absolute time. In fact the lag or operation of lagging is key to performing correlation, and mathematicians sometimes refer to correlation as forming the lagged products of the functions. In plainer words, correlation can be thought of as incrementally sliding one waveform past another, forming the product at each increment, finding the average of each product, and plotting this average at each increment. This is shown graphically in Fig. 10.

The same results can be obtained in an equivalent process using the fast Fourier transform (FFT). This is used almost exclusively in digital signal processing systems because of its ease and efficiency. It consists of Fourier transforming both waveforms to the frequency domain where the spectra appear as complex functions, one array of real spectra and one of imaginary. These can be converted to magnitude and phase spectra for each waveform; however, for correlation, they are left in complex form. The

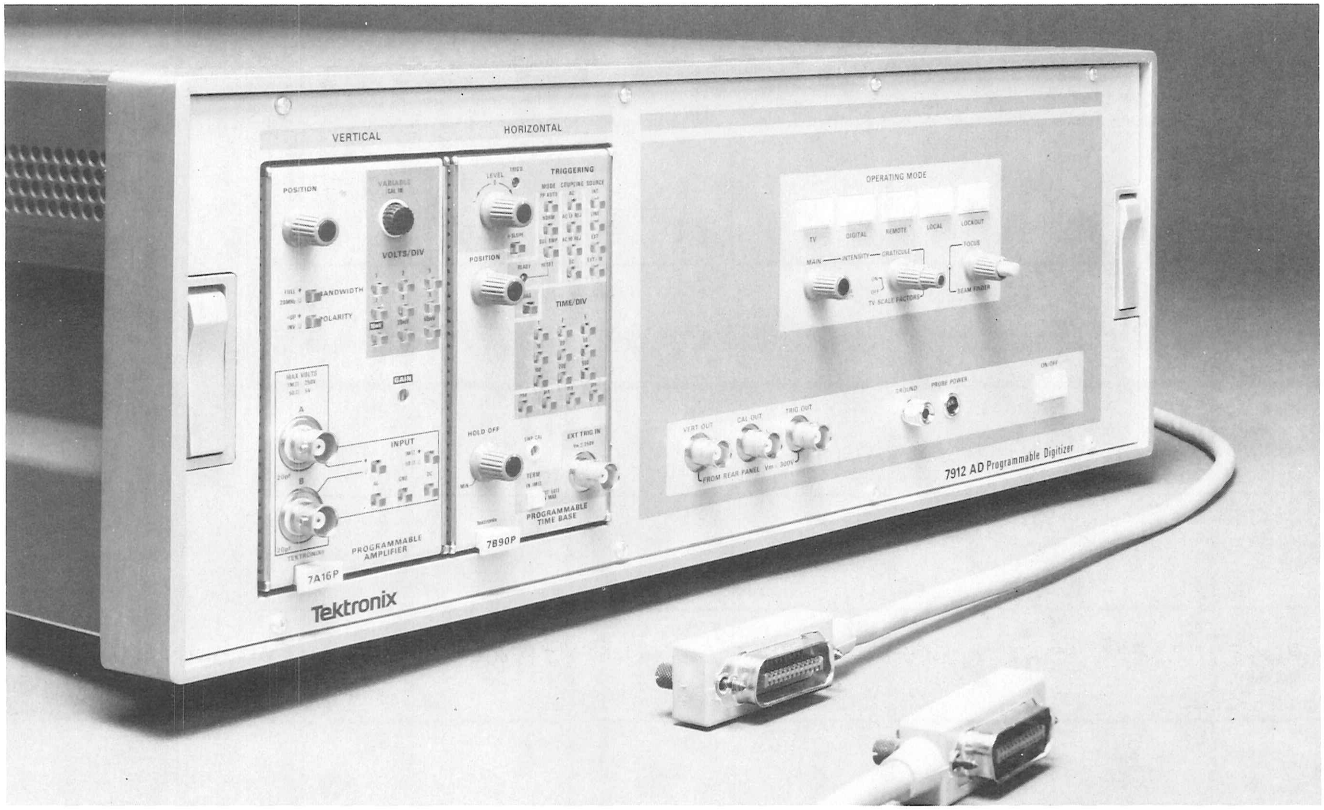


a. Correlation of two positive pulses.

b. Correlation of two opposite polarity pulses.

Fig. 10. *Correlation example. In a and b, the two square pulses are cross correlated to give the triangular correlation function. The process can be viewed as sliding the left-hand pulse to the right. As this sliding (lagging) is done, the pulses are multiplied and the average product found and plotted as the correlation function. The pulse product, hence correlation, is zero when there is no overlap. As the pulses begin to overlap, the area enclosed by the product increases, linearly in this case, to a maximum at full overlap. Then it decreases as the pulses pass each other. The peak of the correlation function indicates maximum correlation, and the peak's location relative to horizontal center indicates the shift (lag) required for maximum correlation (best match). (Note: TEK SPS BASIC V02 Graphics has an XOFFSET operation that can be used to place the zero-point label at any desired point on the horizontal axis.)*

7912AD Programming Information



The TEKTRONIX 7912AD Programmable Digitizer, 7A16P Programmable Amplifier, and 7B90P Programmable Time Base make a fully programmable digitizer package that includes an IEEE 488 bus interface. The 7912AD is also available in TEKTRONIX WP2000-Series Signal Processing Systems. These fully automatic systems are designed, assembled, tested, and documented to satisfy the demand for speed, automation, accuracy, and repeatability in characterizing devices or phenomena which give rise to waveforms in the millisecond to nanosecond range.

7912AD COMMANDS

Header	Argument	Description	Set Only	Query Only
MOD[E]	TV	Set to TV mode		
	DIG	Set to digital mode		
DIG	DAT[A]	Digitize data	X	
	GRA[T]	Digitize graticule only	X	
	SSW	Digitize single-sweep	X	
	DEF,<NR1>	Digitize defects n times	X	
	SA,<NR1>	Signal average 1 to 64 times	X	
DT	ON	Wait for GET to digitize		
	OFF	Do not wait for GET		
TV	ON	Turn on TV scale factors		
	OFF	Turn off TV scale factors		
REM	ON	Assert SRQ when REMOTE pressed		
	OFF	Do not assert SRQ when REMOTE pressed		
OPC	ON	Assert SRQ when operation completes		
	OFF	Do not assert SRQ when completed		
DEF	ON	Flag defects in raw		
	SC2	Transmit channel 2 scale factors	X	
	ATC	Transmit averaged-to-center data	X	
	SA	Transmit signal-averaged data	X	
	EDG[E]	Transmit edge-determined data	X	
	DEF	Transmit defect data	X	
REP	<NR1>	Repeat DIG DAT/READ PTR, VER sequence (0-65535)	X	
DUM[P]	RAW	Dump raw data memory	X	

ASCII & IEEE (GPIB) CODE CHART

B7 B6 B5 BITS	Ø Ø Ø	Ø Ø 1	Ø 1 Ø	Ø 1 1	1 Ø Ø	1 Ø 1	1 1 Ø	1 1 1
B4 B3 B2 B1	CONTROL		NUMBERS SYMBOLS		UPPER CASE		LOWER CASE	
Ø Ø Ø Ø	0 NUL 0 0	20 DLE 10 16	40 SP 20 32	60 0 30 48	100 @ 40 64	120 P 50 80	140 \` 60 96	160 p 70 112
Ø Ø Ø 1	1 GTL SOH 1 1	21 LLO DC1 11 17	41 ! 21 33	61 1 31 49	101 A 41 65	121 Q 51 81	141 a 61 97	161 q 71 113
Ø Ø 1 Ø	2 STX 2 2	22 DC2 12 18	42 " 22 34	62 2 32 50	102 B 42 66	122 R 52 82	142 b 62 98	162 r 72 114
Ø Ø 1 1	3 ETX 3 3	23 DC3 13 19	43 # 23 35	63 3 33 51	103 C 43 67	123 S 53 83	143 c 63 99	163 s 73 115
Ø 1 Ø Ø	4 SDC EOT 4 4	24 DCL DC4 14 20	44 $ 24 36	64 4 34 52	104 D 44 68	124 T 54 84	144 d 64 100	164 t 74 116
Ø 1 Ø 1	5 PPC ENQ 5 5	25 PPU NAK 15 21	45 % 25 37	65 5 35 53	105 E 45 69	125 U 55 85	145 e 65 101	165 u 75 117
Ø 1 1 Ø	6 ACK 6 6	26 SYN 16 22	46 & 26 38	66 6 36 54	106 F 46 70	126 V 56 86	146 f 66 102	166 v 76 118
Ø 1 1 1	7 BEL 7 7	27 ETB 17 23	47 ' 27 39	67 7 37 55	107 G 47 71	127 W 57 87	147 g 67 103	167 w 77 119
1 Ø Ø Ø	10 GET BS 8 8	30 SPE CAN 18 24	50 (28 40	70 8 38 56	110 H 48 72	130 X 58 88	150 h 68 104	170 x 78 120

