AD-A119 976 ARMY MISSILE COMMAND REDSTONE ARSENAL AL RESEARCH D--ETC F/G 9/2
FORTH AND THE IEEE-488 BUS.(U)
JUL 82 K B FARR, J G DUTHIE
UNCLASSIFIED DRSMI/RR-82-9-TR SBI-AD-E950 296 NL

(12)

TECHNICAL REPORT RR-82-9

FORTH AND THE IEEE-488 BUS

K. B. Farr
J. G. Duthie
Research Directorate
US Army Missile Laboratory

2 JULY 1982

U.S. ARMY MISSILE COMMAND
Redstone Arsenal, Alabama 35898

Approved for public release; distribution unlimited.

DISPOSITION INSTRUCTIONS

DESTROY THIS REPORT WHEN IT IS NO LONGER NEEDED. DO NOT RETURN IT TO THE ORIGINATOR.

DISCLAIMER

THE FINDINGS IN THIS REPORT ARE NOT TO BE CONSTRUED AS AN OFFICIAL DEPARTMENT OF THE ARMY POSITION UNLESS SO DESIGNATED BY OTHER AUTHORIZED DOCUMENTS.

TRADE NAMES

USE OF TRADE NAMES OR MANUFACTURERS IN THIS REPORT DOES NOT CONSTITUTE AN OFFICIAL INDORSEMENT OR APPROVAL OF THE USE OF SUCH COMMERCIAL HARDWARE OR SOFTWARE.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RR-82-9	2. GOVT ACCESSION NO. AD-A119976	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) FORTH AND THE IEEE-488 BUS		5. TYPE OF REPORT & PERIOD COVERED TECHNICAL REPORT
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) K. B. Farr and J. G. Duthie		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Commander, US Army Missile Command ATTN: DRSMI-RR Redstone Arsenal, AL 35898		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Commander, US Army Missile Command ATTN: DRSMI-RPT Redstone Arsenal, AL 35898		12. REPORT DATE 2 July 1982
		13. NUMBER OF PAGES 20
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

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

18. SUPPLEMENTARY NOTES

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

Computers
Electronic Control
Software
Forth
IEEE-488 Standards

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

This report demonstrates the natural marriage of the IEEE-488 bus to microcomputers using FORTH in a laboratory situation.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

BLANK PAGE

TABLE OF CONTENTS

I INTRODUCTION

The IEEE-488[1] standard interface (occasionally referred to as the HPIB or the GPIB) bus is used to interconnect many laboratory instruments. The use of the IEEE-488 bus facilitates the operation of experiments by standardizing the means of intercommunicating data or commands between elements of the experiment. The IEEE-488 bus was originally proposed by Hewlett Packard in 1972. It was adopted as an IEEE standard in 1975 and is now offered by many manufacturers as an option on several hundreds of pieces of equipment used for control, test and measurement purposes.

The IEEE-488 bus uses a standard 24 pin connector daisy chaining a 24 wire cable between up to 15 devices. The 24 wires include 8 data lines, 3 hand shaking lines, 5 bus management lines, 7 grounds and one shield. One element on the bus is the controller (frequently a mini or micro computer) which manages the system and directs traffic on the data and control lines.

Programming of the IEEE-488 bus controller may be done in assembly language or in a high level language such as BASIC. While assembly language programming may yield the most compact and efficient operating system, it is not easy for the operator to work with if he needs to change routines in the middle of operating the experiment. On the other hand, however, although BASIC is easily programmed, it is relatively slow to operate and it is difficult to implement the sort of flexibility one often needs in laboratory situations. The threaded language FORTH offers a compromise between the efficiency of assembly language and the user friendliness of BASIC while offering the operator the maximum flexibility in operating an experiment.

FORTH [2] was originally developed at the National Radio Astronomical Observatory as an efficient means to control the operation of, and data acquisition from radio telescopes. It was further developed at Kitt Peak National Observatory where it provided excellent interface between computers, scientific equipment and observers. Other laboratories such as the University of Rochester's Laboratory for Laser Energetics quickly realized the applicability of FORTH in their operations. FORTH is currently available on mainframe computers, minicomputers and on microcomputers. At the microcomputer end where we have our application FORTH or FORTH-like languages such as URTH or STOIC are now widely available from a large variety of sources.

It is the purpose of this report to demonstrate the natural marriage of the IEEE-488 bus to microcomputers using FORTH in a laboratory situation. Like the IEEE-488 bus, there is a standard FORTH. However, different versions of FORTH have their idiosyncrasies as do individual pieces of IEEE-488 hardware both of which are supposed to be standard. We hope that by documenting our efforts to control the IEEE-488 bus in FORTH we can illustrate the procedures we have employed and indicate some of

the difficulties one can meet due to different manufacturers interpretations of the standards.

II THE IEEE-488 BUS

The IEEE-488 bus includes 8 data lines, 3 handshake lines and 5 bus management lines. The data lines, labelled DIO1 through DIO8, allow for bit parallel byte serial data exchange. DIO8 is the most significant bit (MSB). The three handshake lines are

i) DAV - Data valid
ii) NRFD - Not ready for data
iii) NDAC - Not data accepted

The five bus management lines are

i) EOI - End or identify
ii) IFC - Interface clear
iii) SRQ - Service request
iv) REN - Remote enable
v) ATN - Attention

The bus uses an active-low principle in which the logical 0 or false corresponds to +5 volts while a logical 1 or true corresponds to 0 volts. Thus many devices can be connected to a common line and if any device wishes to assert a line true it has the authority to do so. A line can only be false if none of the devices connected to it are trying to hold it true. Up to 15 devices can be on the bus at any time.

Handshaking is required of all information exchanges. One device *(frequently but not always the computer) is the talker* while others on the bus are listeners. Not all the devices are either talkers or listeners at any time - they have to be instructed by the controller to be such, or by default sit passively on the bus.

Handshaking is performed as follows. All the listeners should be asserting NDAC true. Following the previous handshake or immediately after being commanded to become a listener some listeners may assert NRFD true but this is a temporary situation and within a few milliseconds all listeners will assert the NRFD line false. If both NRFD and NDAC are false this indicates that no listeners are on the bus - an error condition which must be taken care of. The talker senses the NRFD lines and waits until it is false indicating that the slowest listener is ready to accept data. The talker then outputs the data on lines DIO1 through DIO8. It waits until the data has settled down then the talker asserts DAV true. The fastest listener asserts NRFD true followed by all the others. Each listener at its own speed accepts the data byte and releases NDAC. When the slowest listener has responded the talker will sense the NDAC line as being false indicating the byte transfer is complete and it then sets DAV false . All the listeners set NDAC true and, when ready to receive the next data they will set NRFD false.

