

1-1-1977

An application of microprocessor technology to remote station analysis of seismic signals.

Robert Gregory Novas

Follow this and additional works at: <http://preserve.lehigh.edu/etd>



Part of the [Computer Sciences Commons](#)

Recommended Citation

Novas, Robert Gregory, "An application of microprocessor technology to remote station analysis of seismic signals." (1977). *Theses and Dissertations*. Paper 2253.

AN APPLICATION OF MICROPROCESSOR TECHNOLOGY
TO REMOTE STATION ANALYSIS OF SEISMIC SIGNALS

by

Robert Gregory Novas

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Information Science

Lehigh University

1977

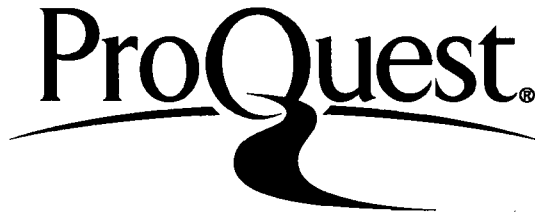
ProQuest Number: EP76529

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



ProQuest EP76529

Published by ProQuest LLC (2015). Copyright of the Dissertation is held by the Author.

All rights reserved.

This work is protected against unauthorized copying under Title 17, United States Code
Microform Edition © ProQuest LLC.

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science.

September 13, 1977
(date)

Professor in Charge

Chairman of Department

ACKNOWLEDGEMENTS

Assistance from the following sources is gratefully acknowledged:

- Geophysics Branch, NASA, Goddard Space Flight Center, Greenbelt, Maryland for purchasing the analog-to-digital and microcomputer hardware.
- Microcomputer Development Laboratory, NASA, GSFC, for use of the INTEL MDS, and helpful advice.
- Will Webster (NASA, GSFC) for the original suggestion, advice and criticism; Bob Barnes for encouragement from afar; and my wife Charlie for typing and faith.

TABLE OF CONTENTS

	<u>Page</u>
ABSTRACT	1
1. INTRODUCTION	3
2. SYSTEM OVERVIEW	6
3. ANALOG-TO-DIGITAL CONVERSION SUBSYSTEM	8
Description	8
Implementation	11
4. DIGITAL SUBSYSTEM HARDWARE SELECTION	12
Microprocessor and Microcomputer	12
Microcomputer Applications	13
Microprocessor Directions	15
Factors Considered in the Microprocessor Selection	17
Data Bus Size	18
Address Bus Size	18
Number of Program Useable Registers	18
Size of Instruction Set/Number of Branch Instructions	19
Instruction Execution Time	19
Multiply/Divide Time	20
Push Down Stack Capability	20
Interrupt Capability	21
DMA Capability	23
Available Software Support	23
Available Hardware Support	24
Simplicity of Hardware Implementation	25
Power Supply Requirements	26
Technology	26
Microcomputer Comparison	27
Intel 8080	27
Motorola 6800	28
Signetics 2650	29
National PACE	31
National IMP-16L	32
Intel 3000	32
Microprocessor Selection	33
5. SOFTWARE SUBSYSTEM	35
Requirements Analysis	35
Data Acquisition	35
Event Detector Algorithm	37
Display Subsystem	38

	<u>Page</u>
5. SOFTWARE SUBSYSTEM (cont'd)	
Software Description	38
Overview	38
Data Acquisition Task	39
Earthquake Detector Task	44
Output Subsystem	44
Implementation	48
6. CONCLUSIONS	50
FOOTNOTE REFERENCES	52
ADDITIONAL REFERENCES	53
APPENDIX A: ANALOG-TO-DIGITAL CONVERSION SUBSYSTEM SCHEMATIC DIAGRAM	55
APPENDIX B: SOFTWARE MULTIPLY/DIVIDE ROUTINE CODING	59
APPENDIX C: MICROPROCESSOR COMPARISON SHEETS	68
APPENDIX D: ALGORITHM ANALYSIS	75
APPENDIX E: VITA	80

LIST OF FIGURES

1. System Block Diagram	7
2. Analog-to-Digital Conversion Subsystem	9
3. Microprocessor Selection Factors	17
4. Altair 8800 Microcomputer	34
5. Data Acquisition Task Interrupt Handler	40
6. A-D Data Processor	40
7. FIFO Load Routine	42
8. Gain Adjust Algorithm	43
9. Earthquake Detection Task Control Routine	45
10. Earthquake Detector Routine	46
11. Output Subsystem Display - Altair Very Low Cost Terminal	47
12. Algorithm Equations	77
13. Digital Filter Diagram	78

AN APPLICATION OF MICROPROCESSOR TECHNOLOGY
TO REMOTE STATION ANALYSIS OF SEISMIC SIGNALS

by Robert Gregory Novas

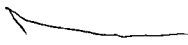
This thesis describes research into and implementation of microprocessor technology to minimize two present earthquake precursor monitoring limitations: noisy and sometimes unreliable United States Geological Survey (USGS) transmission to a central processing site via ground links, and the impossibility of National Aeronautics and Space Administration (NASA) satellites' relaying microearthquake data in real-time due to data volume. Although NASA routinely relays information from remote data collection platforms (DCP's) by satellite systems, existing systems' capabilities are far exceeded by the requirements of a USGS monitoring network.

A project to design and construct a microcomputer controlled testbed earthquake precursor detection system was approved and funded by the Geophysics Branch of NASA, Goddard Space Flight Center, Greenbelt, Maryland, and was undertaken for submittal as a thesis. The system function was to pre-process seismometer data to reduce the bandwidth required to convey earthquake precursor information.

The project initially surveyed the various microcomputers available to select a suitable hardware system. Necessary hardware, including an 8080 based microcomputer system, was procured. A testbed software system was implemented which controls collecting data from the seismometer, pre-setting data to a detection algorithm, and outputting results. The

system allows important algorithm parameters to be modified in real-time for experimental purposes.

Additionally, a USGS algorithm was implemented and installed in the testbed. The thesis describes this effort, documents the system implemented and demonstrates the feasibility and practicality of microprocessor technology in this application.



1. INTRODUCTION

Earthquakes are a serious natural hazard which have caused loss of life and property damage with little or no advance warning. Research by the U.S. Geological Survey, as well as by Japanese, Russian and Chinese agencies, has shown that by monitoring geophysical activity in earthquake prone areas, it is possible to accurately predict the place, time and magnitude of an earthquake.¹

Earthquake prediction is based upon measuring events termed pre-cursors, such as the relative frequency and duration of microearthquakes, or disturbances in ground tilt, magnetic field, or ground water radon gas content. The USGS has found that conventional remote data acquisition schemes to collect these data via telephone lines and local radio links are costly and unreliable, especially during earthquake activity. As a result, the Earthquake Precursor Monitor System has been proposed to merge the geophysical work of the USGS with the satellite relay technology of NASA, to relay data from remote monitoring sites to a central processing location by means of artificial satellite.

Initially, stations will be deployed in arrays around active earthquake faults in the San Jocino Valley of California and the southern coast of Alaska. Data collected at each station will be transmitted by satellite to a central earthquake prediction site. Each station is envisioned as containing a satellite uplink transmitter and controller, called a data collection platform, an input processor which collects the various instrument readings, and the actual instruments. A typical instrument complement would be:

- one or more seismometers;
- two tiltmeters;
- a differencing magnetometer; and
- a tidal gauge, well level meter or radon gas detector.

An immediate problem in implementing this system is that current and foreseen satellite relay systems cannot handle the expected volume of raw data generated by the station arrays. While most of the sensors need to be sampled only occasionally, producing at most a few hundred bits of information every six hours, the seismometers generate perhaps one megabit of data per precursor event. Since many seismic precursor events precede an earthquake, the total data volume can be enormous. The USGS believes the important information contained in the seismic data flow could be conveyed in several hundred bits if data processing could be done at the instrument.

In addition, the problem of instrument control and calibration is of importance. A control command system, linking the central site back to the remote stations could be implemented, but at great expense. Tasks such as gain changing, instrument calibration, and status checking have been accomplished in the past by separate control lines, periodic visits by site inspection teams, or detailed data inspection. All of these methods are either expensive or unreliable.

The need to solve these problems has led to the conclusion that a small computer system at each remote site must be part of the input processor. Experience with telemetered data has shown that a computer can reduce the volume of seismic data to manageable proportions as well as handle instrument control. The newly developed microprocessor technology

is ideal for this application because of its sophistication, size, and low cost. Additionally, the control of microcomputers by programs in read-only memory storage provides flexibility unmatched by hardwired data processors and controllers, and at less expense. The control and processing tasks can be adapted to suit the environment of a particular station without expensive and difficult hardware modification.

The intent of this thesis is to demonstrate the applicability of microprocessor technology to the solution of these types of problems. In particular, a testbed system for processing raw seismic data is described. This system was constructed as a prototype, and evaluated. The testbed system incorporates a USGS microearthquake detector algorithm implemented in software, and so requires the processing capabilities of a computer. The thesis includes:

- Definition of a suitable system.
- Requirements analysis, design, procurement, and implementation of the component subsystems.

2. SYSTEM OVERVIEW

The Microprocessor Earthquake Precursor Detector System contains components to:

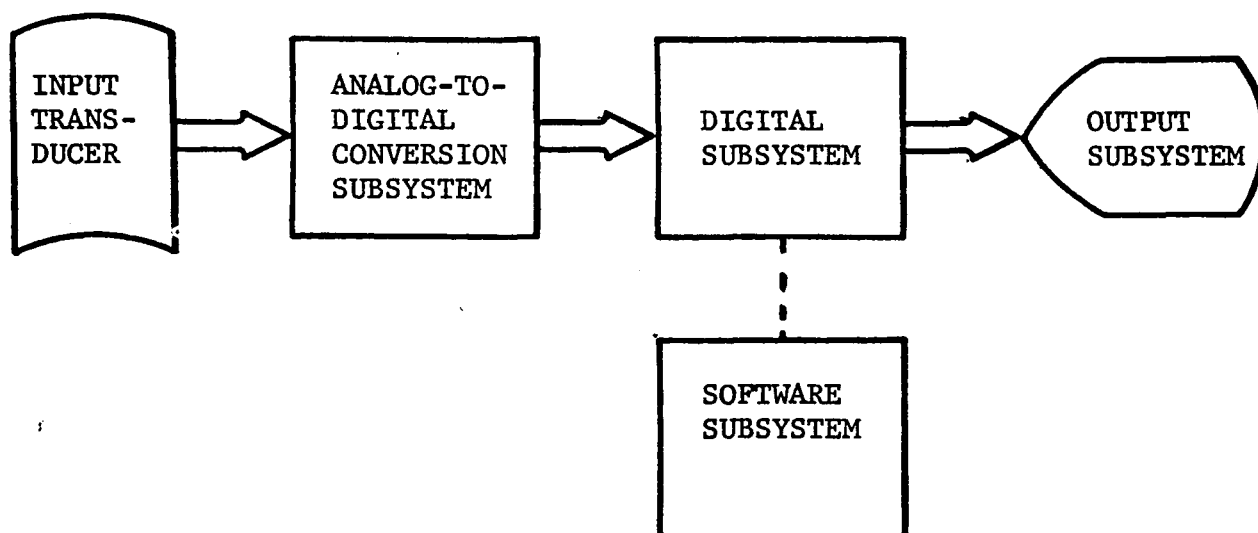
- detect a seismic signal in the presence of environmental noise,
- convert it to a form suitable for digital processing,
- perform digital processing as directed by an algorithm, and
- display the results.

The system block diagram of Figure 1 shows the subsystems which perform these functions.

Each subsystem is presented in detail in the following sections. Briefly, the analog signal output by a seismometer is the system input. The interface requirements of this device, and of the digital subsystem's hardware and software sections are the functional requirements for the analog-to-digital conversion subsystem. This subsystem performs the necessary signal conversion to allow digital signal processing. The digital subsystem hardware (a microcomputer), directed by the subsystem software, executes the microearthquake detector algorithm. The results of this processing are displayed by the output subsystem.

Figure 1.

SYSTEM BLOCK DIAGRAM



3. ANALOG-TO-DIGITAL CONVERSION SUBSYSTEM

Description

This subsystem converts the analog voltage produced by a seismometer to a weighted binary number. A block diagram of the subsystem is shown in Figure 2.

The seismometer used is a Mark Products Incorporated L4A Geophone, with 500 ohm coil. This unit, when properly damped, has a useable frequency response from 2 Hz to beyond 100 Hz. It is commonly used in explosion seismology and seismic array systems.

