ORNL/TM-11928

ornl

OAK RIDGE
NATIONAL
LABORATORY

MARTIN MARIETTA

Development of 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

ORNL/TM--11928

DE92 004107

Physics Division

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

A. M. Hasanul Basher

Manuscript Completed–August 16, 1991
Date Published–September 1991

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

DEVELOPMENT OF A REMOTE CONTROL CONSOLE

FOR THE

HHIRF 25-MV TANDEM ACCELERATOR*

A. M. Hasanul Basher[1]
Physics Division
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831-6368

August 16, 1991

* Research sponsored by the U.S. Department of Energy under contract DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc.

[1] Summer Faculty Research Participant, School of Engineering Technology, South Carolina State College, Orangeburg, South Carolina 29117.

Table of Contents

List of Figures

Acknowledgements

This program was sponsored by Oak Ridge Associated Universities (ORAU) at Oak Ridge, Tennessee, and conducted under the supervision of Raymond C. Juras 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 my supervisor, Raymond C. Juras, for his invaluable suggestions and proper guidance in carrying out this project. I also wish to express my gratitude to Martha J. Meigs and B. Alan Tatum 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. They are D. K. Olsen and S. W. Mosko. I would like to thank the staff members for their help and cooperation.

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

DEVELOPMENT OF A REMOTE CONTROL CONSOLE FOR THE 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.

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 and a communication software package. Hardware configuration has been established, a communication software package has been selected, and a software program that reads the pages from the shared memory has been developed.

In the following sections, we present the implementation strategy, works completed, existing and new page format, future action plans, explanation of pages and use of related global variables, a sample session, and flowcharts. The software developed is given in the appendix.

2. IMPLEMENTATION STRATEGY

This project will be carried out in the following three phases:

1. Development of software that will read the shared memory of the Perkin-Elmer computer and correctly display each CRT page in the desired format.

2. Development of a protocol that will allow a PC to communicate with the Perkin-Elmer computer through RS-232. Development of software communication "layers" on the PC and Perkin-Elmer computer to implement the protocol.

3. Development of software that will provide an operator with the following desirable features:

    a. Dedicated display of pages on PC with continuously updated parameters.

    b. Display of 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. ELEMENTS COMPLETED

The first phase of the implementation strategy has been successfully completed. The software is written in FORTRAN.

Upon request from the user, this program reads the contents of the selected page from the shared memory, calculates the value (actual value and percent of the value) and displays on the screen. The program also indicates the status of each parameter. With a single keystroke, the user can redisplay and update the parameter values and device status of the current page, end the session, or see a different page by entering a page number. New pages are displayed with updated parameters, including values and status.

The PC page display is slightly different from the existing one. The existing CRT page has 27 lines and each line has 64 characters. The PC monitor has only 24 lines with 80 characters in each line. Since, in the final form of the project the pages will be displayed on and parameters will be controlled from a PC terminal, page format needed to be changed. The following two figures (Fig. 1 and Fig. 2) explain the differences between the two formats.

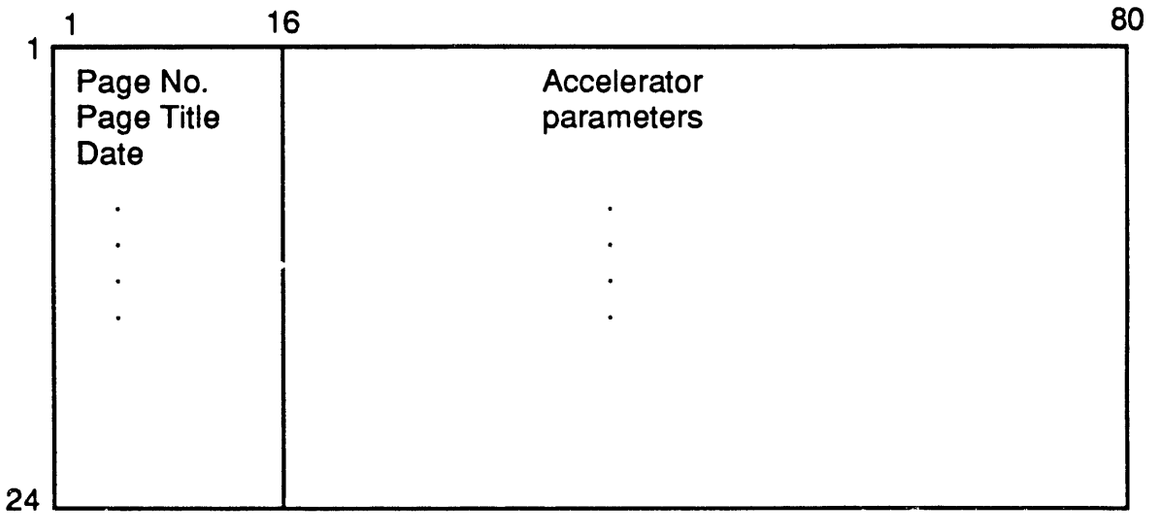

Figure 1. New Page Format

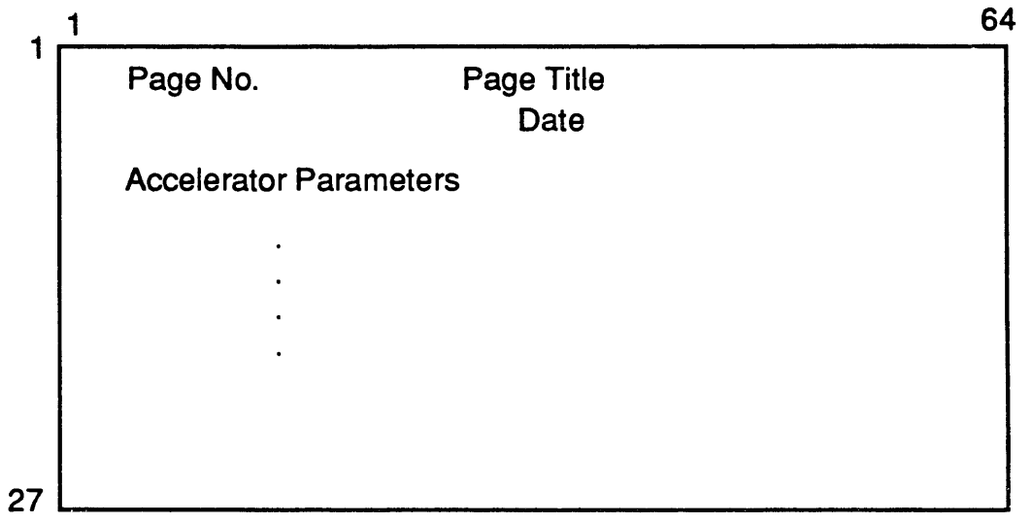

Figure 2. Existing Page Format

The first line of the existing page (Fig. 2) is a blank followed by the line that contains page number and page title. The next line contains the date of the last page revision. The rest of the page contains the information about the accelerator parameters.

In the PC page (Fig. 1), the page number, page title and the date are displayed in the leftmost 16 columns. The accelerator parameters are displayed in the 64 rightmost columns. Any blank lines are removed. The program indicates active status parameters (i.e., ON/OFF, IN/OUT, OPN/CLO, etc.) by printing an asterisk (*) adjacent to the appropriate status label. An example of the new and the existing CRT page display are shown in the following figures (Fig. 3 and Fig. 4).

The detailed description of the software, the global variables and their functions and connections to different accelerator parameters on the pages are presented later in this report.

