A11106 402349

NIST
PUBLICATIONS

NISTIR 6418

User's Manual for Lidar Target Simulator

United States Department of Commerce
Technology Administration
National Institute of Standards and Technology

QC
100
.U56
NO.6418
2000

ABOUT THE LAW ENFORCEMENT AND CORRECTIONS STANDARDS AND TESTING PROGRAM

The Law Enforcement and Corrections Standards and Testing Program is sponsored by the Office of Science and Technology of the National Institute of Justice (NIJ), U.S. Department of Justice. The program responds to the mandate of the Justice System Improvement Act of 1979, which created NIJ and directed it to encourage research and development to improve the criminal justice system and to disseminate the results to Federal, State, and local agencies.

The Law Enforcement and Corrections Standards and Testing Program is an applied research effort that determines the technological needs of justice system agencies, sets minimum performance standards for specific devices, tests commercially available equipment against those standards, and disseminates the standards and the test results to criminal justice agencies nationally and internationally.

The program operates through:

The *Law Enforcement and Corrections Technology Advisory Council* (LECTAC) consisting of nationally recognized criminal justice practitioners from Federal, State, and local agencies, which assesses technological needs and sets priorities for research programs and items to be evaluated and tested.

The *Office of Law Enforcement Standards* (OLES) at the National Institute of Standards and Technology, which develops voluntary national performance standards for compliance testing to ensure that individual items of equipment are suitable for use by criminal justice agencies. The standards are based upon laboratory testing and evaluation of representative samples of each item of equipment to determine the key attributes, develop test methods, and establish minimum performance requirements for each essential attribute. In addition to the highly technical standards, OLES also produces technical reports and user guidelines that explain in nontechnical terms the capabilities of available equipment.

The *National Law Enforcement and Corrections Technology Center (NLECTC),* operated by a grantee, which supervises a national compliance testing program conducted by independent laboratories. The standards developed by OLES serve as performance benchmarks against which commercial equipment is measured. The facilities, personnel, and testing capabilities of the independent laboratories are evaluated by OLES prior to testing each item of equipment, and OLES helps the NLECTC staff review and analyze data. Test results are published in Equipment Performance Reports designed to help justice system procurement officials make informed purchasing decisions.

Publications are available at no charge through the National Law Enforcement and Corrections Technology Center. Some documents are also available online through the Internet/World Wide Web. To request a document or additional information, call 800-248-2742 or 301-519-5060, or write:

National Law Enforcement and Corrections Technology Center
P.O. Box 1160
Rockville, MD 20849-1160
E-Mail: *asknlectc@nlectc.org*
World Wide Web address: *http://www.nlectc.org*

The National Institute of Justice is a component of the Office of Justice Programs, which also includes the Bureau of Justice Assistance, Bureau of Justice Statistics, Office of Juvenile Justice and Delinquency Prevention, and the Office for Victims of Crime.

NISTIR 6418

User's Manual for Lidar Target Simulator

James A. Worthey
Electronics and Electrical Engineering Laboratory
Office of Law Enforcement Standards
National Institute of Standards and Technology

March 2000

U.S. DEPARTMENT OF COMMERCE
William M. Daley, Secretary

TECHNOLOGY ADMINISTRATION
Dr. Cheryl Shavers, Under Secretary of Commerce for Technology

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
Raymond G. Kammer, Director

ACKNOWLEDGMENTS

This report was prepared by the Office of Law Enforcement Standards (OLES) of the National Institute of Standards and Technology under the direction of A. George Lieberman, Program Manager, Detection, Inspection and Enforcement Technologies, and Kathleen M. Higgins, Director of OLES. The preparation of this report was sponsored by the National Highway Traffic Safety Administration (NHTSA). The technical effort to develop this report was conducted under Interagency Agreement DTNH22-98-X-05046.

Abstract

This is the operator's manual for a target simulator for speed-measuring lidars, and for the computer program VS, which is a key part of the simulator. This is the simulator referred to in DOT HS 808 539, *Model minimum performance specifications for lidar speed measurement devices*, published by the National Highway Traffic Safety Administration, February 1997. The program is flexible and interacts with the user through menus and forms. The components of the simulator include a commercial delay generator, a custom-made receive-send unit, a PC with two commercial interface cards in it, plus cables, switches, and the program VS. During the simulation, the program has the time-critical task of sending the next delay to the delay generator. Before and after each simulation, the program aids the user in recording the data.

The simulator works with lidar units that have a fixed pulse repetition rate, or PRR. The PRR is measured in a preliminary experiment. The user selects the speed and initial range of the simulated target vehicle. These settings, together with the PRR and the speed of light in air are used to compute a sequence of delay times, modeling the echoes from a vehicle moving at constant speed. The lidar under test should respond to the sequence of delays as it would to a real moving target, and its reading should agree with the simulated speed. The software guides the user through the description of the unit under test, the PRR measurement, the test parameters, and the running of the simulation. Then it can record the data to computer disk.

The sequence of delays generated by the software, when graphed against elapsed time, normally follows a straight-line graph, representing constant motion. The sequence can optionally be perturbed by a periodic function that the user supplies. When constant motion is simulated, this provides an accuracy reference for the unit under test. The accuracy of the simulator itself can be checked against a suitable high-speed oscilloscope, and a detailed method is presented for doing this.

While the simulator components are expensive to assemble, many parts of the program can be exercised on a PC without the other equipment. In this way, readers may experiment with the program before building a complete simulator.

Keywords: laser; lidar; motor vehicle; police; speed measurement; traffic.

User's Manual for Lidar Target Simulator

Chapter 1: Introduction and Technical Background

Introduction. This is the operator's manual for a target simulator for speed-measuring lidars, and for the software which is a key element of the simulator, called VS. The software is flexible and interacts with the user through menus and forms. This chapter presents some engineering background for lidar speed measurement, and the simulator itself. Later chapters deal with system setup and the mechanics of running the computer program. Of 7 appendices, 5 pertain to the formats of files used with the program, one concerns a technical issue in checking simulator operation, and the last appendix concerns wiring.

Background issue: kinematics. A lidar speed measuring device transmits a series of laser pulses in a narrow beam. If the beam is aimed at a stationary target, part of the pulse energy is reflected, and a small part of the reflected energy is detected by the lidar device. Timing circuitry in the lidar device measures the round-trip time-of-flight of the pulse. Since the round-trip distance is twice the target range, the target range can be calculated:

$$d = c_{Air} \cdot t_{RT}/2 \ , \tag{1-1}$$

where d is target range, c_{Air} is the speed of light in air, and t_{RT} is the round-trip time-of-flight. Consistent units are assumed, so if c_{Air} is in meters per second, and t_{RT} is in seconds, d is in meters.

Equation (1-1) applies whether the target is stationary or moving. If the target is moving, and the laser is pulsed repeatedly, then a series of ranges will be obtained. If the target is a vehicle receding from the instrument, successive ranges will be steadily larger. If the target recedes *at a constant speed,* a graph of d as a function of time is a straight line,

$$d = vt, \tag{1-2}$$

where d is again range but t is elapsed time (rather than time-of-flight for the laser pulse) and v is the speed of the target vehicle. If a graph of d versus t is determined experimentally, which is essentially what a lidar gun does, then the slope of the line is the target speed, again assuming consistent units.

To phrase this more carefully, the slope is the time rate of change *of range*; it is the *radial component* of the target's velocity with respect to the instrument. This is implicit in any reference to speed measurement by lidar, or by microwave doppler radar for that matter.

A practical police lidar unit pulses at a steady pulse repetition rate (PRR) on the order of 100 pulses per second to 400 pulses per second. Whatever the PRR is, it is a stable crystal-controlled rate. Internally, the lidar determines the slope in units such as "nanoseconds of delay per laser pulse interval," then multiplies by a constant to account for the speed of light, the factor of 2, the PRR, and the conversion from m/s to km/h or mph.

Background issue: engineering particulars. The first two lidars introduced for police use in the USA use diode lasers of wavelength 905 nm (1 nm = 1 nanometer = 10^{-9} m). They emit a pulse with a rise time of a few nanoseconds (1 nanosecond = 1 ns = 10^{-9} s) and a duration on the order of 20 ns. The existing lidar devices require a minimum of about ⅓ s to collect data. Based on

the stated range of PRR's, they work with something like 30 to 130 range measurements. The lidar device's internal computer checks these data for consistency, and if they fit a straight line, it determines the slope by least-squares regression, then converts slope to speed in the proper units. These facts impose a constraint on the accuracy with which the lidar must measure t_{RT}, and the accuracy with which a simulator must generate a series of values t_{RT} in order to simulate a target moving at an intended velocity.

A car traveling 100 km/h is going 27.78 m/s. The speed of light is approximately 3×10^8 m/s, which is about 0.3 meter per nanosecond, or about one foot per nanosecond. (Actually c = 0.29971 m/ns = 0.98357 ft/ns.) If the measurement time is about ⅓ s, the car's speed must be determined over a baseline of about 10 m. If speed is to be determined to 1% accuracy, the car's actual movement must be measured to an accuracy of 10 cm, meaning the round-trip distance covered by the light pulse is measured to 20 cm. If the speed of light is 0.3 m/ns, the time-of-flight must be measured to an accuracy of a fraction of a nanosecond.

This discussion is meant to promote general understanding, and not to assert precise legal or engineering accuracy criteria. The process of fitting a straight line reduces the net effect of random errors and truncation errors. For now, the simple conclusion is this:

> **A practical speed-measuring lidar must measure time-of-flight with split-nanosecond accuracy. The lidar target simulator must generate delays with similar split-nanosecond accuracy.**

The finer the nanosecond is split, the better it will be. Resolution of 0.1 ns corresponds to range resolution of about 1.5 cm. This pertains to measuring *changes* in range accurately so that speed can be determined in a reasonable time; it has nothing to do with accuracy or precision in the lidar's range display.

Existing speed measuring lidar units display speed as positive if the target is approaching, negative if the target is receding. The simulator uses the same sign convention. That is, practical lidars insert a minus sign (−) that is **not** present in equation (1-2).

Background issue: speed of light. Internally, the simulator must use a value for the speed of light to relate a simulated range to a delay value, via equation (1-1). The speed of light in vacuum is c = 299 792 458 m/s (exact)[1]. The speed of light in air is $c_{Air} = c/n_{Air}$ where n_{Air} is the index of refraction of air. The index decreases with increasing temperature and also with increasing humidity.

The simulator is based on n = 1.0002896, a value which applies at a pressure of 760 mm Hg, 0% humidity, and temperature of 0° C. This is almost negligibly different from 1.0000000 exactly, but it is easy to use the more accurate value. At temperatures above 0°C and elevations above sea level, the change in index operates in the motorist's favor. The effect of humidity is truly negligible, amounting at most to 2 parts in 10^7.

---

[1]E. Richard Cohen and Barry N. Taylor, "The 1986 CODATA Recommended Values of the Fundamental Physical Constants," *Journal of Research of the National Bureau of Standards* **92**(2):85-95, March-April 1987.

Based on this fixed value for n,

$$c_{Air} = 299\ 792\ 458/1.0002896 = 299\ 705\ 663 \text{ m/s} . \quad (1\text{-}3)$$

Timing methods. It was stated above that the lidar device must resolve time to a precision of about 10^{-10} s. It is routine to have digital integrated circuits operating at 10^7 Hz and an integrated-circuit counter that can count at this rate. A counter gated by a signal of interest can time events with a resolution of about 10^{-7} s — split-microsecond accuracy. To get split-nanosecond resolution, more complicated timing measures are required.

One method for more precise timing involves a counter plus an analog "ramp" circuit which can subdivide the oscillator period. This method is used in some lidar models.

The simulator incorporates both complex and simple timing devices. The digital delay generator can be set in increments of 50 ps (1 ps = 1 picosecond = 10^{-12} s); it uses a 100 MHz reference oscillator, and a counter plus a system of delay lines. The counter-timer card, a smaller and simpler system mounted inside the personal computer, provides a 5 MHz reference oscillator plus 5 programmable counter-timers contained in a single integrated circuit. The counter-timer card measures time with a resolution of 200 ns, while its big brother, the digital delay generator, resolves time to 0.050 ns.

In actual use, a lidar makes a series of range measurements that follow a ramp, subject to some random error. The simulator, with its digital delay generator, will simulate the ramp as a stairstep, again subject to random error. For the simulation to work, the unit under test must accept the simulated ramp as an approximation to the ramp that it would see in clocking a vehicle. The 0.050 ns step size meets the basic requirement of splitting the nanosecond rather finely, and the lidars tested have accepted the simulated ramp. There is also the simulation accuracy question: when the effects of step size and random errors combine in the laboratory, how well does the slope of the best-fit ramp correspond to the speed that one intended to simulate? This is addressed in a later chapter.

Chapter 2: System Setup

Block diagram. On a subsequent page is a block diagram of the target simulator with a typical lidar unit being tested. The same figure was used in DOT HS 808 214, *Model minimum performance specifications for lidar speed measurement devices.* This block diagram shows the general relationships among the components, but not the wiring details. Connections are altered during setup and use of the simulator. A switch panel must be used, since the BNC connectors cannot withstand numerous connect-disconnect cycles. A version of the switch panel is described in Appendix W.