Whether a device is a talker, a listener or neither is under

the control of the controller. The ATN line may be asserted true only by the controller. With ATN true all information on the data lines is regarded as a command and is listened to by all devices on the bus. While ATN is false the information on the data lines is considered as a message or data to be read by designated listeners.

All IEEE-488 devices have a 5 bit address which may be fixed or may be set by means of switches on or in the device. This is the device's basic address and may have a value between 0 and 31. The address 31 is reserved for a special purpose however. If the controller sets ATN true it can command devices to become listeners or talkers. The commands are simply the basic address of the device plus 32 for a listener or the basic address plus 64 for a talker. Thus if a device had address 4 then to command it to be a listener the controller would, with ATN true, issue the binary equivalent of 36 on the data lines. If the base address 31 is transmitted with 32 or 64 then the controller is telling all devices to become non listeners or non talkers respectively. These are known as the UNIVERSAL UNLISTEN and UNIVERSAL UNTALK commands. When an address is sent with both lines DIO6 and DIO7 true then this is known as a secondary address and is discussed further below in the example showing use of the micropositioning equipment.

The remaining IEEE-488 control lines are IFC, SRQ, REN, and EOI. When IFC is true all the instruments are forced into a defined condition which is generally unlisten and untalk. The line SRQ is used by instruments to indicate that they are in a condition that needs attention by the controller or some human intervention. Such a request could be made for example by a motor drive controller if the motor hits some limit switch. It is necessary for the controller to respond to a service request by polling all devices to see which one called for service and then taking necessary corrective measures.

Many instruments have front panel controls which allow them to be operated independently from the IEEE-488 bus. For these devices to operate under bus control it is essential for the controller to assert REN true. At the end of each transmission of a set of bytes EOI is made true by the talker. This is done in conjunction with the transmission of the last byte.

There are situations where more than one of the control lines are set true simultaneously. For some of these there is a special interpretation of the control function. For example the combination of EOI and ATN both true signals the start of a parallel poll to check out a service request call. Such special combinations are discussed in the IEEE-488 standards documentation and except for a few special cases will not be treated here.

III FORTH

FORTH is a programming language which offers speed and flexibility. It is highly interactive with the user and in addition requires very little memory (about 7k for the entire system). Near error-free programs can easily be obtained because any part of a routine can be tested before it is compiled. Originally developed for control applications, FORTH lends itself well to the management of the IEEE-488 bus.

Being a threaded language, programming in FORTH is done by building new functions (similar to subroutines) from previously defined functions until finally, one performs the desired task. Initially FORTH is a kernel of about one hundred functions (words). New words are compiled as they are defined and immediately become part of FORTH (no subroutine calls are needed). These new words may be used to define other words. Thus the vocabulary of FORTH grows outward towards higher memory.

A word may be invoked (interpreted) at any time simply by typing it on the console. FORTH's outer interpreter, at that time, finds the word in its dictionary and executes it. If the word is not found an error message occurs informing the operator that he is attempting to use an undefined word.

Definition of a new word takes the form

: nnnn < definition (old FORTH words) > ;

The word ":" tells FORTH to create a new word with the label nnnn. Subsequent words up to ";" will be compiled instead of executed. If the label nnnn has already been used, a redefine message is given. The old word may not be accessed directly afterwards, but earlier routines using that word are not affected. The latest definition of a word is always executed by the interpreter.

Access to machine code is allowed by the word CODE. This word invokes the FORTH assembler and allows machine-language subroutines to be defined in the form:

CODE nnnn < definition > EDOC.

Words defined in this way are compiled and become part of the FORTH vocabulary just like any other FORTH word. In this manner all the efficiency of assembly language is offered along with the programming ease of the higher-level language.

Arithmetic operations are done in FORTH using a stack and postfix notation system much like that used in many hand calculators. Arithmetic formulae are written in postfix notation with the operators after the arguments instead of between them, with the arguments taken off the stack on a last in, first out basis. For example, to add the numbers 5 and 7 one would enter the FORTH word 5, which places the number 5 on the stack followed by the word 7 and finally the word + which removes the top two numbers from the stack and leaves their sum on the top of the stack. Almost all FORTH operations communicate only through the

stack, taking data from the top of the stack and leaving results on the top of the stack.

IV IEEE-488 SOURCE HANDSHAKE

In order for the talker (in this case a microcomputer) to transfer data to a listener, a software handshake must be implemented. This handshake must be able to manipulate the three handshake lines and place data on the eight data lines in accordance with the IEEE-488 standards as discussed above. We will discuss here how that handshake may be implemented in FORTH.

First of all the talker must determine if there are any listeners on the bus, and if not output an error message and return control to the operator for corrective action. This error condition is indicated by the handshake lines NDAC and NRFD both being false (high).

To bring the handshake line information from the port and place it on the stack, the word ?EARSUP is defined from the FORTH assembly language. All numbers are hexadecimal.

```
CODE ?EARSUP 7D IN 60 AND 0 H LD A L LD HL PUSH $NEXT JP EDOC
```

This brings in the handshake and management data from port 7D to the accumulator, logically ands it with a hexadecimal 60 to mask off all but the two lines we are interested in, puts a 0 in the H register and loads the contents of the accumulator into the L register, then pushes the HL register pair onto the stack.The ports for control and data on the IEEE-488 bus in our setup are 7D and 7E respectively.

The word to be executed on the error condition is:

```
: MS1 T" NO LISTENERS ON 488 BUS " CR RESTART ;
```

The error message is printed on the screen (with a carriage return) and FORTH is restarted.

These two words are used in the following routine which performs the entire function:

```
: EARSUP ?EARSUP 60 - 0= IF MS1 ENDIF ;
```

MS1 is executed if and only if ?EARSUP places a 60 on the stack.

Next the talker must place the data to be transferred onto the data bus, wait at least two milliseconds for the lines to settle, then determine that all listeners are ready for data.