4. FUTURE ACTION PLANS

Since the first phase is completed, the next logical step will be to start the second phase. In this phase, a protocol will be developed that will let the PC and the Perkin-Elmer communicate with each other by sharing information. The PC software will send requests from the PC to the Perkin-Elmer. The transfer of the requested data will be accomplished with the aid of the protocol via RS-232. The request could be to send information about a CRT page or a certain accelerator parameter or any control action. The request received by the Perkin-Elmer will be decoded. Once decoded, the Perkin-Elmer will respond either by sending information or by changing the control parameters in the shared memory, as requested. Development of the protocol for communicating between the PC and the Perkin-Elmer is crucial. A reliable communication package will be developed so that the information and/or data is not corrupted during the transfer. For the same reason, it will be wise to keep the amount of data transfer between the two computers to a minimum. To accomplish this, we

```
   PAGE 15      1  FC  I1-1  -2.71E-07 A      IN/OUT** ON/OFF
                2  EQX 11-1       5.77 KV  * ON/OFF                40.21     40.21
HORIZONTAL PRE- 3  EQX 11-1      -5.87 KV  * ON/OFF                40.21     40.21
ACCELERATION B- 4  EQY 11-1       6.56 KV  * ON/OFF                41.06     41.06
EAM LINE COMPO- 5  EQY 11-1      -6.14 KV  * ON/OFF                41.06     41.06
NENTS           7  ESX 11-1       0.00 KV    ON/OFF*               51.68     51.68
                8  ESX 11-1      -0.03 KV    ON/OFF*               51.68     51.68
    12/10/87    9  ESY 11-1       0.02 KV    ON/OFF*               49.02     49.02
               10  ESY 11-1      -0.02 KV    ON/OFF*               49.02     49.02
               12  IGC 11-2   0.84E -8 T    * ON^OFF  ALM
               13  BPM 11-1
               15  DSL 11-1  -1.92E-06 A    * ON/OFF
               16  DSR 11-1  -1.12E-06 A    * ON/OFF
               17  DSU 11-1  -1.01E-10 A    * ON/OFF
               18  DSD 11-1  -1.01E-10 A    * ON/OFF
               20  FC  11-1  -1.01E-10 A      IN/OUT** ON/OFF
               21  BLV 11-2                 *OPN^CLO  ALM
               22  BM  13-1       73.46 A   * ON^OFF * HI LO       32.88     32.88
               23  FC  13-1  -1.01E-10 A      IN/OUT** ON/OFF
               24  BPM 13-1
```

Figure 3. New Page Display

```
      PAGE 15    HORIZONTAL PREACCELERATION BEAM LINE COMPONENTS
                                                          12/10/87
 1  FC  I1-1   -2.71E-07 A     IN/OUT   ON/OFF
 2  EQX 11-1        5.77 KV    ON/OFF                40.21     40.21
 3  EQX 11-1       -5.87 KV    ON/OFF                40.21     40.21
 4  EQY 11-1        6.56 KV    ON/OFF                41.06     41.06
 5  EQY 11-1       -6.15 KV    ON/OFF                41.06     41.06

 7  ESX 11-1        0.00 KV    ON/OFF                51.68     51.68
 8  ESX 11-1       -0.03 KV    ON/OFF                51.68     51.68
 9  ESY 11-1        0.02 KV    ON/OFF                49.02     49.02
10  ESY 11-1       -0.02 KV    ON/OFF                49.02     49.02

12  IGC 11-2    0.83E -8 T     ON^OFF  ALM
13  BPM 11-1

15  DSL 11-1   -1.92E-06 A     ON/OFF
16  DSR 11-1   -1.12E-06 A     ON/OFF
17  DSU 11-1   -1.01E-10 A     ON/OFF
18  DSD 11-1   -1.01E-10 A     ON/OFF

20  FC  11-1   -1.01E-10 A     IN/OUT   ON/OFF
21  BLV 11-2                  OPN^CLO  ALM
22  BM  13-1        73.46 A    ON^OFF   HI LO       32.88     32.88
23  FC  13-1   -1.01E-10 A     IN/OUT   ON/OFF
24  BPM 13-1
```

Figure 4. Existing Page Display

plan to do most of the numerical calculations (wherever possible) on the Perkin-Elmer side. To make sure the information transfer from one side to the other is reliable, the protocol will be tested for various actions.

Once the protocol is developed, we will move to the third phase. In this phase, we have several things to do. The current page does not update the parameters automatically and the status of parameters are indicated by an asterisk. Software will be developed on the PC side that will keep the page text static, but the parameters will be updated continuously and also the parameter status will be displayed in color. Parameter value and status information will be fetched from shared memory with the aid of the protocol developed in the second stage.

There will be some additional monitoring and control parameter display on the CRT pages. These are labels for assignable knobs and values for meters. 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 for either 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 changed parameter will be saved.

5. GLOBAL VARIABLES AND PAGES

This section presents the detailed descriptions of global variables and their relations to the CRT pages. These variables provide us with information such

as page text, actual data, data type, data size, data display format, control setting and input/output status of control parameters, and display format for labels, etc. for each CRT page.

In the next section (a sample session) of this report, a specific example will be given and the use of some of the variables in obtaining pertinent information about a CRT page will be explained.

a. ABUF (16, 32):

This variable contains page text. The second subscript (32) of this variable indicates the possible number of lines in a page. Only 27 lines are implemented.

First subscript (16) indicates the number of words in each line. Each of these subscripts contains four (4) characters, including blanks and other special symbols, if any. Since ASCII format is used, each character requires one byte. Thus, a word is 4 bytes (0-3).

If this variable is read from shared memory and all lines are printed on screen, it will display all the text of a page (whatever selected), excluding parameter values and/or control settings. Figure 5 on the next page is an example of the page text display of CRT page 15. The top line with asterisks is not a part of the page, rather it is printed to make it easier to locate the words of the variable ABUF. The actual page starts on the next line, which is a blank line.

```
***1***2***3***4***5***6***7***8***9***0***1***2***3***4***5***6

      PAGE 15    HORIZONTAL PREACCELERATION BEAM LINE COMPONENTS
                                                          12/10/87
 1   FC  I1-1            A      IN/OUT    ON/OFF
 2   EQX 11-1            KV     ON/OFF
 3   EQX 11-1            KV     ON/OFF
 4   EQY 11-1            KV     ON/OFF
 5   EQY 11-1            KV     ON/OFF

 7   ESX 11-1            KV     ON/OFF
 8   ESX 11-1            KV     ON/OFF
 9   ESY 11-1            KV     ON/OFF
10   ESY 11-1            KV     ON/OFF

12   IGC 11-2            T      ON^OFF   ALM
13   BPM 11-1

15   DSL 11-1            A      ON/OFF
16   DSR 11-1            A      ON/OFF
17   DSU 11-1            A      ON/OFF
18   DSD 11-1            A      ON/OFF

20   FC  11-1            A      IN/OUT    ON/OFF
21   BLV 11-2                  OPN^CLO   ALM
22   BM  13-1            A      ON^OFF     HI LO
23   FC  13-1            A      IN/OUT    ON/OFF
24   BPM 13-1
```

Figure 5. Page Text

Let us assume that we are on the second line of the page. We can now identify each word and each byte of a word of ABUF. Let us further assume the characters within the quote marks are the characters in the word and character b represents a blank.

```
ABUF (1, 2)   =   'bbbb' (four blanks)
ABUF (2, 2)   =   'bbPA'
ABUF (3, 2)   =   'GEb1'
      .
      .
      .
ABUF (12, 2)  =   'BEAM'
      .
      .
      .
so on.
```

The words of other lines of ABUF can be identified in the same way.