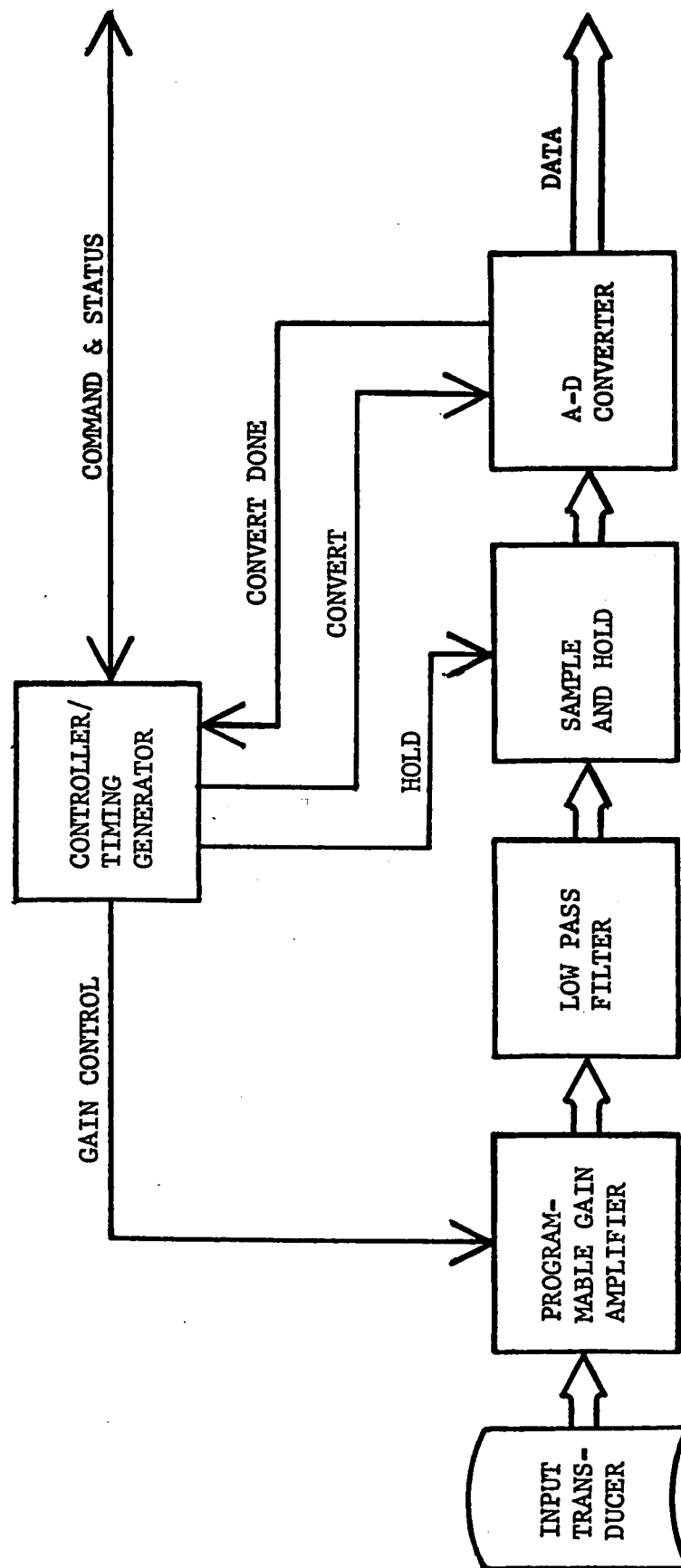
The electrical balanced output of the geophone is applied to a Burr Brown 3600 programmable gain instrument amplifier. This device has the following characteristics:

- high impedance differential (balanced) input,
- high common mode rejection ratio,
- temperature stability,
- low noise,
- programmable gain,
- gain accuracy and linearity, and
- low impedance unbalanced output.

The balanced line interface between the geophone and the instrument amplifier, coupled with the high common mode rejection ratio of the instrument amplifier insure a high input electrical noise rejection capability. The instrument amplifier converts the balanced geophone signal to the unbalanced format used by the succeeding stages of the subsystem.

Figure 2.

ANALOG-TO-DIGITAL CONVERSION SUBSYSTEM



The instrument amplifier selected also has a gain (amplification factor) programmable from 0 to 3840. This was considered essential for expanding the dynamic range. As explained in the Software Subsystem description, this feature was used to provide a system dynamic range of 60 db.

The signal is next input to a Burr Brown ATF76-L8MC-35R0 Low Pass Filter. This filter has an eight pole, low ripple Chebyshev response, with a 35 hertz cutoff frequency. The filter is necessary for proper operation of the analog-to-digital converter. The frequency of signals input to the converter must be less than half the A-D conversion frequency of 80 Hz. The 35 Hz cutoff frequency was chosen to allow a generous range of seismic signals, while filtering unwanted high frequency environmental and electrical noise.

Next, the signal is input to a DATEL SHM-1 sample and hold. This device samples the signal at the conversion frequency, with an aperture time or sampling window of 50 nanoseconds. The value acquired during the sample period is held for the remainder of the conversion interval. This device improves the accuracy of the worst case analog-to-digital conversion from 1 percent of the full scale signal value to 0.001 percent.

The signal value held by the sample and hold is input to a DATEL ADC-K10 10-bit analog-to-digital converter. This converter produces a 10-bit two's complement binary number every conversion period. The result is directly proportional to the sign and magnitude of the sampled signal, weighted by the gain of the programmable gain amplifier.

The controller/timing generator component of the subsystem generates the conversion frequency clock signals, and interfaces the subsystem with the digital subsystem. A crystal controlled clock oscillator and digital

logic division circuitry produce the 80 Hz system clock. The system interface consists of a 6-bit gain control bus, a 10-bit data bus, and a convert done control line.

Implementation

Appendix A presents a schematic diagram of this subsystem. The subsystem components are assembled on a perforated VECTOR board mounted in a 3 x 5 x 12 inch cabinet. The subsystem has the following power requirements:

+ 5 v	at	350 milliamps.
+15 v	at	100 milliamps.
-15 v	at	100 milliamps.

A concern with this subsystem was the need to adjust and calibrate the electronics. The interface circuit was designed so that the Altair Very Low Cost Terminal (VLCT) could substitute for the digital subsystem. In this configuration, the VLCT keyboard commands the programmable gain amplifier gain factor, and the VLCT display presents the 8 most significant bits output by the A-D converter.

This mode allows calibration of the system. It also allows verification that the subsystem is functioning properly, independently from the other subsystems. This mode was used to debug the hardware. It will be used operationally to verify the subsystem, and, periodically, to recalibrate it.

4. DIGITAL SUBSYSTEM HARDWARE SELECTION

This section presents a review of six microprocessors, which were either available or announced as soon to be available in late 1975. While the microprocessor field was not exhaustively studied, these units represent a sampling of three major technologies used in producing large-scale integrated (LSI) circuits. The microprocessor selection for this project was made on the basis of this review. Before the individual microprocessors are discussed, a brief description of microprocessors and microcomputers is in order.

Microprocessor and Microcomputer

A microprocessor is generally considered to be an implementation of a computer system's Central Processor Unit (CPU) on a small number of LSI circuits or chips. A CPU is the portion of a computer that contains the control and computational circuitry to execute a program residing in the computer's memory. A simple stored program computer structure has an input/output (I/O) unit for communication with the outside world, a CPU, and memory unit. Instructions for the computer are contained in the memory, and specify operations that occur between the I/O unit and the CPU, or the CPU and the memory, or elements internal to the CPU. In this simple structure, the I/O unit and memory cannot directly interact, but must communicate through the CPU.

A single chip microprocessor can be contained in a 2.1" by 0.514" LSI package. The package may have as many as 40 connector pins which

serve to interface to the rest of the system. A multi-chip (or chip set) microprocessor may consist of four to ten LSI packages connected externally to provide similar or superior performance to a single chip unit. Although the more complex chip set microprocessors are faster and have greater processing capabilities, a penalty is paid in material cost, design complexity, power requirements, and perhaps reduced reliability.

A microcomputer is defined as a computer built using a microprocessor as the CPU. A CPU can be the most complex part of a computer system; in a microcomputer it can be as small as a single LSI chip. While it is not easy to draw a functional distinction between microcomputers, minicomputers, and large-scale computers, several characteristics of microcomputer applications set them apart.

Microcomputer Applications

A microcomputer can serve as a sophisticated device controller, replacing hardware logic circuitry. In comparison to hardwired logic, a microcomputer controller is more versatile, and often smaller and less expensive. An example is the Intelligent Typewriter System² based on the Signetics 2650 microprocessor. This one hundred dollar, six package microcomputer system replaces a several hundred dollar, 75 package circuit, built of medium scale integration (MSI) parts. The microcomputer system coupled with an I/O device performs as a basic typewriter, but additionally contains a character memory and text editing capabilities. These features allow typing to be entered in rough form, edited, and then finish copy typed in one operation. Since the system is actually

programmed on only one of the six microcomputer packages, a modification to the features of the system is made by replacing that one package.

Many scientific applications could use the advantages of computer-based control but cannot bear the price of a dedicated minicomputer, which can cost \$50,000 or more. The low cost of microcomputers allows them to be used in such dedicated applications where a full-size minicomputer would not be cost-effective.

A microcomputer used in this fashion, dedicated to a particular task, stores its instructions in permanently programmed (non-volatile) read only memory (ROM). This form of instruction storage would be impractical for mini- or large-scale computers. Such storage is ideal for an unattended dedicated computer, as it means that a microcomputer system can survive power failures with no loss of program. A microcomputer can be programmed to automatically re-initialize on power-up. Operator intervention, even in unusual circumstances, is unnecessary.

Other applications exist in which the power of the microcomputer seems far greater than required by the simplicity of the application. A traditional minicomputer user might feel that in this type application, a microcomputer would be an unsuitable choice of solutions. However, the criteria governing the choice of solution can now be the versatility and simplicity of the microcomputer, rather than the expense, as that will be small.

Microcomputer applications can be distinguished from hardwired or minicomputer applications by speed. Microcomputers often do not have the speed necessary for truly high speed operations. This is very apparent in the single chip microprocessors reviewed. These not only are slow,

but are hampered by limited (8-bit) data paths. Operations with data larger than the data paths allow must be done in repeated, smaller steps. This restricts simple microprocessors from applications requiring high speed real-time data processing.

All of the features of microcomputers, including others not discussed such as low power requirements, imply that a microcomputer is a feasible alternative to fixed control logic in unattended field instrument packages. The low cost incurred allows microcomputers to be considered for other applications which previously could not afford automatic control of any type. It is just this low cost versatile control that this project is hoping to exploit.

Microprocessor Directions

The microprocessor field has split in several directions. In one aspect, the trend is toward smaller package count, dedicated controller units. As an example, the Rockwell PPS-4/1 chip is a 4-bit microprocessor with 50 instructions, 10,752 bits of program storage read only memory, 384 bits of data storage read/write memory, and 31 input/output ports. This unit is a complete microcomputer on a chip.³ The PPS-4/1 costs less than ten dollars, and represents an effort to reduce the "overkill" situation mentioned previously. A characteristic of this branch is the high relative proportion of I/O capability to computational capability.

A different direction is represented by the single chip 8- or 16-bit microprocessor which requires additional memory and I/O support to

function as a microcomputer. While not yet up to the level of minicomputer capability, this variety of unit is much less restricted in memory capability than the microcontroller. The personality of the chip - its instruction set, I/O, and control capabilities - is set by the manufacturer. This type of chip generally is structured to look very much like a minicomputer, and has replaced the minicomputer in some nondemanding applications.

The third direction is that taken by the chip set microprocessors, which can compete with minicomputers. In one alternative, the data manipulating capabilities of the CPU are broken into 2- or 4-bit processing elements (each on a single chip) called bit slices. These can be configured in as wide a word size microprocessor as desired by paralleling the bit slices. These units require the additional support of control read only memory, to contain the microprograms determining the interpretation of machine instructions, and a control sequencing unit, which steps through the microprogram in the execution of a single instruction cycle. These units allow the personality of the machine to be custom-tailored for an application by changing the microprogram in control memory. Microprogramming is a very powerful feature. For example, it can permit the exact emulation of other computers. It is a mixed blessing, however, as it adds another level of complexity to the program design.⁴

Another alternative exists, in which the microprocessor is split into functional blocks. These can be configured, building block fashion, to produce a system of the required capacity. This type is restricted to a particular word size and personality by the central processor component.

It is expandable in terms of memory, I/O, or interrupt capability, with additional components.⁵

Factors Considered in the Microprocessor Selection

The factors used to compare the six microprocessors reviewed here are listed in Figure 3, and discussed in this section.

Figure 3.

MICROPROCESSOR SELECTION FACTORS

1. Data Bus Size
2. Address Bus Size
3. Number of Program Useable Registers
4. Size of Instruction Set/
Number of Branch Instructions
5. Instruction Execution Time
6. Multiply/Divide Time
7. Push Down Stack Capability
8. Interrupt Capability
9. DMA Capability
10. Available Software Support
11. Available Hardware Support
12. Simplicity of Hardware
Implementation
13. Power Supply Requirements
14. Technology

Data Bus Size

The size of the data bus determines the basic word size of the microprocessor. Only 8- and 16-bit word size machines were considered for this application because at least 8-bit precision was required for the input data. An arithmetic operation performed in one instruction upon two's complement data can generate a result that does not fit in the word size of the machine. An arithmetic operation performed to greater significance requires multiple precision, implemented as a number of machine instructions, which costs space and time. The 16-bit microprocessors have an obvious advantage in any analysis application.

Address Bus Size

The memory capacity of a microcomputer is determined by the number of bits it can use as an address in accessing memory. The address bus size, except in the case where paging is used, is fixed by the number of address pins physically supplied on the microprocessor chip. An 8-bit microprocessor generally has 15 or 16 pins for this purpose, and so can directly access about 32 or 65 thousand 8-bit words of data and program storage. This is generally more than adequate.

Number of Program Useable Registers

In a single address computer (all the microprocessors considered here are single address units) one memory location can be specified by one instruction. Any data manipulation takes place between this memory location and a CPU register. In the simplest case, only one register is available, called the accumulator. More convenient machines have other registers available for use in data access or manipulation.

An important factor in the power of a microprocessor is the number and type of additional registers. Next in power to an accumulator are the general purpose registers. These can be used for logical and arithmetic operations upon data, but are more restricted in scope than the accumulator. The least powerful registers for direct data manipulation are the index registers. These are used, in conjunction with the accumulator, for referencing memory locations. In some machines, a general purpose register can also be used as an index register.

