Final Report
Iowa Highway Research Board
Project HR-269

DATA ACQUISITION
AND
COMPUTER PLOTTING
OF
DELAMTECT DATA

D & D Digital Systems, Inc.

September 1985
In cooperation with the
Highway Division

Iowa Department of Transportation
FINAL REPORT

PROJECT: HR - 269

DATA ACQUISITION AND COMPUTER PLOTTING OF DELAMTEC DATA

SUBMITTED TO: IOWA DEPARTMENT OF TRANSPORTATION

SUBMITTED BY: D & D DIGITAL SYSTEMS INC
111 LYNN
AMES, IOWA 50010

DATE: JULY 1, 1985
1.0 SYSTEM OVERVIEW

The overall system is designed to permit automatic collection of delamination field data for bridge decks. In addition to measuring and recording the data in the field, the system provides for transferring the recorded data to a personal computer for processing and plotting. This permits rapid turnaround from data collection to a finished plot of the results in a fraction of the time previously required for manual analysis of the analog data captured on a strip chart recorder.

1.1 DELAMTECT

In normal operation the Delamtect provides an analog voltage for each of two channels which is proportional to the extent of any delamination. These voltages are recorded on a strip chart for later visual analysis. An event marker voltage, produced by a momentary push button on the handle, is also provided by the Delamtect and recorded on a third channel of the analog recorder.

1.2 D & D DATA ACQUISITION SYSTEM (DAS)

A distance measuring wheel was added to provide digital pulses indicating distance traveled.

A microprocessor based digital computer was designed to sample both analog signals from the Delamtect, convert them to digital numbers, and transmit these numbers to a cassette recorder. The Delamtect event marker switch and distance measuring wheel are monitored by this computer to control when to begin and stop the process of sampling the analog signals and how often samples should be taken.

1.3 TECHTRAN RECORDER

A digital cassette recorder was added to the system to record the digital numbers which are equivalent to the analog voltages from the two Delamtect channels. Digital recorders have the ability to directly receive digital data and record at high densities. The recorder has been mounted such that it can be removed from the Delamtect and transported to a computer site. There the digitally recorded data can be read and transferred from the recorder to the computer for further processing and plotting.
1.4 SPERRY/ZENITH/IBM PC PLOTTING PROGRAMS

Two programs have been developed to be run on the SPERRY/ZENITH/IBM PC.

The first program is designed to read data files recorded on the Techtran. Each pass across the bridge constitutes a file of data. As each data file is read, it is checked for correct sequence and length characteristics before being written to a disk file on the PC.

The second program is designed to process the data files and produce a plot of the results on a dot matrix printer operating in graphics mode which visually shows where delaminations exist in the area surveyed. Selected choices for type of plot and printer are provided. The voltage level to be used as a threshold for determining the presence of a delamination is also provided as a choice.

Communication between the Techtran and the PC is through the asynchronous COM1 port. The printer is attached through the Centronics compatible PRN port.

1.5 TEXAS INSTRUMENTS 855 PRINTER

The TI 855 dot matrix printer was specified to be the primary print device. Choices for the kind of plot desired include one which fills the width of the paper and one which presents uniform horizontal and vertical scales.

Header information describing the bridge ID, date, total area, delaminated area, etc. is also printed on each plot.
2.0 THEORY OF OPERATION

The following sections describe some of the technical aspects of the design by functional component. Technical design details can be found from the circuit schematics and program listings in the appendices of this report.

2.1 DELAMTECT

Each channel of the Delamtect provides an analog signal with a voltage ranging from 0 to approximately 5 volts. In areas of no delamination, the voltage is approximately 0.1 to 0.3 volt. Major delamination areas have been judged to occur where changes in voltage exceed 400 millivolts which correspond to 4 mm deflection on the chart paper. Where extreme delamination occurs, voltage changes or fluctuations in excess of 2 volts occur.

The sensitivity of the strip chart recorder is typically set for 1 volt/cm or 100 millivolt/mm.

The event marker switch creates a signal of 0 or approximately 2.0 volts when it is open or closed.

Power supply voltages of +12 and -12 volts DC are supplied by the Delamtect.

No modifications have been made to the original Delamtect circuits.

Only wiring taps to the power supply, channel voltages and event marker switch have been added to route these signals to the D & D DAS for processing. These taps are wired to a new connector added to the Delamtect but do not alter the original signals. An external wiring harness extends these signals to the DAS mounted in the lid of the Techtran recorder.

2.2 D & D DATA ACQUISITION SYSTEM (DAS)

A distance measuring wheel was added to provide 10 pulses per inch of travel. These become 5 volt pulse inputs to the DAS. This was chosen to avoid problems with integrating a DC voltage from the tachometer on the Techtran to compute distance traveled.

Two eight-bit analog-to-digital converters (one for each channel) are used with a 5 volt maximum input voltage which provides for a 20 millivolt per bit sensitivity. This sensitivity corresponds to a 0.2 mm deflection on the strip chart recorder.
In 3 inches of travel, 30 distance pulses are received. The analog voltages from the Delamteect are sampled and digitized after 6 pulses (0.6 inch) are received. This is repeated five times during every 3 inches of travel. The average of the first 4 samples is computed and transmitted to the Techtran recorder after the fifth sample period. The fifth sample point is ignored. Thus, each 3 inches of travel results in a value being sent to the recorder for each channel (two bytes; left channel followed by right channel). These values represent the computed average of 4 sample values during 3 inch interval.

An offset of 32 (base 10) is added to each digitized value. The maximum value after offset adjustment is limited to 127 (base 10). This insures no spurious CTRL characters (0 to 31) are sent to the recorder which might alter its operational status. Additionally, characters are transmitted over a serial RS-232C interface in 7 bit even parity codes which limits the maximum value to 127. This corresponds to 1.9 volts which is well in excess of any reasonable delamination threshold level.

To insure the tape is properly positioned with respect to tape leader in the Techtran recorder, a sequence of four one-second read commands followed by a rewind command is initiated on powerup. The green light on the top of the recorder case momentarily turns on and then off at the beginning of the powerup sequence.

The green light on the top of the recorder case should be on when a pass begins and off at the end of the pass. This is accomplished by the operator confidently pushing the marker event switch once at the beginning and end of each pass to toggle the light from off to on or from on to off.

Data files for each pass are transmitted to the Techtran recorder which begin with a 'Pass number sequence character' (A,B,C,D,E, etc.), followed by pairs of averaged data bytes (Left, Right), and ending with a file termination character CTRL-S.

A custom program stored in an EPROM on the DAS controls all this activity among the Delamteect, distance wheel, and recorder.
2.3 TECHTRAN RECORDER

The 9600PRL recorder has several switch settings which should remain unchanged throughout the operations.

**LINE MODE SWITCH = OFF (DOWN POSITION)**
**BINARY MODE SWITCH = ONLINE (CENTER POSITION)**

**DIP SWITCHES**

```
1: + 9600 BAUD
2: + 9600 BAUD
3: - DISABLE BS
4: - HALF DUPLEX
5: - DISABLE DELAY
6: - LF LINEMODE
7: - BIN CTRL OFF
8: - DELAY ON LF
9: - EVEN PARITY
10: + ENABLE PARITY
```

Cassette insertion and removal is done by pressing the POWER pushbutton (it will illuminate), then manually lifting the door latch allowing the door to swing open, and inserting or removing the cassette. The magnetic tape side should face down into the recorder, the cassette label should be visible through the window, and the large tape spool should be on the left if it is rewound.

New tapes should be rewound forward and backward to remove any static binding which may initially exist.

Used tapes should be demagnetized with the bulk eraser before being removed. Approximately 5 - 10 seconds is long enough to remove all previously recorded data.

The internal battery pack should be charged during the night preceding any field tests. Techtran verbally estimates a minimum of 5-6 hour operation time between charges.

Each digital cassette tape is rated to hold up to 220,000 characters or bytes which is well within the maximum limit of 64,000 characters or bytes permitted by the applications program for the SPERRY/ZENITH/IBM PC.

During data collection on a bridge, the TERMINAL PORT is used to connect the Techtran to the DAS. The MODEM/CPU PORT should be disconnected during this operation.

After data has been collected, the MODEM/CPU PORT is used to connect the Techtran to the SPERRY/ZENITH/IBM PC COM1 port.
The TERMINAL PORT should be disconnected during this operation.

2.4 SPERRY/ZENITH/IBM PC PROGRAMS

Two programs have been written to process the data collected on the Techtran recorder called TAPEREAD and BRIDGE.

TAPEREAD reads the data from cassette tape, does some verification of the data, and writes it to a disk file for subsequent processing.

BRIDGE reads the data from the disk file and plots it on a dot matrix printer according to user selected options.

The maximum length on the bridge for a single pass is software constrained to 6000 feet.

A data space of 64K bytes (8 bits = 1 byte) is reserved in the PC for a bit map of the bridge deck where each bit represents a 3 inch distance traveled in a 9 inch wide path. Two channels provide for combined 18 inch wide passes.

Every foot of travel provides 4 data values (one for every 3 inches of travel) for each of 2 channels. Thus, a total of 8 data values exist for every foot of travel. Each of these values represents the condition of a 3 inch by 9 inch surface area which is either delaminated or not delaminated. Once yes or no decision about delamination has been determined, only a single bit is required to represent that result (1 = delaminated, 0 = not delaminated). Thus, 1 byte (8 bits) can be used to represent the status of 8 areas each 3x9 inches. This is equivalent to a total area of 12 x 18 inches or 1 foot of travel for the 18 inch wide Delamtect path.

Thus, the maximum surface area which may be represented in a 64K (64,000) bit map can be found as follows:

<table>
<thead>
<tr>
<th>PASSES</th>
<th>LENGTH</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>6000 ft</td>
<td>60,000 bytes</td>
</tr>
<tr>
<td>21</td>
<td>3000</td>
<td>63,000</td>
</tr>
<tr>
<td>42</td>
<td>1500</td>
<td>63,000</td>
</tr>
<tr>
<td>64</td>
<td>1000</td>
<td>64,000</td>
</tr>
<tr>
<td>128</td>
<td>500</td>
<td>64,000</td>
</tr>
<tr>
<td>etc.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

There is also a limitation of the maximum surface area which may be surveyed at one time based upon the disk capacity
associated with the SPERRY/ZENITH/IBM PC. Floppy disk drives have a maximum capacity of 320K or 360K bytes depending upon the version of DOS which is being used.

Two files are created by the TAPEREAD program. The first is a temporary file which contains a copy of the data read from the Techtran recorder. This temporary file is always written to the default system disk drive. The second file is the disk file which contains the data without the sequence numbers and end of file characters. In addition, the second file contains a header record with information about the bridge. A user prompt for the file name can also include a disk drive designation if one other than the present default drive is desired.

Since each pass contains 2 channels of data and pairs of data are recorded for every 3 inches of travel, 8 data bytes are recorded for every foot of travel on a pass. Thus, the maximum pass-foot distance which can be written to a 320K disk file is 40K and the maximum for a 360K disk file is 45K.

This would lower the size of a deck surface which can be handled for a 320K diskette to

<table>
<thead>
<tr>
<th>PASSES</th>
<th>LENGTH</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>4000 ft</td>
<td>40,000 bytes</td>
</tr>
<tr>
<td>20</td>
<td>2000</td>
<td>40,000</td>
</tr>
</tbody>
</table>

etc.

A PC with a winchester (hard) disk drive does not have this size constraint since files in excess of 100,000 bytes present no space problem provided the disk is not full with other information.
A 64 byte header record is used in the PC data file for bridge identification and layout parameters.

Documented in the source code listing, this 64 character string variable contains:

<table>
<thead>
<tr>
<th>Information</th>
<th>Bytes In File</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bridge Id</td>
<td>0</td>
<td>32</td>
</tr>
<tr>
<td># Passes</td>
<td>34</td>
<td>- 35</td>
</tr>
<tr>
<td>Length in samples</td>
<td>36</td>
<td>Binary, high byte 1st</td>
</tr>
<tr>
<td>Normal or Slewed</td>
<td>37</td>
<td>N,n,S,s</td>
</tr>
<tr>
<td>Distance in inches (R)</td>
<td>38</td>
<td>Binary, high byte 1st</td>
</tr>
<tr>
<td>Distance in inches (L)</td>
<td>39</td>
<td>Binary, high byte 1st</td>
</tr>
<tr>
<td>Start Right/Left</td>
<td>40</td>
<td>R,r,L,l</td>
</tr>
<tr>
<td>Date Info</td>
<td>42</td>
<td>- 54</td>
</tr>
<tr>
<td>Extra space</td>
<td>55</td>
<td>- 63</td>
</tr>
<tr>
<td>Bridge data</td>
<td>64</td>
<td>- ??</td>
</tr>
</tbody>
</table>

Bridge data in the PC files do not alternate between left and right channels every byte. Instead, all left channel data for one pass is written to the disk, then all right channel data for the same pass is written. This pattern then repeats for the number of valid passes associated with the bridge deck.

Disk filenames follow standard MS DOS conventions and can have up to 8 characters with an optional 3 character extension, such as 'BR125437.DAT'. Disk drive information in addition to this limit is allowed, i.e. B:MAXWELL.DAT.

Bridge names can be up to 32 characters long. Dates can be up to 10 characters long.

It is important that the length of the first pass be accurate since it is used as the reference for all following passes. Passes which are shorter in length have additional low voltage data added equally to both ends of the file to fill it out to a standard length. Passes which are longer in length have data truncated equally from both ends until the standard length is reached.

If the first pass is short (less than 10 feet) the program will display a prompt on the PC screen to ask if it is to really be used. If the answer is n (no), it will read the second pass and treat it as a new first pass.

If a subsequent pass is short (less than 90% of the first pass), a displayed prompt will request if it is to be used or ignored. This could occur in the middle of a field test.
if the event marker switch was inadvertently pushed at the wrong time.

If a record is out of sequence (i.e. record 5 follows record 3), an option is displayed to permit a choice of using it, skipping it, or inserting a blank (no delaminations) record of data before this record. This should not occur, but gremlins occasionally show up in any system, and this provides a degree of partial recovery of the data.

The goal of the TAPEREAD program features described above is to create a disk file of sequential records from cassette files. This disk file will have identical data lengths for each pass with appropriate master header information about the name, date of test, size, and type of bridge being tested.

The BRIDGE program processes data from the PC data file previously created by TAPEREAD.

While reading in a disk data file but prior to plotting, an average baseline value for each side of each pass is computed by finding the most frequent value. Delaminations are then determined by deviations above this baseline value by some selected amount. A default of 400 millivolts (20 decimal above the baseline value) may be altered by a user prompt. This information is then placed into a memory bit map which is referred to by subsequent plotting programs.

Two delamination plots can be selected.

1: 8 Inches Wide
2: 4 Dots Per 3 Inches

An 8 inches wide plot utilizes the full width of the paper. Horizontal scaling varies with the number of passes.

A 4 dots per 3 inches of deck surface plot achieves uniform horizontal and vertical scaling within the physical limits of the printer.

One percentage plot can also be selected.

P: Percentages

A percentage plot separates the deck into 4 ft by 3 ft grids and prints the percentage of delaminated area in each grid box in addition to an overall percentage of delaminated area for the entire deck.
Bridge decks wider than 30 feet (20 passes) will be plotted in a compressed mode when using the percentage plot (P). The maximum width in this mode is 60 feet.

Bridge decks wider than 33 (22 passes) feet will be plotted in a compressed mode for graphical delamination plots (1, 2). The maximum width in this mode is 66 feet.

Bridge decks can be rectangular or skewed. If skewed, the distance in inches from the left and from the right rectangular starting line to the actual deck is requested by the TAPERED program.

Each plot prints information about the bridge name, date of field test, delamination threshold, length and width of the bridge deck, total area, and the percentage of the total area where delaminations exist. If the bridge is skewed, the left and right skew distances are also printed. Calculations of total area and percentage of delaminated area do not include the skewed area outside of the bridge deck.

Passes for a bridge can start on the left or on the right. Once started, subsequent passes are always in the opposite direction to the previous pass in an up and back manner. The bottom of the printed plot will correspond to the starting end of the bridge survey.

2.5 TEXAS INSTRUMENTS 855 PRINTER

Printers generally have more problems with compatibility than computers. While the printers may indeed have comparable capabilities, programming them into the same operating mode is often done in ways unique to each printer.

There is a big difference between functional compatibility and hardware compatibility. Therefore, each new printer should be approached with doubt as to having it function in anything but a standard manner without hardware or software modifications.