If we look at the parameter lines that start on the 4th line of the CRT page, we see that words 4th to 6th and the last 3 bytes of word 12 through the 16th word are blanks in most of the lines. Actual parameter values and percent values are displayed in those places, whenever appropriate.

Words 4th to 6th are reserved for actual values. Words 12th through 16th are reserved for percent value obtained from control setting and actual saved data.

Each word of ABUF has four bytes counted as byte 0, 1, 2 and 3. The bytes of the word ABUF (3, 2) for this page can be identified as follows:

Byte 0 = G
Byte 1 = E
Byte 2 = b (blank)
Byte 3 = 1

b. POBUF (6, 24):

A global variable that contains pointer buffers, 6 per line and 24 parameter lines for a single CRT page.

Pointer for a word (a word pointer) can be obtained from pointer buffer utilizing the following mathematical relationship:

Work Pointer = (Pointer Buffer)/ 8 + 1.

The actual word in the shared memory can be located and read with the help of word pointer.

POBUF (5, L) pointer is used to find the actual data, data type, etc. which can be used to calculate a parameter value for parameter line L.

POBUF (6, L) gives pointer for data and data size that are used to calculate the percent of control setting for line L.

POBUF (1, L) is the pointer buffer for input status field 1 in line L. In Fig. 5 on parameter line 1, the status field 1 is IN/OUT.

POBUF (3, L) is the pointer buffer for input status field 2 in line L. The status field 2 is ON/OFF on parameter line 1 as shown in Fig. 5.

POBUF (2, L) and POBUF (4, L) are the pointer buffers for output status field 1 and 2, respectively, for parameter line L.

c. BITBUF (8, 24):

This is bit identifying buffer and another global variable. Each BITBUF contains a pointer for a bit in status word that can be used to identify the status bit. Each status field has two parts. For example, IN and OUT are the two parts of status field 1 on line 1 (Fig. 5). Each of these parts has input bit position and output bit position. The same is true for ON/OFF. So, for each line there are 8 bit pointers and a CRT page has 24 lines. That explains the subscripts 8 and 24 of the variable.

BITBUF (1, L) and BITBUF (2,L) are the input and output status bit position pointers, respectively, for the first part of status field 1 on line L.

BITBUF (3, L) and BITBUF (4, L) are the input and output status bit position pointers, respectively, for the second part of the status field 1 on line L.

BITBUF (5, L) and BITBUF (6, L) are the input and output status bit position pointers, respectively, for the first part of the status field 2 on line L.

BITBUF (7, L) and BITBUF (8, L) are the input and output status bit position pointers, respectively, for the second part of the status field 2 on line L.

d. DFBUF (5, 24):

DFBUF is display format buffer. The first subscript (5) indicates that there are five display fields in each line and the second subscript (24) indicates the total number of lines in each CRT page. The first four fields of DFBUF determine the colors of the functions (ON/OFF, IN/OUT, etc.) of the two status fields in a line. Each DFBUF field contains a code for two possible combinations of colors. Depending on status bit (a status of the function) either the first part or the second part of each status field can have any of the two possible colors.

Each DFBUF contains a 16-bit code, the most significant 8 bits of which indicate one color and the least significant 8 bits indicate another color.

DFBUF (1 or 2, L) carries the color code for the first part of status field 1 and status field 2 in line L for both input and output operations.

DFBUF (3 or 4, L) contains the color code for the second part of both the status fields in line L for both input and output operations.

The color codes are as follows:

02H –> Blue on black
04H –> Green on black
06H –> Blue plus green on black
0CH –> Red plus green on black
42H –> Blue on red

Example:

If the color code for a certain function of status field is 040CH, then the color is determined as follows:

If the bit pointer is positive, then

i. Color is most significant byte, i.e. 04H if the status for the function is 0.
ii. Color is least significant byte, i.e. OCH if the status for the function is 1.

If the bit pointer is negative, then

i. Most significant byte, i.e. 04H is the color for status 1.
ii. Least significant byte, i.e. OCH is the color for status 0.

DFBUF (5, L) contains the code for data display format, for line L. The codes and the respective formats are described below:

Code	Format
0	± xxxx
1	± xxxx.x
2	± xxxx.xx
3	± xxxx.xxx
4	E - format
5	Hexadecimal
6	BCD

* x is a digit

. = a decimal point

e. BLC02 (2, 1024):

This variable carries CNAF information and data. BLC02 is a 4-byte word whose second subscript is a pointer that is obtained from pointer buffer. BLC02 (2, I), where I is the pointer, contains data that is used to calculate parameter value and/or percent value. Parameter value depends on data type and data type comes from another global variable "TYPSAV". TYPSAV is described later in this section. Mathematical relations used to calculate parameter values are given in the Appendix B. Please refer to "A Sample Session" section for a specific example.

BLC02 (1, I) has the information about the crate number to be used, slot, subaddress and function (read or write). These are, in short, called CNAF. Here is how C, N, A and F can be found from BLC02 (1, I).

Bits	0-5	C (crate)
Bits	8-12	N (slot)
Bits	13-16	A (subaddress)
Bits	17-21	F (function)

f. MBLIST (2, 1024):

As mentioned before, to calculate a parameter value, data type is determined first that comes from TYPSAV. Then a mathematical formula is used to calculate the value. For some types of data, MBLIST (*, *) values are required for these calculations, in addition to BLC02 (2, *). Please refer to Appendix B for more details. Wherever needed, MBLIST (1, I) and MBLIST (2, I) will be acting as slope and intercept, respectively, for the calculations where I is word pointer obtained from pointer buffer.

g. TYPSAV (2, 1024):

TYPSAV (1, I) provides information about the type of data. It tells whether data is active or inactive, logical or numeric, input or output. It also gives the data size and data format. The second subscript I acts as a pointer which is again obtained from buffer pointer. Further information about this variable is available in the next section. Now the way the data type is determined will be explained.

Bit 0	1 = Active, 0 = Inactive
Bit 1	1 = Logical, 0 = Numeric
Bit 2	1 = Input, 0 = Output
Bits 3-7	Data size
Bits 8-15	Raw data format

Further explanation about raw data format is given below:

Decimal value of Bits 8-15	Type
0	Linear
1	Table look-up
2 and 3	Linear and antilog (Type 3 negate)
4	Ion gauge
6	Foil stripper (8-bit BCD)
7	NMR
8	Hiconex cone wheel position
9	Gaussmeter
10	Sublimator position
18	NMR search range
19	General linear

TYPSAV (2, I) serves as temporary storage used by shaft encoder save buttons and acknowledges.

Finally, we summarize the types and bytes used by each of the global variables below:

Global Variables with dimension(s)	Type	Bytes
ABUF (16, 32)	INTEGER*4	2048
POBUF (6, 24)	INTEGER*2	288
BITBUF (8, 24)	INTEGER*2	384
DFBUF (5, 24)	INTEGER*2	240
BLC02 (2, 1024)	INTEGER*4	4 (Each)
MBLIST (2, 1024)	REAL	4 (Each)
TYPSAV (2, 1024)	INTEGER*4	4 (Each)

6. A SAMPLE SESSION

In this section, we will show how the buffers are used to obtain word or bit pointers and then how we can read the actual word from the shared memory using those pointers. The words read from the shared memory will eventually provide us with any or all of the information listed below:

a. Actual data
b. Data type
c. Size of data
d. Function active or inactive
e. Function status
f. Function display color

While describing each of the above parameters, reference will be made to Figs. 6, 7 and 8, which are given in the following few pages. In addition to the actual page, some related important parameters are listed that will help us understand each of the following steps.