1 Ø Ø 1	11 TCT HT 9 9	31 SPD EM 19 25	51) 29 41	71 9 39 57	111 I 49 73	131 Y 59 89	151 i 69 105	171 y 79 121
1 Ø 1 Ø	12 LF A 10	32 SUB 1A 26	52 * 2A 42	72 : 3A 58	112 J 4A 74	132 Z 5A 90	152 j 6A 106	172 z 7A 122
1 Ø 1 1	13 VT B 11	33 ESC 1B 27	53 + 2B 43	73 ; 3B 59	113 K 4B 75	133 [5B 91	153 k 6B 107	173 { 7B 123
1 1 Ø Ø	14 FF C 12	34 FS 1C 28	54 , 2C 44	74 < 3C 60	114 L 4C 76	134 \ 5C 92	154 l 6C 108	174 | 7C 124
1 1 Ø 1	15 CR D 13	35 GS 1D 29	55 – 2D 45	75 = 3D 61	115 M 4D 77	135] 5D 93	155 m 6D 109	175 } 7D 125
1 1 1 Ø	16 SO E 14	36 RS 1E 30	56 . 2E 46	76 > 3E 62	116 N 4E 78	136 ∧ 5E 4	156 n 6E 110	176 ~ 7E 126
1 1 1 1	17 SI F 15	37 US 1F 31	57 / 2F 47	77 UNL ? 3F 63	117 O 4F 79	137 UNT — 5F 95	157 o 6F 111	177 RUBOUT (DEL) 7F 127
	ADDRESSED COMMANDS	UNIVERSAL COMMANDS	LISTEN ADDRESSES		TALK ADDRESSES		SECONDARY ADDRESSES OR COMMANDS	

KEY

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

Tektronix®
COMMITTED TO EXCELLENCE

		to digitize		
GRA[T]	ON	Write only the graticule		
	OFF	Reset graticule-only mode		
XYZ	ON	Enable raw data display		
	OFF	Disable XYZ outputs		
	RAW	Enable raw data display		
	ATC	Enable ATC data display		
	SA	Enable signal-averaged data display		
	EDG[E]	Enable EDGE data display		
	DEF	Enable defects data display		
MAI	<NR1>	Set main intensity (0-1023)		
GRI	<NR1>	Set graticule intensity (0-255)		
FOC	<NR1>	Set focus (0-63)		
SSW	ARM	Arm single-sweep trigger		
	DIS	In single-sweep mode, but disarmed		X
	NSS	Not in single-sweep mode		X

		vertical data		
	OFF	Reset defect flags		
LOA[D]	<BINARY>	Load defects array from IEEE 488 bus	X	
ATC		Average raw vertical data for center of trace	X	
INT	<NR1> or NONE	Max. no. of consecutive interpolated data points by ATC or DIG SA		X
EDG[E]		Determine trace edges from raw vertical data	X	
TW	<NR1>	Set EDGE max. trace width		
RT	<NR1>	Set EDGE max. rate of increase		
SET	<MESSAGE>	Settings of programmable functions		X
TES[T]		Self-test data memory	X	
REA[D]	VER	Transmit vertical data array	X	
	PTR	Transmit pointers data array	X	
	SC1	Transmit channel 1 scale factors	X	

		area		
	PR	Dump processed data memory area	X	
VS1	<NR3> or NONE	Vertical channel 1 scale factor		X
VS2	<NR3> or NONE	Vertical channel 2 scale factor		X
HS1	<NR3> or NONE	Horizontal channel 1 scale factor		X
HS2	<NR3> or NONE	Horizontal channel 2 scale factor		X
VU1	<CHARACTERS> or NONE	Vertical channel 1 units		X
VU2	<CHARACTERS> or NONE	Vertical channel 2 units		X
HU1	<CHARACTERS> or NONE	Horizontal channel 1 units		X
HU2	<CHARACTERS> or NONE	Horizontal channel 2 units		X
ERR	<NR1> or NONE	Error code indicated by status byte		X
SRQ	NULL	Service request query		X
ID	<CHARACTERS>	Identity of instrument		X

7912AD STATUS

Error Codes

Code	Description
NONE	No error to report
102	Invalid command header
103	Invalid command argument
201	Attempt to arm single sweep while time base is not in single-sweep mode
202	Checksum error for binary block input by LOAD command
203	Byte count error for binary block input by LOAD command
206	Attempt to digitize with invalid sweep rate (slower than 1 millisecond/division)
302	Fault detected in data memory or 2900 memory controller
304	Invalid or missing readout from plug-ins
305	Data memory (waveform) overwritten
306	No data to average
307	Defects array full
308	6800 MPU interrupt fault
401	Power failure is imminent

Status Byte

		Service Requested		Busy				
	8	7	6	5	4	3	2	1
Remote request	1	X	0	X	0	0	0	1
No condition	0	0	0	X	0	0	0	0
Power up	0	1	0	X	0	0	0	1
Operation complete	0	X	0	X	0	0	1	0
Command error	0	1	1	X	0	0	0	1
Execution error	0	1	1	X	0	0	1	0
Internal error	0	1	1	X	0	0	1	1
Power fail error	0	1	1	X	0	1	0	0

Numbers

<NR1>	Integers with or without signs
<NR3>	Signed scientific notation

IEEE 488 Interface Messages

The 7912AD responds to the following messages:

GTL, LLO, SDC, DCL, SPE, SPD, GET, IFC

The 7912AD does **not** respond to the following messages:

PPC, PPU, TCT

Tektronix®
COMMITTED TO EXCELLENCE

correlation is done by first forming the conjugate of one of the waveform spectral sets, then forming the complex product of the spectra, and finally inverse transforming this product back to the time domain where it appears as the correlation function. In mathematical notation, this process can be expressed in the following manner.

$$R_{xy}(\tau) = F^{-1}\left(F(f_x(t))(F(f_y(t)))^*\right)$$

In this expression, F denotes Fourier transformation, F^{-1} denotes inverse Fourier transformation, and * denotes complex conjugation.

Looking at the operation in the frequency domain offers some further insight into the meaning of the correlation function. Multiplying frequency spectra results in a product containing only the frequency components common to both waveforms. Therefore, the cross-correlation function contains only frequency components common to the two waveforms being correlated. So correlation can be thought of as a process of describing the phase relation and common frequency content of two waveforms.

Notice in Fig. 10, however, that the correlation function only indicates the lag (location of the maximum) required for best match of the two pulses. There's nothing obvious indicating quality of match. What does a maximum correlation of 450 E-3 mean?

To get an indication of quality, the correlation function needs to be normalized by dividing by the product of the RMS values of the waveforms being correlated. The result of doing this is shown in Fig. 11, another example of correlation.

There are a number of properties of digital correlation to be pointed out in Fig. 11. However, let's deal first with the normalized results.

Normalized correlation functions can range in value from -1 to +1. Zero correlation always indicates total lack of relationship between the two waveforms. A negative value of correlation simply indicates opposing polarities in the two waveforms being correlated. A value of ±1 for the maximum indicates a direct functional relationship between the waveforms—they exactly match each other in shape (frequency content). Note, however, that two waveforms can have a normalized correlation of ±1 and still differ in amplitude. But a square wave is still a square wave. What we are primarily concerned with here is similarity of shape, similarity in all aspects except amplitude and time position.