A choice of 4 printer types is given in the BRIDGE program.

T : TI 855 Printer
S : Star Radix Printer
E : Epson Printer
A : Alphanumeric Printer

The Alphanumeric Printer assumes no graphics dot matrix capability is present, so it will only print header summary information typically found at the beginning of each
delamination plot. In addition, it will also plot the Percentages (P) plot since it prints only standard alphanumeric characters.

The TI 855 printer is placed into DP mode under program control and left there when done. If this is not the normal mode, it can easily be restored by turning the power switch of the printer off and then back on.

TI and the Star RADIX printers have a 72 dots/inch plotting density. Epson MX/FX printers have a 60 dots/inch plotting density. Even more of a problem is that each type of printer typically has a different code sequence to put it into a so-called 'compatible' mode.

The uniform vertical and horizontal scale plot at 4 dots per 3 inches is achieved by occasionally skipping a dot in the vertical direction. Perfect scaling is not feasible. However, when reduced to a 10 scale size (1 in = 10 ft), the perturbation should not be observable.
3.0 OPERATING INSTRUCTIONS

3.1 DATA ACQUISITION PROCEDURES

1. TURN ON THE DELAMTECT FOR WARMUP AND RUN NORMAL CALIBRATION PROCEDURES.

2. PRESS POWER SWITCH OF RECORDER ON (SWITCH LIGHT WILL TURN ON). SET RECORDER LINE MODE SWITCH OFF (DOWN) AND BINARY SWITCH ONLINE (CENTER). DO NOT CLOSE THE LID YET.

3. INSERT TAPE IN RECORDER (TAPE EDGE IN FIRST & LABEL SHOWING IN WINDOW). PLUG IN RIBBON CABLE FROM RECORDER LID TO TERMINAL CONNECTOR ON RECORDER.

4. LOWER THE DISTANCE WHEEL AND INSERT HOLDING PIN IN OTHER HOLE.

5. TURN POWER SWITCH OF DELAMTECT OFF & ON.

(***DO NOT TURN POWER SWITCH OFF AGAIN UNTIL SURVEY IS COMPLETE**).

THE TRANSMITTER AND OPERATE/CALIBRATE SWITCH CAN BE CYCLED ON/OFF AT WILL.

(*** THE GREEN LIGHT ON THE RECORDER LID WILL CYCLE ON AND OFF ONCE. THE TAPE RECORDER READ SWITCH SHOULD THEN AUTOMATICALLY CYCLE ON & OFF 4 TIMES FOLLOWED BY AN AUTOMATIC REWIND CYCLE TO CORRECTLY POSITION THE TAPE LEADER. THE REWIND, READ, AND WRITE SWITCH LIGHTS SHOULD BE OFF AND THE POWER SWITCH LIGHT SHOULD BE ON AT THIS TIME. ***)

REPEAT THIS STEP UNTIL THE CORRECT RECORDER STATUS IS OBSERVED.

6. CLOSE THE RECORDER LID AND MOVE INTO POSITION ON BRIDGE IF NOT ALREADY THERE. IF IT IS A SKEWED BRIDGE, NOTE THE DISTANCE FROM BASE LINE TO BRIDGE DECK IN INCHES OF THE RIGHT AND LEFT SIDES.

7. TURN ON DELAMTECT TRANSMITTER (THIS MAY BE TURNED ON AND OFF AS NEEDED DURING THE SURVEY).

8. PUSH EVENT SWITCH ONLY ONCE TO START A PASS (GREEN LIGHT ON RECORDER SHOULD MUST BE ON TO RECORD!!) IF INADVERTENTLY PUSHED TWICE ON FIRST PASS, GO BACK TO STEP 5 AND START OVER. IF ADAVENTENTLY PUSHED TWICE ON SUBSEQUENT PASSES, PUSH AGAIN UNTIL LIGHT COMES ON AND
PROCEED BUT NOTE THIS OCCURRENCE SO THAT THE SHORT RECORD
OF DATA CAN BE TOSSSED OUT BY THE COMPUTER OPERATOR WHEN
LATER READ BY THE COMPUTER FOR PROCESSING.

9. AT END OF EACH PASS, PUSH EVENT SWITCH ONCE TO TURN GREEN
RECODER LIGHT OFF AND STOP DATA COLLECTION FOR THAT
PASS.

10. TURN THE DELAMTECT AROUND (PRESS DOWN ON THE HANDLE TO
ALLOW THE DISTANCE MEASURING WHEEL TO RAISE FROM THE
SURFACE AND AVOID DAMAGE BY TWISTING).

11. PUSH EVENT BUTTON TO START THE NEXT PASS (THE GREEN
RECODER LIGHT SHOULD TURN ON AGAIN, ETC).

12. WHEN DONE WITH LAST PASS, TURN OFF THE DELAMTECT
TRANSMITTER, OPEN THE RECORDER LID, PUSH THE REWIND
PUSHBUTTON SWITCH OF THE RECORDER TO REWIND TAPE. REMOVE
THE TAPE, TURN THE TAPE RECORDER POWER OFF, AND THEN TURN
DELAMTECT POWER OFF. RAISE DISTANCE WHEEL AND LOCK UP
WITH THE HOLDING PIN.
3.2 COMPUTER PLOTTING PROCEDURES

1. A SYSTEM DISK SHOULD BE FORMATTED ON THE TARGET COMPUTER AND A COPY OF TAPEREAD.EXE AND BRIDGE.EXE COPIED TO IT. THIS BECOMES A MASTER PROGRAM DISK. TO CALL UP EITHER PROGRAM, SIMPLY TYPE 'TAPEREAD' OR 'BRIDGE'.

   AT LEAST 2 FORMATTED DATA DISKS SHOULD BE AVAILABLE TO USE WITH THE TAPEREAD PROGRAM. FOR SMALL BRIDGES, BOTH THE TEMPORARY FILE AND FINAL DATA FILE CAN PROBABLY FIT ON ONE DATA DISKETTE IN DRIVE B:. FOR LARGER BRIDGES, A DATA DISK MIGHT BE NECESSARY IN BOTH FLOPPY DRIVES (A: & B:). AFTER THE TAPEREAD PROGRAM IS LOADED, THE EXTRA DATA DISK CAN BE PLACED IN THE SYSTEM DEFAULT DRIVE A:.

2. AN RS-232 CABLE IS REQUIRED TO CONNECT THE TECHTRAN RECORDER TO THE COM1 PORT ON THE SPERRY/azenith/IBM PC. THE SPERRY COM1 PORT IS FOUND ON THE BACK NEAR THE CENTER BOTTOM IN A HORIZONTAL POSITION.


4. PROGRAM 'TAPEREAD' IS USED FIRST TO TRANSFER THE DATA FROM THE RECORDER TO THE COMPUTER. PROMPTS ARE DISPLAYED AND RESPONSED NEED TO BE TYPED FOR

   BRIDGE ID:
   DATE:
   START ON RIGHT OR LEFT SIDE
   NORMAL OR SKEWED:
   IF SKEWED, RIGHT AND LEFT SKEW DISTANCE:

   INFORMATION ABOUT THE PASS NUMBER AND FILE LENGTHS ARE DISPLAYED HERE. IF SHORT PASSES OR SEQUENCE PROBLEMS EXIST, PROMPTS FOR DESIRED OPERATOR ACTIONS ARE REQUESTED HERE.

   DISK FILE NAME:

   THE DISK FILE NAME CAN ALSO BE PRECEDEED WITH THE DISK DRIVE ID IF OTHER THAN THE DEFAULT, IE. B:BR123456.DAT

   IF NO DRIVE IS SPECIFIED, FILES WILL BE WRITTEN TO THE DEFAULT DRIVE.

5. THE 'BRIDGE' PROGRAM CAN NOW BE USED TO CALCULATE A BIT MAP AND PLOT THE RESULTS ON THE ATTACHED TI 855 PRINTER.
A prompt for the disk file name with the survey data is issued. The bridge name is displayed on the screen, and a prompt is issued for the delamination voltage with 400 MV being the default value which is used if the return key is simply pushed rather than a number.

At this time the data is read in, averages computed, and delamination decisions are written to a memory bit map. The longer the data file, the longer this takes. The disk drive may be observed to determine if records are still being read in and processed prior to commencement of plotting.

Plotting choices of printer type are displayed with the current choice highlighted on the screen.

Three types of plots may now be selected by typing

1: 8 inch wide graphics plot
2: 4 dots per 3 inch graphics plot with equal scales
or P: Percentages in each 4' x 3' grid box

The same data can be plotted again with different choices for delamination threshold voltage.
A.0 APPENDICES : TECHNICAL REFERENCE MATERIALS

A.1 DELAMTECT SCHEMATIC AND MODIFICATIONS
A.2 D & D DATA ACQUISITION SYSTEM
MODEL LSC MEASURING WHEEL LENGTH SENSOR

AN ECONOMICAL ANSWER TO HIGH SPEED, UNI-DIRECTIONAL LENGTH MEASUREMENT FOR PAPER, TEXTILES, FILM, FOIL, FOAM & METAL STRIP PRODUCTS RUNNING THROUGH REWINDERS, PRINTING PRESSES, SLITTERS, SPOOLERS, SHEETERS & OTHER WEB, STRIP OR RIBBON HANDLING APPLICATIONS.

- CURRENT SINK OUTPUT
- AVAILABLE WITH:
  ONE OR TWO WHEELS
  ROUND, FLAT & KVNRLED TREAD

- FOR MEASURING
  INCHES, FEET, YARDS
  CENTIMETERS, METERS
  PT/ MIN, YDS/ MIN, MTRS/ MIN

The direction of rotation provided the direction does not change. In applications where reversing occurs, the up/down counter direction must be controlled by an external switch contact to correspond to the direction of motion. (As an alternative an RPU with quadrature output can be outfitted as a length sensor, see following page)

Unlike mechanical measuring wheels the LSC has no internal cam switches or dividing gears to wear or create drag. This means lighter wheel pressures since the only resistance the wheel encounters is due to a low bearing friction. Low wheel pressures in turn, means less wear on the wheel tread and less danger of marking the product.

Other Specifications Include:

- OUTPUT: RPN Open-Collector, current-limited at 4oma 28 VDC maximum
- MAX SHAFT SPEED: 3600 RPM (See wheel information, following page for wheel speed restrictions)

CONSTRUCTION: Cast Aluminum housing, Stainless Steel Tube extension. Oil impregnated sintered bronze bearings, lifetime lubricated. Shielded, 3-wire signal cable, 10' long. Operating Temperature range 0° to 60°C, weight 1.8 lbs.

TO ORDER: ASSEMBLE COMPLETE PART NO. AS FOLLOWS:

WHEEL CODES
(See description of wheels & codes next page)

- GO - Less Wheels
- GR - Round Neoprene
- GK - Flat Tread Polyurethane
- GK - Flat Tread Knurled

DESIGNED UNITS
OF MEASURE

| INCHES | 1 INCH | 1 |
| YARDS | 1 YARD | 1 |
| FEET | 1 FEET | 1 |
| METERS | 1 METER | 1 |

OUTPUT PULSE RATE (Pulses/Unit)

| SCROLL | 1/4 INCH | 1 |
| SCROLL | 1/2 INCH | 2 |
| SCROLL | 3/4 INCH | 3 |
| SCROLL | 1 INCH | 4 |

INSERT 2-DIGIT CODE

<table>
<thead>
<tr>
<th>INPUT</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>2</td>
</tr>
<tr>
<td>02</td>
<td>3</td>
</tr>
<tr>
<td>03</td>
<td>4</td>
</tr>
</tbody>
</table>

* Check counter or indicator using this output for compatibility. Cannot be frequency doubled externally.
** Output from 10, YO & M0 units code are 50usec pulses (See waveforms at left). Check compatibility of instrument, control, or counter for use with this waveform. Cannot be frequency doubled externally.
LENGTH SENSOR CONVERSION BRACKET (P/N LSCB-1000)

ADAPTS RPGB ROTARY PULSE GENERATOR TO LENGTH MEASUREMENT

- FOR BI-DIRECTIONAL MOTION APPLICATIONS REQUIRING QUADRATURE
- FOR FINE RESOLUTION, HIGH-PULSE-RATE APPLICATIONS

This conversion bracket allows the customer to assemble a custom length sensor by purchasing the following items separately.

1. Length Sensor Conversion Bracket (P/N LSCB-1000)
2. RPGB with appropriate PPR and Single Channel or Quadrature Output (See RPGB data sheet, Section D of the Catalog)
3. One or two measuring wheels (Listed below)
4. Hinge Clamp Assembly (Listed below)

The tubular arm length of this bracket, related to the wheel axis center-line of the RPGB is 6.8" similar to the LSC (see previous page). The 10 ft long, shielded cable (included with conversion bracket) has the same color coding as described for the RPGB cable (see page 71). Screws for mounting the conversion bracket to the RPGB are included. To order see table below.

SEPARATE LENGTH MEASURING WHEELS

<table>
<thead>
<tr>
<th>WHEEL CODE</th>
<th>WHEEL CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q8</td>
<td>QF</td>
</tr>
</tbody>
</table>

"Round Section Replaceable Tire .210" Section Dia. Black Neoprene

FOR USE ON: Metal, paper, foil, film and hard plastics.

Line contact on material being measured, convenient when available measuring track is narrow or for measuring end of roller beside passing material.

Max. Speed: 3000 RPM

SELECTING APPROPRIATE WHEEL SIZE & PPR (Pulses Per Rev.) OF ROTARY PULSE GENERATOR

When the desired output of an RPGB and wheel combination is either in feet or inch units, selection of the proper combination is relative straight forward. For example, with a 1-foot wheel circumference a 1 PPR Rotary Pulse Generator will deliver 1 pulse/ft. 12 PPR would deliver 12 pulses/ft. 100 PPR would deliver 100 pulses/ft. In general, 100 PPR would permit measuring to 1/10 of an inch (1/120th of a foot).

Measuring in yards or meters however is a bit more involved since a 1 yard or 1 meter circumference wheel would be prohibitively large. Instead, 4/10 yard and 4/10 meter wheels can be used in conjunction with RPGB's in either of two ways. First, RPGB's with PPR's of 4, 10 and 100 can be used with a Model BDMD (Bi-Directional Motion Decoder, See Accessories Section of catalog).