Size of Instruction Set/Number of Branch Instructions

The number of different instructions available is a factor in solving an application problem. For example, if a single instruction multiply or divide is not available, the operation will probably be performed in a subroutine, consisting of many instructions, which take more room and time to execute. Additionally, because any software routine requires conditional branch capability, a machine with a simple branch instruction set may result in cumbersome coding and slow execution speed. A computer's decision making power comes from its conditional branch instructions. Thus, such instructions must be considered important in the instruction set.

Instruction Execution Time

The time required for the execution of a single instruction varies from one instruction to another. Although the clock frequency provides a basic indication of microprocessor speed, each instruction requires a different number of integral clock cycles to execute. The instruction

execution time is best specified as a minimum/maximum range, and provides some indication of expected processing speed.

Multiply/Divide Time

Most microprocessors do not have multiply or divide instructions provided. Software multiply and divide subroutines were coded and hand timed at maximum clock frequencies to obtain a problem execution time for the microprocessors examined here. This time provides a fairer comparison of different microprocessors, as it combines the power of the instruction set with the instruction execution time.

Listings of the code written appear in Appendix B. Since an assembler was not available, the code may not be error free. The logic has been carefully inspected, and the timing and memory requirements should be very close to actual.

A problem with this technique is that the same algorithm was coded using the various assembly languages. This may not be the optimal algorithm for a particular machine. The coding task did provide an opportunity to gain familiarity with each instruction set.

Push Down Stack Capability

A push down stack is a very useful programming mechanism. For example, a re-entrant routine is program code which may be executed on a shared basis by several levels of program. More than one program level may be using a re-entrant routine concurrently in interrupt driven processes. A push down stack is a necessary means for data storage in this case. A push down stack is also desirable for subroutine call nesting. The stack provides extremely convenient return address storage.

Some microprocessors provide an on-chip stack of limited size useable only for return address storage. This feature has been claimed to be an advantage useful in simple applications. If a system can be implemented with all data manipulation taking place within microprocessor registers, no read/write memory is required for data storage. The return address stack provides a means for nesting subroutine calls, and so the system can be configured with only ROM. This is a restrictive technique in that it provides little room for future system enhancement.

Other microprocessors implement a stack pointer and stack instructions on chip, and require the stack to be in memory. This technique provides the most versatile capability as it allows for return address and data storage, limited only by the amount of memory available.

Interrupt Capability

An interrupt is a signal generated external to the current processing task which causes a computer to execute a different section of program, generally without disturbing the interrupted section's process flow. Interrupts are usually caused by infrequent, high priority processes which require immediate servicing. Interrupt servicing is commonly related to managing Input or Output operations proceeding asynchronously to the program processing flow. After an interrupt request is serviced, the original processing stream is continued with no detrimental effect.

In most microprocessors, the status of the machine, contained in a program status word, is saved automatically by hardware upon responding to an interrupt. Any registers which will be disturbed by the interrupt

service routine must be specifically saved by the routine. This insures that the interrupted program can be restarted at the end of the interrupt routine processing.

The simplest interrupt structure is a single level interrupt where triggering an input line causes the CPU to transfer to the interrupt service mode. In this mode, the interrupt routine software may need to save the current machine status, and poll the system devices to determine the cause of the interrupt. This is termed a software vectored interrupt. Then a branch to the appropriate service routine is executed. The interrupt processing is terminated by restoring the original machine status and continuing the interrupted task. An alternate interrupt process, termed a hardware vectored interrupt, requires an interrupting device to supply a device specific address called an interrupt vector address. This address is used to branch to a service routine at that location. A similar type of vectored interrupt uses the interrupt level to determine the location of an interrupt vector address in memory. The next microprocessor instruction is executed at the location contained in the interrupt vector address. The first technique allows several devices on a common interrupt line to vector to unique service routine addresses. The second technique allows the program to specify the interrupt level service routine address, and is common on multi-level interrupt systems.

When a single interrupt level may be activated by several different devices desiring services, and each device service request is of varying urgency, the interrupt routine may be coded so as to allow concurrent processing to occur. This would then be a re-entrant interrupt, as it

could be entered several times before completing the initial service request. In this case, it would be necessary to preserve the machine status on a push down stack.

DMA Capability

DMA or Direct Memory Access capability is necessary if a device is to access memory without using the CPU as an intermediary. During the device's memory access, the CPU must be locked out of memory to avoid address or data bus conflicts (the device requiring a bit to be high, the CPU, low). Since the CPU accesses memory relatively infrequently, and can operate internally without accessing memory for some time, DMA can allow devices to use memory and only minimally impact CPU execution speed. Once a microprocessor system is operational, the necessity of DMA capability is determined by the I/O requirements of the application, and may be superfluous. However, while the same system is in the debugging phase, a DMA capability is required for looking at the contents of memory from a debug console or front panel. Otherwise, there is no visibility into the memory which does not disturb the state of the CPU. This lack of visibility is the major problem in microprocessor system development, as it severely hinders the debugging of the system.

Available Software Support

Any system support available "off the shelf" is a definite asset in software development. Assemblers, compilers, editors, loaders, and program debugging software are necessary for application program development. Manufacturers may provide resident assemblers, which require the target microprocessor system to run, or cross assemblers which run on a

different, available computer, so that software and hardware development may proceed in parallel. They may also provide a simulator program which can mimic the execution of software intended for a target system on a different computer.

Additionally, a library of tried and tested application routines such as an arithmetic package with multiple precision add, subtract, multiply, and divide is invaluable. A software application which uses debugged routines for processing has a much better chance of quick success than one in which every line of code must be debugged. System software, such as a monitor with peripheral device support, is another valuable support item. Such software is traditionally difficult to implement.

Available Hardware Support

Just as for software, hardware support is necessary for system development. Manufacturing support varies from providing CPU components only, to offering development systems with CRT, teletype, line printer, and floppy disk peripherals. Another tool is the hardware emulator, which is a module plugged into the target system's microprocessor socket. In addition to acting identically to the desired microprocessor, as if it were plugged in the socket, the emulator establishes full debug visibility for elements internal to the microprocessor (it is very difficult to observe operations inside a single chip microprocessor).

It should be noted that the major cost in a microprocessor application development will be "up front" in the development system. The final application package may well be a minimum cost configuration, but the

system development is easiest accomplished on a full development system structured for visibility and debug capability.

Simplicity of Hardware Implementation

Unless an application has complex, high technology requirements, the quickest success will be achieved with the system simplest to implement. This approach is valid if the application falls into the area where a microprocessor represents an overkill solution. As an application becomes more complex, more care must be taken in analyzing application requirements versus hardware capabilities.

A good estimate of hardware complexity can be made by observing the number of additional support components required by a microprocessor to form a microcomputer. A single chip microprocessor examined on this basis may not be simpler to implement than a chip set unit. The single chip microprocessor may require extensive circuitry to interface with the rest of the system, while a well integrated chip set microprocessor may avoid this problem.

Another factor affecting the difficulty of debugging a hardware implementation is whether the microprocessor uses static or dynamic logic. A CPU implemented with static logic can retain its status indefinitely with the machine clock stopped. In this state, the system can be inspected for problems a single cycle at a time, with all logic signals steady. This is simpler than debugging a dynamic CPU which requires continuous clock pulses to retain its state. Equipment such as logic analyzers and in-circuit emulators is available for dynamic systems, and makes this less of a concern.

As an alternative to the hardware development task, a packaged microcomputer, effectively a single module, may be used to reduce hardware implementation complexity.

Power Supply Requirements

The microprocessor power requirements are critical in several ways. In a field installation, a single limited capacity power supply will be used for the field instruments and the microcomputer. A microprocessor must have a low power requirement so as to minimize the drain on the field power supply. A microprocessor with a low power drain that requires several different supply potentials is still undesirable because of the power cost of producing the various voltages. Each additional supply required is also another possible failure point, decreasing the system reliability.

Technology

The "technology" describes the process used to manufacture a microprocessor. Each technology has been optimized for certain capabilities, with resulting trade-offs in other capabilities. For example, the bipolar technology has the fastest execution time, but also the greatest power requirements. N or P channel MOS technologies are slower and less power hungry. A technology has inherent characteristics such as a power-delay product, reliability, radiation hardness, and cost. Differences are due to manufacturer, complexity, or the maturity of a technology. New technologies and advances in old ones quickly change the microprocessor picture, and this report does not consider this area in great detail.

Microcomputer Comparison

The six microprocessors were evaluated on the points presented in the above section. A discussion of each microprocessor is given in this section. Appendix C provides a quick comparison reference to the microprocessors in a standard format. Appendix B contains the multiply and divide subroutine coding used for the multiply/divide time measurement.

Intel 8080

The 8080 is currently the most popular 8-bit microprocessor.⁶ The unit results from enhancements to the original 8008 microprocessor, and is available with further improvement as the 8080A.

The processor registers directly available for program use are an 8-bit accumulator, six 8-bit general purpose registers, and a 16-bit last-in/first-out (push down) stack pointer. Most data manipulation is performed by the accumulator. An important set of single byte instructions specify operations occurring between the accumulator and a 16-bit memory reference address. In these instructions, two general purpose registers are used to specify the 16-bit memory reference address. Otherwise, a three byte instruction would be required to contain the instruction operation code and the memory address.

The push down stack addressed by the stack pointer register can reside anywhere in read/write memory. The stack is used by the processor for return address storage in subroutine calls and next instruction address storage in interrupt service procedures. The program may also use the stack for register and data storage. This is a very versatile and powerful feature of the microprocessor.

The processor provides a single interrupt request input. When an interrupt request is acknowledged by the processor, the interrupting device must specify one of eight fixed vector addresses. The processor pushes the program status onto the stack, and obtains its next instruction at the vector address location. If more than eight levels of interrupt are to be serviced, the software must vector to the service routine.

Extensive software and hardware support is available from the manufacturer and other sources. Drawbacks to the 8080 are the moderately complex minimum microcomputer configuration, and the three voltage power requirements.

Motorola 680C

The M6800 microprocessor design allows a six package microcomputer implementation. The processor has a simple internal organization with a register complement of two 8-bit accumulators, a 16-bit index register, and a 16-bit stack pointer. Data manipulation may take place between memory and an accumulator, or between the accumulators. The index register is used only for indexed addressing or incrementing.

The four interrupt levels are restart, non-maskable interrupt, software interrupt, and maskable interrupt. Each level is associated with a fixed pair of memory words which contain the interrupt service routine address. There are two interrupts available for I/O device use, only one of which can be inhibited. Software interrupt vectoring is mandatory for any but the simplest I/O requirements. Interrupt recognition, except for the restart interrupt, automatically saves the contents of the program counter, index register, accumulators, and condition code register on the

push down stack. While this totally frees the software from explicitly saving and restoring any machine status, it poses an additional overhead on each interrupt recognition.

The software and hardware support for this chip is not as extensive as for the 8080. Its simplicity and single voltage power requirements still make it a very popular microprocessor. As evidenced by the software coded for the multiply and divide routines, it is also a moderately fast unit.

Signetics 2650

This microprocessor is designed for hardware simplicity. The unit references up to 32K memory on a 15-bit address bus. The two high order address pins are multi-purpose, and are alternately used for I/O control signals during I/O instruction execution. The processor references memory in 8192 byte pages, and a special branch must be executed to traverse page boundaries for either instruction or data access. Any memory reference by an instruction controls the low order 13 bits of the address bus. The two high order address bits stay fixed until modified by a special instruction.

The register complement is one accumulator, and six general purpose registers, accessible in two sets of three. A programmable bit in the machine status register determines which set is active. Another bit in this status register determines if arithmetic operations and shifts are performed with or without carry. A third bit determines if comparisons are made in logical or arithmetic mode. Thus an application which requires both possibilities of a certain bit spends a number of instructions

modifying the machine status register. Other microprocessors resolve these possibilities with distinct instruction operation codes. The double duty instructions used by 2650 allow a smaller instruction set but pay a time penalty when switching modes.

The push down stack is useable only for subroutine or interrupt return address storage. Eight levels are available on-chip, with no provision for expansion. While eight levels are adequate for subroutine nesting, a re-entrant interrupt structure cannot be accommodated. A potentially serious debug problem exists if the push down stack limit is exceeded, as this cannot be detected.

The interrupt scheme allows a device to directly specify 128 vector addresses in a one byte interrupt acknowledge response. Alternately, a two byte indirect response can specify any location in memory as the vector address. This is an unusual capability for a microprocessor. One of the two register banks can be devoted to interrupt servicing tasks, thus speeding the restoration of the original machine status, as only the accumulator and machine status need be saved. The lack of a push down stack suitable for data storage is a disadvantage, however.

The microprocessor has a well developed I/O instruction set including an on-chip serial data interface. There are three alternate methods of accomplishing parallel I/O: non-extended, extended, and memory I/O. In particular, the extended mode allows up to 256 I/O units to be addressed independently from memory. The I/O capability of the chip is very well developed, and includes a special set of I/O control instructions.

This microprocessor was the simplest unit reviewed. A single voltage power requirement, static logic implementation, and on-chip serial interface all contribute to ease of use. The faults of the microprocessor lie in the restrictive on-chip push down stack, and the use of the machine state flags to modify the instruction set, both of which will adversely impact software simplicity.

National PACE

The PACE is a single chip 16-bit microprocessor. In order to fit into a 40-pin package, the memory and data bus are multiplexed onto a common 16-pin set. An additional 4 pins are used to control the bus interface. This immediately requires additional hardware to de-multiplex the address and data lines.