Components

Personal computer. The general function of the subsystems may be reviewed in the diagram. The personal computer is labeled "Fast PC," which was a more apt description in 1993. Now it could be labeled "Slow PC." The ideal computer is an AT compatible with a 486 chip running at 66 MHz. Hardware floating-point support is *required.* Useful data have been taken using a 20 MHz 386 PC. A Pentium® PC or faster would not increase overall capability very much, and might cause timing

problems involving communication with the counter-timer card, for instance. Such problems could potentially be fixed in software. [2]

[2]In the block diagram and elsewhere, commercial products are identified by brand name or model number. This is done to help identify the components and not to indicate that they are endorsed by the U. S. Government, or that they are best suited for the task.

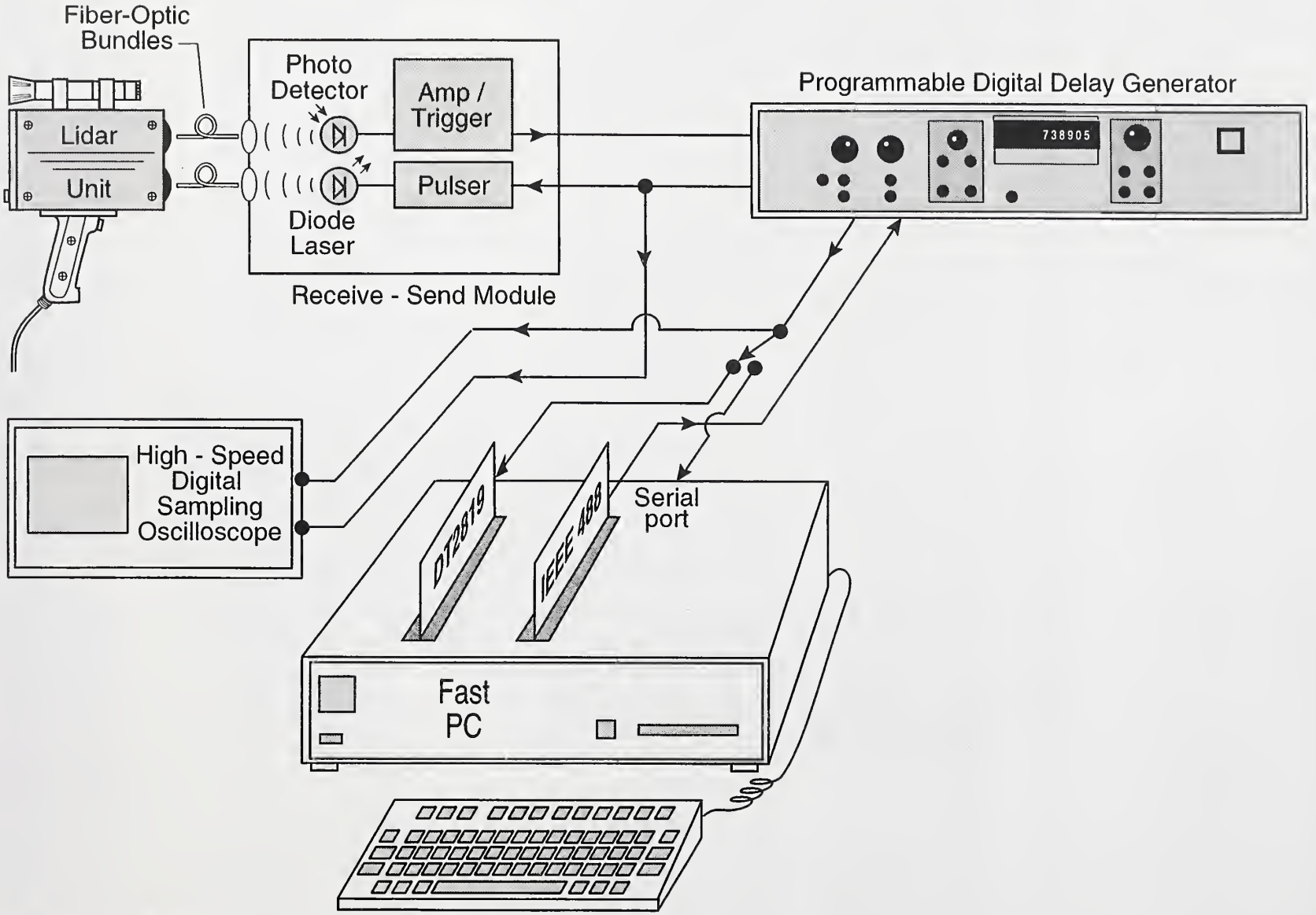

Figure 1, Block diagram.

IEEE-488 interface card. IEEE-488 is a standard that permits computers and instruments to transfer information at rates on the order of 1 MB/s (= 1 megabyte/second). IEEE-488 is also known as GPIB, for General Purpose Interface Bus. To the casual observer, the IEEE-488 bus is a stiff cable that connects the devices. Up to 15 devices may be connected, but in the simulator, only the computer and the delay generator are connected. (The oscilloscope has a GPIB interface, but it has not been used.) The computer has a card that provides an interface from the computer's internal bus to an IEEE-488 connector on the back of the computer. The card that has been used is a National Instruments AT-GPIB or AT-GPIB/TNT model. (In 1997, even the AT-GPIB/TNT model is called the "legacy" card in the catalog. The newest thing is plug-and-play.) This card comes with software: an interactive configuration program, IBCONF.EXE, must be run at least once; and a driver that will usually be loaded at boot time, so that it is present in memory before the simulator program is run. The delay generator must be ordered with a GPIB interface; it will then have the appropriate connector on the back.

Serial port. Most personal computers will have two or more serial ports. Either COM1 or COM2 must be used and the software configured accordingly. The serial port is not used for data transfer in the normal way. Rather, it is used to synchronize the program running in the PC with the pulsing of the unit under test (UUT). The DSR line of the serial port must be pulsed after the UUT has fired a pulse. Normally, the trigger pulse is taken from the "initial pulse" output of the delay generator, but it could be derived differently, so long as it occurs after the receive-send module has received a laser pulse from the UUT.

Pulsing the DSR line of the serial port generates an interrupt, triggering an interrupt service routine which then transmits a fresh delay value over the GPIB. Technically, a "race condition" is created. That is, the computer is triggered to feed the delay generator a new delay value via the GPIB, before the previous delay is completed. Tests have shown that this is not a problem; if it were, the trigger pulse could be taken from the delayed pulse.

Counter-timer card. The counter-timer card that has been used is the DT2819 from Data Translation, Inc. It is indicated by this model number in the block diagram. The primary function of the counter-timer card is to measure the PRR of the unit under test. This is done in a preliminary step before the actual simulation is run. The counter-timer card has a 50-pin connector on the back. Optionally available from the same vendor is the DT758-C connection panel, which brings the 50 conductors out to screw terminals. The switch panel can be rigidly attached to the screw terminal board. In one version of the switch panel, described in Appendix W, it is made from clear plastic, so that a mysterious "black box" is not created, and the user can trace the wires to the numbered screw terminals, switches, BNC connectors, etc.

Although the counter-timer card is extensively programmable in software, the simulator application calls for one permanent external connection. On the DT758-C connection panel, screw terminals 3 and 20 must be connected by a length of wire. Screw 13 is the input for pulses whose PRR is to be determined. When needed, screw 11 is a source for pulses of programmable PRR. There are multiple signal grounds, including screw 21.

The counter-timer card comes with some software, but in this case the software was not used. All interactions with the counter-timer card are done via C-language routines, written by the present author. The source code is available, as is all the non-proprietary code. This is in distinction to the

IEEE-488 subsystem, where the handshaking problems are potentially more complicated, but are handled through a vendor-supplied driver.

Simulator software VS. Before any lidar can be tested, certain setup steps must be taken, such as putting software settings in line with hardware settings. The simulator program permits this to be done interactively. In the testing of a lidar device, the first step is to measure the pulse repetition rate (PRR) of the unit under test (UUT). This is done by the counter-timer card, under program control.

Once the PRR has been determined and the user has input the speed and initial range to be simulated, the program prepares the simulation itself. The initial range, converted to a delay, is loaded into the delay generator, and the next delay is calculated, ready for use.

When the lidar emits the first laser pulse, the receive path of the receive-send module sends an electrical pulse to the delay generator. The delay generator measures off the delay time, then pulses the input of the receive-send module, triggering a laser pulse to the UUT. In this data path, only the receive-send unit, the delay generator, and the coaxial cables, take part. The delay generator must be programmed with the proper delay value and fully ready when the UUT emits a pulse. When the delay generator has been triggered, it pulses the computer's serial port. This begins the chain of events in which a new numerical delay is transmitted by the computer over the IEEE-488 bus, then a further delay time is calculated.

The basic simulator function is to create a series of delays that increase or decrease along a single straight line. The software has an additional feature that permits a sawtooth perturbation to be added to the straight line. This simulates the series of delays that might occur if an operator did not track the target perfectly, but swept the laser beam along a smooth body panel, such as the hood of a car. Although a standard sawtooth is defined in the model minimum performance specification, the software permits more complicated functions to be used. The perturbation must be periodic.

Event Timing Issues Within the Simulator. The system as described has been tested at pulse repetition rates up to 400 Hz. This implies that the creation of the delay, and the programming of the new delay can occur in as little as 2.5 ms (0.0025 s). There is no easy way to assign times to all the component events which must occur, and some events can overlap. However, it is believed that much of the 2.5 ms is consumed by the data transfer over the IEEE-488 bus, and the internal "setup" of the delay generator.

Provision for Testing the Simulator Itself. The DT2819 counter-timer card incorporates 5 programmable counter-timer units within a single AM9513A chip. The function of timing the pulses of the UUT occupies 3 counters. In routine use of the simulator, this is the only function of the counter-timer card.

To test the simulator for calibrated and consistent operation requires a source of pulses with a constant (crystal-controlled) repetition rate. A feature of the program VS permits the remaining two counters to generate a frequency in the range of 76.2951 Hz to 2.5 MHz. This pulse train, generated by dividing down the 5 MHz reference clock on the DT2819 card, appears on screw terminal 11. Once started, this pulse generator functions entirely on the counter-timer card.

Putting aside the "perturbation" feature, the simulator's task is to generate a series of delays that track a "ramp," a straight line whose slope is determined by the speed to be simulated. This functionality can be tested with any source of pulses (at a stable repetition rate) routed to the computer's serial port input. The source need not be a lidar, and indeed a lidar is restrictive because its PRR is fixed. Any laboratory pulse generator could be used, but the pulses provided on screw terminal 11 are convenient because they are crystal controlled.

Perturbation feature. The perturbation feature exists entirely in software. If used, the perturbation must be defined by an ASCII file that the user has prepared in advance. The operation of this feature is such that:

1. The normal unperturbed simulation produces a series of delays that, if graphed versus time, closely track a straight line. Such a straight-line variation is often referred to as a "ramp." Applying formula (1-1) to convert delay to range, this is a ramp of range versus time, and its slope is the simulated speed. The perturbation adds a non-zero and non-random function so that simulated range, when graphed against time, no longer follows a straight line.
2. The perturbation is always periodic. When the specified perturbation is used up, it immediately starts over. This periodic function adds to the ramp.
3. The perturbation is specified as a disturbance in range, rather than velocity. A smooth change in range implies a velocity, but the user specifies times and displacements in units of meters or feet.
4. The user specifies the perturbation at arbitrary times, but it then is converted to a series of displacements at multiples of the pulse interval. The repetition period of the perturbation may be adjusted slightly by the program. The exact pulse-by-pulse perturbation is precalculated and may be reviewed interactively by the user. The detailed perturbation is derived from the given data by straight-line interpolation.
5. The perturbation must be specified by no more than 32 time-distance pairs, and must be completed within 200 pulses, whatever the PRR is. These parameters could be changed by a re-compilation of the program, but those are the values as of 1997 December.
6. The standard calls for a specific sawtooth perturbation, with optional shifts in the origin of time. More general perturbations can be generated by changing only the ASCII data file.

Practical speed-measuring lidars are designed to "trap" bad data, and only to display a reading when the data are consistent with the lidar tracking one spot on one vehicle. If the lidar is aimed at one car, and a truck cuts the laser beam during the measurement period, this will put an abrupt discontinuity in the series of range data, which should cause the data to be rejected.

It is of practical concern that under certain conditions speed might be falsified by an error which is harder to trap. If, during a measurement, the laser beam is allowed to sweep along a surface of the target vehicle, this can give rise to so-called sweep error. Suppose a lidar is aimed at an approaching car, and during the measurement the beam sweeps down the engine bonnet, from the windshield towards the radiator. If not discarded, data taken in this way could imply a speed greater than the true speed of the vehicle. The perturbation feature can simulate the sort of imperfect ramp that the lidar may see when it is not held steady.

The original "official" perturbation. The test perturbation specified in the "Model minimum performance specification" is defined by four points. It simulates sweeping towards the lidar, 1.52 m (5 ft) along a smooth sheet-metal area in 0.178 s:

Time, s	Distance, ft
0.0	0.0
0.010	0.0
0.012	5.0
0.200	0.0

Why the simulated perturbation is periodic. The reader may question why the simulator adds a periodic perturbation. In real life, the perturbation might occur only once. With a periodic perturbation, the UUT can do one of three things:

1. Give an error message or no reading (considered equivalent in this context). This is an acceptable response.
2. Give an erroneous reading. This is an unacceptable response.
3. Give a correct reading based on a very sophisticated decoding of events after collecting data for a long time. It is not expected that any practical lidar will do this, but it is a theoretical possibility. This is a hypothetical acceptable response.

