NASA Technical Memorandum 4106

Interfacing Laboratory Instruments to Multiuser, Virtual Memory Computers

Edward R. Generazio
Lewis Research Center
Cleveland, Ohio

David B. Stang
Sverdrup Technology, Inc.
Cleveland, Ohio

Don J. Roth
Lewis Research Center
Cleveland, Ohio

National Aeronautics and
Space Administration
Office of Management
Scientific and Technical
Information Division

1989

Abstract

Incentives, problems and solutions associated with interfacing laboratory equipment with multiuser, virtual memory computers are presented. The major difficulty concerns how to utilize these computers effectively in a medium sized research group. This entails optimization of hardware interconnections and software to facilitate multiple instrument control, data acquisition and processing. The architecture of the system that was devised and associated programming and subroutines are described. An example program involving computer controlled hardware for ultrasonic scan imaging is provided to illustrate the operational features.

Introduction

Computers based on "virtual memory", multiuser architecture have recently become economically available to medium sized research groups. These new computers offer advantages to the experimentalist, though acceptance has been sometimes limited. One of the most important advantages of a "virtual memory" machine is the capability of fast access to arrays having dimensions greater than 64 kbytes. Arrays larger than 64 kbytes were available on some older computers based on "physical memory." The access time to the memory beyond 64 kbytes is extremely slow for these systems. Fast access to large arrays is particularly important now where large data sets are required for research systems.

Many researchers are accustomed to a dedicated computer running a single-user operating and data acquisition system. When there are several experimentalists in a research group, the advantage of a multiuser system becomes important. The resources a computer provides can be shared among the users at a much lower cost than that of maintaining a separate dedicated computer and peripherals for each project. Software routines for instrument control and data acquisition as well as data bases are sharable resources. Expensive hardware resources, such as array processors, images processors, terminals, printers and color plotters can also be shared.

There are difficulties in assembling the hardware and software facilities required for a multiuser, experimental research oriented system. The single-user operating systems typically use built-in commands for easy instrument communication and other functions well suited for the experimenter's needs. A multiuser computer's input-output capabilities are more powerful but harder to understand. The user is left with the responsibility for bridging the gap between the capabilities provided and his particular needs for data acquisition and communication with devices.

The general purpose interface bus (GPIB) is used extensively in single-user computer systems for data acquisition and instrument control. For single-user systems the software for data acquisition and instrument control is often provided by the manufacturer of the system. This same bus can be used in a multiuser system. However, the necessary software is not provided. This lack of software has been a major obstacle for researchers and has prevented them from moving from a single to a multiuser virtual memory system.

In this work we will provide the reader with the information required for developing a multiuser computer system that meets the needs of a medium sized research group. The basic computer system and its components are described. Specific examples of GPIB instrument control and data acquisition with a multiuser system are described for use in both BASIC and FORTRAN programming. Additionally, a collection of high and low level subroutines for computer control of instruments have been listed in the appendixes.

Computer System and Instruments

Computer System

A computer system for a research group has several components. Figure 1 shows a block diagram of a balanced computer system specifically designed for an experimentally intensive research group. The system is based on a virtual memory computer. There are three central processing units (CPUs) in this configuration, each with 16 Mbyte memory, and two, 474 Mbyte, hard disks attached. The second hard disk is for backup of system and user software. Generally 16 Mbytes of memory is required for the processing of large data arrays. Each CPU (subsystem) has a major specific function. CPU 1 is designed for data analysis and contains an array processor for fast Fourier transformations. In addition, CPU 1 shares an image processor with CPU 2 for display images viewed with the image processor. CPU 2 is designed for image analysis and display. The 32 Gbyte optical disk drive provides easy access and storage of images. CPU 3 is designed for data

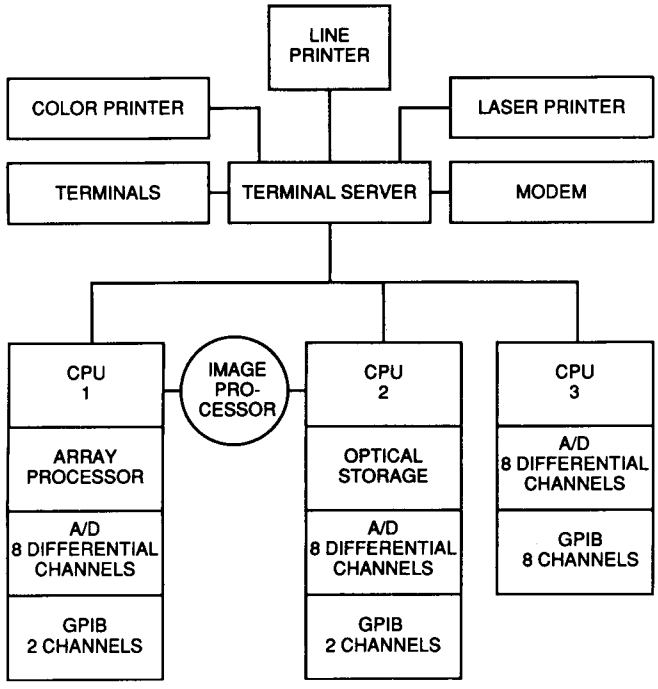

Figure 1.—Computer system.

acquisition and contains four GPIB interfaces yielding eight GPIB channels (ports) and an analog to digital (A/D) input/output interface with 16 channels. Up to nine users can use the third CPU simultaneously for data acquisition. Of course A/D interfaces may be added for additional users. Alternatively, fewer users could run multiple experiments simultaneously.

It is important to note that, although each CPU subsystem has been designed for a specific function, it is not limited to that function. Therefore, CPUs 1 and 2 also have GPIB and A/D interfaces. As shown a total of 15 researchers may acquire data or control instruments simultaneously via the GPIB or A/D interfaces. Equally important to the number of users working with the instruments is that additional users may be simultaneously analyzing data, image processing, developing software and generating reports.

The three CPUs are connected via an Ethernet connection for easy transfer of files between systems. A terminal server connected to the Ethernet allows for multiuser, remote access to each of the CPUs. This remote access allows researchers to have terminals and instruments placed at different locations throughout the site. Off-site access to the system is permitted via modems attached directly to the CPU or Ethernet. Modems are used for initiating, stopping or checking the status of an experiment on evenings or weekends.

Instruments

A variety of instruments may be attached to the GPIB interface. It is this interface that is often the stumbling block in meeting the researchers' needs. The problem is two-fold for a multiuser system. First, the GPIB bus has a limit on total length of 20 m. One cannot expect to have all experiments taking place in one location (i.e., room, building, etc.); therefore, the need arises to extend the GPIB bus beyond this limit. Second, friendly software for multiuser, virtual memory, GPIB based instrument control and data acquisition is not available.

A typical configuration for a nondestructive evaluation research group is shown in figure 2. Several research instruments including ultrasonic, x-ray tomographic, and tunneling scanners are attached to the computer system. Recently, GPIB bus extenders have been made available from several manufacturers. By using bus extenders, these instruments may be placed at locations remote from the CPUs. The GPIB bus extenders are transparent to the researchers' software so that no software modifications are needed.

The software required to control these instruments and for data acquisition is the most complicated of the two problems and for this reason it will be presented in detail.

Programming the components of the precision acoustic scanner covers a wide range of software problems encountered when connecting instruments to a computer via the GPIB bus. We will focus on this system, as it represents the most common problems.