The processor provides four accumulators. One is designated the principal accumulator, and two are useable as index registers. This constitutes a powerful structure, controlled by fixed length single word instructions, very similar to minicomputer architecture.

The push down stack is limited to ten 16-bit entries on-chip. However, a stack full/stack empty interrupt allows software to manipulate entries to and from external memory. The stack is available for data storage, and is a very powerful feature.

The six level priority interrupt structure uses the push down stack for program counter storage. Six fixed locations are defined in memory for interrupt vector addresses. Interrupt processing is ordered in priority by on-chip hardware. This microprocessor has the most complex and powerful interrupt structure of the units reviewed.

Extensive software support is available for this unit as an offshoot of the IMP-16L support. An advantage to this unit is that software development can be done on the versatile IMP-16L development system, and then transferred to actual production microcomputers using the lower performance, lower cost PACE chip. The PACE microcomputer implementation is complex, but very powerful due to the 16-bit data format, and minicomputer chip architecture.

National IMP-16L

The IMP-16L is a packaged microcomputer made from National's IMP microprogrammable bit slice chip set. The basic unit is microprogrammed to be very similar to the single chip PACE microprocessor. An optional control read only memory chip is available which provides multiply/divide and other instructions. A user can microprogram the IMP chip set to any desired implementation, but the IMP-16L implementation is a very nice example of what is possible. Additionally, the IMP-16L provides a package with front panel control, built-in power supplies, and extensive hardware and software support.

The basic IMP-16L package differs from the PACE chip only in a 16-level hardware push down stack, and faster instruction execution. With the addition of the expanded instruction set, the unit offers excellent scientific program support.

Intel 3000

The Intel 3000 series chip set microprocessor features a high-speed and powerful architecture. This unit may be used in two ways. The application may be programmed directly in the microprogramming language.

In this case, the programmer is working with the lowest level language possible, and the most difficult to use, correct, or modify. Alternately, a microprogram may be written to define a machine language programmable machine. Then the application may be programmed in the defined machine language.

Either approach presents more difficulty than any other microprocessor reviewed. After determining the low level of support, and the high degree of implementation complexity with this unit, it was not further considered for this project.

Microprocessor Selection

The 8080 microprocessor was selected for use in the project. The disadvantages of the unit are a slow divide routine execution time, a single level interrupt, and a three voltage power requirement. The first two disadvantages are not serious. A faster divide routine can be coded. The single level interrupt can be expanded to an eight level vectored interrupt by a single commercially available chip. The biggest disadvantage is the power requirement, which in a field installation could be a fatal one.

The advantage of the 8080, and the reason it was chosen, was the software and hardware support available. In addition to the support shown on the comparison sheet, an INTEL MDS development system was locally available for use by this project. A NASA contract with the University of Tennessee will provide an additional 8080 system in the near future. A cross assembler for the unit has been installed on the GSFC

360/91 computer, and is useable by this project. These are all important features.

Rather than implement a microcomputer from basic parts, an Altair 8800 packaged microcomputer was ordered. This system is based on the 8080 microprocessor. The characteristics of the particular unit specified are shown in Figure 4.

Figure 4.

ALTAIR 8800 MICROCOMPUTER

- 1 - Assembled Altair 8800 including
 - Mainframe
 - CPU boards
 - Power supply
 - Front panel interface
- 1 - Assembled expander board
- 1 - Cooling fan
- 1 - 2K static random access memory board
- 1 - 2K programmable read only memory board
- 3 - Parallel Input/Output interface boards
- 1 - Very low cost terminal (VLCT)

5. SOFTWARE SUBSYSTEM

Requirements Analysis

The software subsystem requirements are determined by the following system concerns:

- controlling data acquisition,
- implementing the event detector algorithm, and
- displaying the algorithm results.

These are discussed next.

Data Acquisition

Controlling the data acquisition depends upon the characteristics of the Analog-to-Digital Conversion Subsystem. These are:

- 80 Hz conversion frequency,
- 0 to 3840 amplifier gain factor,
- ± 5 volt A-D converter signal input range, and
- 10-bit binary two's complement A-D converter output.

The system clock which drives the A-D conversion functions independently from the Software Subsystem. The data acquisition task must accept a new digitized signal value 80 times a second, regardless of the other concerns of the software. This necessitates the use of an interrupt driven acquisition process, and a first-in/first-out data buffer. These guarantee data acquisition and event detector processing independence as long as the long-term average data processing rate for the event detector is not exceeded by the signal acquisition rate.

The Analog-to-Digital Conversion Subsystem programmable gain amplifier and 10-bit A-D converter allow representation for signal voltages over a 63 db voltage range, as shown below:

$$\begin{aligned}
 \text{minimum representable signal voltage} &= \frac{\text{quantization step of A-D}}{\text{maximum amplifier gain}} \\
 &= \frac{(10 \text{ volts}/2^{10} \text{ steps}) \times 1 \text{ step}}{3840} \\
 &= 2.54 \times 10^{-6} \text{ volts.}
 \end{aligned}$$

Likewise:

$$\begin{aligned}
 \text{maximum representable signal voltage} &= \frac{10 \text{ volts}}{2^{10} \text{ steps}} \times 511 \text{ steps} \\
 &= 4.99 \text{ volts.}
 \end{aligned}$$

Therefore:

$$\text{dynamic range} = 10 \log \frac{4.99}{2.54 \times 10^{-6}} = 63 \text{ db.}$$

Using the full range of programmable amplifier gain factors is cumbersome and redundant. By only commanding amplifier gain factors corresponding to powers of two, the gain factor can be used as a binary weighting factor for the digitized sample. This is implemented as a binary shift operation on the sample value. The dynamic range of the system loses not quite 3 db in the implementation, a loss more than compensated for by the simplicity of the weighted binary number representation. This dynamic range was judged adequate for the event detector algorithm.

An additional requirement of the data acquisition software is to manipulate the gain of the programmable gain amplifier for two purposes. One (see above) is to maintain the significance of the digitized sample

value as high as possible. There exist several digitized representations of the same signal value, with different weights (amplifier gains) and consequent number of significant bits. Maintaining the highest significance requires keeping the amplifier gain as high as possible without causing the A-D converter input to exceed ± 5 volts. The other purpose is to control the amplifier gain to automatically adjust for different background (environmental) noise levels. To accomplish this, the data acquisition software must, in effect, implement an automatic gain control.

Event Detector Algorithm

The intent of the earthquake detector algorithm is to accurately determine the time of occurrence of an earthquake event. Also of importance is the magnitude and duration of the event detected.

The earthquake detector algorithm was chosen more for its reasonable processing requirements than for any claims as to its efficacy. The algorithm used is described by Steward et al.⁷

The algorithm described requires three to four division operations, depending on data, per data sample. Division was known to be time consuming, especially for the multiple precision integer number format contemplated. The analysis presented in Appendix D, which also describes the detector algorithm, was performed to demonstrate the feasibility of eliminating two division operations. These were replaced by division by a nearby power of two, implemented as an arithmetic right shift. The time saving gained by this substitution was necessary for the processing timing required.

Display Subsystem

The display of algorithm results should minimally indicate the start and stop time of an earthquake event, and its magnitude. In a fully operational system, earthquake onset would trigger additional processing such as a permanent record of the earthquake in a form suitable for satellite relay. In the current implementation, the display subsystem was modified to indicate the state of a binary flag specifying an earthquake event happening. The magnitude of the event was also displayed. The binary flag value can later be recorded on a strip chart along with a time track and the seismic signal trace to permit direct comparison.

An additional requirement was that certain of the earthquake detector algorithm parameters be settable from the Output Subsystem display device (Altair Very Low Cost Terminal). This was specified so that modification of algorithm parameters could be made in a test environment without re-assembling the algorithm software and reprogramming the PROM.

Software Description

Overview

The software is divided into two processing tasks. All data acquisition processing is performed in an interrupt task. All earthquake detector algorithm processing is performed in a non-interrupt task. The Analog-to-Digital Conversion Subsystem interrupts the detector algorithm processing task 80 times a second. The interrupt is serviced by the data acquisition task. This processing is transparent to the earthquake detector task.

The two tasks interface through a first-in/first-out (FIFO) data buffer. When the data acquisition task acquires a new digitized sample, it enters the sample in the FIFO. Whenever the earthquake detector task can process a new sample, it fetches one from the FIFO. If no sample is available, the task continues to attempt fetches until a sample is fetched. This occurs immediately after the data acquisition task interrupts and enters a new sample in the FIFO.

Data Acquisition Task

This task executes in response to the data available interrupt generated when the Analog-to-Digital Conversion Subsystem has digitized a new sample. The task also polls the Altair VLCT, and accepts operator input. This input modifies certain earthquake detector algorithm parameters. Figure 5 presents pseudo-code for the control routine of the task, the interrupt handler INTRP.

A-D Data Processor. This routine controls the acquisition of digitized samples from the Analog-to-Digital Conversion Subsystem. It calls a series of routines which do the actual work. Pseudo-code for this routine is shown in Figure 6.

The system clock is a 24-bit counter incremented each time this routine executes. The clock therefore has a granularity of 1/80 seconds. The clock rolls around to zero approximately every 29 hours.

Data Input Routine. The Data Input routine, DATIN, reads a 10-bit sample from the A-D I/O port. The raw sample is left justified, zero-filled, and stored in the two byte storage cell RAWDT for further processing.

Figure 5.

DATA ACQUISITION TASK INTERRUPT HANDLER

```
INTRP  
    save machine state  
  
    IF A-D data available  
  
        THEN  
            call A-D DATA PROCESSOR  
  
        END-IF  
  
    IF VLCT data available  
  
        THEN  
            call VLCT DATA PROCESSOR  
  
        END-IF  
  
    restore machine state  
  
    return  
  
END
```

Figure 6.

A-D DATA PROCESSOR

```
ADATA  
  
    increment system clock  
  
    call DATA INPUT ROUTINE  
  
    call SCALE DATA ROUTINE  
  
    call FIFO LOAD ROUTINE  
  
    call GAIN ADJUST ROUTINE  
  
    return  
  
END
```

Scale Data Routine. The scale data routine scales the raw data by the previous amplifier gain factor. This factor is always a power of two. Scaling is accomplished by arithmetic right shifting the data n times, where $n = \log_2$ (gain factor). The scaled data is left in the three byte storage cell NORDT.

FIFO Load Routine. The FIFO Load Routine places the data sample in NORDT in the FIFO. The FIFO is 500 data entries long. Since the detector algorithm can keep up with the data acquisition task, the FIFO stores 6 seconds of detector historical data. Pseudo-code for the routine is shown in Figure 7.

Gain Adjust Routine. The Gain Adjust Routine sets the gain factor of the programmable gain input amplifier. The gain setting applies to the next sample digitized. The gain setting depends on the range of the present sample, and the previous state of the gain adjust algorithm.

The algorithm examines the present sample, and assigns it a range of too high, acceptable, or too low. The state of the algorithm is determined by the previous data range. If the data range is too low n times in a row, the algorithm increases the gain factor of the programmable gain input amplifier, and vice versa. Pseudo-code for the algorithm is shown in Figure 8.

VLCT Data Processor. This routine reads an 8-bit value from the Altair Very Low Cost Terminal. The most significant bits of data indicate one of three parameters the routine is to modify. The 6 or 7 remaining bits are the new parameter value.

Figure 7.

FIFO LOAD ROUTINE

STORD

IF FIFO IN = FIFO OUT and

LAST OPERATION = ENTER

THEN

branch to system error reset

END-IF

move the data entry NORDT into FIFO

increment FIFO IN

IF FIFO IN \geq FIFO END

THEN

FIFO IN = FIFO START

END-IF

LAST OPERATION = ENTER

return

END

Figure 8.

GAIN ADJUST ALGORITHMGAINA

```

determine DATA RANGE of NORDT
IF STATE = LOW and DATA RANGE = LOW
THEN
    increment LOW COUNT
    IF LOW COUNT  $\geq$  LOW THRESHOLD
    THEN
        call INCREASE GAIN ROUTINE
        STATE = OK
    END-IF
ELSE-IF STATE = LOW and DATA RANGE = OK
THEN
    STATE = OK
ELSE-IF STATE = LOW and DATA RANGE = HIGH
THEN
    STATE = HIGH
    HIGH COUNT =  $\emptyset$ 
ELSE-IF STATE = OK and DATA RANGE = LOW
THEN
    STATE = LOW
    LOW COUNT =  $\emptyset$ 
ELSE-IF STATE = OK and DATA RANGE = HIGH
THEN
    STATE = HIGH
    HIGH COUNT =  $\emptyset$ 
ELSE-IF STATE = HIGH and DATA RANGE = LOW
THEN
    STATE = LOW
    LOW COUNT =  $\emptyset$ 
ELSE-IF STATE = HIGH and DATA RANGE = OK
THEN
    STATE = OK
ELSE-IF STATE = HIGH and DATA RANGE = HIGH
THEN
    increment HIGH COUNT
    IF HIGH COUNT  $\geq$  HIGH THRESHOLD
    THEN
        call DECREASE GAIN ROUTINE
        STATE = OK
    END-IF
END-IF
END

```

The parameters and indicator bit values are:

-- Alpha event threshold	10XXXXXX
-- Beta event threshold	11XXXXXX
-- Alpha Time Out delay	0XXXXXXX.

Earthquake Detector Task

The Earthquake Detector Task continually attempts to fetch a data sample from the FIFO. When a fetch is successful, the task calls the Earthquake Detector Algorithm routine to perform the digital signal processing. Pseudo-code for the control routine is shown in Figure 9. The Display Status Routine is discussed in the Output Subsystem section.

