Vol. 4 No. 3 SPRING/SUMMER 1979

Tektronix®
COMMITTED TO EXCELLENCE

HANDSHAKE

Newsletter of the Signal Processing Systems Users Group

Focus on Pulse Analysis

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

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

A new name, expanded coverage

Since it began several years ago, the *Software Maintenance Newsletter* has kept users of SPS software products up to date on the latest developments in signal-processing software. Now, coverage has been expanded to include all aspects relating to programming SPS products--hardware, software, and firmware. This is reflected in a new name for the newsletter, *SPS Programming Update*.

Future issues of the *SPS Programming Update* will provide listings of verified problems encountered in current SPS products, information for correcting or bypassing verified problems, listings of documentation corrections, notices of availability of corrected software and firmware, helpful programming hints, and descriptions of new products.

If you are a user of SPS products and would like to receive a sample copy of the *SPS Programming Update*, check the appropriate box on the HANDSHAKE reply card. Since this publication is intended for users of SPS products, please include the serial number of your system or software.

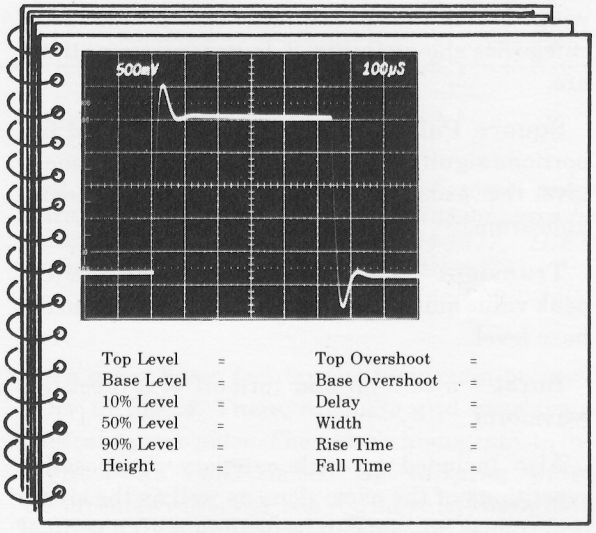
Printed in U.S.A.

Some useful approaches to pulse analysis

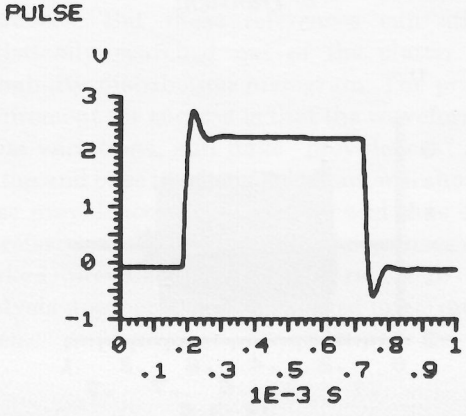
A typical pulse photo with a page from a measurement log is shown at the top of Fig. 1. How long would it take you to fill in that page by studying the photo? Five...ten...maybe fifteen minutes?

You can do it in 20 seconds or less with a Tektronix signal processing system, and the results will carry more accuracy and precision (see bottom of Fig. 1). After the pulse is acquired, the analysis is done automatically, under program control. This means faster measurements, less measurement drudgery, and less chance for human error. Also, with a program doing the bulk of the analysis, the measurement skill required is reduced in many cases to simply setting up the pulse for acquisition. This releases skilled people for more creative work, an important consideration in light of the continuing shortage of technicians and engineers.

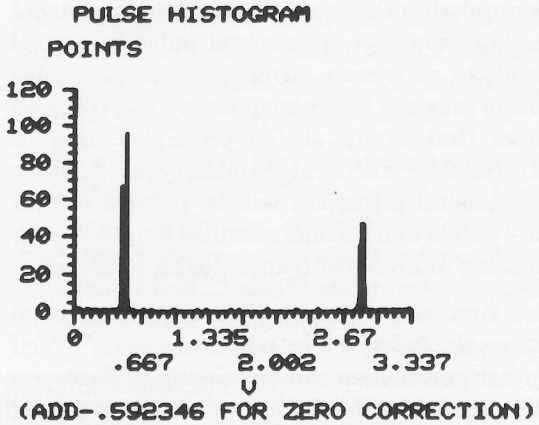
Of course, all these advantages hinge on making the system work. You'll have to write a pulse analysis program, and that requires some signal processing savvy. But, if you've been making the measurements with an oscilloscope, you're already aware of the major concepts. They're part of the normal thought process that occurs as you interpret an oscilloscope display. All you have to do is translate that thinking into a signal analysis program. To help you do this for your own particular needs, let's look at the process for some general analysis cases.



a. A measurement log requires CRT photos and someone to fill in the blanks.



TOP LEVEL = 2.256 V
BASE LEVEL =-.0839465 V
10% LEVEL = .150048 V
50% LEVEL = 1.08602 V
90% LEVEL = 2.022 V
HEIGHT = 2.33994 V
TOP OVERSHOOT = .482328 V
BASE OVERSHOOT= .5084 V



(ADD-.592346 FOR ZERO CORRECTION)

DELAY = 1.91749E-04 S
WIDTH = 5.36058E-04 S
RISE TIME= 1.32496E-05 S
FALL TIME= 1.33198E-05 S

b. In about 20 seconds, the same page of information can be computed by a signal processing system--with a paper copy for your records. Or the results can be archived on a variety of computer storage media.

Fig. 1. *Which approach to pulse analysis would you take?*

Some useful approaches to pulse analysis

What kind of signal?

A pulse is a brief excursion of a quantity from normal. Now, if you think about it, that takes in quite an area--too much for any one processing routine. Different approaches must be taken for different categories of pulses.

So, the first analysis steps are really up to you. You must identify the type of pulse to be dealt with. Usually, it'll fall into one of the three categories shown in Fig. 2. In general terms these are:

Square Pulse -- distinguishable rise and fall portions significantly enough separated in time to give the excursion a squarish or trapezoidal appearance.

Transient -- marked by a rapid transition to a peak value quickly followed by a decay back to the base level.

Burst -- a turn-on and turn-off of a repetitive waveform.

Also included in each category are possible repetitions of the excursions as well as the single occurrence. So, you can be dealing with a train of pulses, too.

Square pulses first

As probably the most frequently encountered type, the square or trapezoidal pulse is a good example to start with. Although your particular interests may not lie with square pulses, it is well to note that many of the principles used in analyzing them also apply to other types of pulses. These general principles will be pointed out for square pulses and become assumed knowledge for discussing analyses of transients and bursts.

The first step in analyzing any pulse is to discover the reference or references from which the other parameters can be computed. These are partly implied by the definition: a pulse is a brief excursion of a quantity from normal. That is, there must be a normal level or a base from which the excursion takes place. That base is one reference. Also, there must be some limit to the excursion before it returns to the base. This limit, itself, or some related point or an average or stable value in the vicinity of the limit is usually chosen as the second reference, referred to as the pulse top. In almost every case, the pulse base (0% level) and pulse top (100% level) are the references from which other pulse parameters are defined or computed.

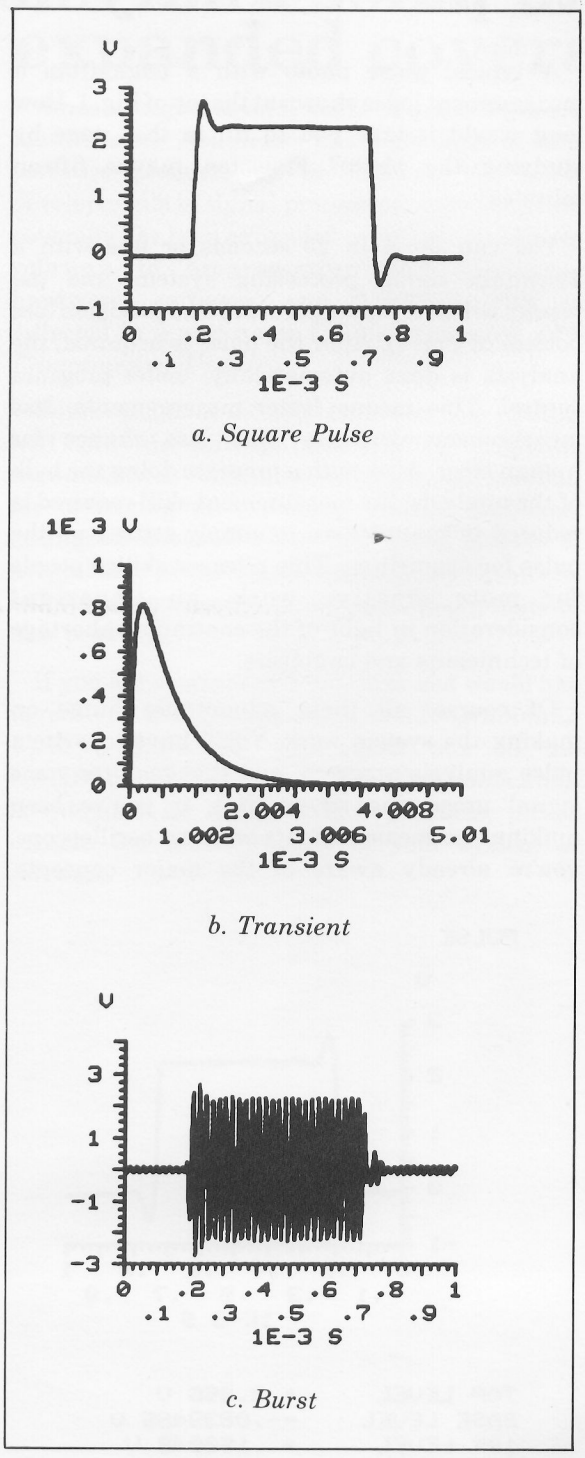


Fig. 2. *Typical examples of three common pulse types.*

For a clearer picture of what might generally be defined as the pulse base and the pulse top, refer to Fig. 3. A variety of other parameters are also defined there. However, note that the base and top of the particular square pulse in Fig. 3 are not necessarily the minimum and maximum of the excursion. Such a coincidence only occurs for ideal square pulses. In the case of Fig. 3, the top and base lines seem to be defined by some normal or average value that excludes noise and various aberrations. This coincides with what most people would visually pick as the top and base when observing a pulse on an oscilloscope. Given this, the question now becomes: "How do you write a signal processing program that simulates that visual selection process?"

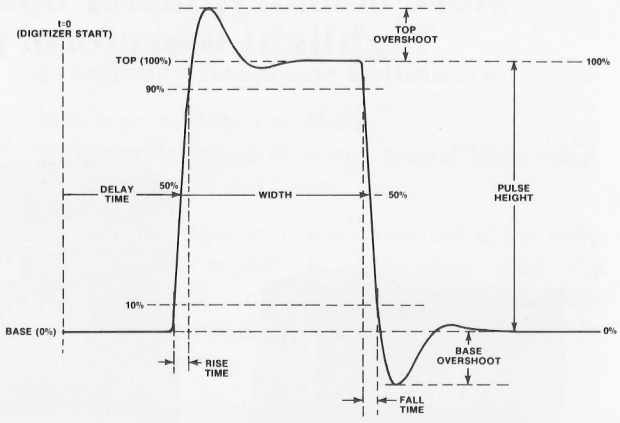


Fig. 3. *Graphical definitions of some pulse parameters. A complete list of definitions can be found in "IEEE Standard 194-1977, IEEE Standard Pulse Terms and Definitions."*

Histograms do for most square pulses

For ideal, undistorted square (or trapezoidal) pulses stored in a signal processing system, using software maximum and minimum functions provides top and base-level information. From these references, the remaining pulse parameters can be derived by simple software routines. Such an ideal approach was discussed in the last issue of HANDSHAKE (Vol. 4 No. 2, Winter 78-79, p. 10) under the title of "Basic pulse analysis."