OUTDATA places the number on top of the stack out to the IEEE-488 data port 7E, performing a ones complement to compensate for the IEEE-488 high-false logic.

```
CODE OUTDATA HL POP L A LD CPL 7E OUT $NEXT JP EDOC
```

That the listeners are ready for data is indicated by NRFD being false. No action is taken until this condition is satisfied.

```
: ?READY BEGIN ?EARSUP 40 & 40 - 0= END ;
```

The word & is the FORTH logical AND of the top two numbers on the stack.

Now the talker must wait at least two milliseconds before indicating that the data is available. This is accomplished by redefining ?READY:

```
: ?READY ?READY WAIT ;
```

where WAIT is essentially a delay loop.

These words are now combined.

```
: OUTEM OUTDATA ?READY DAV ON ZAPP ;
```

The words DAV ON ZAPP force the handshake line DAV true, informing the listeners that the data on the bus is valid.

The talker now determines that the listeners have accepted the data, and makes the DAV line false indicating that the data lines no longer carry valid data.

```
: ?AXCEPT BEGIN ?EARSUP 20 - 0= END ;
: ENDSHAKE ?AXCEPT DAV OFF ZAPP 00 OUTDATA ;
```

Now the above words can be tied together into one word wh will carry out the entire source handshake, sending the top the stack over the data lines to the listeners.

```
: CHRSEND EARSUP OUTEM ENDSHAKE ;
```

Thus the sequence 24 CHRSEND will result in the number 24 being sent out by the talker (the computer) and received by all the listeners on the bus. Care should be paid to the current numeric base being utilized by FORTH. The base can be binary (base 2) octal (base 8) decimal (base 10) or hexadecimal (base 16). For most control applications decimal or hexadecimal numbers are used. In this article hexadecimal numbers prevail.

V THE FORTH DRIVERS

The particular hardware configuration used by each experimenter will differ according to his individual needs. Thus FORTH routines will have to be developed for his situation. The listing of FORTH routines in Appendix A have been created to use with a particular set of laboratory hardware. Although the set may not exactly match that found in other laboratories they are probably typical of what one might expect to find. The solution to IEEE-488 interfacing in other laboratories can be expected to resemble the ones we present here. In the present case the controller is a NorthStar Horizon microcomputer with two double density disc drives operating under CP/M (CP/M is a trademark of Digital Research of Pacific Grove CA.). The version of FORTH used is SL5 (SuperSoft Associates, Champaign ILL) and the IEEE-488 bus controller was a Pickles and Trout 488 board.

Specific hardware to be controlled in our experiments were

1) Quantex Model DS-30 Digital Video Processor.

2) Hewlett Packard Model 9872A Plotter.

3) Two Ealing System 5 micropositioners

(One vertical and one horizontal)

The routines we have developed are given in Appendix A . A brief summary of these follows.

Upon loading the bus drivers the first thing done is to store the current arithmetic base used by FORTH then designating the base as HEXadecimal. (The words BASE @ store the current base on the stack while HEX makes the base hexadecimal). This is followed by a definition of a word to set the base to binary for future use.

A series of variables and constants are then defined. GIMTCH is a variable whose initial value is zero as defined in the statement

```
0 VARIABLE GIMTCH
```

The purpose of GIMTCH is to store the data which will be put on the handshake and bus management lines. Similarly the variable BUS is used to store the actual state of the bus.

A set of words is then defined culminating in the definition of the word ZAPP which places the contents of GIMTCH on the eight handshake and management lines.

The diagnostic word LINES is defined from the previous words to give the status of the handshake and management lines thus :

```
DAV... ON
NRFD.. OFF
NDAC.. OFF
IFC... ON
ATN... OFF
SRQ... OFF
REN... ON
EOI... OFF
```

for example.

These words are then followed by routines which allow the controller to perform source and acceptor handshakes much as described above. The listing then shows the definitions of the the IEEE-488 addresses (in HEX) of the various devices in the system as well as the definition of an initialization routine for the Pickles and Trout interface board.

The remainder of the listing gives the definitions of FORTH words specific to the devices in the particular laboratory situation of our application. Finally the system performs an INIT to set up the interface board, rings a bell on the console (7 TCH) and re-establishes the base to what it was prior to the loading of the routines

VI EXAMPLES

In this section examples will be given to demonstrate the

use of the software in appendix A in a laboratory situation.

The Quantex digital video processor is capable of storing in its memory one frame of input video and can output to a monitor its input, its memory, or some processed version of its memory. In order to access the Quantex unit over the IEEE-488 bus, it must first be assigned as a listener. This is done by typing

QUANTEX LISTENER

QUANTEX is a constant (04, the address of the video processor) and LISTENER carries out the controller function of assigning a device as a listener, taking the number on the top of the stack as the device address.

Now to save a frame of video input, one would type

STORE-INPUT

If the difference between the input and memory is desired

DIFF

displays the mathematical difference between the two digitized scenes on the monitor.

To compare the difference between the input scene and the memory, one might define the following:

```
: COMPARE 100 0 DO INPUT PAUSE MEMORY PAUSE LOOP ;
```

which will cause the video processor to switch from input to memory display 100 hex times (256 decimal). To execute this command, simply type

COMPARE

We now wish to drive the Ealing micropositioners. Note that the Ealing System 5 hardware makes extensive use of secondary addresses. These are addresses of internal parts of the hardware or software that must be addressed in order to transfer data to or from those parts. These addresses must be sent with ATN true after the primary address has been sent. The secondary listen addresses cover the functions local, remote, drive, reverse, forward, travel and speed while secondary talk addresses are used for transmission of travel and speed data. These addresses can be seen under the EALING COMMANDS in Appendix A to range from 61 to 68 HEX. Another complication is that for some commands such as TRAVEL the Ealing system is looking for a three byte data transfer whereas FORTH uses two byte numbers. The capability of using double precision numbers is available in FORTH but in our application these are seldom if ever used. In our application we simply use single precision numbers and transmit zero for the high order byte.

For illustrative purposes we will discuss one of the EALING COMMANDS in some detail. The word TRAVEL consists of the following sequence. First the word TRAVEL expects a number to be on the stack. This is the number of discrete steps the stage is going to have to make. The word TRAVEL first checks to see if the system is ready to receive data. Next ATN is set true and a 3F is sent out to make everyone into an unlistener. The variable EALING contains the basic address of the device to be made into a

listener. A logical OR (FORTH word |) is performed on the basic device address and the hex number 20, the result being transmitted to the bus making the device a listener. With ATN still true the secondary address 66 is sent to advise the unit to expect three bytes of travel data. ATN is released. The word TWOBITS takes the number on top of the stack and converts it into two numbers for transmission to the micropositioner low order byte first. Finally the high order byte zero is sent with EOI true and the computer ends the transmission with the data lines cleared.

