

MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES



3 4456 0376549 6

**ornl**

ORNL/TM-12459

**OAK RIDGE  
NATIONAL  
LABORATORY**

**MARTIN MARIETTA**

**A Remote Control Console  
for the HHIRF  
25-MV Tandem Accelerator**

A. M. Hasanul Basher

MANAGED BY  
MARTIN MARIETTA ENERGY SYSTEMS, INC.  
FOR THE UNITED STATES  
DEPARTMENT OF ENERGY

OAK RIDGE NATIONAL LABORATORY  
CENTRAL RESEARCH LIBRARY  
CIRCULATION SECTION  
450N ROOM 125

**LIBRARY LOAN COPY**

DO NOT TRANSFER TO ANOTHER PERSON

If you wish someone else to see this report, send in name with report and the library will arrange a loan.

306-2300 11 9 77

This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from the Office of Scientific and Technical Information, P. O. Box 62, Oak Ridge, TN 37831; prices available from (615)576-8401, FTS 626-8401.

Available to the public from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161.

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

50%  
28

Physics Division

**A REMOTE CONTROL CONSOLE  
FOR THE HHIRF 25-MV TANDEM ACCELERATOR**

A. M. Hasanul Basher

Manuscript Completed—August 14, 1992  
Date Published—September 1993

**NOTICE** This document contains information of a preliminary nature.  
It is subject to revision or correction and therefore does not represent a  
final report.

Prepared for the  
Office of Energy Research  
KB 02 02 00 0

Prepared by the  
Oak Ridge National Laboratory  
Oak Ridge, Tennessee 37831-6285  
managed by  
MARTIN MARIETTA ENERGY SYSTEMS, INC.  
for the  
U.S. DEPARTMENT OF ENERGY  
under contract DE-AC05-84OR21400



3 4456 0376549 6



A REMOTE CONTROL CONSOLE  
FOR THE HHIRF 25-MV TANDEM ACCELERATOR\*

A. M. Hasanul Basher<sup>1</sup>  
Physics Division  
Oak Ridge National Laboratory  
Oak Ridge, Tennessee 37831-6368

August 14, 1992

---

\* Operated by Martin Marietta Energy Systems, Inc. for the U.S. Department of Energy under contract No. DE-AC05-84OR21400 .

<sup>1</sup> Summer Faculty Research Participant, School of Engineering Technology, South Carolina State College, Orangeburg, South Carolina 29117.



## Table of Contents

	<u>Page</u>
<b>Acknowledgments</b>	
1. Introduction	1
2. Implementation Strategy	2
3. Work Completed	3
4. Future Action Plans	4
5. Data Transfer Protocol	5
6. File Structure and Program Execution	9
7. Update CRT Page Text	17
8. Flowcharts	21
 <b>Appendix</b>	
<b>Program Listing</b>	



## Acknowledgments

This program was sponsored by Oak Ridge Associated Universities (ORAU) at Oak Ridge, Tennessee, and conducted under the supervision of Sigmund W. Mosko in the Physics Division at Oak Ridge National Laboratory. I greatly appreciate ORAU for giving me an opportunity to work on this project. I wish to express my gratitude to Sigmund W. Mosko and Raymond C. Juras for their invaluable suggestions and tremendous help in carrying out this project. I also wish to express my gratitude to D. K. Olsen, Alan B. Tatum, Martha J. Meigs who helped me on many occasions by providing advice and discussing problems.

My thanks are due for helpful suggestions in preparing this report provided by Raymond C. Juras, Martha J. Meigs and B. Alan Tatum.

I greatly appreciate the help from several other members of the Physics Division who provided advice and helpful discussions.

I am also indebted to Ms. Jeanette McBride for her assistance in organizing and editing the report.



# A REMOTE CONTROL CONSOLE FOR HHIRF 25-MV TANDEM ACCELERATOR

## 1. INTRODUCTION

The CAMAC-based control system for the 25-MV Tandem Accelerator at HHIRF uses two Perkin-Elmer, 32-bit minicomputers: a message-switching computer and a supervisory computer. Two operator consoles are located on one of the six serial highways. Operator control is provided by means of a console CRT, trackball, assignable shaft encoders, and meters. The message-switching computer transmits and receives control information on the serial highways.

At present, the CRT pages with updated parameters can be displayed and parameters can be controlled only from the two existing consoles, one in the Tandem control room and the other in the ORIC control room. It has become necessary to expand the control capability to several other locations in the building. With the expansion of control and monitoring capability of accelerator parameters to other locations, the operators will be able to control and observe the result of the control action at the same time. This capability will be useful in the new Radioactive Ion Beam project of the division.

Since the new control console will be PC-based, the existing page format will be changed. The PC will be communicating with the Perkin-Elmer through RS-232 with the aid of a communication protocol. Hardware configuration has been established, a software program that reads the pages from the shared memory, and a communication protocol have been developed. The following sections present the implementation strategy, work completed, future action

plans, and the functional details of the communication protocol. The appendix contains the software programs developed in this phase.

## 2. IMPLEMENTATION STRATEGY

This project will be carried out in the following phases:

- Phase I. Develop software that reads the shared memory of the Perkin-Elmer computer and correctly display each CRT page in the desired format.
  
- Phase II. Develop a protocol that allows a PC to communicate with the Perkin-Elmer computer through RS-232. Develop software communication "layers" on the PC and the Perkin-Elmer computer to implement the protocol.
  
- Phase III. Develop software that provides an operator with the following desirable features:
  - a. Dedicated display of pages on PC with parameters updated continuously with proper colors indicating the status of those.
  
  - b. Display assignable knobs and meters on PC with their labels wherever appropriate.

- c. Options to control different parameters with mouse and monitor them from PC.

### 3. WORK COMPLETED

In the first phase of the project, a software is developed that reads the shared memory of the Perkin-Elmer computer and displays each CRT page in the desired format. The program reads the contents of the selected page from the shared memory, calculates the value (both actual and percent of the value) and displays on the screen. In this new format, the page title, page number, and the date of the last page revision that are on the top three lines of the existing CRT page are displayed on the leftmost 16 columns on the PC screen. The accelerator parameters that start on the fourth line of the existing page are displayed, starting from the first line and in the 64 rightmost columns in the new format.

In this phase, a protocol is developed that allows the PC and the Perkin-Elmer to communicate with each other by sharing information. To implement the communication, separate software is developed for the PC and the Perkin-Elmer. Software for the PC is written in BASIC and that of the Perkin-Elmer is in FORTRAN. The transfer of requested data is accomplished via RS-232. Since the speed and the reliable transfer of data are very crucial, the amount of data transfer is kept to a minimum. The PC software sends the user-requested page number. The Perkin-Elmer software reads the page number. Then it calculates all values, finds color codes for the status fields, checks if a function is assigned, then prepares a file with this information and sends the file through the serial port via RS-232. The PC software reads the file and uses the information to

update the parameter values, the status fields' colors and the assigned function color, and finally displays the page by calling the page text already saved in the PC's hard drive. The user can display a different page with updated parameters by entering a page number from the PC. In the first phase, active status parameters were indicated by printing an asterisk adjacent to the status field. Now this label is displayed in color by fetching the status information from the shared memory. The function assigned is also displayed in proper color. The parameter values and the status colors are updated continuously at a predetermined interval, while the page text remains static.

The FORTRAN program developed before is modified and is used to prepare the texts of the CRT pages from 0 through 99 in the new format, which is then transmitted to the PC with the help of KERMIT file transfer protocol. The user will only run a simple BASIC program. This program runs KERMIT and KERMIT then runs the FORTRAN program on the Perkin-Elmer. The FORTRAN program then prepares a file of page text in the new format. This file is then transmitted to the PC by KERMIT. All these can be done just by entering the name of the BASIC program. It will take a few minutes. Since the transfer of page text is not needed very often, the time required for the file transfer is not a major concern.

#### 4. FUTURE ACTION PLANS

A program is now developed that will allow the user to update the page text whenever that becomes necessary. A protocol is also developed that allows the PC and the Perkin-Elmer to communicate with each other by sharing information. To display a CRT page, the page text is read from the PC hard drive and the other information, such as the parameter values and status fields'

status colors, are received from the Perkin-Elmer with the aid of the protocol developed. In order to reduce the time required for data transfer and the chance of data being corrupted, the amount of data transfer is kept to a minimum.

Some additional monitoring and display of control parameters on the CRT pages, such as labels for assignable knobs and values for meters, are necessary. In the next phase, these will be completed. These will be displayed in the leftmost 16 columns of the PC screen below the line where the date is displayed. The user would be able to move the cursor to the appropriate label by means of a mouse and select it either for monitoring or controlling that parameter. If the label is selected for monitoring, the respective meter will display the value; but if it is selected for control, the function keys can be assigned to change the parameter at different rates. Then the parameter will be saved.

## 5. DATA TRANSFER PROTOCOL

A file or data is transferred from one computer to another (the PC and the Perkin-Elmer) by a pair of programs, one running on each computer. The programs carry out the data transfer process by sending messages to each other through their communication ports. This protocol is character-oriented, i.e., data is transmitted in the form of discrete characters. The data transfer is synchronized because the data sender waits for a response from the receiver after each character is sent. The sender sends a ready signal before the transmission starts to tell the receiver that data is ready to send. The receiver then sends back an acknowledge signal, indicating it is ready to receive data.

The sender then sends data and waits for an acknowledge signal from the receiver before it sends the next data. Then the receiver sends the acknowledge signal after each character being received. This software handshake is necessary to avoid data overrun error during transmission. Since the speed of file transfer is one of the major concerns, the frequency of handshake is kept to a minimum. Since the Perkin-Elmer is slow in transmitting or receiving data to and/or from the outside devices, few delay loops are incorporated in the PC software to synchronize the timing between the two computers. The amount of delay is minimized by trial-and-error method.