Assume that we are on parameter line 2 of page 15.

Step 1. Find buffer pointer (BUFPTR)

BUFPTR = POBUF (5, 2) = 428 (Hex) (Fig. 6)

Buffer pointer is +ve

Step 2. Find word pointer (WRDPTR)

WRDPTR = POBUF (5, 2)/8 + 1

= 428 (HEX)/8 + 1 = 134 (Dec)

Step 3. Use the word pointer to get word and determine if the function is active or inactive, and if active, find the data type.

WORD = TYPSAV (1, WRDPTR)

= TYPSAV (1, 134)

= 8C008000 (Hex) (Fig. 6)

Test bit 0 of TYPSAV (1, 128)

It is 1, that indicates it is active.

Since it is active, determine data type.

Find the value of bit 8 through 15 of TYPSAV (1, 134).

= 0000 0000 = 0 (Dec)

-> data is of type 0.

Step 4. Calculate the value using type 0 formula.

VALUE = MBLIST (1, 134) * (value of lower 16 bits of BLCO2 (1, 134)) + MBLIST (2, 134)

MBLIST (1, 134) = 0.37E-02 (Fig. 6)

MBLIST (2, 134) = 0.00E+00 (Fig. 6)

BLC02 (2, 134) = 628 (Hex) (Fig. 6)

VALUE = 0.37E-02 x 1576 + 0.00E+00

= 5.83

Step 5. Check POBUF (6, 2) if percent of value needs to be calculated.

POBUF (6, 2) = 4B8 (Hex) > 0

So calculate percent.

(If type is not 0 or if type is 0 and POBUF (6, L) IS NOT +ve, no percent is calculated.)

Step 6. Calculate percent.

Find percent word pointer (PRCPNT)

PRCPNT = POBUF (6, 2)/8 + 1

= 4B8 (Hex)/8 + 1

= 152 (Dec.) (Fig. 7)

Data size = value of bit 3 through bit 7 of TYPSAV (1, PRCPNT)

= Value of bit 3 through bit 7 of 8 C008000H

= 12 (Dec) (Fig. 7)

Calculate percent value using percent formula:

PERCENT = (BLC02 (2, PRCPNT) / 2** Data Size) x 100

= ((674H) / 2 ** 12) x 100 (Fig. 7)

= (1652 / 2 ** 12) x 100

= 40.332

Step 7. Find status of a function. Only one example is provided here. Let us find the status of first function of status field 1 on line 2.

Get pointer buffer:

POBUF (1, 2) = 580 (Hex) > 0 (Fig. 8)

Find the word pointer for status word:

STPTR = 580 (Hex) / 8 + 1 = 177 (Dec)

Find if active or inactive:

Bit 0 of TYPSAV (1, 177)

= Bit 0 of C110 0000 (Hex) (Fig. 8)

= 1

–> Active

Get status word:

STWRD = BLC02 (2, 177) = 41A (Hex) (Fig. 8)

Find bit pointer for status:

BITPTR = BITBUF (1, 2) = 1C (Hex)

= 28 (Dec) > 0

Find status:

STATUS = Bit 28 of BLC02 (2, 177)

= Bit 28 of 0000 041A (Hex) (Fig. 8)

= 1

Since the status of the first part of status field is 1, print an asterisk (instead of color) next to that function (ON) in line 2.

Value and Related Parameters

The following figure gives the value that is displayed on CRT page 15 and a list of other parameters needed to calculate this value. On the top of the figure, the definitions of the variables used in the printout are given.

Variables		Meanings
LINE	=	Parameter line number
VALUE	=	Actual parameter value
PBF (5, L)	=	POBUF (5, L), L is parameter line
PTR	=	Word pointer (WRDPTR)
TYPE	=	Type of data
TS	=	TYPSAV
BLC	=	BLC02
MB	=	MBLIST

LINE	VALUE	PBF(5,L)	PTR	TYPE	TS(1,PTR)	BLC(2,PTR)	MB(1,PTR)	MB(2,PTR
1	-3.71E-07	3F8	80	3	8C030000	A4A	-0.24E-02	-0.19E-05
2	5.79	428	86	0	8C008000	62E	0.37E-02	0.00E+00
3	-5.89	430	87	0	8C008000	648	-0.37E-02	0.00E+00
4	6.58	438	88	0	8C008000	704	0.37E-02	0.00E+00
5	-6.17	440	89	0	8C008000	696	-0.37E-02	0.00E+00
6		0						
7	0.00	458	8C	0	8C006000	0	0.15E-02	0.00E+00
8	-0.03	5D0	BB	0	8C006000	15	-0.15E-02	0.00E+00
9	0.02	460	8D	0	8C006000	E	0.15E-02	0.00E+00
10	-0.02	5D8	BC	0	8C006000	F	-0.15E-02	0.00E+00
11		0						
12	0.81E -8	508	A2	4	8C042000	14A	0.24E-02	0.00E+00
13		0						
14		0						
15	-1.92E-06	480	91	3	8C030000	926	-0.24E-02	-0.19E-05
16	-1.12E-06	488	92	3	8C030000	986	-0.24E-02	-0.19E-05
17	-1.01E-10	490	93	3	8C030000	FFE	-0.24E-02	-0.19E-05
18	-1.01E-10	498	94	3	8C030000	FFE	-0.24E-02	-0.19E-05
19		0						
20	-1.01E-10	4A0	95	3	8C030000	FFE	-0.24E-02	-0.19E-05
21		0						
22	73.90	570	AF	0	8C00A000	3F1	0.73E-01	0.00E+00
23	-1.01E-10	548	AA	3	8C030000	FFF	-0.24E-02	-0.19E-05
24		0						

Figure 6. Value and Parameters

Percent and Related Parameters

The percent value and control setting and the parameters that are utilized to find those two are listed in the figure below. The variables used in the figure are explained.

Variables Used		Meanings
Line	=	Parameter line number
PBF (6, L)	=	POBUF (6, L), L is parameter line
PRCPNT	=	Word pointer for percent
PERCENT	=	Control setting in percent
SAVWRD	=	Percent from saved data
TS	=	TYPSAV
BLC	=	BLC02

LINE	PBF(6,L)	PRCPNT	PERCENT	SAVWRD	TS(1,PRCPNT)	TS(2,PRCPNT)	BLC(2,PRCPNT)
1	0						
2	4B8	98	40.33	40.33	8C008000	674	674
3	4B8	98	40.33	40.33	8C008000	674	674
4	4C0	99	41.21	41.21	8C008000	698	698
5	4C0	99	41.21	41.21	8C008000	698	698
6	0						
7	4A8	96	51.68	51.68	8C006000	845	845
8	4A8	96	51.68	51.68	8C006000	845	845
9	4B0	97	49.02	49.02	8C006000	7D8	7D8
10	4B0	97	49.02	49.02	8C006000	7D8	7D8
11	0						
12	0						
13	0						
14	0						
15	0						
16	0						
17	0						
18	0						
19	0						
20	0						
21	0						
22	590	B3	33.08	33.07	9300A000	8002A54D	2A56F
23	0						
24	0						

Figure 7. Percent Parameters

Status and Related Parameters

The status indicates whether a function in a line is active or not. The following figure presents the listing of parameters required to determine the status of a function. The listing is partial. The variables used are defined.