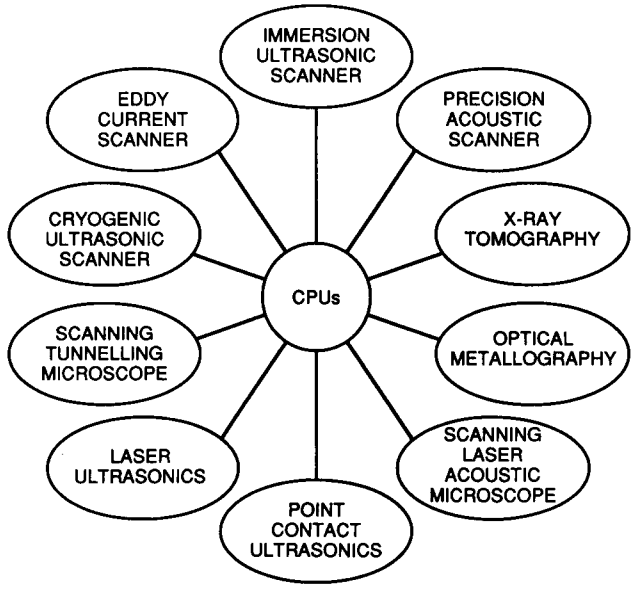

Figure 2.—Instrument configuration for a nondestructive evaluation research group.

The components of the precision acoustic scanning system are shown in figure 3. The specimen to be interrogated is mounted on a X-Y-Z positioning table. In contact with the sample is a ultrasonic transducer mounted on a displacement pressure gauge. During operation, the positioning table is moved to a series of points that describe an array. This is done while maintaining a constant pressure between the transducer and specimen. At each point ultrasonic waveforms are selected, digitized and stored for later analysis.

This system has the following six components that communicate over the GPIB bus: three positioning tables, waveform digitizer with a time base and voltage base, time delay and voltmeter. The programming for this rather complex scanning system contains all of the features needed for adaptation to other computer controlled systems.

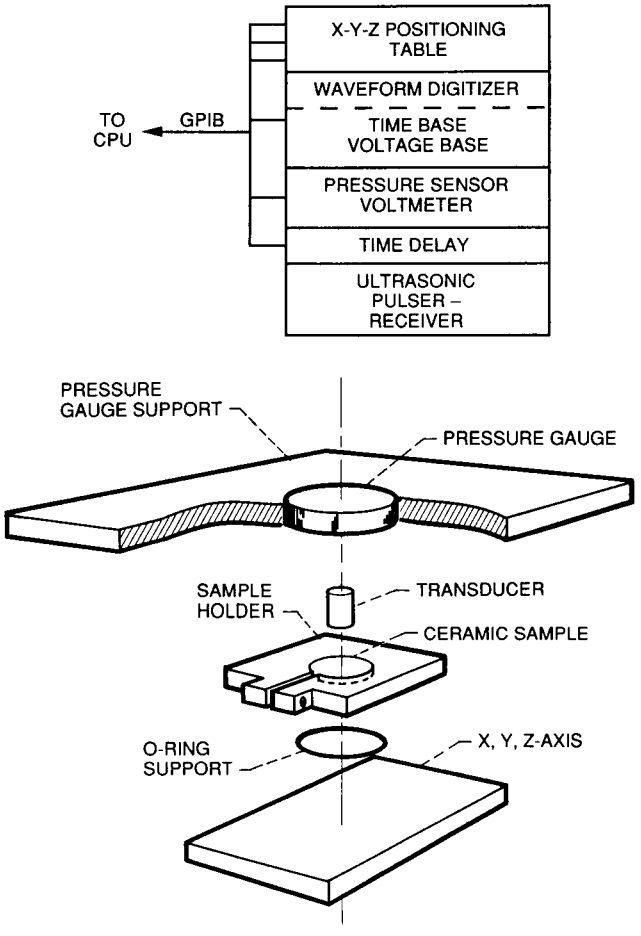

Figure 3.—Precision acoustic scanning system.

Programming for Instrument Control and Data Acquisition

In this section a fundamental program, written in FORTRAN, is presented that operates the precision acoustic scanning system. The subroutines for this program are provided in the appendixes.

Initially a GPIB channel is assigned to the precision acoustic scanner. Here a channel represents a physical GPIB port on the CPU.

The FORTRAN statement

```
CALL STRTGPIB (0)
```

initializes channel 0. Any instrument addressed subsequently in the program will communicate through channel 0. Next each component of the scanning system must be initialized. There are six components, and their physical addresses have been adjusted to be contiguous from 32 to 37. These are also called addresses 0 to 5 where the it is implied that the standard base GPIB address of 32 is to be added. The physical addresses for the digitizer, time delay, X-axis, Y-axis, Z-axis and voltmeter are set by the researcher at 0, 1, 2, 3, 4 and 5, respectively.

```
DO 10 NUMBER = 0, 5
CALL INITINSTR (NUMBER)
10    CONTINUE
```

As indicated earlier, the specimen is scanned in an organized array in the X-Y plane. The step size in the X, Y and Z directions are stipulated by the user as

```
XSTEP = 100
YSTEP = 100
ZSTEP = 10      ! the z axis will not be used for this
                ! template. But an argument is
                ! required for these subroutines.
```

and sent to the X-, Y- and Z-axis of the positioning table

```
CALL SETXYZ (2, XSTEP, 3, YSTEP, 4, ZSTEP)
```

where 2, 3 and 4 correspond to the X, Y and Z positioner axes. The ultrasonic waveform needs to be digitized and stored. The ultrasonic wave occurs at specific time and is set by the time delay subroutine

```
CALL SETDELAY (1, DELAY(1))
```

where the first argument is the address of the time delay. DELAY(1) is the time delay required to place the ultrasonic wave within the digitizing window. The volts and time per division of the time and voltage base are set

```
CALL PUTTIME (0, SETTING)
CALL SETVOLTDIV (0, SETTINGS(1))
```

where the first argument is the primary address of digitizer that contains the time and voltage bases. The SETTING and SETTINGS(1) are the proper time and voltages per division for displaying the ultrasonic wave in the digitizing window. The actual digitization of the ultrasonic waveform occurs with the command

```
CALL GETSA (0, 64, 320, A)
```

where the first argument is the address of the digitizer and 64 is the number of samples that are to be averaged, 320 is the intensity of the writing beam in the digitizer and "A" is the array that will contain the digitized waveform. The waveform is saved to a file opened as unit 6 with

```
WRITE (6) A
```

After a waveform is collected at one point, another point is selected with the X-Y-Z positioning table with

```
CALL MOVXYZ (2, 1, 1)
```

where the first argument corresponds to the address of the X-axis. The second argument corresponds to the axis to be moved one step (1, 2 and 3 for the X-, Y- and Z-axis, respectively). If the third argument is greater or less than zero, then the motion is in the positive or negative direction, respectively.

This program would loop back to the CALL PRESSURE statement to complete a scan line. The above FORTRAN source code gives a template for writing other source code. The complete source code for scanning a 5- by 5-mm square area is given in appendix A.

Appendix B contains a listing and description of each of the user callable high level subroutines used for instrument control. The low level source code for these routines is provided in appendix C. This source code may be adapted for writing other instrument control subroutines for additional instruments not covered in this work.

The complete program SCAN.FOR must be compiled first

```
$ FORTRAN SCAN
```

and linked

```
$ LINK SCAN, SYS$SYSROOT:[SYSLIB]IMAGELIB/LIB –
SYS$SYSROOT:[SYSLIB]IEXSUI.OBJ
```

then run with

```
$ RUN SCAN
```

The subroutines provided as well as this and other data acquisition and instrument control programs may be shared with other users. Researchers may also write programs for operating instruments attached to other GPIB channels. With this capability, several researchers may be operating instruments and acquiring data simultaneously. A considerable amount of time is saved by sharing these developed programs.

We have found that, when programs and instruments are easily shared, program development occurs at a rapid rate. This rapid growth occurs because researchers modify each control program to meet their particular needs. In doing so, the fundamental program expands and is fine tuned by all the researchers to form a very versatile system.

Summary

A multiuser, virtual memory computer system was devised and programmed for use by a medium sized research group. Problems associated with optimizing the system for shared, multiuser applications were resolved. Particular attention was given to hardware interfacing and development and compilation of subroutines. The result is a straightforward operating network for instrument control, data acquisition and processing. A comprehensive example program is provided for illustrative purposes.