For most real-life square pulses, however, aberrations tend to cloud the top and base references. But these references can still be statistically searched out of the clutter by a probability distribution histogram. The primary requirement for success is that the waveform, for all its variations, still have "prevalences" about the top and base levels. A histogram will show you these prevalences, if they exist, and thus define the references for further analysis. In cases where marked prevalences are not revealed, other analysis approaches (to be covered later) must be taken.

To get a basic feel for the histogram process, refer to Fig. 4. There, a course grid overlays a typical square pulse. The pulse's histogram, to the right, was constructed by tallying pulse occurrences in each Δ t as you move left-to-right in each level (or cell) of Δ A. In other words, count the squares across each Δ A strip that contain any part of the waveform. Then, at the same Δ A on the histogram grid, draw a bar equal in length to the number of occurrences. This gives the prevalence of that amplitude. Plotting the prevalence for each Δ A produces the histogram shown.

The histogram in Fig. 4 contains two prevalent cells of data (modes). These are extended back to the pulse as shaded strips to indicate regions containing the top and base reference levels: between 22 and 23 amplitude units for the top and between 10 and 11 for the base.

continued on page 14

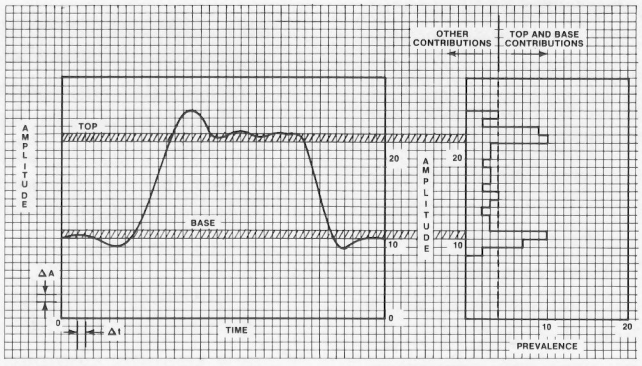
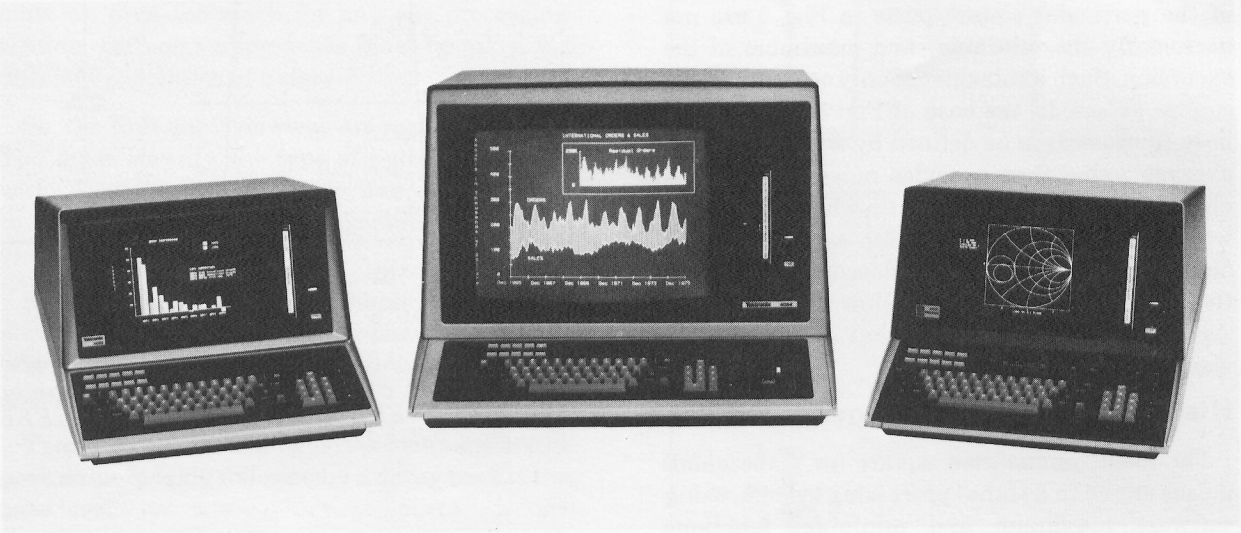


Fig. 4. *Basic histogram construction consists of setting up cells (defined by an overlaying grid), counting by amplitude (Δ A) the cells crossed by the waveform, and recording the count on an amplitude vs. prevalence plot.*

New desktop Graphic Computing Systems --

More memory, faster operation, expanded graphics highlight waveform processing capabilities



The 4050 Series Graphic Computing Systems...processing versatility for business, statistics, engineering, test and measurement.

The 4051 Graphic Computing System is no longer alone. Two new 4050 Series additions, the TEKTRONIX 4052 and 4054 Graphic Computing Systems, now join the 4051 in building a family of fully compatible desktop systems.

Fully compatible means that the 4051, 4052, and 4054 all use the same typewriter-like keyboard, the same high-capacity internal magnetic tape for complete storage compatibility, and the same interfacing scheme. This means that the 4052 and 4054 can also use the existing 4051 peripherals, PLOT 50 software packages, and the 4050 Series Applications Library.

Fully compatible also means you don't have to rewrite your 4051 BASIC programs. They'll run on both the 4052 and 4054. And, when they are run on either of these new additions, you'll notice something really exciting--a startling increase in speed.

A new Tektronix-designed, 16-bit processor based on bit-slice technology is responsible for faster processing. Compared to the 4051, the new processor used in the 4052 and 4054 Graphic Computing Systems gives you a speed advantage of 4 to 40 times, depending on the complexity of the operations performed. For example, the INTEGER function is about 4 times faster while the TAN (tangent) function is about 38 times faster. In terms of waveform processing routines, this can make the difference between "too long for practicality" and "enough time for more complete analysis."

Not only that, but when room for longer programs or multiple waveform storage and manipulation becomes critical, the 4052 and 4054 give you what you need--more memory. Both come standard with 32K bytes of memory. Or, you can pick the optional 64K bytes (56K bytes user accessible).

And, when it comes to displaying your signal analysis results, the 4052 offers the same capabilities enjoyed by 4051 users--only much faster.

If graphic resolution is critical--such as in portraying vibration models and various design simulations--the 4054 offers you a big 19-inch display with 13 million addressable points and 1500 cm/sec constant-velocity vectoring for additional graphic speed. Also, with the 4054, you get stroke-generated characters for added crispness, a variety of character sizes for selectively emphasizing or subordinating alphanumeric data, selectable dot-dash patterns for drawing lines, and a full-screen crosshair cursor. All of this additional graphic capability comes with additional speed--50% faster than the 4052's graphics.

The speed increases in the 4052 and 4054 don't end with just graphics and processing either. Both new Graphic Computing Systems are also GPIB compatible (IEEE 488-1975), and their bus operations are faster. You'll be able to collect waveform data from waveform digitizers about

continued on page 11

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 wide-band 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

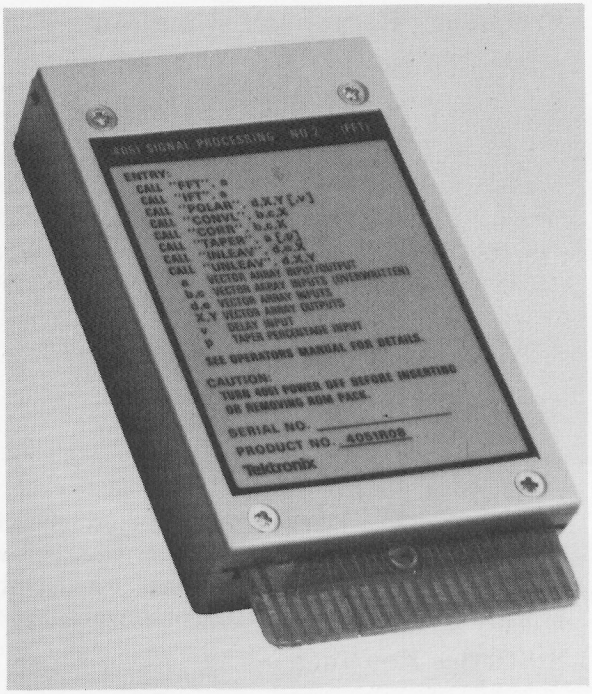
Instruments Required: None.
Software Packages Required: Graphics.
Listing Length: 653 lines.

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.

Signal Processing ROM Pack does an encore



The 4050-Series R08 Signal Processing ROM Pack No. 2 performs advanced signal processing operations such as FFT, IFT, convolution and correlation.

The winter *HANDSHAKE* featured the 4051R07 Signal Processing ROM Pack No. 1, a small Read-Only-Memory device that fits into a ROM pack slot in a TEKTRONIX 4051 Graphic Computing System. As noted previously, the 4051R07 contains basic signal processing functions that allow such computations as the minimum and maximum values of an array, as well as the integration and differentiation of an array. With this issue of *HANDSHAKE*, we announce several new products that provide even more signal processing power.

One of these products is a little ROM pack with a big name: 4050-Series R08 Signal Processing ROM Pack No. 2 (FFT). This new ROM pack takes up where ROM Pack No. 1 left off. It contains advanced signal processing commands for performing fast Fourier transforms, inverse Fourier transforms, convolution, and correlation. In addition, it includes four other commands (UNLEAV, INLEAV, TAPER, and POLAR) that refine or convert the data into other formats. With the 4050-Series R08 ROM pack, a TEKTRONIX Graphic Computing System can now do complicated signal analyses that were previously reserved for larger computers.

Still on the subject of encores, we're also proud to announce that the 4051 Graphic Computing System now has a couple of new additions to the family: the 4052 and 4054 Graphic Computing Systems. The 4051, 4052, and 4054 all speak the same language, allowing applications programs developed on a 4051 to be run on a 4052 or 4054. Also, the contents of all 4051 ROM packs are being incorporated into new ROM packs for the 4052 and 4054--hence the name "4050-Series" in each of the new ROM packs. A separate article in this issue describes the salient features of the 4052 and 4054, but for now, let's take a closer look at the 4050-Series R08.

FFTs in a flash

As with all 4050-Series ROM packs, the 4050-Series R08 features commands that are accessed via a CALL statement (e.g. **CALL "FFT",X** executes the FFT command). Each of the eight commands can be executed in immediate mode directly from the keyboard, or they may be executed as part of a BASIC program.

Operations provided by ROM Pack No. 2 are executed much faster than BASIC programs that perform the same function. For example, the FFT command computes a Fourier transform approximately eight times faster than an equivalent BASIC program. (The ROM FFT of a typical 1024-point waveform requires about 65 seconds on a 4051, and about four seconds on a 4052 or 4054.) An additional benefit is that the ROM pack signal processing routines require no Graphic System memory space. The routines are permanently stored in ROM pack memory, leaving the Graphic System RAM space free for BASIC programs and data.

The following summary lists each of the ROM Pack No. 2 commands and their functions; each command operates on one-dimensional arrays:

FFT -- Computes the fast Fourier transform of an array, placing the result in the same array. The FFT data contains interleaved real and imaginary numbers (rectangular form).

IFT -- Computes the inverse Fourier transform of an array, placing the result in the same array.

CONVL -- Convolves two input arrays, placing the result in a third array.

CORR -- Correlates two input arrays, placing the result in a third array.

POLAR -- Converts an array of FFT data from rectangular form (interleaved reals and imaginaries) to polar form (separate magnitude and phase arrays).

TAPER -- Multiplies an array by a cosine window of program-selectable tapering weights. When 50% tapering is selected, it provides a Hanning window.