Variables		Meanings
Line	=	Parameter line number
PBF	=	POBUF
STPTR	=	Status word pointer
TPS	=	TYPSAV
BTBF	=	BITBUF
BLC	=	BLC02
STAT	=	Status (0 or 1)
JJ	=	Odd for input status

JJ = 1 –> First function of status field 1
JJ = 3 –> Second function of status field 1
JJ = 5 –> First function of status field 2
JJ = 7 –> Second function of status field 2

LINE	I	PBF(I,LN)	STPTR	TPS(1,PTRST)	J	BTBF(JJ,LN)	BLC(2,PTRST)	STAT	JJ
1	1	2E0	5D	C10C0000	1	E	DD16F0	0	1
1	1	2E0	5D	C10C0000	3	D	DD16F0	1	3
1	3	2E0	5D	C10C0000	1	C	DD16F0	1	5
1	3	2E0	5D	C10C0000	3	FFF4	DD16F0	0	7
2	1	580	B1	C1100000	1	1C	41A	1	1
2	1	580	B1	C1100000	3	FFE4	41A	0	3
3	1	580	B1	C1100000	1	1C	41A	1	1
3	1	580	B1	C1100000	3	FFE4	41A	0	3
4	1	580	B1	C1100000	1	1C	41A	1	1
4	1	580	B1	C1100000	3	FFE4	41A	0	3
5	1	580	B1	C1100000	1	1C	41A	1	1
5	1	580	B1	C1100000	3	FFE4	41A	0	3
7	1	580	B1	C1100000	1	1A	41A	0	1
7	1	580	B1	C1100000	3	FFE6	41A	1	3
8	1	580	B1	C1100000	1	1A	41A	0	1
8	1	580	B1	C1100000	3	FFE6	41A	1	3
9	1	580	B1	C1100000	1	1A	41A	0	1
9	1	580	B1	C1100000	3	FFE6	41A	1	3
10	1	580	B1	C1100000	1	1A	41A	0	1
10	1	580	B1	C1100000	3	FFE6	41A	1	3

Figure 8. Status Parameters

7. FLOWCHARTS

The flowcharts of the software developed are included in this section. One of these is the overall flowchart of complete software and others are flowcharts of three major sections of the program. These are

a. Page Number or Date
b. Page Title
c. Function Status
d. CRT Page Display

The Page Number or Date flowchart reads either page number from line 2 or date from line 3 of the existing CRT page and saves in a new variable NABUF.

The Page Title reads the title of the existing page and saves in the new variable starting with a new line number. This may take several lines because each line of this new variable has only sixteen bytes.

The Function Status flowchart determines the status of each function in a line and places an asterisk adjacent to the label of a function that is active.

The CRT Page Display includes all three sections and displays the complete page on the screen with updated values. This part determines the data type, calculates values, determines control setting and percent, if any, and finally prints the page in the new format.