Lewis Research Center
National Aeronautics and Space Administration
Cleveland, Ohio, February 17, 1989

Appendix A
FORTRAN Program for Running Precision Acoustic Scanning System

```
C PROGRAM SCAN.FOR   (November 7, 1988)
C       by Edward R. Generazio and David B. Stang
C
C WRITTEN IN DEC FORTRAN VERSION, COMPILED AND RUN ON A DEC MICROVAX II
C WITH MVMS VERSION 4.5 OPERATING SYSTEM
C
C THIS PROGRAM RUNS THE PRECISION ACOUSTIC SCANNING SYSTEM DEVELOPED AT NASA
C LEWIS RESEARCH CENTER, STRUCTURAL INTEGRITY BRANCH, STRUCTURES DIVISION,
C CLEVELAND, OHIO.
C
C THE INSTRUMENTS ATTACHED ARE
C
C       TWO KLINGER SCIENTIFIC C-1.22 PROGRAMMABLE STEPPER
C       MOTOR CONTROLLER
C
C       TEKTRONIX 7912 PROGRAMMABLE DIGITIZER WITH A  TEKTRONIX
C       7B90P PROGRAMMABLE TIME BASE AND
C       A TEKTRONIX 7A16P PROGRAMMABLE AMPLIFLIER (VOLTAGE BASE)
C
C       HEWLETT-PACKARD 5359A TIME SYNTHESIZER (TIME DELAY)
C
C       FLUKE 8520A DIGITAL MULTIMETER (VOLTMETER)
C
C       SENSOTEC LOAD SYSTEM.  THE SENSOTEC ANALOG OUTPUT IS PROPORTIONAL
C       TO THE PRESSURE ON THE LOAD CELL AND IS  MEASURED BY THE
C       VOLTMETER.
C
C
C
C       THIS PROGRAM MOVES THE SCANNER IN AN ORGANIZED 5MM BY 5MM
C       SQUARE ARRAY.
C
C       AT EACH POINT THE CONTACT PRESSURE BETWEEN THE TRANSDUCER
C       AND THE SPECIMEN BEING SCANNED IS  15  TO 17 POUNDS.
C
C       ALSO AT EACH POINT TWO WAVEFORMS ARE DIGITIZED AND STORED IN
C       MASS STORAGE FOR LATER ANALYSIS.

        IMPLICIT NONE
        INTEGER*4 XSTEP,YSTEP,ZSTEP,XAXIS,YAXIS ! DEFINITIONS
        INTEGER*4 I,J,K,NAVE,MAI
        INTEGER*2 A(512)
        BYTE BUFFER(32),WRKO(80)
        REAL DELAY(2),VOLTSET(2)
        CHARACTER*10 FILNAME
C
C       INSTRUMENTS ARE SET AT FOLLOWING ADDRESSES
C
C       DIGITIZER  = 0
C       TIME DELAY = 1
C       X AXIS     = 2
C       Y AXIS     = 3
C       Z AXIS     = 4
C       VOLTMETER  = 5
C
```

```
      FILNAME='SCANNER.DAT'    ! NAME OF FILE TO SAVE WAVEFORM IN

      OPEN(UNIT=6,STATUS='NEW',NAME='FILNAME',FORM='UNFORMATTED')
                               ! OPEN THE FILE

      DELAY(1)=4.35E-6         ! TIME DELAY FOR FIRST WAVEFORM
      DELAY(2)=5.20E-6         ! TIME DELAY FOR SECOND WAVEFORM

      VOLTSET(1)=0.5           ! VOLTS PER DIVISION FOR FIRST WAVEFORM
      VOLTSET(2)=0.2           ! VOLTS PER DIVISION FOR SECOND WAVEFORM

      XSTEP=1000               ! STEP SIZE FOR THE X AXIS
      YSTEP=1000               ! STEP SIZE FOR THE Y AXIS
      ZSTEP=10                 ! STEP SIZE FOR THE Z AXIS

      XAXIS=5                  ! NUMBER OF STEPS ALONG THE X AXIS
      YAXIS=5                  ! NUMBER OF STEPS ALONG THE Y AXIS

      NAVE=64                  ! NUMBER OF SAMPLES TO AVERAGE


      CALL STRTGPIB(0)         ! INITIALIZE CHANNEL 0 OF THE GPIB
                               ! INTERFACE

      DO 10 I=0,5              ! INITIALIZE INSTRUMENTS AT GPIB
10    CALL INITINSTR(I)        ! ADDRESSES 0,1,2,3,4 AND 5

      CALL SETXYZ(2,XSTEP,3,YSTEP,4,ZSTEP)
                               ! TELL THE POSITIONER THE SIZE OF THE
                               ! STEPS TO TAKE

      CALL GETMAI(0,MAI)       ! GET THE CURRENT INTENSITY SETTING
                               ! ON THE DIGITIZER

      DO 20 I=1,YAXIS          ! THE YAXIS WILL MOVE 5 TIMES

      DO 30 J=1,XAXIS          ! THE XAXIS WILL MOVE 5 TIMES

      CALL PRESSURE            ! ADJUST THE PRESSURE TO THE
                               ! APPROPRIATE VALUE

      DO 40 K=1,2              ! DO THIS FOR EACH WAVEFORM

      CALL PUTTIME(1,DELAY(K))! SET THE DELAY TIME FOR THE WAVEFORM

      CALL SETVOLTDIV(0,VOLTSET(K))
                               ! SET THE VOLTS PER DIVISION FOR
                               ! THE WAVEFORM

      CALL GETSA(0,NAVE,MAI,A)
                               ! SET THE DIGITIZER INTENSITY TO MAI
                               ! AVERAGE OVER 64 SAMPLES
                               ! PUT DIGITIZED WAVEFORM IN ARRAY
                               ! CALLED "A"

40    WRITE(6)A                ! SAVE WAVEFORM IN FILE NAMED SCANNER.DAT

      IF(J.EQ.XAXIS) GOTO 30   ! IF THE END OF A SCAN LINE IS REACHED
                               ! RETURN TO THE X AXIS ORIGIN AND
                               ! INCREMENT THE Y AXIS
```

```
        CALL MOVXYZ(2,1,1)       ! OTHERWISE MOVE THE X AXIS ANOTHER STEP

 30     CONTINUE                 !

        IF(I.EQ.YAXIS) GOTO 20   ! THE SCAN IS OVER END THE PROGRAM
                                 ! CLOSE THE SCANNER.DAT FILE AND
                                 ! RELEASE THE GPIB CHANNEL

        CALL MOVORG(1,XAXIS)     ! OTHERWISE RETURN TO THE X AXIS ORIGIN
                                 ! AND
        CALL MOVXYZ(3,2,1)       ! MOVE THE Y AXIS ANOTHER STEP

 20     CONTINUE

        CLOSE(6)                 ! CLOSE THE FILE SCANNER.DAT

        END                      ! END THE PROGRAM SCAN.FOR

        INCLUDE 'GPIBASE.FOR'    ! INCLUDE THE INSTRUMENT CONTROL
                                 ! AND DATA ACQUISITION SUBROUTINES
                                 ! IN THE FILE BASE.FOR
                                 ! WHEN COMPILING.
                                 ! THESE SUBROUTINES ARE IN THE
                                 ! APPENDIX OF THIS ARTICLE

C TWO SUBROUTINES THAT CALL SUBROUTINES IN  BASE.FOR
C
C
      SUBROUTINE PRESSURE
C
C          ADJUST Z AXIS TO GET A GOOD PRESSURE
C           <<<   +1 = DOWN   >>>
C
        PUPPER = .85             ! UPPER PRESSURE = 17 lbs
        PLOWER = .75             ! LOWER PRESSURE = 15 LBS

  200   CALL GETFLUKE( 5,P )     ! GET THE OUTPUT VOLTAGE FROM THE
                                 ! PRESSURE SENSOR

        IF( P.LT.PLOWER )CALL MOVXYZ(4, 3, 1 )
                                 ! IF PRESSURE IS TO LOW THEN INCREASE
                                 ! THE PRESSURE BETWEEN THE SPECIMEN
                                 ! AND THE TRANSDUCER

        IF( P.GT.PUPPER )CALL MOVXYZ(4, 3, -1 )
                                 ! IF PRESSURE IS TO HIGH THEN DECREASE
                                 ! THE PRESSURE BETWEEN THE SPECIMEN
                                 ! AND THE TRANSDUCER

        IF( P.GE.PLOWER .AND. P.LE.PUPPER )GOTO 900
                                 ! THE PRESSURE IS WITHIN AN ACCEPTABLE
                                 ! RANGE

        GOTO 200                 ! CHECK PRESSURE

  900   RETURN                   ! PRESSURE ADJUSTED- RETURN TO MAIN PROGRAM
        end
```

```
      SUBROUTINE MOVORG( SCANAXIS, NSTEPS )
C
C        MOVE SCANNER BACK TO THE SCANAXIS ORIGIN
C
        INTEGER*4   SCANAXIS
        INTEGER*2   NSTEPS

        DO 1000 K=1,1000
 1000   CALL MOVXYZ(4, 3,-1 )            ! MOVE Z AXIS UP
        DO 1200 K=1,NSTEPS-1
 1200   CALL MOVXYZ( 2, SCANAXIS, -1 )   ! MOVE X AXIS BACK
        DO 1400 K=1,1000
 1400   CALL MOVXYZ( 4, 3, 1 )           ! MOVE Z AXIS DOWN
        CALL PRESSURE                    ! ADJUST PRESSURE
        CALL MOVXYZ( 2,SCANAXIS, -1 )    ! MOVE BACK AND FORTH
        CALL MOVXYZ( 2,SCANAXIS, 1 )     ! TO MAKE A GOOD CONTACT
        RETURN
        end
```

