RAI-FTR-04-83-01

(5)

# FINAL REPORT

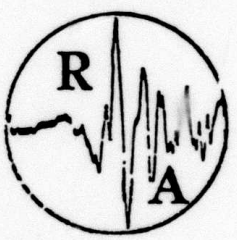
1 OCTOBER 1979 TO 31 DECEMBER 1982

VOLUME I

# ENHANCE AND TEST THE REMOTE SEISMIC TERMINAL

1 JANUARY 1983








RONDOUT ASSOCIATES INCORPORATED
P.O. Box 224
STONE RIDGE, NEW YORK 12484



Sponsored By
Advanced Research Projects Agency (DOD)
ARPA Order No. 3291-31
Monitored By AFOSR Under Contract #F49620-80-C-0021

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|---|
| 1. REPORT NUMBER<br>AFOSR-TR- 83-0433 | 2. GOVT ACCESSION NO. | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle)<br>Enhance and Test the Remote Seismic Terminal | | 5. TYPE OF REPORT & PERIOD COVERED<br>Final Technical<br>1 Oct. 1979 - 31 Dec. 1982<br>6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s)<br>Paul W. Pomeroy, George H. Sutton, and Jerry A. Carter | | 8. CONTRACT OR GRANT NUMBER(s)<br>F49620-80-C-0021 |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS<br>Rondout Associates, Incorporated | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS<br>Element 61101 E<br>Code 0D60<br>ARPA Order 3291-32 |
| 11. CONTROLLING OFFICE NAME AND ADDRESS<br>Air Force Office of Scientific Research (NP)<br>Building 410<br>Bolling Air Force Base, D.C. 20332 | | 12. REPORT DATE<br>1 January 1983<br>13. NUMBER OF PAGES<br>211 |
| 14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)<br>DCASMA, Bridgeport<br>550 South Main Street<br>Stratford, CT 06497 | | 15. SECURITY CLASS. (of this report)<br>Unclassified<br>15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

NA

18. SUPPLEMENTARY NOTES

NA

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Regional Seismic Phases
Lg Waves
Catskill Seismic Array
Wake Island Hydrophone Array
Oceanic Pn and Sn
HARZER
Seismic Discrimination
Yield Determination
Remote Seismic Terminal

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report, which is presented in two volumes as follows:

Volume I entitled "Enhance and Test the Remote Seismic Terminal" describes the terminal system and, in greater detail, the Seismic Recording System add-on to the terminal.

Volume II entitled "The Use of Regional Seismic Waves for Discrimination and Yield Determination" deals with the following topics:

DD FORM 1 JAN 73 1473



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

a. Discrimination Techniques at Regional Distances
b. Yield Determination Using Regional Seismic Waves
c. The Catskill Seismic Array (CSA)
d. The Nevada Test Site explosion HARZER Recorded at CSA
e. Explosion P Waves Recorded at CSA and the Wake Island Hydrophone Array (WHA)
f. Q of the Northwest Pacific Lithosphere
g. The Instrumental Upgrade of WHA to Digital Recording.

## Table of Contents



## List of Figures



## List of Tables

## EXECUTIVE SUMMARY

This volume of the contract Final Report deals with the enhancement and testing of the Remote Seismic Terminal (RST) and an associated Seismic Recording System (SRS). The principal conclusions of this study are:

1. Through the development of the RST and the SRS, the feasibility of a moderately priced, microprocessor based remote seismic terminal with the added capability of digital seismic data collection and storage has been clearly demonstrated.

2. Although the RST system achieved its original goals, a new generation of 16 bit microcomputers has been introduced in the past few months which have significantly increased computing capability. Future seismic work station systems should be based on these machines.

3. Any RST system would benefit from having the same operating system as that used at the Seismic Data Centers (SDC's). Software compatibility problems would thus be minimized.

4. In addition to some hardware problems, the interlocking of the RST and SRS is not desirable and future SRS systems should be 'stand alone' units.

5. Within the SRS, the A/D converter and multiplexer should be capable of operating remotely from the SRS-CPU. Data on the SRS should be stored on ½" magnetic tape for compatibility with the SDC's.

6. The RST-SRS concepts have the potential for significantly altering the mode in which current day seismological research is conducted. Large research centers may become less significant as new seismolgical work stations evolve which can operate effectively far from existing host computers and library facilities.

# RST/SRS DEVELOPMENT

## Introduction

The Remote Seismic Terminal (RST) is a microcomputer based display and communication system designed to support a seismologist in the processing of data from a seismic station; the communication of seismic data to a seismic data center (SDC, e.g., the Center for Seismic Studies, CSS, in Arlington, Virginia);and receipt of data from the SDC at the remote location. The Seismic Recording System (SRS) is an extension of the RST developed to demonstrate the capability to digitally record seismic signals at the remote location; to manipulate that data with the RST; and to transmit that data to the SDC for archiving and/or further processing and return to the RST. During the development of the RST/SRS, emphasis was placed on the use of microprocessors to provide a useful, convenient, and flexible system at moderate cost.

Versions of the RST have been demonstrated at the United Nations Committee on Disarmament Group of Scientific Experts (GSE) meeting in Geneva and at the International Association of Seismology and Physics of the Earth's Interior (IASPEI) conference in London, Canada. Several foreign organizations associated with the GSE have expressed interest in obtaining copies of the system, which would considerably enhance the capabilities for international exchange of seismic data in digital form.

Starting in 1980, the RST was developed by the Defense Advanced Research Projects Agency (DARPA), initially, through a contract with the Massachusetts Institute of Technology Lincoln Laboratory (Lincoln), which in turn funded CAC of Sanford as a subcontractor to do the system design and construction. In April of 1982, RAI replaced Lincoln as the primary contractor, with CAC remaining the subcontractor for design and construction.

Tests of the complete RST/SRS system were conducted in May 1982 between State College, Pennsylvania and the CSS, and in September 1982 between Stone Ridge, New York and the CSS. Seismic signals from Penn State instruments were used as data inputs to the SRS in the earlier tescs and audio-oscillator signals were used at RAI in the later tests. Although the results of the earlier tests were marred by a noisy communication link and a software error in the SRS, these problems were corrected and the later tests at RAI successfuliy demonstrated digital recording using the SRS and reliable communication of digital seismic

data and parameter data between the RST and CSS.

Three prototype RST's were assembled during the development program; each one configured somewhat differently. At present, one unit is at CIRES, University of Colorado and another at NORSAR in Norway. This report is based on the development and testing of a third unit which was delivered to RAI by CAC in September 1982.

The report includes: 1) a description of the RST and SRS systems as currently configured, including a listing of hardware and software components; 2) a description of operating procedures for the RST and SRS systems as currently configured; 3) conclusions and recommendations, indicating advantages and shortcomings of the current systems and possible direction to take to improve future generations of the RST/SRS; and 4) four appendicies which provide details of the development and operation of the RST and SRS and a listing of programs developed for manipulating SRS data.

Within the past few months a number of powerful microcomputer systems based on the MC68000, or equivalent, 16 bit, microprocessor have come on the market (e.g. IBM, Radio Shack, Wycat, Apollo, Sun). For a comparable price, these units can duplicate essentially all of the system functions performed by the RST and, because of their 16 bit (versus 8 bit in the RST) architecture, they possess strikingly greater computing power. Unfortunately, the SRS function does not appear to be easily handled by any of the popular systems.

As mentioned earlier, the current RST/SRS system has demonstrated the capabilities originally envisioned and actually considerably more. However, future RST systems should be based on one of the more popular advanced computer systems in order to take advantage of the greater computing power, availability, service benefits, and economy of a widely used system. The SRS, as presently configured, has some hardware and software shortcomings which should be overcome in any future design. It is most likely that the best approach is to make the SRS essentially independent of the RST except for compatability for data transfer and, where practical, the use of interchangeable hardware components (e.g. tape drives).

System Description

In this section, we give a brief description of the system and the functions of its components. Detailed descriptions of the RST and SRS hardware and software are included in the appendicies. The components and their manufacturers are listed in Appendix II of the RST User Guide (Appendix B) for the RST and in Table I for the SRS.

The RST is a communications and display station designed to support a seismologist in the processing of digital seismic data. It consists of a microcomputer with a CP/M (CP/M is a registered trademark of Digital Research) operating system, a 26 Mbyte hard disk, twin 8 inch floppy disks, an alphanumeric terminal, a graphics display, an impact printer, a digitizing tablet, and a modem (Figure 1).

In the communications mode, it can transmit and receive both parametric and waveform data via the modem to a SDC or another RST. It can also obtain digital seismic data from a SRS at the remote station which can then be sent to a SDC.

In the display mode, the RST operates as a stand-alone single-user microcomputer with graphics capability. The graphics display and digitizing tablet allow the user to display waveforms, determine amplitudes and periods, and mark the waveform with arrival information. Hard copies of the waveforms may be obtained by using the printer. If the user has analog data, the digitizing pad may be used to put the data in digital form for analysis and transmission to a SDC.

The SRS in its present configuration records three passbands of a three component seismic station on ¼ inch digital tape. The nine analog inputs are multiplexed and digitized with a 12 bit A/D converter which is syncronized with a satellite clock. Two ¼ inch tapes are available for recording enabling the user to access the data on one tape while incoming data is being recorded on the other tape. The SRS is dependent upon the RST to initialize its operation, but once started, can stand alone.

Because the RST is a single user system, the SRS must have its own CPU so that the RST and SRS may run simultaneously. The SRS actually contains two CPU's, one to perform the seismic processing and the other to control the input/output processing (Figure 2). A detailed description of the SRS is given in Appendix C.

System Operation

Part of RAI's task in this project was to confirm that the design goals of the RST/SRS system were met. To this end we have recorded data on the SRS; checked its timing; sent data to and received data from the Center for Seismic Studies in Arlington, Virginia using the RST; displayed and manipulated that data; and obtained hard copies of it for presentation in this report.

The procedures for operating the RST are well documented in Appendix B of this report, Remote Seismic Terminal User Guide. Included in this guide are instructions on how to start the machine; documentation of the software used to cummunicate with a SDC and to display or digitize waveforms; a brief introduction to the CP/M operating system; and a list of the hardware that comprises the RST as currently configured.

Although the SRS has its own CPU, its operation must be initialized through the RST. This has the advantages of not needing duplicate hardware (e.g. tape drives, terminals, etc.) and having easy access to the SRS data but has the disadvantage of being a one-of-a-kind system. To start the SRS from the RST, 'PROTIME' is typed to check the SRS internal clock. If the time returned is not satisfactory then the clock must be reset. When the time is correct, 'SETGET' is used to initialize the buffers and then 'XIOPUT' to begin the recording. Before recording, XIOPUT rewinds the tapes. Each of the two ¼ inch tapes holds eight hours of data on four tracks thus allowing 16 continuous hours of recording before having to replace tapes. At the end of each track, the tape is rewound and recording begins on the next track. While the tape is being rewound, incoming data is stored in a buffer where it can be saved to place at the beginning of the next track. Once the recording has begun, the RST is no longer needed to operate the SRS and may be used for other purposes or even turned off.

There have been considerable difficulties in keeping the SRS running for any length of time. We suspect that this is due to a bad clock board. We are currently on our third clock board and it still is not working properly. As a new clock board would require a certain amount of redesign, no attempt has been made to use another manufacturer's board.

The multiplexing and demultiplexing capabilities of the SRS were checked by recording a triangle wave at three different amplitudes on the high frequency band channels. The signals had to be digitized and multiplexed for recording and then demultiplexed for display. The three waveforms are shown in Figure 3 . Relative amplitudes for the waveforms were 1, 8/9, and 7/9 at input and after multiplexing, digitizing, and demultiplexing had not changed.

The resolution of the RST plotting routine was tested by plotting 6.5 Hz triangle waves (Figure 4). Although we expected and observed poor resolution near the peaks and troughs of the triangle waves due to aliasing, we felt that a better picture could have been obtained along the straight portion of the waveforms through interpolation. However, this is mostly a matter of preference and is only mentioned for possible consideration in future software upgrades.

Timing was one of the most significant problems that we ran across during our testing of the RST/SRS system. The goal was to implement a scheme that would use the satellite clock time as an input to update the SRS time being written in the data headers. This was never achieved and an alternate method was necessary in order to obtain time corrections to the data. Figure 5 demonstrates this method. The satellite IRIG-H time code is input as one of the short period inputs and recorded. This time is then displayed and printed out along with the time determined by the SRS for the beginning of the data. The IRIG-H time code is read and compared to the printed time to obtain a time correction. This method is unsatisfactory for several reasons. First, the IRIG-H time code is difficult to read accurately and can only be read to 0.025 seconds at best due to the 40 sample/second sample rate. The second problem with this method is that it must be done at the beginning and end of each continuous data set to obtain linear clock drift. If the machine should stop for any reason, clock drift could not be determined and an assumed drift would have to be used instead. This is a major problem and should be rectified in any future generations of the SRS. Actually, until recently, with the advent of reliable standard time signals, a procedure similar to this was standard practice.

The commands used for manipulating the SRS data using the RST are given in Appendix D. To enter these commands, the user must use the FORTH operating system which we feel is an unfortunate complication. However, switching from CP/M to FORTH and back is simple and relatively user-transparent when using these commands.

In addition to the commands given in Appendix D, there are a few additional commands which allow data transfer to the SDC at 1200 baud rather than at 300 baud as described in the RST User's Guide (Appendix B).

To set up a modem operation, get into FORTH by typing 'RONWORK' then type '1 LOAD' then 'DUMTER1200'. At that point anything typed goes out the modem, anything received by the modem comes up on the screen; you get back to FORTH by keying 'ESC'--the escape key. 'SEND-MEDIUM' and 'SEND-SHORT' send respectively the contents of the MEDIUM-BUFFER or the TAPE-BUFFER out the modem. Thus to send short period data saved on some screen file one would do an 'n TAPE-BUFFER-FETCH' followed by a 'SEND-SHORT'. Here is a typical sequence: type DUMTER1200 and then dial up the center, log in and enter the file you wish to put the data in through the editor, give the command a--for append and exit the modem program with 'ESC', send the data by typing SEND-SHORT, get back to the modem program with a 'DUMTER1200', give ed a '.' and a 'w' and a 'q' to save the file, log out and return to FORTH with 'ESC'.

When it is desired to bring data up from the SDC and display it on the RST, one types 'FORTH' at the CP/M level. Once the FORTH greeting message appears, 'USING NEWADD' gets access to this screen file. A '1 LOAD' then sets up a modem communication package similar to the one on FORTH SCR in that it is activated by 'DUMTER1200' but different in that a core buffer is supplied for retaining data received by the modem. The first appearance of a left curly bracket '{' in the incoming data stream causes all following data to be stored in the buffer until a right curly bracket '}' is received. This task is performed at the CSS in Arlington by the program /c/rondout/sutton/newrat filename. One then gets out of the modem program by hitting the 'ESC' key, and can dump the buffer to the Scion screen with the word 'SHOWEM'. Return to the modem program by again typing 'DUMTER1200' and repeat ad lib.

## Conclusions and Recommendations

The main goal of demonstrating the feasability of a remote seismic station with the added capability of digital seismic data collection and storage has been achieved through the development of the RST and the SRS. The experience gained through working with this system leads to suggestions for future changes and development.

As a communications and display station, the RST has more than fulfilled its design goals. Since its development however, several microcomputer systems based on the new 16 bit microprocessors have been marketed which can duplicate the functions of the slower, less powerful eight bit RST. The cost of these new machines is comparable to that of the RST and there are several advantages to buying an off-the-shelf model. First, an off-the-shelf machine will be more generally available to anyone who wishes to purchase one. In its present configuration, a RST would be difficult to obtain and maintain. A certain degree of versitility will also be achieved in that for many systems, peripherals are manufactured by other companies specifically for use with the popular models. In addition, most of the systems level software has already been developed for their interconnection with the CPU. Perhaps the most important advantage is that a maintenance program will already be in place making system repairs much easier and less expensive.

Widespread appeal of the RST will probably only come if it is both pow r-ful and easy to use. Because of this, there is something to be said for making the RST operating system the same as that used at the SDC's. If possible, this would simplify the use of the RST in conjunction with the SDC.