With a one-time perturbation, the UUT could:

1. Give an error message or no reading. Acceptable.
2. Give an erroneous reading. Unacceptable.
3. Give a correct reading by detecting the perturbation and waiting for it to end; give a correct reading by some chance process of errors canceling out; or give a correct reading by always discarding the first 0.200 s (or more) of data. This is acceptable or not, depending on what really happened.

In short, the periodic perturbation is intended to *force* a reading of an error signal or a blank display.

Chapter 3: VS, the Top Menu.

Introduction. The program, VS, is menu-driven and interactive. This chapter will review the top-level menu and the major activities accessed from the top menu. For further understanding, the user may experiment with the program, or read the earlier chapters of this manual. The menus are represented schematically and do not appear on paper exactly as on the screen. All interaction is through the keyboard.

When the apparatus has been assembled and preliminary steps have been done, such as loading the TSR driver for the PC's IEEE-488 card, then this chapter is the main "how-to manual" for the steps in collecting data. Chapter 4 tells more about the lower-level menus.

To conduct an actual experiment, the program must be run under straight DOS, not within OS/2® or Windows®. For the purpose of learning and experimentation, the program may be run with the specialized hardware absent and may be run in a DOS window under OS/2. If it is run under such circumstances, error messages should be ignored, and not all parts of the user interface will be accessible.

The program was developed using Microsoft C 6, a Microsoft Quick Assembler, a text-windowing package called Window Boss, and a subroutine library supplied with the IEEE-488 card. From the user's point of view, the key thing is the text windowing package, which allows for overlapping windows and a moderately consistent user interface from one window to another. On menus, the Tab key or space bar generally moves forward and Shift-Tab moves back; generally the four arrow keys have some effect, and the Enter key selects a menu item. In most cases, hitting the initial letter of a menu item will jump the highlight to that item.

From most places in the program, it is possible to "cancel" or back out to a previous level. There may be a menu item to permit this and the Esc key will usually have this effect.

Data entry screens are more complicated than menus. In filling in a data item, the user is in effect interacting with a tiny editor which permits a data value to be changed, from a default value which may be other than a blank or a zero. This tiny editor is a feature of Window Boss, as modified slightly by the present author. It may not work exactly like other familiar editors. The user should experiment with data entry and double-check entries before accepting a data screen. In many cases, the user must supply a special key-stroke, other than the Enter key, in order to accept a set of data. This was done on purpose to promote a little extra checking.

Error messages. When a data entry screen is active, the user can see a help message concerning the current data item by pressing F1. If the user enters an invalid string, an error message is displayed, and the highlight returns to the faulty datum. The error message or help message appears in a window at the bottom of the screen, and in the data entry situation, the error and help messages are often the same.

Some instructional messages appear each time the program is run. For instance, when the user selects from the top menu to "identify the unit under test," this message appears in a window:

Time Check
Data will be labeled with a time and date referring to a time right before the data are taken, based on the computer's internal clock. If the following date and time are incorrect, please use NCC or some other utility to set the computer's clock. 1997 Sep 22, 15:36:52
Press Any Key to Continue

The time will be continuously updating according to the PC's clock. If the clock is not correct, the user will need to exit the program, in the usual way, in order to set it. The reference to "NCC or some other utility" is archaic. In earlier versions of DOS, typing TIME at the DOS prompt did not lead to an actual setting of the hardware clock. In DOS 5.x or 6.x, I believe that typing TIME at the DOS prompt will lead to setting the real clock. No special utility is needed.

In any case, this is one example of specialized messages that appear as the program is run. The user should take some time to read the messages when they appear. In some cases, a message appears once, or a limited number of times, and then is gone "forever" (until the program is run again).

A number of error messages or suggestions will appear if the user attempts to do things out of sequence.

A firm decision was made on one issue of sequencing. That is, each time the program is run, the pulse repetition rate of the UUT must be *measured* before a simulation is run. There is no provision for the user to type in the PRR or recall it from a previous run. It was felt that this gives the data more credibility. This becomes a burden if the user is testing the equipment and not collecting "real" data, but for right now, this sequencing is always enforced.

Top menu. The top menu is usually visible. Only one window is active at a time, so the top menu may be visible but not active. The following table is a schematic representation of the top menu.

VS = Velocity Simulator		
Files	Standard Speed Series	Alter Perturbation
Observe Pulse Rep Rate.	DT2819 Experiment	Clock of DT2819.
Identify UUT, etc.	Measure PRR repeatedly	Option 12
Write Header to Results File		Quit
1 Speed, 1 Range.	Put Pulses on Screw # 11	
James Worthey, OLES, 1992		

VS = Velocity Simulator. This is not a menu item, but the title of the program.

Top menu items:

Files. This brings up the Files Menu. The first item on the Files Menu permits the Results File Name to be set, something that will be done routinely. Other items on the Files Menu pertain to basic settings that will seldom be changed.

In the routine testing of a lidar unit, after the program is started, the first 4 or 5 menu items down the first column of the top menu should be selected in sequence. When one item among the first 4 has been chosen and then control is returned to the top level menu, the highlight will move to the next item. This is intended as a hint. The user remains in control.

Observe Pulse Rep Rate. Selecting this item starts the measurement of the PRR of the UUT. Although the measurement begins immediately, it will wait "forever" for the user to trigger the lidar. Also, the measurement repeats 6 times and is not considered complete until all 6 values are in close agreement. The details of what is happening appear on the screen. To observe the pulse repetition rate, the user must switch the pulse receiver output into the counter-timer input, screw 13; normally, this signal is taken from the initial pulse output of the delay generator.

Measurement of the PRR may take from about one second, up to several seconds. Some lidar devices are designed to pulse for less than one half second, then stop. To make such a lidar continue pulsing while the trigger is held down, it is necessary to open the loop in some way so that the simulator is not returning pulses into the lidar's detection circuit. An electrical part of the delayed-pulse path can be opened, but it should not be done by putting wear and tear on BNC connectors. The switch panel of Appendix W provides a switch for this purpose. In some cases, it may be easy to open the optical path.

Identify UUT, etc. The simulator not only performs experiments, it records the data. In order for the data to be recorded in a useful way, the user must identify the unit under test. This item brings up a data-entry screen where the model, serial number, purpose of the experiment, etc., may be entered. The data items may have non-blank default values, but because of a quirk in the software, the default values appear one at a time as the user proceeds through the screen. The PRR is treated as part of the description of the UUT; however, it cannot be entered manually. It must be measured every time.

With no special effort by the user, the values entered on this screen are saved in a file called PREVIOUS.UUT, and become the default values the next time the program is run. One of the options on the Files menu is to substitute a different filename for the saved UUT description, giving another level of control.

Before the data entry screen comes up, an informational window appears, displaying the current time by the computer's clock. The user should take a few seconds to verify this time reading. Data will be automatically time-stamped when they are written to the results file, so it is important that the computer clock be correct. To set the clock, exit from VS by hitting Escape several times, then use a DOS utility or other program to set the hardware clock. (Be careful. In earlier versions of DOS there is a command TIME that sets a temporary clock, but not the real hardware clock.)

Write Header to Results File. One goal is to have the computer generate a complete description of each experiment in a "results file." This item leads into a rather complicated, but self-explanatory menu screen. The "header" can be the description of the UUT and/or the basic settings. Normally, it will be both of these, but if one part has changed and the other has not, it is appropriate to save just the changed part. The description of the UUT cannot be saved until PRR is measured. It will usually suffice to accept the default on this screen; *the current status determines what the default is.* Read the messages.

1 Speed, 1 Range. When the top level menu's previous items have all been chosen in sequence to choose a results file, measure the PRR, describe the UUT, and save the header information, then an actual simulation may be run. If complete preparations have been made, when this item is chosen,

a data entry screen appears on which the speed and range of the simulation may be entered. If preparations are **not** complete, a warning message appears with a menu of options.

The data entry screen has blanks for range and speed, and shows the numerical limits on each item. Range is requested in feet or meters, while speed may be in meters per second, kilometers per hour, or miles per hour. The units are those which were specified for the UUT on the "Identify UUT" screen. Speed may be zero to simulate a stopped target; less than zero to simulate a target going away; or greater than zero to simulate a target approaching.

When range and speed have been set, and the user has elected to "Keep" the values, the simulation window itself appears. The title across the top and bottom of this window says "Run the ... Speed Simulation." An information line across the bottom says "UNTIL flashes start: Esc to exit or R if a Reading is Ready." When this screen appears, the first delay value has been loaded into the delay generator, and a queue of further values has been calculated. The program is checking to see if the first laser pulse (flash) has occurred and also polling the keyboard, looking for the Escape or R key to be hit. When the first flash does occur, the program stops polling the keyboard and only runs the simulation. The user regains control when the UUT stops flashing.

The simulation screen indicates initial range in both meters and feet. It shows speed in all three units: km/hr, miles/hr, and m/s. It indicates if the perturbation is on or off. It shows the pulse repetition rate last measured, and it shows "delta distance" in meters. Remembering that the UUT flashes with a steady PRR, and the simulated target has constant speed, delta distance is the fixed distance that the simulated car moves between flashes of the laser. There is then a statement that the simulated vehicle is approaching, stopped, or receding, and a statement that the vehicle goes out of range after a certain number of flashes, or a certain number of seconds. These calculated approximate values serve as reminders of the kinematic constraints: for instance, a target that is near the maximum range and receding at a high speed will quickly go out of range.

Two last numerical values are displayed with these words and symbols:

Flashes since reset to initial range = 6000 √
Minimum depth reached by queue = 11 √ .

Actual numbers may vary. The square-root symbols are intended to be check-marks; the square-root happens to be available on the old-fashioned character mode display. The check-marks appear when the numbers are final values, after the simulation has been run and then stopped. An important function of the "flashes since..." value is to show when the simulation is running; the number will be visibly changing, though not precisely correct until the simulation has stopped and the check-mark is displayed.

The "minimum depth reached by the queue" requires deeper explanation. In Chapter 2, this sentence appeared: "The initial range, converted to a delay, is loaded into the delay generator, and the next delay is calculated, ready for use." This statement is correct as far as it goes, but it fails to mention the queue. For a more complete discussion, refer to Appendix Q.

While the simulation is running, the program does not respond to the keyboard. When the simulation is not running, the user may hit the Escape key to exit directly or, as it says on the screen, hit "R if

a Reading is Ready." After hitting R, the user sees a small data entry window in which to enter the range reading and the speed reading from the unit under test. If the readings are entered, a description of the entire trial, including the time and the simulation parameters is added to the results file; if they are not entered, nothing is written to the file.

The data entry screen carries a notation Use -999.0 for range or speed to denote "no reading". In general, the program will not permit invalid numerical values to be entered, but this special value may be entered to indicate that the UUT gave an error signal or other blank reading. Recall that a blank reading may be the "right answer" if electromagnetic interference or a perturbed simulation is used.

As in other data entry windows, the user must hit "K" for Keep, then Enter, in order to save the data. The user is particularly advised to develop careful habits in entering these two numerical data. After entering range and then speed, it is best to hit the Enter key several times, moving the data entry cursor past the numbers. This has the effect of converting the displayed character string into a number and back into a string, putting it into a standardized form.

After the user hits "K," a special menu window appears. It displays the data line that will be written to the results file, and gives a choice to save the string, to put a comment after the string, or to cancel. If the user selects to put a comment, a line appears for its entry. The Enter key terminates the string and saves the data.

Standard Speed Series. Choosing this item takes an alternate path to the simulation window just described. Rather than set the speed and range manually, the user chooses from a list that was prepared in advance. The PageUp and PageDown keys scroll the list itself, if there are more than 8 items. The up and down arrows move the highlight up and down. The Enter key selects the highlighted value and runs the simulation. When the user exits from the simulation window, possibly saving a set of readings along the way, then control returns to the Standard Speed Series screen, with the highlight advanced to the next item.

An additional column in this window displays "Times_Tested." This is an aid to the user, not intended to be foolproof. If the user selects a range-speed pair, then exits from the simulation screen without completing a simulation and recording the data, the "Times_Tested" number will nonetheless be incremented.

The list of range-speed pairs resides in an ASCII file which must be prepared in advance using an editor. The default filename is SPEEDS.LST, and an example file called SPEEDS.LST is supplied with the program. A good method for creating a different list would be to make a couple copies of the example file with other filenames, such as SPEEDS.BAK and SPEEDS.NEW, then edit SPEEDS.NEW to create the new one that is needed. Further explanation appears in Appendix S.

DT2819 Experiment. During initial setup and test, the user may wish to verify that the DT2819 card is functioning properly. Selecting this item chains all 5 counters to divide down a 1 MHz clock. The counters are sampled from time to time and the values displayed on the screen. Also, the output of clock #5 appears on screw terminal 11 as positive pulses; an LED connected from terminal 11 to ground (such as screw 12) will blink. In fact, pulses should appear on screws 3, 5, 7, 9, and 11, and an LED could be attached to each screw. (The low-power Schottky outputs can supply only a limited current, so no current-limiting resistor is needed.)

Measure PRR repeatedly. This selection brings up the same display as Observe Pulse Rep. Rate. The pulse repetition rate is measured; the result is displayed briefly; then a new measurement starts. This is intended for use in setting up and adjusting the hardware. For instance, the user might be adjusting the external pulse trigger controls of the digital delay generator, or simply setting an input pulse rate. If a consistent pulse repetition rate is measured, it will display on this screen, along with the raw counts of 6 trials. Any inconsistency will show up as a failed measurement. The process will repeat until terminated by hitting Esc.