ORDERING INFORMATION

WHEELS & REPLACEMENT TIRES FOR CODE OR WHEELS

<table>
<thead>
<tr>
<th>WHEEL CODE</th>
<th>CIRCUMFERENCE</th>
<th>PART NO.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q8</td>
<td>1 foot (1/3 yd)</td>
<td>WF-1000-0R/A</td>
</tr>
<tr>
<td></td>
<td>1/4ths yard</td>
<td>WM-0333-0F/A</td>
</tr>
<tr>
<td></td>
<td>1/2 meter</td>
<td>WM-0400-0F/A</td>
</tr>
<tr>
<td>QF</td>
<td>1 foot (1/3 yd)</td>
<td>WF-1000-0R/A</td>
</tr>
<tr>
<td></td>
<td>1/4ths yard</td>
<td>WM-0400-0F/A</td>
</tr>
<tr>
<td></td>
<td>1/2 meter</td>
<td>WM-0400-0F/A</td>
</tr>
</tbody>
</table>

ACCESSORIES

<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>PART NO.</th>
</tr>
</thead>
<tbody>
<tr>
<td>LENGTH SENSOR CONVERSION BRACKET FOR RPGB</td>
<td>LSCB-1000/A</td>
</tr>
<tr>
<td>HINGE CLAMP ASSEMBLY FOR MODEL LSC &amp; CONVERSION BRACKET</td>
<td>LSA-NC0-08/A</td>
</tr>
</tbody>
</table>

Willow Springs Circle, RD 5, York, Pa. 17402
(717) 767-6511 TWX: 510 657 4214 RLC YRK
### PARTS LIST

#### 6502 A/D BOARD

<table>
<thead>
<tr>
<th>COMPONENT NUMBER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>0.047 ufd. Capacitor</td>
</tr>
<tr>
<td>C2</td>
<td>470 pf. Mica Capacitor</td>
</tr>
<tr>
<td>C3,C4,C5,C6,C7,C8,C9,C10,C11</td>
<td>0.1 ufd. Ceramic Capacitor</td>
</tr>
<tr>
<td>C12</td>
<td>0.01 ufd. Ceramic Capacitor</td>
</tr>
<tr>
<td>C13</td>
<td>1 ufd. 50 v. Electrolytic Capacitor</td>
</tr>
<tr>
<td>C14,C15</td>
<td>150 pf. Mica Capacitor</td>
</tr>
<tr>
<td>J1</td>
<td>DB-25S Rt. Angle P.C. Mount Connector</td>
</tr>
<tr>
<td>P2</td>
<td>34 Pin P.C. Mount Header</td>
</tr>
<tr>
<td>P3</td>
<td>6 Pin Amp Mate-N-Lock w/pins</td>
</tr>
<tr>
<td>R1,R2</td>
<td>2.2K Ohm 1/4 Watt 5% Resistor</td>
</tr>
<tr>
<td>R3</td>
<td>22K Ohm 1/4 Watt 5% Resistor</td>
</tr>
<tr>
<td>R4,R5</td>
<td>1M Ohm 1/4 Watt 5% Resistor</td>
</tr>
<tr>
<td>R6,R7,R8,R9,R10,R11,R12,R13,R14,R15</td>
<td>10K Ohm 1/4 Watt 5% Resistor</td>
</tr>
<tr>
<td>R16,R17</td>
<td>1K Ohm 1/4 Watt 1% Resistor</td>
</tr>
<tr>
<td>U1</td>
<td>65C02A Microprocessor I.C.</td>
</tr>
<tr>
<td>U2</td>
<td>6116-15 Ram I.C.</td>
</tr>
<tr>
<td>U3</td>
<td>2716 Eprom I.C.</td>
</tr>
<tr>
<td>U4,U5</td>
<td>68C21A Peripheral Interface Adaptor</td>
</tr>
<tr>
<td>U6</td>
<td>6551A Communications Interface I.C.</td>
</tr>
<tr>
<td>U7</td>
<td>SN74LS138N TTL I.C.</td>
</tr>
<tr>
<td>U8</td>
<td>SN7404 TTL I.C.</td>
</tr>
<tr>
<td>U9</td>
<td>NE555P Timer I.C.</td>
</tr>
<tr>
<td>U10,U11</td>
<td>ADC0804LCN A/D Converter</td>
</tr>
<tr>
<td>U12</td>
<td>ua1488 RS-232c Driver I.C.</td>
</tr>
<tr>
<td>U13</td>
<td>ua1489 RS-232c Receiver I.C.</td>
</tr>
<tr>
<td>XTAL1</td>
<td>1.8432 Mhz. Crystal</td>
</tr>
</tbody>
</table>
**Interface Cable:**
- 2: 34 Contact Ribbon Socket Connectors
- 3 1/2": 28 AWG 34 Conductor Ribbon Cable

**Tape Recorder Cable:**
- 2: DB-25P Ribbon Connectors
- 2': 28 AWG 25 Conductor Ribbon Cable

**Hardware:**
- 4: 4-40 x 3/8" Hex Posts
- 8: 4-40 x 3/8" Machine Screws
- 1: 24 Pin I.C. Socket
- 1/2": 22 AWG Red Hookup Wire
- 1/2": 22 AWG Yellow Hookup Wire
- 1/2": 22 AWG Blue Hookup Wire
- 1/2": 22 AWG White Hookup Wire
### COMPONENT NUMBER OR QUANTITY

<table>
<thead>
<tr>
<th>COMPONENT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>22 ufd. 25 v. Tantalum Capacitor</td>
</tr>
<tr>
<td>C2</td>
<td>100 ufd. 25 v. Electrolytic Capacitor</td>
</tr>
<tr>
<td>J3, J4</td>
<td>6 Pin Amp Mate-N-Lock Connector w/sockets</td>
</tr>
<tr>
<td>K1</td>
<td>Reed Relay-Radio Shack 275-232</td>
</tr>
<tr>
<td>L1</td>
<td>12V. Lamp Holder-Radio Shack 272-324</td>
</tr>
<tr>
<td>P2</td>
<td>34 Pin P.C. Mount Header</td>
</tr>
<tr>
<td>P4</td>
<td>6 Pin Amp Mate-N-Lock Connector w/pins</td>
</tr>
<tr>
<td>P5</td>
<td>9 Pin Receptacle-Amp CPC w/pins</td>
</tr>
<tr>
<td>P6</td>
<td>4 Pin Receptacle-Amp CPC w/pins</td>
</tr>
<tr>
<td>R1, R4</td>
<td>1K Ohm 1/4 Watt 5% Resistor</td>
</tr>
<tr>
<td>R2</td>
<td>9.1K Ohm 1/4 Watt 5% Resistor</td>
</tr>
<tr>
<td>R3</td>
<td>820K Ohm 1/4 Watt 5% Resistor</td>
</tr>
<tr>
<td>R5</td>
<td>5 Ohm 5 Watt 1% Resistor</td>
</tr>
<tr>
<td>U1</td>
<td>ua78H05 Voltage Regulator</td>
</tr>
<tr>
<td>U2</td>
<td>LM311N Comparator I.C.</td>
</tr>
<tr>
<td>U3</td>
<td>SN7406N TTL I.C.</td>
</tr>
</tbody>
</table>

### Hardware:

<table>
<thead>
<tr>
<th>QUANTITY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>#1487 Lamp</td>
</tr>
<tr>
<td>2</td>
<td>4-40 Standoff Terminals</td>
</tr>
<tr>
<td>4</td>
<td>4-40 X 3/8&quot; Hex Posts</td>
</tr>
<tr>
<td>1</td>
<td>TO-3 Spacesaver Heat Sink HS102</td>
</tr>
<tr>
<td>20</td>
<td>4-40 X 3/8&quot; Machine Screws</td>
</tr>
<tr>
<td>10</td>
<td>4-40 Hex Nuts</td>
</tr>
<tr>
<td>10</td>
<td>4-40 Lockwashers</td>
</tr>
<tr>
<td>2'</td>
<td>22 AWG Black Hookup Wire</td>
</tr>
<tr>
<td>3'</td>
<td>22 AWG Red Hookup Wire</td>
</tr>
<tr>
<td>1/2'</td>
<td>22 AWG Orange Hookup Wire</td>
</tr>
<tr>
<td>1 1/2'</td>
<td>22 AWG Yellow Hookup Wire</td>
</tr>
<tr>
<td>2'</td>
<td>22 AWG Green Hookup Wire</td>
</tr>
<tr>
<td>1'</td>
<td>22 AWG Blue Hookup Wire</td>
</tr>
<tr>
<td>1'</td>
<td>22 AWG Gray Hookup Wire</td>
</tr>
<tr>
<td>2'</td>
<td>22 AWG White Hookup Wire</td>
</tr>
</tbody>
</table>
## DELAMTEC MODIFICATION

<table>
<thead>
<tr>
<th>COMPONENT NUMBER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>P5</td>
<td>9 Pin Receptacle-Amp CPC w/pins</td>
</tr>
</tbody>
</table>

**Hardware:**
- 4-40 X 3/8" Machine Screws
- 4-40 Lockwashers
- 4-40 Hex Nuts
- 2' Single Conductor Shielded Wire
- 1' 22 AWG Orange Hookup Wire
- 1' 22 AWG White Hookup Wire
- 1' 22 AWG Gray Hookup Wire
- 1' 22 AWG Red Hookup Wire
- 1' 22 AWG Black Hookup Wire
### DISTANCE SENSOR

<table>
<thead>
<tr>
<th>COMPONENT NUMBER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS1</td>
<td>Distance Measuring Transducer</td>
</tr>
<tr>
<td></td>
<td>Red Lion LSCD-10-0F Wheel Sensor</td>
</tr>
<tr>
<td></td>
<td>Red Lion LSA-HCO-00 Bracket</td>
</tr>
</tbody>
</table>

| J6               | 6 Pin Plug-Amp CPC w/sockets                                  |

### COMPUTER CABLE

<table>
<thead>
<tr>
<th>COMPONENT NUMBER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DB-25P D Connector-Solder Type</td>
</tr>
<tr>
<td>1</td>
<td>DB-25S D Connector-Solder Type</td>
</tr>
<tr>
<td>2</td>
<td>DB-25 D Connector Hoods</td>
</tr>
<tr>
<td>12'</td>
<td>22 AWG 7 Conductor Cable</td>
</tr>
</tbody>
</table>

### SIGNAL & POWER CABLE

<table>
<thead>
<tr>
<th>COMPONENT NUMBER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>6 Pin Plugs-Amp CPC w/sockets</td>
</tr>
<tr>
<td>2 1/2'</td>
<td>22 AWG 4 Conductor Cable</td>
</tr>
<tr>
<td>7 1/2'</td>
<td>Single Conductor Shielded Wire</td>
</tr>
<tr>
<td>2 1/2'</td>
<td>Spiral Cable Wrap</td>
</tr>
</tbody>
</table>
A.3 TECHTRAN RECORDER
**NOTE:** The Line Mode Switch affects the Read function as follows:

<table>
<thead>
<tr>
<th>LINE MODE SWITCH ACTION</th>
<th>CHARACTER PATTERN</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ Line by Line</td>
<td>N/A</td>
</tr>
<tr>
<td>READ File by File</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**Command Reference**

<table>
<thead>
<tr>
<th>Command</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMOTE</td>
<td>Binary Switch</td>
</tr>
<tr>
<td>CONTROL</td>
<td>Manual Command</td>
</tr>
<tr>
<td>POSITION</td>
<td>Any</td>
</tr>
<tr>
<td>BINARY</td>
<td>Any</td>
</tr>
<tr>
<td>MOVE</td>
<td>Any</td>
</tr>
<tr>
<td>READ</td>
<td>Any</td>
</tr>
<tr>
<td>WRITE</td>
<td>Any</td>
</tr>
</tbody>
</table>

**Table:**

<table>
<thead>
<tr>
<th>Command</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMOTE</td>
<td>Binary Switch</td>
</tr>
<tr>
<td>CONTROL</td>
<td>Manual Command</td>
</tr>
<tr>
<td>POSITION</td>
<td>Any</td>
</tr>
<tr>
<td>BINARY</td>
<td>Any</td>
</tr>
<tr>
<td>MOVE</td>
<td>Any</td>
</tr>
<tr>
<td>READ</td>
<td>Any</td>
</tr>
<tr>
<td>WRITE</td>
<td>Any</td>
</tr>
</tbody>
</table>
1 System Overview

Introduction: This manual is designed for the user of the 9600PRL. All of the required operations associated with the unit are explained including:

- installation procedures
- cassette usage and care
- preventive maintenance
- operating instructions

This section should be thoroughly read and understood before proceeding to any of the above operations. It contains valuable information about using the 9600PRL.

9600PRL Operation

The 9600PRL is a high-speed portable data recorder/program loader designed for storing, transmitting and receiving ASCII-coded data. Among the standard operating features are:

- **WRITE** - recording data at selected speeds
- **READ** - displaying the contents of a cassette
- **FILE SKIP** - providing rapid forward tape advance
- **CHARACTER DELETE** - eliminating undesired characters during data recording

The 9600PRL has the versatility for numerous applications including:

- loading programs into PBX systems and process control devices
- loading diagnostic routines into programmable systems and devices
- digital recording for datalogging systems
- remote testing and troubleshooting

Figures 1 and 2 illustrate typical system configurations of the 9600PRL.
The transport door latch and remove the cassette.

To remove a cassette, make sure that no functions are in progress. Lift the transport door latch and the large tab under the latch. Close the transport door.

2. Pull out the cassette into the open guide with the label facing up.

1. Turn the unit on. Proceed to the POWER ON position (the projection will illuminate). Lift the transport door latch, sliding the door to open.

CASSETTE INSPECTION AND REMOVAL

In excess of 10°F (4°C),
a clean cassette is indicated by the absence of fingerprints or contamination.

To remove a cassette, open the cassette unit. Do not use excessive force. Ensure that the cassette is properly seated. Do not overtighten. The 9600PL can easily be manipulated with a standard cassette under normal conditions. The cassette is only replaced on one side of the cassette. Cassettes are not interchangeable with other cassettes. Do not discard used cassettes.

CASSETTE DESCRIPTION

Figure 2: Remove 9600PL System

Figure 1: Local 9600PL System

In addition, two ports are provided for computer interface. The IN and OUT ports provide special operational functions.

The IN port and printer provide special operational functions.

In the MODE/PRINT or TERMINAL ports, the IN and OUT ports are set to match the interface connection. AC power connections can be used in place of the IN port. The IN port is used to connect the unit to the Host. The MODE/PRINT port is used to connect the unit to the Host. The MODE/PRINT port is used to connect the unit to the Host.
CONTROL CODES

Several 9600PR1. operations are facilitated by using control codes. All control codes in this manual are identified as follows:

(CTRL X)

CTRL represents the CONTROL key on the terminal and X represents a designated character. To use a control code, press the CTRL key and simultaneously press the designated character. All control codes are listed on the back cover.

PREVENTIVE MAINTENANCE

The READ/WRITE tape head of the recorder must be cleaned daily using a soft cotton swab dampened in isopropyl alcohol. Neglecting this care may result in abnormal tape wear, transport damage or operational errors.

MANUAL ORGANIZATION

This manual contains three additional sections. Section 2 explains how to install the 9600PR1 and provides unit specifications. Section 3 provides operating instructions for the 9600PR1, and Section 4 lists available options. Proceed to Section 2 for installation procedures.

2

Installation and Specifications

INTRODUCTION

The following section contains information about installing the 9600PR1. Once the unit is unpacked, check that the following standard equipment was included:

- 9600PR1
- one cassette
- male to female EIA RS-232C cable
- power cord
- 9600PR1. Installation and Operating Instructions

Installation consists of setting the dip switches, connecting cables and powering on. Before beginning installation, verify that the peripheral equipment in use meets the specifications listed subsequently.

CASSETTE SPECIFICATIONS

- Philips-type
- 300 feet (length)
- 1600 bits per inch (density)

Only Tecktron cassettes (P/N 4300001) or an approved equal should be used with 9600PRL. Use of other cassette types may cause equipment damage and could void the machine warranty.

TERMINAL PORT INTERFACE

A device connected to the TERMINAL port must have the following characteristics:

- Full or Half Duplex, asynchronous
- 8-level USASCII coded
- EIA RS-232C/CCITT V.24

If using a Model 33* or similar terminal having a current interface, the terminal must be prepared to operate in the Full Duplex 20 milliampere Neutral mode.

*Model 33 is a trademark of Teletype Corporation.
DIP SWITCH SETTINGS

The dip switches are housed beneath a lift-off cover as indicated in Figure 3. Figure 5 illustrates the dip switch panel; follow the instructions provided to set these switches. Use a pencil tip to manipulate the switches.

NOTE: Each switch sets the interface requirements for all devices attached to the unit. Data to be recorded must come to the MODEM/CPU port under the following two conditions:

* Pin 5 at the MODEM/CPU port is enabled
* Binary switch set to ONLINE or BINARY

Input must come from the TERMINAL port if pin 5 at the MODEM/CPU port is disabled or the Binary switch is OFFLINE.

Baud Rate

Determine the appropriate baud (transmission) rate of the attached devices and set switches 1 and 2 accordingly. Refer to the upper right of the dip switch panel to determine the correct switch positions (ON or OFF). Characters with eleven bits (two stop bits) will be transmitted at the 110 speed; ten bit characters (one stop bit) will be transmitted at all other speeds. If using the ONLINE SPEED CONTROL option, refer to Section 4 before setting these switches.

Transmit/Disable (BS) Character

Switch 3 affects the CHARACTER DELETE function (refer to Section 3). When removing unwanted characters from data during this function, you may set this switch to cause the following results at the TERMINAL and MODEM/CPU ports:

* If the switch is set to ON(+), then once a character is removed from data, a (BS) code is sent and the cursor of the attached terminal device will move back to the position of the removed character. The next character received will be placed in this position.
* If the switch is set to OFF(-), then once a character is removed from data, no (BS) code is sent and the cursor prints the next character received in the following position.

Full/Half Duplex

The effect of switch 4 varies according to the position of the Binary switch on the control panel (refer to Figures 6 and 7).

* When the Binary switch is OFFLINE or pin 5 at the MODEM/CPU port is disabled and the Binary switch ONLINE, data received at the TERMINAL port is affected as follows:

  **Full Duplex:** All data received is echoed back to the source. In other words, if the device sends a character, it will be processed by the 9600PRL and also sent back to the originating device as verification.
  **Half Duplex:** No data is echoed back to the source.
When the Binary switch is ONLINE and pin 5 at the MODEM port is enabled, no data is echoed back to the terminal. Any full duplex device connected to the TEKMIN port will receive the same data as a device connected to the TERMINAL port. The MODEM port is for echo or non-echo, any device connected to it will echo the data it receives back to the TEKMIN port. The MODEM port is also used to exchange data with other devices. Data sent to the MODEM port is echoed back to the terminal from the 9000RUL. When the binary switch is ONLINE and pin 2 at the MODEM port is disabled, no data is echoed back to the terminal. Any full duplex device connected to the TERMINAL port is enabled, any device connected to the MODEM port will receive the same data as a device connected to the TEKMIN port.
Line Terminator

When the Line Mode switch on the control panel is ON, either a Carriage Return (CR) or Line Feed (LF) character (as set by this switch) is recognized as the line terminator. Determine which character ends lines of data for your application and set switch 6 accordingly.