There are several problems with the present configuration of the SRS. The most serious problem is that the multiplexer and A/D converter must be close to the seismometers which requires that the SRS and RST be in inconvenient locations. To make the SRS useful, the A/D converter should be sepa.ate from the recording station.

Although the ¼ inch tape drives are compact and cassettes are simple to use, they limit data transfer to the SDC to phone line communication. If any large volume of data is to be transferred, this would be a time consuming and expensive process. Also, the SRS requires three cassettes per day if operated in a continuous mode. We recommend that future SRS designs incorporate ½ inch tape drives for easy exchange of data with the SDC.

As with the RST, the SRS is hard to get and difficult to maintain. The clock board has not performed reliably and would require extensive modification to make it work. We thus recommend that the SRS be made independent of the RST. We feel that the advantages of shared peripherals and easy data access are outweighed by the specialized software and hardware required to make it work. By using ½ inch tape drives on both the SRS and the RST, data would still be easy to access and with clever design, certain hardware components such as tape drives and terminals could be interchangeable between the RST and the SRS. In addition, most digital seismic stations already in existance record their data on ½ inch tapes so, in a sense, the SRS already exists at many sites. By adding an RST with a ½ inch tape drive, the user would have easy access to a powerful processing tool at a reasonable cost.

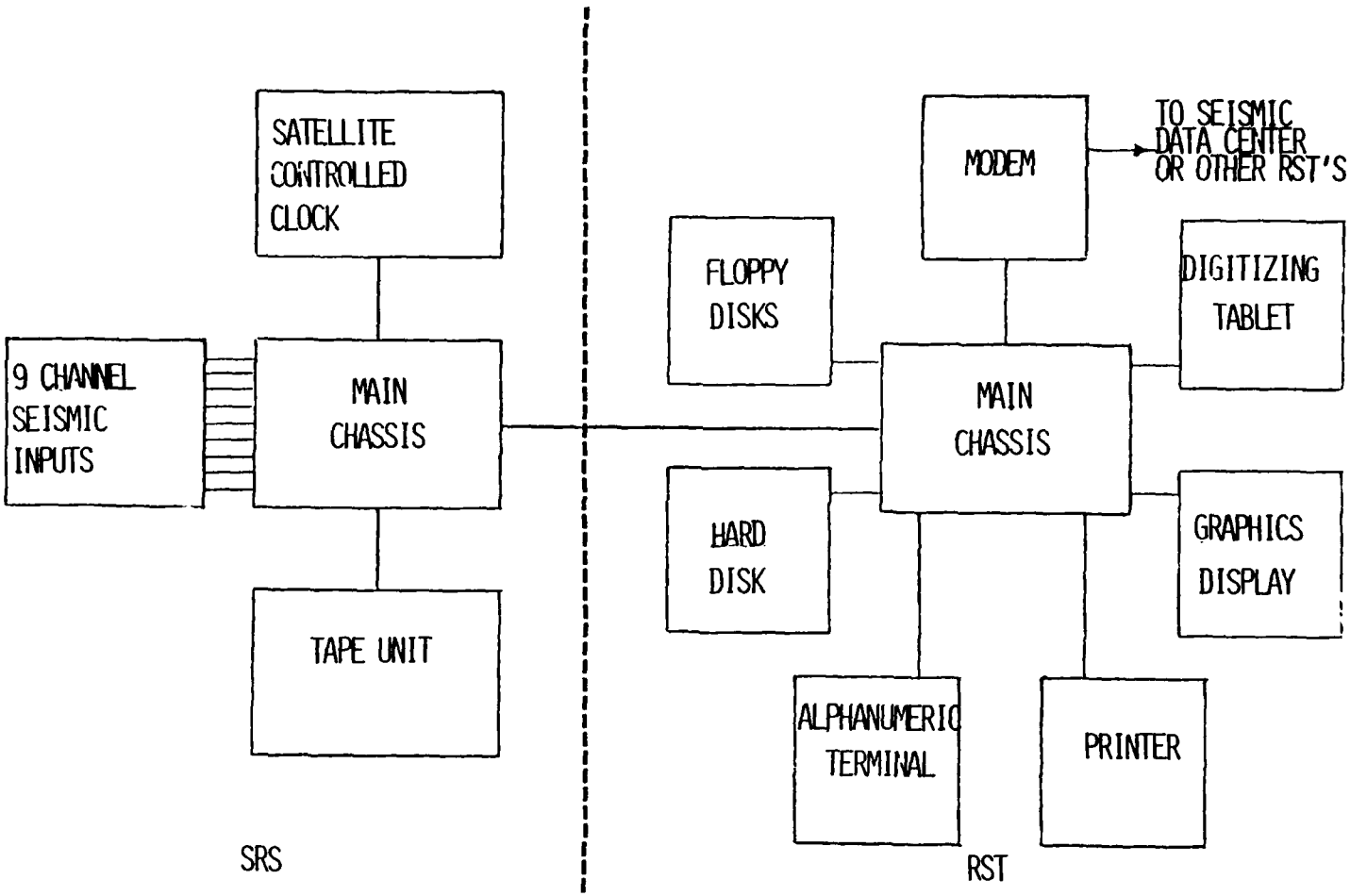



Figure 1. Simplified block diagram of Seismic Recording System (SRS) and the Remote Seismic Terminal (RST).

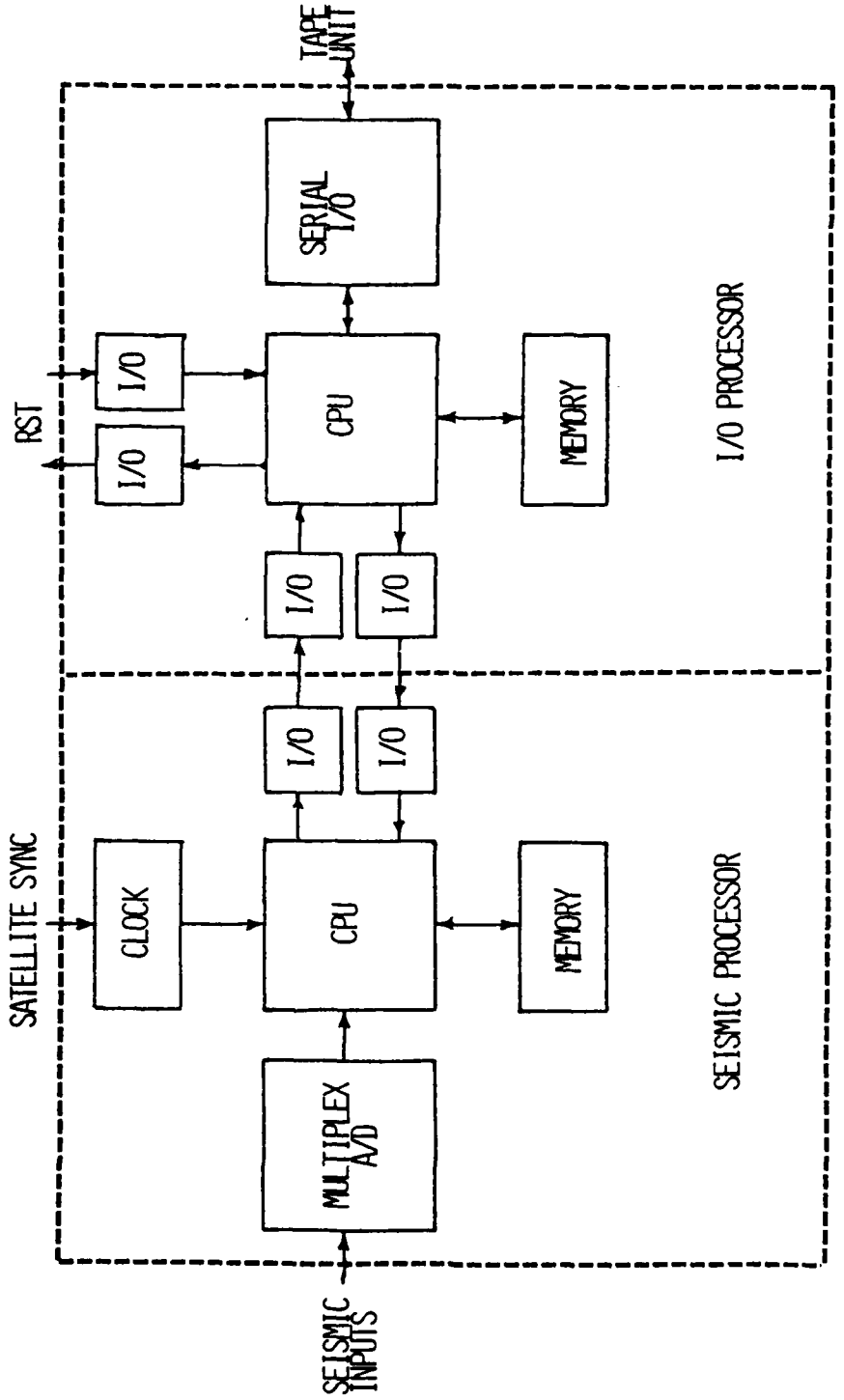



Figure 2. SRS main chassis block diagram.

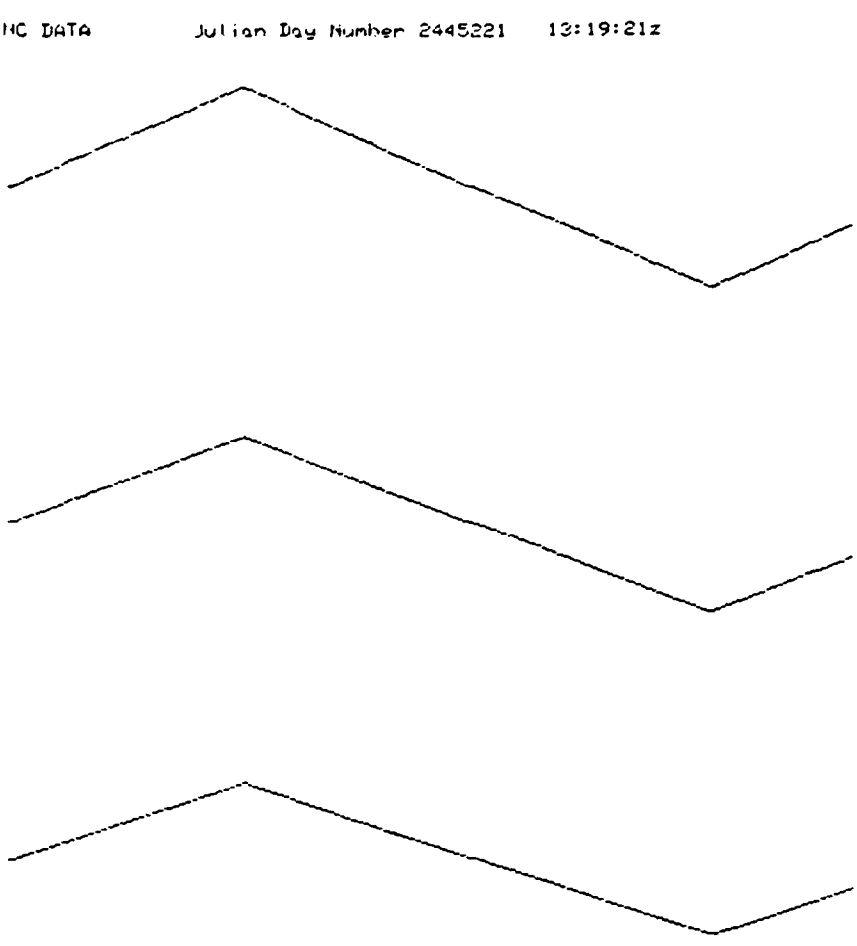
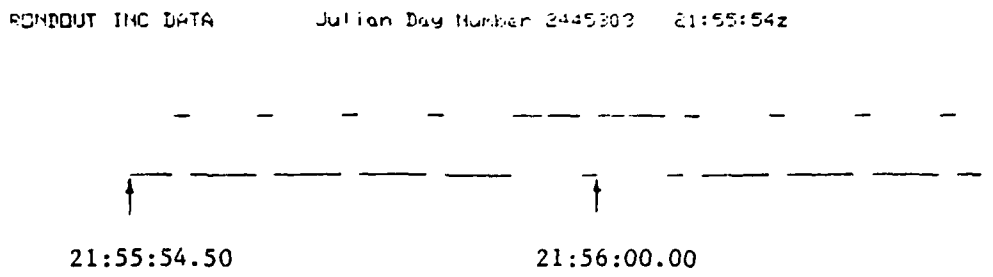



Figure 3. A 0.1 Hz triangle wave recorded at three different amplitudes. 10 seconds of data recorded at 40 samp/sec are displayed. The amplitudes are 1, 8/9, and 7/9 from top to bottom.


Figure 4. A 6.3 Hz triangle wave recorded at 40 samp/sec demonstrates the limits of resolution due to the digitizing rate. Amplitudes are 1, 8/9, and 7/9 from top to bottom.

Satellite IRIG - H time code recorded by the SRS




SRS time assigned to the beginning of the above data

```
6758: 524F 4E44 4F55 5420 494E 4320 4441 5441  RONDOUT INC DATA
6768: CC60 0154 5521 0353 0000 0000 8100 BBAA  .'.TU!.S........
6778: 4000 0000 0000 00AA 0A08 F907 FB07 0A08  @...............
6788: F907 FD07 0B08 F807 FE07 0A08 F907 FB07  ................
```

Julian day 5303

Time 21:55:54.0160

Figure 5. Time corrections are made by recording the Satellite IRIG code and comparing that time to the SRS time associated with the beginning of the block. In this example the SRS time is 0.48 sec slow.

TABLE I

SRS HARDWARE

| DESCRIPTION | |
|---|---|
| MAINFRAME: | DESIGNED BY CAC OF SANFORD |
| CPU BOARD | SSM MICROCOMPUTER PRODUCTS<br>CB2 Z-80 CPU BOARD S-100 BUS |
| MEMORY | MORROW DESIGNS<br>24K MEMORY MASTER STATIC RAM |
| INTERFACE BOARDS | MORROW DESIGNS<br>SWITCHBOARD |
| INTERNAL CLOCK | MOUNTAIN COMPUTER INC.<br>10,000 DAY CLOCK |
| 12 BIT A/D CONVERTER | DUAL SYSTEMS CONTROL CORP.<br>AIM-12 |
| ¼" TAPE DRIVE CONTROLLER | ALLOY ENGINEERING COMPANY<br>DRS-232 CONTROLLER INTERFACE |
| OTHER HARDWARE: | |
| ¼" TAPE DRIVE (DUAL) | DATA ELECTRONICS INC.<br>3400S2 |
| SATELLITE CLOCK | KINEMETRICS, TRUE TIME DIVISION<br>468-DC |

# APPENDIX A

APPENDIX A

Massachusetts Institute of Technology
Lincoln Laboratory

# REMOTE SEISMIC TERMINAL*

by

Alvin G. Gann
MIT Lincoln Laboratory
Applied Seismology Group
42 Carleton Street
Cambridge, MA 02142

and

Richard M. Morcney
CAC of Sanford
50 Jagger Mill Road
Sanford, ME 04073

*This work was sponsored by the Defense Advanced Research Projects Agency.

The U. S. Government assumes no responsibility for the information presented.

# REMOTE SEISMIC TERMINAL

## Introduction

The Remote Seismic Terminal (RST) is a microcomputer based display and communication system designed to support the analyst faced with the processing of data from a seismic station, the communication of seismic data to the Seismic Data Center and the receipt of the Seismic Data Center bulletins at a remote location. It is an alternative to the manual processing of seismic data and will provide computer assisted communication with the Seismic Data Center. In its basic form, the RST will provide support to the analysis of digital waveform data and will partially automate the measurement and extraction of seismic parameters. In expanded versions, the RST can digitize waveform data and provide a variety of tools for the analysis of digital data. An overview of the design of the RST and a discussion of its capabilities are given. The configuration described below is that demonstrated for the U.N. Committee on Disarmament Group of Scientific Experts at their meeting in February, 1981. The enhancements discussed are being implemented at the time of the writing of this paper.