To drive the vertical stage 200 steps at 50 steps per second we would type:

```
DECIMAL VERTICAL REMOTE 50 SPEED 200 TRAVEL FORWARD DRIVE
```

DECIMAL converts to base 10, VERTICAL indicates which unit to address and DRIVE initiates the motion of the positioners.

If a raster scan of 500 x 500 micrometers (one step is equal to one micrometer) is desired, the following word may be written:

```
: RASTER HORIZONTAL 100 SPEED VERTICAL 10 SPEED
         500 0 DO HORIZONTAL I 2 MOD 0= IF FORWARD
         ELSE REVERSE ENDIF
         500 TRAVEL DRIVE
         VERTICAL FORWARD 1 TRAVEL DRIVE LOOP ;
```

A useful facility that has been developed is the routine VIEWGRAPH. Using a commercially available word processor, a CP/M text file may be edited and saved onto disc. The operator then, after loading FORTH, types the command VIEWGRAPH. The computer responds with "ENTER THE NAME OF THE VIEWGRAPH FILE". After the file name is entered the text is then printed on the plotter. A useful feature of the routine is that it senses when the text contains the reserved charachter ASC(47H). If this charachter is present the plotting is suspended to allow the user to enter a single FORTH word which is the executed before plotting resumes. This can be used, for example,to change plotter pens. This is a valuable tool for producing text viewgraphs. Especially useful results are obtained if transparent film and special pens are used.

VII CONCLUSIONS

We have developed software for control of the IEEE-488 interface using FORTH as a programming language. The resulting set of routines give a fast, flexible operating environment for an experimenter working with a laboratory system under computer control. The elementary routines in Appendix A can be individually executed or new FORTH words can be easily defined to combine earlier defined words into a single command.

The laboratory situation in which we have implemented FORTH drivers for the interface bus may be typical of others. In each case however it is to be expected that different pieces of hardware will be used on the IEEE-488 bus. Routines similar to those given in Appendix A will have to be individually tailored. We hope this discussion will provide a useful basis for other workers.

We have not discussed details of some additional features of the IEEE-488 interface. Included are the ways to respond to a service request (SRQ), the use of secondary addresses, the correct method of ending a set of instructions with an EOI and the use of serial polls. These are all implemented in the sample of FORTH routines given in Appendix A. For more details on the IEEE-488 bus the reader is referred to reference 1.

BIBLIOGRAPHY

1) IEEE Standard Digital Interface for Programmable Instrumentation. IEEE Std. 488-1978
2) Moore, C. H. , "FORTH a New Way to Program a Computer ",Astronomy and Astrophysics Supp 15,497,1974

APPENDIX A

```
BASE @ HEX
: BINARY 2 BASE ! ;
0 VARIABLE GIMTCH  0 VARIABLE BUS
80 CONSTANT DAV  40 CONSTANT NRFD  20 CONSTANT NDAC  10 CONSTANT IFC
08 CONSTANT ATN  04 CONSTANT SRQ   02 CONSTANT REN   01 CONSTANT EOI

( CONTROL LINE MANIPULATION COMMANDS )
CODE TOGGLE HL POP L A LD CPL A L LD HL PUSH $NEXT JP EDOC
: ON TOGGLE GIMTCH @ & GIMTCH ! ;
: OFF GIMTCH @ | GIMTCH ! ;
CODE CMND HL POP L A LD 7D OUT HL PUSH $NEXT JP EDOC
: COMMAND CMND GIMTCH ! ;
: ZAPP GIMTCH @ COMMAND ;

( CONTROL LINE INQUIRY COMMANDS )
: RESPOND 0= IF T" ON " ELSE T" OFF " ENDIF CR ;
: ?DAV T" DAV... " BUS @ 80 & RESPOND ;
: ?NRFD T" NRFD.. " BUS @ 40 & RESPOND ;
: ?NDAC T" NDAC.. " BUS @ 20 & RESPOND ;
: ?IFC T" IFC... " BUS @ 10 & RESPOND ;
: ?ATN T" ATN... " BUS @ 08 & RESPOND ;
: ?SRQ T" SRQ... " BUS @ 04 & RESPOND ;
: ?REN T" REN... " BUS @ 02 & RESPOND ;
: ?EOI T" EOI... " BUS @ 01 & RESPOND ;
CODE LINES 7D IN 0 H LD A L LD HL PUSH $NEXT JP EDOC
: >LINES LINES BUS ! CR ?DAV ?NRFD ?NDAC ?IFC ?ATN ?SRQ ?REN ?EOI ;

( IEEE 488 SOURCE HANDSHAKE COMMANDS )
: MS1 T" NO LISTENERS ON 488 BUS " CR RESTART ;
CODE ?EARSUP 7D IN 60 AND 0 H LD A L LD HL PUSH $NEXT JP EDOC
: EARSUP ?EARSUP 60 - 0= IF MS1 ENDIF ;
: ?READY BEGIN ?EARSUP 40 & 40 - 0= END ;
( ONES COMPLEMENT [CPL] PREFORMED IN OUTDATA TO COMPENSATE FOR )
( PICKLE AND TROUT NEGATIVE LOGIC )
CODE OUTDATA HL POP L A LD CPL 7E OUT $NEXT JP EDOC
: WAIT 2 0 DO I DROP LOOP ;
: ?READY ?READY WAIT ;
: OUTEM OUTDATA ?READY DAV ON ZAPP ;
: ?AXCEPT BEGIN ?EARSUP  20 - 0= END ;
: ENDSHAKE ?AXCEPT DAV OFF ZAPP 00 OUTDATA  ;
: CHRSEND  EARSUP OUTEM ENDSHAKE ;

( ACCEPTOR HANDSHAKE )
CODE INDATA  7E IN CPL A L LD 0 H LD HL PUSH $NEXT JP EDOC
: INDATA 00 OUTDATA INDATA ;
: ?DAVON BEGIN LINES 80 & 0= END ;
: ?DAVOFF BEGIN LINES 80 & 80 - 0= END ;
: CHRRCV NRFD ON NDAC ON ZAPP WAIT NRFD OFF ZAPP ?DAVON
        NRFD ON ZAPP INDATA NDAC OFF ZAPP ?DAVOFF NDAC ON ZAPP ;
: >DATA BASE @ INDATA BINARY . BASE ! ;

( IEEE-488 DEVICE ADDRESSES )
03 CONSTANT NORTHSTAR  04 CONSTANT QUANTEX  05 CONSTANT PLOTTER
0B CONSTANT VER-EAL    0C CONSTANT HOR-EAL
```