Appendix B
User Callable High Level Subroutines for Instrument Control

GPIBASE subroutines

GPIBASE refers to a DEC VAX/VMS object code set containing FORTRAN callable subroutines which can be used to communicate with various GPIB (General Purpose Interface Bus, also known as IEEE-488) instruments found in the Structural Integrity Branch laboratories. The "DEC IEX Interface" consists an hardware driver, version 3.0, along with software which is documented in the "IEX-VMS-DRIVER" manual. The following describes subroutines which utilize the IEX Interface to perform tasks typically done in the labs.
Only one IEX channel is operated per main program, and each subroutine requires an integer argument which is the GPIB address of the instrument to be addressed. For instance, the following is a table of the configuration in the Stuctural Integrity Laboratories:

CHANNEL, GPIB ADDRESS	INSTRUMENT
0, 0	Tektronix 7912AD Digitizer
0, 1	Hewlett Packard 5359A Time Synthesizer 1
0, 2	Klinger Scientific c.122 XYZ Controller- X Axis
0, 3	Y Axis
0, 4	Z Axis
0, 5	Fluke 8520A Multimeter
1, 1	Tektronix 7854 Digital Oscilloscope
1, 2	Testech LS86-C Ultrasonic Immersion Scanner
1, 3	Hewlett Packard Time Synthesizer 2
1, 5	Hewlett Packard 53591A A/D Converter

GENERAL GPIBASE SUBROUTINES

STRTGPIB(N) - Assign and initialize IEX channel N

This must be called before calling channel N instrument subroutines. N is less than M and M is the number of physical ports. It assigns the VAX/VMS internal address and also clears the interface. Any instrument addressed subsequently in the program will communicate through channel N.

INITINSTR(ADDR) - Send initialization commands to instrument.

ADDR - GPIB address (integer)

Does Selected Device Clear and Remote ENable. Should be called for every instrument intended for use.

SUBROUTINES FOR TEKTRONIX 7912 DIGITIZER

GETBESTMAI(ADDR, MAI) - Get the optimum intensity value

MAI - Returns intensity value (integer)

The number of interpolated points is found at current intensity; if 0, intensity is increased. Intensity is lowered step by step until the number of interpolated points reaches 0. This should be called before receiving data from 7912.

GETMAI(ADDR, MAI) - Get current value of intensity from 7912

MAI - Returns current intensity value (integer)

SETVOLTDIV(ADDR, SETTING) - Set volts/div on 7912

SETTING - Must be one of: .01, .02, .05, .1, .2, .5, 1., 2., 5.

SETTIMEDIV(ADDR, SETTING) - Set seconds/div on 7912

SETTING - Must be one of: 1., 2., or 5.
E -3, -6, -9, or -12 (e.g. 2.E-6)

AUTOSETVOLTS(ADDR, MAI, SETTING) - Find best voltage/div setting

MAI - Intensity to use (integer)
SETTING - Returns volts/div setting (real)

The lowest voltage setting (.01) is tried initially and number of interpolated points on the 7912 is found. The setting is increased until interpolated points is 0. MAI should be value returned from GETBESTMAI.

DIGDEF(ADDR) - Do a "Digitize Defects"

This operation should be done before taking data. It causes the 7912 to remember the defective points and does not send them when data is requested.

GETGRID(ADDR, TIME, VOLTS) - Get Time/division and Volts/Division

TIME - Returns Seconds per division as set on the 7912 (real)
VOLTS - Returns Volts per division as set on the 7912 (real)

GETSA(ADDR, NAVE, MAI, A) - Acquire Sample-Averaged waveform

NAVE - Number of averages desired (integer; 0 thru 64)
MAI - Intensity to use (integer)
A(512) - Integer array returned of added samples

This subroutine sets intensity (should be value returned from GETBESTMAI), requests sample averaging, and requests read. If an error ocurrs, TEKRESET is called and read is requested again. The array A must be divided by NAVE to result in (1-255) = bottom to top of oscilloscope display.

TEKGTL(ADDR) - Send GO TO LOCAL and TV MODE commands to 7912

TEKRESET(ADDR) - Reset 7912

This calls INITINSTR, GETBESTMAI, and DIGDEF

SUBROUTINES FOR HEWLETT-PACKARD TIME SYNTHESIZER

PUTTIME(ADDR, TIME) - Set Time Delay

TIME - Desired Delay Time in seconds (0. < TIME < 166.E-3)

GETTIME(ADDR, TIME) - Get Delay

TIME - Returns time delay currently displayed (real)

GETSTATE(ADDR, STATE) - Get State of Time Synthesizer

STATE(66) - Byte array of codes describing the current complete state of the device. Useful for returning it back to a particular state with PUTSTATE.

PUTSTATE(ADDR, STATE) - Set State of Time Synthesizer

STATE(66) - A byte array of codes describing a state to set the device to. (See GETSTATE)

SUBROUTINES FOR KLINGER XYZ CONTROLLER

SETXYZ(ADDRX, IX, ADDRY, IY, ADDRZ, IZ) - Initialize controller

Before issuing any commands to the Klinger, this subroutine must be called to indicate the stepsize needed for each axis.

IX - X axis stepsize, in microns (integer; 0 < IX < 10000)
IY - Y axis stepsize " IY
IZ - Z axis stepsize " IZ

MOVXYZ(ADDR, IAXIS, IPM) - Move one step

IAXIS - Axis to move the distance according to most recent call of SETXYZ (integer; [1,2,3]=[X,Y,Z]).

IPM - Flag for direction (integer; <0 =backward, >0 =forward)

SUBROUTINES FOR FLUKE MULTIMETER

GETFLUKE(ADDR, VOLTAGE) - Get voltage from Fluke Multimeter

VOLTAGE - Value read by meter in volts. (real)

SUBROUTINES FOR TEKTRONIX 7854 DIGITAL OSCILLOSCOPE

GETWFM(ADDR, AVES, DATA, XINCR, YZERO, YSCALE) - Acquire 512-point averaged currently displayed waveform

AVES - Number of averages (integer)
DATA(512) - Real array representing averaged waveform
XINCR - Time increment in seconds between points (real)
YZERO - Value in volts represented by DATA value of 0 (real)
YSCALE - Scale factor of DATA (real; VOLTS/DATA value)

TEK78RESET(ADDR) - Reset 7854

This calls INITINSTR and does a serial poll.