The RST is a modular microcomputer and peripheral system plus a modular suite of software designed to provide communications with the Seismic Data Center, waveform display capability and support for the creation of International Exchange of Seismic Data (IESD) level I (parametric data) data reports. The software suite includes the CP/M* operating system with its file management system, editor, compilers, etc. This supports the

* CP/M is a registered trademark of Digital Research

preparation of level I reports and the printing of bulletins. It also supports the generation of analysis programs and the analysis of data. The communications function is supported by specialized programs which observe a communications discipline and allow the transfer of files of data to and from the Data Center. Finally the analysis of waveform data is supported by a display program which allows the display of and measurement from waveform data stored on the local floppy disks.

The RST has three different "operating modes". First, it is a normal CP/M-based microcomputer with all of the CP/M features and utilities such as editor, assembler, debugger, etc. The RST has extra features such as a BASIC compiler and a complete relocatable macro assembler package, Microsoft M80.

Next, the RST can be configured to be an intelligent terminal for use in communicating with the Seismic Data Center via a modem. This operating mode is entered from the CP/M mode by activating an appropriate software package.

Finally, the RST has a graphics display mode, again entered from the CP/M mode via a call to DISP. DISP is a self-contained software package which prompts the operator to perform various waveform display and analysis operations. DISP takes commands from the terminal keyboard and allows the use of a graphics cursor controlled by the operator to extract amplitude and period measurements from the waveforms.

## Hardware Design

The RST consists of a chassis containing a microcomputer and associated memory and peripheral boards, an alphanumeric terminal, a graphics

terminal, a floppy disk and a modem for communications. The main chassis provides power supplies and the means for interconnection of the other boards via the standard S-100 bus.

In the main chassis there are four boards plus the memory boards. The first is a California Computer Systems (CCS) CPU board, which provides a Z80 CPU and a programmable serial interface port. Second, the Morrow disk controller board operates the floppy disks and also provides a serial port that is used to communicate with the video terminal for system control. Third, the Scion graphics display is controlled by a separate board which has its own Z80 and memory connected to an internal bus. Communication with the system is via I/O ports, and the board produces video for driving the monitor. The final mainframe board is a Morrow Designs "Switchboard" which incorporates two serial and four parallel ports plus a status port. There are two memory boards in the RST. A 16K byte static RAM is used for the stack and a 64K byte dynamic ram board with the overlapping 16K bytes disabled, leaving 48K bytes active.

The rest of the hardware consists of independently packaged units interconnected with cables. The cursor control box is simply a collection of on/off switches, suitably debounced and connected to a parallel port of the system. This controls, via software, the graphics cursor. The graphics display is simply a video monitor; it accepts standard signals via its coax input connector and converts them to light on the screen. The alphanumeric terminal is a standard Hazeltine 1421, which communicates with the system through a serial I/O port. A Racal Vadic modem is used. It is capable of 300 or 1200 baud operation, the latter with the Bell 212A (industry standard) protocol or with the older Vadic protocol. The dual

floppy disk drive is a Morrow Designs Discus 2+2D and is capable of reading both single and double density diskettes.

## Options and Extensions

A number of different enhancements have been suggested and are currently being implemented. They group into five categories:

CPU Arithmetic Enhancement
Hard Disk
Software Enhancements including
    Hard Copy of Graphics Display
    Maps
    Communications Protocol and Lines
Plotting Board
Real-Time Operation including
    Direct Digital Data Acquisition from a Seismometer

## Arithmetic Enhancement

The current RST has limited arithmetic capability, as one would expect of an eight-bit microprocessor. There is no currently supported task where this weakness is a problem. The only area where the slow arithmetic is noticeable is in the data scaling in the display routine. The data writing speed is currently limited by the speed of the CPU's scaling the output data. The slow arithmetic could become a problem if one wished to perform data filtering, for example via Fourier transforms, or other "computing" as opposed to "data processing" chores. After some consideration of alternatives, the best available arithmetically-enhanced hardware is the so-called "p Engine" manufactured by Western Digital. The design of the RST requires that all additions use the S-100 standard bus. Insofar as S-100 boards "on the market" are concerned the Digi-comp p Engine is the best alternative. To use this, one has to use the UCSD Pascal system because "p code" is the Pascal compiler intermediate output, and it is that code that the chip is

designed to support. A Fortran compiler is also available to use with the system. The use of the Pascal system adds a fourth mode of operation which gives an alternative in operation and program development to the use of the CP/M system. A utility package under the Pascal system gives an alternative to many of the utilities under CP/M and may largely supplant the use of CP/M which will remain available however.

Hard Disk

The addition of a hard disk with the greatly increased on-line storage capacity over the floppies and with increased performance from the faster access of the hard disk is under way. The operating system, user programs and data will be stored on the hard disk. The floppies will be retained to use in program and data exchange and for making back-up copies of programs and critical data.

Software Enhancements

There are a number of software enhancements planned for the RST. These will be acquired where possible from external vendors and implemented where necessary especially for the RST as time and personnel permit. Included in this category are additional utilities such as compilers, file manipulation routines, etc. Another software package will give hard copy capability for waveform data. This involves conversion of the waveform data to codes suitable for plotting on the line printer in the graphics mode.

Another software enhancement is the addition of map plotting capability. This requires the establishment of a map data base (which is readily available) and the implementation of algorithms which will select, scale

and plot selected data and map locations. Another planned software enhancement is an improved method for transferring waveform data over (potentially noisy) communications channels. This software will provide for automatic error control and flow control to provide reliable transfer of waveform data. This is to replace the current method of waveform data transfer which requires extensive operator involvement.

Plotting Board

The RST can support a manual digitizer. This hardware and software package will allow the manual input of analog waveform data from paper records. This is suitable for digitizing small amounts of waveform data, especially the IESD level II data, selected small segements of waveform data of special interest.

Real-time Operation

The final suggested enhancement is being studied, but is not currently being implemented. This is the real-time operation of the RST in a simultaneous data collection and analysis mode.

Everyone asks "Can this thing be hooked up to a seismometer?". In a sense the answer is "sure, easily", because for $500 a very high-quality 16-channel A/D converter board can be purchased and dropped into the mainframe with no more work than making the connections to the outside world. Some software would be required to exercise the board, but for simply taking a reading, this would be trivial.

The trouble is this simple solution does nothing for the basic limitation which is: the RST is able to do only one thing at a time. One of the

currently available modes must be selected to the exclusion of the others at any given time. Adding an A/D converter would allow the recording of seismic data to the exclusion of all other uses. There would be no support for communication or analysis while data were being recorded.

There is another problem. At 250 bytes/sec a 1.2 megabyte diskette fills in about an hour and a 26 megabyte hard disk in about a day. Thus, some manual intervention would be needed to perform continuous recording, either once an hour (to reload diskettes) or once a day (to process the previous day's worth of data). Note that in the last case we must have two systems (one taking data and one processing data). To top all this off, to retain the seismograms indefinitely, implies magnetic tapes (over one reel per day) or one huge pile of diskettes (24 per day).

The best approach to this problem, however, is a rather unconventional solution made available by the low cost of adding another computer. A mainframe plus CPU plus memory, the barest of computers, costs only about $1,000. Adding a few parallel I/O ports ($250) would enable this "satellite" computer to communicate with the main RST, and it would then be ready to take over any single task, here, specifically that of recording the waveform data. The A/D converter and the the disk controller board could be placed in the satellite. The RST would then perform all disk operations by transferring a command word to the satellite, followed by the block of data, using a parallel port. Thereafter the RST would be free to do "other things" while the satellite accomplished the actual transfer of data to the disk.

The idea here is simple: if a device does not have enough smarts to perform unattended, dedicate a new computer to running it! Such a solution

would have been unthinkable a few years ago, but today the cost of hardware is so low that it seems perfectly realistic.

Alternative Configurations

At various demonstrations of the RST, considerable interest was expressed in the extent to which the demonstration RST configuration could be "cut down". In the following table five systems of increasing complexity are given with their costs. Three columns of cost data are included: the first column is the incremental cost of adding the capability described; the second column is the accumulation of these increments, and represents the cost of a system at this level in a form that allows future expansion; the third column shows the cost of an all-in-one system at the desired level--such a system will be cheaper but will have no growth capability.

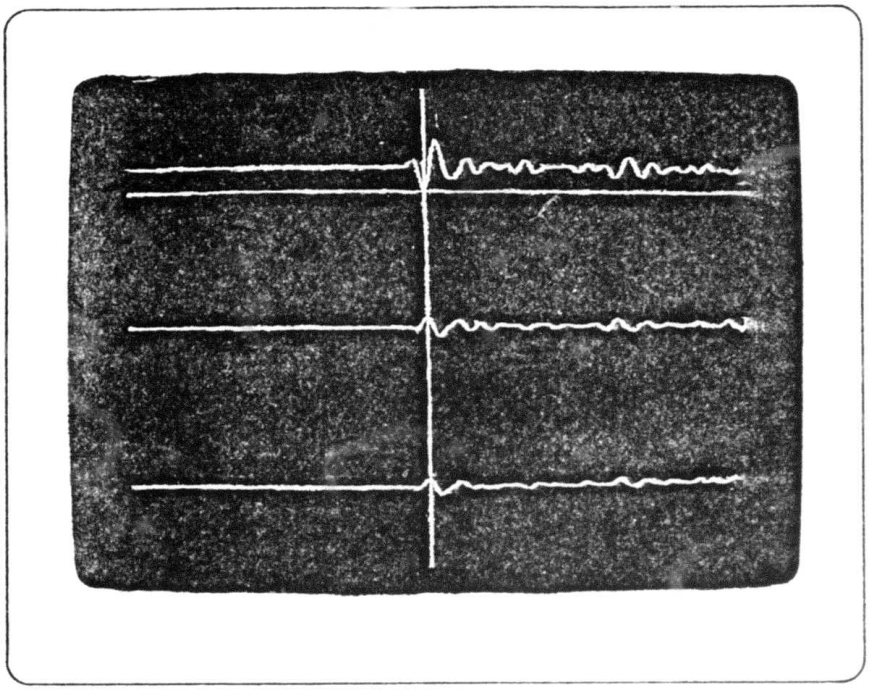
The first system allows simple communication of text, it has no computing or graphics capability. An even smaller system is possible if one dispenses with the lineprinter, but nobody seems willing to do without hard copy. The next increment adds a computing capability (hardware and software) and produces a viable small computer. Addition of high-resolution graphics comes next; the (5,000) in the third column is intended to indicate that generally the preceding system will have some graphics capability, perhaps 190X280 or better. This level is that of the present RST. The next two increments provide respectively a realistic data base capacity with analog data reduction capability, and a true seismic station.

| Level of Capability | incre-mental cost | total cost | minimum cost alternative |
|---|---|---|---|
| Video Terminal, Modem, Printer | 3,000 | 3,000 | 2,000 |
| plus Disk, CPU, Compiler | 4,500 | 7,500 | 5,000 |
| plus Graphics Display & Cursor * | 2,500 | 10,000 | (5,000) † |
| plus Plotter, Hard Disk ** | 5,000 | 15,000 | (9,000) † |
| plus Seismometer Hookup | 21,000 | 36,000 | N/A |

* Model demonstrated at the Group of Scientific Experts Meeting
† Limited resolution graphics, limited arithmetic capability
** Current level of capability

APPENDIX B

# Remote Seismic Terminal User Guide



Applied Seismology Group
MIT/Lincoln Laboratory

# REMOTE SEISMIC TERMINAL USER GUIDE

Applied Seismology Group
MIT/Lincoln Laboratory

Lincoln Manual 131
31 December 1981



The work reported in this document was performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology. This research is a part of Project Vela Uniform, which is sponsored by the Defense Advanced Research Projects Agency under Air Force Contract F19628-80-C-0002 (ARPA Order 512).



Massachusetts Institute of Technology
Lincoln Laboratory
Lexington, Massachusetts 02173

CONTENTS

# Remote Seismic Terminal
# User Guide

MIT/Lincoln Laboratory
Applied Seismology Group

The Remote Seismic Terminal (RST) is a microcomputer based system for analyzing and communicating Seismic Data Center information.

In its basic form the RST communicates with the Seismic Data Center to:

- receive messages from the SDC and other RSTs
- receive bulletin data
- receive and transmit parametric data

Locally it will:

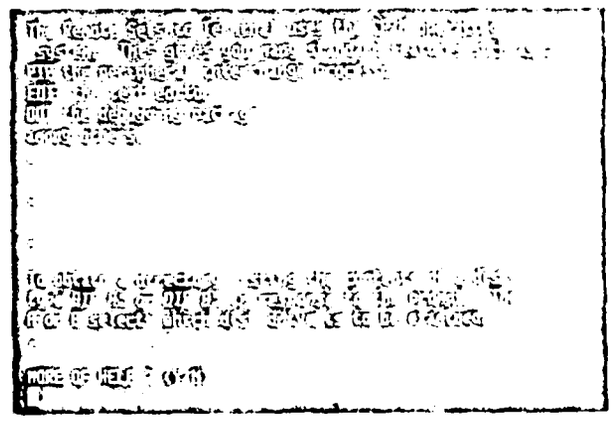
- analyze digital waveform data
- prepare parametric data
- edit parametric data
- digitize hard copy waveforms
- produce hard copy alphanumerics and waveforms

The RST operates in 3 ways:

1. Processing:

   As a microcomputer running under the CP/M operating system the RST offers such features as: editor, file management system, assembler, debugger. Additional software includes a BASIC compiler and a complete relocatable macro assembler package, Microsoft MACRO-80 (c). It also offers PASCAL and FORTH (each with its own operating system).

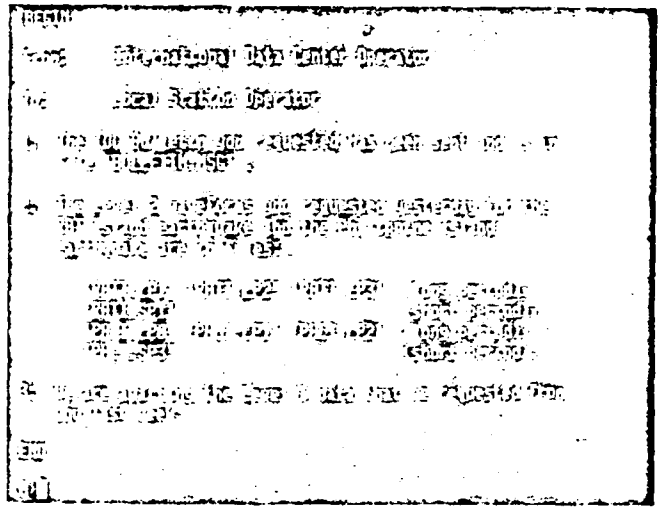
113729 S



2. Communications:

The RST communicates with the Seismic Data Center or another RST via a modem. In this mode it sends and receives parametric (level 1) data, bulletin data, and other alphanumeric information.