Put Pulses on Screw # 11. This selection permits a pulse train of calibrated frequency to be output to screw terminal #11 of the DT758-C connection panel. Screw terminal #12 is one of several ground points that can be used. Because it is crystal-controlled and adjustable, this pulse train is particularly useful in calibrating the simulator against the digital sampling oscilloscope. The pulses are generated using clock module #5 of the DT2819 card. Clocks #1 through #4 are used in measuring the PRR of the UUT, so it is possible to run both functions at once.

On the input window for this function, the user may set the pulses on or off. If pulses are on when the user exits the window, they will remain on during other activities, and will even continue after the user exits from the program entirely. While the program continues, a small window is displayed to indicate that screw #11 is being pulsed, and the PRR of the pulses being generated.

This pulse frequency is generated by dividing down a 5 MHz reference frequency. The divisor varies from 2 to $2^{16}-1$ (= 65,535), giving a frequency from 2.5 MHz to 76.2951 Hz. Both the divisor and the resulting frequency are displayed. The user increases or decreases frequency using the up or down arrow key. To scroll through the frequencies faster, the user may hold down the Shift key; to scroll fastest, hold down the Alt key.

Alter Perturbation. This selection on the top menu brings up a 4-item menu and a window that displays some information about the perturbation currently selected. Chapter 2 above gave some rationale for this feature. The perturbation must be entirely specified in an ASCII file prepared before VS is run.

The perturbation is specified by a series of points of the form (time, displacement). The displacement is linearly interpolated between these points. The points need not be at equal time intervals, but should be chosen for convenience to give the desired piecewise linear function. The format of the file is explained in Appendix P.

When Alter Perturbation is selected, two windows appear. The Perturbation Menu is active, with 4 items. Below it, the Perturbation Info Window gives a summary description of the perturbation in use. The given perturbation points are displayed, or the first 15 if more than that number are given.

From those points, a peak-to-peak description is calculated and displayed. Other identifying information is also displayed. A short sentence states if the perturbation is turned on or off.

On the perturbation menu, the first item toggles the perturbation on or off, the status being displayed in the window below. The second menu item, Enter New File Name, brings up a data entry screen for the new filename. The third menu item, View Perturbation Array, enables the user to review the entire perturbation as it will play out, flash by flash. The array is generated from the given points, combined with the measured pulse repetition rate, with necessary adjustment so that the period of the perturbation is a multiple of the pulse period. The "info" window jumps to the top of the screen, and a scrolling view of the perturbation array appears at the bottom. Arrow, PageUp and PageDown keys control scrolling. The Esc key returns control to the perturbation menu.

Users have indicated a desire to use the perturbation feature with larger sets of defining points, such as empirical instantaneous range data from a field test. It may well occur that enhancements will be made to this section of VS. The reader may then find that the functionality of the perturbation feature slightly exceeds what is claimed in the present user's manual.

Clock of DT2819. Like the earlier item, "DT2819 Experiment," this item exists primarily for the instance in which the user is assembling the apparatus and wants to verify that the counter-timer card is working. The DT2819 is put into its time-of-day mode, the time of day is loaded from the computer's normal clock. The display then shows the time, constantly updated from the counter-timer card.

Option 12. This is a do-nothing option, which displays the message "Option 12." In the event that a programmer wishes to add one more feature, a call to the new feature can be put in the main routine, in place of the message display call.

Chapter 4: Further Menus and Data-Entry Screens.

Chapter 3 discussed primarily the top menu, omitting some details of the lower-level menus and data-entry screens. This chapter explains more of the detailed workings of menus and screens below the top menu.

Files. Selecting this item from the top-level menu displays the 8-item Files Menu:

Files Menu	
Set Results File Name	Interactively Change Basic Parameters
Look at Basic Parameter Settings	Default Basic Parameters (Restore 'em)
Read Basic Settings from a File	UUT, Change Name of the Small UUT File
Write Basic Settings <only> to a small File	Quit to Top Menu
Choose 1 Action	

Along with the Files Menu, this table of information is displayed:

```
                          Current Basic Settings
   file name: parms.def
 stored name: parms.def
 CommSelect : 1  {1=COM1, 2=COM2}    TimeOut :     100000 {#loops to wait}
  QueueSize : 12                 RtsPulseFlag :  0
   BOARD488 :  0                       BNC7095 :  6
               Base Address of DT2819 Card :  230   hex
               <display only---Files Menu remains active>
```

The user is encouraged to run the program and experience the menus and data-entry tables on the computer screen. The screen displays are colorful and interactive. Even on a computer that lacks the counter-timer and IEEE-488 cards, many program elements can be accessed, including the Files Menu and the functions that it calls.

Five of the items on the Files Menu pertain to the "Basic Parameter Settings," which may be explained as follows:

file name	The file from which the basic parameters were read. The default file name is parms.def .
stored name	The name of the file as stored within the file itself.
CommSelect	An integer that must equal 1 or 2. The value 1 chooses COM1 as the port whose DSR line will be pulsed to synchronize the computer with the pulsing of the UUT. Value 2 chooses COM2.
QueueSize	The number of delay values that are pre-calculated and ready for use when the simulation starts. See Appendix Q. The default is 12, and it is unlikely that a different value is needed.
BOARD488	The "reference number" of the IEEE-488 board to be used. This is **not** an address on the computer's bus and it is **not** an address on the IEEE-488 bus either. It is a reference number that would become important if there were more than one IEEE-488 card in the computer. In most cases, this value is 0

	and the address of the card on the IEEE-488 bus is in fact 0 also. These issues may become more clear when you run the program IBCONF, supplied by the vendor of the IEEE-488 card.
TimeOut	This is the number of iterations for a software timing loop. The default is 100000. This parameter might indeed need to be changed if the program is run on a computer that is much faster than the ones originally used. When all the preparations have been made, and the simulation is started by the first pulse from the UUT, the program stops polling the keyboard. If the UUT then stops pulsing, the program needs a measure of time in order to "notice" that the next pulse is overdue. While waiting for the next pulse of the UUT, the program counts down from TimeOut to 0, then returns control to the keyboard.
RtsPulseFlag	For diagnostic purposes, the interrupt service routine within VS has the ability to pulse the RTS line of the comm port that is being used after a delay value has been loaded to the delay generator. If this value is nonzero, the pulse is enabled.
BNC7095	This is the address of the delay generator on the IEEE488 bus. Permitted values are 0 to 30. The default is 6. The same number must be set in switches on the delay generator, and through the program IBCONF, supplied by the vendor of the IEEE-488 card.
Base Address of DT2819 Card	The base address of the timer-counter card, on the PC bus, in hexadecimal. The default is 230.

For most users, the default values of these "basic settings" will suffice. These things have been put under user control so that unusual cases can be handled without the need to recompile the program.

Set Results File Name. Results must be saved to a file which the user specifies. If the file already exists, data will be appended. Each time the program is started, the user must specify the name and there is no default name. The user should be familiar with DOS file-naming rules, including the use of the path. The following box discusses file-naming issues. The discussion applies to all files used by the program, but it is particularly important with regard to the results files, because the user must invent the filenames and must keep track of all the data as they are collected.

Selecting this item brings up a data entry window with a text entry line. The user types the name of a new or existing file. The shortest version of the file name is the *relative path*, relative to the working directory. (A relative path **does not** begin with backslash, '\'.) When the user hits Enter, the full path is displayed. There are then 3 choices, explained in the window: A to accept the name entered, E to edit the name further, or Esc to cancel the entire operation, leaving the results file name as it was before this screen was opened.

File-Naming Considerations

First a caution: Proper DOS filenames do not contain blanks, but it has been found that this program will accept names with embedded blanks, creating files that are difficult to deal with. Use only legal DOS names of the form aaaaaaaa.xxx, where there are 1-8 alphanumeric characters, followed optionally by a dot and 0-3 more characters, called "the extension," with no embedded blanks.

When the program is running, the current default directory is the same as it was when the program was started. It is suggested that the current directory always be the one containing the file `VS.EXE`. This directory should also contain key data files such as `parms.def`, perturbation files, files with lists of speeds, etc.

Here's a simple idea for using directories properly. Put `VS.EXE` in a special directory, such as `C:\VS\`. Do **not** add this directory to the DOS path. Instead, in `C:\`, or another directory that is in the path, put a batch file `VS.BAT`. This short file could contain the single line CD C:\VS . Then, the user can always get started by entering VS. The first time this is done, it sets the current directory (by running the batch file). The second time, it runs the program. The program is always run from the same "current" directory, and it will always use the same copies of needed files, especially those files that have default names. The working directory does not need to be `C:\VS\` of course; it can be `E:\SPEED\`, or whatever is convenient.

In the scenario just described, where essential files are kept in a working directory, the author's preference is to keep the data files in a separate directory, or several of them. Suppose that the working directory is `E:\SPEED\`. A single data directory could be called `E:\SPEED\DATA`. Some data on the XYZ company's model ABC lidar gun may be in a file called `xyzabc.001`. (The user must invent the name.) Assuming now that the program is run from the working directory, the user can refer to this file as `data\xyzabc.001`, or as `\speed\data\xyzabc.001`, or as `e:\speed\data\xyzabc.001`. The first form gives the "relative path," with the working directory as the starting point. Note that all forms do specify `data` as the directory and within VS there is no way to make this a default; it must be specified every time.

The data file can be put in any existing directory on the computer. The program does give the user feedback on these issues by converting the given path to a "fully qualified path," and displaying it.

In many other respects, this software guides the user by providing informative messages and disallowing actions that are out of sequence. In this one area of file naming, the program is a little primitive and the user must take the lead by understanding DOS filenames and using them constructively.

Look at Basic Parameter Settings. Selecting this item from the Files Menu displays the Current Basic Settings in a window that cannot be directly edited. The meanings of these settings were described above.

Read Basic Settings from a File. Selecting this item from the Files Menu allows the user to enter the name of a file from which a set of basic settings will be read. The default file is PARMS.DEF, in the working directory. In routine use, there is no need to read the settings from PARMS.DEF, as this is done automatically when the program is started. This feature would be needed only in an unusual scenario where alternate sets of parameters are in use.

PARMS.DEF, or another file of basic parameters is a short DOS text file (ASCII). It is fairly self-explanatory and could be edited by the user with an ASCII editor. Keywords and data appear as blank-delimited strings. Rather than edit the file with an editor, the user may change the settings interactively within the program, and then save the file.

Write Basic Settings <only> to a small File. Selecting this item from the Files Menu allows the user to name a file and save the current settings to that file. The default file is PARMS.DEF, in the working directory. Changes made to the basic settings are **not** automatically saved, but may be saved through this window.

Interactively Change Basic Parameters. Selecting this item from the Files Menu allows the user to edit the basic parameters on a data entry screen. If the user exits by entering K, the changes take effect immediately. They may be saved to PARMS.DEF or to another file through the Write Basic Settings... selection on the Files Menu.

File PARMS.DEF has a special status. If this file exists in the working directory, it is read automatically when VS starts. In most cases, the user need not give this any thought. If it is desired to make permanent changes in the basic parameters, they should be set as needed and saved to PARMS.DEF through the Write Basic Settings... option.

Default Basic Parameters (Restore 'em). Selecting this item from the Files Menu immediately restores the basic parameters to their default values. The values are then displayed. The values set are the absolute original defaults, stored in the program code. These are the values that are used if PARMS.DEF does not exist when the program starts.

UUT, Change Name of the Small UUT File. Selecting this item from the Files Menu allows the user to alter the file name of a file that, if it exists, will hold the description of the unit under test (UUT). In the normal course of taking data, the user will enter a description of the UUT, and this will be written to the results file. It is needed to identify the test data. As a convenience to the user, the description of the UUT is also stored separately in the file referred to here. There are two ways to use the UUT file:

1. Just forget about it. The UUT data will be stored under the default filename PREVIOUS.UUT. They will be retrieved the next time the program is run and will appear as the defaults on the appropriate data entry screen. In many cases this will be helpful, and no special user action is required.
2. Any number of UUT descriptions can be kept under different names by changing the name of the UUT file through this screen. The reading and writing of this file occur at the time of normal entry and exit of the Identify UUT, etc data screen. In this approach, the user would choose this option on the files menu each time he starts the program.

Quit to Top Menu. Selecting this item from the Files Menu has the same effect as hitting Esc. It closes the Files Menu and returns control to the top menu, with the highlight advanced to Observe Pulse Rep Rate.

Observe Pulse Rep Rate. Selecting this item on the top menu causes the hardware immediately to measure the pulse repetition rate, as discussed in Chapter 4. No further menu appears.

Identify UUT, etc. Choosing this item on the top menu brings up a data-entry first calls up the time display and a reminder to set the clock. Then comes a window something like this:

```
      Manuf: XYZ_____________________________________________________________
      Model: 123____________________________________________________________
  Serial No: 5385_____________________________
Manuf. Date: 11/03/92   ( Month/Day/Year )
   ft_or_m : 1   {0=meters, 1=feet}
speed_unit : 2   {0=m/s, 1=km/hr, 2=miles/hr}
Description: our_permanent_test_unit__________________________________________
Comment:     The new 486 PC_is_now_in_the_loop._____________________________.
Keep these changes? -->  _   { Enter K to Keep, R to Restore earlier values }
Pulse Rep. Rate = {Pulse Rep. Rate NOT MEASURED YET.}
             Hit F1 for Help. <Works on other data screens, too!>
```

The items Manuf:, Model:, Serial No:, Description:, and Comment: are string fields which will accept any combination of letters and numbers, up to a maximum size which is indicated by the number of characters and blanks that are in the field initially. In fact, the 4 longer strings are limited to 63 characters, and the serial number string is limited to 32 characters. It will be helpful to use these strings for the purposes indicated: the manufacturer's name, the model name or number, etc. The date of manufacture must be input in the form indicated, with little flexibility. The items ft_or_m : and speed_unit : are numeric inputs limited to the sets {0, 1}, and {0, 1, 2} as indicated. These parameters convey the system of units that the UUT uses. The settings entered here govern the system of units which is employed on other screens when the parameters of the simulation are entered.