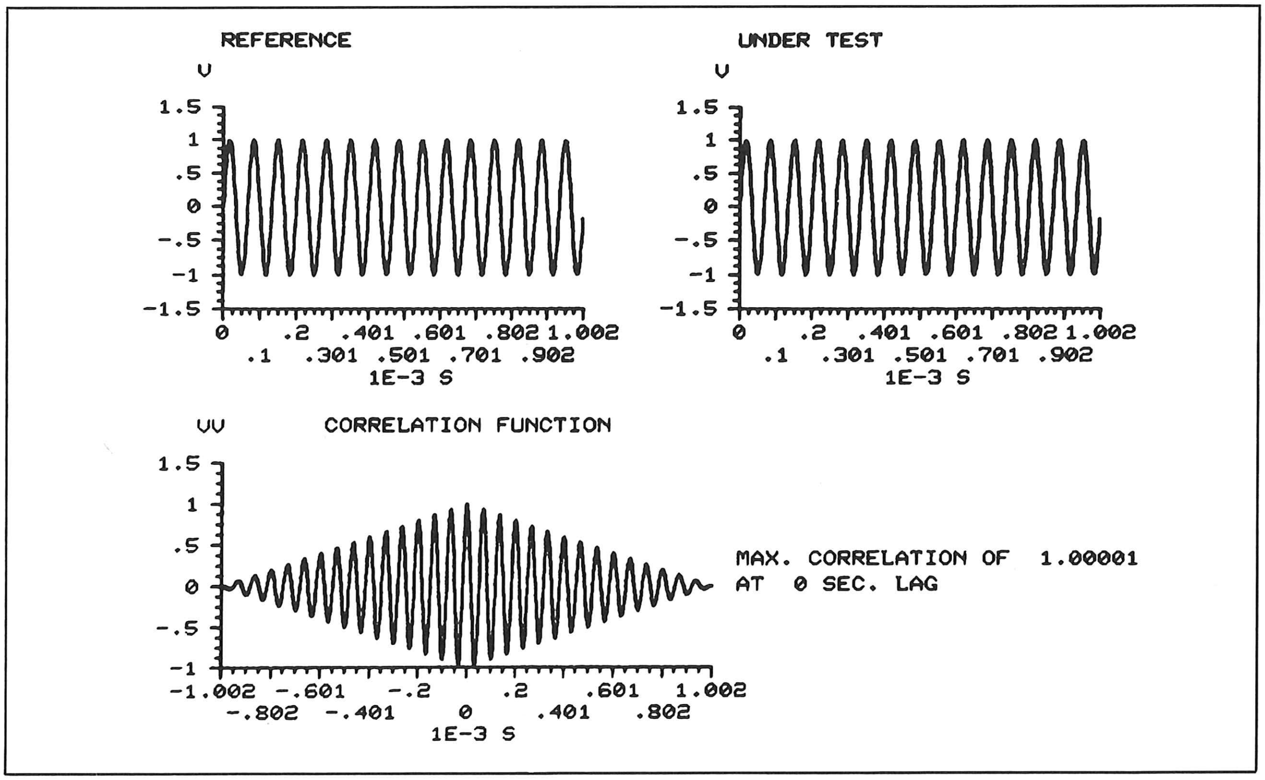


Fig. 11. *Correlation of periodic waveforms (top) produces a periodic correlation function (bottom).*

Limit testing, linearity, aberrations...

In Fig. 11, notice that maximum correlation is 1.00001. Slightly better than perfect correlation? No...actually its the result of slightly less than perfect computation in the digital system—roundoff error accumulating over several operations, etc. The point is, with digital systems (hand-held calculators, minicomputers, or full-sized computers) always be suspicious of the least significant place in the answer. So, in Fig. 11, maximum correlation is +1. The two sinusoids match exactly with zero seconds of lag required for the match.

There's another thing that should be pointed out, too, since it will have tremendous impact on using digital correlation techniques for periodic waveforms. Except at zero lag (center of horizontal), all of the other correlation values are either right or wrong, depending upon your point of view. If you think about two identical, continuous sinusoids extending from plus to minus infinity, and think about shifting one past another and forming the averaged product, you'll find that the correlation function will also be a continuous sinusoid with a normalized amplitude from -1 to +1. That obviously isn't what is shown in Fig. 11. But then the digitized sinusoids used there do not extend from plus to minus infinity. They are zero, digitally, beyond the portions displayed. In other words, the end of the storage array is the end of the stored waveform, period. It's as if the sinusoids were mutliplied by a square pulse. And we've seen the correlation function for a square pulse in Fig. 10. It's triangular. So it's easy to assume that two waveforms, each the product of a square pulse and sinusoid, will produce a correlation function that appears to have been multiplied by a triangular pulse. It turns out this is exactly what should be produced by digital techniques—it is mathematically correct for the data operated on. But the answer can be wrong if what we are thinking of operating on is a continous sinusoid. This error—more correctly a discrepancy in viewpoint—is demonstrated in Fig. 12.

In Fig. 12a, the reference and signal under test are in phase. This is indicated by maximum correlation occurring at zero lag. No correction is needed since there is zero lag. Also, the maximum is 0.988666, indicating a high degree of correlation between the two signals.

In Fig. 12b, the same two signals are used, except the signal under test is shifted by 90°. Still, the degree of correlation should be the same. It'll just be lagged by an amount equal to 90°. However, the triangular window effect demonstrated in Fig. 11 results in a reduced value for maximum correlation, 0.832243. This can be corrected by a $T/(T-|\tau|)$ factor, where T is the signal array duration and τ is the lag. From the information in Fig. 12b and using a hand-held calculator, the corrected degree of correlation is:

0.832243 x 1.002E-3/(1.002E-3 - 1.50685E-4)=0.97955.

continued on page 16

We're Interested in Hearing From You

If you are a user of digital signal processing techniques, HANDSHAKE is your newsletter. We welcome your comments on information contained herein.

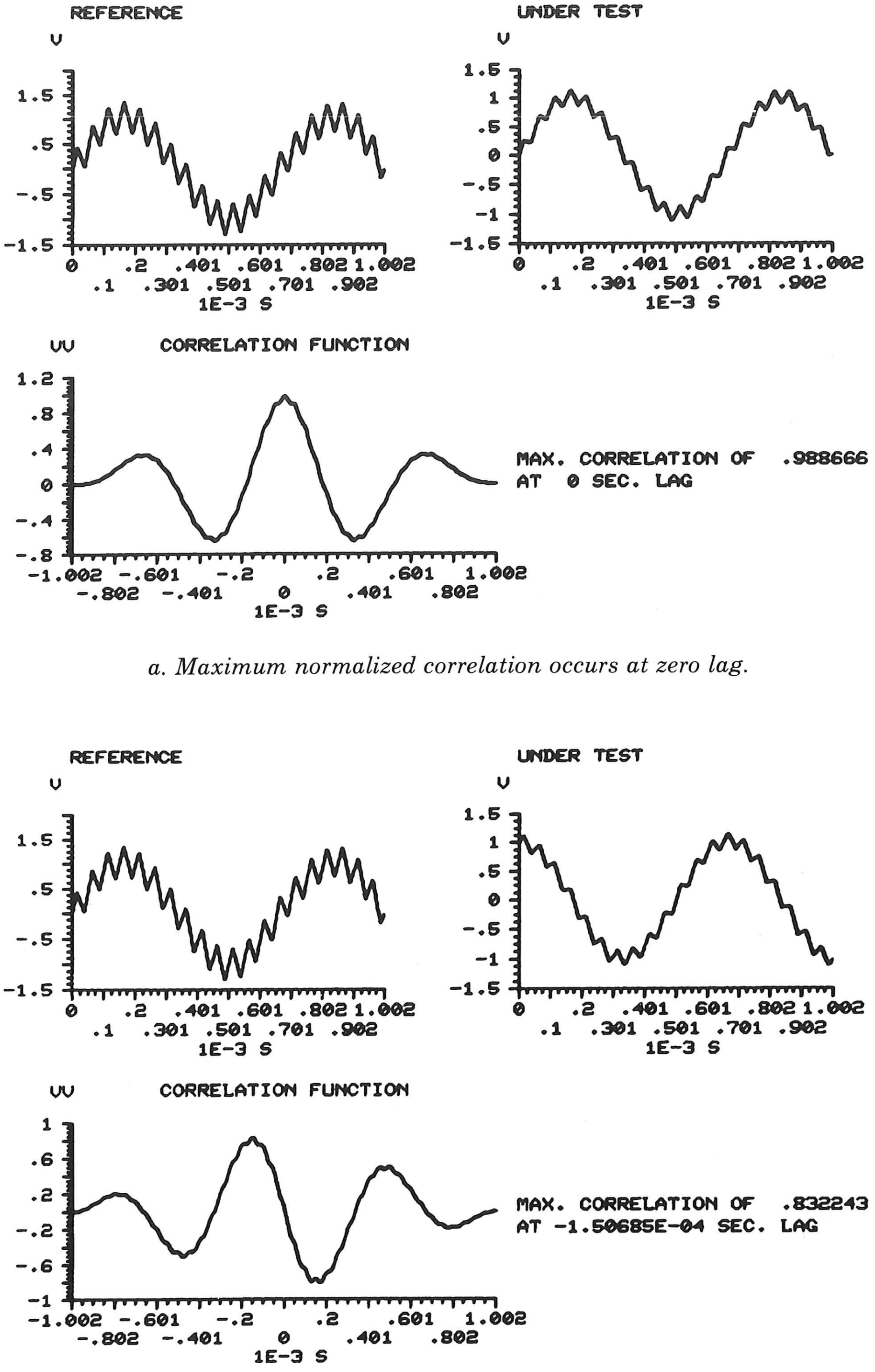
Our continuing goal is to make HANDSHAKE as meaningful as possible. The best way to do this is to know what your exact signal processing needs and applications are. Write and let us know what topics you would like to see discussed in future issues.