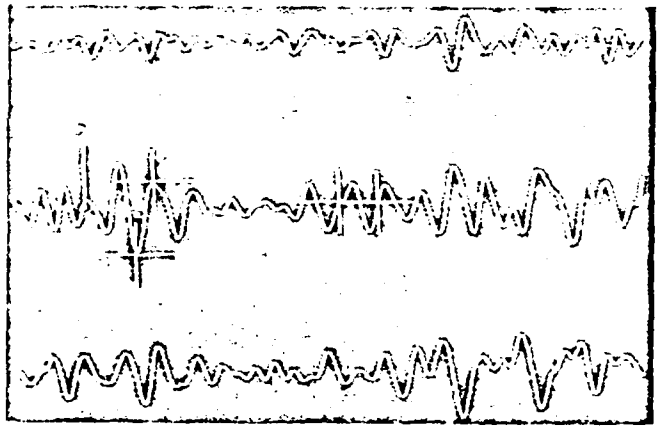
113730 S



3. Graphics:

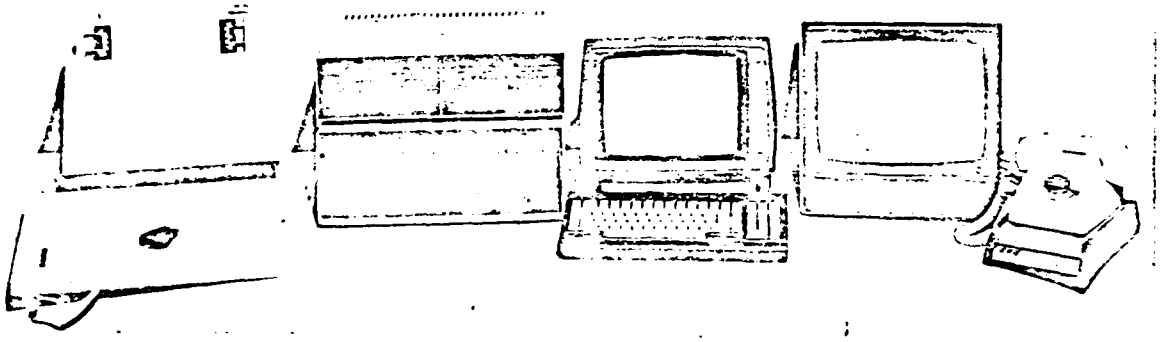
As a graphics display device the RST does various waveform display and analysis operations such as measuring waveform amplitude and period. The RST can produce hard copy of the graphics display at any time, and supports a data tablet for manually digitizing hard copy waveforms.

113731-S



The RST consists of a main chassis which holds a microcomputer and memory and peripheral boards; an alphanumeric terminal; a graphics terminal; twin 8-inch floppy drives; a hard disk for additional on-line memory; a printer; and a data tablet for digitizing waveforms. (See Appendix II.)

113732-S



The future of the RST will bring enhancements in the following areas:

CPU arithmetic
Maps
Communications protocol and lines
Real-Time operation
(including direct digital data acquisition from a seismometer)

This guide is an orientation to the Remote Seismic Terminal (RST). For more information on its operating system (CP/M) see Appendix I.

## 1. Start up

a. A power strip supplies power to the components of the RST.

To turn on the RST simply turn the power strip switch on.

PRECAUTION: NEVER TURN THE COMPUTER ON OR OFF WITH DISKETTES IN THE DRIVE.

NEVER LEAVE THE SYSTEM ALONE WITH DISKETTES IN THE DRIVE.

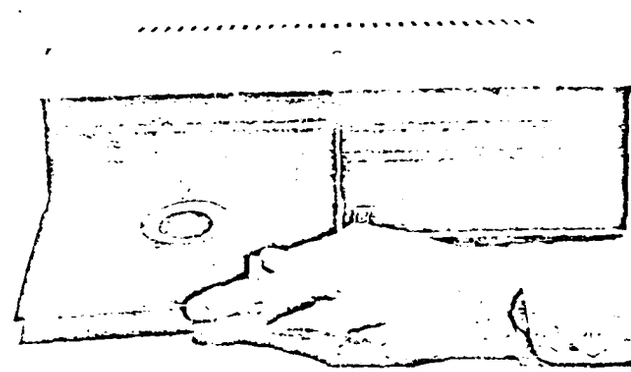
(Turning the system on or off with diskettes in the drives will not hurt the computer, but it could foul up the diskettes.)

b. Once the system is turned on, insert the diskettes. (The main label should face up as you insert the diskette.)

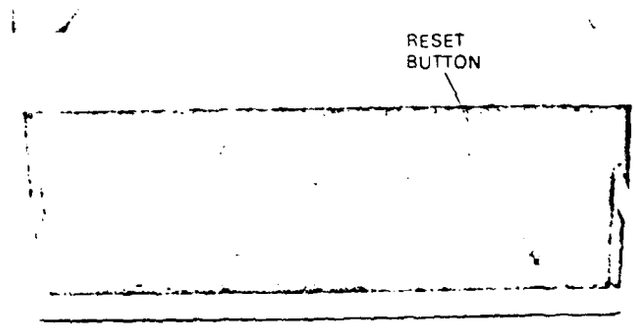
c. Push the COMPUTER RESET button (located on the front of the main frame). This starts the system from the program it reads on drive A.

d. Type HELP for some simple directions and a guide to the different RST operations.

113733 S



113734 S

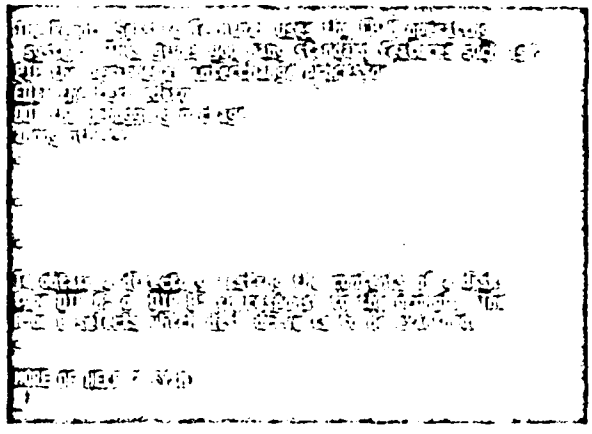


When the system starts up it is running under the CP/M operating system and displays the CP/M prompt, A>.

The system automatically assigns floppy drive A as the default drive. This means that unless you specify otherwise, the system will look for files only on drive A. (Appendix I: CP/M Operations explains this further.) B is on the other floppy drive. The hard disk has directories, C, D, and E.

If the system is new to you, ignore the rest of the switches. A safe approach would be to leave any and all switches alone if you don't know what they do

113729 S



## 2. Communications — CENTRAL and TERM1200

CENTRAL is the program for communicating via the modem with the Seismic Data Center or with another RST.

TERM1200 works at 1200 baud, but does not offer the intelligent mode features of CENTRAL.

### 2.1. Establishing the Connection

a. Set the modem switch off HS for 300 baud and make sure the modem cable is connected to the 300-baud input on the back of the mainframe — there are different connections for 300 and 1200.

b. Set the VO-DA switch to VO (voice).

c. Type CENTRAL and hit RETURN.

The system responds by filling the screen with information including TIP phone numbers and an operating manual of its own. Follow the instructions on the screen to connect to the central site.

d. Pick up the handset from the telephone and pull the cradle post all the way up.

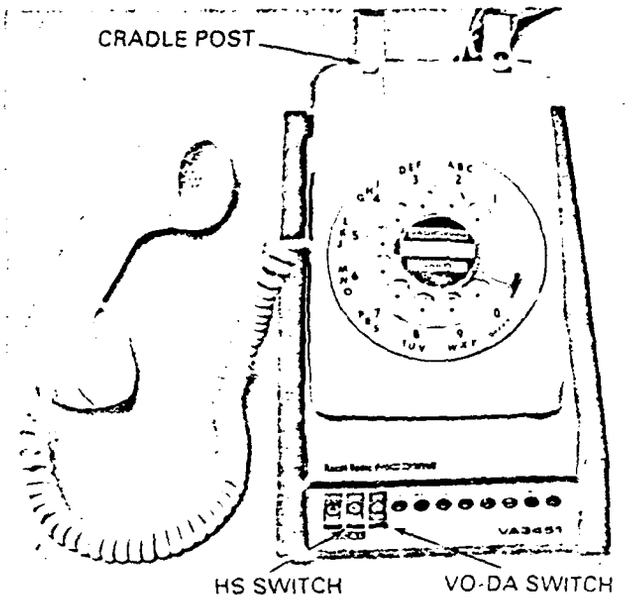
e. Dial the host computer. When the beep sounds, flip the VO-DA switch to DA and hang up the telephone.

f. Once the connection is made, hit RETURN again.

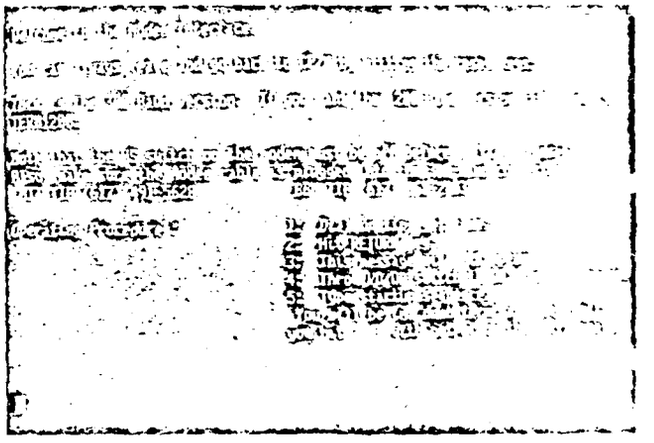
The system is now in a "dumb terminal" mode and displays the host system login message.

You can login, get mail, and so forth just as on your usual terminal.

113736-S




113737-S



g. To use TERM1200 establish the connection in the same manner, but set the HS switch to HS (high speed) and connect the modem cable to the 1200 connection on the back of the mainframe. TERM1200 is good for sending and receiving mail. It displays on the screen and does not send to disk. Use ESC to toggle the printer on and off. It is initially off.

## 2.2. Disconnecting

a Type CTRL *d* to log off the central system.

b Hang up the phone.

c Hit the reset button to reboot the system.

## 2.3. Data Transmission — Intelligent Mode

The intelligent mode is part of the CENTRAL communications program. Use it to transmit and receive data files from the SDC or from another RST.

The terminal goes into intelligent mode when you start a line by typing | (a vertical bar) which displays a ~ and advises the terminal that the rest of the line is a command and not a message to be sent to the central site until you type a RETURN.

Under intelligent mode:

Typed characters appear on the video terminal screen, whereas in dumb mode they appear only on the graphics display screen.

Intelligent mode uses a composing line: nothing is sent out to the central site until you type a RETURN. Backspace deletes a character at a time and CTRL U erases the whole line.

In most cases the composing line appears at the bottom of the screen. When you hit RETURN the RST sends the line to the host system which usually echoes the line back to the RST terminal, where it appears at the top of the screen — to the user this looks as though the line jumps from the bottom of the screen to the top.

### 2.3.1. Intelligent Mode Commands

Remember, begin all commands with | (vertical bar).

#### 2.3.1.1. Printouts

|PRINTON Turns on the printer. As each line scrolls off the top of the screen the line printer prints it.

|NOPRINT Turns off printing started by PRINTON.

|FLUSH Flushes the screen. It pushes everything through the top of the screen — like receiving 22 carriage returns. It turns off the printer when finished printing.

Examples:

a. To print the contents of the screen on the printer:

   1. Type |PRINTON and hit RETURN.
   2. Type |FLUSH and hit RETURN.

b. To print an incoming message:

   1. Type: |FLUSH and hit RETURN. (The FLUSH clears the screen so that nothing "old" appears in the printout.)
   2. Type |PRINTON and hit RETURN.
   3. Once the message is printed, type |NOPRINT and hit RETURN.

### 2.3.2. Receiving Messages or Files — OUTDISK

Use OUTDISK to receive messages or files. It opens a file for writing on drive B.

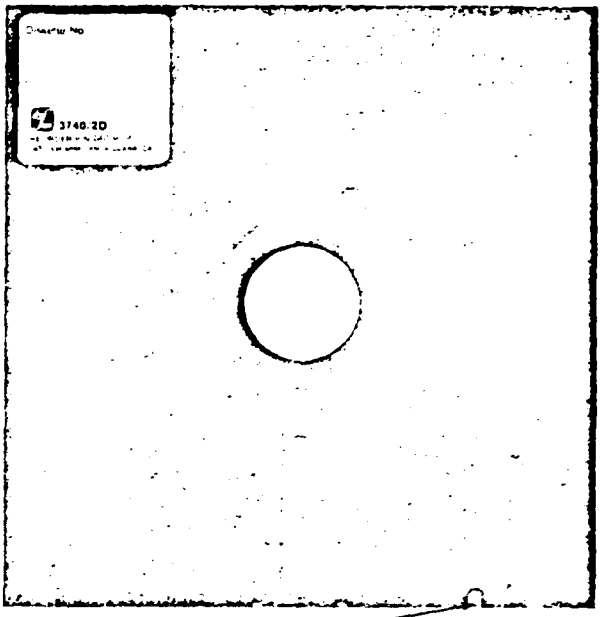
Three important things about OUTDISK:

a. OUTDISK must be accompanied by a filename: OUTDISK *'FILENAME.TYP'* (i.e., OUTDISK followed by a blank, then an apostrophe, the file name of at most 8 characters, then period, then file type of at most 3 characters, and finally a closing apostrophe. The file type may be omitted if the period is also left off.) Some *examples of valid file names*:
'FILE'
'GLUG.M'
'DISPLAY.MAC'

b. Strange things happen if, in drive B:
- there is a write-protected diskette
- there is no diskette
- the diskette was inserted after the last system boot. (CP/M checks *disk status at boot — disks inserted* after boot are made read-only.)

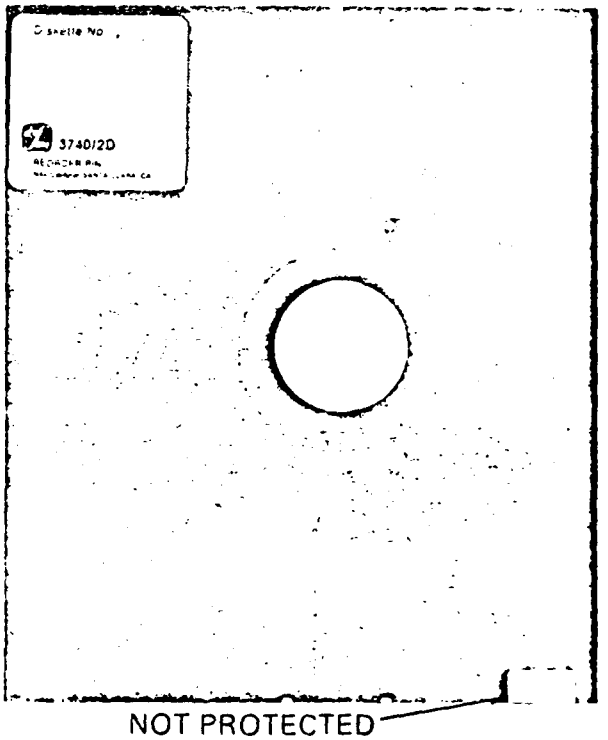
c. The effect of OUTDISK is similar to PRINTON in that as each line scrolls off the top of the screen it is written on the disk — therefore it is essential to do a FLUSH at the end of disk writing to close the output file properly.

113738-S




PROTECTED

113739 S




NOT PROTECTED

#### 2.3.2.1. Avoiding Data Dropout — BUFFER

Long messages sometimes have data dropout problems. To avoid this, use the command, !BUFFER. It sends incoming data to a core buffer rather than to the disk. The maximum file size is 30K bytes — exceeding this causes a crash.

Use the command, !ENDCORE, after !BUFFER has received the message. This flushes the buffer and writes the file it received to the disk. The file is now stored on the disk and can be read at any time with CP/M commands (see Appendix I).

### 2.3.3. Sending Messages or Files — INDISK

Use INDISK for sending long messages (or files that are already created). INDISK opens a named filed on drive A for reading only. First compose the message using the CP/M editor. Do this with the RST off-line, (not on the modem connection). Once the message is composed and stored on drive A establish the modem connection (CENTRAL) and then use INDISK.

a. Type!INDISK '*FILENAME.TYP*' and RETURN.

   The system prints the first line of the file on the screen and terminates it with a "?". It is asking if you want to send the line.

b. To send the line, hit RETURN.

   To discard the line, hit CTRL U.

c. To bring up the next line from the input file hit the three keys: SHIFT CTRL at the same time.

d. Add extra lines at any point by typing them in instead of calling up the next line.

e. If you get an error message it will flash until you type || (two vertical bars).

Characters in the Clear

SHIFT UPARROW (the cursor arrow on the special purpose key pad) sends a single carriage return without a line feed. The return key, when terminating a line from the composing line sends out a return and a line feed. Some systems require characters in the clear for checking in. Use SHIFT UPARROW.