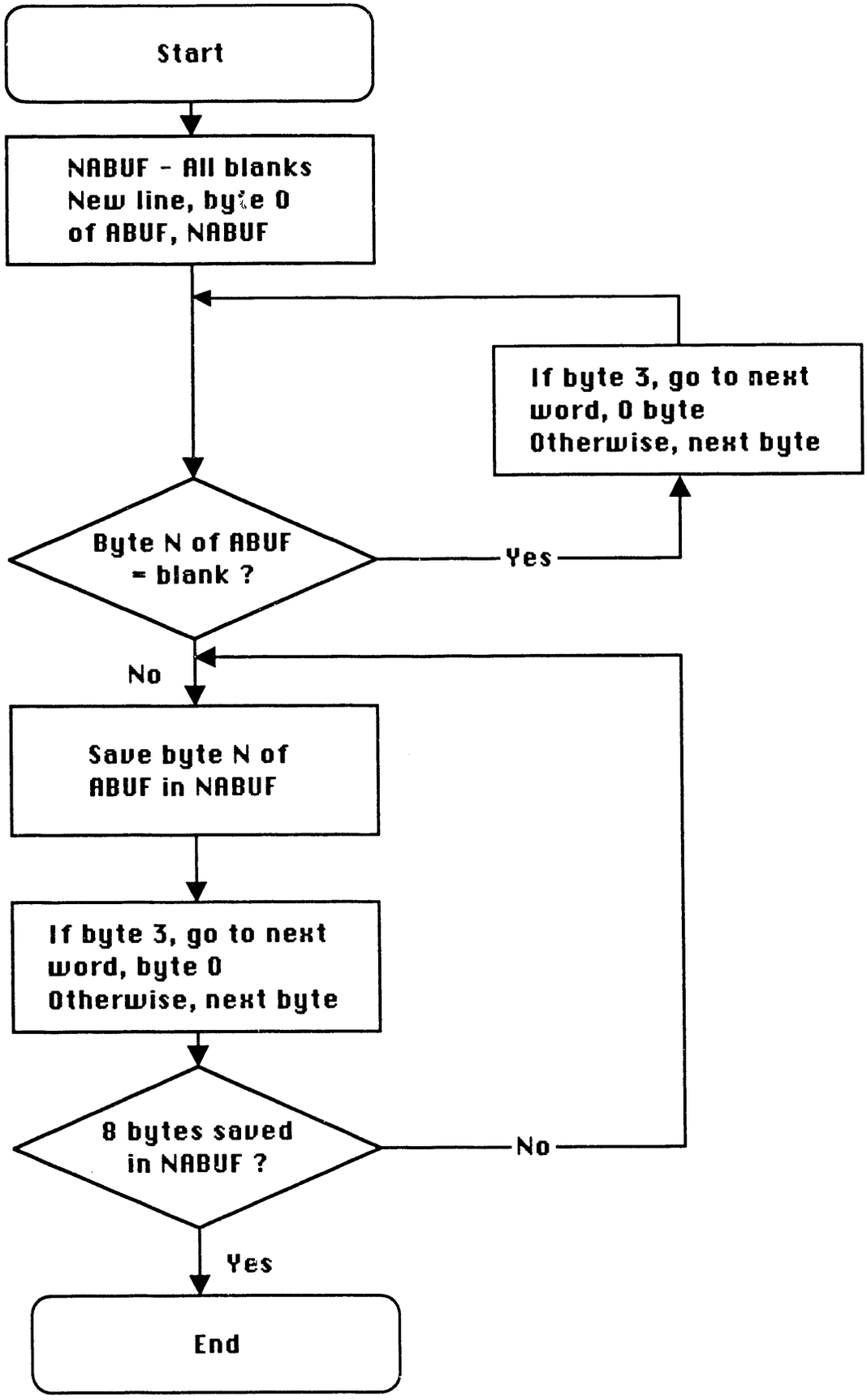

Figure 9. Page Number or Date

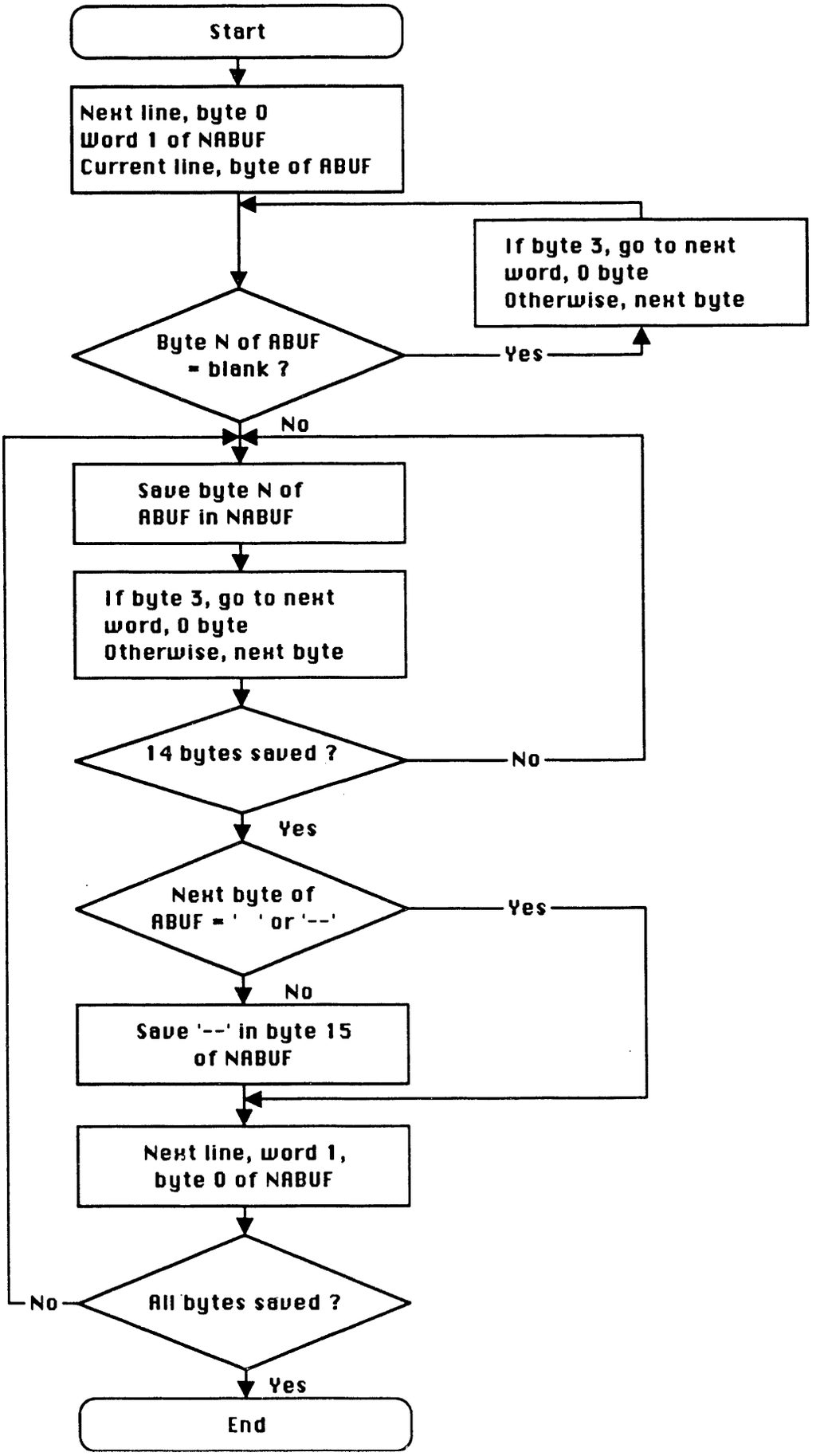

Figure 10. Page Title

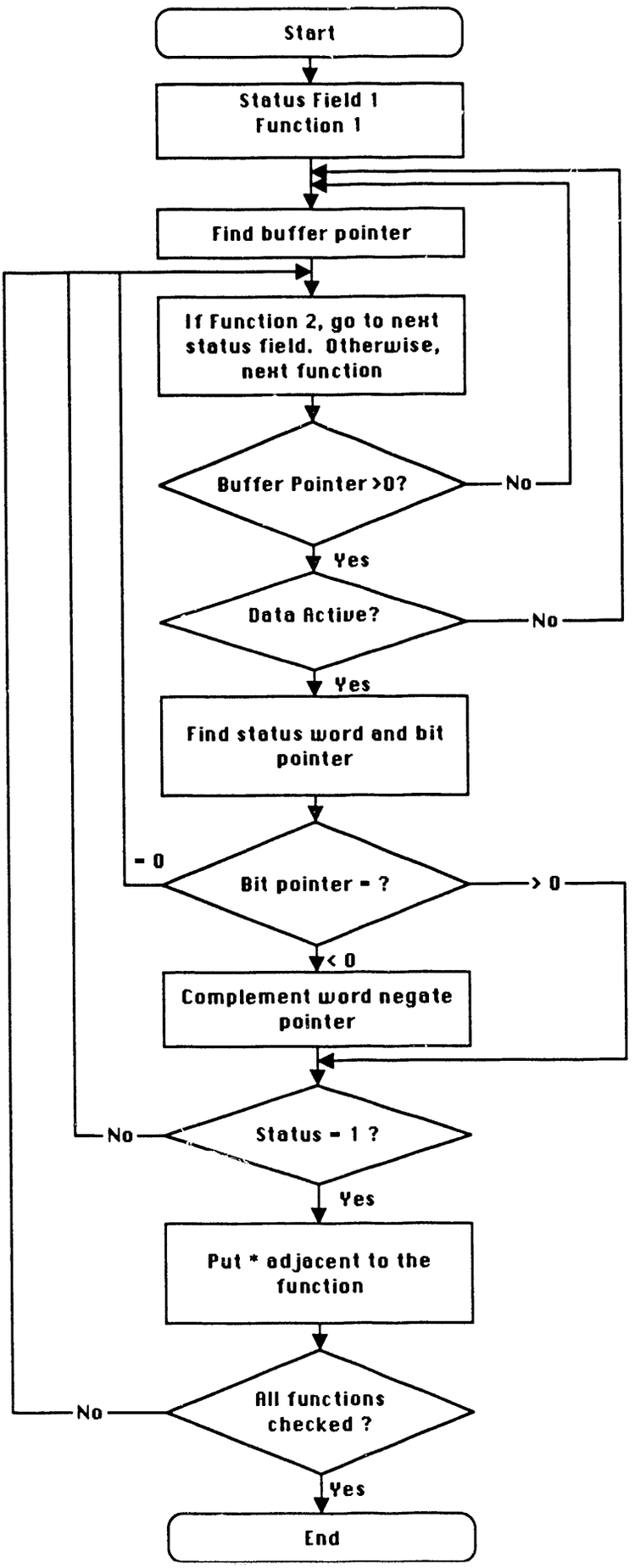

Figure 11. Function Status

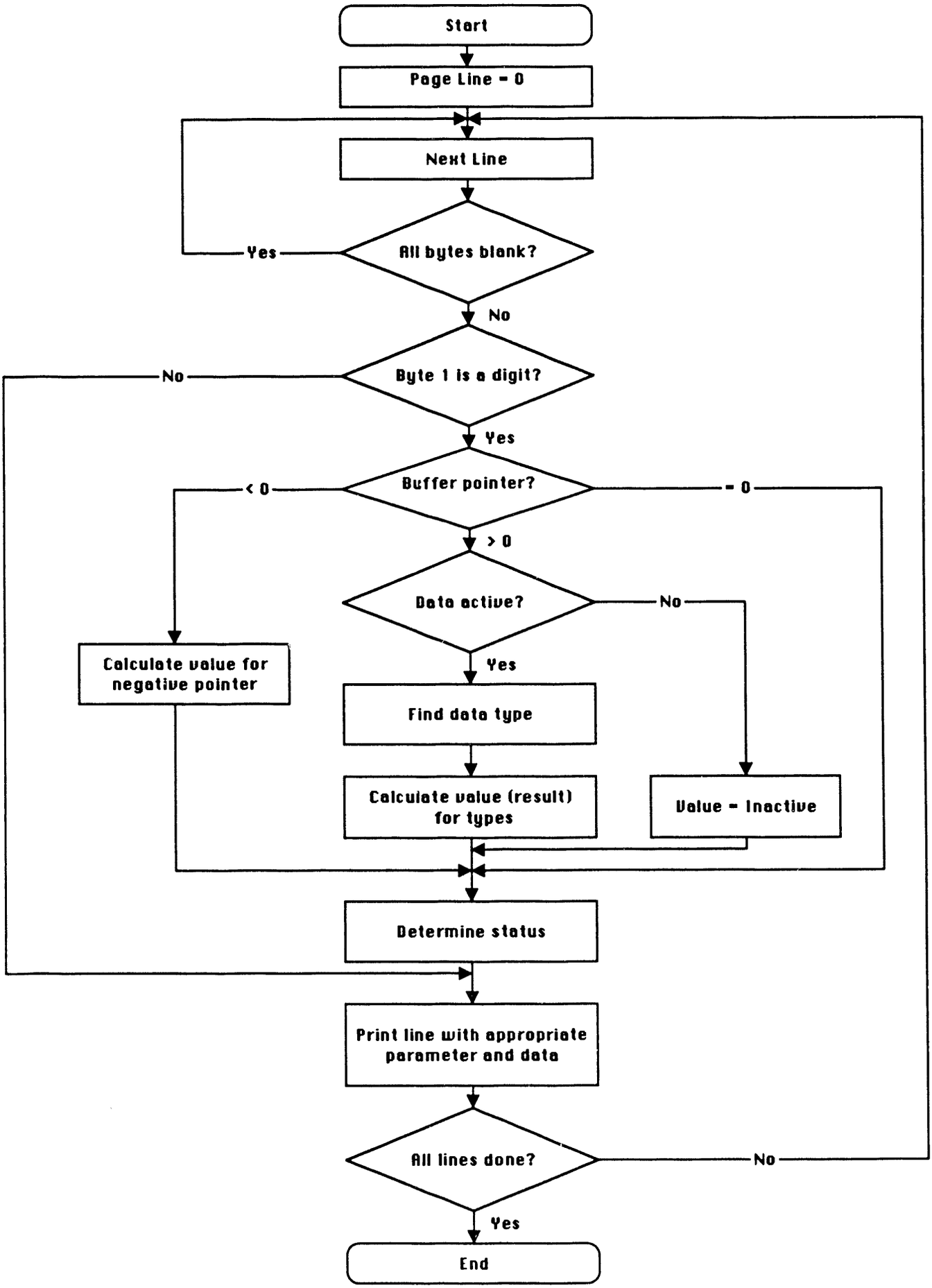

Figure 12. CRT Page Display

APPENDIX A

```
C  ****************************************************************
C  *                                                              *
C  *     Filename : PCPAGE.FTN                                    *
C  *     Developed by:   A. M. Hasanul Basher                     *
C  *     Date last revised: August 16,1991                        *
C  *                                                              *
C  *     The Program Displays CRT Pages in New Format             *
C  *                                                              *
C  *  This program reads the shared memory of PERKIN-ELMER computer *
C  *  computer and displays updated Accelerator pages 0 through 99. *
C  *  The page number, page title and the date when the page was   *
C  *  last revised are all displayed in the leftmost 16 columns and *
C  *  the Accelerator parameters are displayed on the rightmost 64  *
C  *  columns of CRT. Any blank lines are removed.                *
C  *                                                              *
C  *     To start the program: Type PCPAGE and Hit Enter          *
C  *     For Page Display:     Type Page Number and Hit Enter     *
C  *     For Page Redisplay:   Hit Enter Only                     *
C  *     To End the Session:   Type 'E' or 'END' 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 PAGE.LNK                                        *
C  *  ESTABLISH TASK                                              *
C  *  OP WORK=XA00,FLOAT,DFLOAT                                   *
C  *  RESOLVE GBLCOM.SEG/S,ACCESS=R,STRUC=(GBLCOM./X79FC)         *
C  *  INCLUDE PAGE.OBJ                                            *
C  *  INCLUDE BCDBIN.OBJ/S                                        *
C  *  LIBRARY RCJLIB.OBJ/S,F7RTL.OBJ/S                            *
C  *  MAP NULL:                                                   *
C  *  BUILD PAGE.TSK                                              *
C  *  END                                                         *
C  *  /*                                                          *
C  *                                                              *
C  *  ** THE CSS FILE **                                          *
C  *                                                              *
C  *  ** FILENAME: PAGE.CSS                                       *
C  *  L PAGE                                                      *
C  *  AS 1,PAGES.BUF/S,SRO                                        *
C  *  ST                                                          *
C  *  $EXIT                                                       *
C  *  /*                                                          *
C  *                                                              *
C  *--------------------------------------------------------------*
C  *                                                              *
C  *  DEFINITION OF VARIABLES:                                    *
C  *                                                              *
C  *  CLBUF(20,24): INTEGER*4 Type variable. All words of this    *
C  *                variable contain blanks and used to clear the *
C  *                screen.                                       *
```

```
C   *  NABUF(4,24) : An INTEGER*4 Type variable, contains Page Number*
C   *                ,Page Title and Date that are printed in the    *
C   *                leftmost 16 columns.                            *
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       : Staus word that contains the status bit. The    *