We will try to include articles on topics of general interest in future issues of HANDSHAKE. Better yet, why not write an article yourself! Tell us about your measurement problems and how to solve them. Articles accepted for publication in HANDSHAKE will carry full credit to author and company.

We would also like to invite you to send in programs or programming techniques and hints that you have found helpful. Also of special interest are articles using the graphic capabilities of TEK SPS BASIC Software.

Address comments to:
HANDSHAKE Editor
Group 157 (94-384)
Tektronix, Inc.
P.O. Box 500
Beaverton, OR 97077

Use the attached postcard to order your HANDSHAKE subscription; to tell us if you're interested in more information on equipment, applications, and software mentioned in HANDSHAKE; or if you want to contribute to the newsletter. We'll be in touch with you promptly.



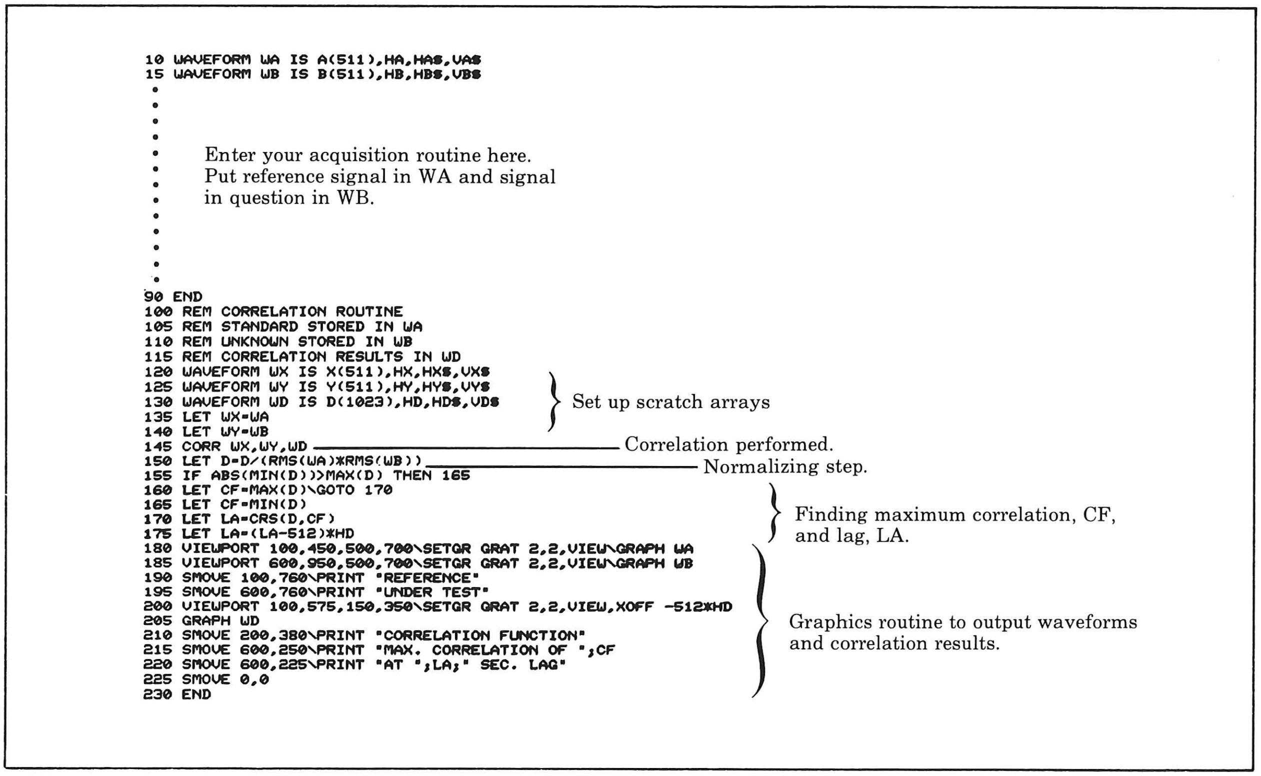
```
MAX. CORRELATION OF   .988666
AT  0 SEC. LAG
```

a. Maximum normalized correlation occurs at zero lag.

```
MAX. CORRELATION OF   .832243
AT -1.50685E-04 SEC. LAG
```

b. Maximum normalized correlation occurs at left of center (nonzero lag) and should be multiplied by a $T/(T-|\tau|)$ correction factor.

Fig. 12. *Two examples of output produced by the correlation program listed in Fig. 13. For periodic waveforms, maximum correlation at other than zero lag must be adjusted by a $T/(T-|\tau|)$ factor.*



```
10 WAVEFORM WA IS A(511),HA,HA$,VA$
15 WAVEFORM WB IS B(511),HB,HB$,VB$
•
•
•
•
•      Enter your acquisition routine here.
•      Put reference signal in WA and signal
•      in question in WB.
•
•
•
•
•
90 END
100 REM CORRELATION ROUTINE
105 REM STANDARD STORED IN WA
110 REM UNKNOWN STORED IN WB
115 REM CORRELATION RESULTS IN WD
120 WAVEFORM WX IS X(511),HX,HX$,VX$       }
125 WAVEFORM WY IS Y(511),HY,HY$,VY$       }
130 WAVEFORM WD IS D(1023),HD,HD$,VD$      } Set up scratch arrays
135 LET WX=WA                              }
140 LET WY=WB                              }
145 CORR WX,WY,WD ————————————————————————— Correlation performed.
150 LET D=D/(RMS(WA)*RMS(WB))——————————————————— Normalizing step.
155 IF ABS(MIN(D))>MAX(D) THEN 165
160 LET CF=MAX(D)\GOTO 170                               }
165 LET CF=MIN(D)                                        } Finding maximum correlation, CF,
170 LET LA=CRS(D,CF)                                     } and lag, LA.
175 LET LA=(LA-512)*HD                                   }
180 VIEWPORT 100,450,500,700\SETGR GRAT 2,2,VIEW\GRAPH WA  }
185 VIEWPORT 600,950,500,700\SETGR GRAT 2,2,VIEW\GRAPH WB  }
190 SMOVE 100,760\PRINT "REFERENCE"                        }
195 SMOVE 600,760\PRINT "UNDER TEST"                       }
200 VIEWPORT 100,575,150,350\SETGR GRAT 2,2,VIEW,XOFF -512*HD }
205 GRAPH WD                                               } Graphics routine to output waveforms
210 SMOVE 200,380\PRINT "CORRELATION FUNCTION"             } and correlation results.
215 SMOVE 600,250\PRINT "MAX. CORRELATION OF ";CF          }
220 SMOVE 600,225\PRINT "AT ";LA;" SEC. LAG"               }
225 SMOVE 0,0                                              }
230 END                                                    }
```

Fig. 13. *TEK SPS BASIC program used for the correlation results shown in Fig. 12.*

It is better, though, to include data correction as a part of the correlation program. The program used for the results in Fig. 12 is listed in Fig. 13. The correction can be added between lines 175 and 180 with this statement:

LET CF=CF*(511*HD)/(511*HD-ABS(LA)).