## 3. Displaying Waveforms — DISP

DISP is a local program to display and analyze waveform data.

a To use the program make sure the system is displaying the CP/M prompt, then type DISP *FILENAME* followed by RETURN

Each *FILENAME* must contain waveform data in "gram file" format.
(You can call up to 3 files at a time using the form: DISP *FILENAME* [*FILE2* [*FILE3*]] )
The initial display scale uses the full window for the maximum amplitude in the entire waveform.

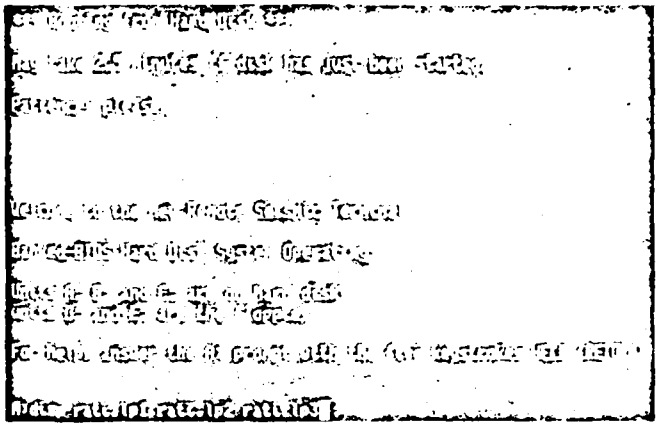
b. DISP Commands:

| | |
|---|---|
| ? | Print Command menu |
| A | Amplitude |
| P | Period |
| C | Cursor |
| S | Change scale factor |
| L | Shift waveform left |
| R | Shift waveform right |
| SHIFT HOME | Produce hard copy of the graphics screen |

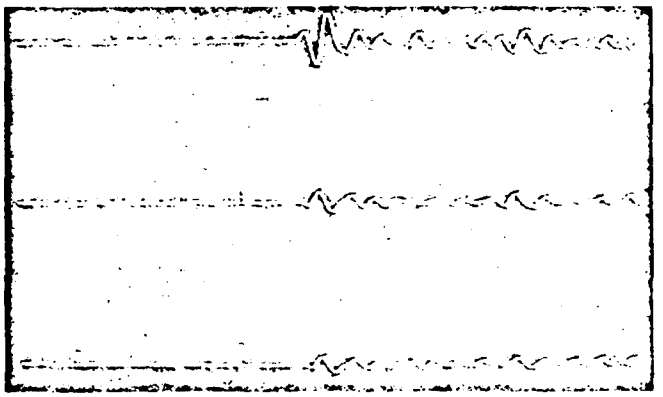
c. Use of DISP commands:

- Each command requires a RETURN.
- If parameters are required a prompt asks for them.

113740 S



113741 S



Amplitude measures the amplitude between 2 cursors that the user places on a waveform. Be sure to set the bottom cursor first and then the top — if you do it the other way around you will get a negative measurement:

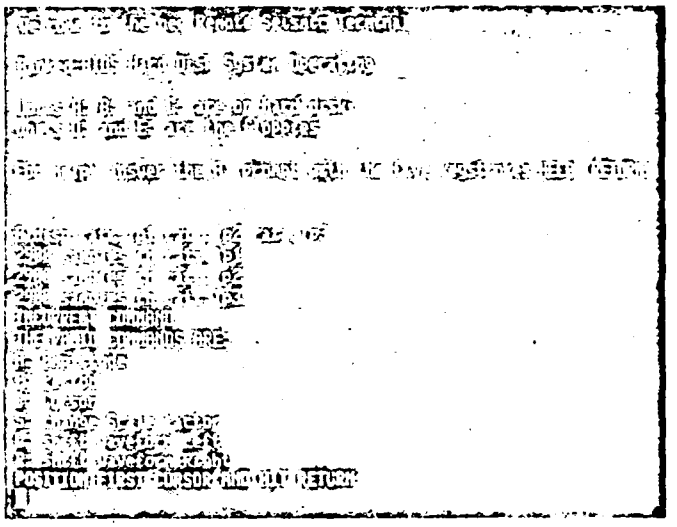
Type A and hit RETURN.

System responds: POSITION FIRST CURSOR AND HIT RETURN

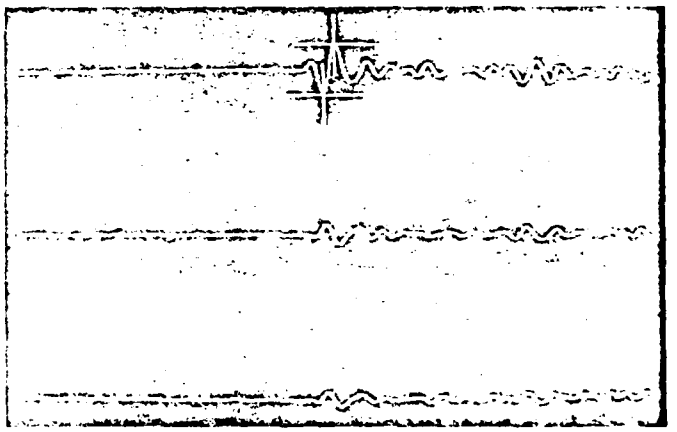
Switch the data tablet to STREAM MODE and position the crosshairs on the bottom of the waveform. Take your hand off the cursor control and make sure the cursor is still in the proper position. (This takes some practice as the cursor tends to slip off the mark as your hand leaves it.) Hit RETURN.

Repeat the process for the second cursor. After you have positioned the second cursor and hit RETURN, the system displays the amplitude measured between the two cursors.

113742-S



113743-S

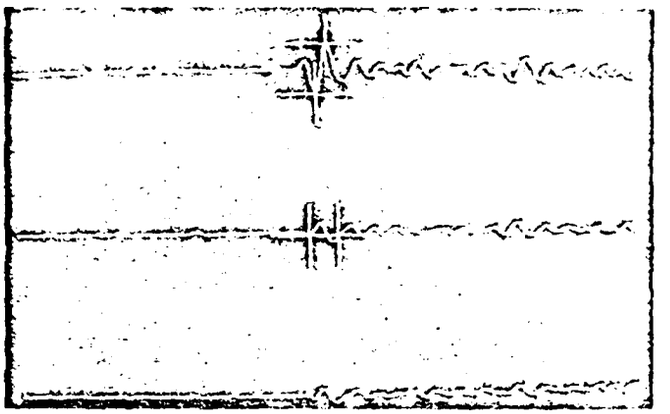


Period

Measures and displays the number of samples from the first to the second cursor The procedure is the same as for Amplitude

Type P and RETURN.

113744 S



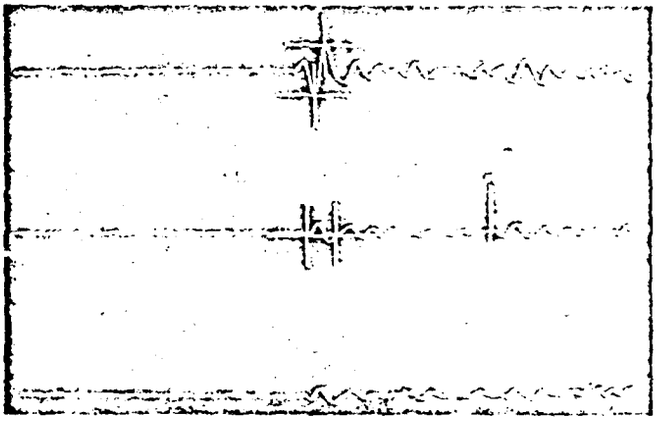
Cursor

Marks a waveform with a named cursor.

Type C and hit RETURN.

*Proceed as in Amplitude* and Period. You supply the name for the cursor.

113745 S



Scale Factor

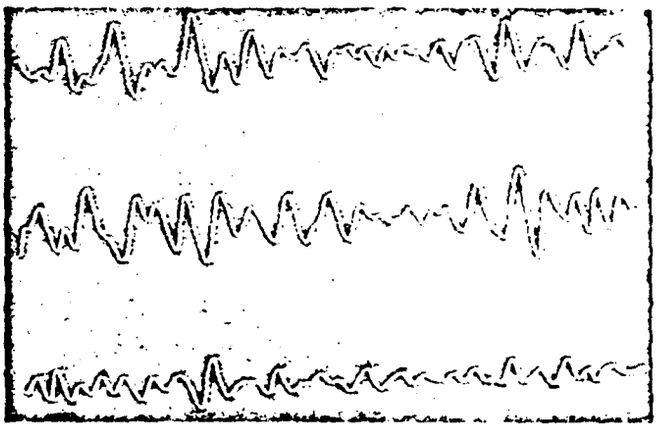
Changing the scale factor expands or contracts the waveform.

Note: Do not set the scale factor to 0. If you do it will crash the program and you will have to call it back.

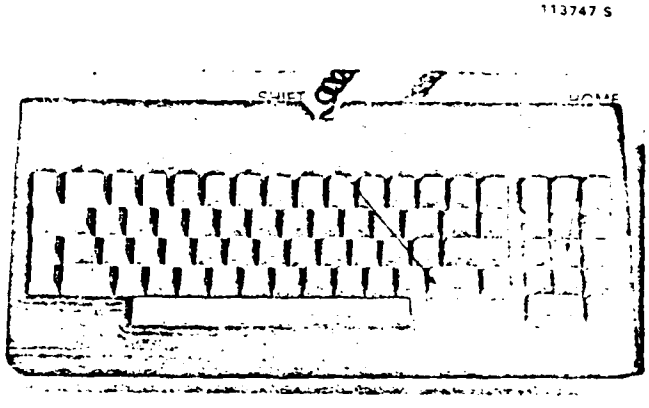
Type S and hit RETURN.

The system asks for the scale factor expressed as a floating point number.

113746 S



| | |
|---|---|
| Left Shift | Shifts the waveform left by a specified amount (measured in screen widths).<br><br>Type L and hit RETURN.<br><br>The system asks for scale factor, but give it the number of screen widths to shift the waveforms.<br><br>Type in the number and hit RETURN. |
| Right Shift | Shifts the waveform right by a specified amount.<br><br>Type R and hit RETURN.<br><br>*Proceed as with Left Shift.* |
| Hard Copy | While in DISP print out whatever is on the graphics screen by hitting:<br><br>SHIFT and HOME keys at the same time.<br><br>If you leave the DISP but want to print the contents of the graphics screen use the command, GETSCRN. |




### 3.1. Leaving DISP

To leave DISP type CTRL P or simply reboot the system (hit the reset button).

## 4. Digitizing Waveforms — DIGITEST

Use DIGITEST to digitize small amounts of waveform data. DIGITEST takes output from the data tablet and displays it on the graphic screen. It takes 512 data points at a time. When the 513th point is sent the 1st one disappears.

a. Type DIGITEST and hit RETURN

DON'T TOUCH THE KEYBOARD AGAIN! HITTING ANY ADDITIONAL KEY EXITS FROM THE DIGITEST PROGRAM.

b. Use the data tablet to trace the waveform:

Put the waveform on the pad. (Tape it down to keep it from shifting.)

Press the RESET button.

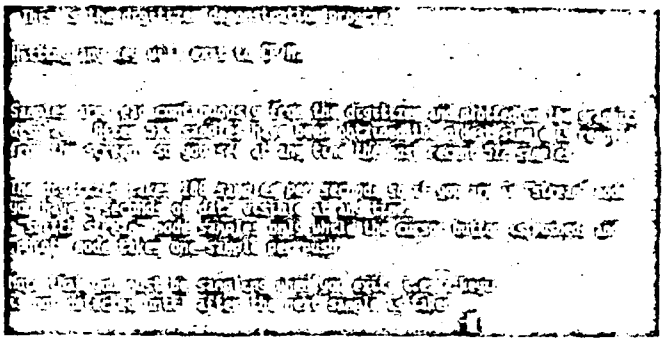
Choose the digitizing mode:

| | |
|---|---|
| Point | sends one point each time the button is pushed. |
| Switch | sends points continuously as long as the cursor button is pressed. |
| *Stream* | *sends points continuously without your having to press the button.* |

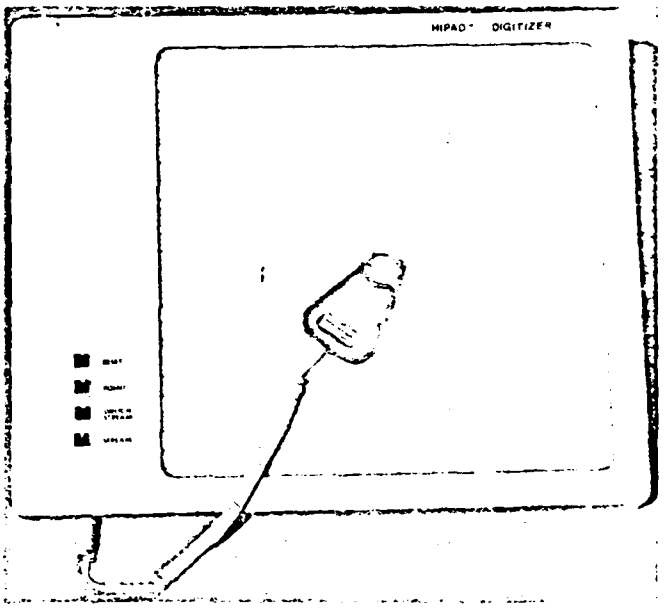
c. Trace the waveform.

d. To leave DIGITEST hit any key on the terminal. The system returns to CP/M.

113748-S



113749-S



## 5. Summary

| | |
|---|---|
| ON/OFF | Switches on power strip. |
| BOOT | Computer Reset Button on mainframe boots system from Drive A. |
| HELP | Prints instructions and options. The system must be in CP/M for HELP to function. |
| CENTRAL | Program for communicating with central site or another RST via modem (300 baud). |
| TERM1200 | Program for communicating at 1200 baud. Does not offer intelligent mode. |

INTELLIGENT MODE COMMANDS (CENTRAL)

Begin line with ! a terminate with RETURN.

| | |
|---|---|
| PRINTON | Turns printer on |
| NOPRINT | Turns printer off |
| FLUSH | Flushes screen |
| OUTDISK | Receives messages. Used with filename, opens that file for writing on Drive B. |
| INDISK | Sends messages. Used with filename, opens that file for read only on Drive A. |
| SHIFT | Sends single carriage return without line feed. (uses uparrow from the cursor control keys) |
| BUFFER | Sends incoming data to core buffer. |
| ENDCORE | Flushes buffer, sends contents to disk. |
| DISP | Displays waveform data and provides tools for analyzing it. DISP Commands: |

| | |
|---|---|
| A | Amplitude |
| P | Period |
| C | Cursor |
| S | Change scale factor |
| L | Shift waveform left |
| R | Shift waveform right |
| ? | Prints DISP menu |

| | |
|---|---|
| SHIFT HOME | Print contents of graphics display screen |
| GETSCRN | Prints the contents of the graphics display. (Use when not in DISP.) |
| DIGITEST | Uses the data tablet to digitize hard copy waveforms. |

## 6. Appendix I: CP/M Operations

When the RST is used as a normal microcomputer it is running under the operating system, CP/M. This section covers the CP/M operations: DIR, EDIT, ERA(SE), PIP, REN(AME), STAT(US). CP/M offers much more than this document outlines. For more about *CP/M see the CP/M handbook* supplied with the system.

### 6.1. The CP/M Prompt

When CP/M is in charge, i.e., when no special program is running, it displays the prompt, A>, if disk drive A is assigned, or B> if disk drive B is assigned.

To move from drive A to drive B —

Type B: followed by a RETURN

CP/M responds: B>

### 6.2. DIR (DIRectory)

A. To list all of the files on a diskette —

Type DIR followed by a RETURN

B. If you are on drive A and want to list the files in drive B —

Type DIR B followed by a RETURN.

C. To list a specific file (i.e., to see if it's on the currently assigned disk) —

Type DIR *FILENAME.EXT* followed by a RETURN.

If "filename.ext" is on the disk, DIR prints the name.

D. DIR uses the special characters, * and ?, in the same way PIP does. Often a directory is too large to fit the display. Use the special characters to narrow the listing to those files you are interested in, for example —

RATI.*

lists all files beginning with RATI and ignores everything else.