Earthquake Detector Routine. A description of the earthquake detection algorithm implemented is given in Appendix D. Pseudo-code for the routine is shown in Figure 10.

Output Subsystem

The software routines ONSET and OFFSET are called by DETEC to signal an earthquake start and end. The routines do nothing but return. Eventually, they can be coded to perform all the event detected processing required.

Extensive hardware for the Output Subsystem was not available. The VLCT was therefore utilized as shown in Figure 11 to indicate the status of the earthquake algorithm. This provides a useful diagnostic display. The VLCT is driven by the Display Status Routine called from the Earthquake Detector Task.

Figure 9.

EARTHQUAKE DETECTION TASK CONTROL ROUTINE

START

IF a system error has occurred

THEN

set the system error display flag

ELSE

reset the system error display flag

END-IF

perform all system initialization

REPEAT forever

call DISPLAY STATUS ROUTINE

call FETCH FIFO ROUTINE

IF a sample value is available

THEN

call EARTHQUAKE DETECTOR ROUTINE

END-IF

END-REPEAT

END

Figure 10.

EARTHQUAKE DETECTOR ROUTINEDETEC

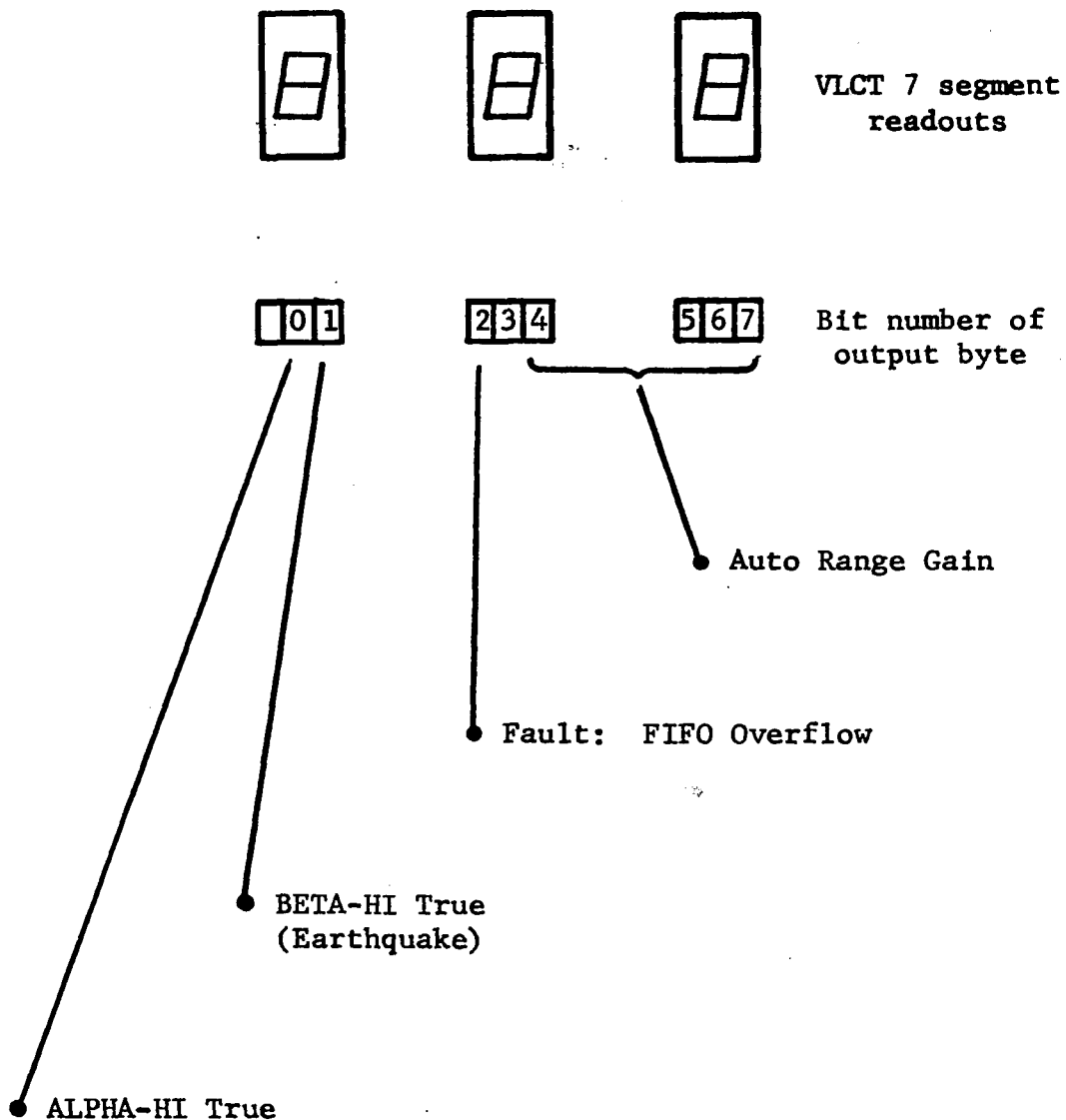
```

DX = |XK - XKMI|
XKMI = XK
WX = 7/8 * WX + 1/8 * DX
ZX = 255/256 * ZX + 1/256 * WX
IF ZX > WX
THEN
    ZX = 3/4 * ZX + 1/4 * WX
END-IF
IF BETA-HI = FALSE
THEN
    IF ALPHA-HI = FALSE
    THEN
        ALPHA = DX/ZX
        IF ALPHA > ALPHA-THRESHOLD
        THEN
            ALPHA-HI = TRUE
            ALPHA-DELAY = 0
        END-IF
    ELSE
        IF ALPHA-DELAY > ALPHA-TIME-OUT
        THEN
            ALPHA-HI = FALSE
        ELSE
            BETA = WX/ZX
            IF BETA > BETA-THRESHOLD
            THEN
                BETA-HI = TRUE
                DURATION = 0
                call ONSET ROUTINE
            ELSE
                increment ALPHA-DELAY
            END-IF
        END-IF
    END-IF
ELSE
    BETA = WX/ZX
    IF BETA > BETA-THRESHOLD
    THEN
        increment DURATION
    ELSE
        call OFFSET ROUTINE
        ALPHA-HI = FALSE
        BETA-HI = FALSE
    END-IF
END-IF
END

```

Figure 11.

OUTPUT SUBSYSTEM DISPLAY
ALTAIR VERY LOW COST TERMINAL



The VLCT indicates an earthquake when the BETA-HI bit is on. The magnitude of the detected event is indicated by the current auto-ranging gain factor displayed.

The output parameters derived by the algorithm are:

- earthquake onset time,
- earthquake duration, and
- earthquake magnitude.

All three are available internally to the ONSET and OFFSET routines. A six second history of data samples is also available to ONSET. This is intended to be used for storing the trace beginning with the ALPHA-HI true transition.

Implementation

The software was coded in the 8080 assembly language.. This was entered into an INTEL MDS development system. The system contained the following hardware:

- terminal,
- line printer,
- floppy disk,
- in-circuit emulator, and
- PROM programmer.

The choice of 8080 hardware just for the use of this development system capability was fortunate.

Extensive use was made of the macro capability of the assembler. Low level mathematical subroutines were always called by macro subroutine drivers. This insures that the data representation implemented is

independent of the detector algorithm. A different data representation can be implemented by only recoding the mathematical subroutines and subroutine drivers.

The software was debugged using the In-Circuit Emulator (ICE-80) capability of the MDS. The Altair Very Low Cost Terminal (VLCT) was substituted for the Analog-to-Digital Conversion Subsystem in the software debug phase. (Note that the VLCT has already been discussed as substituting for the computer and software while debugging the A-D subsystem, as well as serving as the Output Subsystem display device.) In this mode, the VLCT keyboard can input the 8 most significant bits of a manually digitized sample trace. The VLCT display indicates the gain setting currently being sent to the A-D subsystem by the gain adjust algorithm. The ICE-80 capability was used to find software errors as input was manually entered.

A listing of the Software Subsystem is available from the Lehigh University Computer Center Librarian.

6. CONCLUSIONS

The implementation of the Microprocessor Earthquake Precursor Event Detection System was a success. System operation has been demonstrated at several table-banging sessions. The hardware and software success of the system do not really test the success of the earthquake detector algorithm; however, the system has been proved as an algorithm testbed.

The system has the following characteristics:

- field debug capability,
- wide dynamic range input,
- algorithm independence, and
- expandability.

A field debug capability is provided by the use of the Altair VLCT to substitute for the various subsystems in checking out the others. The VLCT substitution also provides the necessary means to calibrate the Analog-to-Digital Conversion Subsystem in the field.

The 60 db dynamic range of the Analog-to-Digital Subsystem, and the data representation scheme provide ample capability to automatically adjust to varying background noise levels. This is valuable, as noise levels vary widely even at a single location.

The system was carefully separated from the algorithm implemented in a well-defined manner. There will be no difficulty implementing a different algorithm, within the capabilities of the Digital Subsystem.

The system is extremely expandable. The entire 8080 computer of the Digital Subsystem occupies 6 of 20 available card slots in the computer mainframe. The memory size can be expanded from the current 2K PROM,

2K RAM to a combined total of 64K. The algorithm execution speed can be improved by a factor of 1.5 - 2.0 by substituting slightly more expensive PROM memory chips.

The only drawback to the field operation of the present system is the performance of the earthquake detector algorithm chosen. The system will be tested in the field, at a seismic observatory in the immediate future. Recent work suggests that a somewhat different algorithm might improve noise immunity.⁸ However, an algorithm very similar to the algorithm implemented here is currently in operational use by the USGS.⁹

The system has certainly demonstrated its stated objective of producing a testbed microprocessor system for processing raw seismic data to detect microearthquake events.

FOOTNOTE REFERENCES

1. Raleigh, B., Bennet, G., Craig, H., Hanks, T., Molnar, P., Nur, A., Savage, J., Scholtz, C., Turner, R., and Wu, F. "Prediction of the Haicheng Earthquake." EOS, Vol. 58 (1977), p. 236.
2. Designing with Microcomputers. A Course prepared by Applied Micro-technology, Santa Clara, California: Signetics Corporation, 1976.
3. "Full one-chip controller sells for under \$10." Electronics (5), Vol. 49 (1976), p. 31.
4. Torrero, Edward A., editor. Microprocessors, New Directions for Designers. New Jersey: Hayden Book Co., 1975, p. 89.
5. Altman, Laurence, editor. Microprocessors. New York: McGraw-Hill Publications Co., 1975, p. 29.
6. "Microprocessors." Electronics (8), Vol. 49 (1976), p. 78.
7. Steward, S.W., Lee, W.H.K., and Eaton, J.P. "Location and Real-Time Detection of Microearthquakes Along the San Andreas Fault System in Central California." Royal Society of New Zealand, Bulletin 9 (1971), pp. 205-209.
8. Allen, R. Personal communication with W. Webster, NASA. 1977.
9. Steward, S.W. "Real-Time Detection and Location of Local Seismic Events in Central California." Bulletin of the Seismological Society of America, Vol. 67 (1977), p. 433.
10. Bogner, R.E. and Constantinides. Introduction to Digital Filtering. London: John Wiley and Sons, 1975.
11. Oppenheim, A.V. and Schaffer, R.W. Digital Signal Processing. New Jersey: Prentice-Hall, 1975.
12. Rabiner, L.R. and Gold, B. Theory and Application of Digital Signal Processing. New Jersey: Prentice-Hall, 1975.

ADDITIONAL REFERENCES

- Ambuter, B.P. and Solomon, S.C. "An Event-Recording System for Monitoring Small Earthquakes." Bulletin of the Seismological Society of America (4), Vol. 64 (1974), pp. 1181-1188.
- "Convertible Data Collection Platform." Tulsa, Oklahoma: LaBarge, Inc., 1975.
- Graeme, J.G. Applications of Operational Amplifiers - Third-Generation Techniques. New York: McGraw-Hill Book Co., 1973.
- Graeme, J.G., Tobey, G.E., and Huelsman, L.P., editors. Operational Amplifiers - Design and Applications. New York: McGraw-Hill Book Co., 1971.
- Hoff, M.E., Jr. "Central Processor Designs Using the Intel Series 3000 Computing Elements." Santa Clara, California: Intel Corp., 1975.
- "IMP-16L Product Description." Santa Clara, California: National Semiconductor Corp., 1974.
- "Intel 8080 Microcomputer System User's Manual." Santa Clara, California: Intel Corp., 1975.
- Ioannidis, G.A., Means, J.D., and George, F.R. "A Multistation Sampled Data Acquisition and Transmission System for Studies of ULF Waves." Hanscom A.F.B., Mass.: 1975.
- "IPC-16A/500D MOS/LSI single chip microprocessor (PACE)." Santa Clara, California: National Semiconductor Corp., 1974.
- Johnson, D.E. and Hilburn, J.L., Rapid Practical Designs of Active Filters. New York: John Wiley and Sons, 1975.
- Johnston, M.J.S., Myren, G.D., O'Hara, N.W., and Rodgers, J.H. "A Possible Seismomagnetic Observation on the Garlock Fault, California." Bulletin of the Seismological Society of America (5), Vol. 65 (1975), pp. 1129-1132.
- Morris, M. "The Online Event Detection System at NCER." Unpublished paper of USGS, Menlo Park, California: 1973.
- Morris, R.L. and Miller, J.R., editors. Designing with TTL Integrated Circuits. New York: McGraw-Hill Book Co., 1971.
- "Motorola 6800 Systems Reference Data Sheets." Phoenix, Arizona: Motorola, Inc., 1975.