The program result for the lag situation of Fig. 12b produced 0.9799 as the corrected value of maximum normalized correlation.

It is important to note that the $T/(T-|\tau|)$ correction is not wanted for cases where the reference signal and signal under test are pulses rising from zero and returning to zero within the arrays. But, for periodic signals, the correction is necessary.

So now we're able to compute normalized correlation for comparing either pulse or periodic signals. And, if the maximum normalized correlation obtained is ±1, we know that the signals are exactly the same, except possibly in amplitude and positioning. But what about maximum values less than one?

A correlation of 0.9 indicates a greater degree of similarity than 0.8. It doesn't say what dissimilarities exist or where they exist. But it does say they exist, and it provides an index for other judgements. Correlation is a convenient and systematic way of answering the question: "On a scale of zero to one, how would you judge these two waveforms?" Also, in specific test situations, it is possible to empirically determine what level of correlation constitutes an acceptable match. With this, correlation results, like the results of previous comparison examples, can be used as simple GO/NO-GO indicators. Or the correlation function can be used as a stepping stone to even higher levels of waveform analysis.

By Bob Ramirez,
HANDSHAKE Staff

Faster 7912AD data transfers to 4050-Series controllers

The 4051 Graphic Computing System makes a nice, small controller for the 7912AD Programmable Digitizer. Programming and techniques for reading 7912AD data were discussed in the Summer 1978 (Vol. 3, No. 4) 4050 family of Graphic Computing Systems—the 4052 and 4054. The 4052 is quite similar to the 4051, but it is about ten times as fast and can support up to 64 kilobytes of memory. The 4054 adds to this a large, higher-resolution screen (19 inches diagonally) with optional refresh graphics. The 7912AD is now supported by a family of software compatible controllers which combines low cost, high performance, and ease of use.

The Summer 1978 HANDSHAKE article contains a simple program for reading binary data from the 7912AD. It has recently been pointed out that the data transfer rate can be greatly increased if matrix multiplication is available and room is provided for an additional temporary array twice the size of the data array to be read. With a 4051 you'll need to install the matrix ROM (4051RO1) to get matrix multiplication. If you have a 4052 or 4054, the matrix functions are built in.

With the matrix functions provided, let's review the common way of reading 7912AD data. As an example, say we have executed the following 4050-Series statement:

```
PRINT @U,0: "DIG DAT;READ VER;"
```

where U contains the 7912AD primary address and 0 is the secondary address. These commands tell the instrument to digitize a waveform and prepare to send the raw vertical array the next time the instrument is addressed to "talk."

To read the data into a 4050-Series system, the commands listed in Fig. 1 are used. (If you are not familiar with the data format, refer to the HANDSHAKE article mentioned above or the Programming Section of the 7912AD Operators Manual, part no. 070-2384-01.)

If memory is available for a temporary matrix, read time can be greatly reduced. To read the same data using the faster method, start with the same three commands:

```
WBYTE @64+U,96:
RBYTE X,Y1,Y2
S=(Y1*256+Y2-1)/2
```

Now, dimension the destination array, V, and two temporary arrays, V1 and V2.

```
DELETE V,V1,V2
DIM V(S),V1(S,2),V2(2,1)
```

Then read the entire data array into V1, as follows:

```
RBYTE V1
```

When reading into a matrix, the rightmost subscript varies the fastest. So, when the transfer is complete, the following conditions exist:

V1(1,1) contains word 1, high byte
V1(1,2) contains word 1, low byte

and generally,

```
WBYTE @64+U,96: ................................. Send the 7912AD talk address
RBYTE X,Y1,Y2 ................................... Read a "%" and the byte count
S=(Y1*256+Y2-1)/2 ............................... The byte count includes a one-byte check sum,
                                                  and there are two bytes per data word.
                                                  'S' will be the number of data words.
DELETE V
DIM V(S) ........................................ Dimension array V
FOR I=1 TO S
RBYTE Y1,Y2 ..................................... Read a data word
V(I)=Y1*256+Y2
NEXT I
RBYTE Y1,Y2 ..................................... Pack byte into array element
WBYTE @95: ...................................... Untalk the 7912AD
```

Fig. 1. *Standard approach to reading 7912AD-data into a 4050-Series Graphic Computing System.*

```
8000 PRINT @U,0: "VS1?" ......................... Query and read vertical scale
8010 INPUT @U,0: X,V3 ........................... X gets 1 from VS1, V3 gets actual volts/div
8020 PRINT @U,0: "DIG DAT;ATC;READ ATC"
8030 WBYTE @64+U,96: ............................ Send talk address
8040 RBYTE X,Y1,Y2 .............................. Read "%" and byte count
8050 SY(Y1*256+Y2-1)/2........................... Compute size of data
8060 DELETE V,V1,V2
8070 DIM V(S),V1(S,2),V2(2,1)
8080 RBYTE V1,X,X ............................... Read data, then read and discard
                                                  checksum and ";"
8090 WBYTE @95: ................................. Untalk
8100 V2(1,1)=2*V3................................ =256*V3/128
8110 V2(2,1)=V3/128 ............................. =1*V3/128
8120 V=V1 MPY V2
8130 DELETE V1,V2
8140 V=V-Z ...................................... Subtract zero reference
8150 RETURN
```

Fig. 2. *A faster method of reading 7912AD data into a 4050-Series Graphic Computing System. This uses matrix functions and a double-sized temporary array.*

V1(i,1) contains word i, high byte
V1(i,2) contains word i, low byte.

At this point, V2 can be loaded with some constants and a multiplication done.

```
V2(1,1)=256
V2(2,1)=1
V=V1 MPY V2
```

When doing the matrix multiply, array V of size S is treated as an S x 1 matrix. Thus,

```
V(1)=V(1,1)  =V1(1,1)*V2(1,1)+V1(1,2)*V2(2,1)
             =V1(1,1)*256+V1(1,2)
V(2)=V(2,1)  =V1(2,1)*V2(1,1)+V1(2,2)*V2(2,1)
             =V2(2,1)*256+V1(2,2)
```

and generally,

```
V(i)=V(i,1)  =V1(i,1)*V2(1,1)+V1(i,2)*V2(2,1)
             =V1(i,1)*256+V1(i,2)
```

After the multiply, the temporary matrix can be deleted and the last two bytes read as follows:

```
RBYTE Y1,Y2
DELETE V1,V2
```

The above method is faster than that listed in Fig. 1 for two reasons. First, reading directly into the array eliminates the overhead of having a FOR loop. And, second, the matrix multiply performs the two multiplies and addition faster than the BASIC interpreter can perform V(I)=X*256+Y.

A further expansion of this faster method is also possible. Averaged or ATC data from the 7912AD is nearly always normalized before being processed. This is usually done by subtracting a zero-reference value and multiplying by the number of volts per quantizing level. For ATC data this is the displayed volts/division divided by 128. This constant can be used in matrix V2 to automatically scale the data, thus eliminating a step in the normalizing process. To do this, use

```
V2(1,1)=256*(volts/div)/128 and

V2(2,1)=(volts/div)/128
```

Then all that remains is subtracting out the zero reference,

```
V=V-Z
```

Since the matrix multiply leaves the data in volts, the zero reference must be an offset in volts. This can be accomplished by digitizing a ground trace and setting the variable Z to zero, then performing the steps above to acquire the data in array V. Then simply set Z to the mean of V by the following statement:

```
Z=SUM (V)/512
```

Figure 2 lists a complete routine for reading ATC data from the 7912AD with a 4050-Series Graphic Computing System. It will get data to your system faster so you can get answers faster.