Remote READ Control

Switch 7 determines if remote control with the READ function is permitted when the Binary switch is in BINARY. Using remote control, the READ function can be controlled by sending control codes Q and S to the MODEM/CPU and TERMINAL ports. If remote control is desired, set this switch to the ON(+) position. If remote control is not required, set this switch to the OFF(-) position and only manual control panel commands will be recognized (READ, WRITE and REWIND pushbuttons).

READ Delay Character

Set switch 8 only if dip switch 5 is set to the ON(+) position. The READ delay selected by dip switch 5 (Enable/Disable READ Delay) is activated by either a (CR) or (LF) character (as set by this dip switch). Specify the character to cause the delay by setting this switch.

Odd/Even Parity

Set switch 9 only if odd or even parity is required for the attached devices. This switch identifies the odd or even parity requirements of the attached devices. Specify if the devices in use have odd or even parity by setting this switch accordingly.

Enable/Disable Parity

Switch 10 further specifies the parity requirements of the attached devices. If this switch is set to the ON(+) position, odd or even parity will be sent as specified by dip switch 5. If this switch is set to the OFF(-) position, data will be recorded as it was received (8-bit bytes) and sent as it was recorded.

CABLE CONNECTIONS

The 9600PRI can be connected to peripheral devices by either direct or remote means as follows:

1. Connect the male to female cable supplied from the MODEM/CPU port on the unit to one of the following devices:

   - CPU or Intelligent Device (direct connection)
   - Modem or Acoustic Coupler (remote connection)

   Refer to Figures 1 and 2. Pin 5 must be enabled by the attached device. Local cable distances should not exceed 50 feet, according to EIA RS-232C specification.

2. Connect a male to male EIA RS-232C cable (not supplied) from the TERMINAL port on the unit to the terminal, printer, or electronic device in use. Be sure to activate the Remote/Online mode on a terminal or printer.

3. Attach the power cord supplied from the power plug on the unit to a local AC power source. Select the appropriate local AC power requirements on the control panel before turning the unit on. Press the POWER pushbutton on the control panel to turn the unit on (pushbutton will Illuminate). Turn on peripheral devices.
Introduction

Operating Instructions

3
Before beginning to WRITE:

- adhere to the guidelines listed previously.
- be sure that the cassette in use is not write-protected.
The plastic tab on the top left of the cassette must be in place.

To record data:

1. Insert the cassette and close the transport door.

2. Enter (CTRL R) or press the WRITE pushbutton. The WRITE pushbutton will illuminate. Note: (CTRL R) cannot be used if the Binary switch is in BINARY.

3. Enter the information to be recorded.

4. Enter (CTRL S) to identify the end of a file.

To terminate the WRITE function, enter (CTRL T) or press the WRITE pushbutton. The WRITE function must be terminated to insure that all information is recorded on the cassette.

READ

The READ function permits viewing of the cassette contents. There are three ways to READ cassettes (notice the required switch settings for each method):

- READ cassette information line by line (Line Mode switch set to ON, Binary switch set to ONLINE or OFFLINE)
- READ cassette information file by file (Line Mode switch set to off, Binary switch set to ONLINE or OFFLINE)
- READ entire cassette without stops (Binary switch set to BINARY)

Before you begin to READ, adhere to the guidelines listed previously.

To READ a cassette:

1. Insert the cassette and close the transport door.

2. Enter (CTRL Q) or press the READ pushbutton. The READ pushbutton will illuminate. The information will be displayed as previously indicated by the switch settings.

To end the READ function, enter (CTRL S) or press the READ pushbutton. The READ function will automatically terminate if:

- the end of all data is reached
- a blank cassette is inserted
- the end of the cassette tape is reached
- the unit encounters a file terminator, (CTRL S), recorded on the cassette (unless the Binary switch is in BINARY)
- the unit encounters a line terminator, (CR) or (LF), recorded on the cassette (if the Line Mode switch is ON)

REWIND

You can rewind the cassette tape provided no other function is in progress. The tape rewinds completely and cannot be stopped once begun. The transport door should never be opened during REWIND.

To REWIND the tape, enter (CTRL Z) or press the REWIND pushbutton. The REWIND pushbutton will illuminate and the tape will fully rewind. Always rewind the tape before removal. If the tape does not respond to this command, follow these steps:

1. Enter (CTRL Q) or press the READ pushbutton. Allow the tape to advance for a few seconds.

2. Enter (CTRL S) or press the READ pushbutton.

3. Enter (CTRL Z) or press the REWIND pushbutton.

CHARACTER DELETE

This function removes unwanted characters during data recording. When the unit receives the CHARACTER DELETE command (CTRL X), the last character sent to the unit is erased. The (CTRL X) command may be used consecutively to remove up to the last 64 characters. This function is inoperative when the Binary switch is in BINARY.
ONLINE SPEED CONTROL

This option cannot be used in the Read or the Read function.

Stop Read: +3V
Short Read: +3V

The port will increment and resume the READ function.

Type in Motion: +1V (minimum 10ma)
Target Speed: +1V

The slowing voltage on pin 16 of other port.

READY BUSY OUTPUT

Optional Features

FILE SKIP

1. Insert the cassette and close the transport door.
2. Enter (CTRL) O, the READ pushbutton is depressed.
3. Continue the FILE SKIP function with Begin and the Read function is not interrupted when the cassette is near the display. Cassette data is displayed by the Read function.
4. In the FILE SKIP function, press the desired key to advance to the position where the next file is located.
5. Press the FILE SKIP function.

To stop a FILE SKIP program, enter (CTRL) S, or press the READ pushbutton.

In the normal operation, (CTRL) S is recorded on the cassette.

A blank cassette is inserted, the end of the cassette stops.

The FILE SKIP function does not need to be terminated once a stop command is issued. If you wish to begin another operation. This function will advance to the next stop.
LIMITED WARRANTY

Techtran Industries, Inc. warrants that this product is free from defects in material and workmanship for a period of ninety (90) days from the date of shipment. Techtran’s obligation under this limited warranty shall be to replace or repair, at its option, at its designated site, any part or parts thereof (except expendable parts thereof, defective cassettes and damage caused in shipment) that within the warranty period are returned to Techtran in the original shipping package under a Return Authorization (RA) number issued by Techtran and that are found by Techtran to be defective in proper usage. Techtran reserves the right to refuse to accept delivery of any shipment containing any shipping cartons which do not have the RA number displayed on the outside. Buyer shall ship freight prepaid to Techtran’s designated site. If Techtran determines that the product is not defective within the terms of this limited warranty, Buyer shall pay Techtran the cost of repairs at the then prevailing Techtran repair rates.

The foregoing limited warranty is in lieu of all warranties, either expressed or implied, including without limitation any implied warranty of merchantability or fitness for a particular purpose, and of any other obligation on the part of Techtran. Techtran shall not be liable for any injury, loss or damage, direct or consequential to persons or property caused either directly or indirectly by the use or inability to use the equipment and/or the related documents. Such limitations in liability shall remain in full force and effect even when Techtran may have been advised of the possibility of such injuries, losses or damages.

Any provision herein to the contrary notwithstanding, in no event shall Techtran be liable for indirect, incidental or consequential damages and in no event shall the liability to Techtran arising in connection with any product (whether such liability arises from a claim based on contract, warranty, tort or otherwise) exceed the actual amount paid to Techtran for the defective product.

Before purchasing or using the product, Buyer shall determine the suitability of Techtran products and related documents for his or her intended use and assumes all risk and liability whatsoever in connection therewith.

©Copyright 1982.
TECHTRAN INDUSTRIES, INC., 200 COMMERCE DRIVE, ROCHESTER, N.Y. 14623. Printed in U.S.A. All rights reserved. No part of this publication may be reproduced without written permission from Techtran Industries, Inc.
A.4 MONITOR PROGRAM
© D & D Digital Systems Inc., 1985

; DOTREH3.ASM 6-11-85
; DOT DATA COLLECTION PROGRAM
; ;
; ; PW 80 ; PAGE WIDTH 80 COLS
; ;
; ;******************************************************************************
; ; PAGE ZERO VARIABLES
; ;******************************************************************************
16
17 00 00 RAM EQU $0000 ; RAM AREA
18 01 00 PLSCNT EQU $01 ; PULSE COUNT STORAGE
19 02 00 DATCNT EQU $02 ; DATA BYTE COUNT STORAGE
20 03 00 AD1 EQU $03 ; AD1 DATA STORAGE
21 04 00 AD2 EQU $04 ; AD2 DATA STORAGE
22 05 00 OUT1H EQU $05 ; DATA1 UPPER BYTE
23 06 00 OUT1L EQU $06 ; DATA1 LOWER BYTE
24 07 00 OUT2H EQU $07 ; DATA2 UPPER BYTE
25 08 00 OUT2L EQU $08 ; DATA2 LOWER BYTE
26 09 00 PDATA EQU $09 ; PRINT DATA FLAG-80H=DATA TA1, 40H=OUT2
27 0A 00 COUNT1 EQU $0A ; DELAY COUNTER1
28 0B 00 COUNT2 EQU $0B ; DELAY COUNTER2
29 0C 00 PASS EQU $0C ; PASS NUMBER
30 0D 00 SWFLAG EQU $0D ; MARKER SWITCH FLAG
31 0E 00 SWCNT EQU $0E ; MARKER SW COUNT
32 0F 00 CHAN1 EQU $0F ; CHANNEL1 DATA
33 10 00 CHAN2 EQU $10 ; CHANNEL2 DATA
34 35
36
37
38
39
40
41
42 11 00 RDFILE EQU $11 ; CNTL-8, READ TAPE FILE
OPFILE EQU $12 ; CNTL-R, OPEN TAPE FILE
CLFILE EQU $14 ; CNTL-T, CLOSE TAPE FILE
REWND EQU $1A ; CNTL-Z, REWIND TAPE FILE
ENFILE EQU $13 ; CNTL-S, ENDD_READ TAPE FILE
ACK_RW EQU $07 ; CNTL-G, REWIND ACK
ACK_WR EQU $06 ; CNTL-F, WRITE ACK

---------------------------------------------------------------------------

PERIPHERAL CHIPS

---------------------------------------------------------------------------

ADDPRA EQU $2000 ; A TO D 6821-DIR REG A
ADCPRA EQU $2000 ; PER REG A
ADDCRA EQU $2001 ; CNT REG A
ADDRB EQU $2002 ; DIR REG B
ADPRB EQU $2002 ; PER REG B
ADCRB EQU $2003 ; CNT REG B

---------------------------------------------------------------------------

GPDRA EQU $4000 ; GEN P 6821-DIR REG A
GPPRA EQU $4000 ; PER REG A
GPCRA EQU $4001 ; CNT REG A
GPDDB EQU $4002 ; DIR REG B
GPPRB EQU $4002 ; PER REG B
GPCRB EQU $4003 ; CNT REG B

---------------------------------------------------------------------------

ACIADAT EQU $6000 ; SERIAL PORT-DATA REG
ACIASTA EQU $6001 ; STATUS REG
ACIMACMD EQU $6002 ; CMD REG
ACIACHTL EQU $6003 ; CTL REG
ICMD EQU $6B ; DTR=L,IRD=H,RT S=L, NO ECHO,EVEN PAR

ICTL EQU $3E ; 9600 BAUD, 7 BI T, 1 STOP

; .ORG F800H ; BEGIN PROGRAM ASSEMBLY

; INITIALIZATION

; INTCPU

LDX #$FF
90  F802  9A  TOP OF STACK
91  F803  78  ;DISABLE CPU INTERRUPTS
92  F804  D8  ;BINARY MODE
93  ;INITSER
94  F805  A9  6B  LDA  #ICMD ;INIT COM REG
95  F805  A9  6B  STA  ACIACMD
96  F807  8D  02  60  LDA  #ICTL ;INIT CTL REG
97  F80A  A9  3E  STA  ACIACTL
98  F80C  8D  03  60
99  F80F  ;INITAD
100 F80F  A9  04  LDA  #$04 ;SET TO PER REG
101 F811  8D  03  20  STA  ADCRB
102 F814  A9  0F  LDA  #$0F ;SET A/D CNTL LINE HIG
103 F816  8D  02  20  STA  ADPRB
104 F819  A9  00  LDA  #$00 ;SET TO DIR REGS
105 F81B  8D  01  20  STA  ADCRA
106 F81E  8D  03  20  STA  ADCRB
107 F821  8D  00  20  STA  ADDRA ;SET PAO-PA7 INPUTS
108 F824  A9  0F  LDA  #$0F ;PB0-PB3 OUTPUTS,PB4-PB7 INPUTS
109 F826  8D  02  20  STA  ADDRB
110 F829  A9  04  LDA  #$04 ;SET TO PER REG
111 F82B  8D  01  20  STA  ADRCRA
112 F82E  8D  03  20  STA  ADRCRB
113 F831  A9  04  LDA  #$04 ;CS1,CS2,WR=L
114 F833  8D  02  20  STA  ADPRB
115 F836  A9  0F  LDA  #$0F ;SET ADLINES HIGH
116 F838  8D  02  20  STA  ADPRB
117 F83B  A9  08  LDA  #$08 ;CS1,CS2,RD=L
118 F83D  8D  02  20  STA  ADPRB
119 F840  A9  0F  LDA  #$0F ;SET ADLINES HIGH
120 F842  8D  02  20  STA  ADPRB
121 F845  ;INITGP
122 F845  A9  00  LDA  #$00 ;SET TO DIR REG
123 F847  8D  01  40  STA  GPACRB
124 F84A  8D  03  40  STA  GPCRRA
125 F84D  8D  00  40  STA  GPDRA ;PAO-PA7 INPUTS
126 F850  A9  01  LDA  #$01 ;PB0 OUT, PB1-7 IN
127 F852  8D  02  40  STA  GPDRB
128 F855  A9  0D  LDA  #$0D
129 F857  8D  01  40  STA  GPACRA ;CA1,2 NEG TRANS, PER REG
130 F85A  8D  03  40  STA  GPCRRA ;CB1,2 NEG TRANS, PER REG
131 F85D  ;INITREAD
132 F85D  A0  04  LDY  #04 ;READ 4 TIMES
133 F85F  A9  11  TAPERD LDA  #RDFILE ;CNTL-Q START READ
134 F861  20  C7  F9  JSR  PRINT
135 F864  A2  05  LDX  #05 ;ONE SEC LOOP INIT
136 F866  20  B6  F9  ONESEC JSR  DELAY ;200 MS DELAY
137 F869  CA  DEX
138 F86A  D0  FA  BNE  ONESEC
139 F86C  A9  13  LDA  #ENFILE ;CNTL-S STOP READ
140 F86E  20  C7  F9  JSR  PRINT
141 F871  88  DEY
142 F872  D0  EB  BNE  TAPERD ;NEXT READ

-58-
INITRWD
LDA #REWND ; REWIND TAPE
JSR PRINT

INITLP1
JSR READTP ; GET CHAR
CMP #$0F
BEQ INITRWD
CMP #$ACK_RW ; ACK REWIND CNTL-G
BNE INITLP1 ; TRY NEXT CHAR IN BUFFER

INITPASS
LDA #$41 ; SET FIRST PASS TO "A"
STA PASS

INITVAR
LDA #$100 ; SWITCH COUNT
STA SWCNT
LDA #$00 ; SWITCH FLAG
STA SWFLAG

; ************
; MAIN PROGRAM
; ************
; ************
; START ; WAIT FOR START BUTTON
LDA GPPRB ; READ SWITCH PORT
AND #$02 ; MASK FOR SWITCH
BNE START ; SWITCH OPEN
JSR DEBOUN ; DEBOUNCE SWITCH
LDA SWFLAG
BEQ START ; FALSE SWITCH CLOSE
FILED
LDA #$0F ; OPEN TAPE FILE
JSR CNTL-R ; START WRITE