UNLEAV -- Sorts an array of interleaved FFT data into two arrays, one containing real components and one containing imaginary components.

INLEAV -- Interleaves real and imaginary data from two input arrays into a third array whose format is acceptable as input to the IFT command.

And applications to match

When equipped with the Signal Processing ROM packs No. 1 and 2, a Tektronix 4050-Series Graphic Computing System becomes a powerful desktop computer for signal analysis applications. The following examples illustrate some of the ways ROM pack No. 2 can be used:

An FFT Application. Since the FFT transforms a signal from the time domain to the frequency domain, the FFT routine can be used to study the frequency spectrum of a signal. For example, in harmonic distortion analysis, you compute the FFT of the signal, then derive the magnitude spectrum via the POLAR routine. Next, the amplitude of each harmonic is determined and the total harmonic distortion (THD) is computed using the equation:

$$THD\% = 100\,\frac{\sqrt{H_1{}^2+H_2{}^2+H_3{}^2+\ \ldots}}{F}$$

where F is the amplitude of the fundamental and H1, H2, and H3 are the amplitudes of the first, second, and third harmonics, respectively. This is a common measurement of amplifier and sound system quality.

An IFT Application. The IFT command does the inverse operation of the FFT command. That is, it transforms a signal from the frequency domain to the time domain. When used in conjunction with the FFT command, IFT can be used to perform very selective digital filtering on a waveform sample. As an example, suppose that the magnitude spectrum of a signal has been obtained and that you wish to attenuate all harmonics of a signal to a certain specified level. This could be done by selectively multiplying all harmonics by a chosen value. Following this procedure, you could then convert the signal back to the time domain by computing the IFT of the resulting spectrum.

A CORR Application. In a practical sense, correlation can be thought of as successively shifting (by some horizontal increment), multiplying, and integrating the two signals to be correlated. From a mathematical standpoint, correlation is achieved by computing the FFT of each signal to be correlated, performing a complex conjugate multiplication on these FFTs, and then taking the IFT of this product. When the two signals being correlated are the same, the process is referred to as *autocorrelation*; when they are different, it is called *cross-correlation*.

A common application for autocorrelation is detecting the presense of signals buried in noise. When a noisy signal is autocorrelated, the result appears as a periodic waveform modulated with a triangular envelope. This technique of signal detection is used in biomedical studies, astronomy, tone control systems, and numerous other applications.

A common application for cross correlation is the detection and ranging of radar, sonar, and other pulsed waveforms, whereby the transmitted and received signals are cross correlated to determine the delay (and hence distance) to a target. Cross correlation is also useful in certain business applications for determining degrees of association between varying quantities (such as sales and pricing). Such studies are useful in predicting future market trends based on past experience.

A CONVL Application. Like correlation, convolution can be thought of as successively shifting, multiplying, and integrating the two arrays (waveforms) to be convolved. However, in the case of convolution, one of the waveforms is reversed in time before performing the shifting-multiplication-integration process. Mathematically, convolution is performed by computing the FFT of each signal to be convolved, multiplying these two FFTs, and then computing the IFT of the resulting product.

A common engineering application for convolution is determining the output of a linear, time-invariant system (such as a passive filter or network). Given the input signal, x(t), and the impulse response, h(t), the output, y(t), can be predicted simply by convolving x(t) with h(t).

continued on page 11

New features added in second version of TEK SPS BASIC

"Why tamper with success?"

That might be your first reaction if you are one of the many satisfied users of TEK SPS BASIC V01 software.

"Was there something wrong with V01?" might be another reaction.

In answer to the latter: No. There was nothing wrong with the first version of TEK SPS BASIC. It has a proven record as a powerful and versatile package for instrument control and signal processing. In fact, it turned out so well that we made the second version similar to the first in most respects--so similar that V02 is really an extension of V01. It, too, provides...

* Full floating-point array processing for single-command operations on digitized signals--including integration, differentiation, fast Fourier transformatio
n, and more

* 32-bit processing with 7-decimal-digit output

* Extensive graphics capabilities

* Single- or multiple-instrument control through special instrument drivers or a GPIB driver which includes direct memory access capability (GPIB is the General Purpose Interface Bus conforming to IEEE Standard 488)

* Modular organization--based on an in-memory or resident monitor augmented by callable nonresident command--for memory conservation as well as future expansion or addition of your own unique routines and commands

Of course, these aren't all of the features common to both versions. But they are some of the major ones contributing to the success of the first version of TEK SPS BASIC, which brings us back to the original question, "Why tamper with success?"

The answer: Field experience suggested needs for faster input/output, more capabilities, and more flexible instrument control. So TEK SPS BASIC V02 was designed to include...

*** A more memory-efficient GPIB driver.**
Commonly used GPIB routines are now part of the TEK SPS BASIC V02 resident monitor so that they can be shared by the GPIB driver and nonresident commands. The result--an overall memory savings for most interfacing tasks.

*** Faster program overlaying.**
Two new overlaying commands speed--up to eight times faster than previously--overlaying of subprograms. In a program operating by loading (overlaying) and execution of many sub-programs, this results in a dramatic reduction in program run time.

*** Faster command and driver access.**
Rather than searching peripheral storage for nonresident command and driver files, the file addresses are now stored with the resident monitor. This dramatically cuts file search time, with the result that programs using numerous nonresident files run considerably faster on flexible disk systems. Hard disk systems also realize increased program running speeds.

*** Versatile file access.**
Data files can now be accessed and manipulated at any level from byte to block. They can be read or written in ASCII or binary. They can be formatted or unformatted. And they can be accessed serially or randomly.

The choice is yours.

*** New terminal driver features.**
The terminal driver has been separated from the resident monitor. This allows specialized drivers for each type of terminal-- TEKTRONIX 4010 Graphic Terminals, non-graphic terminals, and TV-type terminals.

Also, you can now enter commands or program lines while a program is running. You can, for example, print out variable values at any time, and you can enter responses to INPUT requests before the request is even made.

*** More graphics commands.**
Several new commands have been added to the graphics package. Most notable is XYPLOT, which plots one array against another. This is a definite boon for anyone involved in hysteresis studies.

Other added commands are SEEVIEW and SEEWINDOW, which give you previously defined VIEWPORT and WINDOW coordinates.

*** Time and date capability.**
You can now set time and schedule events or tasks according to time. Also, the date can be set and stored automatically whenever a file is created.

*** More High-Level Support Package commands.**
The High-Level Support Package gives you the ability to program bit-level changes with BASIC commands instead of writing the assembly language routines normally required by other software systems. This significantly eases the task of writing your own custom instrument and peripheral drivers.

Now--with the new VARCLR, VARSET, and VARTST commands--you can do this with even greater ease and flexibility. These new commands let you clear, set, or test bit patterns of variables. Also, another new command, RSTBUS, lets you send an INIT pulse to the external bus.

*** Tasking and error control.**
In a multi-instrument atmosphere, your test and measurement programs shouldn't come to a screeching halt just because one of the instruments does. With TEK SPS BASIC V02, they won't.

New task-definition and error-control features let you decide how to respond to instrument interrupts or errors, including abandoning some parts of the program and continuing with others.

*** Multiple power-fail protection.**
When nonvolatile memories are used, TEK SPS BASIC V02 protects your programs and data from loss due to single, even multiple power failures. When power returns to a steady state, V02 comes back up ready to run.

When you add it all up, we think you'll agree that it comes to more than tampering with success. It's taking a proven concept and building on it to provide you with the broadest possible software coverage for your test and measurement needs. To see how much measurement coverage you can get, call your local Tektronix Field Office or sales representative.

continued from page 6

New desktop Graphic Graphic Computing Systems --

three times faster than is possible with the 4051 Graphic Computing System.

Still, the 4051 is the first choice where low-cost instrument control takes precedence over added memory and faster analysis. But, if you do need more speed and memory for quicker waveform processing, the 4052 becomes the new choice, or the 4054 if increased graphics resolution is an additional requirement.

To find out more about any of these three desktop controllers, contact your local Tektronix Field Office or sales representative and ask about the 4050 family of Graphic Computing systems. For data sheets, check the appropriate box on the HANDSHAKE reply card.

continued from page 9

Signal Processing ROM Pack does an encore

A powerful duet

The Signal Processing ROM pack No. 2 works in concert with ROM pack No. 1 to bring about a wide range of signal-processing applications. Another article in this issue (on pulse analysis) suggests a few possible applications, but it really only scratches the surface.

If you would like more information on the 4050-Series Graphic Computing Systems and their signal processing ROM packs, please check the appropriate boxes on the user reply card. Or better yet, contact your nearest Tektronix field office. A Tektronix field engineer will be happy to stage a live performance of the 4050-Series ROM packs on a new 4050-Series Graphic Computing System. We think you'll agree that the 4050-Series R08 is an encore worth waiting for.

By Cliff Morgan,
HANDSHAKE Staff

Getting the most out of TEK BASIC graphics

X-Y-Z plotting adds another dimension to analysis results

Three dimensional or X-Y-Z plots can often expand your measurement perception. For example, spectrum analysis is useful in analyzing vibrations and resonances in a variety of things, from rotating machinery to loudspeakers. But even more can be learned by noting how a spectrum changes with time, how patterns occur intermittently or evolve with time.

A TEK SPS BASIC program for doing such three-axis plots is listed with this article (Fig. 1). It's capable of handling any number of waveforms or arrays up to 100 and produces an orthogonal projection of sufficient accuracy for visual analysis.

The data for the program can come from either a waveform digitizing instrument or from a simulation routine. In either case, you'll need to add waveform acquisition or generation routines to the program. All of the waveforms or traces to be plotted must be stored in a single file, and you'll need to keep track of the number of stored traces for later input into the plotting program.

The program first asks for the name of the file in which the data is stored. It assumes the file is stored on the system device. It then asks for the number of traces to be projected. Next, some sage advice is given on selecting proper angles for a pleasing picture, and you are then asked to input these angles. The program does not pass judgement on the inputs, and if the input angles would cause off-screen plotting, the program does not prevent it.

At this point, the input angles are converted to radians, and the transforms for the X and Y arrays are computed (tanØ and tanθ). The X, Y, and two hidden-line arrays are set up, and then the maximum and minimum Y values are found. In this program, X ranges from 0 to 511 because no data acquisition is done. However, you can easily change the program to reflect the proper values of X.