By Alan Jeddeloh, SPS Marketing
Tektronix, Inc.

Acknowledgements: The author wishes to thank Corey Hirsch, European Marketing Center, Tektronix, Inc., for suggesting the technique discussed in this article.

Getting the most out of TEK BASIC graphics

Graphing waveforms— Still another way with 4050-Series controllers

With increasing use of 4050 Graphic Computing systems as controllers for TEKTRONIX Waveform Digitizers, waveform graphics has become a popular topic. A variety of routines have been written for displaying waveforms, and here is one more to add to your bag of tricks.

The main feature of this routine is that it doesn't plot a full graticule such as found on oscilloscope CRTs. Instead, it provides just a set of tic-marked axes (see Fig. 1). This keeps things uncluttered when you wish to graph several waveforms at reduced size on a single page.

The routine is listed in Fig. 2. It assumes you have already defined a VIEWPORT and that you have a waveform stored in array A. It also assumes that the following waveform variables have been defined:

A(S)	waveform array
S	size of the waveform array
S1	the waveform's digital sampling interval
B$	constant one blank
V$	vertical units (e.g., V for volts)
H$	horizontal units (e.g., S for seconds)
Q3	horizontal label interval, typically 1 for every tic mark or 2 for every other tic mark
X9	offset for horizontal labeling, usually 0

Other variables, defined by the routine, include W3 as a temporary variable, I as a FOR loop index,

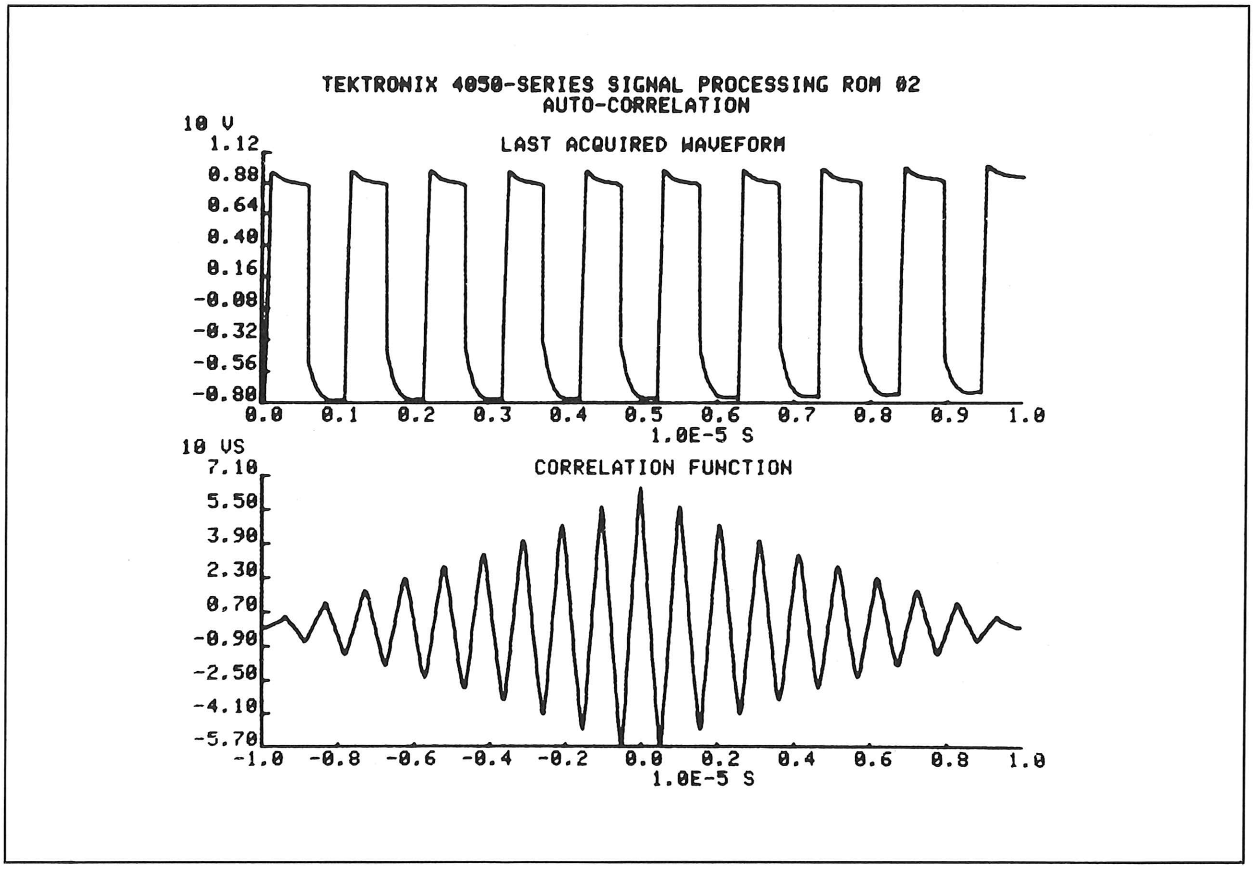
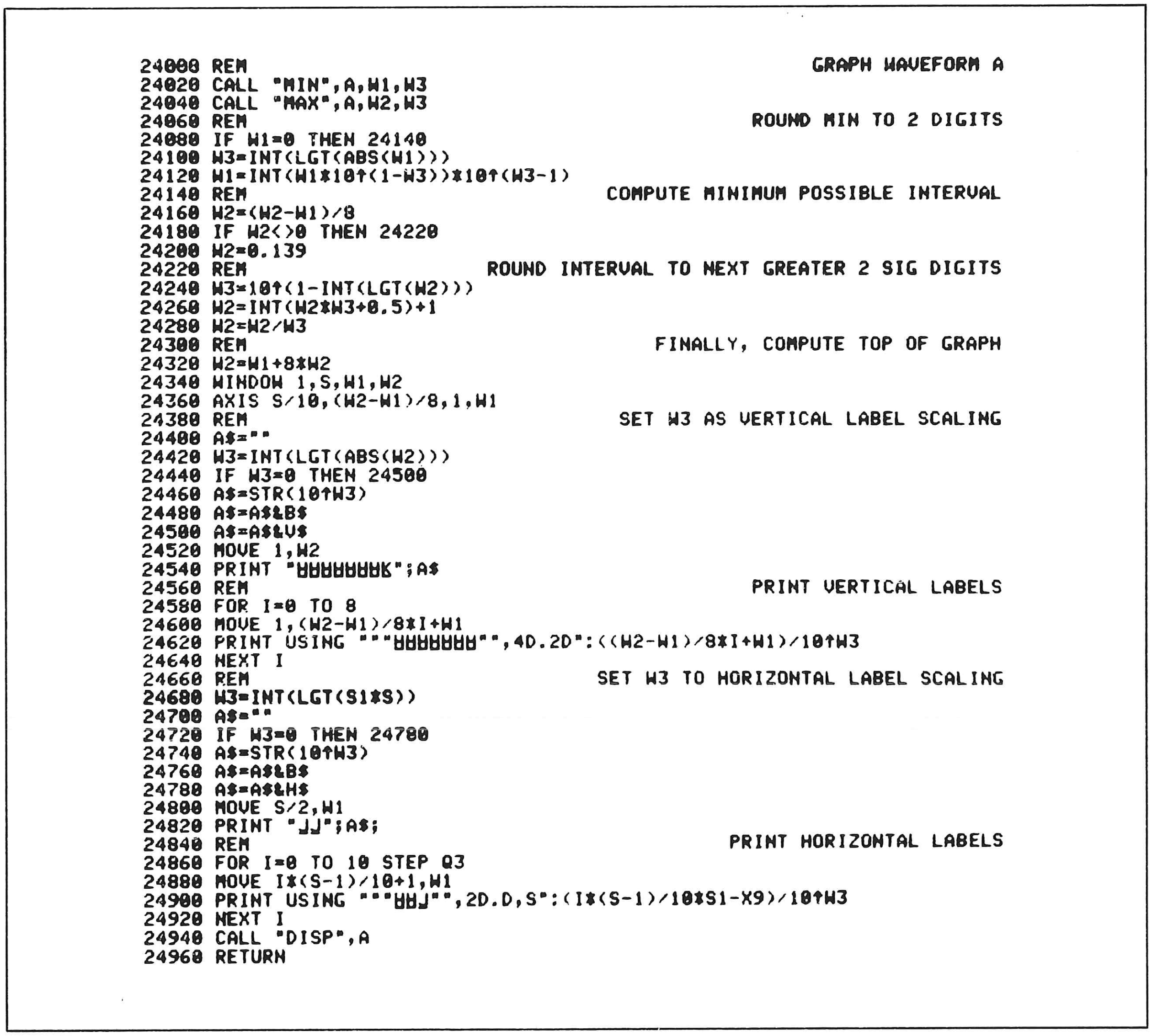


Fig. 1. *Uncluttered graphic output of the 4050-BASIC subroutine listed in Fig. 2. The subroutine was executed twice, once with a viewport defined for the upper half of the terminal screen and again with the viewport defined for the lower half of the screen (headings were added with separate PRINT statements).*

and A$ for temporary string storage. Control characters used are H for back space, J for line feed, and K for moving up one line.