### 6.3. PIP

PIP stands for Peripheral Interchange Program. It transfers files between devices (e.g., from drive A to drive B, or from drive A to the printer) and can process them on the way.

### 6.3.1. Copying Files

Most of the time you will use PIP for file copying. For the purposes of instruction, assume we have a diskette full of files on drive A. We want to copy files from drive A to a diskette in drive B. We can do this in two ways: by using PIP as a single line command, or by using PIP as a program.

#### 6.3.1.1. *PIP as a single line command* —

Type PIP B:NEW.BAK = OLD.TXT followed by a RETURN

(PIP copies OLD.TXT, which it found on drive A, names it NEW.BAK and stores it on drive B. It then responds with the prompt, A> to let you know it has done the task and has returned to the CP/M level.)

The format for copying files is: new file = old file. If you want the new file stored on a different drive, specify that first: PIP device:new file = old file.

#### 6.3.1.2. *PIP as a program* —

Use the program, PIP, if you have more than one or two operations —

A. To enter PIP, make sure the CP/M prompt (A>) is displayed —

   Type PIP followed by a RETURN.

   (PIP displays the prompt, *, to let you know you are in PIP)

   Type B:FILE.BAK = FILE.TXT followed by a RETURN

   (this does exactly the same thing as when we use PIP as a single line command but instead of displaying the prompt, A>, PIP displays the prompt, *, and waits for the next PIP command. We can continue to copy files without entering PIP each time)

B. While you are still under PIP you can copy as many files as you want—

   A:=B:SAMPLE.TXT followed by a RETURN.

   (PIP copies the file, SAMPLE.TXT which it found on drive B onto drive A without renaming it, i.e., SAMPLE.TXT is now on both drive A and drive B.)

### 6.3.2. Special Characters: * and ?

CP/M has two special characters worth knowing about — * and ?.

? matches any single character in its place, so FIL? matches FILE, FIL2, or FILM. This is useful in copying a number of files with slight variations in their names using only one PIP command—

A> PIP B:=FILE.? followed by a RETURN

copies FILE.1, FILE.2, FILE.3 onto drive B. The command does not copy FILE.55 or FILE.BAK.

* matches any character or string of characters on its side of the "."—

FILE.* matches FILE.55 or FILE.BAK as well as FILE.1, FILE.2 and FILE.3.

A> PIP B:=*.* followed by a RETURN

copies every file on drive A onto drive B.

### 6.3.3. Printing Files

To print a file—

Hit CTRL P to turn on the printer. (Hit the CTRL and P keys at the same time.)

Type TYPE *FILENAME.EXT* followed by a RETURN.

## 6.4. EDIT (EDITor)

Use the editor (EDIT) for creating new files or for editing existing ones. Like PIP, EDIT is a program that runs under CP/M. EDIT displays the prompt, *, instead of the CP/M prompt, A> or B>.

### 6.4.1. Creating a New File—

Type EDIT *FILENAME.TAG* followed by a RETURN.

(A word about filenames: these can be any eight-or-less-characters, a ".", and then any three-or-less-characters. The new file we have just created will be called "filename.tag". If a file with the name, filename.tag, was already the directory, EDIT would have prepared that file for editing (N.B. See #A., below. EDIT does not automatically bring in the text.). As there

was no such file, EDIT named a blank file "filename.tag" and opened it up for whatever we want to put in it. We could have named the file Fred.dog or Alice.3 or Ralph. In general it is wise to choose a file name that labels the file. Many people use the extension .BAK to mark a file as a backup file, e.g., this file about CP/M might be called CP/M.TXT and its backup file, CP/M.BAK.)

EDIT is now in charge, has opened up "filename.tag" for editing, and displays the prompt, *, to indicate it is waiting for the next command.

### 6.4.2. Inserting Text —

A. Type I followed by a RETURN

EDIT displays a cursor to mark the spot on which the next character will appear.

B. Type in text much as you would on a typewriter. End each line of text with a RETURN (hit RETURN, do not type the word "return").

C. To stop inserting text type CTRL Z (hold down the CTRL key and hit Z at the same time).

EDIT displays *.

### 6.4.3. To Save the File and Leave EDIT —

Type E followed by a RETURN. (Make sure that you are not in insert mode — i.e., that a * is diplayed.)

CP/M displays the prompt, A> or B>.

### 6.4.4. To Edit an Existing File —

A. Type EDIT *FILENAME.EXT* followed by a RETURN.

EDIT opens that file for editing and displays the prompt, *.

B. Type #A.

EDIT brings the entire source file into the edit buffer. If you want to bring only a portion of the file into the buffer type **n**A where **n** is the number of lines of text to be appended. OA (capital O, not zero) brings in half of the text.

### 6.4.5. Moving Within the File

A. To go to the bottom of the buffer--

Type -B followed by a RETURN.

B. To display the entire buffer—

Type B#T followed by a RETURN.

T is the command to display the text in the buffer.

C. To move through the file use a line number followed by a colon—

Type 4: followed by a RETURN.

EDIT positions the cursor at line 4.

D. To move to a range of lines—

Type 5::9 followed by a RETURN.

EDIT displays lines 5 through 9 and positions the cursor at line 5.

### 6.4.6. Special Editor Commands

L To move up and down lines use +*n*L to move forward and *n*L to move backward where *n* is the number of lines you want to move.

C Moves the cursor by characters. 30C followed by a RETURN moves the cursor 30 characters, and 30C moves the cursor backward 30 characters. (A space is a character.)

K Deletes a line. +*n*K deletes *n* lines forward, -*n*K deletes n lines backward. K deletes the current line if no *n* is specified. #K deletes everything after the cursor, +#K deletes everything before it.

D Deletes a character. +*n* and -*n* arguments work here as well.

F Finds a specified string of characters, e.g., FTEH followed by a RETURN finds TEH and positions the cursor immediately after it.

S Searches for a specified string of characters and substitutes a new string of characters:

Type STEH[CTRLZ]THE followed by a RETURN.

EDIT finds the next occurrence of TEH and substitutes THE. Use an n argument to repeat the substitution n times: 10STEH**CTRLZ**THE followed by a return finds the next 10 occurrences of TEH and changes them to THE. #FTEH**CTRLZ**THE followed by a RETURN makes the substitution throughout the buffer.

### 6.5. ERA (ERAse)

ERA erases entire files. Once erased, a file is gone.

A. To erase a file—

Type ERA FILENAME.EXT followed by a RETURN.

B. ERA recognizes * and ?, and can therefore be dangerous, e.g., ERA *.* erases every file on the disk.

C. ERA can be used with a device name, e.g., ERA B:RALPH.CAT erases the file "Ralph.cat" from the disk in drive B.

### 6.6. REN (REName)

REN renames files.

Use the format: REN NEWFILE = OLDFILE (followed by a RETURN).

When you rename a file, you actually copy the file under a different name. The file ceases to exist under the old name.

### 6.7. STAT (STATus)

STAT displays the status of the system.

A. To find the size of the remaining disk space—

Type STAT followed by a RETURN.

STAT responds with: DEVICE NAME (drive A or B): R/W SPACE: *nnn*K

(R/W means read/write, i.e., that you can create new files, delete old ones, change things in old files. R/O means read only — you can read what is on the disk, but you cannot add to it, nor can you erase anything or change anything. The amount

of space shown is measured in bytes -- the RST 8-inch diskette holds 1.2 M bytes.)

**References**

Rodnay Zaks, **The CP/M Handbook, with mp/m** Sybex, Inc , 1980.

# 7. Appendix II

## 7.1 Hardware

The RST consists of the following components. Individual documentation comes with each item.

Mainframe boards:

a. California Computer Systems CPU board
Holds Z80 CPU and programmable serial interface. Includes 2K monitor PROM which is disabled in the final design.

b. Memory:
One 16K static RAM
One 64K dynamic RAM

c. Morrow Design floppy disk controller board

d. Morrow Design hard disk controller board

e. Scion graphics display controller board. Includes its own Z80 and memory.

Other Hardware:

a. Scion graphics display

b. Adds Viewpoint terminal

c. Racal Vadic modem
Operates at 300 or 1200 baud, the latter with Bell 212A (industry standard) protocol and the older Vadic protocols.

d. Morrow Design Discus 2+2D floppy disk drive

e. Morrow Design 26 megabyte hard disk

f. Houston Instruments Hi Pad digitizing tablet

g. Integral Data Systems 460G Paper Tiger printer

## 7.2 Notes from the Designer

In the mainframe there is a California Computer Systems CPU board which provides a Z80 CPU and a programmable serial interface. A 2K prom is also on the board — it can be used for troubleshooting, but is disabled in normal use. This monitor can be used if a special cable is connected between the terminal and the CCS serial port and certain configuration switches on the board are changed. See the CCS documentation for further information.

The Morrow disk controller board operates the floppy disks and also provides a serial port that is used to communicate with the video terminal for system control. This port is memory mapped and hence will not appear in the list of I/O ports to be given later.

The Scion graphic display is controlled by a separate board which has its own Z80 and memory connected to an internal bus. Communication with the system is via I/O ports, and the board produces video for driving the monitor.

The Morrow Designs "Switchboard" incorporates two serial and four parallel ports plus a status port. The board also has sockets for 2K of RAM and 2K of PROM memory, but these are not used.

**Other Hardware**

The Scion graphics display accepts standard signals via its coax input connector and converts them to light on the screen.

The Morrow Designs dual-disk drive is capable of reading both single- and double-density diskettes.

**I/O Ports**

All numbers in this section are hexidecimal.

The California Computer Systems CPU board contains a programmable serial port that is assigned I/O address 20-25.

The CCS serial port connects to the modem. Port 20 is the transmit/receive port, 25 is the status port, and the others are used for programming the 8251 UART. A data sheet for the latter is included in the CCS CPU manual. The interested reader should examine the routines DUMT300 and DUMT1200 to see how this port is set up and used.

The Morrow Switchboard, which provides two serial and four parallel ports plus a status port, is assigned A0-A7.

Switchboard port A0 is the lineprinter output. A1 is the special port used to establish serial communication with another computer; it is not connected in normal RST operation. A2 is the status port for both serial interfaces. A7 is an output port used to establish CTS and DSR for the special port; A5 is an input port for reading the printer's DTR line. A6 is used to "remember" the bank selection, in effect. A4 is an input port used to read the digitizer pad.

The graphics display occupies F0-F8, although it only uses F0 and F1.

### Memory

The 16K static RAM is assigned to C000 through F7FF. This particular board has the capability for disabling its chips in 2K byte chunks, and the last 2K bytes are indeed disabled to leave the space F800-FFFF for the disk controller's PROM and RAM; thus only 14K of the board is in actual use.

The CCS 64K dynamic RAM board allows disabling in 16K-byte chunks and the high-address 16K of each is, in fact, disabled, leaving 48K of each board active.

### CP/M CBIOS Customization

One part of the CP/M system, the customized basic input/output system (CBIOS) must always be adjusted for a particular installation. For the RST it was only necessary to make a few minor alterations in the CBIOS supplied by Morrow Designs with the disk drives.

The customization most obvious to the user, and the most trivial, is the change in the sign-on message to read "for the Remote Seismic Terminal". Other initialization changes such as this included moving the CP/M "hooks" (the jump vectors stored at locations 0 and 5) to banks two and three, resetting the graphics display while selecting its rolling scroll mode, and initializing the special serial interface (this interface, not the CAC computer, later the Lincoln PDP/11-50, for purposes of rapid transfer of software and data).

CP/M has a jump vector stored at a standard location. This vector was extended to include, as part of the system, some RST specific chores as follows: send a character to the graphics display in alphanumeric mode, and a character to the line printer, send a character to the graphics display in graphics mode, read a character from the graphics display, switch to bank1, switch to bank2, switch to bank3. In addition to extending the transfer vector, small routines to perform the above chores were also inserted in the CBIOS. The interested reader can find these in the listing near "cocrt:" and "tomike:".

The standard CP/M devices are connected to drivers as follows:

| | | | |
|---|---|---|---|
| CRT: | UL1: | UC1: | Graphics display, |
| LPT: | UP2: | | Lineprinter, |
| PTP: | | | Modem output, |
| UP1: | | | Special serial port output, |
| PTR: | | | Modem input, |
| UR1: | | | Special serial port input. |

## 8. Appendix III: Moving Remote Seismic Terminal Hardware

This appendix describes how to pack the RST for shipment or by reversing the steps, how to unpack it and set it up for use. Necessary tools are: standard and Phillips-head screwdrivers and a pair of long-nose pliers.

To begin packing, select an empty box that will contain the manuals for the individual components. As each component is packed, put its manual into the manuals box.

### 8.1 Printer

A. Turn off ac power; unplug cord, coil and lash cord with rubber band or plastic-covered wire.

B. Consult manual and remove cover carefully.

C. Disconnect computer interface cable; check that connector is labeled; if not, add a label so that it can be readily found when reassembling.

D. Consult manual and lock printhead and ink ribbon.

E. Put cover back on.

F. Package is form fit; consult manual for packing instructions.

G. To set up, reverse above steps. Note: Printer has a self-test feature.

### 8.2 Terminal

A. Turn off ac power; unplug cord, coil and lash cord with rubber band or plastic-covered wire.

B. Disconnect computer interface cable; check that connector is labeled; if not, add an appropriate label to facilitate future recognition.

C. Package is form fit.

D. To set up, reverse above steps.

E. Terminal has a self-test feature (consult manual).

## 8.3 Graphics Display Monitor

A. Make sure power cord is labeled, then disconnect from monitor and outlet, coil, and store. Monitor does not have an on/off switch.

B. Disconnect coaxial cable.

C. Monitor package has corner holders for each apex; a piece of cardboard that goes in front of the screen, and "popcorn" to fill all spaces. Best packing procedure is to dump the popcorn into a convenient receptable, temporarily; get the monitor set into the bottom four-corner holders, put in the face cover, fill all sides with popcorn, install the top four-corner holders, add more popcorn.

D. Setup is easier. Open the top cover to check for loose circuit boards — the boards mount on plastic holders and have been known to pop loose during shipment. Consult manual for proper position if one is loose.

## 8.4 Hard Disk

A. Turn off ac power. Make sure power cord is labeled, retain for use in a later step.

B. Make sure the two flat cable connectors are labeled so that you can restore them in the correct position — neither has a key, and either can be put in the "wrong way." Disconnect the two flat cable connectors carefully. Pull on the connector, never on the cable. (May require long-nose pliers.)

C. Remove the top cover by extracting all of the screws on each side.

D. Consult the shipping instructions that are in the packing box. Apply ac power, let the drive come up to speed, and install the damper lock. Remove ac power.

E. Consult shipping instructions and install the spindle lock screw.

F. Install the warning signs near lock screw and damper.

G. Consult shipping instructions and remove drive from bottom of cabinet. Reposition shock mounts on drive as in shipping instructions. Drive package is empty; simply slide drive in.

H. Reassemble the cabinet using the small screws only. Save the large ones, which seat in the drive and will not catch at this point.

I. Cabinet has corner holders in its package.

J. For setup, follow instructions carefully to reverse all of above.

### 8.5 Floppy Disks

A. Turn off ac power and unplug cord.

B. Remove the flat cable connector carefully as described under Section 8.4, Hard Disk.

C. Find the cardboard shipping disks and door locks in the packing box and install as directed.

D. The disk carton has corner holders, or more precisely, end holders, plus a small empty section that can be used for some of the floppys.

### 8.6 Main Frame

A. Turn off ac power; unplug cord.

B. Remove cover by withdrawing screws on each side.

C. Disconnect motherboard power cable from power supply circuit board. Use a firm pull at the plastic connector.

D. Disconnect power supply ac inputs, two single connectors and a double connector.

E. Disconnect power cable from power supply board to the small reset circuit board.

F. Remove three bolts holding power supply to bottom of cabinet. Very carefully remove power supply from cabinet, reinstall the three bolts in bottom plate of supply to avoid loss. The power supply is packed in a separate box of its own.

G. Remove all cables connecting to the DB-25 connectors on the back of the mainframe. Make sure, as each connector is removed, that there is a proper label on each part. Remove the video cable and the reset plug if installed.