"Operating Instructions for the Model TM-1B Biaxial Borehole Tiltmeter."
San Gabriel, California: Kinematics, Inc., 1975.

Press, F. "Earthquake Prediction." Scientific American (5), Vol. 232
(1975), pp. 14-23.

"Signetics 2650 Microprocessor." Sunnyvale, California: Signetics Corp.,
1975.

Van Schaak, J. "Self-Calibrating Seismic Amplifier and Telemetry System."
Unpublished paper of USGS, Menlo Park, California: 1975.

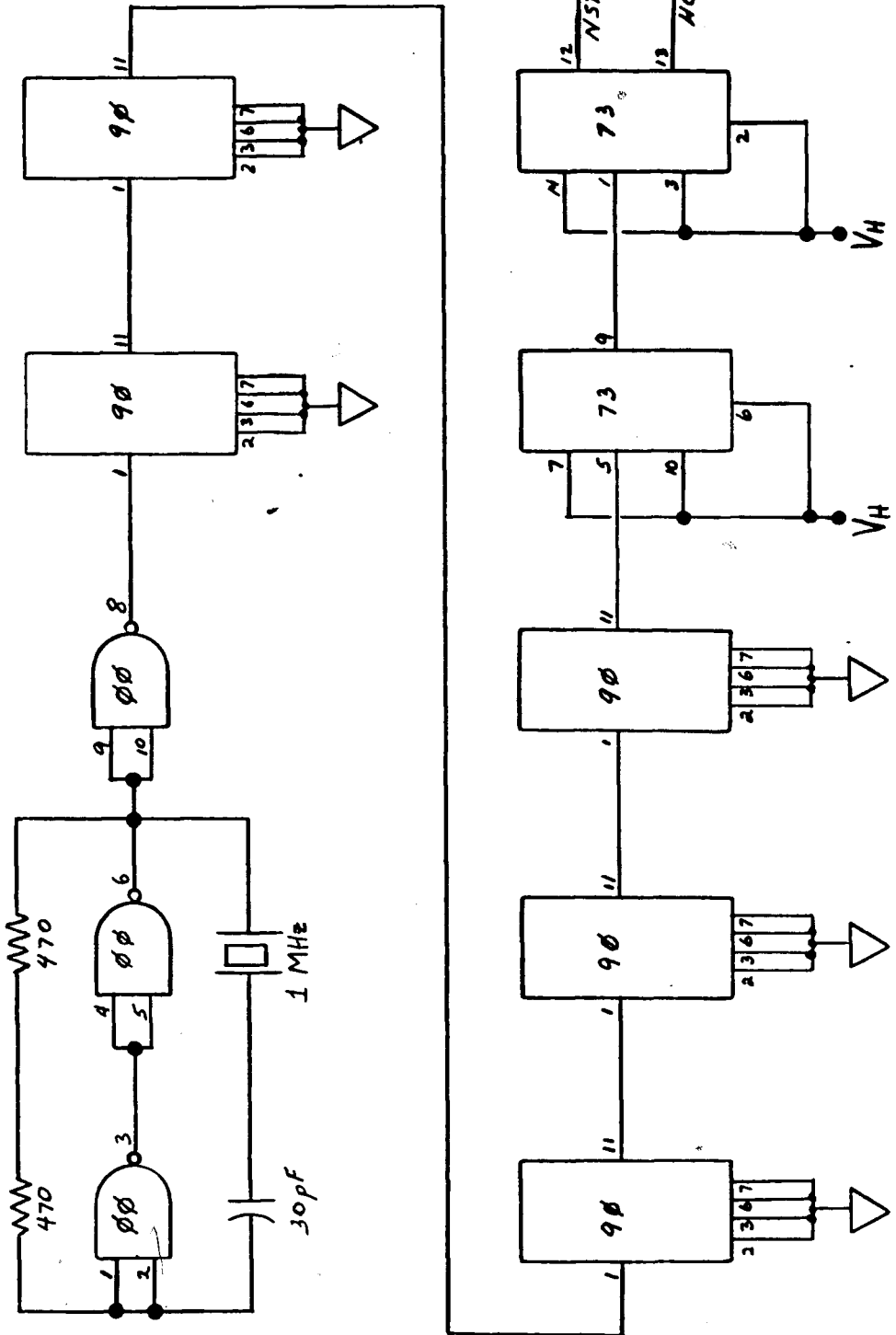
Ward, P.L., Endo, E.T., Harlow, D.H., Allen, R., Marquez, D., and Eaton,
J.P. "Development and Evaluation of a Prototype Global Volcano Sur-
veillance System Utilizing the ERTS-1 Satellite Data Collection System."
A Final Report to NASA, Goddard Space Flight Center. Menlo Park,
California: USGS, 1974.

APPENDIX A

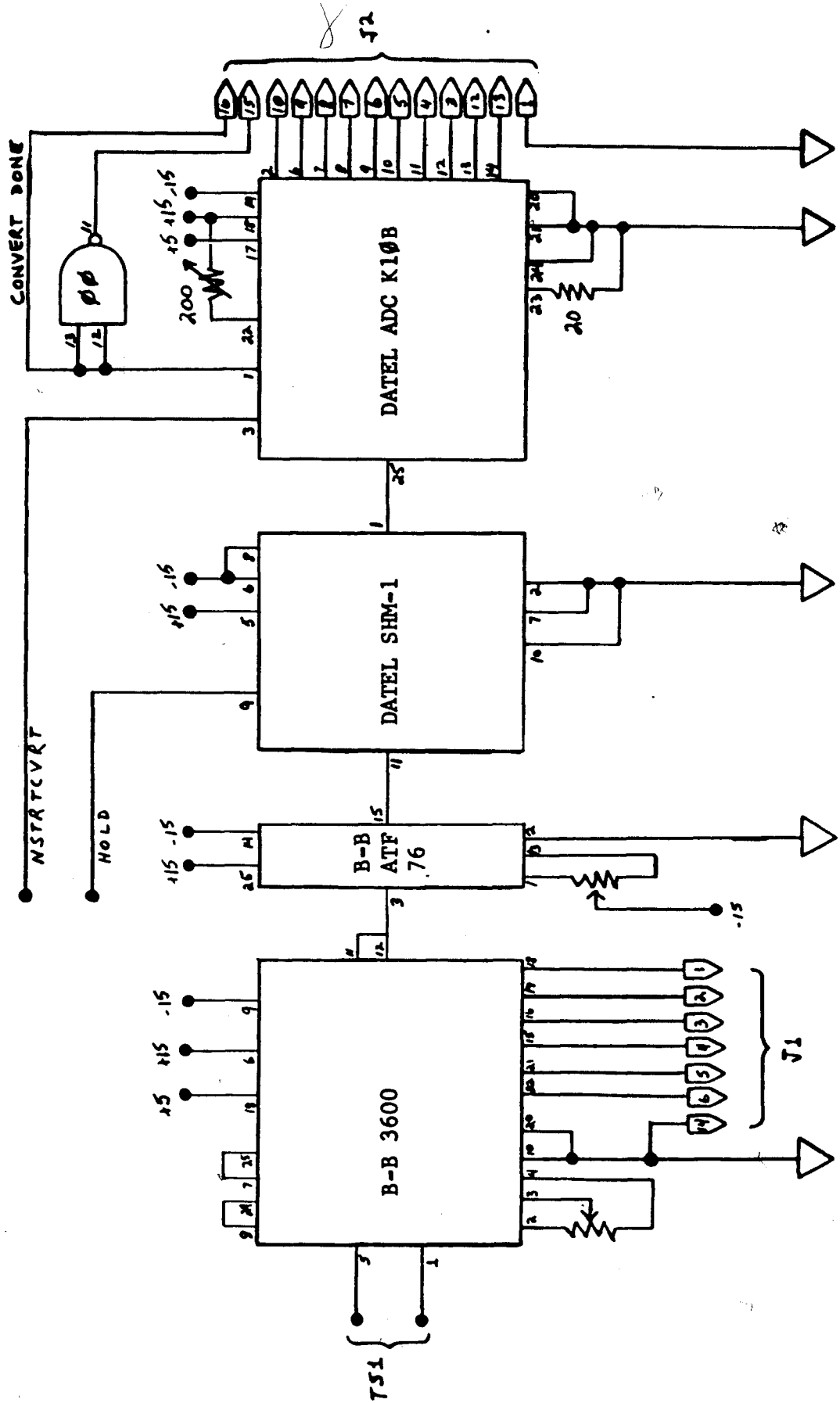
ANALOG-TO-DIGITAL CONVERSION SUBSYSTEM

SCHEMATIC DIAGRAM

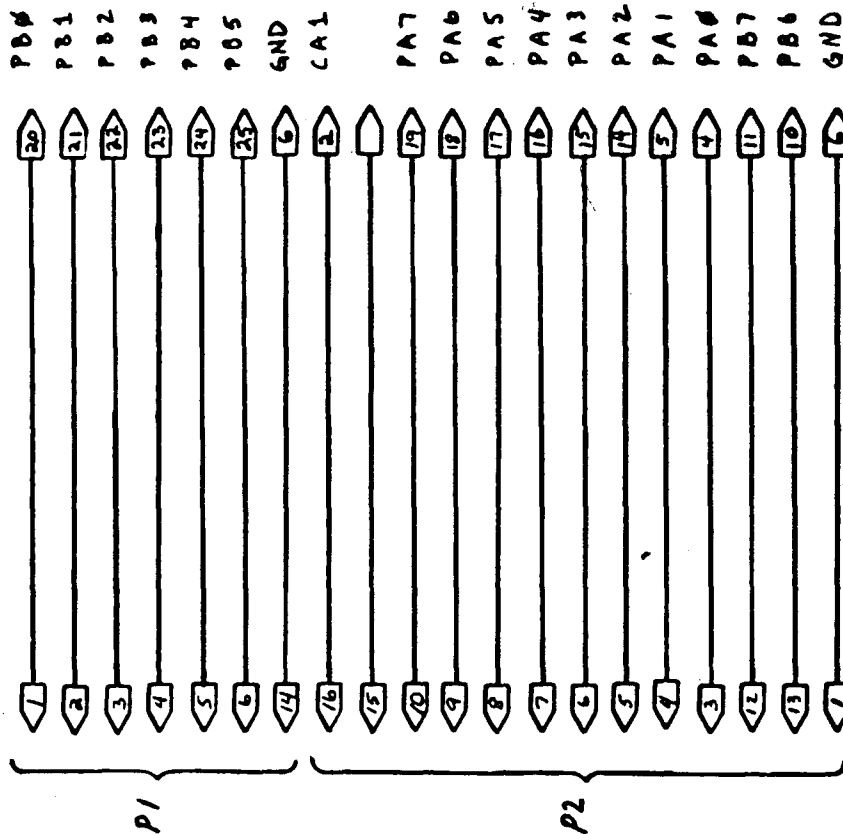
CONTROLLER/TIMING GENERATOR



ANALOG-TO-DIGITAL CONVERSION



INTERFACE CABLE DEFINITION



D connector to 6820 PIO

APPENDIX B

SOFTWARE MULTIPLY/DIVIDE ROUTINE CODING

Intel 8080 - Binary Multiply Subroutine (from Intel manual)

PURPOSE: This routine forms the product of an 8-bit unsigned multiplier in the A register with an 8-bit unsigned multiplicand right justified in the D and E registers. The result and intermediate partial products are formed in the H and L double register. Routine is entered using the CALL instruction, with all arguments pre-loaded in registers. The B register is left at zero.

CALLING SEQUENCE:

<u>Clock Cycles</u>	<u>Bytes</u>	<u>Coding</u>	<u>Comments</u>
17	3	CALL MPY	Multiply D,E by A, result in H,L

ROUTINE CODING:

<u>Clock Cycles</u>	<u>Bytes</u>	<u>Coding</u>	<u>Comments</u>
10	3	MPY: LXI H,0	Zero partial product
7	2	MVI B,8	Preset loop counter to 8
10	1	LOOP: DAD H	Shift partial product 1 bit left
4	1	RAL	Rotate multiplier into carry
10	3	JNC DEC	Test carry, jump if not set
10	1	DAD D	Add multiplicand to partial product
7	2	ACI 0	Clear carry (superfluous)
5	1	DEC: DCR B	Decrement B, set zero flag
10	3	JNZ LOOP	Jump if B ≠ 0
10	1	RET	Return to next instruction in M.P.
	<hr/> 18		

TIMING: Maximum time (at maximum clock frequency) - 238 μ S
Minimum time (at maximum clock frequency) - 170 μ S
Average time (at maximum clock frequency) - 204 μ S

Intel 8080 - Binary Divide Subroutine

PURPOSE: This routine returns the result of dividing a 16-bit unsigned dividend in registers H and L by an 8-bit unsigned divisor in the A register. If the division is performed successfully, the quotient is returned in the L register, the remainder in the H, and the carry is reset. If the division is not performed, either because the divisor is zero or the high order 8 bits of the dividend are greater than or equal to the divisor, the carry is set upon return. Registers B and C are also affected. The routine is entered by the CALL instruction, with all arguments preloaded in registers.