FILLP1
JSR READTP ; GET ACK
CMP #$0F
BEQ FILED
CMP #$ACK_WR ; ACK WRITE CNTL-F
BNE FILLP1 ; TRY NEXT CHAR IN BUFFER
LDA #$01 ; TURN ON COLLECT LIGHT
STA GPPRB
LDA PASS ; SEND PASS NUMBER TO TAPE
JSR PRINT ; CLEAR VARIABLES

CLEAR
LDA #$00
STA AD1
STA AD2
STA PDATA
STA CHAN1
195 F8C2 85 10       STA   CHAN2
196 F8C4 AD 00 40    LDA    GPPRA ;RESET CA1/CA2
197
198 F8C7 A9 00       LDA    #$00 ;CLEAR 4 SAMPLE SUM
199 F8C9 85 05       STA    OUT1H
200 F8CB 85 06       STA    OUT1L
201 F8CD 85 07       STA    OUT2H
202 F8CF 85 08       STA    OUT2L
203 F8D1
204 F8D1 A9 06       LDA    #$06 ;INIT PULSE COUNTER
205 F8D3 85 01       STA    PLSCNT
206 F8D5 A9 05       LDA    #$05 ;INIT DATA COUNTER
207 F8D7 85 02       STA    DATCNT
208
209
210 F8D9 AD 02 40    LDA    GPPRB ;MARKER SWITCH PORT
211 F8DC 29 02       AND    #$02 ;SWITCH MASK
212 F8DE DO 04       BNE    PULSE ;SWITCH STILL OPEN
213 F8E0 C6 0E       DEC    SWCNT ;SW CLOSURE COUNT
214 F8E2 F0 30       BEQ    CLOSE ;END OF PASS
215 F8E4 2C 01 40    PULSE  BIT   GPCA ;DISTANCE PULSE?
216 F8E7 50 F0       BVC    PCNTR ;NO PULSE RECEIVED
217 F8E9 AD 00 40    LDA    GPPRA ;RESET CA1/CA2
218 F8EC C6 01       DEC    PLSCNT
219 F8EE DO 11       BNE    PCHAR ;NOT YET
220 F8F0 A9 06       LDA    #$06 ;RESET PULSE COUNTER
221 F8F2 85 01       STA    PLSCNT
222 F8F4 20 D3 F9    JSR    READAD ;GET DATA
223 F8F7 C6 02       DEC    DATCNT
224 F8F9 F0 0C       BEQ    LASTDAT ;LAST DATA BYTE
225 F8FB 20 32 F9    JSR    TOTAL1 ;ADD DATA
226 F8FE 4C D9 F8    JMP    PCNTR ;GET MORE DATA
227
228 F901 20 6A F9    FCHAR  JSR    CHAR ;CHAR SEND ROUTINE
229 F904 4C D9 F8    JMP    PCNTR ;GET MORE DATA
230
231 F907 20 1E FA    LASTDAT JSR    DIV1 ;DIVIDE DATA BY 4
232 F90A 20 4D F9    JSR    ADJUST ;ADJUST DATA
233 F90D A9 C0       LDA    #$C0 ;SET PRINT DATA FLAG
234 F90F 85 09       STA    PDATA
235 F911 4C C7 F8    JMP    LOOP ;GET MORE DATA
236 F914
237 F914 20 94 F9    CLOSE JSR    DEBOUN ;DEBOUNCE SWITCH
238 F917 A9 00       LDA    #$00 ;TURN OFF COLLECT LIGHT
239 F919 8D 02 40    STA    GPPRB
240 F91C A9 13       LDA    #$ENFILE ;SEND END OF FILE TO TAPE
241 F91E 20 C7 F9    JSR    PRINT
242 F921 A9 14       LDA    #$CLFILE ;CLOSE TAPE FILE
243 F923 20 C7 F9    JSR    PRINT
244 F926 E6 0C       INC    PASS ;SET NEXT PASS NUMBER
245 F928 A9 64       LDA    #$100 ;INIT MARKER SW COUNTER
246 F92A 85 0E       STA    SWCNT
247 F92C AD 00 40    LDA    GPPRA ;RESET CA1/CA2
248 F92F 4C 90 F8    JMP    START
249

-60-
;******************************************************************************
; ; ADD DATA BYTE SUBROUTINE
;******************************************************************************
;
; TOTAL1
; LDA AD1 ; ADD DATA
; CLC
; ADC OUT1L ; LOW BYTE
; STA OUT1L
; LDA #$00 ; ADD CARRY TO HI BYTE
; ADC OUT1H ; HI BYTE
; STA OUT1H
;
; TOTAL2
; LDA AD2 ; ADD DATA BYTE 2
; CLC
; ADC OUT2L ; LOW BYTE
; STA OUT2L
; LDA #$00 ; ADD CARRY TO HI BYTE
; ADC OUT2H ; HI BYTE
; STA OUT2H
;
; RTS ; RETURN
;
;******************************************************************************
; ; ADJUST DATA
;******************************************************************************
;
; ADJUST
; LDA #32 ; ADD OFFSET
; CLC
; ADC OUT1L ; ADJUST BYTE 1
; STA CHAN1
; LDA #32 ; ADD OFFSET
; CLC
; ADC OUT2L ; ADJUST BYTE 2
; STA CHAN2
;
; OVER ; DATA OUT OF BOUNDS TEST
; LDA #127 ; TEST BYTE 1
; CMP CHAN1
; BCS NEXTB ; OKAY
; STA CHAN1 ; LIMIT TO $FF
;
; NEXTB
; CMP CHAN2 ; TEST BYTE 2
; BCS FIXD ; OKAY
; STA CHAN2 ; LIMIT TO $FF
;
; FIXD
; RTS ; RETURN
;
;******************************************************************************
**CHARACTER PRINT**

**CHAR**

```
301 302 303 304
305
306 307 308 309
310 311 312 313
314 315 316 317
318 319 320 321
322 323 324 325
326 327 328 329
330
331
332
333
334
335
336 337 338 339
340 341 342 343
344 345 346 347
348 349 350 351
352
```

```assembly
301 302 303 304
305
306 307 308 309
310 311 312 313
314 315 316 317
318 319 320 321
322 323 324 325
326 327 328 329
330
331
332
333
334
335
336 337 338 339
340 341 342 343
344 345 346 347
348 349 350 351
352
```

```assembly
CHAR:
BIT PDATA ; DATA
BPL CHAR2 ; NO
LDA ACIASTA ; ACIA READY?
AND #$10
BEQ CHARDN ; NOT READY
LDA CHAN1
STA ACIADAT ; SEND CHARACTER
LDA #$40 ; CLEAR 7, SET 6
STA PDATA
RTS ; RETURN
```

```assembly
CHAR2:
BIT PDATA ; MORE DATA
BVC CHARDN ; NO
LDA ACIASTA ; ACIA READY?
AND #$10
BEQ CHARDN ; NOT READY
LDA CHAN2
STA ACIADAT ; SEND CHARACTER
LDA #$00 ; RESET PRINT FLAG
STA PDATA
RTS ; RETURN
```

**CHARDN**

```
RTS ; RETURN
```

**DEBOUNCE SUBROUTINE**

```
DEBOUN:
JSR DELAY ; WAIT 200 MSEC.
LDA GPPRB ; SWITCH PORT
AND #$02 ; SWITCH MASK
BNE FALSE ; EARLY SW OPEN
JSR DELAY ; WAIT 200 MSEC
LDA GPPRB ; SWITCH PORT
AND #$02 ; SWITCH MASK
BEQ SHUT ; SWITCH IS STILL CLOSED
JSR DELAY ; OPEN, BUT WAIT 200 MSEC
LDA #$01
STA SWFLAG ; VALID SWITCH CLOSURE
FALSE BNE RETRN
LDA #$00
STA SWFLAG ; FALSE SWITCH CLOSURE
RETN RTS
```
**DELAY SUBROUTINE**

```
; DELAY
LDA #200 ; 200 MS DELAY
STA COUNT2

ONEMS
LDA #$FA ; 1 MILISEC COUNT
STA COUNT1

DOWN1
DEC COUNT1
BNE DOWN1
DEC COUNT2
BNE ONEMS
RTS
```

**PRINT SUBROUTINE**

```
; PRINT
TSTXMT
LDA ACIASTA ; ACIA READY
AND #$10
BEQ TSTXMT ; NOT READY
STX ACIADAT ; SEND CHARACTER
RTS ; RETURN
```

**READ A/D CONVERTERS**

```
; READAD
CONST
LDA #$0C ; CS1,CS2=L
STA ADPRB
LDA #$04 ; WR=L
STA ADPRB
LDA #$0F ; CNT LINES=H
STA ADPRB
DONCVY ; DONE CONVERSION
```
405 F9E2 AD 02 20
406 F9E5 29 30
407 F9E7 D0 F9
408 F9E9 READ1
409 F9E2 A9 0E
410 F9EB 8D 02 20
411 F9EE A9 0A
412 F9F0 8D 02 20
413 F9F3 AD 00 20
414 F9F6 C9 7F
415 F9F8 90 02
416 F9FA A9 7F
417 F9FC R_OK1
418 F9FC 85 03
419 F9FE A9 0F
420 FA00 8D 02 20
421 FA03 READ2
422 FA03 A9 0D
423 FA05 8D 02 20
424 FA08 A9 09
425 FA0A 8D 02 20
426 FA0D AD 00 20
427 FA10 C9 7F
428 FA12 90 02
429 FA14 A9 7F
430 FA16 R_OK2
431 FA16 85 04
432 FA18 A9 0F
433 FA1A 8D 02 20
434 FA1D 60
435
436
437
438
439
440
441
442 FA1E DIV1
443 FA1F 66 05
444 FA21 66 06
445 FA23 18
446 FA24 66 05
447 FA26 66 06
448 FA28 DIV2
449 FA28 18
450 FA29 66 07
451 FA2B 66 08
452 FA2D 18
453 FA2E 66 07
454 FA30 66 08
455 FA32 60
456
457
458
; READ UART
; WAIT 250 MS FOR RESPONSE
; RETURN FF IF NO RESPONSE
; RETURN CHAR IF RESPONSE

;**********************************************************************************
READTP

LDX #$0F ; 256 TRIES

READLP1

LDA #$01 ; WAIT ONE MS
JSR ONEMS
DEX
BEQ NO_READ ; GIVE UP
LDA ACI_STA ; GET STATUS
AND #$08 ; RDRF FLAG
BEQ READLP1 ; TRY AGAIN
LDA ACIADATA ; GET DATA
RTS

NO_READ

LDA #$0F
RTS

;**********************************************************************************

; VECTORS

;**********************************************************************************

.ORG FFFCH ; RESET VECTOR
.BYTE 00H ; LOW BYTE
.BYTE F8H ; HIGH BYTE

;**********************************************************************************

.END
<table>
<thead>
<tr>
<th>ACIACMD</th>
<th>6002 :</th>
<th>96</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACIACTL</td>
<td>6003 :</td>
<td>98</td>
</tr>
<tr>
<td>ACIADAT</td>
<td>6000 :</td>
<td>313</td>
</tr>
<tr>
<td>ACIACMD</td>
<td>6001 :</td>
<td>309</td>
</tr>
<tr>
<td>ACK_RW</td>
<td>0007 :</td>
<td>15</td>
</tr>
<tr>
<td>ACK_WR</td>
<td>0006 :</td>
<td>18</td>
</tr>
<tr>
<td>AD1</td>
<td>0003 :</td>
<td>19</td>
</tr>
<tr>
<td>AD2</td>
<td>0004 :</td>
<td>19</td>
</tr>
<tr>
<td>ADCRA</td>
<td>2001 :</td>
<td>105</td>
</tr>
<tr>
<td>ADCRB</td>
<td>2003 :</td>
<td>101</td>
</tr>
<tr>
<td>ADDRA</td>
<td>2000 :</td>
<td>10</td>
</tr>
<tr>
<td>ADDR6</td>
<td>2002 :</td>
<td>10</td>
</tr>
<tr>
<td>ADJUST</td>
<td>F94D :</td>
<td>23</td>
</tr>
<tr>
<td>ADPRA</td>
<td>2000 :</td>
<td>413</td>
</tr>
<tr>
<td>ADPRB</td>
<td>2002 :</td>
<td>103</td>
</tr>
<tr>
<td></td>
<td></td>
<td>403</td>
</tr>
<tr>
<td>CHAN1</td>
<td>000F :</td>
<td>194</td>
</tr>
<tr>
<td>CHAN2</td>
<td>0010 :</td>
<td>195</td>
</tr>
<tr>
<td>CHAR</td>
<td>F96A  :</td>
<td>22</td>
</tr>
<tr>
<td>CHAR2</td>
<td>F97F  :</td>
<td>30</td>
</tr>
<tr>
<td>CHARDN</td>
<td>F993  :</td>
<td>31</td>
</tr>
<tr>
<td>CLEAR</td>
<td>F8B8  :</td>
<td></td>
</tr>
<tr>
<td>CLFILE</td>
<td>0014  :</td>
<td>242</td>
</tr>
<tr>
<td>CLOSE</td>
<td>F914  :</td>
<td>21</td>
</tr>
<tr>
<td>CONST</td>
<td>F9D3  :</td>
<td></td>
</tr>
<tr>
<td>COUNT1</td>
<td>000A  :</td>
<td>366</td>
</tr>
<tr>
<td>COUNT2</td>
<td>000B  :</td>
<td>363</td>
</tr>
<tr>
<td>DATCNT</td>
<td>0002  :</td>
<td>207</td>
</tr>
<tr>
<td>DEBOUN</td>
<td>F994  :</td>
<td>173</td>
</tr>
<tr>
<td>DELAY</td>
<td>F9B6  :</td>
<td>136</td>
</tr>
<tr>
<td>DIV1</td>
<td>FA1E  :</td>
<td>231</td>
</tr>
<tr>
<td>DIV2</td>
<td>FA28  :</td>
<td></td>
</tr>
<tr>
<td>DONCV</td>
<td>F9E2  :</td>
<td>407</td>
</tr>
<tr>
<td>DOWN1</td>
<td>F9BE  :</td>
<td>369</td>
</tr>
<tr>
<td>ENFILE</td>
<td>0013  :</td>
<td>139</td>
</tr>
<tr>
<td>FALSE</td>
<td>F9B1  :</td>
<td>340</td>
</tr>
<tr>
<td>FILE0</td>
<td>F99E  :</td>
<td>18</td>
</tr>
<tr>
<td>FILLP1</td>
<td>F9A3  :</td>
<td>18</td>
</tr>
<tr>
<td>FIXD</td>
<td>F969  :</td>
<td>296</td>
</tr>
<tr>
<td>GPCRA</td>
<td>4001  :</td>
<td>123</td>
</tr>
<tr>
<td>GPCRRA</td>
<td>4003  :</td>
<td>124</td>
</tr>
<tr>
<td>GPDRA</td>
<td>4000  :</td>
<td>125</td>
</tr>
<tr>
<td>GDPDB</td>
<td>4002  :</td>
<td>127</td>
</tr>
<tr>
<td>GPPRA</td>
<td>4000  :</td>
<td>196</td>
</tr>
<tr>
<td>GPPRB</td>
<td>4002  :</td>
<td>170</td>
</tr>
<tr>
<td>ICMD</td>
<td>006B  :</td>
<td>95</td>
</tr>
<tr>
<td>ICTL</td>
<td>003E  :</td>
<td>97</td>
</tr>
<tr>
<td>INITAD</td>
<td>F80F  :</td>
<td></td>
</tr>
<tr>
<td>INITCPU</td>
<td>F800  :</td>
<td></td>
</tr>
<tr>
<td>INITGP</td>
<td>F845  :</td>
<td></td>
</tr>
<tr>
<td>INITLP1</td>
<td>F879  :</td>
<td>151</td>
</tr>
<tr>
<td>INITPASS</td>
<td>F884  :</td>
<td></td>
</tr>
<tr>
<td>INITREAD</td>
<td>F85D  :</td>
<td></td>
</tr>
<tr>
<td>Variable</td>
<td>Value</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>INITRWD</td>
<td>F874 : 149</td>
<td></td>
</tr>
<tr>
<td>INITSER</td>
<td>F805 :</td>
<td></td>
</tr>
<tr>
<td>INITVAR</td>
<td>F888 :</td>
<td></td>
</tr>
<tr>
<td>LASTDAT</td>
<td>F907 : 224</td>
<td></td>
</tr>
<tr>
<td>LOOP</td>
<td>F8C7 : 235</td>
<td></td>
</tr>
<tr>
<td>NEXTB</td>
<td>F963 : 292</td>
<td></td>
</tr>
<tr>
<td>NO_READ</td>
<td>FA4A : 474</td>
<td></td>
</tr>
<tr>
<td>ONEMS</td>
<td>F9BA : 371 472</td>
<td></td>
</tr>
<tr>
<td>NESEC</td>
<td>F866 : 138</td>
<td></td>
</tr>
<tr>
<td>JFILE</td>
<td>= 0012 : 177</td>
<td></td>
</tr>
<tr>
<td>OUT1H</td>
<td>= 0005 : 199 262 263 444 447</td>
<td></td>
</tr>
<tr>
<td>OUTIL</td>
<td>= 0006 : 200 259 260 283 445 448</td>
<td></td>
</tr>
<tr>
<td>OUT2H</td>
<td>= 0007 : 201 270 271 451 454</td>
<td></td>
</tr>
<tr>
<td>OUT2L</td>
<td>= 0008 : 202 267 268 287 452 455</td>
<td></td>
</tr>
<tr>
<td>OVER</td>
<td>F95B :</td>
<td></td>
</tr>
<tr>
<td>PASS</td>
<td>= 000C : 154 187 244</td>
<td></td>
</tr>
<tr>
<td>PCHAR</td>
<td>F901 : 219</td>
<td></td>
</tr>
<tr>
<td>PCNTR</td>
<td>F8D9 : 216 226 229</td>
<td></td>
</tr>
<tr>
<td>PDATA</td>
<td>= 0009 : 193 234 307 315 318 326</td>
<td></td>
</tr>
<tr>
<td>PSRCNT</td>
<td>= 0001 : 205 218 221</td>
<td></td>
</tr>
<tr>
<td>PRINT</td>
<td>F9C7 : 134 140 145 178 188 241 243</td>
<td></td>
</tr>
<tr>
<td>PULSE</td>
<td>F8E4 : 212</td>
<td></td>
</tr>
<tr>
<td>RAM</td>
<td>= 0000 :</td>
<td></td>
</tr>
<tr>
<td>RDFILE</td>
<td>= 0011 : 133</td>
<td></td>
</tr>
<tr>
<td>READ1</td>
<td>F9E9 :</td>
<td></td>
</tr>
<tr>
<td>READ2</td>
<td>FA03 :</td>
<td></td>
</tr>
<tr>
<td>READAD</td>
<td>F9D3 : 222</td>
<td></td>
</tr>
<tr>
<td>READD1</td>
<td>FA35 : 477</td>
<td></td>
</tr>
<tr>
<td>READD2</td>
<td>FA33 : 147 180</td>
<td></td>
</tr>
<tr>
<td>RESTART</td>
<td>F8D1 :</td>
<td></td>
</tr>
<tr>
<td>RETRN</td>
<td>F9B5 : 349</td>
<td></td>
</tr>
<tr>
<td>REWIND</td>
<td>= 001A : 144</td>
<td></td>
</tr>
<tr>
<td>R_OK1</td>
<td>F9FC : 415</td>
<td></td>
</tr>
<tr>
<td>R_OK2</td>
<td>FA16 : 428</td>
<td></td>
</tr>
<tr>
<td>SHUT</td>
<td>F99E : 345</td>
<td></td>
</tr>
<tr>
<td>START</td>
<td>F890 : 172 175 248</td>
<td></td>
</tr>
<tr>
<td>SWCNT</td>
<td>= 000E : 157 213 246</td>
<td></td>
</tr>
<tr>
<td>SFLAG</td>
<td>= 000D : 159 174 348 351</td>
<td></td>
</tr>
<tr>
<td>TAPERD</td>
<td>F85F : 142</td>
<td></td>
</tr>
<tr>
<td>TOTAL1</td>
<td>F932 : 225</td>
<td></td>
</tr>
<tr>
<td>TOTAL2</td>
<td>F93F :</td>
<td></td>
</tr>
<tr>
<td>TSTXMT</td>
<td>F9C8 : 386</td>
<td></td>
</tr>
</tbody>
</table>