The point bears repeating: This screen, with the heading Edit UUT description, is the key point where the system of units is determined for other operations of data input and output. The only place where an alternate system of units can be used is in the file from which a list of simulation settings is read. (See Appendix S.)

The pulse repetition rate is a feature of the UUT, but it must be measured, not entered manually. If it has been measured, it is displayed on this screen. In the results file, it is saved as part of the UUT description.

Write Header to Results File. Choosing this item on the top menu is the means to save, to the results file, information about the instrumentation setup and the UUT.

What is the "Results File?"

Traditionally, a scientist or engineer maintains a notebook which is a sequential record of what was done, and what results came out of it. A goal in developing VS was that the user should maintain such a notebook, but *the computer should do much of the work.* In a laboratory notebook, there might be a block of information describing the experimental setup: test instruments and unit under test, for instance. Following that there might be a table of data, in which most of the setup stays the same, but one or two input variables vary, and the results vary.

With this in mind, the "Results File" will have a "Header" followed by lines of data collected in the laboratory. If the experiment is conducted properly, the header information applies to all the lines of data immediately following. When part of the setup is changed, a new header should be written, then more data can follow.

When the program opens a "results file," it opens it "for appending." This means that one could continue indefinitely adding data to the same file. In normal operation, the program will not delete or overwrite previous data. For added protection, additional copies of data files should be made at every opportunity, perhaps saved to a different directory, or to a floppy. Limiting the size of each results file also helps to guard against lost work. [Theoretically, the results file could be over-written by saving the basic settings file or one of the other small files using the same filename. Organizing the results files into one or more special data directories makes this less likely.]

The results file is an ASCII file that can be read by humans, incorporated into a report, or printed out. For convenience in controlling mechanical features such as margins and fonts, the user might wish to read the file into a word processing program, then print it out. No special data analysis program has been written to read the results file, although it is organized in such a way as to make this feasible.

The program tracks whether the results file has been used at all; whether the basic parameters have been written to the results file, or changed since; whether the UUT data have been filled in or saved; and whether the pulse repetition rate (PRR) has been measured. According to the status of these items, when the user selects Write Header to Results File, messages and a default action are generated. The messages appear in yellow on black. In a separate window below the messages, a 7-item menu appears. The default action is highlighted, and should be consistent with the status as indicated by the messages

The following 9 items on the top menu were sufficiently discussed in Chapter 4:

1 Speed, 1 Range.
Standard Speed Series.
DT2819 Experiment
Measure PRR repeatedly
Put Pulses on Screw # 11
Alter Perturbation
Clock of DT2819.
Option 12
Quit

Chapter 5: Overall Calibration Test.

This is a test **not** mentioned in DOT HS 808 214, *Model minimum performance specifications for lidar speed measurement devices.* It is not a test of a particular lidar device, but a calibration check on the simulator itself.

The issue of "timing" arises twice in determining the accuracy of a speed-measuring lidar.

Timing issue #1. When the laser pulses, a timing circuit is triggered which then measures the *round-trip time of flight* of the laser pulse. If c_{air} is the speed of light in air, then

$$\text{range} = (\text{time of flight}) * c_{air}/2 \ . \qquad (1\text{-}1)$$

Since the speed of light is about 1 foot per nanosecond (actually 0.299706 m/s = 0.983286 ft/ns), the lidar must measure time of flight with split-nanosecond accuracy. (1 ns = 10^{-9} second.)

Timing issue #2. To determine speed, the lidar must measure the target's range at later times, to determine how much the target has moved in the interval between flashes. Practical lidars flash repeatedly at a steady Pulse Repetition Rate (PRR),

$$(\text{Time between flashes}) = 1/\text{PRR} \ . \qquad (5\text{-}1)$$

The time between flashes is a few **milli**seconds, so **micro**second accuracy or split-microsecond accuracy is needed in setting or measuring the PRR.

The simulator will do its thing with any source of pulses to the trigger input on the Berkeley Nucleonics model 7095 delay generator. The pulses need not come from the flashing of a lidar unit, but they must come from a source that can maintain a stable PRR. Suppose the simulator is set up to simulate a target moving at 50 m/s. (m/s = meters/second.) For this example, it doesn't matter what the PRR is. Let the oscilloscope be set up to record a series of events, logging each trigger pulse and its echo, for as many repetitions as possible. (More later about how to do this.) Then read out the exact time of each trigger pulse and the exact delay from trigger to echo, for each event. Convert delay (simulated time of flight) to range by Eq. (1-1) and graph the range values versus trigger time in seconds. Now the *y*-axis is marked in meters and the *x*-axis is marked in seconds. The slope of this graph should be 50 m/s, the speed being simulated.

Some points of interest:

1. **Timing reference.** The oscilloscope is being used as the reference clock. The oscilloscope could be checked against some other frequency standard. For starters, the oscilloscope can be checked against the DT2819 card.

2. If we accept the oscilloscope as a reference clock, this test puts the whole simulator to a comprehensive test. The DT2819 card, the Berkeley Nucleonics delay generator, the IEEE-488 interfaces, the 486 computer hardware and the simulator software all must do their jobs in order for the series of ranges to plot as a straight line with the correct slope.

3. Besides checking the slope of the line, one can look for other glitches, such as little horizontal flat spots on an otherwise steady slope. A flat spot would indicate that the Berkeley Nucleonics 7095 did not receive a new delay value in time, and re-used the previous value. If one would repeat the whole test at higher and higher PRR's, at some point, the computer and the IEEE-488 link would definitely fall behind, with flat spots as one result. We need to be sure that there are no flat spots at the PRR's of actual lidars being tested. **Section 1 of the standard, on page 9, in effect makes a promise that the simulator will work up to 390 Hz.** It is suggested that it be tested at 400 Hz, as well as a lower PRR or two.

4. When some versions of the simulator have been assembled, a system problem arose in which the BNC 4095 was getting data bytes down the bus faster than it could digest them, resulting in grossly erroneous delays part of the time. This is distinct from the situation just described, of the PRR being too high. If such a thing occurred during this comprehensive test, it would create obvious spikes in the graph. (To eliminate this problem, the vendor of the delay generator replaced some hardware components.)

5. If the graph would be drawn with nanoseconds on the *y*-axis and milliseconds on the *x*-axis, and the simulator would be programmed in miles per hour, then a conversion factor would be needed to convert the slope to miles/hour. The conversion factor would be based on exact numbers such as 5280 feet in a mile, and would not include any mysterious correction factor.

6. This table gives the conversion factor on slope for some specific situations:

unit on *x*	**qty on *y***	**speed unit**	**Conversion from slope to speed**
s	range, m	m/s	1.0000
ms	RT t-o-f, ns	miles/hr	$\frac{(10^{-9})(299\ 705\ 663)(1000)(3600)}{(2)(0.0254)(12)(5280)} = 335.2112$
ms	RT t-o-f, ns	km/hr	$\frac{(10^{-9})(299\ 705\ 663)(1000)(3600)}{(2)(1000)} = 539.4702$
ms	RT t-o-f, ns	m/s	$(10^{-9})(299\ 705\ 663)(1000)/2 = 149.8528$

In this table, "qty" means quantity, "range" means target range, with everything taken into account, in particular the speed of light and the factor of 2 for round-trip. "RT t-o-f" means "round trip time-of-flight;" this means the raw data, taken right off the oscilloscope.

The meanings of the constants are: 10^{-9} s/ns; 299 705 663 m/s = c_{air}; 1000 ms/s; 3600 s/hr; 0.0254 m/in; 12 in/ft; 5280 ft/mile; 1000 m/km; 2 to account for *round-trip* time of flight.

This chapter describes the overall calibration test by reference to a specific oscilloscope, the Hewlett-Packard Model 54522A. The test uses the oscilloscope in a mode that the oscilloscope manufacturer refers to as "repetitive" or "repetitive single shot." This is a specialized mode for recording a series of fast events with long delays between events. Among other things, this chapter addresses some rather non-trivial issues in setting up the oscilloscope and then reading the data out. If another oscilloscope is used, these issues may take a different form, with different terminology.

Experimental Procedure:

7. To generate the graph of delays versus trigger time, it is first necessary to set up the oscilloscope and collect the data. It is then necessary to manipulate the oscilloscope further to get the data out. The following directions assume a Hewlett-Packard 54522A oscilloscope.

8. **Setting the time base.** The mode in which the oscilloscope can record a series of pulse pairs is called "sequential single-shot." In this mode, the sampling rate bears a fixed relationship to the time base, even though in other modes these parameters can be set independently. "Time base" is what you might think of as the scaling on the x-axis, expressed in time/division. In order to set up the oscilloscope with confidence, refer to Table 4-1 from the oscilloscope manual, page 4-13.

   The delay generator can be programmed in increments of 50 ps. (ps = picosecond = 10^{-12} s) The jitter on delay may exceed 50 ps. Pulse timing should be measured with split-nanosecond accuracy. Since the highest available sampling rate is 2 GSa/s, or twice per nanosecond, it is desirable to take all data at the highest sampling rate. (GSa/s = gigasamples/second = 10^9 samples/second; ns = nanosecond = 10^{-9} s) This means that the *time base for data collection* should be set between 500 ps/div and 20 ns/div. (div = division on the scope screen ≈ 1 cm.) After the data have been collected, the time base can be changed for display purposes.

   Even though the oscilloscope can "only" make a measurement every ½ nanosecond, it can measure delays with better than ½-nanosecond precision, by interpolation. Letting the sampling rate drop to some lower value such as 1 GSa/s or 250 MSa/s, however, will introduce erroneous flat spots into the graph of delay versus time.

   Let me re-iterate the original point here. When you use the oscilloscope in a normal way to watch a series of similar waveforms as they happen, you can control the display time base and the sampling rate independently. **When you set the oscilloscope to collect repetitive single-shot data, you set the time base, and the sampling rate is set automatically according to Table 4-1 in the oscilloscope user's manual.**

9. **What pulses to clock?** It would be most logical to capture the pulse going into the delay generator, at the trigger input, and the delayed pulse coming out on the right-hand side of the delay generator. This would involve the fewest assumptions. Unfortunately, this might give problems of available signal power, false triggering due to ringing at a coaxial tee, etc. It is

easier to measure the delay from the "initial pulse" output of the delay generator to the delayed pulse. It might be even easier to use one oscilloscope channel only, and measure the width of the "gate" pulse that is also available from the delay generator. Measuring the width of the gate pulse will make the manipulations of the oscilloscope easier. **You might want to verify the relationships of the pulses as described in Fig. 1-1 of the Berkeley Nucleonics manual.**

In any event, you choose two pulses to clock, or the gate pulse. For purposes of discussion, assume that you are going to clock the delayed pulse with respect to the initial pulse. This is the more complicated case.

10. **Oscilloscope connections.** Connect the Berkeley Nucleonics 7095's "initial pulse" output to the scope's Channel 1. Connect the delayed pulse output to Channel 2. The initial pulse output will also be going to the serial port on the computer, so you need a "tee," or some other means to tap into that circuit. If you can arrange to bring the signal to a small screw or a loop of wire, then you can use an oscilloscope probe for the initial pulse signal, causing less loading and less ringing. You can bring the output pulse directly to the oscilloscope via coax. Set each scope input for the proper input impedance. If you use a probe, manually notify the scope of the ×10 multiplier. Adjust the scope to trigger on the initial pulse, and adjust the vertical scales to display the pulse shapes fully without clipping. In general, expanding the vertical display without clipping reduces digitization error.

    Set the trigger level **near the midpoint of the initial pulse leading edge;** this gives reliable triggering and is especially important because of the way the data will be read out later. "Near the midpoint" means something like the middle 1/3 of the rising edge; it does not mean that you have to locate the midpoint exactly.

11. **Oscilloscope timing setup.** When the connections, triggering, and voltage gain are all set to display one data set at a time, you can set up the sequential data collection. The sequential softkey is explained in the *User's Reference*, pp 4-23 to 4-28. The illustration on p. 4-23 shows the softkey display for this step. On this one screen, the time base, the number of points, and the number of segments can be set. The number of points per segment and the number of segments are limited by the size of "sequential memory." Consider an example:
    a. Target will have an initial range of 500 ft, and approach the observer (speed > 0). The round-trip distance is 1000 feet, and the speed of light is about 1 ft/ns, implying a maximum delay of *about* 1000 ns. Using the value $c = 0.983286$ ft/ns shows that the maximum delay should be 1017 ns. We actually want more than 1017 ns worth of data, in order to see the second pulse in more detail. Say that we would like 1050 ns of data.
    b. The time base can be set to any of the 6 fastest settings, so that the sampling rate is 2 GSa/s = 2 samples/ns. Therefore, the number of points should be set to 1050×2 = 2100. When this is set, the number of segments box will show max# = 93. (According to page 4-24 of the *User's Reference,* 200K points can be recorded; simple math suggests max# = 97.)
    c. The number of segments in one simulation depends on target speed and PRR. Say speed = 60 mph = 88 ft/s, and PRR = 125 Hz. Working in *very approximate* numbers, we can see that it will take over 5 s for the target range to run down to zero,

and the memory segments will be used up in less than a 1 s. Therefore, the number of segments should be set to the maximum, to collect as many data as possible.

d. Actually, it may take more than 93/125 s to fill the oscilloscope's memory, because the scope may not be ready for some events. This does not invalidate anything so long as the data are plotted based on actual trigger times.