CALLING SEQUENCE:

<u>Clock Cycles</u>	<u>Bytes</u>	<u>Coding</u>	<u>Comments</u>
17	3	CALL DIVD	Divide H,L by A, result in H,L

ROUTINE CODING:

<u>Clock Cycles</u>	<u>Bytes</u>	<u>Coding</u>	<u>Comments</u>
4	1	DIVD: CMP H	
10	3	JC ERRC	Error - divisor LT H.O. dividend
10	3	JZ ERRN	Error - divisor EQ H.O. dividend
7	2	CPI 0	
10	3	JZ ERRN	Error - divisor EQ zero
5	1	MOV C,A	Save A in C
7	2	MVI B,8	Preset loop counter to 8
10	1	LOOP: DAD H	Shift H,L left, zero L.O. bit
4	1	CMP H	Compare divisor with H.O. div.
10	3	JC SUBT	Divisor LT HO dividend
10	3	JNZ NOSB	Divisor GT HO dividend
7	2	MVI H,0	Force result = 0
10	3	JMP AD2L	Skip subtract
4	1	SUBT: SUB H	A = A-H
4	1	CMA	A = not A
5	1	INR A	A = A+1 (two's compl.)
5	1	MOV H,A	H = H-A
5	1	MOV A,C	Restore A
5	1	AD2L: INX H	Set quotient L.O. bit
5	1	NOSB: DCR B	Decrement loop counter
10	3	JNZ LOOP	Loop if not zero
7	2	ACI 0	Reset carry if finished successfully
10	1	ERRC RET	Return
4	1	ERRN STC	Set carry
10	3	JMP ERRC	Go to return

45

TIMING: Maximum time (at maximum clock frequency) - 614 μ S
 Minimum time (at maximum clock frequency) - 462 μ S
 Average time (at maximum clock frequency) - 540 μ S

Motorola 6800 - Binary Multiply Subroutine

PURPOSE: This routine forms the product of two 8-bit unsigned numbers and returns a 16-bit result. The arguments are contained in a 4-word block of memory pointed to by the index register. The block format is:

Index register	+0	→	multiplicand
	+1		multiplier
	+2		H.O. partial product
	+3		L.O. partial product

The partial product must be set to zero before entry to the routine. The routine is entered by the BSR or JSR extended instruction.

CALLING SEQUENCE:

<u>Clock Cycles</u>	<u>Bytes</u>	<u>Coding</u>	<u>Comments</u>
8	2	BSR MPY	Routine is within relative range
9	3	JSR MPY	Routine is outside relative range

ROUTINE CODING:

<u>Clock Cycles</u>	<u>Bytes</u>	<u>Coding</u>	<u>Comments</u>
5	2	MPY: LDAA =8	Load acc. A with 8
7	2	LOOP: ASL,1 3	Shift L.O. partial prod. left into C
7	2	ROL,1 2	Shift C left into H.O. P.P.
7	2	ASL,1 1	Shift multiplier left into C
4	2	BCC TEST	Check high order bit of multiplier
5	2	LDAB,1 0	Load acc. B with multiplicand
5	2	ADDB,1 3	Add L.O. partial product
6	2	STAB,1 3	And replace
4	2	BCC TEST	Check for carry
7	2	INC,1 2	Add carry to H.O. partial product
7	2	TEST: DECA	Decrement loop counter
4	2	BGT LOOP	Iterate 8 times
5	1	RTS	Return to H.P.

25

TIMING: Maximum time (at maximum clock frequency) - 514 μ S
 Minimum time (at maximum clock frequency) - 293 μ S
 Average time (at maximum clock frequency) - 406 μ S

Motorola 6800 - Binary Divide Subroutine

PURPOSE: This routine divides a 16-bit unsigned dividend by an 8-bit divisor and returns the quotient and remainder. The data is passed to the routine in a three-word block pointed to by the index register, as defined below.

Index register	+0	→	high order dividend
	+1		low order dividend
	+2		divisor

The routine returns the remainder in the word occupied by the high order dividend, and the quotient in the low order dividend with the carry bit reset if the divide is performed. If the divisor is zero, or less than or equal to the high order dividend, the routine returns with the carry bit set. The routine is entered by the BSR, or JSR extended instruction.

CALLING SEQUENCE:

<u>Clock Cycles</u>	<u>Bytes</u>	<u>Coding</u>	<u>Comments</u>
8	2	BSR DIVD	Routine is within relative range
9	3	JSR DIVD	Routine is outside of relative range

ROUTINE CODING:

<u>Clock Cycles</u>	<u>Bytes</u>	<u>Coding</u>	<u>Comments</u>
-	1	CTR: RES 1	Reserve a location for loop counter
5	2	DIVD: LDAB,1 2	Load acc. B with divisor
4	2	BEQ ERR	Zero divisor error
5	2	CMPB,1 0	Compare divisor with H.O. dividend
4	2	BLE ERR	Divisor LE H.O. dividend error
2	2	LDAA =8	Load all A with an 8
5	3	STAA CTR	And preset CTR for loop
7	2	ASL,1 1	Shift L.O. dividend left into carry
7	2	LOOP: ROL,1 0	Shift carry into H.O. dividend
5	2	CMPB,1 0	Compare divisor to H.O. dividend
4	2	BGT CARY	Jump if GT - carry is reset
5	2	LDAA,1 0	Load acc. A with H.O. dividend
2	1	SBA	Subtract divisor from H.O. dividend
6	2	STAA,1 0	And replace
2	1	SEC	Explicitly set the carry
7	2	CARY: ROL,1 1	Shift L.O. dividend and insert carry
6	3	DEC CTR	Decrement loop CTR
4	2	BGT LOOP	Iterate 8 times
2	1	CLC	Explicitly clear carry
5	1	RET: RTS	Return from subroutine
2	1	ERR: SEC	Set carry on error
4	2	BRA RET	And return
	<u>40</u>		

TIMING: Maximum time (at maximum clock frequency) - 503 μ S
 Minimum time (at maximum clock frequency) - 303 μ S
 Average time (at maximum clock frequency) - 403 μ S

National Pace - Binary Multiply Subroutine (from National manual)

PURPOSE: This routine multiplies a 16-bit unsigned multiplicand in R2 by a 16-bit unsigned multiplier in R0. The 32-bit result high order bits are left in R0, the low order bits in R1. R3 is left at zero.

CALLING SEQUENCE:

<u>Clock Cycles</u>	<u>Words</u>	<u>Coding</u>	<u>Comments</u>
5	1	JSR MP4	Jump to multiply subroutine in range

ROUTINE CODING:

<u>Clock Cycles</u>	<u>Words</u>	<u>Coding</u>	<u>Comments</u>
-	1	CONST: WORD X'FFFF	
4	1	MP4: LI R1,0	Zero partial product
4	1	LI R3,16	Loop counter
5	1	CAI R0,1	Complement multiplier
4	1	LOOP: RADD R1,R1	
4	1	RADC R0,R0	Shift carry into H.O. partial product
5/6	1	BOC CARRY,T1	Branch on no add condition
4	1	RADD R2,R1	Add multiplicand to partial product
4	1	SUBB R0,CONST	Add carry to H.O. partial product
5/6	1	T1: AISZ R3,-1	Decrement loop CTR
4	1	JMP LOOP	Iterate 16 times
5	1	RTS 0	
	<u>12</u>		

TIMING: Maximum time (at maximum clock frequency) - 994 μ S
 Minimum time (at maximum clock frequency) - 775 μ S
 Average time (at maximum clock frequency) - 885 μ S

National Pace - Binary Divide Subroutine

PURPOSE: This subroutine divides a 32-bit unsigned divisor with high order bits in AC1 and low order bits in AC2 by an unsigned 16-bit dividend in AC3. If the divide is performed, the carry is reset, and the 16-bit remainder is returned in ACL, the 16-bit quotient in AC2. If an error is detected, the subroutine returns with carry set. ACO is used as a working register.

CALLING SEQUENCE:

<u>Clock Cycles</u>	<u>Words</u>	<u>Coding</u>	<u>Comments</u>
5	1	JSR DIVD	Jump to divide in range

ROUTINE CODING:

<u>Clock Cycles</u>	<u>Words</u>	<u>Coding</u>	<u>Comments</u>
-	1	CTR: RES 1	Reserve a loop CTR
4	1	DIVD: RCPY 3,0	Copy divisor to R0
5/6	1	BOC REQ0,ERR	Branch if divisor EQ 0
5	1	CAI 0,1	R0 = - divisor
4	1	RADD 1,0	R0 = divisor - H.O. dividend
5/6	1	BOC PSIGN,ERR	Branch if divisor LT H.O. dividend
4	1	LI * 0,16	Initialize
4	1	ST CTR	Loop CTR
8	1	SHL 2,1,1	Shift L.O. dividend left into link
8	1	LOOP: ROL 1,1,1	Shift link onto H.O. dividend, zero into link
4	1	RCPY 3,0	Set up R0 for test
5	1	CAI 0,1	R0 = - divisor
4	1	RADD 1,0	R0 = H.O. dividend - divisor
5/6	1	BOC NSIGN,NOSB	If GE do subtract
4	1	RCPY 0,1	Make result stick
5	1	SFLG LINK	And set the link
8	1	NOSB: ROL 2,1,1	Shift link into L.O. dividend
9/10	1	DSZ CTR	Decrement CTR, skip if zero
4	1	JMP LOOP	Iterate 16 times
6	1	PFLG CARRY	Reset carry
5	1	RTN: RTS	Return from subroutine
5	1	ERR: SFLG CARRY	Set carry
4	1	JMP RTN	JMP to return

23

TIMING: Maximum time (at maximum clock frequency) - 1,912 μ S
 Minimum time (at maximum clock frequency) - 1,656 μ S
 Average time (at maximum clock frequency) - 1,784 μ S

Signetics 2650 - Binary Multiply Subroutine

PURPOSE: This routine multiplies an 8-bit unsigned multiplicand in R2 by an 8-bit unsigned multiplier in R1. The 16-bit result is produced with high order bits in R1 and low order bits in R0. R3 is left at zero, and the with carry flag is set.

CALLING SEQUENCE:

<u>Clock Cycles</u>	<u>Bytes</u>	<u>Coding</u>	<u>Comments</u>
3	3	BSTA,3 MP4	Branch absolute unconditional

ROUTINE CODING:

<u>Clock Cycles</u>	<u>Bytes</u>	<u>Coding</u>	<u>Comments</u>
3	2	MPY: PPSL 8	Set with carry flag
2	2	LODI,0 0	Zero low order partial product
2	2	LODI,3 8	Initialize loop CTR
2	2	EORI,1 X'FF	Complement multiplier
2	2	LOOP: ADDI,0 0	Reset carry
2	1	RRL,0	Rotate LOPP into carry
2	1	RRL,1	Rotate multiplier into carry
3	2	TPSL 1	Test carry bit
3	2	BCTR,0 NOAD	Branch if carry is set
2	1	ADDZ 2	Add multiplier to LOPP
2	2	SUBI,1 X'FF	Trick - this adds carry to HOPP
3	2	NOAD: BDRR,3 LOOP	Iterate 8 times
3	1	RET,3	Unconditional return
	<u>22</u>		

TIMING: Maximum time (at maximum clock frequency) - 394 μ S
 Minimum time (at maximum clock frequency) - 316 μ S
 Average time (at maximum clock frequency) - 355 μ S

Signetics 2650 - Binary Divide Subroutine

PURPOSE: This routine divides a 16-bit unsigned dividend with high order bits in R0 and low order bits in R1 by a divisor in R2. The quotient is returned in R1 with the remainder in R0 and the carry reset, or the routine returns with carry set if an error is detected.

CALLING SEQUENCE:

<u>Clock Cycles</u>	<u>Bytes</u>	<u>Coding</u>	<u>Comments</u>
3	3	BSTA,3 DIVD	Branch absolute unconditional

ROUTINE CODING:

<u>Clock Cycles</u>	<u>Bytes</u>	<u>Coding</u>	<u>Comments</u>
-	1	CTR: RES 1	Loop CTR
3	2	DIVD: BRNR,2 DOK	Divisor NE 0
3	2	BCTR,3 ERR	Take error return
2	1	DOK: COMZ 2	H.O. dividend GE divisor
3	2	BCFR,2 ERR	Is an error
3	2	PPSL X'A	Preset W.C. and com.
2	2	LODI,3 8	Load loop
2	2	ADDI,3 0	Reset carry
2	1	RRL 1	Rotate L.O. divd. into C
2	1	LOOP: RRL 0	Rotate C into H.O. divd.
3	2	STRR,3 CTR	Save CTR
2	1	STRZ 3	Temp. save R0
2	1	SUBZ 2	Subtract divisor
3	2	BCFR,2 SKIP	Branch on GE 0 - C set
2	1	LODZ 3	Restore R0
2	1	SKIP: RRL 1	Shift C into L.O. divd.
3	2	LODR,3 CTR	Restore loop CTR
3	2	BDRR,3 LOOP	Iterate 8 times
2	2	ADDI,3 0	Reset carry
3	1	RTN: RET,3	Unconditional return
3	2	ERR: PPSL 1	Set carry for error flag
3	2	BCTR,3 RTN	And return
	<u>35</u>		

TIMING: Maximum time (at maximum clock frequency) - 475 μ S
 Minimum time (at maximum clock frequency) - 436 μ S
 Average time (at maximum clock frequency) - 456 μ S

APPENDIX C

MICROPROCESSOR COMPARISON SHEETS

MICROPROCESSOR: Intel 8080
DATA BUS SIZE: 8 bits
ADDRESS BUS SIZE: 16 bits
USEABLE DATA REGISTERS: one 8-bit accumulator
 six 8-bit general purpose

INSTRUCTIONS/BRANCH INSTRUCTIONS: 72/28
EXECUTION TIME (MIN/MAX): 2/8.5µS
MULTIPLY/DIVIDE TIME: 204/540 µS
PUSHDOWN STACK: 1) one 16-bit hardware stack pointer
 2) stack limited to available memory
 3) stack available for general use

INTERRUPT CAPABILITY: 1) single level
 2) up to 8 vector addresses specified
 by interrupting device

DMA CAPABILITY: hold and hold acknowledge pins

SOFTWARE SUPPORT: 1) *assembler 5) *text editor
 2) *cross assembler 6) *library
 3) *loader 7) PL/M compiler
 4) *monitor 8) simulator

HARDWARE SUPPORT: 1)*development system 6)*line printer
 2)*emulator 7) special purpose
 3)*floppy disk hardware chips
 4)*prom programmer 8) memory components
 5)*paper tape reader/
 punch

IMPLEMENTATION SIMPLICITY: 1) minimum configuration: 20 packages
 2) dynamic logic
 3) assembled systems available

POWER REQUIREMENTS: 1.5 w:
 +12 v ± 5%, 70 mA
 + 5 v ± 5%, 80 mA
 - 5 v ± 5%, 1 mA

TECHNOLOGY: N Channel MOS

* Available locally.