**Lines Assembled:** 496  **Assembly Errors:** 0
A.5 TAPEREAD PROGRAM
Program Tape_Read (Input,Output);

USES Screen;

{ Program To Read Information Off The Tape Unit And Store it In A Format Usable By Bridge }
{ Tape Unit 9600 Baud 7 Bits Even Parity }

Const
    CtrlZ = Chr(26);
    CtrlS = Chr(19);
    CtrlQ = Chr(17);
    CtrlO = Chr(15);
    Pad = Chr(31);

Type
    Buff_Type = Array [1..24000] Of Char;  { 6000 Feet }

Var
    TimeOut : Boolean;
    Dev_Setup : Dev;
    Dev_Setup2 : Dev;
    Length : Integer;
    Bridge_Id : LString(32);
    Date : LString(12);
    Square : Boolean;
    Right : Boolean;
    DistR : Integer;
    DistL : integer;
    No_Pass : Integer;
    Ch : Char;
    Left_Buff : Buff_Type;
    Left_L : Integer;
    Right_Buff : Buff_Type;
    Right_L : Integer;
    Temp : File Of Char;
    Data : File Of Char;
    OK : Boolean;
    Pass : Char;
    Expected : Char;
    First : Char;

Function Dosxqq ( Command, Parameter : Word ) : Byte; Extern;

Function Get_Int ( Limit : Integer ) : Integer; Extern;

Procedure Fit ( Var Data : Buff_Type; Var Len : Integer );

Const
    Debug = False;

Var
    i,j : Integer;
BEGIN
  { Procedure That Writes The Buffers To The Temporary File To Await }
  { Further Processing }
  Var
    i : Integer;

BEGIN
  For i := 1 To Length Do
    Write(Temp,Left_Buff[i]);
  For i := 1 To Length Do
    Write(Temp,Right_Buff[i]);
Procedure Get_File:
Const
Debug = False;
Var
Buff1 : Stat;
Buff2 : Stat;
InChar : Char;
BEGIN
If Debug Then
  Writeln('Procedure Get_File');
OK := True;
Left_L := 0;
Right_L := 0;
Queue_Status(Buff1,Buff2);
If (Ord(Buff1[2]) = 0) Then ( If Nothing Is In The Tape Buffer )
  PutChar(1,CtrlQ);
  ( Then Send The Read Command CtrlQ )
Pass := Char_From_Tape;
If (Not TimeOut) And (InChar <> CtrlS) Then
  BEGIN
    REPEAT
      InChar := Char_From_Tape;
      If (InChar <> CtrlS) And (Not TimeOut) Then
        BEGIN
          Left_L := Left_L + 1;
          Left_Buff[Left_L] := InChar;
          InChar := Char_From_Tape;
          If (InChar <> CtrlS) And (Not TimeOut) Then
            BEGIN
              Right_L := Right_L + 1;
              Right_Buff[Right_L] := InChar;
            END;
        END;
      END;
    UNTIL (TimeOut) Or (InChar = CtrlS);
  END;
END;
END;
Procedure FitBuffers:
BEGIN
  If (Left_L <> Length) Then
    Fit(Left_Buff,Left_L);
  If (Right_L <> Length) Then
    Fit(Right_Buff,Right_L);
  No_Pass := No_Pass + 1;
  Expected := Succ(Expected);
END;
Procedure Decide ( Var Ok : Boolean );
BEGIN
Ok := True;
If (Not TimeOut) Then
BEGIN
Writeln('Pass #' , (Ord(Pass) - Ord(First) + 1):3, 'Samples Taken');
If (Pass = Expected) Then
  If (No_Pass = 0) Then
    If (Left_L > 30) Then { First Pass Long Enough }
    BEGIN
      Length := Left_L;
      FitBuffers;
    END
  Else BEGIN
    Writeln('The First Pass Is Rather Short');
    Write('Do You Want To Use It ?');
    REPEAT
      Ch := Chr(Dosxqq(6,255));
    UNTIL (Ch In ['n','N','y','Y']);
    Writeln;
    If (Ch In ['y','Y']) Then
    BEGIN
      Length := Left_L;
      FitBuffers;
    END
    Else Ok := False;
    END
  Else If (Left_L > 0.9 * Length) Then
    FitBuffers
  Else BEGIN
    Writeln('This Pass Is Only ' ,(Left_L / Length * 100):5:2, ' As Long As The First');
    Write('Do You Want To Use It ?');
    REPEAT
      Ch := Chr(Dosxqq(6,255));
    UNTIL (Ch In ['n','N','y','Y']);
    Writeln;
    If (Ch In ['y','Y']) Then
      FitBuffers
    Else Ok := False;
  END
Else BEGIN
  Writeln('Pass #' , (Ord(Expected) - Ord(First) + 1):3, 'Was Expected ');
  Writeln(Do You Want To Use It For Pass #' ,(Ord(Expected) - Ord(First) + 1):3, ', ');
  Write('Not Use It Or Insert A Blank Pass (Y,N,I) ?');
  REPEAT
    Ch := Chr(Dosxqq(6,255));
  UNTIL (Ch In ['n','N','y','Y','i','I']);
  Writeln;
  If (Ch In ['y','Y']) Then
  BEGIN
    Expected := Pass;
    If (No_Pass = 0) Then
      First := Pass;
    Decide(Ok);
  END
  Else If (Ch In ['i','I']) Then
  BEGIN
    If (No_Pass = 0) Then
      Length := Left_L;
      If (No_Pass = 0) Then
        First := Pass;
      Decide(Ok);
  END
END
insert_blank;
No_Pass := No_Pass + 1;
Expected := Succ(Expected);
Decide(Ok);
END
Else Ok := False;
END;
END
Else Ok := False;
END;

Procedure Trim_Edges ( Var Buff : Buff_Type; p, c : Integer );
Const
Debug = False;
Var
i : Integer;
Off_Top : Integer;
Off_Bottom : Integer;
BEGIN
If (Not Right) Then
  p := c - p - 1;
  p := p * 9 + 5;
  c := c * 9 + 4;
  Off_Top := (DistR + ((c - p) * (DistL - DistR)) Div c) Div 3;
  Off_Bottom := DistR Div 3 + DistL Div 3 - Off_Top;
  If (Debug) Then
    Writeln('Trim Top = ',Off_Top:3,' Bottom = ',Off_Bottom:3);
    For i := 1 To Off_Top Do
      Buff[i] := Pad;
    For i := Length DownTo (Length - Off_Bottom) Do
      Buff[i] := Pad;
END;

Procedure Write_Data_File;
Const
Debug = False;
Var
Name : LString(32);
i, j : Integer;
BEGIN
Write(' Data File Name : ');
Readln(Name);
Assign(Data,Name);
Reset(Temp);
Rewrite(Data);
For i := 0 To 32 Do
  Write(Data,Bridge_Id[i]);
Write(Data,Chr(No_Pass));
Write(Data,Chr(Length Div 256),Chr(Length Mod 256));
If (Square) Then
  Write(Data,'N')
Else Write(Data,'S');
Write(Data,Chr(DistR Div 256),Chr(DistR Mod 256));
Write(Data,Chr(DistL Div 256),Chr(DistL Mod 256));
If (Right) Then
  Write(Data,'R')
Else Write(Data,'L');
For i := 42 To 54 Do
  Write(Data,Data[i - 42]);
For i := 55 To 63 Do

write(data, 1);

for i := 0 to (no_pass - 1) do

begin

if (odd(i)) then

for j := 1 to length do

read(temp, left_buff[j]);

else for j := length downto 1 do

read(temp, left_buff[j]);

if (odd(i)) then

for j := 1 to length do

read(temp, right_buff[j]);

else for j := length downto 1 do

read(temp, right_buff[j]);

if (not square) then

trim_edges(left_buff, 2 * i, 2 * no_pass - 1);

for j := 1 to length do

write(data, left_buff[j]);

if (not square) then

trim_edges(right_buff, 2 * i + 1, 2 * no_pass - 1);

for j := 1 to length do

write(data, right_buff[j]);

end;

close(data);

end;

begin

clear_screen;

home;

writeln('iowa d.o.t.');

writeln;

readln(dev_setup0, dev_setup2);

dev_setup[0] := chr(13);

dev_setup[1] := chr(2);

dev_setup[2] := chr(120);

dev_setup[3] := chr(0);

dev_setup[4] := chr(0);

dev_setup[5] := chr(0);


dev_setup[7] := ctrl_q;

dev_control1(dev_setup);

ctrl1 true;

putchar(1, ctrlz);

{ rewind the tape }

write(' bridge id : ');  

first := 'a';

readln(bridge_id);

write(' date : ');  

readln(date);

timeout := false;

repeat

write(' start on the right or left : ');  

readln(ch);

until (ch in ['l', 'l', 'r', 'r']);  

right := (ch in ['r', 'r']);

repeat

write(' normal or slewed : ');  

readln(ch);

until (ch in ['n', 'n', 's', 's']);

square := (ch in ['n', 'n']);

distr := 0;

distl := 0;

if (not square) then

begin

write(' distance in inches on the right : ');  

distr := get_int(4);  

write(' distance in inches on the left : ');  

DistL := Get_Int(4);
END;
Length := 0;
Expected := 'A';
Assign(Temp,'Scratch.me');
Rewrite(Temp);
Ch := Char_From_Tape;
While (Not_TimeOut) Do
   BEGIN
      Get_File;
      Decide(Ok);
      If (Ok) Then
         Write_File;
   END;
PutChar(1,CtrlZ);  { Rewind The Tape }  
DisableI;
If (No_Pass <> 0) Then
   Write_Data_File
Else Writeln(' No Passes Found Or Used ');
Discard(Temp);
END.
A.6 BRIDGE PLOT PROGRAM
Program Bridge (Input, Output);

USES Screen(Home, Clear_Screen, Clear_Line, Up, Down, Left, Right, Pos,
            Reverse, Norm, Setup_Screen, Cursor_Off, Cursor_On);

{ Program to analyse digitised soundings of bridge sections }

CONST
Max_Length = 24000; { 6000 Feet }
Min_Value = 31;
Max_Value = 127;
Def_Delam = 400; { In Millivolts }
Increment = 21; { In Millivolts }
First = 'A';
Space = 8;
Esc = Chr(27); { TI 855 Compatible }

Type
Tape_Type = Array [1..Max_Length] Of Char;
Ord_Type = Array [1..Max_Length] Of Integer;
Dist_Type = Array [0..Max_Value] Of Integer;
Bit_Map_Type = Super Array [1..*,1..*] Of Byte;
Bit_Map_Ptr = ^Bit_Map_Type;
ID_Type = LString(32);
Date_Type = LString(12);
Count_Type = Array [0..255] Of Integer;
Dens_Type = LString(2);

VAR
Bridge_ID : ID_Type;
Date : Date_Type;
Num_Passes : Integer;
Normal : Boolean;
Dr, Dl : Integer;
Start_Right : Boolean;
Delam : Integer;
Left_Data : Tape_Type;
Right_Data : Tape_Type;
Length : Integer;
Width : Integer;
Pass : Integer;
OK : Boolean;
Data_Valid : Boolean;
Init : Boolean;
Init_Printer : Boolean;
Bit_Map : Bit_Map_Ptr;
Offset : Integer;
Map_Length : Integer;
Prn : Text;
Info : File Of Char;
Power : Array [0..7] Of Integer;
Total : Integer;
Bad : Integer;
Percent : Real;
Printer_Type : Integer;
Star : Dens_Type;
Function Dosxqq (Command, Parameter : Word) : Byte; Extern;

Function Get_Int (Limit : Integer) : Integer; Extern;

Procedure Get_information;

(Procedure to get from the user the name of a file containing the bridge data with the following format)

<table>
<thead>
<tr>
<th>Information</th>
<th>Bytes In File</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bridge_Id</td>
<td>0 – 32</td>
</tr>
<tr>
<td># Passes</td>
<td>33</td>
</tr>
<tr>
<td>Length in samples</td>
<td>34 – 35</td>
</tr>
<tr>
<td>Normal or Slewed</td>
<td>36</td>
</tr>
<tr>
<td>Distance in inches (R)</td>
<td>37 – 38</td>
</tr>
<tr>
<td>Distance in inches (L)</td>
<td>39 – 40</td>
</tr>
<tr>
<td>Start_Right</td>
<td>41</td>
</tr>
<tr>
<td>Date Info</td>
<td>42 – 54</td>
</tr>
<tr>
<td>Extra space</td>
<td>55 – 63</td>
</tr>
<tr>
<td>Bridge data</td>
<td>64 – ?</td>
</tr>
</tbody>
</table>

CONST
Debug = False;

VAR
Ch : Char;
File_Name : ID_Type;
Found : Boolean;
i : Integer;