A trace is read in and scaled to show up better (line 310) in cases where the Y values are much smaller or larger with respect to the X values. The X and Y arrays are then converted to screen points and sheared appropriately (lines 330 to 390). If this is the first or last trace, either the first or last points are saved to set up the axes later. The

```
 1 REM          ****ORTHOG.PRJ****
 3 REM  A PROGRAM IN TEK SPS BASIC TO DO ORTHOGRAPHIC PROJECTIONS
 4 REM  OF UP TO 100 512 POINT ARRAYS OR WAVE TRACES.
 6 REM  INPUTS:
 7 REM      A$--NAME OF THE DATA FILE WHERE THE DATA TO BE GRAPHED
 8 REM          IS STORED.  THERE SHOULD BE Z TRACES OF 512
 9 REM          POINTS EACH.
10 REM     Z--NUMBER OF TRACES TO BE PROJECTED 1<=Z<=100
11 REM     PHI--Y SHEAR ANGLE IN DEGREES -89<=PHI<=89
12 REM     THETA--X SHEAR ANGLE IN DEGREES -89<=THETA<=89
14 REM OUTPUT:
15 REM     AN ORTHOGRAPHIC PROJECTION OF THE DATA FILE ON THE DISPLAY
16 REM         SCREEN WITH APPROPRIATE AXES AND LABELING.
19 CLOSE #1
20 PRINT "INPUT NAME OF DATA FILE"\INPUT A$
30 OPEN #1 AS A$ FOR READ
40 PRINT "INPUT NUMBER OF TRACES WANTED"
50 INPUT Z
51 PRINT " FOR A SMALL NUMBER OF TRACES; LIKE UNDER 40 , YOU"
52 PRINT " SHOULD SPACE THE LINES FURTHER APART BY INCREASING THE"
53 PRINT " Y SHEAR ANGLE. DO NOT; HOWEVER, LET THE X OR Y SHEAR ANGLES"
54 PRINT " EXCEED PLUS OR MINUS 89 DEGREES!!!"
60 PRINT " INPUT PHI (Y SHEAR ANGLE) AND THETA (X SHEAR ANGLE)"
70 INPUT PH,TH
75 REM CONVERT PHI AND THETA TO RADIANS
80 PH=PH*2*3.14159/360
90 TH=TH*2*3.14159/360
95 REM THE SHEAR ANGLE IS THE TANGENT OF PHI OR THETA, RESPECTIVELY
100 TX=SIN(TH)/COS(TH)
110 TY=SIN(PH)/COS(PH)
120 REM SET UP HIDDEN LINE AND WORKING ARRAYS
130 DIM X(511),Y(511),WH(1023),H(1023)
150 REM  FIND DATA MAX AND MIN
170 GOSUB 820
190 REM  INITIALIZE HIDDEN LINE ARRAYS
200 WH=0
210 H=0
230 REM  START DATA READ IN AND CONVERSION
250 RESET #1
260 PAGE
270 FOR Z1=1 TO Z
280 Z2=Z1-1
290 X=1\INT X,X
300 READ #1,Y
310 Y=Y*200/(YH-ZZ)
320 REM  CONVERT X AND Y TO SCREEN COORDINATES
330 IF TX<=0 THEN GOTO 350
340 X=X+100+TX*Z2*3\GOTO 360
350 X=X+400+TX*Z2*3
360 Y=390/512*Y+100+TY*Z2*3
386 REM  PULL OUT SCREEN POINTS FOR PROPER PLACEMENT OF AXES
390 IF Z1=1 THEN GOSUB 1030
400 IF Z1=Z THEN GOSUB 1110
410 SMOVE X(0),Y(0)
420 REM  DISPLAY LOOP
430 WH=H
440 FOR I=1 TO 511
450 XM=I-1
460 IF Y(XM)<=H(X(XM)) THEN GOTO 490
470 IF Y(I)>H(X(I)) THEN GOTO 540
480 IF Y(I)<=H(X(I)) THEN GOTO 600
490 IF Y(I)<=H(X(I)) THEN GOTO 740
500 IF Y(I)>H(X(I)) THEN GOTO 700
520 REM  OUTSIDE DRAWING TO OUTSIDE
540 SDRAW X(I),Y(I)
550 WH(X(I))=Y(I)
560 GOTO 740
580 REM  OUTSIDE DRAWING TO INSIDE
600 GOSUB 940
610 SDRAW XI,YI
620 GOTO 740
640 REM  INSIDE DRAWING TO INSIDE
660 REM       SKIP IT
680 REM INSIDE DRAWING TO OUTSIDE
700 GOSUB 940
710 SMOVE XI,YI
720 SDRAW X(I),Y(I)
730 WH(X(I))=Y(I)
740 NEXT I
750 H=WH
760 NEXT Z1
766 REM PUT IN AXES AND LABELING
770 GOSUB 1180
780 END
800 REM  FIND DATA MIN AND MAX
820 YH=-1000000
830 YL=1000000
```

```
840 FOR I=1 TO Z
850 READ #1,Y
856 REM SAVE A POINT FOR AXES LABELING
860 IF I=1 THEN GOSUB 1620
870 IF YH<MAX(Y) THEN YH=MAX(Y)
880 IF YL>MIN(Y) THEN YL=MIN(Y)
890 NEXT I
900 RETURN
920 REM  FIND INTERMEDIATE POINT
940 S1=(H(X(I))-H(X(XM)))/(I-XM)
950 S2=(Y(I)-Y(XM))/(I-XM)
960 XI=(H(X(XM))-S1*XM-Y(XM)+S2*XM)/(S2-S1)
970 YI=S2*XI+Y(XM)-S2*XM
980 IF TX<=0 THEN GOTO 1000
990 XI=XI+100+TX*Z2*3\GOTO 1010
1000 XI=XI+400+TX*Z2*3
1010 RETURN
1020 REM SAVING SCREEN LOCATIONS FOR AXIS PLACEMENT
1030 DIM A(1),B(1)
1040 IF TX<=0 THEN GOTO 1060
1050 A(0)=X(511)\GOTO 1070
1060 A(0)=X(0)
1070 A(1)=Y(0)
1080 AA=A(1)-MIN(Y)
1090 RETURN
1100 REM ENTRY FOR LAST LOCATION SAVE IS HERE
1110 IF TX<=0 THEN GOTO 1130
1120 B(0)=X(511)\GOTO 1140
1130 B(0)=X(0)
1140 B(1)=Y(0)
1150 BB=MAX(Y)
1160 RETURN
1170 REM DRAWING THE AXES
1180 IF TX<=0 THEN GOTO 1210
1190 SMOVE A(0)-511,A(1)-AA
1200 GOTO 1220
1210 SMOVE A(0)+511,A(1)-AA
1220 SDRAW A(0),A(1)-AA
1230 SDRAW A(0),A(1)
1240 SMOVE A(0),A(1)-AA
1250 SDRAW B(0),B(1)-AA
1260 SDRAW B(0),BB
1270 SMOVE B(0),B(1)
1280 SDRAW A(0),A(1)
1290 REM ADDING IN UNITS
1310 REM Z AXIS--TIME
1320 DI=5
1330 IF Z<20 THEN DI=3
1340 DX=(B(0)-A(0))/DI
1350 BC=(B(1)-A(1))/DI
1360 Q=ITP(Z/DI)
1370 FOR I=0 TO DI
1380 SMOVE A(0)+I*DX,A(1)-(AA+4)+I*BC
1390 PRINT "-"
1391 SP=-30
1392 IF TX>0 THEN SP=5
1394 SMOVE A(0)+I*DX+SP,A(1)-(AA+7)+I*BC
1395 PRINT Q*I
1396 NEXT I
1397 SP=-70
1398 IF TX>0 THEN SP=35
1399 SMOVE A(0)+SP,A(1)-AA-12\PRINT "TRACE"
1401 REM X AXIS--FREQUENCY
1409 DX=102.4
1410 IF TX>0 THEN A(0)=A(0)-511
1420 FOR I=0 TO 5
1430 SMOVE A(0)+DX*I,A(1)-AA
1440 SDRAW A(0)+DX*I,A(1)-(AA+7)
1450 SMOVE A(0)+DX*I-14,A(1)-AA-20
1460 PRINT I*DX
1470 NEXT I
1480 SMOVE A(0)+2*DX-14,A(1)-AA-40\PRINT "POINTS PER TRACE"
1490 REM Y AXIS--AMPLITUDE
1500 Q=YH-ZZ\QQ=ITP(Q*100/8)/100
1510 ZZ=ITP(ZZ*100)/100
1520 BC=(BB-B(1)+AA)/8
1530 FOR P=0 TO 8
1540 SP=-80
1550 IF TX>0 THEN SP=35
1560 SMOVE B(0),BC*P+B(1)-(AA+4)\PRINT "-"
1570 SMOVE B(0)+SP,BC*P+B(1)-(AA+4)
1580 PRINT ZZ+QQ*P
1590 NEXT P
1591 SMOVE B(0)+SP,BB+10\PRINT "AMPLITUDE"
1600 RETURN
1610 REM GETTING VALUE FOR AXES
1620 ZZ=MIN(Y)
1630 RETURN
```

Fig. 1. Program for orthogonal projection of up to 100 waveform arrays.

determination of which points are saved depends upon whether X is sheared to the right or left.

The display loop is then started. The points are checked against the hidden-line array to see if they should be displayed or not. The decision to display or not display a line depends upon which of the following test cases apply:

1) **Outside drawing to outside:** the line is drawn, and the hidden line is updated.

2) **Outside drawing to inside:** an intersecting point between the array and hidden-line array must be found and drawn to.

3) **Inside drawing to inside:** nothing is done.

4) **Inside drawing to outside:** the point of emergence is found, and a line is drawn from the point of emergence to the current point.

These tests are repeated for all points in all traces to be drawn.

The last actions of the program are to plot the axes and print the accompanying units. Figure 2 shows an example of program output for a simulated data plot of 100 traces. Again, actual units are not acquired by this program, so some default units are used (POINTS PER TRACE, TRACE, and AMPLITUDE). However, you can easily change this.

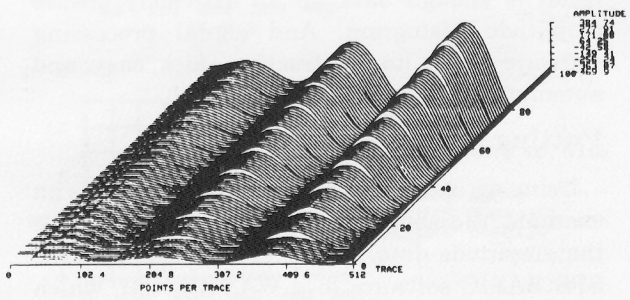


Fig. 2. 100 traces of simulated signal data plotted by the program listed in Fig. 1.

continued on page 24

continued from page 5

Some useful approaches to pulse analysis

For analysis purposes, however, the statistics need to be narrowed to single top and base values--not regions. Statistically, there's equal probability of the actual levels being to either side of the cell midpoint. So, going just on probability, you could expect the cell midpoints to be good guesses for a top and base. This gives 22.5 for the top and 10.5 for the base in Fig. 4. But, looking at the pulse, our eyes tell us that this is not as good a guess as we might make in this instance. Something closer to 10 for the base and 23 for the top would seem more reasonable.

There are at least two ways to get closer. One is to remove contributions of transitions and aberrations from the histogram (done in Fig. 4 by subtracting a value of 4) and compute the means of the two isolated modes (22.95 for the top and 10.1 for the base). Another way is to construct a finer-grained histogram. You can verify the effect yourself by halving Δ t and Δ A in Fig. 4 and reconstructing the histogram. The finer grid moves the modes closer to the actual top and base.

A still finer scale is available with a signal processing system. Most digitizers divide the time axis into at least 512 increments. That's substantially finer than the 40 divisions shown in Fig. 4. Also, at least the same and often greater precision can be provided for amplitude. The result is enough data for an extremely precise amplitude histogram. And signal processing software makes its construction quick, easy, and automatic...with no square counting!