The handshake process works like this: The transmitting computer sends a ready signal telling the receiving computer that it is ready to send data. The receiving computer sends an acknowledge signal. Then it waits for a while and checks the serial port to see if data is ready to read. After data is read, the receiving computer sends an acknowledge signal, advising the other computer to send more data. On the other hand, the transmitting computer waits for an acknowledge signal from the receiver before it starts sending data. Once the signal is received, the transmitting computer sends the first byte of data and sends the remaining bytes, one at a time, only after it receives the acknowledge signal from the receiver. These two operations are shown below:

**Receiving Computer:**

Wait for data ready signal

**Receive:**

Wait

Check port for data ready

Receive byte

Send acknowledge signal

Go to receive while more bytes to read

**Transmitting Computer:**

Send data ready signal

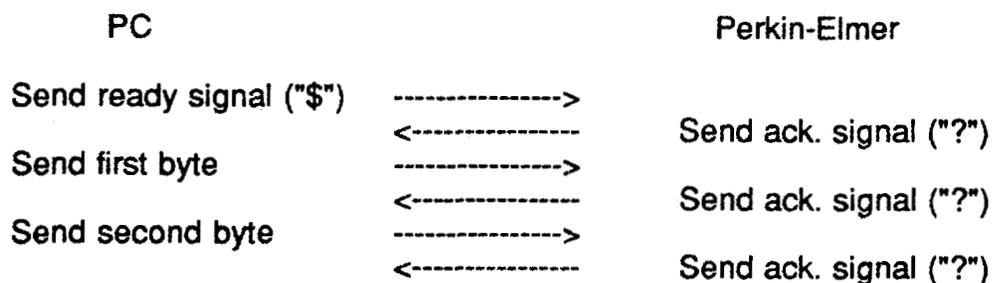
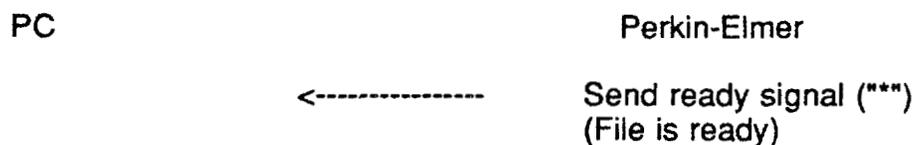
**Send:**

Send a byte

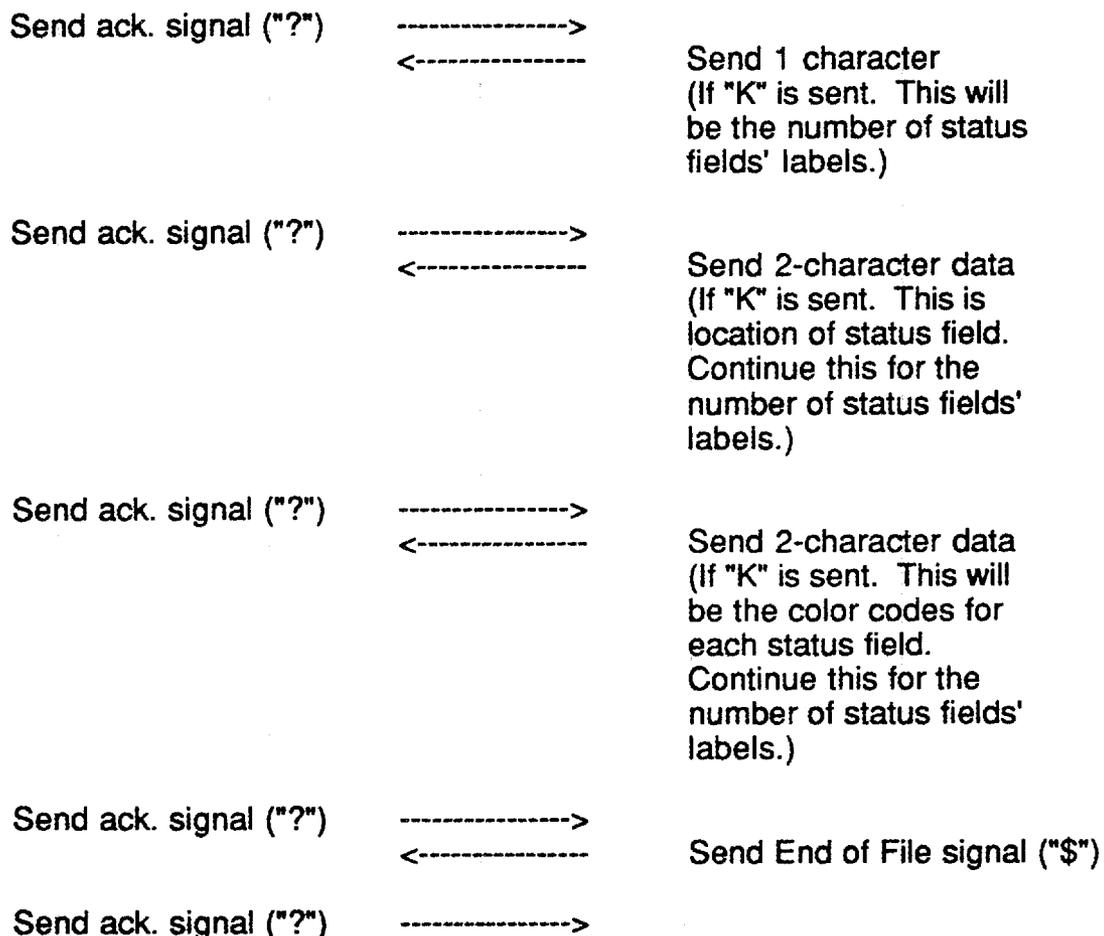
Wait for acknowledge signal

Go to send while more bytes to send

The following figures show how information transfer proceeds.

**a. Send Page Number (PC to Perkin-Elmer):****b. Parameter File Transfer (Perkin-Elmer to PC):**

Send ack. signal ("?" )	-----> <-----	Send 2-digit character (One at a time. This is no. of lines in the file.)
Send ack. signal ("?" )	-----> <-----	Send 2-digit character (One at a time. This is no. of characters in the second line of the file.)
Send ack. signal ("?" )	----->	
	.	
	.	
	continue until the first line is transmitted	
	<-----	Send 2-digit character (CRT line number)
Send ack. signal ("?" )	-----> <-----	Send character "R" (If RESULT is present in the line)
Send ack. signal ("?" )	-----> <-----	Send character "P" (If PERCENT is present)
Send ack. signal ("?" )	-----> <-----	Send character "S" (If SAVWRD is present)
Send ack. signal ("?" )	-----> <-----	Send character "K" (If status field is present)
Send ack. signal ("?" )	-----> <-----	Send Function Assign Value (0 or 1)
Send ack. signal ("?" )	-----> <-----	Send 9-character RESULT (If "R" is sent)
Send ack. signal ("?" )	-----> <-----	Send 9-character PERCENT (If "P" is sent)
Send ack. signal ("?" )	-----> <-----	Send 9-character SAVWRD (If "S" is sent)



This will conclude the transfer of data. The characters such as R, P, S, and K and their respective values will be transmitted only if these are present for a CRT line. If a CRT page does not have any parameter line, no such file needs to be transmitted and in that case, the Perkin-Elmer will send "#" to the PC, indicating no file is created. The PC will then act accordingly.

## 6. FILE STRUCTURE AND PROGRAM EXECUTION

The user enters the CRT page number from the PC. Once the PC receives the page number, it sends it to the Perkin-Elmer through the serial port. The Perkin-Elmer then calculates parameter values and/or finds the status fields' labels' color

codes and function assign status and then prepares a file to transfer to the PC. During this interval, the PC waits for a file ready signal from the Perkin-Elmer. The PC then receives the file with the aid of protocol via RS-232. The paragraphs below explain the structure of this file.

The first line of the file contains information such as the total number of lines in the file, number of characters in each of the following lines. A line is created for a CRT line if it has any of the parameters such as RESULT, PERCENT VALUE, SAVWRD and/or status field(s). The second to the last line of the file contains information about the CRT lines with parameters. The first two places of each of these lines in the file contain the actual CRT line number. The next few places are used for characters like R, P, S, or K, whichever is appropriate for that CRT line. If any of these is not appropriate for a CRT line, the corresponding character and its value is not printed in the file. The character printed next is the function assign status. Next to this are printed the actual values of R, P, and S (9 characters each), if present; then the number of status labels for K (1 character), if present. If K is printed, the next few characters printed in the file are locations of the status labels (2 characters each for K number of labels) and the color codes for these labels (2 characters each for K number of labels)

The number of characters in each of the lines of the file of a CRT page with parameters can be calculated as follows:

Descriptions	Number of Characters
CRT line number	2
Character "R"	1 (if present)
Character "P"	1 (if present)

Character "S"	1 (if present)
Character "K"	1 (if status field present)
Function Assign Status	2 (0 or 1)
RESULT VALUE	9 (if present)
PERCENT VALUE	9 (if present)
SAVWRD	9 (if present)
Number of Status Labels	1 (if status field present)
Locations of Status Labels	2* (Number of Status Labels) (if status field present)
Color Codes	2* (Number of Status Labels) (if status field present)

If all of the above are present in a line of a CRT page, the total number of characters in a line of the file after the first line is calculated as:

$$\text{Total Character} = 36 + 4* (\text{Number of Status Labels})$$

If N is the number of lines in a CRT page with parameters, the first line of this file will have 2\* (N+1) number of characters.

The PC reads the complete file, one character at a time, and prepares a similar file that contains all the lines except the first line. The information from the first line is used to arrange the file in the original format. It provides the PC information such as the total number of lines and the number of characters in each individual line.

The PC software then reads the text of the pages that are already saved in the PC and reads this new file, one line at a time. Once a match is found in the CRT line

number in the two files, that line is updated with new parameters. Then, finally the page is displayed.

It is important to note that if a CRT page does not have any parameter, no file is created by the Perkin-Elmer. In that case, the Perkin-Elmer sends a special signal and the PC does not try to read the serial port; rather, it reads the page text and displays the page.

Figures 1 and 2 on the next page show the CRT page number 10 and the corresponding parameter file created by the Perkin-Elmer computer. This file is being used by the PC to update the parameters, status fields' colors and function assign status, which are discussed here. The first two characters of the file indicate the number of lines of the file and the pairs of characters that follow are the number of characters in each of the following lines. The second and the rest of the lines have similar structures. The first two characters carry the CRT page line number that contains any parameter. The above file indicates the first line of the CRT page 10 that has any parameter is 12. The other lines of the page containing parameters are 13, 14, 18, and 19. Next to the line numbers are characters such as R, P, S, and/or K. If R is present in a line, that means this line has RESULTS, and so on. P indicates the presence of PERCENT VALUE, S stands for SAVWRD, and K for status fields. The character that comes after this (0 or 1) indicates if the function is assigned (1) or not (0). Next to this character are the actual values of R, P, S, and K in the same order as their indicators are printed in the file. The values of RESULT, PERCENT, and SAVWRD each occupies 9 character positions, whereas K occupies one (1) character position, since the value of K is the number of labels present (maximum of 4). The line number 18 of this page has all

PAGE 10	2	***EXTREME EMERGENCY***				
	4	PUSH EMERGENCY STOP BUTTON				
EMERGENCY ACCE-	5	EVACUATE!				
LERATOR SHUTDO-	8	*NORMAL EMERGENCY*				
WN	10	PUSH FC 13-1 IN BUTTON				
	11	PUSH CHAINS OFF BUTTON				
	12	PBP MR-1	0.0 CM	IN/OUT	LIM^CHG	T
8/23/91	13	BLV 17-3		OPN/CLO		CLOSE
	14	BLV 19-2		OPN^CLO	ALM	CLOSE
	16	EVALUATE SITUATION; PERFORM APPROPRIATE FUNCTIONS				
	18	HVS I1-1	0.00 KV	ON/OFF	OVL NLK	T 0.00% 54.15
	19	GSV T -1	2.94 TRN	INC^DEC		C
		BYPASS CAMAC CRATES D-1 THROUGH T-1				
	21	PUSH SHAFTS OFF BUTTON				
	22	EVACUATE!				

Figure 1

## Parameter File

```

52814185224
12RK 0      0.03 1 3 5 4 6 4
13K 02 1 3 4 C
14K 03 1 3 5 4 442
18RPSK 0    0.00 0.00% 54.15 4 1 3 5 7 4 6 242
19RK 0      3.032 1 3 4 4

```

Figure 2

parameters and all four status fields' labels. After these, the line has actual status fields' label numbers and next are the color codes of these labels.

How the color codes are found and fetched from the shared memory, how the status fields' labels are determined, and their relations to the actual colors on the text are explained next.

The following global variables are needed:

POBUF (1,CRTLINE), POBUF (3,CRTLINE), BITBUF (1,CRTLINE),  
BITBUF (3,CRTLINE), BITBUF (5,CRTLINE), BITBUF (7,CRTLINE),  
TYPHAV (1,POINTER), BLCO2 (2,POINTER), DFBUF (1,CRTLINE),  
DFBUF (2,CRTLINE), DFBUF (3,CRTLINE), AND DFBUF (4,CRTLINE).