APPENDIX A

```
( PICKLE AND TROUT IEEE 488  INITIALIZATION )
CODE 1INIT FF A LD 7F OUT 7E OUT A L LD 0 H LD HL PUSH $NEXT JP EDOC
CODE 2INIT 0 A LD 7C OUT $NEXT JP EDOC
: INIT 1INIT COMMAND 2INIT REN ON ZAPP ;

( NORTHSTAR CONTROLER COMMANDS )
: UNLISTEN ATN ON ZAPP 3F CHRSEND ATN OFF ZAPP ;
: UNTALK ATN ON NRFD OFF NDAC OFF ZAPP 5F CHRSEND ATN OFF ZAPP ;
: TALKER ATN ON ZAPP 40 | CHRSEND NDAC ON ZAPP NRFD ON ZAPP
        ATN OFF ZAPP ;
: LISTENER ATN ON ZAPP  20 | CHRSEND ATN OFF ZAPP ;
: SP-ENABLE ATN ON ZAPP 18 CHRSEND ATN OFF ZAPP ;
: SP-DISABLE UNTALK ATN ON ZAPP 19 CHRSEND ATN OFF ZAPP ;

( COMMANDS SPECIFIC TO QUANTEX UNIT )
0 VARIABLE H1 0 VARIABLE H2 0 VARIABLE H3
0 VARIABLE R1 0 VARIABLE R2
: STRIP 10 MOD ;
: UNPICK DUP STRIP H1 ! 10 / DUP STRIP H2 ! 10 / STRIP H3 ! ;
: DATA 7F UNPICK ;   : COMMAND 7E UNPICK ;
: RUNPICK DUP STRIP R1 ! 10 / STRIP R2 ! ;
: RIN A8 RUNPICK ;    : ROUT E8 RUNPICK ;
: SEND UNPICK H1 @ CHRSEND H2 @ CHRSEND H3 @ CHRSEND ;
: QSET EOI ON ZAPP 00 CHRSEND EOI OFF ZAPP
      00 CHRSEND 01 CHRSEND 30 R2 @ | CHRSEND 20 R1 @ | CHRSEND
      02 CHRSEND 20 H1 @ | CHRSEND 30 H2 @ | CHRSEND 40 H3 @ | CHRSEND
      0F CHRSEND ;
: COMMANDLOAD QUANTEX LISTENER RIN COMMAND QSET SEND ;
: COMMANDCHECK QUANTEX LISTENER ROUT COMMAND QSET UNLISTEN
               QUANTEX TALKER
               BEGIN CHRRCV 0F & CHRRCV 0F & + CHRRCV 0F & + 0= END
               NRFD OFF NDAC OFF UNTALK ;
: DATALOAD QUANTEX LISTENER DATA RIN QSET SEND ;
: DATARECEIVE QUANTEX LISTENER DATA ROUT QSET UNLISTEN
             QUANTEX TALKER CHRRCV 0F & CHRRCV 0F & 10 * +
             CHRRCV 0F & 100 * + NDAC OFF NRFD OFF ZAPP UNTALK ;

( QUANTEX FRONT PANEL INSTRUCTION SET )
: SUM-ALIGN DATALOAD 1 COMMANDLOAD COMMANDCHECK ;
: SUM-PRESET-DATA DATALOAD 2 COMMANDLOAD COMMANDCHECK ;
: AVG-PRESET-DATA DATALOAD 3 COMMANDLOAD COMMANDCHECK ;
: AVG-PARAM DATALOAD 4 COMMANDLOAD COMMANDCHECK ;
: TOTAL 5 COMMANDLOAD COMMANDCHECK DATARECEIVE ;
: OFFSET-DATA DATALOAD 6 COMMANDLOAD COMMANDCHECK ;
: GAIN-DATA DATALOAD 7 COMMANDLOAD COMMANDCHECK ;
: PAUSE-SET DATALOAD 8 COMMANDLOAD COMMANDCHECK ;
: PROG-STEP 9 COMMANDLOAD COMMANDCHECK DATARECEIVE ;
: POT-1 DATALOAD 10 COMMANDLOAD COMMANDCHECK ;
: POT-2 DATALOAD 11 COMMANDLOAD COMMANDCHECK ;
: POT-3 DATALOAD 12 COMMANDLOAD COMMANDCHECK ;
: POT-4 DATALOAD 13 COMMANDLOAD COMMANDCHECK ;
: SYS-TEST 15 COMMANDLOAD COMMANDCHECK ;
: SUM 15 COMMANDLOAD COMMANDCHECK ;
: SUM-PRESET 16 COMMANDLOAD COMMANDCHECK ;
```