Putting it into a program

Using an actual pulse (left side of Fig. 1b) as an example, the signal processing system digitizes the amplitude data. The data is stored by TEK SPS BASIC software in a WAVEFORM, which consists of an array for the data along with scale-factor and units variables. For this particular case, the WAVEFORM is referred to as WA, and its associated array, array A, contains 512 data elements with indexing from 0 to 511. The following TEK SPS BASIC routine goes through the data in A and, in another WAVEFORM referred to as WB, constructs an amplitude histogram, complete with appropriate scaling (right side of Fig. 1b).

```
 5 WAVEFORM WA IS A(511),HA,HA$,VA$
10 WAVEFORM WB IS B(511),HB,HB$,VB$
 .
 .
 .
100 REM HISTOGRAM ROUTINE
105 REM FORM HISTOGRAM OF WA IN WB
110 DIM X(511)\LET B=0
115 LET MI=MIN(A)\ LET MA=MAX(A)
120 LET VS=(MA-MI)/10
125 LET X=A-MI
130 LET X=51.1*X/VS
135 FOR I=0 TO 511
140 LET B(X(I))=B(X(I))+1
145 NEXT I
150 LET VB$="POINTS"
155 LET HB$=VA$
160 LET HB=VS/51.1
```

The routine starts in line 110 by defining an X array for intermediate work and setting array B of WB to zero. Next, in line 115, it finds the minimum and maximum values of the pulse stored in A of WA. Using the maximum (MA) and minimum (MI), line 120 computes a vertical scale (VS) based on fitting ten divisions over the pulse height (MA-MI). The selection of ten divisions, rather than six or eight, has to do with fitting the data to the ten horizontal divisions normally used in data display. The full import of this will become clearer shortly. After computing ten-division scaling for the pulse height, the program subtracts the minimum (MI) from the pulse data (A) and stores the result in interim array X. This is done in line 125 and shifts all the pulse amplitude data to positive values ranging from zero to MA-MI. Then in line 130, using 51.1 for elements per division and the ten-division scaling (VS), the pulse amplitude data in X is converted to values for tallying into histogram array B by the FOR loop operation in lines 135, 140, and 145. This loop steps through the X array using its converted element values (ranging from 0 to 511) as indices for array B. A count is incremented at each element in B corresponding to the vertical value in X. The result after 512 iterations is an amplitude histogram in array B. Its scaling is set by the operations in lines 150, 155, and 160, where VB$ is the vertical units, HB$ is the horizontal units, and HB is the horizontal element scaling. This horizontal scaling does not take into account the previously subtracted minimum (MI), so amplitude readings taken from it must be corrected by adding MI to the scale value.

To compute the parameters listed in Fig. 1b and format the results for output in the manner of Fig. 1b, this histogram routine is incorporated into the program listed in Fig. 5. Lines 5 and 10 set up the WAVEFORMs WA for the pulse to be analyzed and WB for the histogram. Add an acquisition routine between lines 10 and 100 for the waveform

```
10 REM WAVEFORM ACQUISITION
15 WAVEFORM WA IS A(511),HA,HA$,VA$
20 WAVEFORM WB IS B(511),HB,HB$,VB$
100 REM HISTOGRAM ROUTINE
105 REM FORM HISTOGRAM OF WA IN WB
110 DIM X(511)\LET B=0
115 LET MI=MIN(A)\LET MA=MAX(A)
120 LET VS=(MA-MI)/10
125 LET X=A-MI
130 LET X=51.1*X/VS
135 FOR I=0 TO 511
140 LET B(X(I))=B(X(I))+1
145 NEXT I
150 LET VB$="POINTS"
155 LET HB$=VA$
160 LET HB=VS/51.1
165 REM FIND TOP AND BASE
170 LET CE=256
175 LET V1=CRS(B(0:CE),MAX(B(0:CE)))
180 LET V2=CRS(B(CE:511),MAX(B(CE:511)))
185 LET VB=V1*HB+MI
190 LET VT=V2*HB+MI
200 REM COMPUTE REMAINING PARAMETERS
205 LET OT=MA-VT\LET OB=VB-MI
210 LET PH=VT-VB\LET V1=.1*PH+VB
215 LET V5=.5*PH+VB\LET V9=.9*PH+VB
220 LET T1=CRS(A,V1)*HA
225 LET T2=CRS(A,V5)*HA
230 LET T3=CRS(A,V9)*HA
235 IF T1<T3 THEN 250
240 LET VX=VT\LET VT=VB\LET VB=VX
245 LET OT=VT-MI\LET OB=MA-VB\GOTO 210
250 LET TX=T3/HA
255 LET T4=TX
260 LET TX=CRS(A(TX+1),V9)
265 IF TX<0 THEN GOTO 275
270 GOTO 255
275 LET T5=CRS(A(T4),V5)*HA
280 LET T6=CRS(A(T4),V1)*HA
285 LET T4=T4*HA
290 LET RT=T3-T1\LET FT=T6-T4
295 LET PW=T5-T2
300 REM OUTPUT DATA SHEET
305 PAGE\PRINT "PULSE"
310 SMOVE 600,760\PRINT "PULSE HISTOGRAM"
315 VIEWPORT 100,400,500,700\SETGR GRAT 2,2,VIEW\GRAPH WA
320 VIEWPORT 600,900,500,700\SETGR GRAT 2,2,VIEW\GRAPH WB
325 SMOVE 550,405\PRINT "(ADD";MI;" FOR ZERO CORRECTION)"
330 LET S$=" "\SMOVE 0,360
335 PRINT "TOP LEVEL      =";VT;S$&VA$
340 PRINT "BASE LEVEL     =";VB;S$&VA$
345 PRINT "10% LEVEL      =";V1;S$&VA$
350 PRINT "50% LEVEL      =";V5;S$&VA$
355 PRINT "90% LEVEL      =";V9;S$&VA$
360 PRINT "HEIGHT         =";PH;S$&VA$
365 PRINT "TOP OVERSHOOT  =";OT;S$&VA$
370 PRINT "BASE OVERSHOOT=";OB;S$&VA$
375 SMOVE 600,360\PRINT "DELAY     =";T2;S$&HA$
380 SMOVE 600,340\PRINT "WIDTH     =";PW;S$&HA$
385 SMOVE 600,320\PRINT "RISE TIME=";RT;S$&HA$
390 SMOVE 600,300\PRINT "FALL TIME=";FT;S$&HA$
395 SMOVE 0,0\END
```

Fig. 5. *TEK SPS BASIC program for analyzing trapezoidal pulses. This program produces the formatted output shown in Fig. 1b.*

digitizing instrument you are using, making sure that the digitized pulse is stored in WA, and you have a complete pulse acquisition and analysis program.

Lines 100 through 160 are the histogram routine that has already been described. Briefly, lines 165 through 190 use the cross function (CRS) to find the top and base from the histogram. Lines 200 through 295 find or compute the remaining parameters, and lines 300 through 395 format the output to the graphics terminal. Further insight into program functioning is best gained by actually stepping through it line-by-line with a test pulse and observing the results of each operation.

A look at slew rate

The program in Fig. 5 computes some of the more common pulse parameters. It certainly doesn't take in the full gamut of what can be computed, though. For example, the pulse under analysis might be one associated with testing a device where response to a rapid voltage change is of interest, and you might prefer to look at the leading transition in terms of slew rate (rate of rise) instead of rise time.

As a simple first approach, slew rate can be computed from the parameters already provided by the program in Fig. 5. For example, the results shown in Fig. 1b for rise time and 10% and 90% levels can be used to compute the slew rate for that pulse by

Slew Rate = (90% level - 10% level)/rise time
= (2.022 V - .150048 V)/1.32496E-5 S
= 141.28 KV/S

However, this simple approach assumes the transition is linear, which may or may not be the case. If you prefer to forego such an assumption, zone differentiation gives you a detailed look at the slope. An example is shown in Fig. 6.

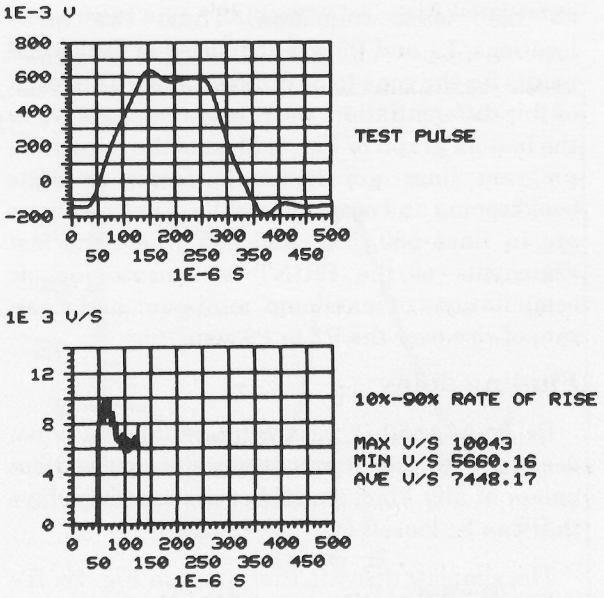


Fig. 6. *For a detailed look at rate of rise, the test pulse (top) is differentiated over a zone extending from the 10% to 90% points. The result is point-by-point rate of rise (bottom) which can be analyzed further.*

Getting the most out of TEK BASIC graphics

Computation of the Fig. 6 data can be done with the following program lines which, if you choose, can be added to the program listed in Fig. 5. In fact, the following program lines depend on running the Fig. 5 program first for preliminary data.

```
500 REM COMPUTE AND OUTPUT RATE-OF-RISE DATA
505 LET B=0
510 LET P1=CRS(A,V1)
515 LET P9=CRS(A,V9)
520 DIFF A(P1:P9),B(P1:P9)
525 LET B=B/HA
530 LET HB=HA
535 LET HB$=HA$
540 LET VB$=VA$&"/"&HA$
545 PAGE
550 VIEWPORT 100,400,500,700
555 SETGR VIEW \ GRAPH WA
560 VIEWPORT 100,400,150,350
565 SETGR VIEW \ GRAPH WB
570 SMOVE 500,600 \ PRINT "TEST PULSE"
575 SMOVE 500,300 \ PRINT "10%-90% RATE OF RISE"
580 SMOVE 500,250 \ PRINT "MAX ";VB$;MAX(B(P1:P9))
585 SMOVE 500,230 \ PRINT "MIN ";VB$;MIN(B(P1:P9))
590 SMOVE 500,210 \ PRINT "AVE ";VB$;MEA(B(P1:P9))
595 SMOVE 0,0 \ EMD
```

Line 505 sets array B of WB to zero. This is the array that receives the derivative of the rising transition. In 510, the location of the 10%-level on the transition, P1, is found by using the cross function on the digitized pulse in A and the 10% level (V1) computed previously. The same thing is done in 515 to find P9, the location of the 90% point on the rising transition. These two array locations, P1 and P9 are then used in 520 as end points for the zone to be differentiated. The result of this differentiation, the rate of rise, is shown in the bottom graph of Fig. 6. Most of the remaining program lines are for scale factor or units bookkeeping and output of results. The exceptions are in lines 580, 585, and 595 where the last arguments of the PRINT statements include computations of maximum, minimum, and mean rate of rise over the P1 to P9 zone.

Finding delay

Delay is another parameter of interest when dealing with pulses, not just square pulses, but pulses of any kind. And there are several delays that can be looked at.

The simplest delay is that listed in Fig. 1b. It's just the time from a triggering event or other selected reference to the 50% point on the leading transition.