SUBROUTINES FOR TESTECH LS-86 SCANNER

MOVTESTECH(ADDR, IAXIS, IDEST) - Move to destination IDEST

IAXIS - Desired axis to move (1,2,3,4)=(X,Y,Z,Swivel)
IDEST - Destination in thousandths of an inch if X,Y, or Z;
hundredths of a degree if Swivel. (integer)
X limits: 0 < IDEST < 36000
Y limits: 0 < IDEST < 12000
Z limits: 0 < IDEST < 15000
Swivel limits: 0 < IDEST < 36000

FINDTESTECH(ADDR, IX, IY, IZ) - Get current location

IX - Returns X location in thousandths (integer)
IY - Returns Y location in thousandths "
IZ - Returns Z location in thousandths "

SUBROUTINES FOR HEWLETT-PACKARD A/D CONVERTER

GETA2D(ADDR, IDATA) - Get value from channel 4 of H.P. A/D converter

IDATA - Returns value from 0 to 1024, scaled such that
1024 = approx 10 volts. (integer)

Appendix C
Low Level Source Code for Subroutines Listed in Appendix B

```
C # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
C                                                                               #
C    GPIBASE.FOR                                                                #
C                                                                               #
C         This is a set of subroutines which communicate                        #
C         with the IEEE-488 instruments found in the Structural                 #
C         Integrity Branch laboratories. For further                            #
C         documentation, see the file GPIBASE.TXT
C                                                                               #
C       David Stang and Edward Generazio                                        #
C                                                                               #
C       Latest update:   3-NOV-1988                                             #
C                                                                               #
C # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #

       SUBROUTINE STRTGPIB( ICHAN )
C
C           Assign GPIB channel ( IEX unit )
C           Call IESTRT to initialize the unit
C           Send IFC command to clear the interface

        byte        WRK0(80), BUFFER(32)
        integer*2   ISTAT(4)
        integer*4   ICHAN
        character   STRING*6,ICHAR*1
        common       /IBLK/ WRK0,BUFFER
10020   format(I1)
        ENCODE( 1,10020, ICHAR )ICHAN
        STRING = '_IXA'//ICHAR//':'
        ISTATUS = SYS$ASSIGN( STRING  , ICH ,,)
        WRITE( 5,10100 )ICH, ISTATUS
        CALL IESTRT( WRK0, ICH, 0,, 0, 1, ISTAT )
        CALL IECMD( WRK0, ,, %REF('IFC'), ISTAT )
10100   FORMAT( '  CHANNEL ASSIGNED =',I5,'   STATUS =',I2 )
        RETURN
        END
C
       SUBROUTINE INITINSTR( LISTEN )
C
C           Call IEPOLL, send SDC, and any needed
C           initialization commands to device
C           at address LISTEN
C
        byte   WRK0(80),  BUFFER(32)
        byte   ADDR(2), CHR8
       common  /IBLK/ WRK0,BUFFER
        integer*2 ISTAT(4)
        NADDR = 1
        ADDR(1) = LISTEN + 32
        ADDR(2) = 96
C       IF( LISTEN.EQ.0 )NADDR=2        ! IF THERE IS A SECONDARY ADDRESS
                                        ! TWO BYTES MUST BE SENT
        CALL IECMD( WRK0, ADDR,NADDR, %REF('SDC'), ISTAT )
        CALL IECMD( WRK0, ADDR,NADDR, %REF('REN'), ISTAT )
        RETURN
        end
C
```

```
      SUBROUTINE GETSA( ADDR, NAVE, MAI, A )
C
C         Acquire 512-point waveform from Tek 7912
C
C           NAVE   = Number of averages
C           MAI    = Main Intensity
C           A(512) = Data

       byte       BUFFER(32), WRK0(80)
       byte       ABYTE(1029), LADDR(2), TADDR(2), B
       character     AINT*3, ANAVE*2, ABUFFER*32
       integer*2     A(512), AWORD(512), ISTAT(4)
       integer*4     ADDR
       equivalence   ( ABUFFER, BUFFER )
       equivalence   ( AWORD(1), ABYTE(4) )
      common         /IBLK/ WRK0,BUFFER

  100  LADDR(1) = ADDR+32
       LADDR(2) = 96
       TADDR(1) = ADDR+64
       TADDR(2) = 96
C
C   Set intensity
C
       ENCODE( 3,10023, AINT )MAI
10023  format( I3 )
       ABUFFER(1:8) = 'MAI '//AINT//';'
       CALL IESEND( WRK0, BUFFER,8, LADDR,2, ISTAT )
C
C   Request Sample Averaging
C
       ENCODE( 2,10022, ANAVE )NAVE
10022  format( I2 )
       ABUFFER(1:10) = 'DIG SA,'//ANAVE//';'
       CALL IESEND( WRK0, BUFFER,10, LADDR,2, ISTAT )
C
C   Request Read, Get data, check for error ( e.g. timeout error )
C
       ABUFFER(1:6) = 'REA SA'
       BUFFER(7) = 13
       CALL IESEND( WRK0, BUFFER,7, LADDR,2, ISTAT )
       CALL WAIT2(400)
       CALL IERECV( WRK0, ABYTE,1029, ,, TADDR, ISTAT )
              IF(ISTAT(1).NE.1)THEN
              WRITE( 8,10800 )
10800         format( '     ERROR AT IERECV IN GETSA, TRYING AGAIN' )
              CALL TEKRESET( AD,MAI )
              GOTO 100
              ENDIF

       DO 1000 I=1,512          !
          II = 2*I + 2          !    Byte swap !
          B= ABYTE( II )
          ABYTE( II ) = ABYTE( II+1 )
          ABYTE( II+1 ) = B
 1000  A(I) = AWORD(I)
       RETURN
       end
C
```

```
      SUBROUTINE AUTOSETVOLTS( AD, MAI, SET )
C
C        Sets appropriate voltage setting by
C        checking on number of interpolated points
C
C              MAI = Intensity
C              SET = Returns setting
C
       byte      BUFFER(32), WRK0(80)
       character ABUFFER*32, AMAI*3
       byte      ADDR(2)
       real      SETTINGS(6)
       integer*2    ISTAT(4)
       integer*4    AD
       equivalence  ( BUFFER, ABUFFER )
      common   /IBLK/ WRK0,BUFFER
       data     SETTINGS/ 0.01,0.02,0.05,0.10,0.20,0.50 /
       ADDR(1) = AD+32
       ADDR(2) = 96
       ENCODE( 3,10023, AMAI )MAI
10023  format( I3 )
       I=1
       IB=0
C      TYPE *,'CALLING SETVOLTSDIV'
  100  CALL SETVOLTDIV( AD,SETTINGS(I) )
       WRITE (5,1133)SETTINGS(I)
 1133  FORMAT('+','CURRENT GAIN',F10.5)
       CALL WAIT2(100)
C      TYPE *,'SETVOLTS ANSWERED',SETTINGS(I)
       ABUFFER(1:26) = 'MAI '//AMAI//';DIG DAT;DEF ON;ATC'
       CALL IESEND( WRK0, BUFFER,26, ADDR,2, ISTAT )
       CALL WAIT2(100)
       CALL GETNUMINT( AD,NUMINT, MAI )
C      TYPE *,NUMINT
           IF( NUMINT.EQ.999 )THEN
               SET = 999.
               GOTO 900
               ENDIF
       IF( NUMINT.EQ.0 )THEN
           IF( IB.EQ.1 )GOTO 200
           IB=1
           GOTO 100
           ENDIF
  200  IF( NUMINT.GT.0 )THEN
           I=I+1
           GOTO 100
           ENDIF
       SET = SETTINGS (I)
  900  RETURN
       end
C
      SUBROUTINE GETBESTMAI( AD,MAI )
C
C       Figure out the best intensity by finding number
C       of interpolated points over a number of trials
C
       byte      BUFFER(32), WRK0(80)
       character ABUFFER*32, AMAI*3
       byte      ADDR(2)
       integer*2    ISTAT(4)
```

```
        integer*4    AD, MAI
        equivalence  ( BUFFER, ABUFFER )
       common   /IBLK/ WRK0,BUFFER
10023   format(I3)
        ADDR(1) = AD+32
        ADDR(2) = 96
        IB = 0
C       TYPE *,'GONE TO LOCAL IN GBM'
        CALL GETMAI( AD,ITRY )
C       TYPE *,'GOT MAI AGAIN IN GET BESTMAI'
        CALL DIGDEF( AD )
C       CALL WAIT2(2000)
  100   ENCODE( 3,10023, AMAI )ITRY
        ABUFFER(1:26) = 'MAI '//AMAI//';DIG DAT;DEF ON;ATC'
        CALL IESEND( WRK0, BUFFER,26, ADDR,2, ISTAT )
        CALL WAIT2(100)
        CALL GETNUMINT( AD, NUMINT, MAI )
C       TYPE *,NUMINT
        IF( NUMINT.GT.0 )THEN