A CRT page can have, at most, two status fields with two labels in each. The labels are designated as 1, 3, 5, and 7, where labels 1 and 3 belong to status field 1, and 5 and 7 to status field 2. Each label has one BITBUF that is used to find the status for the label. Four DFBUFs are there for the four labels. DFBUFs carry the color code for each label.

POBUF (1,CRTLINE) is for Status Field 1.

POBUF (3,CRTLINE) is for Status Field 2.

BITBUF (1, CRTLINE) is for Label 1.

BITBUF (3,CRTLINE) is for Label 3.

BITBUF (5,CRTLINE) is for Label 5.

BITBUF (7,CRTLINE) is for Label 7.

DFBUF (1,CRTLINE) is for Label 1.

DFBUF (2,CRTLINE) is for Label 3.

DFBUF (3,CRTLINE) is for Label 5.

DFBUF (4,CRTLINE) is for Label 7.

The following calculations are done for the line number 18 of the CRT page 10:

Status Field 1, Label 1:

POBUF (1, 18) = 580H = 1408 (dec) [From shared memory]

POINTER = 1408/8 + 1 = 177 [Calculated]

TYPNAV (1, 177) = C11000000H < 0 [From shared memory]

BITBUF (1, 18) = 1BH = 27 (dec) [From shared memory]

BLC02 (2, 18) = 41AH [From shared memory]

= 0000 0000 0000 0000 0000 0100 0001 1010 (binary)

Status = Bit 27 of BLC02 = 1

DFBUF (1, 18) = 0406H

Color Code = AND operation of DFBUF and 00FFH

= 06

The first byte of this code is for background color and the second byte is for the foreground color. The actual color attribute is determined as

Color Attribute = Color Code (Background Color)

Color Attribute = Color Code / 2 (Foreground Color)

Background Color Attribute = 0

Foreground Color Attribute = 3

Status Field 1, Label 3:

The POBUF, POINTER, TYPNAV and BLC02 are same as above.

BITBUF (3, 18) = FFEC < 0 [From shared memory]

– BITBUF (3, 18) = 0014H = 20 (dec) [Calculated]

Status = Bit 20 of BLC02 = 0 [Calculated]

DFBUF (2, 18) = 0406H [From shared memory]

Color Code = AND operation of DFBUF and FF00H

= 04 [Calculated]

Background Color Attribute = 0

Foreground Color Attribute = 2

The following table will demonstrate the actual color of the text.

Attribute (Decimal)	Foreground or Background Color	Foreground Color with Intensity Bit Set
0	Black	Gray
1	Blue	Bright Blue
2	Green	Bright Green
3	Cyan	Bright Cyan
4	Red	Bright Red
5	Magenta	Bright Magenta
6	Brown	Yellow
7	White	Bright White

With the help of this table, the actual colors of the above two labels can be found.

The color of Label 1 of line 18 is cyan on black and that of Label 3 is green on

black. In the existing CRT pages, the text color is green, so when the label color shows green, it is changed to the text color.

## 7. UPDATE CRT PAGE TEXT

The CRT page text for pages 0 through 99 can be updated with the aid of KERMIT file transfer protocol. The FORTRAN program (NEWPAGE.FTN) on the Perkin-Elmer computer will generate a file (PCPAGE.TXT) with page text which will be transferred to the PC with the aid of KERMIT with the same filename. The complete process will be done by running a small BASIC program on the PC. This program has filename UPDATE.BAS. The program destroys the old PCPAGE.TXT file and runs KERMIT. KERMIT runs NEWPAGE that creates updated page text. The PCPAGE.TXT file is then transmitted to the PC. Next, a BASIC program (CONVERT.BAS) is run again by KERMIT on the PC. The CONVERT program rearranges the file to make ready for display.

For successful operation of this file transfer process, the following programs are required:

On the PC:

1. UPDATE.EXE
2. CONVERT.EXE
3. KERMIT.EXE
4. MSKERMIT.INI
5. PAGETEXT

On the Perkin-Elmer:

6. NEWPAGE.EXE

File Created:

PCPAGE.TXT

The listing of programs required for this file transfer are given here:

1. UPDATE.BAS

```
CLS
KILL "PCPAGE.TXT"
RUN "KERMIT"
END
```

2. CONVERT.BAS

```
DEFINT A-Z
OPEN "PCPAGE.TXT" FOR INPUT AS #1
OPEN "PCPAGE.TMP" FOR OUTPUT AS #2
PAGE = 0
DO UNTIL EOF(1)
  LINE INPUT #1, X$
  LENGTH = LEN(X$)
  IF (LENGTH < 80 ) THEN
    FOR K = LENGTH + 1 TO 80
      X$ = X$ + CHR$(32)
    NEXT K
    PRINT #2, X$
  ELSE
    PRINT #2, X$
  END IF
LOOP
CLOSE
OPEN "PCPAGE.TXT" FOR OUTPUT AS #1
OPEN "PCPAGE.TMP" FOR INPUT AS #2
DO UNTIL EOF(2)
  LINE INPUT #2, Y$
  PRINT #1, Y$
LOOP
CLOSE
KILL "PCPAGE.TMP"
END
```

## 3. MSKERMIT.INI

```
SET PORT 1
SET SPEED 9600
SET PARITY EVEN
TAKE PAGETEXT
```

## 4. PAGETEXT

```
CLEAR
SET INPUT ECHO OFF
SET DISPLAY QUIET
OUTPUT \13
INPUT 10 *
OUTPUT NEWPAGE \13
INPUT 60 *
OUTPUT KERMIT\13
INPUT
OUTPUT SEND PCPAGE.TXT \13
INPUT 5
RECEIVE
OUTPUT EXIT \13
INPUT 10 *
RUN CONVERT
OUTPUT \88
OUTPUT \13
EXIT
CONNECT
```

Command needed to update the page text and transfer to the PC is:

At the Prompt Type: UPDATE

Sign-on to Perkin-Elmer prior to issuing this command is required. For automatic sign-on, the following additional commands are needed in the KERMIT script file PAGETEXT. After the command "SET DISPLAY QUIET" it is needed to add:

```
OUTPUT / 13
OUTPUT   user code
INPUT 10 *
```

## 8. FLOWCHARTS

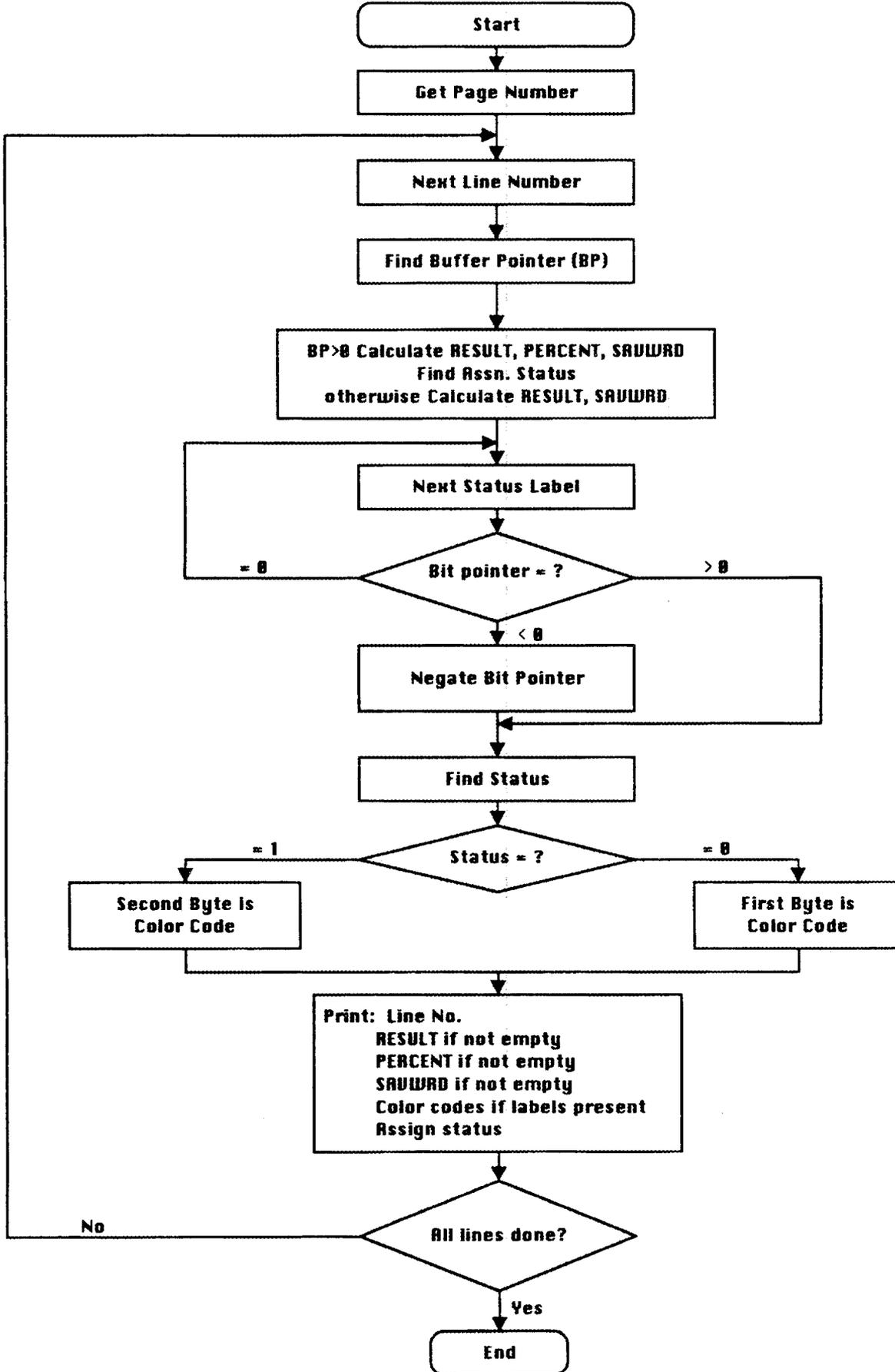
The next few pages display the following flowcharts:

1. Prepare Parameter File (Perkin-Elmer).
2. Page Display on PC.
3. Communication Protocol (Perkin-Elmer).
4. Communication Protocol (PC, Send Page Number).
5. Communication Protocol (PC, Read Parameter File).