Another delay of frequent interest is that between two pulses, generally an input pulse and a resulting output pulse such as occur in switch operations. The general data acquisition procedure (Fig. 7) is to use a dual-channel or dual-instrument setup where both the input and output

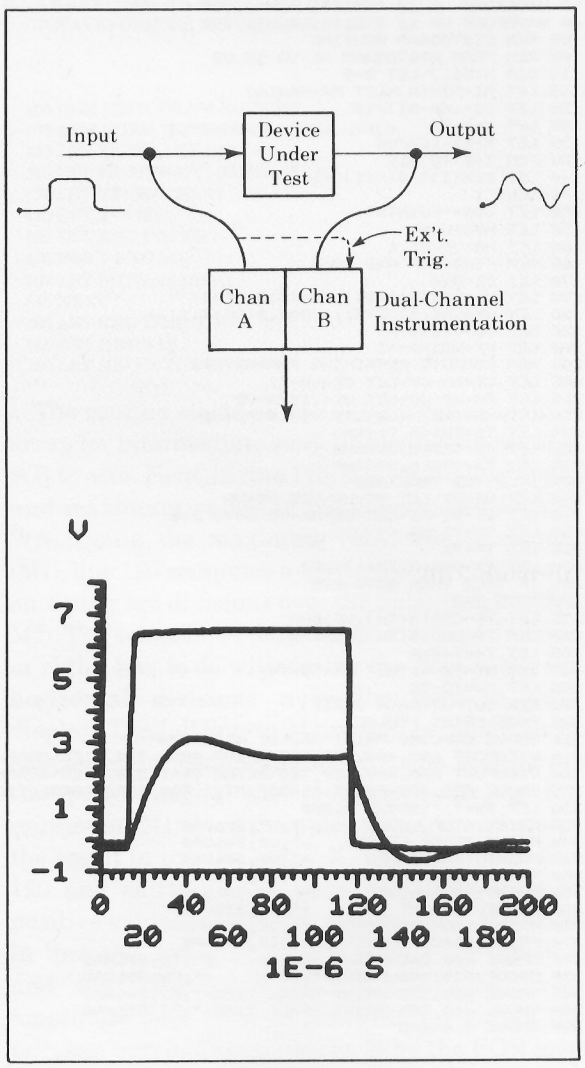


Fig. 7. *Dual-channel acquisition of input and output signals while maintaining their time relationship requires common triggering for both channels. For some instrumentation, hardwire connection of the A-channel signal to the B-channel external trigger input provides a solution. Other instruments provide a selectable "B triggered by A" mode of operation. In still other cases, dual-channel operation is achieved under system software control.*

pulse acquisitions are triggered by the same event. Frequently, the input pulse triggers acquisition for itself as well as the output pulse. Whatever triggering scheme is used, though, the primary goal should be maintenance of the time relationship between the pulses throughout acquisition, digitizing, and storage. The exact procedure for doing this depends upon the instruments and software being used.

Once the signals are stored in arrays, they can be individually analyzed for their parameters by the program listed in Fig. 5. When time relation is maintained, the difference between individual pulse delays is the delay between pulses. These individual delays can be defined at 10% points, 50% points, or any other point appropriate to the particular application. In the case of the Fig. 5 program, delay is found at the 50% point (the cross of array A and V5) at line 240. The T1 cross (10%) in line 235 or the T3 cross (90%) in line 245 could be used instead. Also, other time relationship parameters--pulse spreading, storage time, etc.--can be computed by additional program lines. These parameters depend also upon trailing-edge references which are computed in the Fig. 5 program by lines 275 (T5, 50% cross), 280 (T6, 10% cross), and 285 (T4, 90% cross computed by 250 through 270).

Correlation -- a different look at delay

In some cases--radar, sonar, and ultrasonics, for examples--the delay of interest may be tied to return pulse energy rather than to a point on the pulse transition. In such cases, correlation offers a quick computational approach for delay. It's an approach that doesn't depend upon building histograms or finding tops and bases. In short, it can be used on any pair of pulses, no matter what their shape.

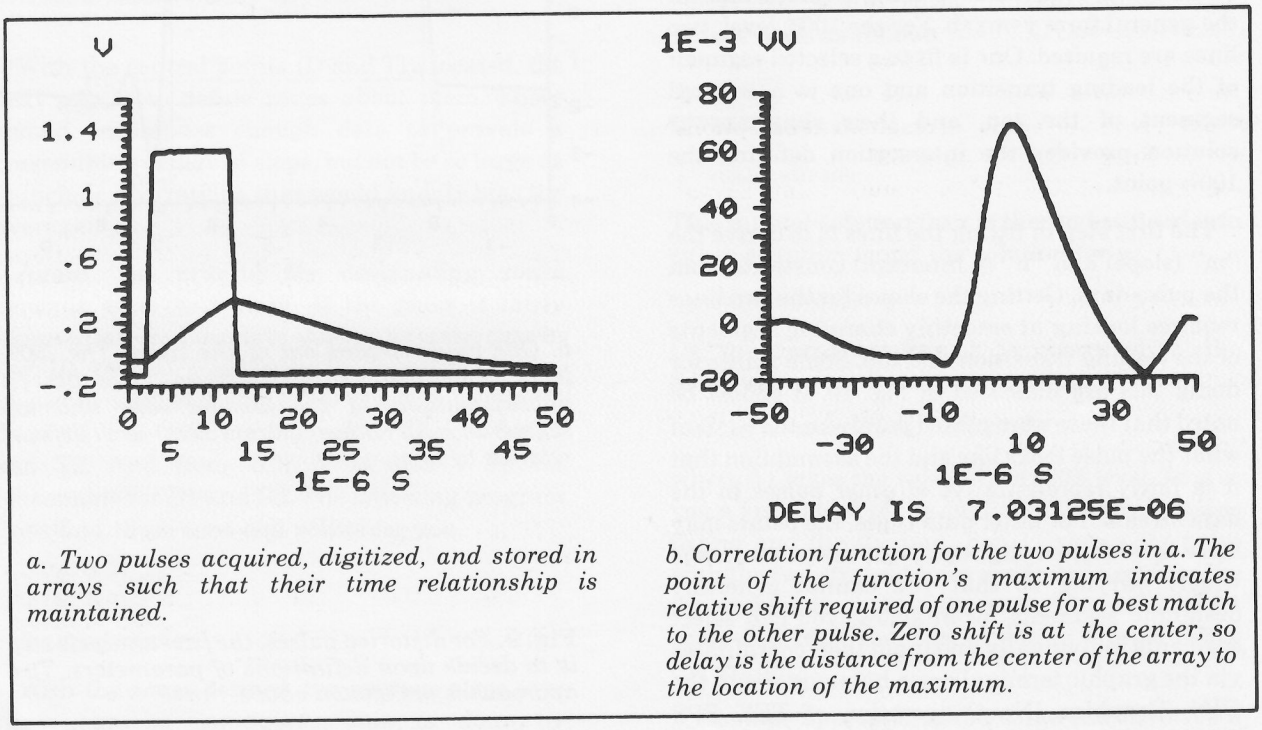
As an example, in Fig. 8a there are two pulses acquired so that their time relationship is maintained. Using a software correlation routine on these pulses returns the correlation function shown in Fig. 8b.

The actual software process of correlation involves the fast Fourier transform (FFT) and a frequency domain multiplication. But this corresponds, in time, to incrementally sliding one pulse past the other, multiplying them at each increment of shift, and finding the area (integrating) after each multiplication. The plot of area vs. shift is the correlation function, and the location of its maximum gives delay in terms of area (or energy since the area contained by a pulse is proportional to energy).

Other types of pulses

Up to this point, the discussion has centered mostly on square pulses. This is because, with the exception of correlation, the analyses have depended upon defining top and base references with a histogram. Doing this depends, in turn, upon the pulse having prevalent data about a top and base level. In other words, it has to be relatively square in shape.

But what happens when the pulse isn't square, when there aren't data prevalences to define a top and base?



a. Two pulses acquired, digitized, and stored in arrays such that their time relationship is maintained.

b. Correlation function for the two pulses in a. The point of the function's maximum indicates relative shift required of one pulse for a best match to the other pulse. Zero shift is at the center, so delay is the distance from the center of the array to the location of the maximum.

Fig. 8. *Correlation is a method that can be used to determine the delay between any two waveforms in terms of their maximum common area (or energy).*

Some useful approaches to pulse analysis

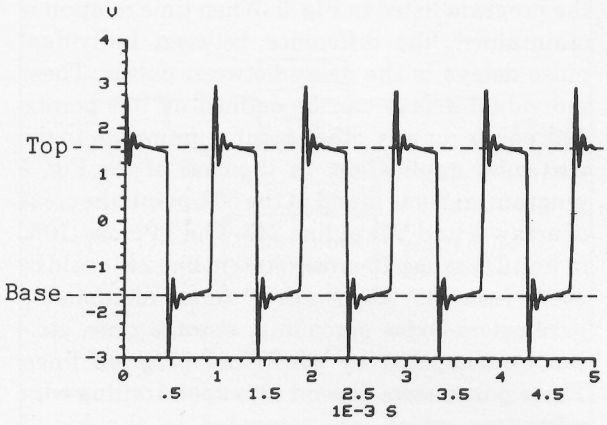
What happens is that the majority of analysis techniques discussed thus far can still be used. Finding rise times and so forth does not depend upon pulse shape. It depends upon having defined top and base values for computing and finding 10% and 90% points.

So, when the histogram approach isn't valid or when the definitions don't fit in with it, take another approach to finding the references. This being a frequent enough requirement, let's look briefly at some other pulse types and how you might go about defining a 100% level.

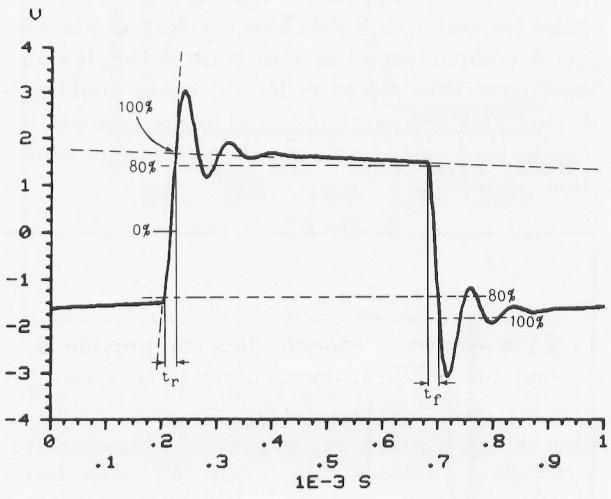
Data pulse. The pulse stream shown in Fig. 9a is typical of pulse data that has undergone transmission distortion. The dotted lines show the top and base levels that would be picked by a histogram routine. But you may not want to use these levels since the noticeable ringing, and particularly the top segment tilt or slant, tend to scatter the amplitude data. In other words, there are no generally flat segments for the histogram routine to build a solid count on--prevalences may be indicated, but they won't be strong prevalences. So another means of definition might be preferred. In fact, Fig. 9b shows a definition used for one specific data type.

Assuming, for example, that the definitions in Fig. 9b are preferred, let's formulate a simple plan of analysis. This includes using straight lines of the general form y=mx+b. For the 100% level, two lines are required. One is fit to a selected segment of the leading transition and one to a selected segment of the top, and their simultaneous solution provides the intersection defining the 100% point.

The first step in fitting the lines is to derive the "m" (slope) and "b" (y intercept) constants from the pulse data. Getting the slopes for the two lines requires looking at smoothly changing segments of the leading transition and top. Some points for doing this are indicated in Fig. 10. It should be noted that these were picked purely on the basis of what the pulse looks like and the assumption that it is fairly representative of other pulses in the data stream. For other data types, the points may have to be shifted slightly. Or, if the pulse data is widely varying so that you cannot generally designate segments for analysis, you can select segments individually and input their coordinates via the graphic terminal cross-hair cursor and the GIN (Graphics INput) operation of TEK SPS BASIC software.



a. A few pulses from a simulated data stream. Dotted lines indicate the top and base levels picked by a histogram.



b. One pulse singled out of the train. The 100% level is defined as the intersection of best-fit approximations of the leading edge and top segments. All other parameters are defined relative to this.