APPENDIX A

```
: SUM-FULL 17 COMMANDLOAD COMMANDCHECK ;
: AVG 18 COMMANDLOAD COMMANDCHECK ;
: AVG-PRESET 19 COMMANDLOAD COMMANDCHECK ;
: STORE-INPUT 20 COMMANDLOAD COMMANDCHECK ;
: STORE-OUTPUT 21 COMMANDLOAD COMMANDCHECK ;
: INV 22 COMMANDLOAD COMMANDCHECK ;
: HOLD/CONTROL 23 COMMANDLOAD COMMANDCHECK ;
: INPUT 24 COMMANDLOAD COMMANDCHECK ;
: MEMORY 25 COMMANDLOAD COMMANDCHECK ;
: OFFSET 26 COMMANDLOAD COMMANDCHECK ;
: GAIN 27 COMMANDLOAD COMMANDCHECK ;
: C'TOUR 28 COMMANDLOAD COMMANDCHECK ;
: DIFF 29 COMMANDLOAD COMMANDCHECK ;
: NEG-DIFF 30 COMMANDLOAD COMMANDCHECK ;
: POS-DIFF 31 COMMANDLOAD COMMANDCHECK ;
: MAG-DIFF 32 COMMANDLOAD COMMANDCHECK ;
: EDGE-ENH 33 COMMANDLOAD COMMANDCHECK ;
: LEARN 34 COMMANDLOAD COMMANDCHECK ;
: ACTIVE 35 COMMANDLOAD COMMANDCHECK ;
: WAIT 36 COMMANDLOAD COMMANDCHECK ;
: PAUSE 37 COMMANDLOAD COMMANDCHECK ;
: RUN 38 COMMANDLOAD COMMANDCHECK ;
: SET-MEM-QUAD DATALOAD 39 COMMANDLOAD COMMANDCHECK ;
: SET-MEM-FIELD DATALOAD 40 COMMANDLOAD COMMANDCHECK ;
: CLEAR 41 COMMANDLOAD COMMANDCHECK ;
: NON-LINEAR 42 COMMANDLOAD COMMANDCHECK ;
: POLARITY-INVERT 43 COMMANDLOAD COMMANDCHECK ;
: INTEGRATE 44 COMMANDLOAD COMMANDCHECK ;
: FRAME-COUNT DATALOAD 45 COMMANDLOAD COMMANDCHECK ;
: CHECK-MODE 46 COMMANDLOAD COMMANDCHECK DATARECEIVE ;
: CHECk-ARITH-STATE 47 COMMANDLOAD COMMANDCHECK DATARECEIVE ;
: CHECK-TOGG-STATE 48 COMMANDLOAD COMMANDCHECK DATARECEIVE ;
: OUTPUT-TRANS-CONT DATALOAD 49 COMMANDLOAD COMMANDCHECK ;
: POWER-ON-INIT 50 COMMANDLOAD COMMANDCHECK ;

( VIDEO RAM INPUT )
: VIN 00 CHRSEND 01 CHRSEND 39 CHRSEND 22 CHRSEND 02 CHRSEND
      20 CHRSEND 30 CHRSEND 40 CHRSEND 50 CHRSEND 6E CHRSEND
      0F CHRSEND ;

( VIDEO OUT )
: VOUT 00 CHRSEND 01 CHRSEND 3D CHRSEND 22 CHRSEND 02 CHRSEND
       20 CHRSEND 30 CHRSEND 40 CHRSEND 50 CHRSEND 6E CHRSEND
       03 CHRSEND 20 CHRSEND 30 CHRSEND 41 CHRSEND 50 CHRSEND
       61 CHRSEND 0F CHRSEND ;

( EALING COMMANDS )
0 VARIABLE EALING
: VERTICAL 0B EALING ! ;
: HORIZONTAL 0C EALING ! ;
  0 VARIABLE HO 0 VARIABLE MO 0 VARIABLE LO
CODE NRFDCK 7D IN 40 AND 0 H LD A L LD HL PUSH $NEXT JP EDOC
: NRFDCHECK BEGIN NRFDCK 40 - 0= END ;
: TWOBITS DUP 100 MOD LO ! 100 / MO ! 0 HO ! ;
: LOCAL NRFDCHECK ATN ON ZAPP 3F CHRSEND 20 EALING @ | CHRSEND
```

```
        61 CHRSEND ATN OFF ZAPP 00 OUTDATA ;
: REMOTE NRFDCHECK ATN ON ZAPP 3F CHRSEND 20 EALING @ | CHRSEND
        62 CHRSEND ATN OFF ZAPP 00 OUTDATA ;
: DRIVE NRFDCHECK ATN ON ZAPP 3F CHRSEND 20  EALING @ | CHRSEND
        63 CHRSEND ATN OFF ZAPP 00 OUTDATA ;
: REVERSE NRFDCHECK ATN ON ZAPP 3F CHRSEND 20  EALING @ | CHRSEND
         64 CHRSEND ATN OFF ZAPP 00 OUTDATA ;
: FORWARD NRFDCHECK ATN ON ZAPP 3F CHRSEND 20 EALING @ | CHRSEND
         65 CHRSEND ATN OFF ZAPP 00 OUTDATA ;
: TRAVEL NRFDCHECK ATN ON ZAPP 3F CHRSEND 20 EALING @ | CHRSEND
         66 CHRSEND ATN OFF ZAPP TWOBITS LO @ CHRSEND MO @ CHRSEND
         EOI ON ZAPP 00 CHRSEND EOI OFF ZAPP 00 OUTDATA ;
: SPEED NRFDCHECK ATN ON ZAPP 3F CHRSEND 20 EALING @ | CHRSEND
        67 CHRSEND ATN OFF ZAPP TWOBITS
        MO @ CHRSEND EOI ON ZAPP LO @ CHRSEND EOI OFF ZAPP 00 OUTDATA ;
: TRAVELREAD NRFDCHECK ATN ON ZAPP 3F CHRSEND 40 EALING @ | CHRSEND
            66 CHRSEND ATN OFF ZAPP 00 OUTDATA ;
: COUNTREAD NRFDCHECK ATN ON ZAPP 3F CHRSEND 40 EALING @ | CHRSEND
           68 CHRSEND ATN OFF ZAPP 00 OUTDATA ;

( HP 9872A PLOTTER )
FALLOC FILE1
: ^ 2DUP DROP DUP ROT 1 - DUP 0= IF DROP DROP DROP
    ELSE 0 DO * SWAP DUP ROT LOOP ROT DROP SWAP DROP ENDIF ;
: STR DUP 0 < IF 2D CHRSEND -1 * ENDIF DUP A 4 ^ / 30 + CHRSEND A 4 ^
      MOD DUP A 3 ^ / 30 + CHRSEND A 3 ^ MOD DUP
      A 2 ^ / 30 + CHRSEND A 2 ^ MOD DUP
      A 1 ^ / 30 + CHRSEND A MOD 30 + CHRSEND ;

( PLOTTER INSTRUCTION SET )
: PLOT 50 CHRSEND 41 CHRSEND SWAP STR 2C CHRSEND STR 3B CHRSEND ;
: PLOT-REL 50 CHRSEND 52 CHRSEND SWAP STR 2C CHRSEND STR 3B CHRSEND ;
: PENUP 50 CHRSEND 55 CHRSEND 3B CHRSEND ;
: PENDOWN 50 CHRSEND 44 CHRSEND 3B CHRSEND ;
: DP 44 CHRSEND 50 CHRSEND 3B CHRSEND ;
: OD 4F CHRSEND 44 CHRSEND 3B CHRSEND ;
: OC 4F CHRSEND 43 CHRSEND 3B CHRSEND ;
: OS 4F CHRSEND 53 CHRSEND 3B CHRSEND ;
: DIGITIZE T" ENTER NAME OF FILE "
           FILE1 DUP NAMIT OPENW
           BEGIN
           UNLISTEN PLOTTER LISTENER DP SP-ENABLE UNLISTEN
           PLOTTER TALKER
           BEGIN 0B DUP CALLCPM 0 = IF CHRRCV 4 & 4 = ELSE
           30 FILE1 WBYTE 2C FILE1 WBYTE 30 FILE1 WBYTE 3B FILE1 WBYTE
           FILE1 FLUSH FILE1 CLOSE
           SP-DISABLE UNTALK PLOTTER LISTENER OD RESTART ENDIF END
           SP-DISABLE UNTALK PLOTTER LISTENER OD
           UNLISTEN PLOTTER TALKER
           BEGIN CHRRCV DUP FILE1 WBYTE 2C = END
           BEGIN CHRRCV DUP FILE1 WBYTE 2C = END UNTALK
           0 END ;
: PLOTD T" ENTER NAME OF FILE TO PLOT "
        FILE1 NAMIT
        FILE1 OPENR 50 CHRSEND 41 CHRSEND
```

```
        BEGIN FILE1 RBYTE DUP CHRSE )
        3B = IF 1 ELSE 0 ENDIF END
        FILE1 CLOSE ;
: SP 53 CHRSEND 50 CHRSEND CHRSEND 3B CHRSEND ;
: RED 31 SP ;
: GREEN 32 SP ;
: BLACK 33 SP ;
: BLUE 34 SP ;
: TEXT 4C CHRSEND 42 CHRSEND BEGIN GCH DUP CHRSEND D - 0= END 3 CHRSEND ;
: VIEWGRAPH INIT T" ENTER NAME OF VIEWGRAPH FILE: " FILE1 NAMIT
            UNLISTEN PLOTTER LISTENER PENUP BLACK
            0 1F40 PLOT FILE1 OPENR 4C CHRSEND 42 CHRSEND
            BEGIN FILE1 RBYTE DUP
            7C = IF 3 CHRSEND WORD FIND IF EXECUTE ELSE NUMBER IF
            LITERAL ELSE UNDEFINED ENDIF ENDIF
            4C CHRSEND 42 CHRSEND DROP 20
            ENDIF DUP 1A = IF 3 CHRSEND ELSE DUP CHRSEND
            ENDIF 1A - 0= END 30 SP 3000 2500 PLOT ININIT ;

( SERIAL POLL )
: EMSG-1 T" DRIVE ON " CR ;
: EMSG-2 T" REQUEST STOP " CR ;
: EMSG-3 T" REVERSE LIMIT REACHED " CR ;
: EMSG-4 T" FORWARD LIMIT REACHED " CR ;
: EMSG-5 T" REQUEST LOCAL CONTROL " CR ;
: VER-EAL-RESPONSE DUP 01 & 01 = IF T" VERTICAL " EMSG-1 ENDIF
                   DUP 02 & 02 = IF T" VERTICAL " EMSG-2 ENDIF
                   DUP 04 & 04 = IF T" VERTICAL " EMSG-3 ENDIF
                   DUP 08 & 08 = IF T" VERTICAL " EMSG-4 ENDIF
                       10 & 10 = IF T" VERTICAL " EMSG-5 ENDIF ;
: HOR-EAL-RESPONSE DUP 01 & 01 = IF T" HORIZONTAL " EMSG-1 ENDIF
                   DUP 02 & 02 = IF T" HORIZONTAL " EMSG-2 ENDIF
                   DUP 04 & 04 = IF T" HORIZONTAL " EMSG-3 ENDIF
                   DUP 08 & 08 = IF T" HORIZONTAL " EMSG-4 ENDIF
                       10 & 10 = IF T" HORIZONTAL " EMSG-5 ENDIF ;
: PLOTTER-RESPONSE T" PLOTTER " ;
: POLLRESPONSE SWAP DUP 05 = IF DROP PLOTTER-RESPONSE ENDIF
                    DUP 0B = IF DROP VER-EAL-RESPONSE ENDIF
                    DUP 0C = IF DROP HOR-EAL-RESPONSE ENDIF ;
: SRQCALL? DUP TALKER CHRRCV DUP 40 & 40 = UNTALK IF POLLRESPONSE
           ELSE DROP DROP ENDIF ;
: SERIALPOLL UNLISTEN UNTALK
             SP-ENABLE
             PLOTTER SRQCALL?
             VER-EAL SRQCALL?
             HOR-EAL SRQCALL?
             SP-DISABLE ;
: SRQ? LINES FF X| 04 & 04 = IF SERIALPOLL ENDIF ;

INIT
7 TCH
BASE !
[END-OF-FILE]
```