12. **Run the test.** When all this setup is done, record all setup conditions on the data sheet, especially PRR, initial range, and simulated speed. Double-check that simulated speed is recorded with proper units. Interrupt the pulse input to the delay generator, so the simulator is inactive and reset to initial range.

a. Press the oscilloscope's Run button; the display should show that it is waiting for a trigger.

b. Now start the pulses. The scope will show that it is "Acquiring."

c. Within a second or two, the data will be collected. Then the display will show "stopped - sequential data processed, select DISPLAY menu."

13. **Record the data.** Now the numbers are in the scope, but they are not displayed in a list. You have to find what you want.

a. Press the Display button.

b. Press the blue button, then Δtime. Using the entry knob and various buttons, set the Δtime measurement to display (channel 2) - (channel 1). Chapter 13 in the user's manual discusses the meaning of this and other measurements. Essentially, you are finding the time from the midpoint of the rising edge in channel 1 to the midpoint of the rising edge in channel 2. Channel 1 is acting as a proxy for the trigger time.

c. Be sure that the display is set to display channel 2. Actually, both channels will be displayed, but this means that the "entry" knob in the upper right-hand corner of the front panel will control which segment of channel 2 is displayed.

d. Note that now you can adjust the time base as you wish, to display the waveforms. The data are "in the can" and sampling rate is now independent of time base.

e. Now as you slowly turn the "entry" knob, the successive channel 2 pulses will be displayed; in the example above, you would see the target approaching the policeman, as each pulse shows a little less delay. There is an irrational element here, in that the channel 1 display is static. The Δtime measurement will give you the delay between pulses *that are not part of the same event.* This may seem irrational, but it works, because the channel 1 pulse simply serves as a proxy for the triggering event; the trigger should have occurred in response to channel 1, but probably not at the exact threshold used here.

Theoretically, you could advance through the segments in both channels, but it takes a lot of button pushing and knob twisting to do so. This method should suffice. Keeping the trigger level near the midpoint of the channel 1 waveform makes the channel 1 time and the trigger time better proxies for each other. (Trigger level was set in step 10 above.)

f. When it is clear that the entry knob is stepping through segments of Channel 2 data in the expected way, go to Segment 1 and begin to write down the data with a pencil. The trigger time will be indicated as something like "ch 2 segment x.ddddddddd ms," and the delay will be shown as Δtime. It is suggested that you record all or most of the significant figures.

14. **Graph the data.** When the numbers are written down, you need to graph them. The most practical way to do this is with a computer program that can draw the graph and fit the best straight line by the method of least squares. This will give a slope value, which can be compared to the simulated speed using the unit conversion factors given previously.

15. **Evaluating results.** In one trial of this test, the simulated speed was set to 160 mph, with PRR = 381. The slope of the graph showed the simulated speed to be 160.123 mph. In this case, the error was slightly less than 1 part in 1000. A similar test with PRR = 400 gave a slope corresponding to 159.98 mph, even better agreement. No specific tolerance has been established, but if results come out significantly worse than these, you should look for a source of trouble. Any obvious departure from straight-line data is a concern and should not be there.

It is important that a detailed record of the test be preserved, including the PRR, the setup of the simulator, and the setup of the oscilloscope, as well as the data.

Appendix B:
Format of the Basic Settings File

Seven of the parameters that govern hardware-software interactions are stored in this file. The use of this file, and the meanings of the parameters, are discussed in Chapter 4. The default filename is PARMS.DEF, and there are default parameter values, which are used in this example:

```
BasicParms     1997 Nov  3, 15:10:11
filename       parms.def
CommSelect     1    {1=COM1, 2=COM2}
TimeOut        100000
QueueSize      12
RtsPulseFlag   0
BOARD488       0
BNC7095        6
DT2819Base     230
```

When read by the program, each line is treated as a collection of blank-delimited strings. Each line begins with a keyword, and the 9 lines must appear with the 9 keywords exactly as shown, except that case (capitalization) is ignored. One blank terminates a string, such as the keyword, and then additional blanks are ignored. On the first line, the keyword is followed by 4 strings noting the time that the file was written by the program. When the file is read by the program, the time information is ignored. On the remaining 8 lines, the string after the keyword is the parameter value, as explained in Chapter 4. Any strings after the parameter value are ignored. For instance, on the third line, the comment `{1=COM1, 2=COM2}` is ignored. It is not a comment because of the curly braces, but because it consists of strings number 3 and number 4 on the line, and only strings 1 and 2 are used.

Since the keywords are always the same, one could say that they are redundant. The idea is to make the file human-readable in a critical way. A human can display the file with a command as simple as type parms.def . The keywords are trustworthy labels, because if any keyword is missing or out of place, the program will give an error message.

In this file, all the numeric values are clearly integers. The last is a hexadecimal integer that could include symbols {A-F}. *In this and all files provided to VS, it will be safest and most logical to write integers as integers, without a decimal point.* In other files, numerical values such as range and speed values are "real numbers," not necessarily integers.

Appendix P:
Format of the Perturbation File

The perturbation must be specified through an ASCII file that the user prepares before running VS. The lines below `in this courier font` are the listing of a file, which has been called `NISTPERT.STD` , but could be called anything:

```
PERTURBATION
this_file      NISTPERT.STD
ft_or_m        1    {0=meters, 1=feet}
DESCRIP  "This is the NIST standard perturbation file."
* This is a comment line, but the previous line isn't!
* tSec   dRangeFt
0.0       0.0     ; First time MUST BE 0.0; first range could be nonzero
0.010     0.0     ; Try to catch 1 flash before jump
0.012     5.0
0.200     0.0     ; Comments may follow the data
-1.0     -1.0     * Provided they start with * or ;
* time < 0.0 (i. e., minus time) may be used to terminate the list

Comments should begin with *, but this line will not cause trouble.
* Don't put more than 32 separate strings on a line.
* This line has 5 strings.
"This is only one string because it has quote marks!"
```

Syntax. Broadly speaking, the file is free-form. The necessary words and numbers may be separated by one or more spaces or tab characters. The user can create a valid file with different data by systematic editing of the example file. It should not be necessary to study the syntax rules closely. Nonetheless, here they are:

1. The file is read one line at a time. Each line is then processed one string at a time, where a string is

- a group of characters separated by white space, *or*
- any characters, including white space, enclosed in double quotation marks, such as
  "This is one string because it is enclosed in quotes."

The quotation marks are *not* part of the string. For the purpose of all files read by this program, white space includes space, tab, *and comma.* Where certain strings have special meanings, the program is not case-sensitive.

2. The first line of the file must read PERTURBATION . In accordance with rule 1, capitalization is optional.

3. The next three lines specify parameters. The first string is the parameter name and the second string is the parameter:

a. Parameter `this_file` should be the name of the file itself; it could be either the complete path, or the path from the working directory. If it does not agree with the actual file from which the data are being read, a warning is issued.

b. Parameter `ft_or_m` should be 0 or 1. 0 means the range values are in meters; 1 means they are in feet.

c. Parameter `DESCRIP` is a string that will be displayed when the file is put in use.

These parameters may be given in any order. Note that only the first 2 strings on each parameter line are interpreted, so additional strings may be added as comments. In most parts of the program, the units of measure used are those of the unit under test, as specified by the user. This is the one other place where units may be specified; the range and speed values from this table will be converted to the units of the unit under test, if different.

4. Any line beginning with * is a comment. Comments may appear anywhere.

5. If a line after the first does not begin with a parameter name or a *, it is considered to be a time-distance pair, and the first two strings should be a time and a displacement, given as decimal numbers. If the string following the displacement begins with asterisk (*) or semicolon (;), the rest of the line is treated as a comment. Up to 32 pairs may be used; further data will be ignored. The special pair (-1, -1) may be used to terminate the list. All data after (-1, -1) are ignored.

6. The first time value must be 0.0 . After that, each time value must be greater than the one preceding. In other words, the time series must begin with 0.0 and increase monotonically. The last time value must be less than 200 pulse periods of the UUT, approximately. This limit is approximate because of the intricate method by which these inputs get translated into a repeating perturbation.

7. In general, if one of these rules is violated, an explanatory message is displayed.

Appendix Q: The Role of the Queue and "minimum depth reached by the queue."

Computer magazines in the era 1990-1995 made much of the fact that OS/2, Windows NT, and Unix are multitasking operating systems, while Windows 3.1 is not. The reality is messier: even the original PC BIOS is multitasking; the BIOS forces the PC to stop whatever it is doing 18.2 times per second in order to update the real-time clock. Operating systems such as DOS 5.0 perform operations such as parking the heads on the hard disk, possibly interrupting a program to do so.

The simulator program is written in a somewhat intricate way to make sure that the computer responds quickly after the UUT has flashed, in spite of any unpredictable actions by the operating system. In setting up a simulation, the program loads one delay into the delay generator, and pre-calculates 12 more values that are put into a queue. (Queue depth is a parameter that the user can set, but the default is 12.) When the UUT flashes, and a pulse is relayed to the computer via the serial port, an interrupt service routine, which is fast and has high priority, pulls one delay from the queue and transmits it over the IEEE-488 bus. In the absence of any other activity, the interrupt service routine would pull a delay from the queue with each flash of the lidar, until they were all used up.

The higher level program, in the subroutine that runs while the simulation window is displayed, monitors the queue depth. When the queue size drops below the target value, the next delay is calculated and put on the queue, incrementing the queue size. This scheme puts some flexibility into the software, so that it can transmit all delay values on schedule. The development of this method and the programming of the interrupt service routine were done by Sam Andrews of Laser Technologies Incorporated.

If the simulation hardware and software interact in the smoothest possible way, the queue size will only drop to one less than its maximum. It will repeatedly drop to this value, but it will be immediately refilled, before the next flash of the lidar. This is what has usually been found to happen at the pulse repetition rates of commercial lidars, and even at higher PRRs. If the queue would occasionally drop below its maximum by more than one count, this could be construed to mean that it is doing its job. To this extent, the minimum queue depth means nothing.

Now consider the practical problem of verifying simulator operation, when a new simulator is assembled, or an existing simulator is operated under new conditions, such as a higher pulse repetition rate. During a simulation, complex events are happening very quickly, and there is no comprehensive way to track the entire sequence of pulses, program steps, and data moving down the cable. In this context, it is valuable to have some simple clue that the simulator is or is not operating smoothly. The minimum queue depth is one such indicator.

Consider some hypothetical examples. Suppose that pulses to drive the simulator are taken from a controllable source, such as the internal counter-timer card. (In Chapter 3, see the section Put Pulses on Screw # 11, on page 20.) Initially, PRR is set to 200 Hz. The queue depth is 12. After a simulation has run for 30 seconds and terminated, the program reports "Minimum depth reached by queue = 11 √." This is a perfect score, meaning that whatever is happening, there is enough time for it to happen. Now PRR is set to 450 Hz, and the simulator again is run for 30 seconds. This time the queue depth falls to 5. At a slightly higher PRR, queue depth falls to zero. (When queue depth has fallen to zero, the user may well find that the entire computer program has crashed.)

After running simulations at several pulse repetition rates, we can infer that the particular simulator is reaching its speed limit at 450 Hz. Further testing would be in order if an actual lidar required testing at this PRR. Again note that the PRRs given are hypothetical and the detailed test of Chapter 5 is sensitive to various errors, including those that may arise as PRR is increase.

Appendix R:
Format of the "Results File"

The "Results File" is the primary means for recording data from experiments with the simulator. The simplest results file will have a header block describing the "basic settings" of the simulator itself, and describing the unit under test, UUT. Following the header, some number of data lines will appear, stating the parameters of the simulated target, and the associated readings from the UUT.

In fact, the header consists of two parts:

The "Basic Parameters" block. In 9 lines, this block records 7 of the settings that govern the interactions of hardware and software. This block is the same as the Basic Settings file explained in Appendix B.

The "Unit Under Test" block. This is the same as the UUT file explained in Appendix U.