Fig. 9. *For distorted pulses, the first analysis step is to decide upon definitions of parameters. Two approaches are shown above.*

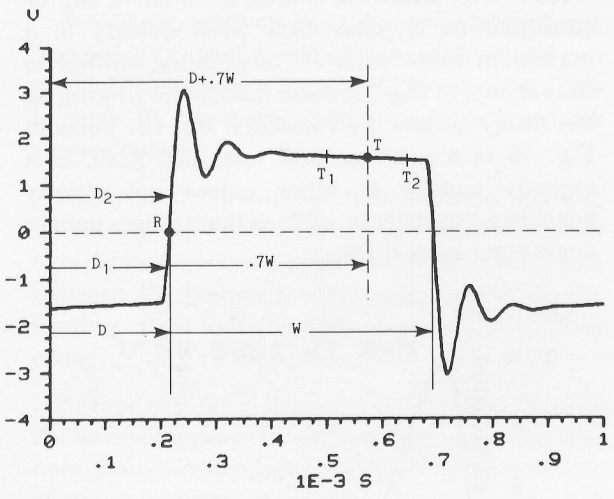


Fig. 10. *Setting up some initial points to begin computations with.*

Returning to Fig. 10, note that the pulse is essentially symmetric about zero amplitude. This saves having to compute a median value--(MAX(A)-ABS(MIN(A)))/2--and allows a cross search for the array locations of R and T, the zone centers, to begin immediately along the zero level. The program lines for doing this, with the pulse data in array A, are:

```
100 REM FIND 100% INTERCEPT
105 LET D=CRS(A,0)
110 LET W=CRS(A(D+1),0)-D
115 LET TL=D+.7*W
```

With the central points (D and TL) located, the next step is to define zones about them. These should encompass enough data to provide a reasonable average of slope, but not be so large as to include aberrations that would unduly bias the average.

Again, the criteria for designating zones depends upon your data. If the pulse is fairly representative of the data stream, as is the case in Fig. 10, you can make some conservative choices by just looking at the data. For example, in Fig. 10, from 0.6W to 0.8W are fairly good choices for T1 and T2. And from -0.75 V to 0.75 V appear reasonable for D1 and D2. The following program lines find these zone end points for you.

```
120 LET T1=D+.6*W
125 LET T2=D+.8*W
130 LET D1=CRS(A,-.75)
135 LET D2=CRS(A,.75)
```

With the zones defined, the average slope over them can be computed. This is done by differentiating the pulse to get point-by-point slope, then taking means of the derivative over the top and leading-edge zones. The following program lines do this for you.

```
140 DIFF WA, WB
145 LET MR=MEA(B(D1:D2))
150 LET MT=MEA(B(T1:T2))
```

Lines 140 through 150 compute the "m" portions for the two straight-line equations,

$$y = m_T x + b_T \quad \text{and} \quad y = m_R x + b_R \ .$$

Now the "b" portions must be found. With the "m" values known, any point in each zone can be picked for "b" value solutions. This arbitrariness does leave room for some slight error since any single point may deviate from the zone trend. However, simplicity of the program is often reward enough for living with a little error.

So, living with a little error, points R and T (coordinates A(D),D and A(TL),TL) from Fig. 10 are used in the following lines to find the "b" values (where HA is the time scaling factor for the "x" or array indexing).

```
155 LET BT=A(TL)-MT*TL*HA
160 LET BR=A(D)-MR*D*HA
```

These lines complete finding the slope and y-intercept values for the two straight lines.

The final step in finding the 100% level is simultaneous solution of the two straight lines for the y value of their intersection.

This is done as follows:

```
 y  =MTx+BT
-y  =MRx+BR
 0  =(MT-MR)x+BT-BR

 x  =(BR-BT)/(MT-MR)
```

This mutual solution for x is then substituted into either equation to get the solution for y.

```
y=MT(BR-BT)/(MT-MR)+BT
```

Or, in a program line, it is computed by the following:

```
165 LET A1=BT+MT*(BR-BT)/(MT-MR)
```

where A1 takes the value of the 100% level. To give you a visual idea of the quality of the straight-line fitting and the solution, Fig. 11 shows the pulse with the two straight lines generated and fit to the top and leading-edge segments. The value of A1 computed from the intersection is printed below the graph.

Another approach to the same end involves a linear, least-squares fit. It operates by minimizing the squares of the differences between the selected

segment of data and a straight line. Although less arbitrary in some instances and certainly more accurate, it is also more complicated and requires more program lines to implement.

If you do prefer improved accuracy over what's indicated in Fig. 11, the essence of the linear, least-squares fit is embodied by the following equations:

$$m = \frac{n\Sigma x_i y_i - (\Sigma x_i)(\Sigma y_i)}{n\Sigma x_i^2 - (\Sigma x_i)^2}$$

$$b = \bar{y} - m\bar{x}$$

In these equations, n is the number of points being considered, x_i and y_i are the coordinates of each point, and $\bar{y}$ and $\bar{x}$ are means of the y_i and x_i values. Notice that you still must select top and leading-edge data segments to operate on, just as was done in the preceding program.

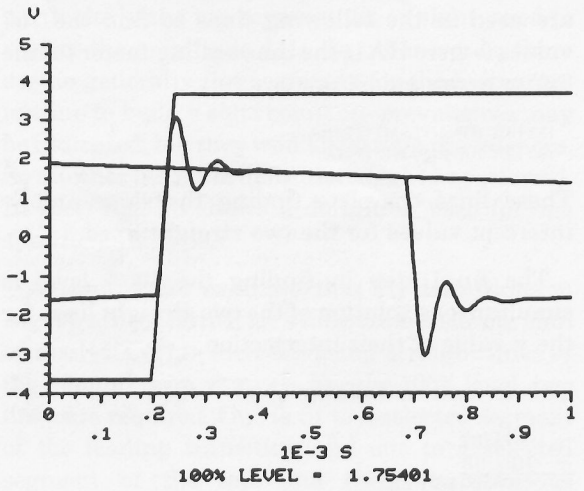


Fig. 11. *Results of straight-line fitting to find the 100% level.*

Transients. Strictly speaking, transients are relatively unpredictable events. They come from a variety of sources--power switching, lightning strikes, and nuclear blasts, to name some of major concern. The latter, it's hoped, will never affect us. But EMP (electromagnetic pulse) testing is still done to analyze transient affects on equipment and to test protection techniques. Even if we discount the possibility of a nuclear blast, power switching, lightning strikes, and static discharges are still a very real concern.

Then there are transients from various fields of research--seismic shock, flourescence decay, and impulse testing for a few examples. These are more predictable, if not in exact size and duration, at least from the standpoint of when the event will occur. Most are intentionally generated under controlled conditions.

No matter what the source, a transient can be classified as a pulse that rises quickly to a maximum then trails off to nothing, similar to that shown in Fig. 12. Some have more ringing on the decay; others die smoothly. But the pulse in Fig. 12 is a good general example. Also, it is vaguely similar to other pulses not strictly considered transients, such as laser pulses used in evaluating optical fibers.

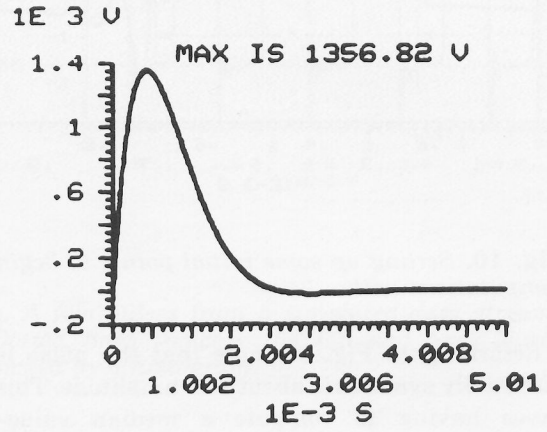


Fig. 12. *A typical transient rising rapidly from zero volts and decaying back to zero at a slower rate. This data was obtained by a TEK SPS BASIC software simulation of a 320 microfarad capacitor discharging 2000 volts into a series RL circuit. The maximum value shown was found by the MAX function.*

Details on analyzing these various transient-like pulses have already been given in several Tektronix publications--"EMP - Testing for Survival," *HANDSHAKE*, Vol. 2 No. 4, Summer 1977; "Digital Signal Processing," (flourescence decay) article reprint from Nov. 1977 *Laser Focus*, and "Keeping Pace with Changing Needs in Optical Fiber Evaluation," Tektronix Concept Note, AX-3903. All are available via the HANDSHAKE Reply Card in this issue.

Though the types of transients and their sources vary widely, the first steps of analysis invariably remain the same. You must find a 0% level and a 100% level before you can go on to anything else. With most transient-shaped pulses, however, this task is at its easiest.

Most transients rise from a steady value--frequently zero volts. So the 0% level is simply zero, or, if there is an offset, it's the value of the offset. And, generally, the 100% level is taken to be the highest level attained by the pulse. The MAX function (or MIN function for negative-going transients) lets you determine this with very little fuss.

From here, analysis can take several directions. The pulse's rate of rise is often of interest and can easily be determined by zone differentiation. This is done in the same manner as discussed for square pulses.

Decay data is another item of wide interest. Often, the time to half decay is required. This is similar to measuring pulse width but takes place between the transient start (usually 10% on the leading edge) and the 50% point on the trailing edge.

Beyond this, analysis becomes too specialized to the various types of transients for more discussion here. However, it is safe to say most of it is a matter of simple programming, attention to details, and a few seconds of software run time.

Bursts. Unlike transients, bursts of sinusoidal energy require some involved programming before analysis can even begin. The problem stems from not really having a pulse but rather a turn-on and turn-off of a repetitive waveform. It's envelope is pulse-like and, indeed, is often the result of applying a square pulse to turn an oscillator or generator on and off. However, the repetitive nature of the pulsed energy creates too many data prevalences to be easily sorted out of a histogram. This is shown in Fig. 13 which resulted from applying the square-pulse program of Fig. 5 to a burst of sinusoidal energy.

Also, a further complication can arise when the burst is caused by nonlinear switching. This often manifests itself as a burst with different top and bottom envelopes. Notice that the burst in Fig. 13 is of a simpler variety having positive and negative symmetry. But this needn't always be the case. So, for the general case, a method of generating burst envelopes of either polarity is needed. This can get involved, so let's take care of finding an easy piece of burst information first.

This easy piece of information is burst frequency. This can be found with the fast Fourier transform (FFT) algorithm. It amounts to transforming the burst to the frequency domain with the FFT, then searching out the array location (frequency) of the maximum spectral component. The following TEK SPS BASIC program lines will do this for you.

```
600 REM BURST ANALYSIS--BURST IN WA
605 DELETE WB,B,WC,C
610 WAVEFORM WB IS B(256),HB,HB$,VB$
615 WAVEFORM WC IS C(256),HC,HC$,VC$
620 RFFT WA,WB,WC
625 LET BF=CRS(B,MAX(B))*HB
630 DELETE WB,B,WC,C
```

In this segment of program, lines 605 through 615 clear and redefine the waveforms (should they have been used previously) to the format required for the output of the RFFT command. In the next line, 620, the fast Fourier transform of waveform WA (the burst) is computed and the results placed