DISTRIBUTION LIST

	No. of Copies
Commander US Army Research Office ATTN: DRXRO-PH, Dr. R. Lontz P. O. Box 12211 Research Triangle Park, NC 27709	5
Headquarters, Department of the Army Office of the DCS for Research, Development & Acquisition ATTN: DAMA-ARZ Room 3A474, The Pentagon Washington, DC 20301	1
Director Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, VA 22209	1
Director US Army Night Vision Laboratory ATTN: John Johnson John Deline Peter VanAtta Fort Belvoir, VA 22060	1
Commander US Army Picatinny Arsenal Dover, NJ 07801	1
Commander US Army Harry Diamond Laboratories 2800 Powder Mill Road Adelphi, MD 20783	1
Commander US Army Foreign Science and Technology Center ATTN: W. S. Alcott Federal Office Building 220 7th Street, NE Charlottesville, VA 22901	1

Commander 1
US Army Training and Doctrine Command
Fort Monroe, VA 22351

Director
Ballistic Missile Defense Advanced Technology Center
ATTN: ATC-D 1
ATC-O 1
ATC-R 1
ATC-T 1
P. O. Box 1500
Huntsville, AL 35808

Environmental Research Institute of Michigan
Radar and Optics Division
ATTN: Dr. A. Kozma 1
Dr. C. C. Aleksoff 2
Juris Upatnieks 1
P. O. Box 8618
Ann Arbor, MI 41807