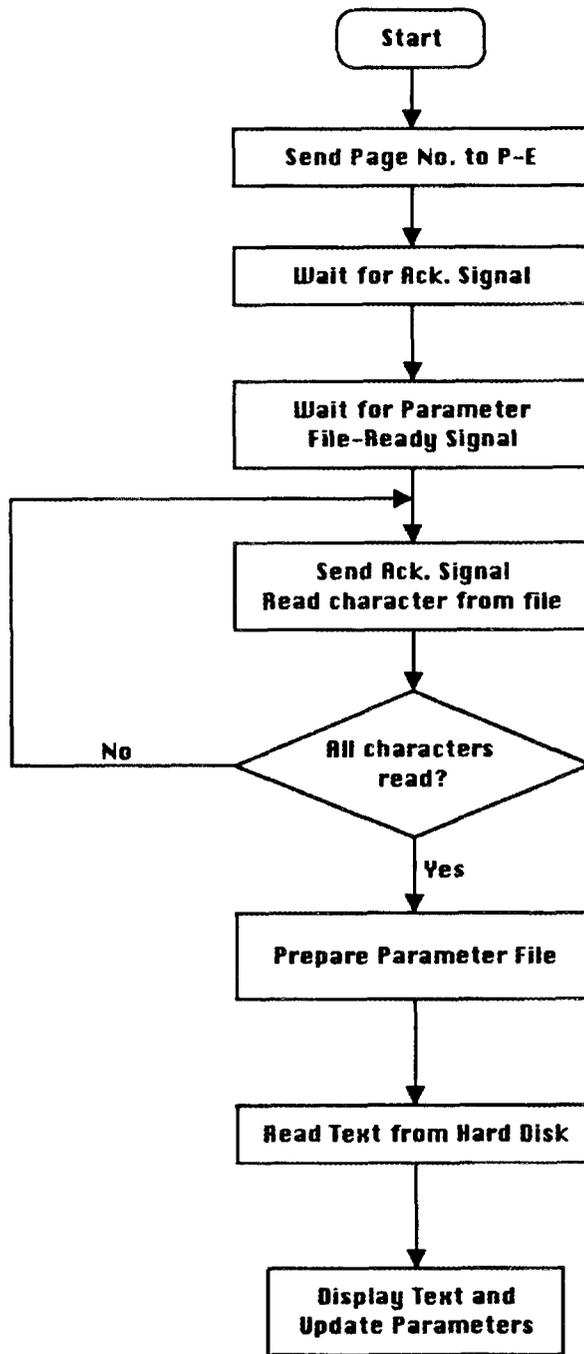
The page number is sent by the PC to the Perkin-Elmer. After reading the page number, the Perkin-Elmer prepares a parameter file which is sent to the PC. The PC reads the parameter file. Then the PC reads the text of the page from its

hard drive, displays the text, and uses the parameter file to update the parameters on the page.

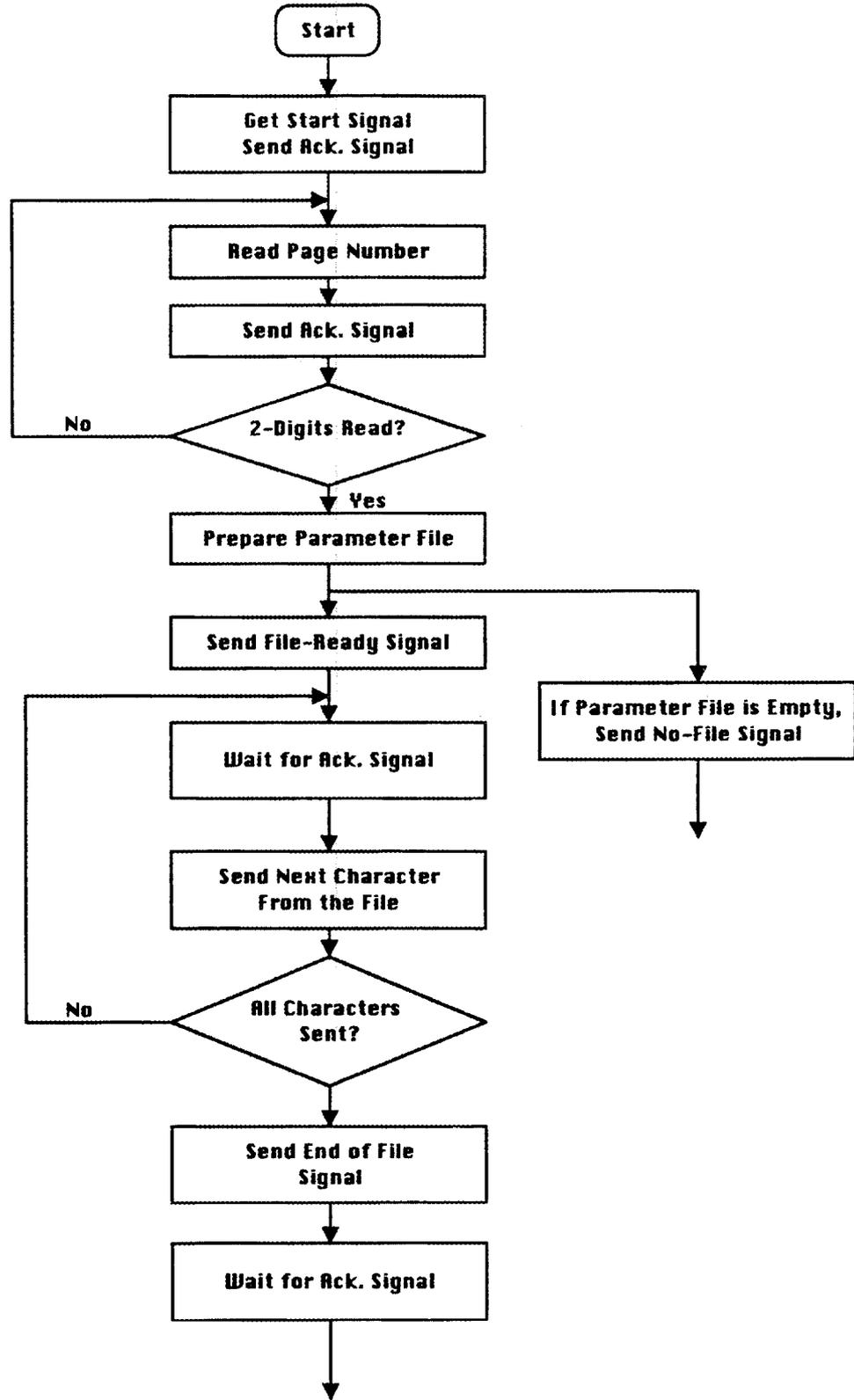
**PREPARE PARAMETER FILE  
PERKIN-ELMER**



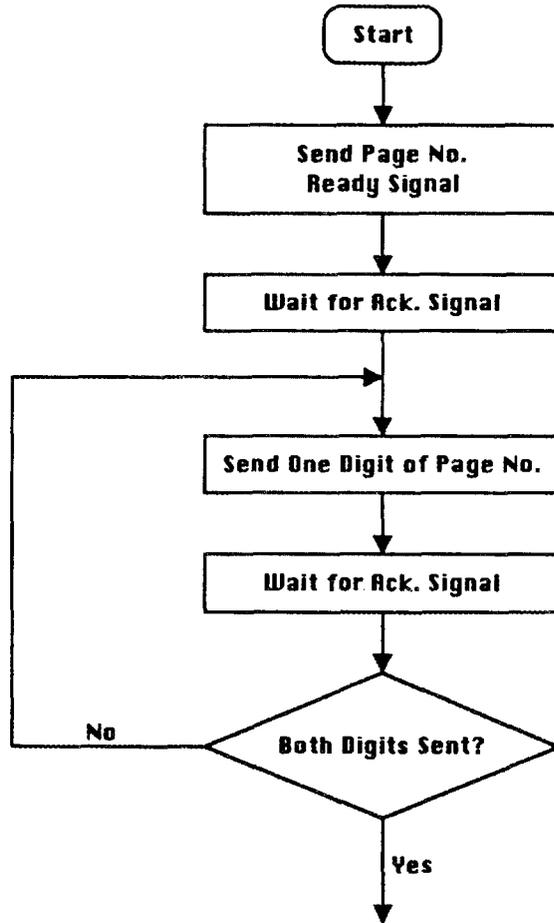
**PAGE DISPLAY  
ON PC**



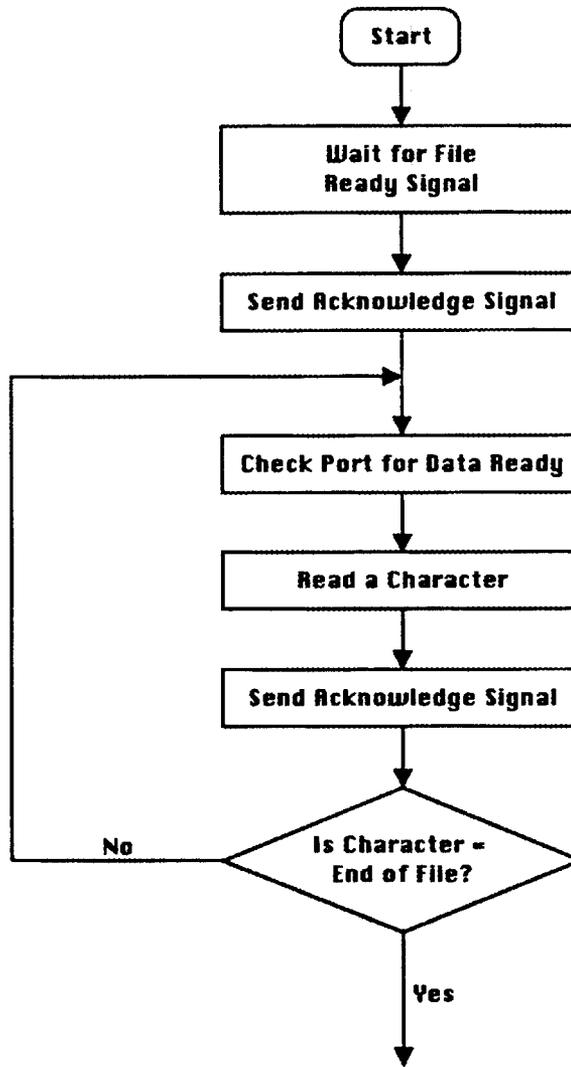
**COMMUNICATION PROTOCOL  
PERKIN-ELMER**



COMMUNICATION PROTOCOL  
PC  
SEND PAGE NUMBER



**COMMUNICATION PROTOCOL**  
**PC**  
**READ PARAMETER FILE**



## APPENDIX

The following pages contain the program listings that are developed in this phase of the project. Two different programs that run on the two computers (PC and Perkin-Elmer) allow them to communicate with each other by exchanging data from one to the other. The program on the PC is written in BASIC and the one that runs on the Perkin-Elmer is written in FORTRAN.

```

C *****
C *
C *      Filename : PAGPRM.FTN
C *      Developed by:   A. M. Hasanul Basher
C *      Date last revised: August 14, 1992
C *
C *      This program prepares a parameter file for a CRT page by
C *      sharing information with another computer through serial port.
C *      Another program must be running concurrently on the other
C *      computer. This program reads a 2-digit page number, then it
C *      reads the shared memory and prepares the parameter file which
C *      is transmitted to the other computer.
C *
C *      To start the program: Type PAGPRM and Hit Enter
C *
C *      Subroutine Used:      NUMCON
C *                          SYSIO
C *      Object Files Used:   BCDBIN.OBJ
C *                          RCJLIB.OBJ
C *                          F7RTL.OBJ
C *
C *      GLOBAL VARIABLES USED: BLC02,MBLIST,TYPSAV
C *
C *-----*
C *
C *      ** THE LINK FILE **
C *
C *      ** FILENAME PAGPRM.LNK
C *      ESTABLISH TASK
C *      OP WORK =XA00,FLOAT,DFLOAT,SYSSPACE=X4000
C *      RESOLVE GBLCOM.SEG/S,ACCESS=R,STRUC=(GBLCOM./X79FC)
C *      INCLUDE PAGPRM.OBJ
C *      INCLUDE BCDBIN.OBJ/S
C *      LIBRARY RCJLIB.OBJ/S,F7RTL.OBJ/S
C *      MAP NULL:
C *      BUILD PAGPRM.TSK
C *      END
C *      /*
C *
C *      ** THE CSS FILE **
C *
C *      ** FILENAME: PAGPRM.CSS
C *      L PAGPRM
C *      AS 2,PAGES.BUF/S,SRO
C *      AS 1,VT5:
C *      XAL PEFIL.DAT,IN,80
C *      XAL TMP.DAT,IN,80
C *      AS 9,PEFILE.DAT
C *      AS 8,STATUS.DAT
C *      ST
C *      $EXIT
C *      /*
C *
C *-----*
C *
C *      DEFINITION OF VARIABLES:
C *
C *      CRTROW      : Actual line of CRT
C *      PRMROW      : Accelerator parameter line that starts at 4th
C *                   row of CRT