H. Fold the two hard disk flat cables and the floppy cable into the interior of the cabinet.

I. Reinstall the cover and screws.

J. The cabinet has a form-fitted box of its own.

K. Setup is the reverse of the above procedures. Check that all circuit boards remained in the cabinet during shipment. Check carefully that these boards are all properly seated as they tend to "pop out" during shipment. Note also that each board has its own manual to be packed, in addition to the mainframe manual.

## 8.7 Modem and Telephone

A. Disconnect the modular phone jacks connecting the telephone to the wall outlet and the modem to the telephone.

B. Unplug the ac power transformer.

C. Disconnect the RS-232 cable after making sure that it is properly labeled.

D. The modem has a special box of its own; the phone lives in a box within that box.

E. Setup is reverse of above. Study the manual to use the modem's self-test feature: In ALB, every key sent from the terminal should be returned by the modem, and hence, displayed on the screen; also, the indicator lights show what is happening.

## 8.8 Cables and Connectors

A. This section lists all of the cables that at this point have no connection on either end, and hence, can be packed in a separate box:

Printer cable
Terminal cable
Modem cable
Video cable
Power cords for hard disk, floppy disk, graphics monitor
Two ac power boxes.

B. Do not forget the box of printer paper and two plastic boxes of floppies.

APPENDIX C

APPENDIX C

# A SEISMOMETER RECORDING SYSTEM FOR THE REMOTE SEISMIC TERMINAL

Richard M Moroney
CAC of Sanford
2 School St
Sanford, Maine 04073

# A SEISMOMETER RECORDING SYSTEM FOR THE REMOTE SEISMIC TERMINAL


## Abstract

The design of a real-time system for seismometry data recording is presented. Although the task to be performed is too demanding for a single eight-bit microcomputer, it is handled rather easily by a collection of them in a distributed-processing system. The primary message is that much can be accomplished by "throwing micros" at a problem.


## Introduction

The Remote Seismic Terminal (RST) has been described elsewhere [1]. Briefly, it is a microcomputer-based work station that allows a seismologist at any location enjoying dial-up telephone service to maintain contact with the world's seismological community, exchanging reports and waveform data. There is also a capability for local analysis and display of such data.

At every demonstration of the RST the first question to be asked was "Can I hook this thing up to a seismometer?". The unit to be described here was constructed to answer that question in the affirmative. It is a box that can be connected between a seismometer and an RST and will perform all of the functions implied by "hooking up to a seismometer".

No knowledge of seismometry is necessary for reading this report (nor, fortunately, for writing it). A few words will be required to introduce the problem, but this paper is about a real-time computer system using microprocessors; the actual application is incidental with the exception that it supplies the parameters that drive the system's design (and, let us not forget, pays the bills!).

# A SEISMOMETER RECORDING SYSTEM FOR THE REMOTE SEISMIC TERMINAL

## 11 About Seismometry

A seismometer senses vibrations of the earth in three orthogonal directions, producing three analog time signals, typically in the range 0-5 volts. The signals are bandpass filtered to obtain long-period, medium-period, and short-period components in each axis, thus raising the number of channels to nine. These signals are to be digitized at a rate high compared to the frequency content, and with a resolution of 15 bits. It can be assumed that any necessary anti-aliasing filtering has been performed.

The long-period channels are to be sampled once per second, the medium-period four times per second, and the short-period forty times per second. Using sixteen instead of fifteen for the number of bits in a sample, or two eight-bit bytes, we can calculate that the input data rate is

| | | | | |
|---|---|---|---|---|
| 3(channels) | X 2(bytes/channel) | X 40(samples/sec) | = | 240 bytes/sec |
| + 3 | X 2 | X 4 | = | 24 |
| + 3 | X 2 | X 1 | = | 6 |
| | | | | 270 bytes/sec. |

The input data must be time-stamped with an accuracy of one millisecond and then recorded permanently somewhere. Allowing a few bytes for the time information leads us to an output data rate of 300 bytes per second or 2400 Hz.

It must be possible, while performing the above data collection and recording task on a round-the-clock basis, to send reports on the data to an external sink (the RST) on request.

# A SEISMOMETER RECORDING SYSTEM FOR THE REMOTE SEISMIC TERMINAL

## System Components

The Seismometer Recording System (SRS) comprises a Tape Unit and two separate microcomputers packaged in the same box; the micros will be called the 'Seismo Processor' and the 'I/O Processor'.

The Tape Unit is a commercial offering that contains two standard 1/4" cartridge magnetic tape drives, together with control electronics and an RS-232C serial port for communicating with the rest of the system. The controller contains a microprocessor which performs such operations as circular redundancy check (crc) generation and comparison, parity checking, and the resultant retrys. This intelligence is very powerful, however it still leaves much that must be done by a driver in the connecting device.

The Seismo Processor has a Central Processing Unit (CPU), 48K bytes of dynamic memory, nine analog-to-digital (A/D) converters, a time-of-day clock, and two parallel input-output (I/O) ports that provide bidirectional communication with the I/O Processor. (Regarding hardware implementation, there exists a separate board corresponding to each item in the foregoing list, however some boards have additional capabilities not used in this system and hence not mentioned. Also, to be precise there is only one A/D converter, a multiplexer connects it to nine input lines.)

The I/O Processor too has a CPU and 48K bytes of dynamic memory, plus a serial I/O port for communicating with the tape unit, and four parallel ports. Two of the parallel ports connect to the corresponding ports on the Seismo Processor, while two provide bidirectional communication with the RST itself (if used). The RST ports have RS-422 converters to permit reasonable connecting cable lengths (up to fifty feet); since the Seismo Processor and the I/O processor are within inches of each other, their parallel connection is direct wiring port-to-port. A floppy disk controller board can be added to the I/O processor, making the SRS a stand-alone unit; thus an RST is not required to operate the SRS.

# A SEISMOMETER RECORDING SYSTEM FOR THE REMOTE SEISMIC TERMINAL

## )esign Considerations

### R E C O R D I N G  M E D I U M

Cartridge tape was selected for the archive function for several reasons. Cartridges are relatively cheap (about $10 each) and of convenient size for mailing and storage. A single 450-foot standard cartridge can hold some 11 megabytes of data, which is more than eight hours worth; this means that medium changes are infrequently required, in particular a two-drive system can be loaded before leaving work at 5pm and still be operating without overwrite the next morning at 9am. (Note that this consideration is not the one that dictates two drives; two of whatever device is chosen are required so that data is not lost during a medium change. In any case, it is always essential to have two of any sort of recording device, for purposes of "making backups" and so on.)

The read/write data rate of cartridges operating at thirty inches per second and 6400 bits per inch density is 2400 bytes per second, well beyond that required even when retrys, rewinds, and other contingencies are considered. To see why this excess capacity is an important consideration, suppose that the recording operation took (say) three quarters of the available capacity. In that case it would never be possible to import data into the RST (or wherever) at the arrival rate: the system could record all of the data, but would not permit examination of all of the data.

Finally, the nature of cartridge recording in four parallel tracks means that any particular data item is available within at worst two rewind times; with a ninety-inch-per-second seek speed this comes to two minutes, an acceptable figure as will be seen later.

### T A P E  I N T E R F A C E

A tape unit with a standard interface was desired to allow that (expensive) device to be re-used conveniently in other systems--the present project is a "demonstration of feasibility", not a "production model". Asynchronous RS-232C serial operation at 19.2 Kbaud was therefore selected; a higher data rate would be much preferable but nonstandard.

Assume one start bit, eight data bits, a parity bit and two stop bits or a total of twelve bits transmitted serially per byte of data. 19.2KB reduces to 1800 bytes per second on this basis, thus the tape can read or write faster than its serial port operates. There is a

double penalty involved here, because data transmission and tape reading/writing must occur in series. (The tape unit operates on a block basis: on a write, for example, a complete data record must first be received before it is written on the tape. This permits crc checking and automatic retrys, but it precludes simultaneous operations on the drive and the serial port.)

Let us pursue this numerically. Assume a record length of 3072 bytes, which will contain ten seconds worth of data. The actual write time will be 3072/2400 seconds, with an additional 3072/1800 seconds needed to transfer the block to the tape controller. The total of 3 seconds shows that the tape unit is 30% loaded by its recording function; an allowance for retrys and rewind times must be added here, but the latter at least are negligible: each track will hold two hours of data before a one-minute rewind is required, which adds less than one percent to the loading. If we take the loading as one-third we have made a very conservative allowance for retrys (one every ten records).

Note that the foregoing implies that during an eight-hour shift an analyst can access at most sixteen hours worth of data (read timing is the same as write): it takes one third of tape unit capability to record the incoming data, meaning that twice as much data can be examined as recorded. Since eight hours worth are recorded in eight hours, sixteen hours worth can be examined concurrently. It would appear, then, that a day's worth of data could not be examined during a single shift, but things are not as bad as they sound. In practice all raw data is processed by an "event detection" algorithm before being presented to an analyst, thereby cutting his workload dramatically, since most of the time there is no seismic activity. The key consideration is that the event detector, which is a computer program, has no objection to working around the clock and hence has a comfortable amount of tape unit capacity available for obtaining its inputs.

So far, in this subsection headed TAPE INTERFACE, we have concentrated pretty much on the tape side, establishing that the design is adequate for the task at hand, but not an overkill. Now it is necessary to see what is required of a driver on the other side of the interface. To begin, we follow a write-block operation through to completion--the reader unfamiliar with the RS-232C signals DTR,DSR,RTS,CTS,RLSD need only realize that these are separate control lines:

1. Driver raises DSR, tape responds by raising DTR.
   (Not necessary for every transaction)

2. Driver raises RLSD and sends the block of data, preceded by a three-byte "command", which in this case calls for a write of a block of 3072 bytes.

3. On receipt of the last data byte, tape raises RTS (Tape may raise RTS prematurely if parity error is detected in the incoming data, when driver must start over from step 1.)

4. Driver responds by raising CTS and listens for status response.

5. After the operation is complete tape sends a two-byte status response.

6. Tape drops RTS after response has been sent, driver drops CTS.

X. A read operation is very similar; a command but no data is sent in step 2, the status bytes in step 5 are followed by a data block instead.

It is clear that the driver is going to have to field a number of interrupts, the primary ones for raw data requests (read or write) coming at a rate of 1800 per second during some intervals. It has been found that an eight-bit processor running at a 4MHz clock rate has an interrupt overhead of at worst about 100 microseconds, that is it can in that time save the machine state, spend a few instructions deciding what has to be done (vectoring), then restore the state and exit; naturally any service time for the operation required must be added to the overhead in calculating the total time consumed by an interrupt. The service times for tape interrupts can be kept small; in normal operation all that is needed is a get/put of one byte in a circular buffer followed by a reactivation of the serial chip, and "operation complete" or "disaster" interrupts can simply kill the current operation while setting a flag for later processing (when there will be no tape interrupts since the operation was killed). In short, driving the tape unit will consume at most 25% of the processor's time when the tape unit is active, but that activity must be expected to be occurring almost all of the time when operations with the second unit (RST support) are allowed for.

## D U A L  P R O C E S S O R  O P E R A T I O N

The above calculations considered only the tape interface. When the task of operating the A/D converters is thrown into the pot a whole new set of interrupts arrive to be handled within one

millisecond. True, the fastest raw data rate at the A/Ds is once every 25 milliseconds; however, if a constant sampling interval is to be maintained, with no skew in the data, each "significant" (25-th) millisecond must be coped with during that one millisecond. (Otherwise put, it is no consolation that there is no problem during "most" one-millisecond intervals.)

Trial coding showed that these new chores would push a processor up to the 50% loading level. Adding in communication with the RST plus other less-demanding operations would, it was felt, be sailing too close to danger: remember what happens to queues as the arrival rate begins to be a significant fraction of the service rate! Fortunately a simple solution was readily available. Since the two big tasks (tape and A/D operation) are essentially orthogonal, just add a new processor.

The writer is old enough to remember well when such a solution was unthinkable; today the cost of a second processor is so low that it is not worth while even to study the situation at length: if you can use one (i.e. divide up the problem neatly) then do so, and do so immediately, because it can be purchased much more cheaply than a few hours of your time! (That valuable lesson was learned by this old crock while working on the RST: after agonizing for two days about an imminent overflow of memory, it was realized that buying another 64K board was cheaper than thinking about the problem. No doubt none of this is news to the younger generation, but there are still a few contemporaries out there.)

Offloading of the A/D operations simplified the software design greatly too, as will be seen.

## OTHER INTERFACES

Parallel interfaces were selected for connecting the Seismo Processor and the RST with the I/O Processor, from considerations of speed. The only place where this was a drawback was in the I/O Processor-to-RST interface, where a reasonably long cable length was desired and hence RS-422 signals had to be introduced, but here again, hardware is cheap and there are chips to do the job, so why not?

Note that the data rate between I/O Processor and the RST should be double the tape data rate if the projected heavy reading of the off-line tape is to be accomplished; use of serial transmission would have made this marginal. On the Seismo Processor side speed is not so critical, but the proximity of the processors makes a parallel

interface the obvious choice

## S O F T W A R E  S Y S T E M  D E S I G N

More details on the software will appear in separate sections later. The discussion here is of the "big picture".

Software for the Seismo Processor is straightforward and was written from scratch. The many tasks of the I/O Processor indicated the need for a more organized approach, so a commercially-available real-time operating system was procured to provide a core Executive on which to build. Given that, the design was not difficult: various tasks and semaphores were identified, together with obvious data queues, and the individual pieces were written and attached to the Executive.

All of the software was written in assembler. The writer has no prejudice against higher-level languages in their place, but does not belong to the cult that uses them even for true machine-level work. The I/O Processor is the system master. On power-up it seeks a download from the RST (this can be replaced by a floppy boot, as mentioned) and then resets and downloads the Seismo Processor.

## Seismo Processor Software

The Seismo Processor has limited tasks. It must read the clock and the A/D converters at appropriate times, stuff this information into a large buffer (large because the I/O processor is occasionally not listening), and transmit the data to the I/O processor whenever possible. The simplicity of this mission allows a simple-minded program. Note, however, that the Seismo processor is doing an important job, namely freeing the I/O processor from all concern with operations requiring a time resolution of one millisecond. It will be seen that the A/D converters are read in groups of three at times precise to one millisecond; easy for the Seismo processor, since it has nothing better to do, but an impossible burden for the I/O processor with its many other chores.

On power-up the Seismo Processor performs normal housekeeping functions and then waits for a download from the I/O Processor over the parallel port. This is a debugging and test feature, since a complete program for the Seismo Processor resides in 2K of PROM on the CPU board; the normal download from the I/O Processor is a single jump instruction to prom-resident code.

The Seismo Processor is interrupt-driven, and has only two interrupts. One of these, a pulse from the I/O Processor, connects to the NMI (non-maskable-interrupt) line and simply causes a processor restart identical to the power-on sequence; it is used by the I/O processor to condition the Seismo Processor to a known state as described above. The second interrupt occurs periodically, once every millisecond, and is supplied by the clock board. That interrupt will be discussed below. First, however, a word about the background or idle process.

The Seismo Processor maintains a circular output buffer that is 32K bytes in length. This buffer is filled by the interrupt handlers to be described. When idle, that is when not servicing an interrupt, the Seismo Processor attempts to empty this buffer by sending bytes over the parallel port to the I/O Processor. Only if the buffer is empty will other actions be contemplated. Should the buffer become empty, the Seismo Processor sets a special "all caught up" flag for later use, and then checks the input port to see whether there is a command waiting from the I/O Processor. At present the only command recognized is a "set clock digit" order, which causes the Seismo Processor to update one entry in the clock memory. The selection of which digit, and the value, are supplied by the I/O

Processor. This allows corrections to be made to the clock "on the fly" during operations, a feature that will be of use when the system is enhanced by the addition of a satellite-time-signal receiver as is planned.

Upon receipt of the one-millisecond interrupt the Seismo Processor increments a counter that runs from zero to twenty-five and is then reset to zero. This counter (call it the "25 counter") allows the processor to select one of 25 different procedures to be run in response to the interrupt. Let us define "Procedure n" as the operations performed when the counter has value n. All but three of these do nothing, we examine the significant ones.