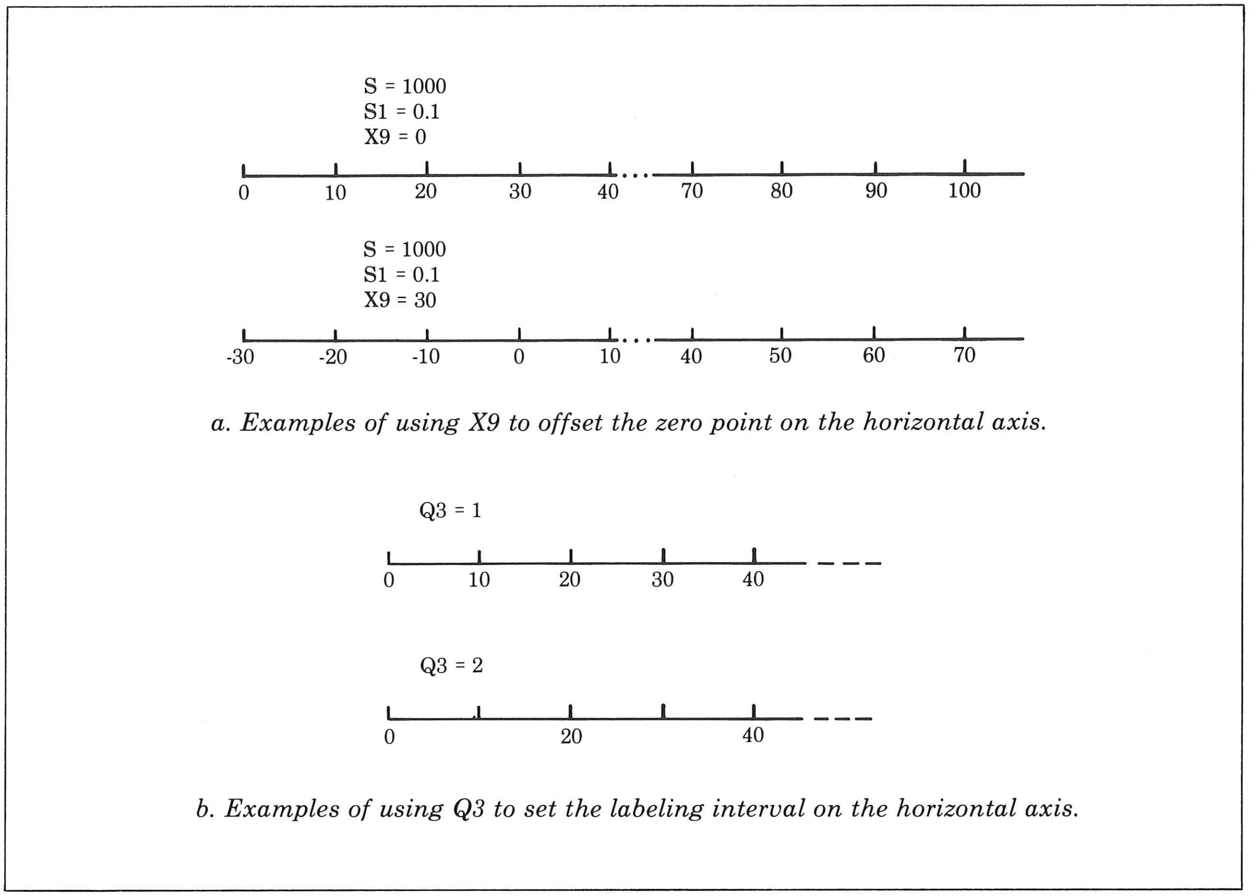
It should also be noted that the routine in Fig. 2 depends upon Signal Processing ROM Pack #1 being installed in the 4050-Series controller while the routine is running. The MIN, MAX, and DISPLAY functions of this ROM pack greatly reduce the amount of programming otherwise required for graphing a waveform.

Briefly, the routine starts by finding the minimum and maximum values of the waveform in A. The two values returned by this, W1 and W2, are refined further in lines 24080-24280 to provide minimum and maximum values for the graphic WINDOW to be used. With these limits set, the window is set up in line 24340. Note that the horizontal limits for the WINDOW function are the subscript limits of the array, 1 and S. Following the WINDOW function, a set of axes is drawn by line 24360 with tics set at 10 major



```
24000 REM                                               GRAPH WAVEFORM A
24020 CALL "MIN",A,W1,W3
24040 CALL "MAX",A,W2,W3
24060 REM                                          ROUND MIN TO 2 DIGITS
24080 IF W1=0 THEN 24140
24100 W3=INT(LGT(ABS(W1)))
24120 W1=INT(W1*10↑(1-W3))*10↑(W3-1)
24140 REM                              COMPUTE MINIMUM POSSIBLE INTERVAL
24160 W2=(W2-W1)/8
24180 IF W2<>0 THEN 24220
24200 W2=0.139
24220 REM                    ROUND INTERVAL TO NEXT GREATER 2 SIG DIGITS
24240 W3=10↑(1-INT(LGT(W2)))
24260 W2=INT(W2*W3+0.5)+1
24280 W2=W2/W3
24300 REM                                  FINALLY, COMPUTE TOP OF GRAPH
24320 W2=W1+8*W2
24340 WINDOW 1,S,W1,W2
24360 AXIS S/10,(W2-W1)/8,1,W1
24380 REM                               SET W3 AS VERTICAL LABEL SCALING
24400 A$=""
24420 W3=INT(LGT(ABS(W2)))
24440 IF W3=0 THEN 24500
24460 A$=STR(10↑W3)
24480 A$=A$&B$
24500 A$=A$&V$
24520 MOVE 1,W2
24540 PRINT "HHHHHHHK";A$
24560 REM                                          PRINT VERTICAL LABELS
24580 FOR I=0 TO 8
24600 MOVE 1,(W2-W1)/8*I+W1
24620 PRINT USING """HHHHHHH"",4D.2D":((W2-W1)/8*I+W1)/10↑W3
24640 NEXT I
24660 REM                             SET W3 TO HORIZONTAL LABEL SCALING
24680 W3=INT(LGT(S1*S))
24700 A$=""
24720 IF W3=0 THEN 24780
24740 A$=STR(10↑W3)
24760 A$=A$&B$
24780 A$=A$&H$
24800 MOVE S/2,W1
24820 PRINT "JJ";A$;
24840 REM                                        PRINT HORIZONTAL LABELS
24860 FOR I=0 TO 10 STEP Q3
24880 MOVE I*(S-1)/10+1,W1
24900 PRINT USING """HHJ"",2D.D,S":(I*(S-1)/10*S1-X9)/10↑W3
24920 NEXT I
24940 CALL "DISP",A
24960 RETURN
```

Fig. 2. *4050-BASIC subroutine for graphing waveforms on labeled axes.*



a. Examples of using X9 to offset the zero point on the horizontal axis.

b. Examples of using Q3 to set the labeling interval on the horizontal axis.