The program user normally saves both parts of the header at the beginning of a session, and then may save either or both parts at will, normally after some information has changed. Following is a fragment from an actual data file, except that various information identifying the UUT has been edited.

```
BasicParms      1994 Sep 15, 15:38:08
filename        parms.def
CommSelect      1    {1=COM1, 2=COM2}
TimeOut         100000
QueueSize       12
RtsPulseFlag    0
BOARD488        0
BNC7095         6
DT2819Base      230

UUT             "1994 Sep 15, 15:38:10"
manuf           "XYZ Lidar Company"
model           "Super Laser"
serial          "SL0013"
mfgdate         "1994 Jul  1, 00:00:00"
UnitOfLength     1    {0=meters, 1=feet}
UnitOfSpeed      2    {0=m/s, 1=km/hr, 2=miles/hr}
descrip         "Using fiber optics, with 2 bundles back into the
XYZ rcvr."
comment         "Simulated FM transceiver, RF pwr 10mW, deviation 5
kHz."
PRR             138.4175

result  1994 Sep 15, 15:38:21  4400.00    40.00  -999.00  -999.00
* Just get "RFI" when we squeeze the trigger at 10 o'clock.
result  1994 Sep 15, 15:39:56  4400.00    40.00  4420.00    39.00
* 9 o'clock, still +10 dBm, 1500 Hz.
result  1994 Sep 15, 15:40:53  4400.00    40.00  4420.00    39.00
* 8 o'clock, +10 dBm, 1500 Hz
result  1994 Sep 15, 15:41:30  4400.00    40.00  4439.00    39.00
* 7 o'clock, +10 dBm; 1500 Hz
result  1994 Sep 15, 15:42:06  4400.00    40.00  4438.00    39.00
* 6 o'clock, still +10 dBm, 1500 Hz
result  1994 Sep 15, 15:42:50  4400.00    40.00  4420.00    39.00
* 0 dBm, 3 o'clock on the vernier, 1500 Hz
```

Following the header blocks are the actual results. The data are from a radio-frequency interference test. References such as "`8 o'clock`," refer to the setting of a knob, not a real clock. After the user runs a simulation, he must type in the readings from the display of the UUT. (There is no provision in the software for accepting the data over a wired connection.) Then he may *optionally* add a comment. On the actual data line appears the keyword `result`, followed by the time, the range and speed settings of the simulation, followed by the range and speed readings from the UUT. If the comment appears, it is on the line following the related data, and the line begins with *.

The units of the data are those specified in the UUT segment of the header. In this case, the ranges are in feet, and the speeds are in miles per hour. Recall that the range setting of the simulation is the initial range, and is not corrected for the insertion delay of the delay generator and the cabling. The unit under test will read some sort of average range during the measurement. The simulated and measured speeds are expected to agree closely, but the range numbers will in general not agree.

Appendix S:
Format of the "Standard Speed Series" File

Example file. The "standard speed series" file must be prepared in advance using an ASCII editor. The lines which appear below `in this Courier font` are from a file which is properly read by the program. A file similar to this is supplied with the program and other speed series may be generated by editing that file.

```
STANDARD SPEED SERIES
this_file      SPEEDS.LST
ft_or_m        1    {0=meters, 1=feet}
speed_unit     2    {0=m/s, 1=km/hr, 2=miles/hr}
*  Example file of the type from which we read a set of ranges &
speeds !
*  Jim Worthey, 1993 August 10
* Range     Speed
200          0.0
210         10.0
220         20.0
230         30.0
240         40.0
250         50.0
260         60.0
270         70.0
280         80.0
290         90.0
300        100.0
310        110.0
320        120.0
330        130.0
340        140.0
400        200.0
600.      -200.0
590       -190.0
580       -180.0
570       -170.0
560       -160.0
550       -150.0
540       -140.0
530.0     -130.0
520.0     -120.0
510.0     -110.0
500.0     -100.0
490        -90.0
480.0      -80.0
470        -70.0
460        -60.0
450        -50.0
440        -40.0
430        -30.0
```

```
420          -20.0
410          -10.0
400           -0.0
-1.0          -1.0
```

Syntax. Broadly speaking, the file is free-form. The necessary words and numbers may be separated by one or more spaces or tab characters. The user can create a valid file with different data by systematic editing of the example file. It should not be necessary to study the syntax rules closely. Nonetheless, here they are:

1. The file is read one line at a time. Each line is then processed one string at a time, where a string is

- a group of characters separated by white space, *or*
- any characters, including white space, enclosed in double quotation marks, such as

"This is one string because it is enclosed in quotes."

The quotation marks are *not* part of the string. For purpose of this file, white space includes space, tab, *and comma.* Where certain strings have special meanings, the program is not case-sensitive.

2. The first line of the file must read STANDARD SPEED SERIES . In accordance with rule 1, capitalization is optional and additional white space could appear between the words.

3. The next three lines specify parameters. The first string is the parameter name and the second string is the parameter:

a. Parameter `this_file` should be the name of the file itself; it could be either the complete path, or the path from the working directory. If it does not agree with the actual file from which the data are being read, a warning is issued.

b. Parameter `ft_or_m` should be 0 or 1. 0 means the range values are in meters; 1 means they are in feet.

c. Parameter `speed_unit` should be 0 (meaning m/s), 1 (km/hr), or 2 (miles/hr).

These parameters may be given in any order. Note that only the first 2 strings on each parameter line are interpreted, so additional strings may be added as comments. In most parts of the program, the units of measure used are those of the unit under test, as specified by the user. This file and the perturbation file are the other places where units may be specified; the range and speed values from the table will be converted to the units of the unit under test, if different.

4. Any line beginning with * is a comment. Comments may appear anywhere.

5. If a line after the first does not begin with a parameter name or a *, it is considered to be a range-speed pair, and the first two strings should be a range and a speed within the simulator's range, given as decimal numbers. Again, strings after the second are ignored. Up to the 100 pairs may be used; further data will be ignored. The special pair (-1, -1) may be used to terminate the list. All data after (-1, -1) are ignored.

Appendix U:
Format of the UUT File

As explained in Chapter 4, the description of the UUT is read from and written to a file with default filename PREVIOUS.UUT. Here is an example of such a file:

```
UUT             "1995 Jun  6, 16:35:37"
manuf           "XYZ"
model           "123"
serial          "5385"
mfgdate         "1992 Nov  3, 00:00:00"
UnitOfLength     1    {0=meters, 1=feet}
UnitOfSpeed      2    {0=m/s, 1=km/hr, 2=miles/hr}
descrip         "our permanent test unit"
comment         "The new 486 PC is now in the loop."
PRR             149.9999
```

As with other files, the file is read by lines, and the lines are read as a series of blank-delimited strings. Material in quotation marks is treated as a single string. Each line begins with a keyword, and the 10 keywords always appear in the order shown. When the file is read, keywords must appear as shown, except that case (capitalization) is ignored. The first keyword, UUT, identifies the file type. It is followed by a time stamp, showing when the file was saved. On the next three lines, the keywords are followed by strings as input by the user. The user inputs the date of manufacture in numerical format, but on the next line it is presented as a string. The unit of length and unit of speed are designated by integers; when the file is read, comments are ignored, not because of the curly braces, but because strings after the second are ignored.

"Descrip" and "comment" are further strings input by the user. The PRR is a measured value which is not used when the file is read. The user must re-measure PRR each time the program is run. (The numerical PRR given here is a made-up number.)

Appendix W:
Wiring Details

Cabling context. Pulses pass in and out of the digital delay generator through BNC connectors on the front panel. The receive-send unit is not tightly specified, but can also have BNC's for the pulse inputs and outputs. If the terminations at the switch panel are made with BNC connectors, then most of the signal cables can be lengths of coax with a BNC on each end, everyday laboratory items.

One special cable. A special cable must be made up to bring a trigger pulse from some point such as the "initial pulse" output of the digital delay generator to one of the serial ports on the PC, either COM1 or COM2.

One end of this cable terminates in a BNC, while the other end is a 9-pin RS-232 connector (DB-9), or conceivably a 25-pin RS-232 connector (DB-25).

The wiring at the RS-232 end can be summarized in this way:

Signal name	Pin# if 9-pin	Pin# if 25-pin	Connect to
Signal ground	5	7	Shield of coaxial cable.
Signal ground	5	7	Bring out on a short black lead with a termination such as a test clip.
DSR	6	6	Central conductor of coaxial cable.
DSR	6	6	Short red lead with a termination such as a test clip. Makes the pulse available to an oscilloscope probe, for instance.
RTS	7	4	Short yellow lead with a termination such as a test clip. If "basic setting" RtsPulseFlag is nonzero, the program will pulse this line after a delay value has been loaded to the delay generator.

The function of this cable is to notify the software (VS.EXE) that the lidar has fired a pulse. The notification pulse is taken from the "initial pulse" output of the digital delay generator. This pulse can be sent to COM1 or COM2; an interactive screen lets you set the port in software. The initial pulse output serves a pulse-stretching function, since it can be adjusted in amplitude and width. When this pulse is received by the COM port, it triggers an interrupt service routine which uses the IEEE-488 bus to set up the next delay in the delay generator. Technically, a race condition is set up, because the interrupt is triggered before the delay generator has measured off the previous delay. The race is over in a MICROsecond or 2, and this has caused no trouble. It can take a couple MILLIseconds to complete the data transfer over the IEEE-488 bus.

Connections at the DT-758 screw terminal board. The usage of the connections to the counter-timer card is discussed repeatedly in the body of this user manual. It is assumed in every case that the optional cable with screw terminal board has been purchased. The following table will summarize the usage of all the screw terminals that must or may be used in the target simulator application.

Connections at the DT-758 screw terminal board		
Screw Terminal(s)	**Function**	**Comment**
3	Pulse output of counter/timer 1.	**Must** be connected by a jumper to terminal 20.
20	Gate input of counter/timer 3	Receives input from terminal 3.
2, 4, 6, 8, 10, 12, 15, 18, 21, 24, 27, 29	Digital ground	
11	Pulse output of counter/timer 5.	Source of pulses for test purposes, controlled by the program
13	Pulse input for counter/timer 1.	Input for the program's determination of the UUT's PRR.
3, 5, 7, 9, 11	Outputs of counter/timers 1-5	LED's connected from these terminals to ground will blink in the DT2819 Experiment, selectable from the top menu. Since there is also a screen display, this is non-essential, unless one would encounter a stubborn problem involving the counter-timer card.
For further explanation of the screw panel, see the DT2819 User Manual, Table 1, p. 15.		

Switch Panel. In everyday use of the simulator, the PRR of the unit under test is first measured, then the simulation is run. For the PRR measurement, the Initial Pulse output of the delay generator must go to screw 13, but during the actual simulation, the same Initial Pulse must go to the computer's serial port. A switch must be provided to make this wiring change conveniently and without excessive wear to the connectors. Other wiring changes may be needed to interrupt pulses to the transmit input of the receive-send unit, to run the calibration procedure of Chapter 5, or for setup and troubleshooting.

The table on the next page shows switching requirements for some potential modes of operation. On the pages after that, figures 2-4 offer plans for a switch panel. Simpler switch panels have been used, but this design gives great flexibility. It is intended that the substrate holding the switches and connectors be clear plastic so that the signal paths are obvious. Referring to figure 4, the BNC connectors should be labeled as indicated, Com Port, Initial Pulse, Rx, Delayed Pulse, Tx, and 7095 Trigger. The label Screw 11 Pulses should also be affixed. The other items should **not** be labelled on the actual panel. The table following figure 2 gives the hole diameters for that figure, and can be adjusted if non-identical parts are used.

Switching Requirements	
Count real pulses in context of a real experiment	#11 to noplace or to LED
	Rx pulse to 7095 trigger input.
	Initial pulse output to #13 for counting.
	Delayed pulse from 7095 must not go to Rx-Tx unit, *or* the loop must be broken optically, for some UUTs.
Simulation running.	#11 to noplace or to LED
	Rx pulse to 7095 trigger input.
	Initial pulse output of 7095 goes to the com port input of the computer.
	Delay output of 7095 must now go to the Tx input of the Rx-Tx unit.
Run phony pulses to 7095 for testing. For instance, troubleshooting or refined calibration of the delay series, with receive-send unit out of the loop.	#11 goes to trigger input of 7095.
	Rx can go noplace.
	Initial pulse output of 7095 goes to the comm. port input of the computer.
Counter-timer testing only.	Screw #11 goes to LED.
	No input to 7095. Rx can go noplace.
	Initial pulse output of 7095 open.
Count phony pulses from screw #11.	#11 goes to #13 for counting.
	No input to 7095. Rx can go noplace.
	Initial pulse output of 7095 can go open.
Count phony pulses through 7095.	#11 goes to trigger input of 7095.
	Rx can go noplace
	Initial pulse output of 7095 to #13 for counting.