```

```

C * LINENO      : Counts line number *
C * BFPTR      : Buffer pointer obtained from POBUF(5,PRMROW) *
C * BUFPTR     : Same as BFPTR *
C * WDPTR      : Word pointer obtained from buffer pointer. This *
C *             is used to locate the appropriate words required *
C *             to calculate parameter values. *
C * PRCPTR     : Pointer for words used to calculate the percent *
C *             of parameter value. *
C * RESULT     : Parameter value calculated from words located *
C *             at word pointer. *
C * SAVWRD     : Percent of parameter value calculated from word *
C *             saved at location percent pointer. *
C * STPTR      : Status pointer for status word. *
C * STWRD      : Status word that contains the status bit. The *
C *             word is located by status pointer. *
C * BITPTR     : Bit pointer that carries the position of status *
C *             bit in status word. *
C * STAT       : Value of status bit in status word. *

```

```

C *-----*

```

```

C * ** SUBROUTINE NUMCON ** *
C * * *
C * The subroutine NUMCON does the numerical conversions. It needs *
C * arguments such as BUFPTR, DFBUF(5,PRMROW), RESULT, *
C * POBUF(6,PRMROW), and PERCENT and returns the value of RESULT *
C * and PERCENT. From buffer pointer the routine determines the *
C * word pointer and the type of data. With the word pointer it *
C * also locates BLC02 and MBLIST and calculate the RESULT and *
C * PERCENT dependent on the type of data. The PERCENT value *
C * indicates the percent of control setting. If data is inactive *
C * RESULT is returned as INACTIVE. The routine uses different *
C * mathematical formula for different types. The data types are *
C * 0, 1, 2, 3, 4, 6, 7, 8, 9, 10, 18, and 19. It also calculates *
C * values (RESULT) for negative pointer ( pointer = -8 or -9). *
C * The PERCENT is calculated only if data type is 0 and *
C * POBUF(6,PRMROW) is +ve. *

```

```

C *****

```

```

C
COMMON /GBLCOM/ BLC02, MBLIST, TYPNAV
COMMON /CRTPAGE/ABUF,POBUF,BITBUF,DFBUF
INTEGER*4 ABUF(16,32), LUPAGE, PBLK(5), STATUS, RANADD
INTEGER*4 BLC02(2,1024), TYPNAV(2,1024), TYPE, SVDAT,STWRD
INTEGER*2 BFPTR,BUFPTR, WDPTR, PRCPTR, EXP,PRMROW,CRTROW
INTEGER*2 POBUF(6,24),BITBUF(8,24),DFBUF(5,24)
INTEGER*2 STPTR,LOCATE,LBYTE,BITPTR,PAGNO
INTEGER*2 COLOR(10),BITPTR1,LL(10),PBUF(40,24)
REAL MBLIST(2,1024), EXPF,SVDATF,TMPRES
LOGICAL STAT
INTEGER*2 LINENO(10),LCOUNT,TOTC(27),NTOTC(27)
INTEGER*2 A(10),C,D(2),L1,L2,K,NC,PLACES,START,L,F(4),G
INTEGER*2 ASSN,POINTER
INTEGER*4 PBLK1(5),PBLK2(5),PBLK3(5)
CHARACTER*9 RESULT, PERCENT, SAVWRD
CHARACTER*10 SAVFMT
CHARACTER*2 PAGE,ANS,B,E
CHARACTER*1 RES,PER,SAV,COLR,AST,NUMB,READY
DATA LINENO/48,49,50,51,52,53,54,55,56,57/

```

```

C

```

```

OPEN(3,FILE='CON:')           ; input from console
OPEN(5,FILE='CON:')           ; output to console

```

```

*****

```

```

      WAIT FOR READY SIGNAL ('$') FROM THE OTHER COMPUTER. IF
      THIS SIGNAL IS RECEIVED THAT MEANS THE PAGE NUMBER IS
      READY TO SEND. THEN READS THE PAGE NUMBER ONE DIGIT AT A
      TIME AND SENDS ACK. SIGNAL ('?') AFETR EACH DIGIT RECVD.

```

```

*****

```

```

999  READY = '?'
950  CALL SYSIO(PBLK,Y'49',1,F,1,0)
      CALL ILBYTE(G,F(1),0)
      IF (G.EQ.Y'0024') GO TO 951
      GO TO 950
951  CALL SYSIO(PBLK,Y'29',1,READY,1,0)
      DO 10 I = 1,2
      CALL SYSIO(PBLK,Y'49',1,A,1,0)
      CALL ILBYTE(D(I),A(1),0)
      CALL SYSIO(PBLK,Y'29',1,READY,1,0)
10   CONTINUE
      B = CHAR(D(1))
      E = CHAR(D(2))
      L2 = CTOI(E,K)
      L1 = CTOI(B,K)
      PAGNO = 10*L1 + L2

```

```

*****

```

```

SYSIO SUBROUTINE PERFORMS I/O FUNCTIONS. IOERR ROUTINE REPORTS
SYSIO ERRORS TO THE CONSOLE.

```

```

LUPAGE - INTEGER*4 ARGUMENT THAT SPECIFIES THE LOGICAL UNIT
        ON WHICH TO PERFORM THE I/O.

```

```

ABUF   - STARTING ADDRESS OF THE BUFFER USED IN THE I/O
        TRANSFER

```

```

RANADD - AN INTEGER*4 ARGUMENT SPECIFYING THE LOGICAL RECORD
        NUMBER TO BE ACCESSED ON DATA TRANSFER REQUEST

```

```

*****

```

```

RANADD = PAGNO*12
LUPAGE = 2
CALL SYSIO (PBLK,Y'4D',LUPAGE,ABUF,3328,RANADD)
CALL IOERR(PBLK,STATUS)

```

```

*****

```

```

      ** PARAMETER CALCULATIONS **

```

```

THIS PART OF THE PROGRAM SKIPS FIRST 3 CRT LINES. AFTER 3
LINES, IF IF BYTE 1 OF ABUF(1,L) IS NOT A DIGIT OR IF FIRST
TWO BYTES OF ABUF(1,L) ARE BLANKS NO NUMERICAL CONVERSION IS
REQUIRED. OTHERWISE NUMCON SUBROUTINE IS CALLED.

```

```

*****

```

```

LCOUNT = 0

```

```

DO 940 KM = 1,27
TOTC(KM) = 0
NTOTC(KM) = 0
940 CONTINUE
OPEN(9,FILE='PEFILE.DAT',STATUS='RENEW')
OPEN(8,FILE='TMP.DAT',STATUS='RENEW')
DO 100 CRTROW=1,27
ENCODE(RESULT,'(9H          )')
ENCODE(PERCENT,'(9H          )')
ENCODE(SAVWRD,'(9H          )')
PRMROW = CRTROW - 3 ; find param. line no
DO 303 I = 1,16
IF (ABUF(I,CRTROW).NE.' ') GO TO 304
303 CONTINUE
GO TO 100
304 IF (CRTROW.LT.4) GO TO 100 ; line<4, do nothing
IF(IAND(ABUF(1,CRTROW),Y'FFFF0000').EQ.Y'20200000') GO TO 100
CALL ILBYTE(LB,ABUF(1,CRTROW),1) ; get byte 1
DO 301 I = 1,10
IF (LB.EQ.LINENO(I)) GO TO 302 ; byte 1 a digit?
301 CONTINUE
GO TO 100
302 BFPTR = POBUF(5,PRMROW) ; buffer pointer
BUFPTR = POBUF(5,PRMROW)
PRCPTR = POBUF(6,PRMROW)
C
C *****
C
C NUMCON SUBROUTINE IS CALLED FOR NUMERICAL CONVERSIONS
C
C *****
C
CALL NUMCON (BUFPTR,DFBUF(5,PRMROW),RESULT,POBUF(6,PRMROW),
1 PERCENT)
C
C DETERMINE DATA TYPE TO CALCULATE PERCENT VALUE
C
IF(BFPTR) 119,119,103
103 WDPTR = BFPTR/8+1 ; find word pointer
IF(BTEST(TYPSAV(1,WDPTR),0)) GO TO 112 ; data active ?
GO TO 119
112 TYPE = IAND(ISHFT(TYPSAV(1,WDPTR),-16),Y'FF') ; data type
IF(TYPE.NE.0) GO TO 119
C
C CALCULATE PERCENT VALUE FROM SAVED WORD
C
ENCODE(SAVWRD,'(9H          )')
IF(PRCPTR.LE.0) GO TO 119
PRCPTR = PRCPTR/8+1 ; get percent pointer
ASSN = ISHFT(IAND(TYPSAV(2,PRCPTR),Y'80000000'),-31)
EXP = ISHFT(IAND(TYPSAV(1,PRCPTR),Y'1F000000'),-24)
EXPF = EXP
SVDAT = IAND(TYPSAV(2,PRCPTR),Y'7FFFF') ; get saved word
SVDATF = SVDAT ; float the word
TMPRES = (SVDATF/2**EXPF)*100
SAVFMT = '(F6.2)'
ENCODE(SAVWRD,SAVFMT) TMPRES ; percent in right format
GO TO 119
C
C *****

```

\*\* DETERMINE STATUS FIELDS' COLORS \*\*

CHECK THE TWO FUNCTIONS OF BOTH THE STATUS FIELDS AND FIND  
THE COLOR CODES FOR THEM IF ACTIVE

\*\*\*\*\*

```

119 K = 0
DO 861 I = 1,3,2
  IF(POBUF(I,PRMROW).LE.0) GO TO 861
  STPTR = POBUF(I,PRMROW)/8 + 1
  IF(TYPSAV(1,STPTR).GE.0) GO TO 861
  DO 862 J = 1,3,2
    JJ = 2*(I-1) + J
    IF(BITBUF(JJ,PRMROW).EQ.0) GO TO 862
  K = K + 1
  LL(K) = JJ
  STWRD = BLC02(2,STPTR)
  BITPTR = BITBUF(JJ,PRMROW)
  BITPTR1 = BITPTR
  IF(BITPTR.LT.0) THEN
    BITPTR = -BITPTR
    STWRD = NOT(STWRD)
  ELSE
    ENDIF
  STAT = BTEST(STWRD,BITPTR)
  IF(STAT) THEN
    COLOR(K) = IAND(DFBUF(K,PRMROW),Y'00FF')
  ELSE
    COLOR(K) = ISHFT(IAND(DFBUF(K,PRMROW),Y'FF00'),-8)
  ENDIF
862 CONTINUE
861 CONTINUE

```

\*\*\*\*\*

\*\* PRINT PAGE INFORMATION \*\*

R - IF RESULT IS CALCULATED FOR A LINE THEN 'R' IS PRINTED  
IN THAT LINE AND THE RESULT VALUE  
P - IF PERCENT IS CALCULATED FOR A LINE THEN 'P' IS PRINTED  
IN THAT LINE AND THE PERCENT VALUE  
S - IF SAVWRD IS CALCULATED THEN 'S' IS PRINTED ALONG WITH  
IT'S VALUE  
K - IF STATUS FIELDS ARE PRESENT FOR A LINE THEN 'K' IS  
PRINTED WITH A VALUE INDICATING THE NUMBER OF LABELS