C   *                word is 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   *  CRTLINE     : Counts the number of CRT line that is printed.  *
C   *                                                                *
C   *----------------------------------------------------------------*
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   ******************************************************************
C
      COMMON  /GBLCOM/ BLC02, MBLIST, TYPSAV
      COMMON  /CRTPAGE/ABUF,POBUF,BITBUF,DFBUF
      INTEGER*4 ABUF(16,32), LUPAGE, PBLK(5), STATUS, RANADD
      INTEGER*4 BLC02(2,1024), TYPSAV(2,1024), TYPE, SVDAT
      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
      REAL MBLIST(2,1024), EXPF,SVDATF,TMPRES
      LOGICAL STAT
      INTEGER*4 NABUF(4,24),CLBUF(20,24),STWRD
      INTEGER*2 NWLINE(27),LINENO(10)
      CHARACTER*9 RESULT, PERCENT, SAVWRD
      CHARACTER*10  SAVFMT
```

```
      CHARACTER*3 ANS
      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
      CRTLINE = 0
C
C     ****************************************************************
C
C                          **  CLEAR SCREEN **
C
C     LOAD CLBUF WITH BLANKS AND PRINTS IT TO CLEAR SCREEN
C
C     ****************************************************************
C
      DO 10 J = 1,24
      DO 11 I = 1,20
      CLBUF(I,J) = '    '                    ; fill with blanks
 11   CONTINUE
 10   CONTINUE
      DO 12 J = 1,24
      WRITE(5,13) (CLBUF(I,J),I=1,20)        ; clear screen
 13   FORMAT(1X,20A4)
 12   CONTINUE
C
C     ****************************************************************
C
C                       ** SELECTION MENU **
C
C     IF PAGE NUMBER IS NOT 0-99 ASKS FOR VALID PAGE NUMBER. FOR PAGE
C     REDISPLAY HIT ENTER AND TO STOP THE PROGRAM ENTER 'E' OR 'EN'
C     OR 'END'.
C
C     ****************************************************************
C
      WRITE(5,*) '                 THIS PROGRAM DISPLAYS CRT PAGES 0-99'
      WRITE(5,*)
      WRITE(5,*) '              1. ENTER PAGE NUMBER FOR PAGE DISPLAY'
      WRITE(5,*) '              2. HIT ENTER TO REDISPLAY THE PAGE '
      WRITE(5,*) '              3. ENTER  E  TO END THE SESSION'
      MM = 0
      IF (MM.EQ.0) GO TO 55
 901  WRITE(5,*)
      WRITE(5,*) '                 NO PAGE NUMBER IS AVAILABLE'
      WRITE(5,*)
      GO TO 16
 55   WRITE(5,*)
      WRITE(5,*)
      WRITE(5,*) '                 ENTER A VALID PAGE NUMBER'
 16   DO 14 J = 1,10
      WRITE(5,15) (CLBUF(I,J),I=1,20)
 15   FORMAT(1X,20A4)
 14   CONTINUE
 902  READ(3,900) ANS
 900  FORMAT(A3)
      IF(MM.EQ.0 .AND. ANS.EQ.'  ') GO TO 901
      IF (ANS.EQ.'  ') GO TO 910
      IF(ANS.EQ.'ND' .OR. ANS.EQ.'EN') GO TO 81 ; end session
      IF(ANS.EQ.'E ' .OR. ANS.EQ.' E') GO TO 81
      IF(ANS.EQ.'END') GO TO 81
```

```
      PAGNO = CTOI(ANS,K)                    ; get page number
      MM = 1
 910  IF (PAGNO.LT.0 .OR. PAGNO.GT.99) GO TO 55 ; page 0-99
C
C     ****************************************************************
C
C                          ** CLEAR SCREEN **
C
C     ****************************************************************
C
      DO 702 J = 1,24
      WRITE(5,703) (CLBUF(I,J),I=1,20)    ; clear screen
 703  FORMAT(1X,20A4)
 702  CONTINUE
C
C     ****************************************************************
C
C     SYSIO SUBROUTINE PERFORMS I/O FUNCTIONS. IOERR ROUTINE REPORTS
C     SYSIO ERRORS TO THE CONSOLE.
C
C     LUPAGE - INTEGER*4 ARGUMENT THAT SPECIFIES THE LOGICAL UNIT
C              ON WHICH TO PERFORM THE I/O.
C     ABUF   - STARTING ADDRESS OF THE BUFFER USED IN THE I/O
C              TRANSFER
C     RANADD - AN INTEGER*4 ARGUMENT SPECIFYING THE LOGICAL RECORD
C              NUMBER TO BE ACCESSED ON DATA TRANSFER REQUEST
C
C     ****************************************************************
C
      RANADD = PAGNO*12
      LUPAGE = 1
      CALL SYSIO (PBLK,Y'4D',LUPAGE,ABUF,3328,RANADD)