MICROPROCESSOR:	Motorola 6800
DATA BUS SIZE:	8 bits
ADDRESS BUS SIZE:	16 bits
USEABLE DATA REGISTERS:	two 8-bit accumulators one 16-bit index register
INSTRUCTIONS/BRANCH INSTRUCTIONS:	72/23
EXECUTION TIME (MIN/MAX):	2/8 μ S
MULTIPLY/DIVIDE TIME:	406/403 μ S
PUSHDOWN STACK:	1) one 16-bit hardware stack pointer 2) stack limited to available memory 3) stack available for general use
INTERRUPT CAPABILITY:	1) four level 2) single vector address for each level
DMA CAPABILITY:	halt and bus available pins
SOFTWARE SUPPORT:	1) cross assembler 2) simulator
HARDWARE SUPPORT:	1) evaluation board 2) emulator 3) special purpose hardware chips 4) memory components
IMPLEMENTATION SIMPLICITY:	1) minimum configuration: 6 packages 2) dynamic logic 3) assembled systems available
POWER REQUIREMENTS:	1.2 w: + 5 v, 240 mA
TECHNOLOGY:	N Channel MOS

MICROPROCESSOR: Signetics 2650
DATA BUS SIZE: 8 bits
ADDRESS BUS SIZE: 15 bits
USEABLE DATA REGISTERS: one 8-bit accumulator
 six 8-bit general purpose registers
 accessible three at a time
INSTRUCTIONS/BRANCH INSTRUCTIONS: 75/22
EXECUTION TIME (MIN/MAX): 4.8/9.6 μ S
MULTIPLY/DIVIDE TIME: 355/456 μ S
PUSHDOWN STACK: 1) 15-bit hardware return address stack
 2) 8 levels deep
 3) available only for subroutine addresses
INTERRUPT CAPABILITY: 1) single level
 2) up to 128 vector addresses specified
 by interrupting device
DMA CAPABILITY: run/wait and bus enable pins provided
SOFTWARE SUPPORT: 1) cross assembler
 2) simulator
 3) program library
HARDWARE SUPPORT: 1) prototyping board
 2) memory components
IMPLEMENTATION SIMPLICITY: 1) minimum configuration: six packages
 2) static logic
POWER REQUIREMENTS: .5 w:
 + 5 v \pm 5%, 100 mA
TECHNOLOGY: Ion Implanted Channel Silicon Gate

MICROPROCESSOR:	National Pace (IPC-16A/500D)
DATA BUS SIZE:	16 bit
ADDRESS BUS SIZE:	16 bit
USEABLE DATA REGISTERS:	one 16-bit principal accumulator three 16-bit auxiliary accumulators
INSTRUCTIONS/BRANCH INSTRUCTIONS:	45/13
EXECUTION TIME (MIN/MAX):	8/14 μ S
MULTIPLY/DIVIDE TIME:	885/1784 μ S
PUSHDOWN STACK:	1) 10 word hardware stack with stack full/ empty interrupt 2) stack available for general use
INTERRUPT CAPABILITY:	1) six level 2) single vector address for each level
DMA CAPABILITY:	additional bus controlling logic required
SOFTWARE SUPPORT:	1) assembler 2) cross assembler 3) loader 4) debug program 5) software is upward compatible with IIP-16
HARDWARE SUPPORT:	memory components
IMPLEMENTATION SIMPLICITY:	1) minimum configuration: 12 packages 2) dynamic logic
POWER REQUIREMENTS:	.7 w: + 5 v 5% + 8 v 5% -12 v 5%
TECHNOLOGY:	Si Gate P Channel Enhancement Mode

MICROPROCESSOR: IMP-16L
DATA BUS SIZE: 16 bits
ADDRESS BUS SIZE: 16 bits
USEABLE DATA REGISTERS: one 16-bit principal accumulator
three 16-bit auxiliary accumulators

INSTRUCTIONS/BRANCH INSTRUCTIONS: 60/12
EXECUTION TIME (MIN/MAX): 4.9/13.5 μ S (basic set)
MULTIPLY/DIVIDE TIME: 171/213 μ S (hardware)
PUSHDOWN STACK: 1) 16 level hardware stack
2) stack available for general use

INTERRUPT CAPABILITY: 1) four level
2) single vector address for each level

DMA CAPABILITY: control pins provided

SOFTWARE SUPPORT: 1) *assembler
2) cross assembler
3) *loader
4) debug program
5) *program library
6) monitor

HARDWARE SUPPORT: 1) card reader 4) floppy disk
2) teletype 5) line printer
3) prom programmer 6) CRT

IMPLEMENTATION SIMPLICITY: packaged microcomputer

POWER REQUIREMENTS: 120 V a.c.

TECHNOLOGY: Si Gate P Channel, Enhancement Mode

* Available locally.

MICROPROCESSOR:	Intel 3000
DATA BUS SIZE:	2 n bits (n=1,2,3,...)
ADDRESS BUS SIZE:	2 n bits (n=1,2,3,...)
USEABLE DATA REGISTERS:	one accumulator ten general purpose registers
INSTRUCTIONS/BRANCH INSTRUCTIONS:	--
EXECUTION TIME (MIN/MAX):	100 nS Clock Cycle
MULTIPLY/DIVIDE TIME:	--
PUSHDOWN STACK:	--
INTERRUPT CAPABILITY:	--
DMA CAPABILITY:	--
SOFTWARE SUPPORT:	cross microprogramming assembler
HARDWARE SUPPORT:	1) special purpose hardware chips 2) emulator
IMPLEMENTATION SIMPLICITY:	minimum configuration: 50 packages static logic
POWER REQUIREMENTS:	5 v
TECHNOLOGY:	Schottky BiPolar

APPENDIX D
ALGORITHM ANALYSIS

ALGORITHM ANALYSIS

Algorithm Description

Referring to Figure 12, the algorithm can calculate five quantities for each digitized sample input. The symbols for these quantities are taken from Steward et al., 1971. DX_k is referred to as the conditioned seismic trace, W_k as the short-term average, and Z_k as the long-term average.

For every sample input, DX_k , W_k , Z_k , and α_k are calculated. If Z_k exceeds W_k after this calculation, Z_k is set equal to $Z_k - (Z_k - W_k)/4$. If α_k exceeds an α threshold, the sample time k is taken to be a possible earthquake event start. The value of β_k is calculated when this is true, instead of α_k . If β_k does not exceed a β threshold within a certain time limit, the possible event start is discarded, and the α_k calculation resumed. Otherwise, the event onset is confirmed. The β_k calculation is performed until β_k no longer exceeds the β threshold. The period of time the β threshold was exceeded is taken to represent the event duration. This is not necessarily a good measure of duration; however, Steward (1977) points out that it is a convenient measure.

The equations of interest are numbered 3 and 4 in Figure 12. If the divisions by 10 and 250 are replaced by 8 and 256, the software implementation is as a shift. This is necessary to avoid the time required for the division specified.

Figure 12.

ALGORITHM EQUATIONS

1. X_k = Seismic trace sample at time k.
2. DX_k = $|X_k - X_{k-1}|$.
3. W_k = $W_{k-1} + (DX_k - W_{k-1})/10$.
4. Z_k = $Z_{k-1} + (W_k - Z_{k-1})/250$.
5. α_k = DX_k/Z_k .
6. β_k = W_k/Z_k .

Digital Filter Analysis

The analysis of the effect of modifying these factors is not complete, but is taken far enough to show that no drastic changes are caused by the modification made.^{10,11,12} If the sampled trace of a signal is represented as:

$$X(t) = x_0\delta(t) + x_1\delta(t-T) + x_2\delta(t-2T) + \dots$$

then the Z transform is:

$$X(Z) = x_0 + x_1Z^{-1} + x_2Z^{-2} + \dots$$

The transfer function of a system is defined:

$$G(Z) = \frac{V(Z)}{U(Z)},$$

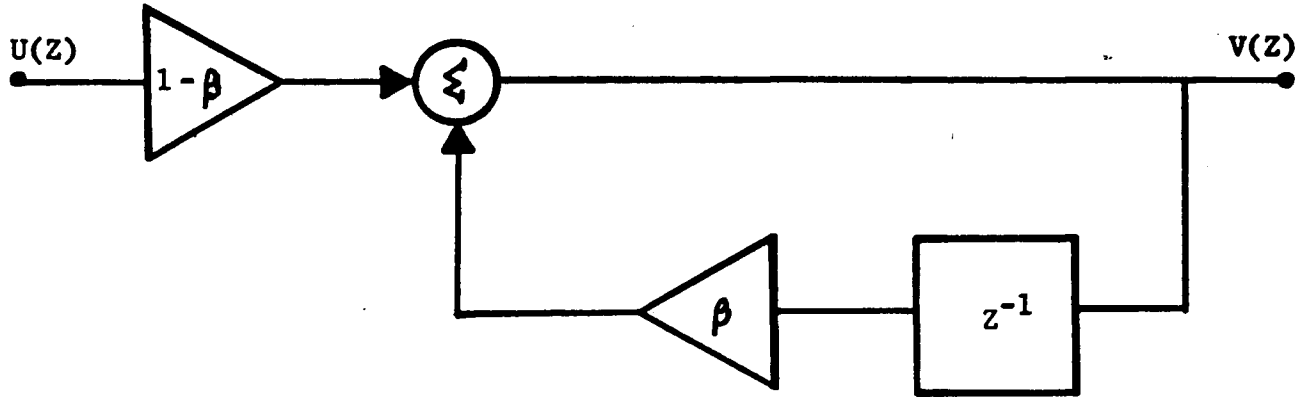
where $V(Z)$ is the response of the system to input $U(Z)$. This function is most conveniently determined for $U(Z) = 1$. Equation 3 (or 4) of Figure 12 can be written as:

$$A_k = \beta \cdot A_{k-1} + (1 - \beta)X_k,$$

where $0 < \beta < 1$, X_k is the input. This is usually diagrammed as shown in Figure 13.

Figure 13.

DIGITAL FILTER DIAGRAM



The response of this system to $U(Z) = 1$ is:

$$V(Z) = 1 - \beta + \beta(1-\beta)Z^{-1} + \beta^2(1-\beta)Z^{-2} + \dots = \frac{(1-\beta)}{1-\beta \cdot Z^{-1}}$$

Therefore:

$$G(Z) = \frac{V(Z)}{U(Z)} = \frac{(1-\beta)}{1-\beta \cdot Z^{-1}}$$

To apply a general sinusoidal input to this filter, it must first be transformed to the Z-plane. Sampling $f(t) = e^{j\omega t}$ at period T:

$$f^*(t) = \delta(t) + e^{j\omega T} \delta(t-T) + e^{2j\omega T} \delta(t-2T) + \dots$$

Transformed to the S-plane:

$$F^*(s) = 1 + e^{j\omega T} e^{-sT} + 2e^{j\omega T} e^{-2sT} + \dots$$

Substituting to the Z-plane:

$$F(Z) = 1 + e^{j\omega T} Z^{-1} + e^{2j\omega T} Z^{-2} + \dots = \frac{1}{1 - e^{j\omega T} Z^{-1}}$$

Note that this is a causal signal, since it is sampled from $t = 0$ on.

Finally:

$$V(Z) = U(Z) \cdot G(Z) = \frac{Z^2 (1 - \beta)}{(Z - e^{j\omega T}) (Z - \beta)} .$$

The inverse Z transform is evaluated using the Cauchy Integral Theorem, or by partial fractions.

If the Z transform is defined:

$$X(Z) = \sum_{n=-\infty}^{\infty} X(n) Z^{-n},$$

then the inverse Z transform of $V(Z)$ above is:

$$V(t) = \sum_{n=0}^{\infty} (1 - \beta) \frac{[e^{(n+1)j\omega T} - \beta^{(n+1)}]}{e^{j\omega T} - \beta} .$$

Referring this to equation 3 of Figure 12, $\beta = 1/10$, modified to $\beta = 1/8$ is a change in value of 25 percent. This causes a definite change in the characteristics of the filter. The frequency response, phase response, and transient response are all slightly affected. The amount of processing saved by this change made it mandatory, however. In the case of equation 4, where β was changed from $1/250$ to $1/256$, the change is much smaller.

APPENDIX E

VITA

VITA

Robert Gregory Novas, the son of Robert Anthony and Antoinette Giacinto Novas, was born in Englewood, New Jersey on December 2, 1951. He is married with a seven year old daughter and resides in suburban Washington, D.C. He attended Lehigh University, receiving his Bachelor of Arts degree in Mathematics in June of 1972. Currently, he is a computer applications analyst with General Electric Company - Space Division, based in Beltsville, Maryland.

The involvement with NASA, Geophysics Branch came about through Will Webster, a fellow amateur radio operator, while working for the LANDSAT project in 1975.