\*\*\*\*\*

```

RES = 'R'
PER = 'P'
SAV = 'S'
COLR = 'K'

```

IF(RESULT.EQ.' ' .AND. PERCENT.EQ.' ' .AND.

```

C      1 SAVWRD.EQ.'          ' .AND. K.EQ.0) GO TO 100
C      IF(RESULT.EQ.'        ' .AND. PERCENT.EQ.'          ' .AND.
1      SAVWRD.EQ.'          ' .AND. K.NE.0) GO TO 1
C      IF(RESULT.EQ.'        ' .AND. PERCENT.EQ.'          ' .AND.
1      SAVWRD.NE.'          ' .AND. K.EQ.0) GO TO 2
C      IF(RESULT.EQ.'        ' .AND. PERCENT.EQ.'          ' .AND.
1      SAVWRD.NE.'          ' .AND. K.NE.0) GO TO 3
C      IF(RESULT.EQ.'        ' .AND. PERCENT.NE.'          ' .AND.
1      SAVWRD.EQ.'          ' .AND. K.EQ.0) GO TO 4
C      IF(RESULT.EQ.'        ' .AND. PERCENT.NE.'          ' .AND.
1      SAVWRD.EQ.'          ' .AND. K.NE.0) GO TO 5
C      IF(RESULT.EQ.'        ' .AND. PERCENT.NE.'          ' .AND.
1      SAVWRD.NE.'          ' .AND. K.EQ.0) GO TO 6
C      IF(RESULT.EQ.'        ' .AND. PERCENT.NE.'          ' .AND.
1      SAVWRD.NE.'          ' .AND. K.NE.0) GO TO 7
C      IF(RESULT.NE.'        ' .AND. PERCENT.EQ.'          ' .AND.
1      SAVWRD.EQ.'          ' .AND. K.EQ.0) GO TO 8
C      IF(RESULT.NE.'        ' .AND. PERCENT.EQ.'          ' .AND.
1      SAVWRD.EQ.'          ' .AND. K.NE.0) GO TO 9
C      IF(RESULT.NE.'        ' .AND. PERCENT.EQ.'          ' .AND.
1      SAVWRD.NE.'          ' .AND. K.EQ.0) GO TO 20
C      IF(RESULT.NE.'        ' .AND. PERCENT.EQ.'          ' .AND.
1      SAVWRD.NE.'          ' .AND. K.NE.0) GO TO 21
C      IF(RESULT.NE.'        ' .AND. PERCENT.NE.'          ' .AND.
1      SAVWRD.EQ.'          ' .AND. K.EQ.0) GO TO 22
C      IF(RESULT.NE.'        ' .AND. PERCENT.NE.'          ' .AND.
1      SAVWRD.EQ.'          ' .AND. K.NE.0) GO TO 23
C      IF(RESULT.NE.'        ' .AND. PERCENT.NE.'          ' .AND.
1      SAVWRD.NE.'          ' .AND. K.EQ.0) GO TO 24
C
C *****
C
C      ** PRINT STARTS HERE **
C
C *****
C
C      LCOUNT = LCOUNT + 1
C      TOTC(LCOUNT) = 36 + 4*K
C      WRITE(8,47) PRMROW,RES,PER,SAV,COLR,ASSN,RESULT,PERCENT,SAVWRD,
1      K,(LL(L),L=1,K),(COLOR(L),L=1,K)
47 FORMAT(I2,A1,A1,A1,A1,I2,A9,A9,A9,I1,8(Z2))
C      GO TO 100
C
C      1 LCOUNT = LCOUNT + 1
C      TOTC(LCOUNT) = 6 + 4*K
C      WRITE(8,44) PRMROW,COLR,ASSN,K,(LL(L),L=1,K),(COLOR(L),L=1,K)

```

44 FORMAT(I2,A1,I2,I1,8(Z2))  
GO TO 100

C

2 LCOUNT = LCOUNT + 1  
TOTC(LCOUNT) = 14  
WRITE(8,26) PRMROW, SAV, ASSN, SAVWRD  
26 FORMAT(I2,A1,I2,A9)  
GO TO 100

C

3 LCOUNT = LCOUNT + 1  
TOTC(LCOUNT) = 16 + 4\*K  
WRITE(8,27) PRMROW, SAV, COLR, ASSN, SAVWRD, K, (LL(L), L=1, K),  
1 (COLOR(L), L=1, K)  
27 FORMAT(I2,A1,A1,I2,A9,I1,8(Z2))  
GO TO 100

C

4 LCOUNT = LCOUNT + 1  
TOTC(LCOUNT) = 14  
WRITE(8,28) PRMROW, PER, ASSN, PERCENT  
28 FORMAT(I2,A1,I2,A9)  
GO TO 100

C

5 LCOUNT = LCOUNT + 1  
TOTC(LCOUNT) = 16 + 4\*K  
WRITE(8,29) PRMROW, PER, COLR, ASSN, PERCENT, K, (LL(L), L=1, K),  
1 (COLOR(L), L=1, K)  
29 FORMAT(I2,A1,A1,I2,A9,I1,8(Z2))  
GO TO 100

C

6 LCOUNT = LCOUNT + 1  
TOTC(LCOUNT) = 24  
WRITE(8,30) PRMROW, PER, SAV, PERCENT, SAVWRD  
30 FORMAT(I2,A1,A1,A9,A9)  
GO TO 100

C

7 LCOUNT = LCOUNT + 1  
TOTC(LCOUNT) = 24 + 4\*K  
WRITE(8,31) PRMROW, PER, SAV, COLR, ASSN, PERCENT, SAVWRD,  
1 (LL(L), L=1, K), (COLOR(L), L=1, K)  
31 FORMAT(I2,A1,A1,A1,I2,A9,A9,8(Z2))  
GO TO 100

C

8 LCOUNT = LCOUNT + 1  
TOTC(LCOUNT) = 14  
WRITE(8,32) PRMROW, RES, ASSN, RESULT  
32 FORMAT(I2,A1,I2,A9)  
GO TO 100

C

9 LCOUNT = LCOUNT + 1  
TOTC(LCOUNT) = 16 + 4\*K  
WRITE(8,33) PRMROW, RES, COLR, ASSN, RESULT, K, (LL(L), L=1, K),  
1 (COLOR(L), L=1, K)  
33 FORMAT(I2,A1,A1,I2,A9,I1,8(Z2))  
GO TO 100

C

20 LCOUNT = LCOUNT + 1  
TOTC(LCOUNT) = 24  
WRITE(8,34) PRMROW, RES, SAV, ASSN, RESULT, SAVWRD  
34 FORMAT(I2,A1,A1,I2,A9,A9)  
GO TO 100

```

C
21 LCOUNT = LCOUNT + 1
   TOTC(LCOUNT) = 26 + 4*K
   WRITE(8,35) PRMROW,RES,SAV,COLR,ASSN,RESULT,SAVWRD,K,
1   (LL(L),L=1,K),(COLOR(L),L=1,K)
35 FORMAT(I2,A1,A1,A1,I2,A9,A9,I1,8(Z2))
   GO TO 100

C
22 LCOUNT = LCOUNT + 1
   TOTC(LCOUNT) = 24
   WRITE(8,36) PRMROW,RES,PER,ASSN,RESULT,PERCENT
36 FORMAT(I2,A1,A1,I2,A9,A9)
   GO TO 100

C
23 LCOUNT = LCOUNT + 1
   TOTC(LCOUNT) = 26 + 4*K
   WRITE(8,37) PRMROW,RES,PER,COLR,ASSN,RESULT,PERCENT,K,
1   (LL(L),L=1,K),(COLOR(L),L=1,K)
37 FORMAT(I2,A1,A1,A1,I2,A9,A9,I1,8(Z2))
   GO TO 100

C
24 LCOUNT = LCOUNT + 1
   TOTC(LCOUNT) = 34
   WRITE(8,38) PRMROW,RES,PER,SAV,ASSN,RESULT,PERCENT,SAVWRD
38 FORMAT(I2,A1,A1,A1,I2,A9,A9,A9)

C
100 CONTINUE

C
C *****
C IF LCOUNT IS 0, NO FILE IS PREPARED, SO SEND NO-FILE SIGNAL ('#')
C IF NOT, ARRANGE THE FILE IN RIGHT FORMAT
C *****
C
   IF (LCOUNT.EQ.0) GO TO 910
   DO 39 I = 1,LCOUNT
   NTOTC(I+1) = TOTC(I)
39 CONTINUE
   NTOTC(1) = LCOUNT
   WRITE(9,40) (NTOTC(N),N=1,LCOUNT+1)
40 FORMAT(40(I2))
   CLOSE(8)
   OPEN(8,FILE='TMP.DAT')
   L = NTOTC(1)
   DO 42 J = 1,L
   NC = NTOTC(J+1)/2
   READ(8,43,END=81) (PBUF(I,J),I=1,NC)
   WRITE(9,43) (PBUF(I,J),I=1,NC)
43 FORMAT(40(A2))
42 CONTINUE

C
C
81 CLOSE(9)
   OPEN(9,FILE='PEFILE.DAT')
   L = NTOTC(1) + 1
   DO 900 J = 1,L
   NC = L
   IF(J.EQ.1) GO TO 901
   NC = NTOTC(J)/2

```

```

901 READ(9,902,END=903) (PBUF(I,J),I=1,NC)
902 FORMAT(40(A2))
900 CONTINUE

```

```

C
C *****
C
C SEND FILE-READY SIGNAL ('*'), WAIT FOR ACKNOWLEDGE SIGNAL ('?')
C THEN SEND DATA ONE BYTE AT A TIME AND WAIT FOR ACK. SIGNAL
C BEFORE THE NEXT BYTE IS SENT. SEND THE EOF SIGNAL ('$') WHEN
C DONE AND WAIT FOR ACK. SIGNAL. GO AND WAIT FOR THE NEXT PAGE
C NUMBER
C
C *****
C
903 AST = '*'
    CALL SYSIO(PBLK,Y'29',1,AST,1,0)
200 CALL SYSIO(PBLK1,Y'49',1,A,1,0)
    CALL ILBYTE(C,A(1),0)
    IF (C.EQ.Y'003F') GO TO 201
    GO TO 200
201 L = NTOTC(1) + 1
    DO 904 J = 1,L
        NC = L
        IF(J.EQ.1) GO TO 905
        NC = NTOTC(J)/2
905 DO 906 I = 1,NC
        DO 907 K = 1,2
            PLACES = (K-1)*8
            START = ISHFT(PBUF(I,J),PLACES)
            CALL SYSIO(PBLK2,Y'29',1,START,1,0)
908 CALL SYSIO(PBLK3,Y'49',1,A,1,0)
            CALL ILBYTE(C,A(1),0)
            IF(C.EQ.Y'003F') GO TO 907
            GO TO 908
907 CONTINUE
906 CONTINUE
904 CONTINUE
    DOLLAR = '$'
    CALL SYSIO(PBLK,Y'29',1,DOLLAR,1,0)
202 CALL SYSIO(PBLK1,Y'49',1,A,1,0)
    CALL ILBYTE(C,A(1),0)
    IF (C.EQ.Y'003F') GO TO 203
    GO TO 202
203 CLOSE(8)
    CLOSE(9)
    GO TO 999
910 NUMB = '#'
    CALL SYSIO(PBLK,Y'29',1,NUMB,1,0)
    CLOSE(8)
    CLOSE(9)
    GO TO 999
END