IIT Research Institute 1
ATTN: GAC_AC
10 West 35th Street
Chicago, IL 60616

Dr. J. G. Castle 1
9801 San Gabriel, NE
Albuquerque, NM 87111

Commander, Center for Naval Analyses 1
ATTN: Document Control
1401 Wilson Boulevard
Arlington, VA 22209

Dr. R. Brown 2
Department of Industrial Systems Engineering
University of Alabama
Huntsville, AL

Dr. Richard Berg 1
DMA/SPOEM
8301 Greensboro Drive
McLean, VA 22102

Dr. Nicholas George 2
The Institute of Optics
University of Rochester
Rochester, NY 14627

Dr. L. Forsely 1
Laboratory for Laser Enegetics
University of Rochester
Rochester, NY 14627

Professor Anil K. Jain 1
Department of Electrical Engineering
University of California, Davis
Davis, CA 95616

Terry Turpin 1
Department of Defense
9800 Savage Road
Fort George G. Meade, MD 20755

Dr. Stuart A. Collins 1
Electrical Engineering Department
Ohio State University
1320 Kennear Rod
Columbus, OH 43212

US Army Materiel Systems Analysis Activity 1
ATTN: DRXSY-MP
Aberdeen Proving Ground, MD 21005

US Army Night Vision Laboratorry 1
ATTN: DELNV-L, Dr. R. Buser
Fort Belvoir, VA 22060

Dr. F. T. S. Yu 1
Penn State University
Department of Electrical Engineering
University Park, PA 16802

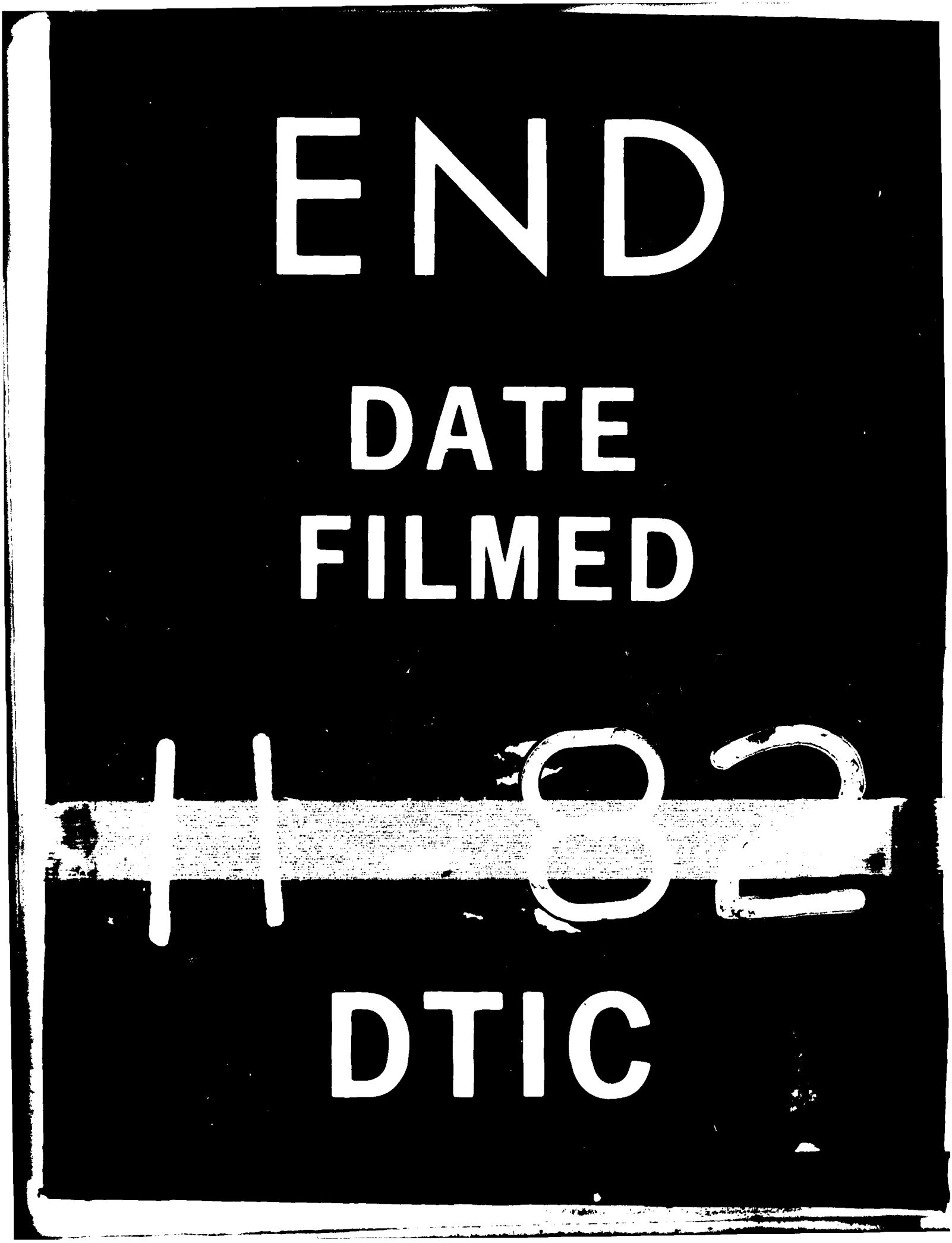
Don Ramsey 3
United Controls Corporation
P. O. Box 4620
Farley Station, AL 35802

K. B. Farr 5
210 Thomas Street
Tuscaloosa, AL 35401

Ed Runnion 2
J. M. Cockerham and Associates
4717 University Drive
Huntsville, AL

Dr. R. Kurtz TAI, Inc. 12010 South Parkway Huntsville, AL 35803	2
DRCPM-PE-E, John Pettitt	1
-PE	1
DRSMI-LP, Mr. Voigt	1
-O	1
-Y	1
-R	1
-RN, Jerry Hagood	1
-RE, W. Pittman	1
-RD	3
-RG	1
-RG, J. A. McLean	1
-RR, Dr. R. L. Hartman	1
Dr. J. S. Bennett	1
Dr. C. R. Christensen	1
Dr. J. G. Duthie	50
Mr. H. Dudel	1
-RPR	15
-RPT	1

☆U.S. GOVERNMENT PRINTING OFFICE: 1982—546-116/764


END
DATE
FILMED
DTIC