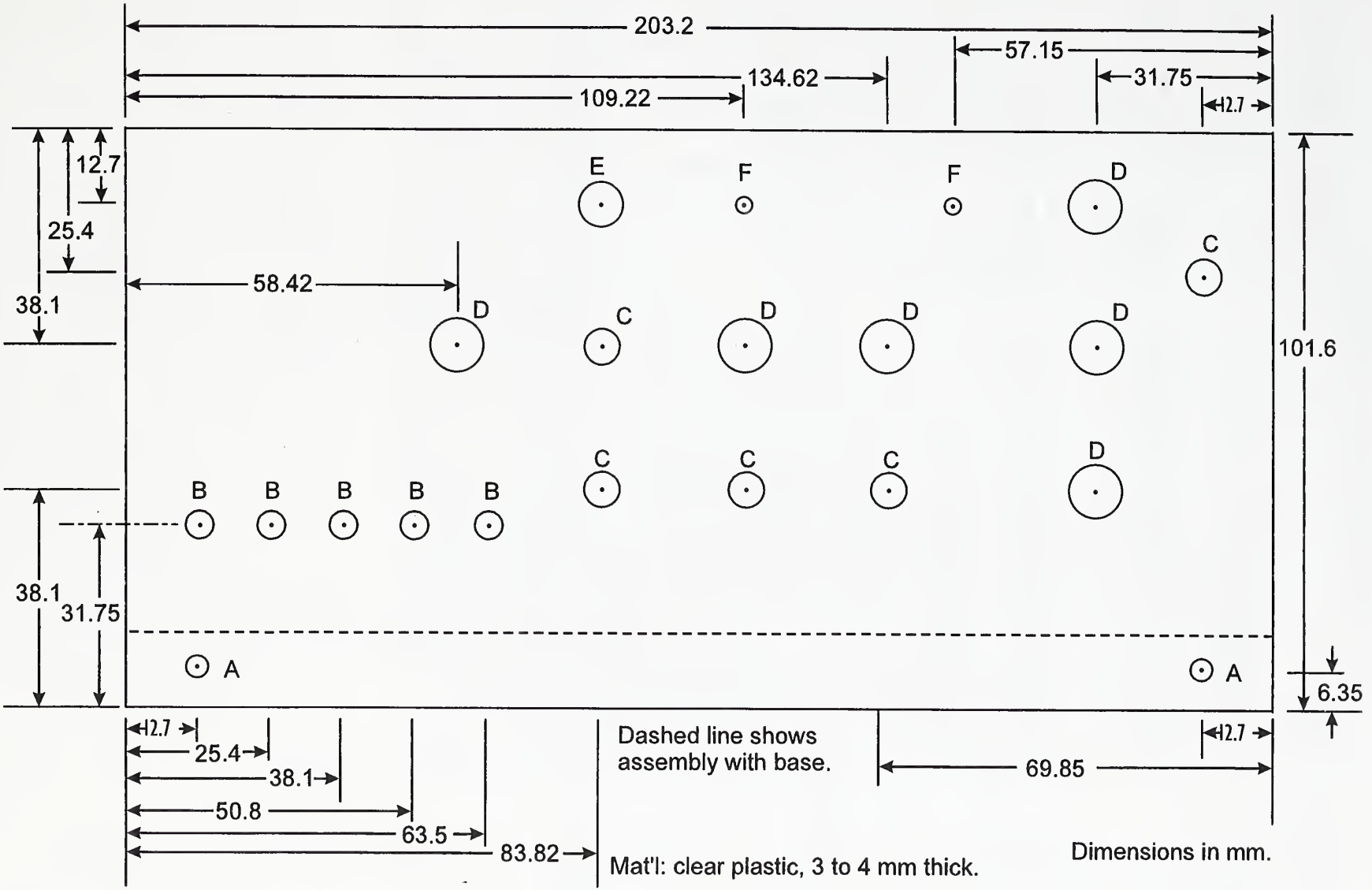

Figure 2, front panel. Hole diameters according to table.

Table of Hole Diameters for Switch Panel			
Designator	Description	Estimated Diam.	Final Diameter or Drill #
A	To clear 6-32 screws that anchor this panel to the base.	0.136"	5/32 = 0.156" = 3.962 mm
B	LEDs - care in drilling or reaming these holes can give a snug fit, avoiding glue.	5 mm 5.00 mm meas	#9 = 0.1960" = 4.978 mm (real tight)
C	Mounting hole for subminiature toggle switch.	6.28 mm meas	0.250" = 6.35 mm
D	Mounting hole for BNC chassis connector	9.525 m m	3/8 = 0.375 = 9.525 mm
E	Mounting hole for banana socket	7.72 mm	5/16 = 0.3125 = 7.938 mm
F	Hole for test point. Suggested test point is a 4-40×1-3/4 or 1½ screw with 4 or more nuts, set so that it protrudes front and back of the panel. Stiff wire may be soldered on to provide additional anchorage for scope probe. Solder lugs in back allow wiring.	2.92 mm	#32 = 0.1160 = 2.946 mm
Please note	These diameters are based on parts available in Gaithersburg and work done by a professional machinist. For instance, the 5 mm LEDs are a common size. By doing a little trial-and-error on scrap material, it was possible to achieve a friction fit on the LEDs and an attractive result. Parts should be obtained first, and checked for size. For identity of parts, see figure 4, the wiring diagram, and its accompanying table.		

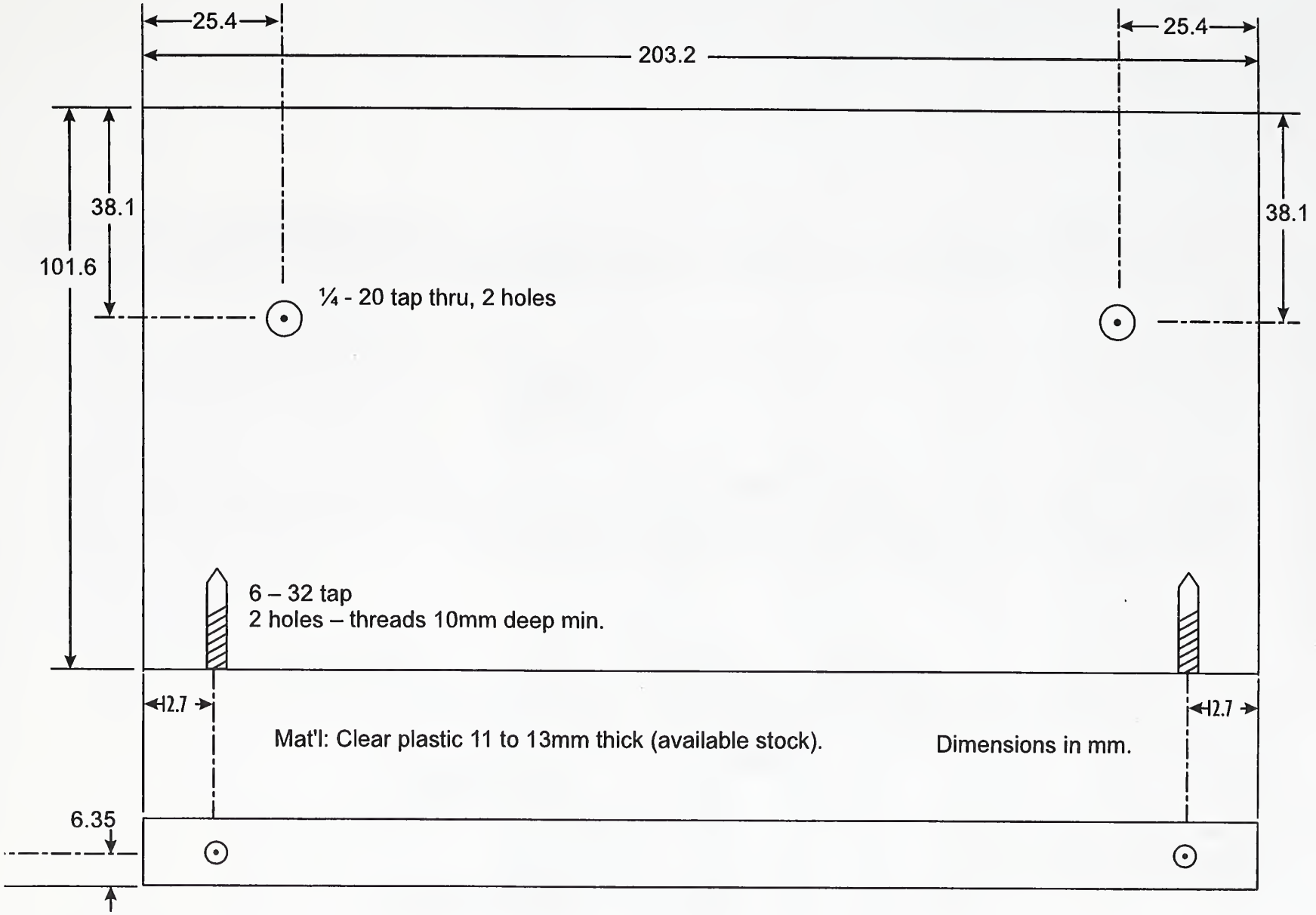

Figure 3, base. The author's preference is to assemble the front panel and the screw terminal board to this piece with cap screws.

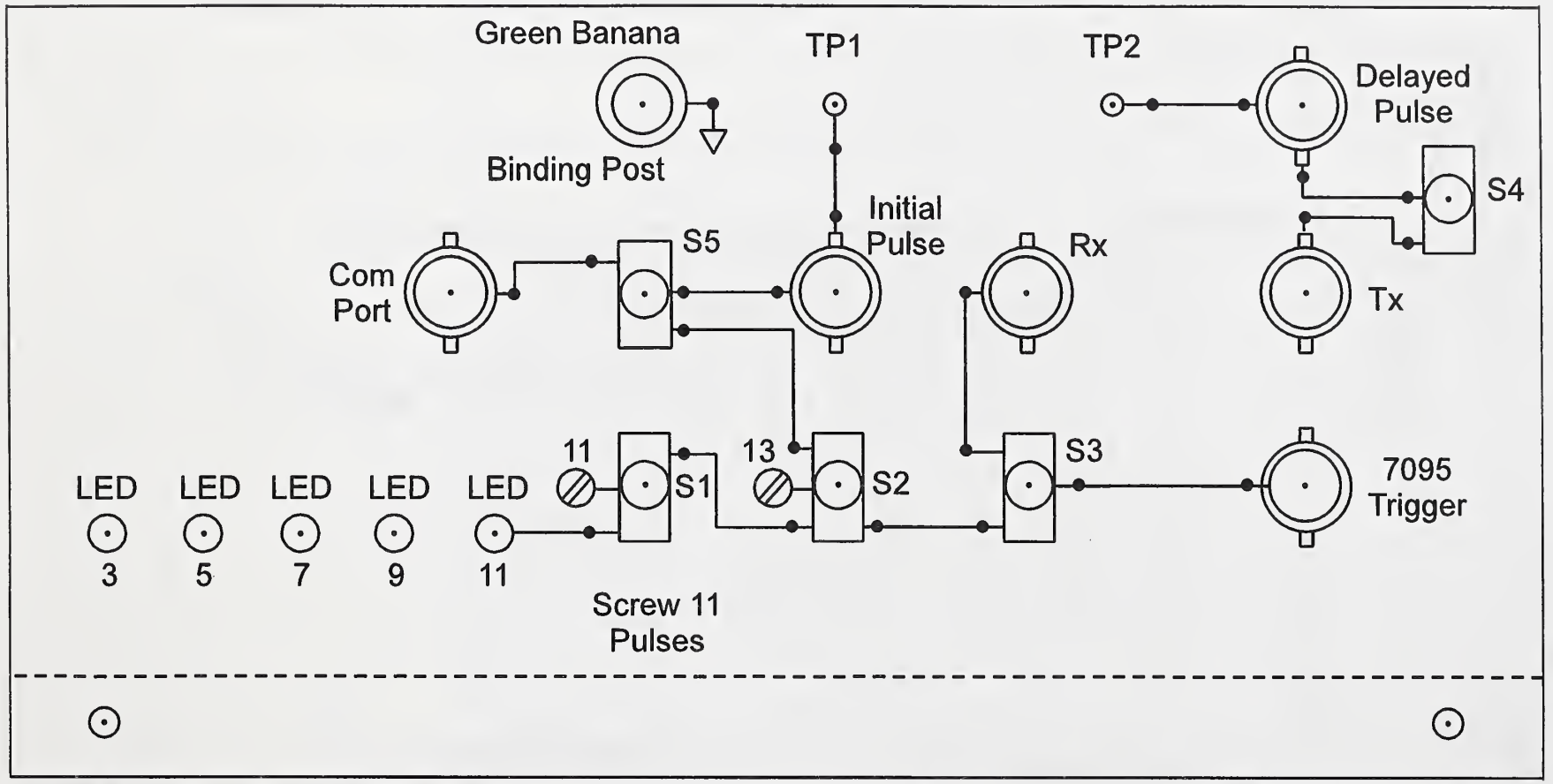

Figure 4, Parts layout and wiring diagram. Labels on the BNC connectors and the label "Screw 11 Pulses" should appear on the finished panel. Other labels should not appear.

Key to Wiring Diagram, Figure 4.		
Designation	**Description**	**Switch positions**
LEDs 3, 5, 7, 9	Connected from screw 3, 5, 7, 9 to a ground screw. Cathode to ground; ballast resistor not needed.	
LED 11	According to switch, either unused or displays pulses from screw 11.	
S1	Subminiature toggle switch, SPDT, on-off-on. Select destination for pulses from screw #11.	LED 11
		open
		Pulses go onward.
S2	Same device as S1. Select PRR counting input, screw #13.	Direct from screw #11
		open
		From initial pulse output of delay generator
S3	Same device as S1. Trigger source for the delay generator.	Pulses from screw #11
		open
		Pulses from Rx (receive) output of receive-send unit.
S4	Subminiature toggle switch, SPDT, on-none-on. On-off switch for the Tx (transmit or send) input of the receive-send unit.	Optical return pulses on.
		Optical return pulses off.
S5	Same device as S1. Select use for initial pulse output from delay generator.	To screw #13 for counting (PRR measurement)
		open
		To com port for simulation.
6 BNCs	Inputs and outputs as labeled. Solder lugs, separately purchased, facilitate the ground connection.	
Screws #11, 13	Shown for wiring. They are on the screw panel.	
TP1, TP2	Test Points, long screws such as 4-40×50 mm, with bits of solid wire soldered on to accept oscilloscope probe. As many as 6 nuts per screw permit attachment to panel and solder lugs front and rear.	
Binding post	Green banana binding post. May also have bits of wire attached. Ground for oscilloscope probe, for instance. Bridge all BNC shells and wire them to this post and to some of the ribbon cable grounds.	