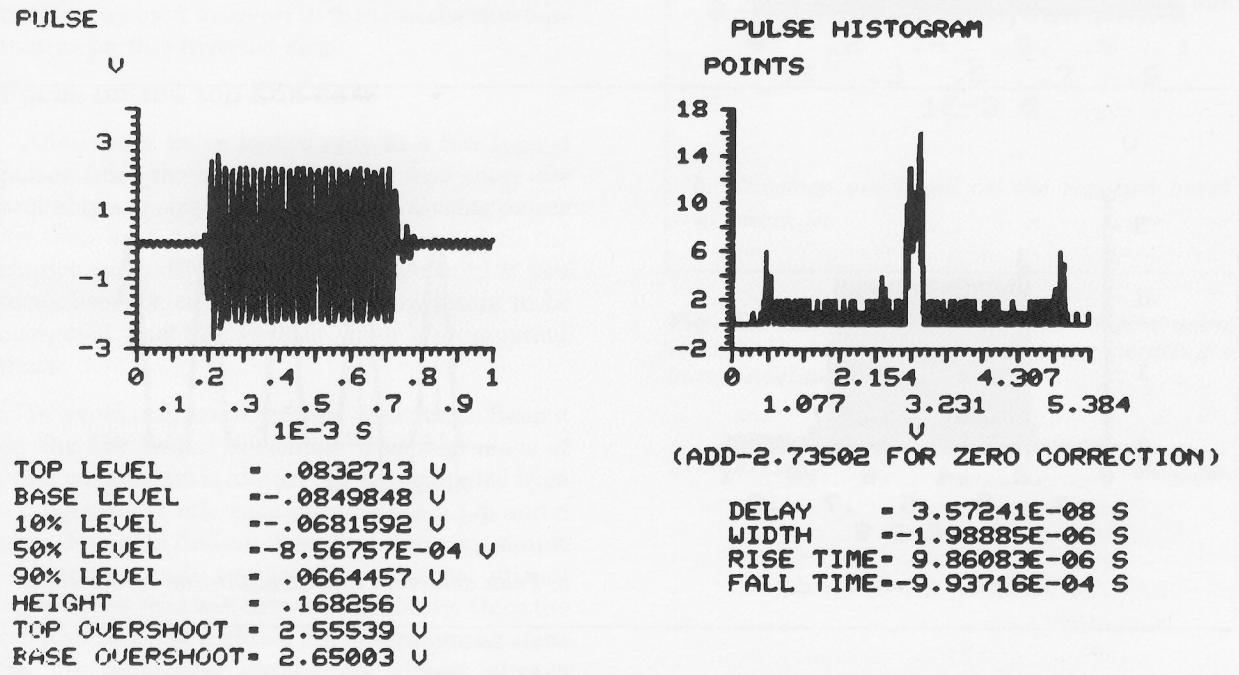


Fig. 13. *Although they are similar to square pulses, bursts can confound standard pulse analysis methods. A histogram reveals numerous data prevalences for the burst instead of the two normally associated with 0% and 100% levels. These competing prevalences can create uncertainty in choosing valid top and base levels.*

in waveforms WB (real part of the frequency domain) and WC (the imaginary part). Then, in line 625, the burst frequency, BF, is computed from the frequency domain data. And, as a "housecleaning" step, line 630 reclears WB and WC in case they should be needed later in their standard 512-element format. More details on these operations, including the RFFT command, can be found in the TEK SPS BASIC software manuals.

If burst frequency is all that is needed, the analysis is complete. But, if more information is needed, information from the burst envelope, some additional programming is necessary...maybe.

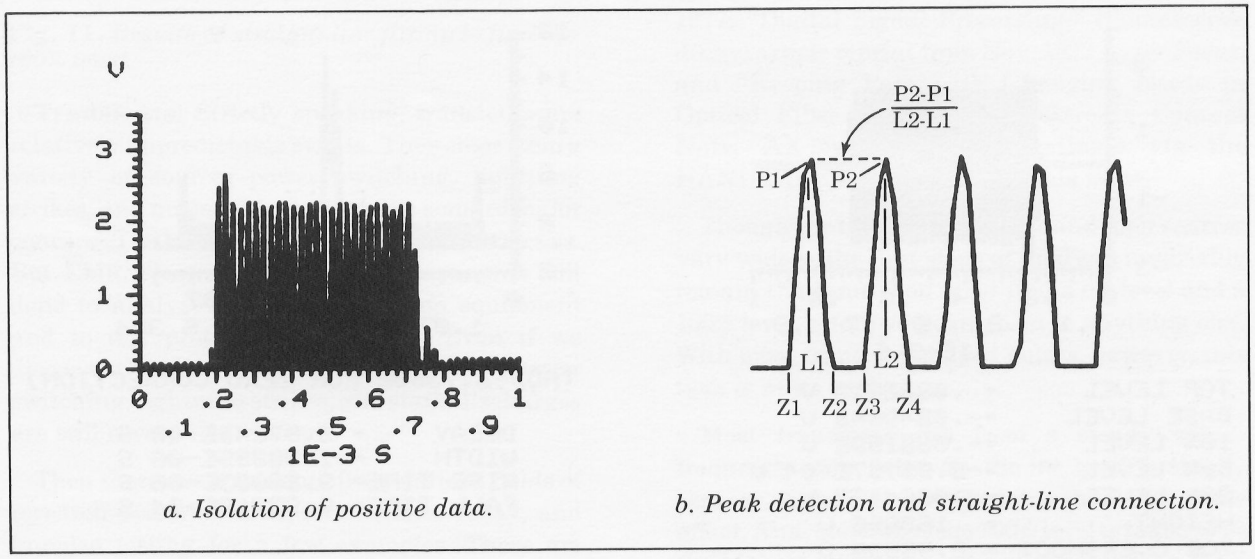
Quite often bursts are used as part of an information or control system. If this is the case, you can save some programming by acquiring the envelope after normal detection by the system. However, if you don't have access to an adequate detection circuit or if the burst of interest is of a nature that doesn't allow easy or accurate electronic envelope separation, you can use software to accomplish the feat.

There are several software approaches to envelope detection. Some are quite elegant, but they can also require substantial knowledge of digital signal processing techniques for successful implementation. In most cases, though, a simple, straightforward approach is adequate. So let's take such an approach.

First, let's attack just the positive-going half of the burst and find the upper envelope. To simplify doing this, all negative-going data is set to zero. With the burst in WA, this is done by the following program lines; the results are indicated in Fig. 14a.

```
635 WAVEFORM WB IS B(511),HB,HB$,VB$
640 LET WB=WA
645 FOR I=0 TO 511
650 IF B(I) <0 THEN LET B(I)=0
655 NEXT I
```

After isolating positive data, the next step is finding the individual positive peaks. Then these peaks are straight-line connected to form the envelope. A simple technique for doing this uses zoning and FOR loops to move the zones through the data. The zones are indicated in Fig. 14b, with the first zone beginning at Z1 and ending at Z2. The second zone is encompassed by Z3 and Z4. Note that both these zones begin with a half cycle and terminate at the end of the half cycle, and they are tied to adjacent half cycles. With these zones defined, they are then individually searched to find their maximums (the peaks, P1 and P2) and the locations of these maximums (L1 and L2). From this information, the incremental slope of a straight line connecting P1 to P2 is computed as (P2-P1)/(L2-L1). The straight-line connection is generated by adding this slope incrementally to the array elements between P1 and P2. When this is done for each succeeding half cycle in the clipped burst, the result is an envelope outlining



a. Isolation of positive data.

b. Peak detection and straight-line connection.

Fig. 14. *The first step in positive-envelope detection is removal of negative data. Then the positive peaks are straight-line connected to outline the envelope.*

the burst. The following program lines can be used to produce just such a result for you, automatically.

```
660 LET Z1=0
665 IF B(Z1+1)> 0 THEN GOTO 675
670 LET Z1=Z1+1\GOTO 665
675 LET Z2=CRS(B(Z1+.5),0)
680 IF Z2 <0 THEN GOTO 8000
685 LET P1=MAX(B(Z1:Z2))
690 LET L1=CRS(B(Z1:Z2),P1)
695 LET Z3=Z2
700 IF Z3+1> 511 THEN GOTO 770
705 IF B(Z3+1)>0 THEN 715
710 LET Z3=Z3+1\GOTO 700
715 LET Z4=CRS(B(Z3+.5),0)
720 IF Z4 <0 THEN LET Z4=511
725 LET P2=MAX(B(Z3:Z4))
730 LET L2=CRS(B(Z3:Z4),P2)
735 LET SL=(P2-P1)/(L2-L1)
740 FOR I=L1+1 TO L2-1
745 LET B(I)=B(I-1)+SL
750 NEXT I
755 LET Z1=Z3\LET Z2=Z4
760 LET P1=P2\LET L1=L2
765 GOTO 695
770 REM BEGIN PULSE ANALYSIS ON WB
    .
    .
    .
8000 PRINT INVALID DATA\STOP
```

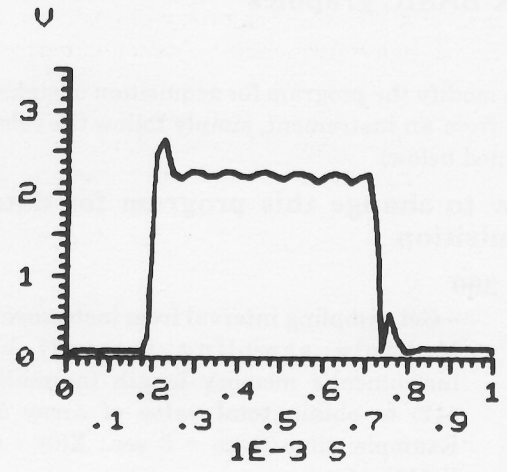
When this routine completes, WB contains the upper envelope of the burst stored in WA. An example of the result in WB is shown in Fig. 15.

From line 770 in the above routine, you can go on with standard pulse analysis techniques to determine the burst envelope rise time, width, and other parameters of interest. In some cases, you'll want to repeat the analysis for the bottom or negative envelope, also. To do this, multiply the burst array by -1 to invert it; then run the envelope routine on this inverted data.

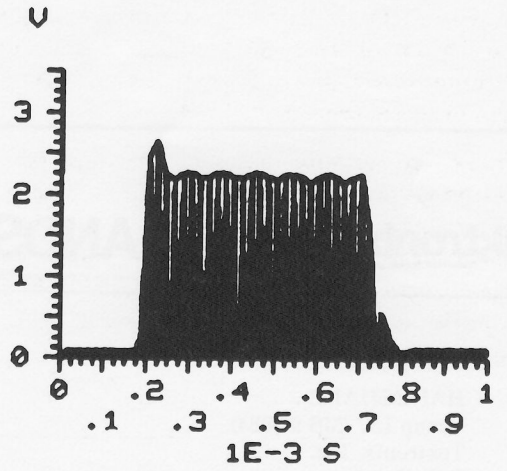
Focus on the top and base

Admittedly we've looked only at a few typical pulses from the existing variety. And there are probably as many approaches to analyzing pulses are there are types of pulses to be analyzed. So the chance for bafflement is high--especially if you focus broadly on the mass of parameters to be computed and the various ways of computing them.

To avoid confusion, narrow your focus. Keep it on the key items. Remember that that mass of pulse parameters is almost always computed from a single set of reference parameters--a top and a base. Focus on finding them first: drawing simple diagrams and trying the simple, direct approaches first will help tremendously. Once the top and base are defined, the programming steps for the remaining parameters almost always become surprisingly clear.



a. The computed envelope.



b. Envelope overlayed on the rectified burst to check fit.

Fig. 15. *Results of the software routine using straight-line peak connection for generating a burst envelope.*

By Bob Ramirez
HANDSHAKE Staff

continued from page 13

Getting the most out of TEK BASIC graphics

To modify the program for acquisition of signal data from an instrument, simply follow the rules outlined below:

How to change this program for data acquisition

line 290

-- Get sampling interval from instrument. Multiply sampling interval by instrument's memory length (normally 512) to obtain total value of array X. Example: time/trace = 5 sec.; X(0) = 0, X(511) = 5.

line 1320

-- For variable DI, get elapsed time between acquisition of first and last trace; also get units.

line 1399

-- Instead of TRACE, print the time (or other) units.

line 1409

-- DX = X(511)/5

line 1480

-- Instead of POINTS PER TRACE, print the frequency (or other) units.

line 1591

-- Instead of AMPLITUDE, print the magnitude (or other) units.

The actual data acquisition routine is up to you. For help on writing a routine, consult the instruction manual for your acquisition instrument.

By Lynne Axel, SPS Software Engineer, and Cliff Morgan, HANDSHAKE Staff

---

Tektronix
COMMITTED TO EXCELLENCE

HANDSHAKE
Newsletter of the Signal Processing Systems Users Group

BULK RATE
U.S. POSTAGE PAID
TEKTRONIX, INC.

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

39-111

CLARK P FOLEY

HANDSHAKE

AX-4273