```

```

*****
'*
'*
'*          Filename:  CRTPAGE.BAS
'*
'*          Developed by:  A. M. Hasanul Basher
'*          Date last revised:  August 14, 1992
'*
'*
'*  This program displays CRT pages on the PC screen by sharing
'*  information with Perkin-Elmer computer. Another program must be
'*  running concurrently on the Perkin-Elmer that can communicate with
'*  the PC. It gets the page number from the user, transmits to the
'*  Perkin-Elmer via RS-232. The Perkin-Elmer, when prepares a parameter
'*  file for the page, sends back to the PC. Using this file the PC
'*  updates the parameter of the page where page text is read from it's
'*  own hard-drive. The parameter will be updated every 25 seconds if
'*  no new page display is requested.
'*
'*
'*          To display a page do the following:
'*
'*          a. Run PAGPRM.FTN on the Perkin-Elmer
'*          b. Run CRTPAGE.BAS on the PC
'*          c. Type page number (don't press ENTER)
'*
'*          Type E to exit program
'*
*****

```

```

DEFINT A-Z
DECLARE SUB RDREADY (RDSIGNAL, PORT)
DECLARE SUB CHKPRT (STCODE, PORT)
DECLARE SUB PAGESEND (SNDCODE, PORT, PAGE)
DECLARE FUNCTION RDCHR$ ()
DECLARE SUB SNDREADY (SNDSIGNAL, PORT)
DECLARE SUB LWAIT (DELAY, DUM)
DECLARE FUNCTION ACKN$ ()
DECLARE SUB SETCRSR (CSRSETCODE, ROW, COLUMN, CURPAGE)
DECLARE SUB RDSCREEN (RDSCRNCODE, CHAR, CURPAGE)
DECLARE SUB WTSCREEN (WTSCRNCODE, ATTRIBUTE, CHAR)
DECLARE SUB UPDTCRSR (CSRUPDCODE, CHAR, CURPAGE)

```

```
'$INCLUDE: 'QB.BI'
```

```

DIM SHARED INREGS AS REGTYPE, OUTREGS AS REGTYPE
DIM SHARED PAGNO AS LONG
DIM SHARED BYTPOS AS LONG
DIM B$(24)
DIM NOCHR$(2)
DIM TOTCHR(24)
DIM LNDIGIT$(2)
DIM PRMDAT$(2000)
DIM DUMDAT$(2000)
DIM CRTLINE$(24)
DIM X$(24)
DIM Y$(24)
DIM FLIN$(24)
DIM PRMCHR$(24, 4)
DIM PRMNO(24)

```

## 'Variable initialization

```

INICODE = &H00FA
STCODE = &H0300
RDCODE = &H0200
RDSIGNAL = &H013F
SNDSIGNAL = &H0124
SNDCODE = &H0100

```

```

CSRSETCODE = &H0200
RDSCRNCODE = &H0800
CSRUPDCODE = &H0E00
CURPAGE = 0
PORT = 0

```

```

PGDELAY = 3500
RDDELAY = 800
PRMDELAY = 300
MAXLINE = 24
RCOLUMN = 32
PCOLUMN = 66
SCOLUMN = 75

```

```

ASSNSTCOL = 17
ASSENDCOL = 43
ASSNATTR = 4

```

```

STLAB1COL = 46
STLAB2COL = 50
STLAB3COL = 55
STLAB4COL = 59
DUM = 0
TXTCOL = 7

```

```

/*****
/*
/*          ** PROGRAM BEGINS HERE **
/*
/*****

```

'Clear the screen, set screen color, 7 for white, 2 for green

```

CLS
COLOR TXTCOL, 0, 3

```

'Initialize port. AH is 00 and AL has the initialization data,  
'and DX has the port number. Port is initialized for 9600  
'baud rate, even parity, 1 stop bit and 7-bit word.

```

INREGS.AX = INICODE
INREGS.DX = PORT
CALL INTERRUPT(&H14, INREGS, OUTREGS)

```

'First time get the page number

```

DO
  K$ = K$ + INKEY$
LOOP WHILE LEN(K$) < 2

```

```

Q$ = K$
ICHECK = 0
GOSUB DISPPAGE
K$ = ""

```

```

*****
/*
/*
/*          *** MAIN LOOP ***
/*
/* Second time on, stay in the loop. Set timer on for continuous update
/* of page parameters every 25 seconds. If page number is entered display
/* the page. Stop program if "E" is pressed.
/*
/*
*****

```

```

TIMER ON
ON TIMER(25) GOSUB DISPPAGE

```

```

L$ = ""
DO
  ICHECK = 1
  K$ = INKEY$
  IF K$ <> "" THEN
    IF K$ = "E" THEN EXIT DO
    TIMER OFF
    DO
      K$ = K$ + INKEY$
      LOOP WHILE LEN(K$) < 2
      Q$ = K$
      DG1$ = MID$(Q$, 1, 1)
      DG2$ = MID$(Q$, 2, 1)
      IF LEN(DG1$) = 0 THEN DG1$ = "0"
      IF LEN(DG2$) = 0 THEN DG2$ = "0"
      Q$ = DG1$ + DG2$
      ICHECK = 0
      GOSUB DISPPAGE
      TIMER ON
      K$ = ""
    ELSE
      END IF
  LOOP WHILE L$ = ""
  LOCATE 23,1
  END

```

```

*****
/*
/*          ** PAGE DISPLAY, PARAMETER UPDATE **
/*
/* Display a page if page number is entered, otherwise just update the
/* page parameters.
/*
/*
*****

```

DISPPAGE:

```

GOSUB SNDPAGNO
GOSUB FLREADY

IF C$ = "#" THEN

```

```

        GOSUB GETPAGTEXT
        GOSUB DISPLTEXT
ELSE
        GOSUB GETPARM
        GOSUB MKPRMFL

IF ICHECK = 0 THEN
        CLS
        GOSUB GETPAGTEXT
        GOSUB DISPLTEXT
ELSE
END IF
        GOSUB PRMTYPES
        GOSUB UPDTPRM
        GOSUB STCOLOR
        GOSUB ASSIGN
END IF

```

```
RETURN
```

```

*****
*
*           *** SUBROUTINES ***
*
*****

```

```

'Send page number send ready signal and wait for ack. signal
'send ready signal is "$", ack. signal is "?". After ack. signal
'send the page digits.

```

```
SNDPAGNO:
```

```

        DELAY = PGDELAY
        CALL SNDREADY(SNDSIGNAL,PORT)
        CALL LWAIT(DELAY,DUM)
        CALL CHKPRT(STCODE,PORT)
        DUMMY$ = ACKN$

FOR I = 1 TO 2
        PAGE = ASC(MID$(Q$, I, 1))
        CALL PAGSEND(SNDCODE,PORT,PAGE)
        CALL LWAIT(DELAY,DUM)
        CALL CHKPRT(STCODE,PORT)
        DUMMY$ = ACKN$
NEXT I

```

```
RETURN
```

```

*****
'Wait for parameter file ready signal ("*") from Perkin-Elmer. No file
'is prepared if signal recvd. is "#". If file ready, read the file.

```

```
FLREADY:
```

```

        CALL CHKPRT(STCODE,PORT)
DO
        INREGS.AX = RDCODE
        INREGS.DX = PORT
        CALL INTERRUPT(&H14,INREGS,OUTREGS)

```

```

C$ = CHR$(OUTREGS.AX AND &H7F)
IF C$ = "#" THEN EXIT DO
LOOP WHILE C$ <> "*"

```

```

RETURN

```

```

/*****

```

```

'Get the parameters of the file from Perkin-Elmer. Send read-ready signal,
'then read a character followed by sending ack. signal after each
'character read. Read-ready and ack. signal are both "?".

```

```

GETPARM:

```

```

    CALL RDREADY(RDSIGNAL,PORT)
    DELAY = PRMDELAY
    II = 0
DO
    CALL LWAIT(DELAY,DUM)
    CALL CHKPRT(STCODE,PORT)
    II = II + 1
    DUMDAT$(II) = RDCHR$
    CALL RDREADY(RDSIGNAL,PORT)
LOOP WHILE DUMDAT$(II) <> "$"

```

```

    LENGTH = II - 1

```

```

RETURN

```

```

/*****

```

```

'After receiving data from Perkin-Elmer, prepare similar file. Discard
'undesired characters that may be coming from the other side.
'Characters we need are all in CHECK$.

```

```

MKPRMFL:

```

```

    CHECK$ = "C0123456789 .+-%RPSKEINATIVE"

```

```

    JJ = 0

```

```

FOR I = 1 TO LENGTH
    FOR J = 1 TO 28
        TMP$ = MID$(CHECK$, J, 1)
        IF DUMDAT$(I) = TMP$ THEN
            JJ = JJ + 1
            PRMDAT$(JJ) = DUMDAT$(I)
            EXIT FOR
        ELSE
            END IF
    NEXT J
NEXT I

```

```

    TOTLINE = 10*VAL(PRMDAT$(1)) + VAL(PRMDAT$(2))

```

```

    LL = 0

```

```

    NEXTLOC = 3

```

```

FOR J = 1 TO TOTLINE
    FOR K = NEXTLOC TO NEXTLOC+1

```

```

        LL = LL + 1
        NOCHR$(LL) = PRMDAT$(K)
    NEXT K
    NEXTLOC = NEXTLOC + 2
    TOTCHR(J) = 10*VAL(NOCHR$(1)) + VAL(NOCHR$(2))
    LL = 0
NEXT J

```

'Prepare the file of parameters here.

```

        OPEN "PEFILE.DAT" FOR OUTPUT AS #2

    FOR II = 1 TO TOTLINE
        LINECHR = TOTCHR(II)
        LINEND = NEXTLOC + LINECHR - 1
    FOR JJ = NEXTLOC TO LINEND
        PRINT #2, PRMDAT$(JJ);
    NEXT
        PRINT #2, CHR$(13)
        NEXTLOC = LINEND + 1
    NEXT
        CLOSE #2

```

'Rearrange the file by discarding the blank lines

```

        I = 0
        OPEN "PEFILE.DAT" FOR INPUT AS #2
    DO UNTIL EOF(2)
        I = I + 1
        LINE INPUT #2, B$(I)
        LINE INPUT #2, P$
    LOOP
        CLOSE #2
        OPEN "PEFILE.DAT" FOR OUTPUT AS #2
    FOR K = 1 TO I
        PRINT #2, B$(K)
    NEXT
        CLOSE #2

```

RETURN

\*\*\*\*\*

'Locate the position at the beginning of the page, read 23 lines  
'from there.

GETPAGTEXT:

```

        PAGNO = VAL(Q$)
        OPEN "PCPAGE.TXT" FOR INPUT AS #1
        IF PAGNO = 0 THEN BYTPOS = 1
        IF PAGNO > 0 THEN BYTPOS = CLNG(PAGNO*1968 + 1)
        SEEK #1, BYTPOS

        TLINE = 0
    DO
        TLINE = TLINE + 1
        LINE INPUT #1, X$(TLINE)
        CRTLINE$(TLINE) = MID$(X$(TLINE), 18, 2)
    LOOP WHILE TLINE < 23

```

CLOSE #1

RETURN

\*\*\*\*\*

'Find types of parameters and number of parameters in each line

PRMTYPES:

```

OPEN "PEFILE.DAT" FOR INPUT AS #2
COUNT = 0
DO UNTIL EOF(2)

COUNT = COUNT + 1
LINE INPUT #2, Y$(COUNT)
FLINE$(COUNT) = MID$(Y$(COUNT), 1, 2)
PRMCNT = 0

FOR J = 1 TO 4
  W$ = MID$(Y$(COUNT), 2+J, 1)
  IF W$ = "R" OR W$ = "S" OR W$ = "P" OR W$ = "K" THEN

      PRMCNT = PRMCNT + 1
      PRMCHR$(COUNT,PRMCNT) = W$

  ELSE
    END IF
  PRMNO(COUNT) = PRMCNT
NEXT J
LOOP
CLOSE #2
RETURN
```

\*\*\*\*\*

'Update parameter values and/or colors on the screen. Find CRT line  
'number, check parameter type, set column location, print new values  
'and/or colors.