## PROCEDURE 23 (P23)

P23 checks the values of two counters. One of these, call it the "long counter" runs from zero up to four and is then reset to zero. The other counter, call it the "medium counter" is similar but with a maximum count of ten. If the long counter is at four and the medium counter is at ten then this procedure inserts items into the output buffer, otherwise it simply releases.

It will be seen that the long and medium counters have the values four and ten respectively precisely once each second, so in fact this procedure operates once each second. First, the long counter is reset to zero. Next, the "all caught up" flag is checked and, if set, a special flag (CC hex) is placed in the output buffer. Then a special "beginnning of second" flag (BB hex) is put into the buffer followed by seven bytes of time information. Each of those bytes contains two binary coded decimal (BCD) time items, so fourteen time digits are transmitted. There are five digits of days since some epoch, two digits each for hours, minutes, and seconds, and four digits of decimal fraction seconds; thus this "time block" has a resolution of 100 microseconds (and a dynamic range of hundreds of years !). Following the time block a trailer flag (also BB hex) is placed in the output buffer and then three of the A/D converters ( numbers 7, 8 and 9) are read and their outputs placed in the buffer, two bytes for each. In summary, this procedure has placed in the output buffer a long time message, guarded on each end by BB flags possibly preceded by the "all caught up" indicator CC, and followed by the values of three A/D converters--the long-period data.

## PROCEDURE 24 (P24)

P24 checks the "medium counter" for a value of ten; exits on any

other value The value will be in fact be ten just four times per second, so this procedure operates four times per second; furthermore, it is synchronized with P23 in that whenever P23 runs P24 runs at the next millisecond (because P23 did not reset the medium counter) P24 resets the medium counter, increments the long counter, and places bytes in the output buffer as follows: first, a special "four per second" flag of AA hex; then one byte of time data (seconds and tenths of seconds); then the values of A/Ds 4, 5, and 6; finally a trailing AA flag. In summary, the medium-period data.

## PROCEDURE 25 (P25)

This procedure runs every 25 milliseconds. It clears the "25 counter" and increments the medium counter, then reads and places in the output buffer the values of A/Ds 1,2,3: the short-period data. The reader can now verify the previous assertion regarding the frequency of invocation of PROCEDURE 23.

## /O Processor Software

The 1/O Processor has two major functions: it must take data from the Seismo Processor and send it to the Tape Unit, and it must talk to the RST; the dialog with the RST will usually involve positioning and reading the inactive tape drive, that is the drive not currently receiving the seismo data. All this leads to a complex software package impossible to describe very well within reasonable length. To perform its tasks the 1/O Processor makes heavy use of queues, servers, and interrupts; the verbosity solution adopted here is to give a short discussion of each of those entities and nothing more. The magic of making them play together is standard stuff for real-time programmers and of no interest to others.

### Q U E U E S

A queue is a collection of objects that can be enlarged (written to) or depleted (read from) on a first-in-first-out basis; all of the queues in the 1/O processor contain pointers, that is the starting addresses of certain blocks of memory.

#### TAPE-BLOCKS QUEUE

On program initialization all queues are emptied with the exception of certain "free block" queues which are loaded with pointers to all of the allocated memory blocks of a certain type. This is one such queue: it starts out containing pointers to the four 3072-byte blocks assigned to holding tape records. Those blocks will be called 'tape blocks'.

#### SEISMO-BLOCKS QUEUE

This queue is similar, at initialization time it contains pointers to the five 291-byte blocks assigned to holding input data from the Seismo Processor. Blocks of that type will be called 'seismo blocks'. (One second's worth of data from the Seismo Processor, with any leading CC byte removed, is 291 bytes).

#### TAPE-TO-RST QUEUE

This queue contains pointers to message blocks holding tape server responses to operations requested by the RST.

RST-TO-TAPE QUEUE

Contains pointers to message blocks holding tape operation requests originated by the RST.

SEISMO-INPUT QUEUE

Contains pointers to seismo blocks that hold data received from the Seismo Processor.

SEISMO-TO-TAPE QUEUE

Contains pointers to tape blocks full of processed data from the Seismo Processor.

## S E R V E R S

A server is a code module that is executed (dispatched) by the operating system when two conditions are met. One of these conditions is that the server is "ready", that is it has something to do and is requesting use of the processor. Each server is assigned a unique priority. The other condition is that of all "ready" servers this one has the highest priority.

One way a server can indicate its readiness is by accessing a queue. Using read as an example, a server can request the next object from some queue while specifying that if the queue is empty then the server is to be considered unready. The operating system will remember when such a request made a server unready, and will "wake up" the server when something is written to the queue. This is called an unconditional read of the queue. A server can also opt for a conditional read; when an empty queue will not cause a wait but simply the return of an appropriate flag.

Another readiness tool is the semaphore. A server can ask to be made ready when some semaphore is set. Unlike queues, each semaphore is unique to some server; it is set by an interrupt handler as will be seen.

A server can also request to go unready for a specified time interval, i.e. to "sleep" and then be "awakened". In this way a server waiting for some event can release the processor to lower-priority servers for a controlled length of time.

A SEISMOMETER RECORDING SYSTEM FOR THE REMOTE SEISMIC TERMINAL

TAPE-SERVER

This is the most complicated of the servers. It does a conditional read of the seismo-to-tape queue; if there is a pointer in that queue then the pointer is passed to the tape driver subroutine along with a write request. This causes the block to be written to tape; the tape driver will eventually return, when tape-server writes the pointer of the block just written to the tape-blocks queue--the block is available for another use. Tape-server then returns to the conditional read to look for another block of seismo data.

If the conditional read above fails, tape-server contemplates servicing an RST request for operations. It first checks for an "all caught up" flag to be introduced later. If that flag is not set, tape-server requests a one-second sleep followed by return to the conditional read; this allows lower-priority servers to operate. If the "all caught up" flag is set, the server then checks a "rewind imminent" flag it maintains. If a rewind is imminent (within one minute) the same sleep as above is entered.

The point of all of the foregoing is that tape-server considers writing seismometer data to tape as its primary function; only when no such work is available or expected in the near future will RST requests be honored. The rewind check arises because during the one minute it takes the tape driver to complete a rewind there will certainly be a backup of seismometer data waiting to be recorded; to avoid adding to this backup, tape-driver refuses to get involved in an RST operation.

In spite of all of these obstacles, tape-server will get to this point in contemplating RST operations every ten seconds or so, except for gaps of a few minutes every two hours when a rewind is performed on the master tape. It now checks to see whether there is an uncompleted rewind from the prior RST operation. When the RST requests a rewind of its tape, tape-server changes the command if necessary to a "start rewind" before passing it on to the tape driver; this assures that the system will never be tied up waiting for an RST rewind. If that was done, however, tape-server must remember, and at this point check for completion (another command to the tape driver). If the operation is incomplete, control returns to the conditional read back at the start of this subsection. Otherwise the operation-complete message covered later is returned to the RST-server before going once again to the conditional read.

We reach this point if there is no RST rewind in process. Only now is a check made to see whether there is in fact an RST request pending. This check is made by a conditional read of the RST-to-tape queue. If that queue is empty then tape-server goes to the sleep step above. Otherwise, the requested operation is passed on to the tape driver (with a rewind accorded the special treatment mentioned). When the driver returns, tape-server notifies RST-server by writing a message to the tape-to-rst queue and once more goes back to the conditional read.

A final facet of tape-server must be considered, namely the control of retrys. The tape driver returns a status response indicating whether it has been successful in carrying out the assigned operation. Tape-server checks this flag and, in case of failure, repeats the operation if and only if it was not an RST request. A failed RST request simply gives a change in the message written back to RST-server; it is the responsibility of the RST to initiate a retry. The point of this is to force the RST retry to fight its way back through all of the interlocks that assure seismometer data the highest priority.

RST-SERVER

This server waits on a semaphore which is set when a special "end of message" byte has been read by the interrupt handler for the RST-input parallel port. RST-server checks the message, which may be simply a request for the time of day that can be answered without help from tape-server. Otherwise, "all caught up" and "rewind imminent" are checked, and the request refused at this point without bothering tape server if that is indicated. In all of these cases an immediate answer is sent to the RST by firing up the RST-output parallel port driver.

Assuming all is well, the requested operation is checked to see whether it is a read. If so, a conditional read of the tape-blocks queue is made to obtain a buffer to hold the result. If no tape-blocks are available there is again an immediate return of a refusal message. Otherwise a pointer to the request message is written to the RST-to-tape queue and this server does an unconditional read of the tape-to-RST queue, thereby releasing until tape-server has finished, as covered earlier.

When control is re-obtained, RST-server sends a response message, and possibly a tape block of data, back to the RST. It returns any tape block used (by writing to the tape-block queue), then restarts the RST input parallel port and awaits another

semaphore

SEISMO-READ SERVER

This server always "owns" two seismo blocks that it has obtained from the seismo-blocks queue. The pointer to one block is given to the seismo parallel input port interrupt handler, which when that block is filled switches automatically to the other block and activates a semaphore. This server is waiting on the semaphore. When received, it reads the seismo-blocks queue to get a new empty block, and posts the full block to the seismo-processing server by writing its pointer to the seismo-input queue.

SEISMO-PROCESSING SERVER

This server always owns a tape block that it is in the process of filling. It reads the seismo-input queue to obtain a new batch of data. If the new data would not overfill the tape block then it is simply written in, preceded by a sixteen-byte header.

If the new data would overfill the current tape block, a crc for the block is calculated and placed in its last two bytes, then the block is posted to the tape server by writing its pointer into the seismo-to-tape queue. A fresh tape block pointer is obtained from the tape-blocks queue, the new inputs are entered, then control returns to the check of the seismo-input queue.

## I N T E R R U P T S

The four parallel ports (two to/from the RST and two to/from the Seismo Processor) have an interrupt each. The Tape Unit serial port has one interrupt assigned, as does a one-millisecond clock pulse generator. (This clock has nothing to do with the time of day clock in the Seismo Processor; it is obtained by dividing down outputs of the baud-rate generator of the serial port, and is used only to assist the operating system in its "sleep" functions.)

Tape interrupt handling is done by the tape driver subroutine, which enables and disables various possible sources of the interrupt on its own. In order to implement the interface described earlier both parallel output port handlers are trained to access a buffer and send the next byte if one is available, otherwise set a semaphore and release without restarting the port. (A port is started by a server originating some message.) The RST input port operates in the same

A SEISMOMETER RECORDING SYSTEM FOR THE REMOTE SEISMIC TERMINAL

Mellner

Seismo Processor parallel input is continuous. A semaphore is set at end-of-buffer but then an automatic buffer switch is made and the port restarted. The only throttling that occurs results from a temporary masking off of the interrupt. This is done during rewinds to avoid exhausting the seismo-blocks queue; the Seismo Processor is forced to hold its output at those times, and is equipped with a large memory to do so.

The Seismo-Processor-parallel-input interrupt handler also senses the special "all caught up" byte (CC hex) sent on occasion by the Seismo Processor. That byte is not placed in the buffer, instead a semaphore is activated and the "caught up" flag is set. The semaphore is sensed by a server not previously mentioned, called the devourer. Devourer waits on the caught-up semaphore, when that is received devourer sleeps for eight tenths of a second, then resets the "caught up" flag and returns to waiting for the semaphore. The function of devourer is to prevent "caught up" from remaining true for more than a brief interval.

Receipt of the "all caught up" byte also causes the next few input bytes to be saved in a special buffer; those bytes will be the full time report sent every second by the Seismo Processor, and are used by RST-server to respond to time-of-day requests as mentioned.

## In Retrospect

Work on the SRS continues. Mention has been made of the plan to insert an accurate time-reference system, there is also talk of building in an event detector, several other improvements and enhancements are under consideration. The purpose of this section, however, is to pretend that the project is complete and make some comments based on the famous 20/20 hindsight.

### TAPE UNIT INTERFACE

Selection of a serial interface turned out to be penny wise and pound foolish. A separate Tape Processor (another micro) with an RS-422 parallel interface to the I/O Processor (identical to the RST interface) would provide a cleaner system with more capability, while still giving a well-defined if not so readily-available interface for future use. As things stand now, large chunks of code in the I/O Processor will have to be duplicated in any future system wishing to use the tape. In a word, the tape driver should have been exported to another microcomputer. This can be put in a more general context, it is felt. *All peripherals, floppys, tapes, printers* or whatever, should be built as intelligent subsystems containing their own processors; drivers for peripherals should degenerate into simple high-level-command issuers ("Here is the data, do your thing and call me when done".)

### REAL-TIME OPERATING SYSTEM

The standard package purchased got the job done, but is now a liability. For one thing, it is not ROMable; if it were, the SRS could be made to stand alone without requiring a floppy subsystem to be added. Further, the absence of source code prevents cutting the system down to the very little that is actually used during operation. There are also several known bugs that have to be avoided because they can't be tracked down and fixed. The lesson here (learned by the writer years ago but ignored under time pressure) is that a real-time system must be coded from scratch, large pieces of source code can be pillaged from other jobs, but "black boxes" are to be eschewed.

A SEISMOMETER RECORDING SYSTEM FOR THE REMOTE SEISMIC TERMINAL

Acknowledgment

Throughout this project the writer had the benefit of a continuous exchange of ideas with A. G. Gann. Work was performed under contracts with the MIT Lincoln Laboratory and Rondout Inc of Stoneridge NY, they in turn were funded by DARPA (Defense Advanced Research Projects Agency).

Reference

1) Remote Seismic Terminal, Alvin G. Gann and Richard M. Moroney

IASPEI Conference Proceedings, London Ontario 1981

APPENDIX D

## APPENDIX D

TYPE RONWORK NOT
Notes on using RONWORK (simply type 'RONWORK' in response to the A> prompt.)

This is a FORTH system, including editor and whatnot. For present purposes, however, the following words are all that need be known.

REWIND
Rewinds the off-line tape to prepare for operations thereon.

TRAC"n
Sets the track number n of interest, n = 0, 1, 2, or 3

READ-FIRST-BLOCK
Form of read command to use after a rewind. Reads the first block from the off-line tape at the track selected (TRACK defaults to 0).

READ-LATER-BLOCK
Form of read command for all but the first read of a track. Both of these commands will report status if the read was unsuccessful for any reason, otherwise they will plot the short-period data.

n TAPE-BUFFER-SAVE
Save the tape buffer (filled by the last successful read) on screens n, n+1 and n+2 of the current screen file.

n TAPE-BUFFER-FETCH
Load the tape buffer from screens n, n+1 and n+2.

PLOT-SHORT
Plot the short-period data in the tape buffer.

CHECK-SHORT
Verify the CRC of the tape buffer, then do PLOT-SHORT. Successful reads do a CHECK-SHORT automatically.

BACKSPACE
Backspace the active tape one record. (Used to retry if a tape error occurred.)

SKIP-FILE
Skips forward to the next file mark (these are written every 5 minutes).

CUR-LOOK
Import the next record written on the one-line tape, otherwise the same as any other read.

n EXTRACT-MEDIUM
Assumes that ten tape buffers have been saved starting at screen n, that is a total of 30 screens. Goes through these and extracts the medium-period data, storing it in the medium buffer.

PLOT-MEDIUM
Plot the data in the medium buffer.

n MEDIUM-BUFFER-SAVE
Save the medium buffer on screens n, n+1 and n+2.

n MEDIUM-BUFFER-RESTORE
Load the medium buffer from screens n, n+1, n+2.

REPORT-START-TIME
Type the header of the first one-second block in the tape buffer.

The following abbreviations are active

| | |
|---|---|
| RS | REPORT-START-TIME |
| RLB | READ-LATER-BLOCK |
| BS | BACKSPACE |
| CK | CHECK-SHORT |
| SB? | Dump the status buffer for examination. |

LIST-MEDIUM
Print out the medium buffer in volts on the lineprinter.

MODEM OPERATIONS

To enter the modem mode, type 1 LOAD, then DUMTER1200. ESCAPE escapes back to regular mode, $ sends the medium buffer out over the communications port in ASCII (and LIST-MEDIUM is not to be used!)