Fig. 3. *Using X9 and Q3 to specify optional labeling of the horizontal axis.*

horizontal points and 8 vertical points. This is immediately followed by computing vertical tic-mark labels and then printing the labels by the FOR-loop segment beginning at line 24580. This is followed by another segment of lines for computing and printing the horizontal labels. Some options are offered here by variables X9 and Q3 which are explained further in Fig. 3. And finally, in line 24940, the ROM Pack #1 DISPLAY function is used to overlay the waveform data from array A on the plotted and labeled axes.

As indicated by the RETURN in the last line, the Fig. 2 listing is merely a subroutine for use with a larger program. You may wish to have several other waveform graphing subroutines attached to your main program, too...just to keep your graphics options open.

By Bob Ramirez, HANDSHAKE Staff.
Based on a program submitted by
Alan Jeddeloh, SPS Marketing,
Tektronix, Inc.

Signal processing systems user's application program library

If one of the following programs interests you, a listing and any available support literature will be sent to you--free of charge. Order your program by title from SPS Software Support at the address shown below, or contact your local Tektronix representative. For a copy of the latest index of programs in the library, check the appropriate box on the reply card in this issue of *HANDSHAKE.*

Remember, if you have a TEK BASIC program you would like to share with other Signal Processing System users, you may enter it into the library by sending the program listing to:

Tektronix, Inc.
SPS Software Support
94-319
P.O. Box 500
Beaverton, OR 97077

Outside the USA, send your programs to SPS Software support via your local Tektronix representative. Please include with your program a short description of what it does and how it does it. We would also like to know about any special data conditions, instruments, or software package requirements. The memory requirements for running the program would also be very helpful.

TEK SPS BASIC V01 Program Abstracts

Swept-Frequency VSWR and Insertion Loss Measurements

Instruments Required: DPO, CP1100- or CP4100-Series Controller, two 7A22 Differential Amplifier plug-ins, 7B80 Time Base plug-in, swept-frequency generator, two directional couplers, two square-law detectors.
Software Packages Required: DPO Driver with ENVDPO command and Graphics.
Listing Length: 1600 lines.

There are three programs in this package. The first acquires swept-frequency waveforms from a slotted line and then computes insertion loss. The third program is a graphics routine for plotting the VSWR and insertion loss results.

Computation of Magnitude and Phase Spectra

Instruments Required: None
Software Packages Required: Signal Processing.
Listing Length: 152 lines.

This program computes the magnitude and phase spectra for both transient and periodic signals.

Frequency Response Estimation

Instruments Required: None.
Software Packages Required: Signal Processing.
Listing Length: 214 lines.

Using digitized and stored versions of the wideband input signal to a system and the corresponding output signal, this program computes an estimate of the system's frequeny-response function.

Impulse Response Estimation

Instruments Required: None.
Software Packages Required: Signal Processing.
Listing Length: 405 lines.

This program, given the input and output signals, estimates the impulse response of a system. It can also estimate the input to a system, given the impulse response and output signal.

Statistics Routines

Instruments Required: None.
Software Packages Required: Graphics.
Listing Length: 653 lines.

This package contains two programs, STAT and GRAPH.

The STAT program accepts either x data alone or both x and y data. It outputs the x and y data points, summary statistics for each, linear regression calculations for the x and y data, and tests for equal means and variances.

The GRAPH program allows selection of a confidence band for the straight-line curve fit. It gives you the choice of fitting the data to a linear, exponential, or power curve or simply plotting the points. You can select the graph limits or allow the machine to do the work. If the limits chosen by the machine are unacceptable, you can reselect them after being asked which curve to plot. If you wish, you can also display the data points as average values with associated error flags.

Some simple statistical computations using TEK SPS BASIC

The array processing capabilities and the MEA and RMS functions of TEK SPS BASIC software make programming of other statistical functions quite easy.

In the examples below, the defining expression of the statistical quantity is given on the left. In these defining expressions, n represents the number of data values to be analyzed, and x represents the ith data value. The mean of the data values is represented by x. The TEK SPS BASIC implementations of these expressions are given on the right. There, N is the number of the data values in array X, with X(I) representing the Ith element, and MX is the mean of array X. Where applicable, an additional array, Y, is used and has corresponding symbols. Also, X1 and Y1 are used as scratch arrays.

By Laurie DeWitt, SPS Signal Analysis Group Tektronix, Inc.

	FORMULA	TEK SPS BASIC
Mean	$\bar{x} = \frac{1}{n}\sum_{i=0}^{n-1} x_i$	MX=MEA(X)
RMS	$rms = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1} x_i^2}$	RX=RMS(X)

Standard Deviation	$S = \sqrt{\frac{\sum_{i=0}^{n-1} (x_i-\bar{x})^2}{n-1}}$	X1=X-MEA(X)
		S=SQR(N/(N-1))*RMS(X1)

Correlation Coefficient between X and Y	$r = \frac{\sum_{i=0}^{n-1} (x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\left[\sum_{i=0}^{n-1} (x_i-\bar{x})^2\right]\left[\sum_{i=0}^{n-1} (y_i-\bar{y})^2\right]}}$	X1=X-MEA(X)
		Y1=Y-MEA(Y)
		Z=X1*Y1
		R=MEA(Z)/(RMS(X1)*RMS(Y1))

Geometric Mean	$GM = \sqrt[n]{x_0 \cdot x_1 \cdot x_2 \cdot \ldots \cdot x_{n-1}}$
	$(EXP(LOG(GM)) = EXP(\frac{1}{n}\sum_{i=0}^{n-1} LOG(x_i)))$
		X1=LOG(X)
		GM=EXP(MEA(X1))
		(This implementation is valid only if all elements of X are greater than 0.)

Harmonic Mean	$HM = \frac{n}{\sum_{i=0}^{n-1} {}^{1}/_{x_i}}$	X1=1/X
		HM=1/(MEA(X1))
		(This implementation is valid only if no element of X equals 0.)

Mean Deviation	$MD = \frac{1}{n}\sum_{i=0}^{n-1} \lvert x_i-\bar{x} \rvert$	X1=ABS(X-MEA(X))
		MD=MEA(X1)

Rth Moments (about origin)	$MR = \frac{1}{n}\sum_{i=0}^{n-1} x_i^r$	X1=SGN(X)*ABS(X)∧R
		IF ITP(R/2)*R=4 THEN X1=ABS(X1)
		MR=MEA(X1)

Rth Moments (about mean)	$MR = \frac{1}{n}\sum_{i=0}^{n-1} (x_i-\bar{x})^r$	X1=X-MEA(X)
		X1=SGN(X1)*ABS(X1)∧R
		IF ITP(R/2)*2=R THEN X1=ABS(X1)
		RM=MEA(X1)

Finding Out More About Tektronix Signal Processing Systems

Tektronix, Inc.
7827 Convoy Court
Suite 401
San Diego, CA 92111
(714)292-7330

Tektronix, Inc.
3333A Octavius Drive
Santa Clara, CA 95051
(408)243-9620

Tektronix, Inc.
P.O. Box 344529
Dallas, TX 75234
(214)233-7791

Tektronix, Inc.
2 Research Court
Rockville, MD 20850
(301)948-7151

Tektronix, Inc.
1258 Ortiz Drive, S.E.
Albuquerque, NM 87108
(505)265-5541

Tektronix, Inc.
40 Gill Lane
Woodbridge, NJ 07095
(201)636-8616

Tektronix, Inc.
4660 Churchill Road
St. Paul, MN 55112
(612)484-8571

Tektronix, Inc.
482 Bedford St.
Lexington, MA 02173
(617)861-6800

Tektronix, Inc.
24155 Drake Road
Farmington, MI 48024
(313)478-5200

Tektronix, Inc.
3320 Holcomb Bridge Road
Peachtree Industrial Blvd.
Norcross, GA 30092
(404)449-4770

Tektronix maintains Field Offices and Service Centers throughout the world. In the United States, the Field Offices listed below have Signal Processing Specialists ready to answer your questions and help you select the system that best suits your measurement needs. Outside of the United States, the Tektronix subsidiary or distributor in your country will be pleased to offer you the same services.

Tektronix®
COMMITTED TO EXCELLENCE

HANDSHAKE
Newsletter of the Signal Processing Systems Users Group

HANDSHAKE Editor
Group 157 (94-384)
Tektronix, Inc.
P.O. Box 500
Beaverton, Oregon 97077

A -4396