UPDTPRM:

```

FOR J = 1 TO COUNT
FOR I = 1 TO MAXLINE

IF CRTLINE$(I) = FLINE$(J) THEN
TOTPRM = PRMNO(J)
FOR N = 1 TO TOTPRM
IF PRMCHR$(J,N) = "K" THEN EXIT FOR
IF PRMCHR$(J,N) = "R" THEN COLUMN = RCOLUMN
IF PRMCHR$(J,N) = "P" THEN COLUMN = PCOLUMN
IF PRMCHR$(J,N) = "S" THEN COLUMN = SCOLUMN
START = 5 + TOTPRM + (N-1)*9
NCHAR = 9
IF PRMCHR$(J,N) = "S" THEN NCHAR = 6
Z$ = MID$(Y$(J), START, NCHAR)
LOCATE I, COLUMN
PRINT Z$
NEXT N
ELSE
```

```

END IF
NEXT I
NEXT J

```

```

RETURN

```

```

'*****

```

```

'Check if a function is assigned, display in color if assigned.

```

```

ASSIGN:

```

```

FOR J = 1 TO COUNT
FOR I = 1 TO MAXLINE
  ROW = I - 1
  COLUMN = ASSNSTCOL
  ENDCOL = ASSENDCOL
  ATTRIBUTE = ASSNATTR
  TOTPRM = PRMNO(J)
  ASSNLOC = 4 + TOTPRM
  ASSGN$ = MID$(Y$(J), ASSNLOC, 1)
IF CRTLINE$(I) = FLINE$(J) THEN
  IF ASSGN$ = "1" THEN

    CALL SETCRSR(CSRSETCODE, ROW, COLUMN, CURPAGE)

    FOR JJ = COLUMN TO ENDCOL

      CALL RDSCREEN(RDSCRNCODE, CHAR, CURPAGE)
      CALL WTSCREEN(WTSCRNCODE, ATTRIBUTE, CHAR)
      CALL UPDTCRSR(CSRUPDCODE, CHAR, CURPAGE)

    NEXT JJ
  ELSE
  END IF
ELSE
END IF
NEXT I
NEXT J

```

```

RETURN

```

```

'*****

```

```

'Update Status Fields' Labels' colors. Find CRT line number, locate
'label column, find the color code and display in color.

```

```

STCOLOR:

```

```

FOR J = 1 TO COUNT
FOR I = 1 TO MAXLINE
  IF CRTLINE$(I) = FLINE$(J) THEN
    TOTPRM = PRMNO(J)
    ROW = I - 1
    FOR N = 1 TO TOTPRM

      IF PRMCHR$(J, N) = "K" THEN
        TOTLABLOC = (N-1)*10 + 6
        TOTLABEL$ = MID$(Y$(J), TOTLABLOC, 1)
        NOSTLAB = VAL(TOTLABEL$)

```

```

LABNOLOC = TOTLABLOC + 2
LABCOLLOC = (N-1)*10 + 7 + 2*NOSTLAB

```

```

FOR M = 1 TO NOSTLAB
  LABELNO$ = MID$(Y$(J), LABNOLOC, 1)
  BGCOLOR$ = MID$(Y$(J), LABCOLLOC, 1)
  FGCOLOR$ = MID$(Y$(J), LABCOLLOC+1, 1)
  LABEL = VAL(LABELNO$)

  IF FGCOLOR$ = "C" THEN
    FG = 8 + 6
  ELSE
    FG = VAL(FGCOLOR$) \ 2
    IF FG = 2 THEN FG = TXTCOL
  END IF
  BG = VAL(BGCOLOR$)
  ATTRIBUTE = ((BG*16) AND &H70) OR (FG AND &HF)
  IF FG <> 0 THEN
    IF LABEL = 1 THEN COLUMN = STLAB1COL
    IF LABEL = 3 THEN COLUMN = STLAB2COL
    IF LABEL = 5 THEN COLUMN = STLAB3COL
    IF LABEL = 7 THEN COLUMN = STLAB4COL

    CALL SETCRSR(CSRSETCODE, ROW, COLUMN, CURPAGE)

    FOR II = COLUMN TO COLUMN + 2

      CALL RDSCREEN(RDSCRNCODE, CHAR, CURPAGE)
      CALL WTSCREEN(WTSCRNCODE, ATTRIBUTE, CHAR)
      CALL UPDCRSR(CSRUPDCODE, CHAR, CURPAGE)

    NEXT II

  ELSE
    END IF
  LABNOLOC = LABNOLOC + 2
  LABCOLLOC = LABCOLLOC + 2
NEXT M

ELSE
END IF

NEXT N

ELSE
END IF
NEXT I
NEXT J

```

```

RETURN

```

```

/*****

```

```

'Display the texts of the page

```

```

DISPLTEXT:

```

```

  FOR I = 1 TO TLINE
    PRINT X$(I)
  NEXT

```

RETURN

\*\*\*\*\*

'RDREADY subroutine writes the character "?" to the serial port COM1,  
'the write is success if bit-7 of AH register is 0. "?" is the  
'read-ready and the acknowledge signal. FUNCTION 01H is used on INT  
'function call with 01H in AH and 3FH in AL register.

```

SUB RDREADY(RDSIGNAL,PORT)
DO
    INREGS.AX = RDSIGNAL
    INREGS.DX = PORT
    CALL INTERRUPT(&H14,INREGS,OUTREGS)
    LOOP WHILE (OUTREGS.AX AND &H8000) <> 0
END SUB

```

\*\*\*\*\*

'Send page number ready signal("\$") to Perkin-Elmer. Wait until  
'successful

```

SUB SNDREADY(SNDSIGNAL,PORT)
DO
    INREGS.AX = SNDSIGNAL
    INREGS.DX = PORT
    CALL INTERRUPT(&H14,INREGS,OUTREGS)
    LOOP WHILE (OUTREGS.AX AND &H8000) <> 0
END SUB

```

\*\*\*\*\*

'Check port status, wait until data ready. If bit0 of AH is 0, data  
'is not ready. Function 03H is used on INT function call.

```

SUB CHKPRT(STCODE,PORT)
DO
    INREGS.AX = STCODE
    INREGS.DX = PORT
    CALL INTERRUPT(&H14,INREGS,OUTREGS)
    LOOP WHILE (OUTREGS.AX AND 256) = 0
END SUB

```

\*\*\*\*\*

'Send page number to the port. Writes a character to serial port.  
'FUNCTION 01H is used on INT function call with 01H in AH and  
'page character to send in AL register.

```

SUB PAGSEND(SNDCODE,PORT,PAGE)
DO
    INREGS.AX = SNDCODE + PAGE
    INREGS.DX = PORT
    CALL INTERRUPT(&H14,INREGS,OUTREGS)
    LOOP WHILE (OUTREGS.AX AND &H8000) <> 0
END SUB

```

\*\*\*\*\*

'Wait loop

```

SUB LWAIT(DELAY,DUM)
  H = DUM
  DO
    H = H + 1
  LOOP WHILE H < DELAY
END SUB

```

\*\*\*\*\*

'Read character from serial port. FUNCTION 02H is used on INT  
'function call with 02H in AH. The character read from the port  
'is returned in AL. Strip off the parity bit.

```

FUNCTION RDCHR$
  DO
    INREGS.AX = &H0200
    INREGS.DX = 0
    CALL INTERRUPT(&H14,INREGS,OUTREGS)
    DAT = OUTREGS.AX AND &H7F
    A$ = CHR$(OUTREGS.AX AND &H7F)
    LOOP WHILE A$ = "?" OR A$ = CHR$(07) OR A$ = CHR$(94)
    RDCHR$ = A$
  END FUNCTION

```

\*\*\*\*\*

'Read acknowledge signal ("?" ) from Perkin-Elmer, wait until read.

```

FUNCTION ACKN$
  DO
    INREGS.AX = &H0200
    INREGS.DX = 0
    CALL INTERRUPT(&H14,INREGS,OUTREGS)
    D$ = CHR$(OUTREGS.AX AND &H7F)
    LOOP WHILE D$ <> "?"
    ACKN$ = D$
  END FUNCTION

```

\*\*\*\*\*

'set cursor position on the screen, in correct row and column for  
'the current page

```

SUB SETCRSR(CSRSETCODE,ROW,COLUMN,CURPAGE)
  INREGS.AX = CSRSETCODE
  INREGS.BX = CURPAGE
  INREGS.DX = ROW*256 + COLUMN
  CALL INTERRUPT(&H10, INREGS, OUTREGS)
END SUB

```

\*\*\*\*\*

'read character at the cursor. Character is returned in AL register.

```
SUB RDSCREEN (RDSCRNCODE, CHAR, CURPAGE)
```

```
    INREGS.AX = RDSCRNCODE
    INREGS.BX = CURPAGE
    CALL INTERRUPT(&H10, INREGS, OUTREGS)
    CHAR = OUTREGS.AX AND &HFF
```

```
END SUB
```

```
'*****'
```

```
'write character and attribute at cursor. Color attribute in BX and  
'character in AL register.
```

```
    SUB WTSCREEN (WTSCRNCODE, ATTRIBUTE, CHAR)
```

```
        INREGS.AX = &H0900 + CHAR
        INREGS.BX = ATTRIBUTE
        INREGS.CX = 1
        CALL INTERRUPT(&H10, INREGS, OUTREGS)
```

```
END SUB
```

```
'*****'
```

```
'update cursor position after a character is written
```

```
    SUB UPDTCRSR (CSRUPDCODE, CHAR, CURPAGE)
```

```
        INREGS.AX = CSRUPDCODE + CHAR
        INREGS.BX = CURPAGE
        CALL INTERRUPT(&H10, INREGS, OUTREGS)
```

```
END SUB
```



## INTERNAL DISTRIBUTION

1. Central Research Library
2. Document Reference
- 3-4. Laboratory Records
5. Laboratory Records, R.C.
6. ORNL Patent Office
7. Y-12 Technical Library
8. G. D. Alton
9. J. B. Ball
10. F. E. Bertrand
11. D. T. Dowling
12. D. L. Haynes
13. C. M. Jones
14. R. C. Juras
15. S. N. Lane
16. M. J. Meigs
17. G. D. Mills
- 18-20. S. W. Mosko
21. D. K. Olsen
22. B. A. Tatum

## EXTERNAL DISTRIBUTION

23. Dr. A. M. Hasanul Basher, School of Engineering Technology, South Carolina State College, Orangeburg, SC 29117
24. J. T. Crockett, Jr., Director, Faculty and HBCU Programs, ORAU, Energy Bldg., Room OB-9, Oak Ridge, Tennessee 37831
25. Assistant Manager, Energy Research and Development, DOE/ORO
- 26-27. Office of Scientific and Technical Information, P. O. Box 62, Oak Ridge, TN 37831