            ITRY=ITRY + 1
            IB = 1
            GOTO 100
            ENDIF
        IF( IB.EQ.1 )GOTO 900
            ITRY=ITRY - 1
            IB = 0
            GOTO 100
  900   MAI = ITRY + 20
        RETURN
        end

C
      SUBROUTINE GETMAI( AD,MAI )
C
C          Get intensity from TEK 7912
C
        byte       BUFFER(32), WRK0(80)
        character ABUFFER*32
        byte       ADDR(2)
        integer*2     ISTAT(4)
        integer*4     AD, MAI
        equivalence  ( BUFFER, ABUFFER )
       common   /IBLK/ WRK0,BUFFER
 21     ADDR(1) = AD+32
        ADDR(2) = 96
        ABUFFER(1:4) = 'MAI?'
        CALL IESEND( WRK0, BUFFER,4, ADDR,2, ISTAT )
        ADDR(1) = 64
        CALL IERECV( WRK0, BUFFER,20, ,, ADDR, ISTAT )
        OUT=0
        DO 5 K=5,20
        IF(OUT.EQ.1)GOTO 5
        IF( BUFFER(K).EQ.';' )THEN
        KK=K
        OUT=1
        ENDIF
 5      CONTINUE
        DECODE( KK-5,10023, ABUFFER(5:KK-1))MAI
C       DECODE( 20,10023, ABUFFER(5:20) )MAI
10023   format( I3 )
        WRITE(5,1133)MAI
```

```
 1133   FORMAT('+','CURRENT INTENSITY',I7)
        RETURN
        end

       SUBROUTINE GETNUMINT( AD, NUM, MAI )
C
C         Get number of interpolated points
C
        byte      BUFFER(32), WRK0(80)
        character ABUFFER*32
        byte      ADDR(2)
        integer*2    ISTAT(4)
        integer*4    AD, NUM, MAI
        equivalence  ( BUFFER, ABUFFER )
       common   /IBLK/ WRK0,BUFFER
10020   format(I)
   40   ADDR(1) = AD+32
        ADDR(2) = 96
        IB=0
  100   ABUFFER(1:4) = 'INT?'

        CALL IESEND( WRK0, BUFFER,4, ADDR,2, ISTAT )
        ADDR(1) = 64
        CALL WAIT2(100)
        CALL IERECV( WRK0, BUFFER,20, ,, ADDR, ISTAT )
        IF( BUFFER(5).EQ.'N' )THEN
            IF( IB.GT.0 )THEN
                WRITE ( 8,10100 )
10100           format('     "N" RECIEVED IN GETNUMINT, RESETTING')
                CALL TEKRESET( AD,MAI )
                GOTO 40
                ENDIF
            IB=1
            GOTO 100
            ENDIF
        K=1
  200      IF( BUFFER(K).EQ.'?' )THEN
                WRITE( 8,10200 )
10200           format('     "?" RECIEVED IN GETNUMINT, RESETTING')
                CALL TEKRESET( AD,MAI )
                GOTO 40
           ENDIF
           IF( BUFFER(K).EQ.';' )THEN
           DECODE( K-4,10020, ABUFFER(4:K-1) )NUM
           GOTO 300
           ENDIF
        K=K+1
        GOTO 200
  300   WRITE(5,1133)NUM
 1133   FORMAT('+','INTERPOLATED POINTS',I7)
        RETURN
        end
C

       SUBROUTINE SETVOLTDIV( AD,SETTING )

        byte      BUFFER(32), WRK0(80)
        character ABUFFER*32
        character AVOLTS*10
        byte      ADDR(2)
        integer*2    ISTAT(4)
```

```
      integer*4    AD
      equivalence  ( BUFFER, ABUFFER )
      common   /IBLK/ WRK0,BUFFER
      ADDR(1) = AD+32
      ADDR(2) = 97
10040 format( E10.5 )
      ENCODE( 10,10040,AVOLTS )SETTING
      ABUFFER(1:14) = 'V/D '//AVOLTS
      CALL IESEND( WRK0, BUFFER,14, ADDR,2, ISTAT )
      RETURN
      end

      SUBROUTINE SETTIMEDIV( AD, SETTING )

      byte      BUFFER(32), WRK0(80)
      character ABUFFER*32
      character ATIME*10
      byte      ADDR(2)
      integer*2    ISTAT(4)
      integer*4    AD
      equivalence  ( BUFFER, ABUFFER )
      common    /IBLK/ WRK0,BUFFER
      ADDR(1) = AD+32
      ADDR(2) = 98
10040 format( E10.5 )
      ENCODE( 10,10040,ATIME )SETTING
      ABUFFER(1:14) = 'T/D '//ATIME
      CALL IESEND( WRK0, BUFFER,14, ADDR,2, ISTAT )
      RETURN
      end

      SUBROUTINE DIGDEF(AD)

      byte      BUFFER(32), WRK0(80)
      character ABUFFER*32
      byte      ADDR(2)
      integer*4    ad
      integer*2    ISTAT(4)
      equivalence  ( BUFFER, ABUFFER )
      common    /IBLK/ WRK0,BUFFER
      ADDR(1) = AD+32
      ADDR(2) = 96
      ABUFFER(1:11) = 'DIG DEF,100'
      CALL IESEND( WRK0, BUFFER,11, ADDR,2, ISTAT )
      CALL WAIT2(2000)
      RETURN
      end

C

      SUBROUTINE GETGRID( AD,TIME,VOLTS )
C
C        Get Time/Div and Volts/Div from TEK 7912
C
      byte      BUFFER(32), WRK0(80)
      character ABUFFER*32
      byte      ADDR(2)
      integer*2    ISTAT(4)
      integer*4    AD
      equivalence  ( BUFFER, ABUFFER )
```

```
        common      /IBLK/ WRK0,BUFFER
         ADDR(1) = AD+32
         ADDR(2) = 96
10040    format( E )

         ABUFFER(1:4) = 'HS1?'
         CALL IESEND( WRK0, BUFFER,4, ADDR,2, ISTAT )
         ADDR(1) = 64
         K=1
         CALL IERECV( WRK0, BUFFER,15, ,, ADDR, ISTAT )
  200       IF( BUFFER(K).EQ.';' )THEN
            DECODE( K-4,10040, ABUFFER(4:K-1) )TIME
            GOTO 300
            ENDIF
         K=K+1
         GOTO 200

  300    ADDR(1) = 32
         ABUFFER(1:4) = 'VS1?'
         CALL IESEND( WRK0, BUFFER,4, ADDR,2, ISTAT )
         ADDR(1) = 64
         K=1
         CALL IERECV( WRK0, BUFFER,15, ,, ADDR, ISTAT )
  400       IF( BUFFER(K).EQ.';')THEN
            DECODE( K-4,10040, ABUFFER(4:K-1) )VOLTS
            GOTO 900
            ENDIF
         K=K+1
         GOTO 400
  900    RETURN
         end
C

       SUBROUTINE TEKGTL(AD)
C
C          Tell 7912 to Go To Local & put into TV MODE
C
         byte       BUFFER(32), WRK0(80)
         character ABUFFER*32
         byte       ADDR(2)
         integer*2     ISTAT(4)
         integer*4     AD
         equivalence  ( BUFFER, ABUFFER )
        common    /IBLK/ WRK0,BUFFER
         ADDR(1) = AD+32
         ADDR(2) = 96
         ABUFFER(1:8) = 'MODE TV;'
         CALL IESEND( WRK0, BUFFER,8, ADDR,2, ISTAT )
         CALL IECMD( WRK0, ADDR,2, %REF('GTL'), ISTAT )
         RETURN
         end

       SUBROUTINE TEKRESET( AD,MAI )
C
         byte  BUFFER(32), WRK0(80)
         integer*4 AD,MAI
       common  /IBLK/ WRK0, BUFFER
         CALL INITINSTR(0)
         CALL SETVOLTDIV( AD,.5 )
         CALL GETMAI( AD,MAI )
         CALL GETBESTMAI( AD,MAI )
```

```
C        CALL DIGDEF(AD)
         RETURN
         end

C

        SUBROUTINE PUTTIME( AD,TIME )
C
C           Set time synthesizer delay to TIME
C
         byte   BUFFER(32), WRK0(80)
         character   CTIME*10
         integer*2   ISTAT(4)
         integer*4   AD
         byte  ADDR
         equivalence ( BUFFER(2), CTIME )
        common      /IBLK/ WRK0,BUFFER
         ADDR = AD+32
         ENCODE( 10,10100, CTIME ) TIME
10100    format( E10.3 )
         BUFFER(1) = 'D'
         BUFFER(12) = ','
         CALL IESEND( WRK0, BUFFER,12, ADDR,1, ISTAT )
         RETURN
         end

       SUBROUTINE GETTIME( AD,D )
C
C         Get Time delay from synth
C
         byte  BUFFER(32), WRK0(80), STATE(66)
         byte  ADDR
         integer*2   DAVE(66), ISTAT(4)
         integer*4   AD
         real        U(12), P(12), G(7)
        common    /IBLK/ WRK0,BUFFER
         CALL GETDAVE( AD,DAVE )
         U(12) = REAL(DAVE(12))
         B = 0.
         DO 200 I= 6,11
         U(I)  = REAL(DAVE(I))
         P(I)  = U(I) - 6.*REAL(( IINT( U(I)/16.) ))
         G(I-5)= P(I)*( 10.**( -2*(I-5) ))
200      B=B + G(I-5)

         A = U(12)-244.
         D = B * ( 10.**( A-12.) )
         RETURN
         end

C
       SUBROUTINE PUTSTATE( AD,STATE )
C
C           Put State into Time Synth
C
         byte  BUFFER(32), WRK0(80), STATE(66),ADDR
         integer*2   ISTAT(4)
         integer*4   AD
        common    /IBLK/ WRK0,BUFFER
         ADDR = AD+32
```