BEGIN
Data_Valid := True;
Num_Passes := 0;
Length := 0;
Normal := True;
Dr := 0;
Dl := 0;
Start_Right := True;
REPEAT
Clear_Screen;
Home;
WriteIn('IOWA D.O.T.');
WriteIn;
WriteIn('Bridge Data File: ');
Readln(File_Name);
Assign(Info, File_Name);
Info.Trap := True; { Enable Error Trapping }
Reset(Info);
If (Info.Errs <> 0) Then { Error With File }
BEGIN
Info.Errs := 0; { Reset Error }
WriteIn('File_Name:Ord(File_Name[0]), Not Found');
Found := False;
END
Else Found := True;
Until (Found); { No Errors With File }
WriteIn;
For i := 0 To 32 Do
If (Not Eof(Info)) Then
Read(Info, Bridge_id[i])
Else Data_Valid := False;
If (Data_Valid) Then
WriteIn(Bridge_Id:48)
Else Writeln(' ERROR IN DATA FILE');
Writeln;
If (Not Eof(Info)) And (Data_Valid) Then
   BEGIN
      Read(Info,Ch);
      Num_Passes := Ord(Ch);
   END
Else Data_Valid := False;
If (Not Eof(Info)) And (Data_Valid) Then
   BEGIN
      Read(Info,Ch);
      Length := Ord(Ch);
   END
Else Data_Valid := False;
If (Not Eof(Info)) And (Data_Valid) Then
   BEGIN
      Read(Info,Ch);
      Length := Length * 256 + Ord(Ch);
   END
Else BEGIN
   Data_Valid := False;
   Length := 0;
   END;
If (Not Eof(Info)) And (Data_Valid) Then
   BEGIN
      Read(Info,Ch);
      Normal := (Ch In ['n','N']);
   END
Else Data_Valid := False;
If (Not Eof(Info)) And (Data_Valid) Then
   BEGIN
      Read(Info,Ch);
      Dr := Ord(Ch);
   END
Else Data_Valid := False;
If (Not Eof(Info)) And (Data_Valid) Then
   BEGIN
      Read(Info,Ch);
      Dr := Dr * 256 + Ord(ch);
   END
Else BEGIN
   Data_Valid := False;
   Dr := 0;
   END;
If (Not Eof(Info)) And (Data_Valid) Then
   BEGIN
      Read(Info,Ch);
      Dl := Ord(Ch);
   END
Else Data_Valid := False;
If (Not Eof(Info)) And (Data_Valid) Then
   BEGIN
      Read(Info,Ch);
      Dl := Dl * 256 + Ord(ch);
   END
Else BEGIN
   Data_Valid := False;
   Dl := 0;
   END;
If (Not Eof(Info)) And (Data_Valid) Then
   BEGIN
      Read(Info,Ch);
      Start_Right := (Ch In ['r','R']);
   END
Else Data_Valid := False;
For i := 42 To 54 Do
If (Not Eof(Info)) Then
   Read(Info,Date[i - 42]);
For i := 55 To 63 Do
   If (Not Eof(Info)) Then
      Read(Info,Ch)
   Else Data_Valid := False;
   REPEAT
      Write(' Delamination In mV ('.Def_Delam:3,') : ');
      Delam := Get_Int(4);
      If (Delam = 0) Then Delam := Def_Delam;
      UNTIL (Delam > 0);
END;

Procedure Read_Data_From_Tape:

CONST
   Debug = False;
VAR
   L_Length : Integer;
   R_Length : Integer;
   Pass_Found : Integer;
   Answer : Char;
   i : Integer;
BEGIN
   If Debug Then
      Writeln(' Read Data From Tape');
   For i := 1 To Length Do
      If (Not Eof(Info)) Then
         Read(Info,Left_Data[i])
      Else Data_Valid := False;
   For i := 1 To Length Do
      If (Not Eof(Info)) Then
         Read(Info,Right_Data[i])
      Else Data_Valid := False;
   Ok := Data_Valid;
END;

Function BaseLine (Var Data : Tape_Type) : Integer;
{ Determines The Baseline Of The Data. Assumes That The Most
   Frequently Occuring Value Is The Baseline }

CONST
   Debug = False;
VAR
   Dist : Dist_Type;
   Base : Integer;
   i : Integer;
BEGIN
   If Debug Then Writeln(' BaseLine');
   For i := Min_Value To Max_Value Do
      Dist[i] := 0;
   For i := 1 To Length Do
      Dist[Ord(Data[i])] := Dist[Ord(Data[i])] + 1;
   i := Min_Value + 1;
   Base := i;
   While (i <= 100) Do
      BEGIN
         If (Dist[Base] < Dist[i]) Then
            Base := i;
      END;
      i := i + 1;
   END;
END;
Baseline := Base;
END;

Procedure insert_into_Bitmap;
( Inserts The Tape Data Into The Bit Map )

CONST
Debug = False;

VAR
i, j : Integer;
Row : Integer;
Col : Integer;
Base_L : Integer;
Base_R : Integer;
Delam_L : Integer;
Delam_R : Integer;
Bit : Byte;

BEGIN
If Debug Then Writeln('Insert_into_Bitmap');
If (Init) Then
BEGIN
Map_Length := ((Length - 1) Div 8 + 1);
Offset := Map_Length * 8 - Length;
New(Bit_Map,Map_Length,Width);
For i := 1 To Map_Length Do
  For j := 1 To Width Do
    Bit_Map^ [i, j] := 0;
  Init := False;
END;

If (Start_Right) Then
  Col := Width - Pass * 2 + 1
Else
  Col := Pass * 2 - 1;
Base_L := Baseline(Left_Data);
Base_R := Baseline(Right_Data);
Delam_L := Base_L + Delam Div Increment;
Delam_R := Base_R + Delam Div Increment;
For i := (Length - 1) DownTo 0 Do
BEGIN
  Bit := Power[(i + Offset) Mod 8];
  Row := (i + Offset) Div 8 + 1;
  If (Ord(Left_Data[i + 1]) >= Delam_L) Then
    Bit_Map^ [Row, Col] := Bit_Map^ [Row, Col] + Bit;
  If (Ord(Right_Data[i + 1]) >= Delam_R) Then
    Bit_Map^ [Row, Col + 1] := Bit_Map^ [Row, Col + 1] + Bit;
END;
END;

Procedure Percentage;

VAR
i, j, k : Integer;

BEGIN
Total := 0;
Bad := 0;
Percent := 0;
For i := 1 To Map_Length Do
  For j := 1 To Width Do
    For k := 0 To 7 Do
      If (Odd(Bit_Map^ [i, j] Div Power[k])) Then
        Bad := Bad + 1;
Total := Length * Width;
If (Not Normal) Then
    Total := Total - (Dr + Dl) * Width;
    Percent := Bad / Total;
END;

Procedure Process_Section;

CONST
    Debug = False;

BEGIN
    If Debug Then WriteIn(’ Process_Section’);
    OK := True;
    Init := True;
    Pass := 1;
    Width := (2 * Num_Passes);
    While (Pass <= Num_Passes) Do
        BEGIN
            Read_Data_From_Tape;
            If (Ok) Then
                Insert Into_BitMap
            Else Pass := Num_Passes;
            Pass := Pass + 1;
        END;
    Percentage;
END;

Procedure Header;

BEGIN
    If (Init_Printer) And (Printer_Type = 1) Then
        BEGIN
            Writeln(Prn,Esc,’@’);
            Write(Prn,Esc,’q’);
        END
    Else If (Init_Printer) And (Printer_Type = 2) Then
        BEGIN
            Writeln(Prn,Esc,’@’);
            Write(Prn,Esc,’G’);
        END
    Else If (Init_Printer) And (Printer_Type = 3) Then
        BEGIN
            Writeln(Prn,Esc,’@’);
            Write(Prn,Esc,’4’);
        END;
    Init_Printer := False;
    Writeln(Prn,’IOWA D.O.T.’:45);
    Writeln(Prn,’DELAMINATION MAP’:48);
    Writeln(Prn);
    Writeln(Prn,Bridge_Id:46);
    Writeln(Prn);
    Writeln(Prn,Date:(Ord(Date[0]) + 18),’ :(21 - Ord(Date[0])),’Delamination at ’,Delam:5,’ mV’);
    Write(Prn);
    Write(Prn,’Length = ’:20,((Length * 3) Div 12):2,’ Feet ’,((Length * 3) Mod 12 ):2,’ Inches’);
    Write(Prn);
    Write(F1t-n,’Total Area = ’:24,(Total / 144 * 27):7:2,’ Square Feet’);
Procedure Dumb_Printer;

BEGIN
  Header;
END;

Procedure Print_Top ( Density : Dens_Type );

VAR
  i : Integer;
BEGIN
  Write(Prn,Esc,Density,Chr((12 * Width + 5) Mod 256),Chr((12 * Width + 2) Div 2 56));
  For i := 1 To (12 * Width + 5) Do
    Write(Prn,Chr(1));
END;

Procedure Print_Bar ( Density : Dens_Type );

BEGIN
  Write(Prn,Esc,Density,Chr(1),Chr(0),Chr(255));
END;

Procedure Print_Tic ( Density : Dens_Type; Which : Integer );

BEGIN
  Write(Prn,Esc,Density,Chr(3),Chr(0));
  If (Odd(Which)) Then
    Write(Prn,Chr(128),Chr(128))
  Else Write(Prn,Chr(0),Chr(0));
  Write(Prn,Chr(128));
END;

Procedure Print_Bottom ( Density : Dens_Type );

VAR
  i,j : Integer;
  Spacing : Integer;
BEGIN
  Write(Prn,Esc,Density,Chr((12 * Width + 5) Mod 256),Chr((12 * Width + 2) Div 2 56));
  Write(Prn,Chr(128),Chr(128),Chr(128));
  For i := 0 To (Width * 12 - 1) Do
    If (i Mod 16 = 0) Then
      Write(Prn,Chr(248))
    Else If (i Mod 8 = 0) Then
      Write(Prn,Chr(224))
    Else Write(Prn,Chr(128));
  If ((Width * 12) Mod 16 = 0) Then
    Write(Prn,Chr(248),Chr(128))
  Else If ((Width * 12) Mod 8 = 0) Then
    Write(Prn,Chr(224),Chr(128))
  Else Write(Prn,Chr(128),Chr(192));
  Writeln(Prn);
  Write(Prn,'O':6);
  If (Printer_Type = 1) Or (Printer_Type = 3) Then
    Spacing := 18
  Else If (Printer_Type = 2) Then
    Spacing := 20
  Else Spacing := 1;
BEGIN
  Write(Prn,Esc,Density,Chr(Spacing),Chr(0));
  For j := 1 To Spacing Do
    Write(Prn,Chr(0));
    Write(Prn,(i + i):2);
  END;
END;

Procedure Scale_4_To_3 ( Density : Dens_Type );

CONST
  Debug = False;

VAR
  N1 : Char;
  N2 : Char;
  i, j, k, m, n : Integer;
  Ch : Integer;
  Bit : Byte;

BEGIN
  Header;
  n := (Length - 1) Div 8;
  If Debug Then
    WriteIn('Scale 4 To 3 - Density = ',Density);
  N1 := Chr((Width * 12) Mod 256);
  N2 := Chr((Width * 12) Div 256);
  Writeln(Prn,Esc,'3',Chr(24)); \{ Sets Line Spacing \}
  Writeln(Prn);
  Writeln(Prn,'FEET':5);
  Writeln(Prn,'':5);
  Print_Top(Density);
  Writeln(Prn);
  For i := 1 To Map_Length Do
    BEGIN
      Write(Prn,(n - i + 2) * 2:5);
      Print_Tic(Density,1);
      Print_Bar(Density);
      Write(Prn,Esc,Density,N1,N2);
      For k := 1 To Width Do
        BEGIN
          Bit := 1;
          If (Odd(Bit_Map^[i,k] Div Bit)) Then
            Ch := 240
          Else Ch := 0;
          Bit := Bit + Bit;
          If (Odd(Bit_Map^[i,k] Div Bit)) Then
            Ch := Ch + 15;
          For m := 1 To 12 Do
            Write(Prn,Chr(Ch));
        END;
      Print_Bar(Density);
      WriteIn(Prn);
      WriteIn(Prn,'':5);
      Print_Tic(Density,2);
      Print_Bar(Density);
      Write(Prn,Esc,Density,N1,N2);
      For k := 1 To Width Do
        BEGIN
          Bit := 4;
          If (Odd(Bit_Map^[i,k] Div Bit)) Then
            Ch := 240
          Else Ch := 0;
          Bit := Bit + Bit;
          If (Odd(Bit_Map^[i,k] Div Bit)) Then
            Ch := Ch + 15;
          For m := 1 To 12 Do
            Write(Prn,Chr(Ch));
        END;
      End;
    END;
  END;
END;
Ch := Ch + 15;
For m := 1 To 12 Do
  Write(Prn, Chr(Ch));
END;
Print_Bar(Density);
WriteIn(Prn);
Write(Prn, ':5);
Print_Tic(Density,3);
Print_Bar(Density);
Write(Prn, Esc, Density, N1, N2);
For k := 1 To Width Do
  BEGIN
    Bit := 16;
    If (Odd(Bit, Map[i,k] Div Bit)) Then
      Ch := 240
    Else Ch := 0;
    Bit := Bit + Bit;
    If (Odd(Bit, Map[i,k] Div Bit)) Then
      Ch := Ch + 15;
    For m := 1 To 12 Do
      Write(Prn, Chr(Ch));
  END;
Print_Bar(Density);
WriteIn(Prn);
Write(Prn, ':5);
Print_Tic(Density,4);
Print_Bar(Density);
Write(Prn, Esc, Density, N1, N2);
For k := 1 To Width Do
  BEGIN
    Bit := 64;
    If (Odd(Bit, Map[i,k] Div Bit)) Then
      Ch := 240
    Else Ch := 0;
    Bit := Bit + Bit;
    If (Odd(Bit, Map[i,k] Div Bit)) Then
      Ch := Ch + 15;
    For m := 1 To 12 Do
      Write(Prn, Chr(Ch));
  END;
Print_Bar(Density);
WriteIn(Prn);
END;
Write(Prn, 0:5);
Print_Bottom(Density);
WriteIn(Prn);
If Start_Right Then
  If (Printer_Type = 2) Then
    WriteIn(Prn, 'START': (8 + Width * 2))
  Else WriteIn(Prn, 'START': (8 + Width + (8 * Width) Div 10))
Else WriteIn(Prn, 'START': 8);
WriteIn(Prn, Esc, '2', Chr(12));
END;

Procedure Graph (Density : Dens_Type);

CONST
  Debug = False;

VAR
  n, m : Integer;
  n1, n2 : Integer;
  mL, m0 : Integer;
  i, j : Integer;
  Bit : Integer;
  Set_Bit : Integer;
Test_Bit : Integer;
Row : Integer;
Col : Integer;
Dist : Integer;
Edge : Integer;
Line : Array [-4..600] Of Integer;

BEGIN
If (Debug) Then
  Writeln(' Procedure Graph');
If (Printer_Type = 1) Or (Printer_Type = 3) Then
  n := (512 Div Width);
Else n := (420 Div Width);  // Number of dots per bit horizontally
m := (n Div 3);            // Number of dots vertically
If (m = 0) Then
  m := 1;
mL := m * Length;
m0 := m * Offset;
Dist := 0;
n := 3 * m;                // Set for true 3:1 ratio
n1 := (Width * n + 6) Mod 256;
n2 := (Width * n + 6) Div 256;
If (Debug) Then
  Writeln(' Vert = ',m:4,' Horz = ',n:4,' n1, n2 ',n1:3,n2:4);
Line[0] := 255;
WriteLn(Prn,Esc,'3',Ct>r(24));
WriteLn(Prn,':GI;
WriteLn(Prn,Esc,Density,Ct~r~n1),Ch~-~n2~,Chr(O~,Chr~O~,Chr~O~~;
For i := 1 To (Width * n + 2) Do
  Writeln(Prn);
Set_Bit := 128;
For i := -4 To 480 Do
  Line[i] := 0;
If (((Length Mod 4) = 0) Then
  BEGIN
    Line[-1] := 128;
    Line[-2] := 128;
  END;
If (((Length Mod 8) = 0) Then
  BEGIN
    Line[-3] := 128;
    Dist := (Length Div 8) * 2;
    Line[-4] := 128;
  END;
For Bit := 0 To (mL - 1) Do
  BEGIN
    Row := (((Bit + mO) Div m) Div 8 + 1;
    i := (((Bit + mO) Div m) Mod 8;
    Test_Bit := Power[i];
    If (((mL - Bit) Mod (4 * m)) = 0) Then
      BEGIN
        Line[-1] := Set_Bit;
        Line[-2] := Set_Bit;
      END;
    If (((mL - Bit) Mod (8 * m)) = 0) Then
      BEGIN
        Line[-3] := Set_Bit;
        Line[-4] := Set_Bit;
        Dist := (mL - Bit) Div (8 * m) * 2;
      END;
    For Col := 1 To Width Do
      If (Odd(Bit_Map^[Row,Col] Div Test_Bit)) Then
        Line[Col] := Line[Col] + Set_Bit;
    Set_Bit := Set_Bit Div 2;
  END;
If (Set_Bit = 0) Then