The algorithm for converting the array DAVE(66) to the time delay D was developed by Alex Vary, NASA Lewis Research Center, Cleveland, Ohio.

```
      BUFFER(1) = 'D'
      BUFFER(2) = 13
      CALL IESEND( WRK0, BUFFER,2, ADDR,1, ISTAT )
      BUFFER(1) = 'L'
      BUFFER(2) = 'N'
      CALL IESEND( WRK0, BUFFER,2, ADDR,1, ISTAT )
      CALL IESEND( WRK0, STATE,66, ADDR,1, ISTAT )
      RETURN
      end

      SUBROUTINE GETSTATE( AD,STATE )
C
C        Get State from Time Synth
C
      byte   BUFFER(32), WRK0(80),  STATE(66)
      byte   LADDR, TADDR
      integer*2  ISTAT(4)
      integer*4  AD
      common      /IBLK/ WRK0,BUFFER

      TADDR = AD+32
      LADDR = AD+64
      BUFFER(1) = 'D'
      BUFFER(2) = 13
      CALL IESEND( WRK0, BUFFER,2, TADDR,1, ISTAT )
      BUFFER(1) = 'T'
      BUFFER(2) = 'E'
      BUFFER(3) = 13
      CALL IESEND( WRK0, BUFFER,3, TADDR,1, ISTAT )
      CALL IERECV( WRK0, STATE,66, ,, LADDR, ISTAT )

      RETURN
      end
C
      SUBROUTINE GETDAVE( AD,DAVE )
C
C        Get State from Time Synth as integers with stuff
C        in the MSB
C
      byte   BUFFER(32), WRK0(80),  STATE(66), B
      byte   TADDR, LADDR
      integer*2  DAVE(66), ISTAT(4), I
      integer*4  ad
      character  MSB*2, LSB*1
      equivalence  ( MSB,I )
      equivalence  ( LSB,B )
      common      /IBLK/ WRK0,BUFFER

      TADDR = AD+32
      LADDR = AD+64
      BUFFER(1) = 'D'
      BUFFER(2) = 13
      CALL IESEND( WRK0, BUFFER,2, TADDR,1, ISTAT )
      BUFFER(1) = 'T'
      BUFFER(2) = 'E'
      BUFFER(3) = 13
      CALL IESEND( WRK0, BUFFER,3, TADDR,1, ISTAT )
      CALL IERECV( WRK0, STATE,66, ,, LADDR, ISTAT )
      I=0
      K=0
```

```
        DO 700  J=1,20          !
        B       = STATE(J+K)    !  Here is where we stick STATE
        MSB(1:1)= LSB           !  into the MOST SIGNIFICANT BYTE
        DAVE(J+K) = I           !  of DAVE
        B       = STATE(J+K+1)
        MSB(1:1)= LSB
        DAVE(J+K+1) = I
  700   K=K + 1
        RETURN
        end

C
        SUBROUTINE SETXYZ( AD,I1, A2,I2,A3,I3 )
C
C         Initialize Klinger with step values
C
        byte    BUFFER(32), WRK0(80)
        byte    ADDR, BSTEP(6)
        integer     ISTEP(3),AD,A1,A2,A3
        integer*2   ISTAT(4)
        character   ABUFFER*25, ASTEP*6
        equivalence ( ABUFFER, BUFFER )
        equivalence ( ASTEP, BSTEP )
        common      /IBLK/ WRK0,BUFFER
10020   format( I )
10026   format( I6 )
10124   format( ' ',A32 )
        ISTEP(1) = I1
        ISTEP(2) = I2
        ISTEP(3) = I3
        ABUFFER(1:5) =  'R 253'
        BUFFER(6)  =13
        ABUFFER(7:9) =  'S 2'
        BUFFER(10) =13
        ABUFFER(11:14) ='F 20'
        BUFFER(15) =13
        BUFFER(16) =    'N'
        BUFFER(17)=' '

        DO 200 I=1,3
        NDIG = IFIX( ALOG10(REAL(ISTEP(I))) )+1
        N6M  = 6-NDIG
        ENCODE( 6,10026, ASTEP )ISTEP(I)
        DO 90 IDIG=1,NDIG
   90   BUFFER(17+IDIG) = BSTEP(N6M+IDIG)
        BUFFER(NDIG+18) = 13
        BUFFER(NDIG+19) = 'A'
        BUFFER(NDIG+20) = 13
        NSEND = 20+NDIG
        ADDR  = AD+32+(I-1)
  200   CALL IESEND( WRK0, BUFFER,NSEND, ADDR,1, ISTAT )
        RETURN
        end

        SUBROUTINE MOVXYZ( AD,IAXIS, IPORN )
C
C          Move Klinger one step
C
        byte    BUFFER(32), WRK0(80), ADDR
```

```
        integer*2   ISTAT(4), LISTEN
        integer*4   AD

       common       /IBLK/ WRK0,BUFFER
        ADDR = AD + 32
        IF( IPORN.GT.0 )BUFFER(1) = '+'
        IF( IPORN.LT.0 )BUFFER(1) = '-'
        BUFFER(2) = 13
        BUFFER(3) = 'G'
        BUFFER(4) = 13
        CALL IESEND( WRK0, BUFFER,4, ADDR,1, ISTAT )
        RETURN
        end
C
       SUBROUTINE GETFLUKE( AD,VOLTS )
C
C          Get value from Fluke Multimeter
C
        byte      BUFFER(32), WRK0(80)
        character ABUFFER*32
        byte      ADDR
        integer*2    ISTAT(4)
        integer*4    AD
        equivalence  ( BUFFER, ABUFFER )
       common   /IBLK/ WRK0,BUFFER
   10   ADDR  = AD+32
        ABUFFER(1:3) = 'VC?'
        CALL IESEND( WRK0, BUFFER,3, ADDR,1, ISTAT )
        ADDR  = 69
        CALL IERECV( WRK0, BUFFER,14, ,, ADDR, ISTAT )
        IF( BUFFER(1).NE.'+' .AND. BUFFER(1).NE.'-' )GOTO 10
        DECODE( 12,10040, ABUFFER(1:12) )VOLTS
10040   format( E )
        RETURN
        end

       SUBROUTINE WAIT2(NWAIT)

        DO 1000 IWAIT=1,NWAIT
        DO 500 I=1,NWAIT
        do 501 j=1,nwait
 501    continue
  500   CONTINUE
 1000   CONTINUE
        RETURN
        END


       SUBROUTINE MOVTESTECH( AD, IXYORZ, IDEST )
C
C          Move TESTECH scanner IXYORZ axis ( 1,2,3,4 )=( X,Y,Z,S )
C          to destination IDEST

        BYTE   BUFFER(48), WRK0(80)
        BYTE   IXYORZ,B,B6(6),SCODE(4),CR,TADDR,LADDR
        INTEGER*4 AD
        CHARACTER*6 C6
        EQUIVALENCE ( B6,C6 )
        COMMON /IBLK/   WRK0,BUFFER
        DATA      SCODE/ 68,70,87,73 /, CR/13/
```

```
        TADDR = AD+32
        LADDR = AD+32
        IF( IDEST.LT.0 )THEN
                    TYPE *,' ILLEGAL DESTINATION '
                    GOTO 900
                    ENDIF
        BUFFER(1) = SCODE( IXYORZ )
C
C   Convert IDEST to a 6-byte character string
C   with leading zeros and place in BUFFER(2:7)
C
        ENCODE( 6,10070, C6 )IDEST
        DO 100 I=1,6
           B=B6(I)
           IF( B.EQ.32 )B=48
  100      BUFFER(I+1) = B
10070   FORMAT( I6 )
        BUFFER(8) = CR
C
C   Call IESEND to move
C
        CALL IESEND( WRK0, BUFFER,8, TADDR,1, ISTAT )
        CALL IERECV( WRK0, BUFFER,2, ,, LADDR, ISTAT )
        IF( IXYORZ.EQ.4 )THEN
            BUFFER(1) = 'Z'
            BUFFER(2) = CR
            CALL IESEND( WRK0, BUFFER,2, TADDR,1, ISTAT )
            CALL IERECV( WRK0, BUFFER,2, ,, LADDR,1, ISTAT )
            ENDIF
  900   RETURN
        END
C
        SUBROUTINE FINDTESTECH( AD,IXPOS,IYPOS,IZPOS )
C
C        Ask TESTECH scanner for current position and
C        return X, Y, and Z axis positions as integers
C        in units of thousandhs of an inch
C
        BYTE    WRK1(80), BUFFER(48)
        BYTE    GETPOS(62), CR
        INTEGER*2   ISTAT(4)
        INTEGER*4   AD
        CHARACTER*6  XPOS, YPOS, ZPOS
        EQUIVALENCE  ( XPOS, GETPOS(1) ), ( YPOS, GETPOS(7) )
        EQUIVALENCE  ( ZPOS, GETPOS(13) )
        COMMON  /IBLK/ WRK1, BUFFER
        DATA     CR/13/

        TADDR = AD+32
        LADDR = AD+64
        BUFFER(1) = 'K'
        BUFFER(2) = CR
        CALL IESEND( WRK1, BUFFER,2, TADDR,1, ISTAT )
        CALL IERECV( WRK1, GETPOS,62, ,, LADDR, ISTAT )
                IF( ISTAT(1).NE.1 )CALL LIB$SIGNAL(%VAL(ISTAT(1)))

        DECODE( 6,10060, XPOS )IXPOS
        DECODE( 6,10060, YPOS )IYPOS
        DECODE( 6,10060, ZPOS )IZPOS
10060   FORMAT( I6 )
```

```
        RETURN
        END
C
C
C
       SUBROUTINE GETA2D( AD, DATA )
C
C   Get one word of data from channel 4 of the H.P. A/D converter
C
        BYTE      WRK0(80), BUFFER(48)
        BYTE      A(2),B(2), TADDR, LADDR
        INTEGER*2   D,DATA, ISTAT(4)
        INTEGER*4   AD
        EQUIVALENCE ( B,D )
        COMMON /IBLK/ WRK0, BUFFER
10010   FORMAT( A3 )
10020   FORMAT( A2 )
        TADDR = AD+32
        LADDR = AD+64
C
C   Send  "H8A", wait, then send "JF"
C
        ENCODE( 6,10010, BUFFER )'H8A'
        CALL IESEND( WRK0, BUFFER,3, TADDR,1, ISTAT )
        CALL WAIT3(1)
        ENCODE( 6,10020, BUFFER )'JF'
        CALL IESEND( WRK0, BUFFER,2, TADDR,1, ISTAT )
C
C   Get data, put 1st byte in DATA's 2nd byte, etc
C
        CALL IERECV( WRK0, A,2, ,, LADDR, ISTAT )
        B(1) = A(2)
        B(2) = A(1)
        DATA = D

        RETURN
        END

       SUBROUTINE WAIT3(NWAIT)

        DO 1000 IWAIT=1,NWAIT
        DO 500 I=1,1000
  500   CONTINUE
 1000   CONTINUE
        RETURN
        END
```

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-4106		

4. Title and Subtitle	5. Report Date
Interfacing Laboratory Instruments to Multiuser, Virtual Memory Computers	March 1989
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Edward R. Generazio, David B. Stang, and Don J. Roth	E-4510
	10. Work Unit No. 535-07-01

9. Performing Organization Name and Address	11. Contract or Grant No.
National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135-3191	
	13. Type of Report and Period Covered Technical Memorandum
12. Sponsoring Agency Name and Address	
National Aeronautics and Space Administration Washington, D.C. 20546-0001	14. Sponsoring Agency Code

15. Supplementary Notes

Edward R. Generazio and Don J. Roth, NASA Lewis Research Center; David B. Stang, Sverdrup Technology, Inc., NASA Lewis Research Center Group, Cleveland, Ohio 44135.

16. Abstract

Incentives, problems and solutions associated with interfacing laboratory equipment with multiuser, virtual memory computers are presented. The major difficulty concerns how to utilize these computers effectively in a medium sized research group. This entails optimization of hardware interconnections and software to facilitate multiple instrument control, data acquisition and processing. The architecture of the system that was devised, and associated programming and subroutines are described. An example program involving computer controlled hardware for ultrasonic scan imaging is provided to illustrate the operational features.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
GPIB; Interfacing; Computers; Instruments; Software; Fortran; Image processing; Ultrasonics; Array processing	Unclassified – Unlimited Subject Category 38

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No of pages	22. Price*
Unclassified	Unclassified	32	A03

NASA FORM 1626 OCT 86

*For sale by the National Technical Information Service, Springfield, Virginia 22161

NASA-Langley, 1989