BEGIN
  If (Dist = 0) Then
    Write(Prn, ', ':6);
  Else Write(Prn, Dist:6);
  Dist := 0;
  Write(Prn, Esc, Density, Chr(nl), Chr(n2));
  For i := -4 To 0 Do
    BEGIN
      Write(Prn, Chr(Line[i]));
      Line[i] := 0;
    END;
  For i := 1 To Width Do
    BEGIN
      For j := 1 To n Do
        Write(Prn, Chr(Line[i]));
      Line[i] := 0;
    END;
  Write(Prn, Esc, Density, Chr(nl), Chr(n2));
END;

For i := -4 To Width Do
  BEGIN
    For j := 1 To n Do
      Write(Prn, Chr(Line[i]));
    Line[i] := 0;
  END;
  Line[0] := 255;
  Write(Prn, 'O':6);
  Line[0] := 255;
  Set_Bit := 128;
  END;

For i := -4 To -1 Do
  BEGIN
    Line[i] := 0;
  END;

For i := -4 To Width Do
  BEGIN
    Line[i] := Line[i] + Set_Bit;
    Edge := 255 - (Set_Bit Div 2) - (Set_Bit Div 4) - (Set_Bit Div 8) - (Set_Bit Div 16) -
    (Set_Bit Div 32) - (Set_Bit Div 64) - (Set_Bit Div 128);
    Line[0] := Edge;
  END;

Write(Prn, 'O':6);

m := 2;
Set_Bit := Set_Bit Div 2;
If (Set_Bit = 0) Then
  BEGIN
    Dist := 0;
    Write(Prn, Esc, Density, Chr(nl), Chr(n2));
    For i := -4 To 0 Do
      BEGIN
        Write(Prn, Chr(Line[i]));
      END;
    For i := 1 To Width Do
      BEGIN
        For j := 1 To n Do
          Write(Prn, Chr(Line[i]));
      END;
    Write(Prn, 'O':6);
  END;

For Bit := 1 To 4 Do
  BEGIN
    For Col := 0 To (Width Div m) Do
      BEGIN
        Line[Col * m] := Line[Col * m] + Set_Bit;
        If (Bit = 2) Then
          m := m + m;
        Set_Bit := Set_Bit Div 2;
      END;
    If (Set_Bit = 0) Then
      BEGIN
        Write(Prn, Esc, Density, Chr(nl), Chr(n2));
        For i := -4 To 0 Do
          BEGIN
            Write(Prn, Chr(Line[i]));
          END;
        Line[i] := 0;
      END;
  END;
For i := 1 To Width Do
BEGIN
    For j := 2 To n Do
        Write(Prn,Chr(Line[i]));
        Write(Prn,Chr(Line[i]));
        Line[i] := 0;
    END;
    Writeln(Prn,Chr(255));
    Line[0] := 255;
    Set_Bit := 128;
    Write(Prn,'':6);
END;

if (Set_Bit <> 128) then
BEGIN
    Write(Prn,Esc,Density,Chr(n1),Chr(n2));
    For i := -4 To 0 Do
        Write(Prn,Chr(Line[i]));
    For i := 1 To Width Do
        BEGIN
            For j := 2 To n Do
                Write(Prn,Chr(Line[i]));
                Write(Prn,Chr(Line[i]));
            END;
        Writeln(Prn,Chr(0));
    END;
    Write(Prn,'0':7);
    For i := 1 To (Width Div 4) Do
        BEGIN
            Write(Prn,Esc,Density,Chr((4 * n - 14) Mod 256),Chr((4 * n - 14) Div 256));
            For j := 1 To (4 * n - 14) Do
                Write(Prn,Chr(0));
                Write(Prn,(i * 3):2);
        END;
    Writeln(Prn);
    If (Start_Right) then
        Writeln(Prn,'START':76)
    else
        Writeln(Prn,'START':%;
    Writeln(Prn,Esc,'Z');
END;

Procedure Map;
BEGIN
    Header;
    If (Printer_Type = 1) then
        Graph('N')
    else if (Printer_Type = 2) then
        Graph('K')
    else if (Printer_Type = 3) then
        Graph(Star)
    else Dumb_Printer;
    Writeln(Prn,Chr(12));
END;

Procedure Init_Count ( Var Count : Count_Type );
CONST
    Debug = False;
VAR
    i,j : Integer;
BEGIN
if (Debug) then
  WriteLn('Procedure Init_Count');
For i := 0 To 255 Do
  BEGIN
    If (Odd(i Div Power[0])) Then
      Count[i] := 1
    Else Count[i] := 0;
    If (Odd(i Div Power[1])) Then
      Count[i] := 1 + Count[i];
    If (Odd(i Div Power[2])) Then
      Count[i] := 1 + Count[i];
    If (Odd(i Div Power[3])) Then
      Count[i] := 1 + Count[i];
    If (Odd(i Div Power[4])) Then
      Count[i] := 1 + Count[i];
    If (Odd(i Div Power[5])) Then
      Count[i] := 1 + Count[i];
    If (Odd(i Div Power[6])) Then
      Count[i] := 1 + Count[i];
    If (Odd(i Div Power[7])) Then
      Count[i] := 1 + Count[i];
  END;
If (Debug) Then
  BEGIN
    WriteLn('');
    WriteLn(Prn, i:4);
    WriteLn(Prn, ' ---
    WriteLn(Prn, ' ---
    For i := 0 To 15 Do
      BEGIN
        WriteLn(Prn, ' ---
        WriteLn(Prn, ' ---
      END;
  END;

Procedure Map_Percent;

CONST
  Debug = False;

VAR
  Count : Count_Type;
  Totals : Array [0..64] Of Integer;
  Col_Off : integer;
  Row_Off : integer;
  No_Percent : integer;
  i, j : integer;
  Index : integer;

BEGIN
  If (Debug) Then
    WriteLn('Procedure Map_Percent');
  IF (Width > 40) AND (Printer_Type In [1,2]) Then  (Compressed)
    WriteLn(Prn, Chr(15));  (Print Mode)
  Header:
  If (Odd(Map_Length)) Then
    Row_Off := 1
  Else Row_Off := 0;
  If (Start_Right) Then
    WriteLn(Prn, ' ---

-89-
COI_OFF := ((Width - 1) DIV 4 + 1) * 4 - Width
Else COI_OFF := 0;
No_Percent := ((Width - 1) Div 4);
Init_Count(Count);
For i := 0 To 64 Do
  Totals[i] := 0;
Writeln(Prn, ' FEET');
Write(Prn, ((Map_Length - 1) Div 2 + 1) * 4):6,'--');
For i := 0 To No_Percent Do
  Write(Prn, '--------');
Writeln(Prn);
For i := 1 To Map_Length Do
  BEGIN
    For j := 0 To (Width - 1) Do
      BEGIN
        Index := Bit_Map'[i,j + 1];
        If (Index < 0) Then
          Index := Index + 256;
      END;
    If (Not Odd(i + Row_OFF)) Then
      BEGIN
        Write(Prn,'#':8);
        For j := 0 To No_Percent Do
          Write(Prn,'#':7);
        Writeln(Prn);
        Write(Prn,'#':8);
        For j := 0 To No_Percent Do
          Write(Prn,'#':7);
        Writeln(Prn);
        Write(Prn,((Map_Length - i) Div 2) * 4):6,'--');
      END;
    For j := 0 To No_Percent Do
      Write(Prn,'#':7);
    Writeln(Prn);
    Write(Prn,#':8);
    For j := 0 To No_Percent Do
      Writeln(Prn);
    Write(Prn,'#':8);
    For j := 0 To No_Percent Do
      Writeln(Prn);
    Write(Prn,(Totals[j] / 0.64):5:1,'');
    Writeln(Prn);
    Write(Prn,'#':8);
    For j := 0 To No_Percent Do
      Write(Totals[j]:5);
    Writeln;
    For j := 0 To No_Percent Do
      Totals[j] := 0;
    END;
    If (Debug) Then
      BEGIN
        For j := 0 To No_Percent Do
          Write(Totals[j]:5);
        Writeln;
      END;
    END;
  END;
Writeln(Prn,'#':8);
For i := 0 To No_Percent Do
  Write(Prn,'#':7);
Writeln(Prn);
Write(Prn,'#':8);
For i := 1 To No_Percent + 1 Do
  Write(Prn,(i * 3):7);
Writeln(Prn);
If (Start_Right) Then
  Writeln(Prn,'START':(18 + 7 * No_Percent))
Else Writeln(Prn,'START':11);
If (Width > 40) AND (Printer_Type In [1,2]) Then
  ( Compressed )
  ( Print Mode )
  ( Form Feed )
END;

Procedure Display_Options;
CONST
Debug = False;
Valid = ['1','2','p','P','t','T','s','S','e','E','a','A'];

VAR
In_Ch : integer;
Ch     : char;
Quit   : boolean;
String : LString(80);

BEGIN
If Debug Then Writeln(' Display_Options');
Quit := False;
REPEAT
  Clear_Screen;
  Home;
  Writeln;
  Writeln('IOWA D.O.T.');
  Writeln('DElamination Maps');
  Writeln('1 : 8 Inches Wide');
  Writeln('2 : 4 Dots Per 3 Inches');
  Writeln;
  Writeln('FUNCTIONS');
  Writeln('P : Percentages');
  Writeln('f :');
  String := 'T : TI 855 Printer';
  If (Printer_Type = 1) Then
    Reverse(String)
  Else Write(String);
  Writeln;
  Writeln;
  String := 'S : Star Printer';
  If (Printer_Type = 3) Then
    Reverse(String)
  Else Write(String);
  Writeln;
  Writeln;
  String := 'E : Epson Printer';
  If (Printer_Type = 2) Then
    Reverse(String)
  Else Write(String);
  Writeln;
  Writeln;
  String := 'A : Alphanumeric Printer';
  If Not (Printer_Type In [1,2,3]) Then
    Reverse(String)
  Else Write(String);
  Writeln;
  Writeln;
  Writeln('Q : Quit');
REPEAT
  In_Ch := Dosxqq(6,255);
  Until (In_Ch <> 0);
  Ch := Chr(In_Ch);
  Writeln;
  Writeln;
  If (Ch In Valid) Then
    Write(' Working...');
  Cursor_Off;
  Case Ch Of
    '1' : Map;
    '2' : If (Printer_Type = 1) Then
      If (Width < 45) Then
        Scale_4_To_3('N') -91-
      Else Scale_4_To_3('O')
Else If (Printer_Type = 2) Then
  If (Width < 35) Then
    Scale_4_To_3('K')
  Else Scale_4_To_3('L')
Else If (Printer_Type = 3) Then
  Scale_4_To_3(Star)
Else Dumb_Printer;

'p','P' : Map_Percent;
't','T' : BEGIN
  Init_Printer := True;
  Printer_Type := 1;
END;
'e','E' : BEGIN
  Init_Printer := True;
  Printer_Type := 2;
END;
's','S' : BEGIN
  Init_Printer := True;
  Printer_Type := 3;
END;
'a','A' : BEGIN
  Init_Printer := False;
  Printer_Type := -1;
END;
'Q','q' : Quit := True;
Otherwise {
  }
END;
UNTIL (Quit);
Cursor_On;
END;
BEGIN
  Star[1] := '*';
  Star[2] := Chr(5);
  Star[0] := Chr(2);
  Power[0] := 1;
  Power[1] := 2;
  Power[3] := 8;
  Assign(Prn,'PRN');
  Rewrite(Prn);
  Printer_Type := Def_Printer;
  Init_Printer := True;
  REPEAT
    Get_Information;
    Process_Section;
    If Ok Then Display_Options;
    WriteIn;
    Write(' Continue With Another Section (<cr> = No) ? ');
    Dispose(Bit_Map);
    Close(Info);
    Readln(Bridge_ID);
UNTIL (Ord(Bridge_ID[0]) = 0) Or (Bridge_ID[1] In ['n','N']);
If (Printer_Type = 1) Or (Printer_Type = 2) Then
  Write(Prn,Esc,'@');
Close(Prn);
END.
Module Utilities;

Function Get_int ( Limit : Integer ) : Integer;

CONST
  Digits = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '0'];

VAR
  i    : Integer4;
  j    : Integer;
  Minus: Boolean;
  Ch   : Char;

BEGIN
  i := 0;
  j := 0;
  Minus := False;
  If Not Eoln Then Read(ch) Else Ch := '?';
  While (Not (Ch In Digits)) And (Ch <> '-') And (Not Eoln) Do
    Read(ch);
  If (Not Eoln) And (Ch = '-') Then
    BEGIN
      Minus := True;
      Read(Ch);
    END;
  IF (Not (Ch In Digits)) And (Not Eoln) Then
    BEGIN
      Minus := False;
      i := Get_int(Limit);
    END
  Else While (Ch In Digits) And (j < Limit) Do
    BEGIN
      i := i * 10 + Ord(Ch) - Ord('0');
      j := j + 1;
      If Eoln Then j := Limit
      Else Read(ch);
    END;
  If Minus Then i := -i;
  If (i <= MaxInt) And (i >= -MaxInt) Then
    Get_int := Retype(Integer,i)
  Else BEGIN
    Writeln(' Integer Quantity Overflow; Value Set To ZERO');
    Writeln(' Value Must Be Between',MaxInt:7,' And',-MaxInt:8);
    Get_int := 0;
  END;
  Readln;
END;
END.

-93-
A.7 SAMPLE PLOTS
IOWA D.O.T.
DELAMINATION MAP

D.O.T. Example Plot

July 1 1984 Delamination at 400 mV
Length = 18 Feet 0 Inches Width = 18 Feet 0 Inches
Total Area = 324.00 Square Feet Percentage Bad = 3.13%

START
IOWA D.O.T.
DELAMINATION MAP

D.O.T. Example Plot

July 1 1984 Delamination at 400 mV

Length = 18 Feet 0 Inches Width = 18 Feet 0 Inches

Total Area = 324.00 Square Feet Percentage Bad = 3.13%
IOWA D.O.T.  
DELAMINATION MAP  

D.O.T. Example Plot  
July 1 1984  Delamination at  400 mV  
Length = 18 Feet  0 Inches  Width = 18 Feet  0 Inches  
Total Area = 324.00 Square Feet  Percentage Bad = 3.13%

<table>
<thead>
<tr>
<th>FEET</th>
<th>1.6</th>
<th>0.0</th>
<th>0.0</th>
<th>0.0</th>
<th>0.0</th>
<th>1.6</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>20.3</td>
<td>1.6</td>
<td>1.6</td>
<td>0.0</td>
<td>0.0</td>
<td>3.1</td>
</tr>
<tr>
<td>12</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>4.7</td>
</tr>
<tr>
<td>8</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>7.8</td>
<td>28.1</td>
<td>0.0</td>
</tr>
<tr>
<td>4</td>
<td>3.1</td>
<td>0.0</td>
<td>0.0</td>
<td>10.9</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>12</td>
<td>15</td>
</tr>
</tbody>
</table>

START
IOWA D.O.T.
DELAMINATION MAP

Very Wide Bridge

6 - 20 -85 Delamination at 400 mV

Length = 12 Feet  6 Inches  Width = 33 Feet  0 Inches

Total Area = 412.50 Square Feet  Percentage Bad = 2.73%
**IOWA D.O.T.**
**DELAMINATION MAP**

**Very Wide Bridge**

6 - 20 - 85  Delamination at 400 mV

Length = 12 Feet 6 Inches  Width = 33 Feet 0 Inches

Total Area = 412.50 Square Feet  Percentage Bad = 2.73%

<table>
<thead>
<tr>
<th>FEET</th>
<th></th>
<th>16</th>
<th></th>
<th>12</th>
<th></th>
<th>8</th>
<th></th>
<th>4</th>
<th></th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>9.4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>75.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>9.4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

START

-100-