      CALL IOERR(PBLK,STATUS)
C
C     ****************************************************************
C
C     FIRST INITIALIZE NABUF WITH ALL BLANKS AND THEN READ PAGE XX
C     FROM THE VARIABLE ABUF AND SAVE IN NABUF
C
C     ****************************************************************
C
      DO 75 J = 1,24
      DO 76 I = 1,4
      NABUF(I,J) = '    '                    ; initialize with blanks
 76   CONTINUE
 75   CONTINUE
      NC = 0
      DO 306 J = 4,27
      DO 305 I = 1,16
      IF(ABUF(I,J).NE.'    ') GO TO 307   ; test if blank
 305  CONTINUE
      GO TO 306
 307  NC = NC + 1
      NWLINE(NC) = J-3                       ; record non-blank lines
 306  CONTINUE
        IF (NC.EQ.0) GO TO 123               ; all lines blank
      NK = 1
      NN = -1                                ; byte no of ABUF
      NM = 2                                 ; 2nd word of ABUF
      DO 91 I1 = 1,16
```

```
      DO 92 I2 = 1,4
      CALL ILBYTE (LB,ABUF(I1,2),I2-1)    ; get the byte
      IF (NM.EQ.3) GO TO 93
      IF (NN.EQ.-1 .AND. LB.EQ.Y'20') GO TO 92 ; test for blank
 93   NN = NN + 1
      CALL ISBYTE (LB,NABUF(NM,NWLINE(NK)),NN)
      IF (NN.EQ.3 .AND. NM.EQ.3) GO TO 63  ; two words saved?
      IF (NN.EQ.3) THEN
          NM = NM + 1                      ; next word of NABUF
          NN = - 1                         ; initialize byte no
      ELSE
      ENDIF
 92   CONTINUE
 91   CONTINUE
C
C     ****************************************************************
C
C     THIS SECTION READS THE PAGE TITLE FROM ABUF AND SAVE IT IN
C     NABUF. AFTER 14 BYTES ARE SAVED BYTE 15 IS TESTED FOR BLANK.
C     IF BLANK GO TO NEW LINE OF NABUF, OR IF NOT BLANK SAVE A '-'
C     IN BYTE 15 AND GO TO NEW LINE. THE SAME PROCESS CONTINUES UNTIL
C     PAGE TITLE IS COMPLETE.
C
C     ****************************************************************
C
 63   LC = 0                               ; character count of NABUF
      NK = NK+2                            ; skip line after Page XX
      NM = 1
      NN = -1
      IF (I2.EQ.4) THEN                    ; if word finished,
         I2 = 1                            ; initialize byte
         I1 = I1 + 1                       ; next word of ABUF
      ELSE
         I2 = I2 + 1                       ; otherwise, next byte
      ENDIF
      DO 65 I3 = I1,16
      DO 66 I4 = I2,4
      CALL ILBYTE (LB,ABUF(I3,2),I4-1)     ; get byte
      IF (I4.EQ.4) I2 = 1
      IF (LC.GT.0) GO TO 67
      IF (LB.EQ.Y'20') GO TO 66            ; if byte blank, next line
 67   NN = NN + 1
      LC = LC + 1
      CALL ISBYTE (LB,NABUF(NM,NWLINE(NK)),NN) ; save if not blank
      IF (LC.EQ.14) GO TO 68               ; 14 bytes saved?
      IF (NN.EQ.3) THEN                    ; word done
         NN = -1                           ; initialize byte
         NM = NM + 1                       ; next word
      ELSE
      ENDIF
      GO TO 66
C
C     IF NEXT BYTE IS A BLANK, GO TO NEXT LINE, OR IF NEXT BYTE IS
C     NOT A BLANK PUT A HYPHEN AND GO TO NEXT LINE
C
 68   NN = -1
      IF (LB.EQ.Y'20') GO TO 74            ; byte 14 blank?
      IF (I4.EQ.4) THEN                    ; if byte of ABUF 3,
      CALL ILBYTE (LB,ABUF(I3+1,2),0)      ; byte 0, next word
      ELSE
```

```
      CALL ILBYTE (LB,ABUF(I3,2),I4)      ; byte not 3, next byte
      ENDIF
      IF (LB.EQ.Y'20') THEN
         CALL ISBYTE (Y'20',NABUF(NM,NWLINE(NK)),2)
      ELSE
      CALL ISBYTE (Y'2D',NABUF(NM,NWLINE(NK)),2) ; save '-'
      ENDIF
 74   LC = 0                                  ; initialize char. count
      NM = 1                                  ; initialize word
      NK = NK + 1                             ; next line
 66   CONTINUE
 65   CONTINUE
C
C     ****************************************************************
C
C     DATE IS READ FROM THE 3RD LINE OF ABUF AND SAVED IN NABUF.
C     AFTER THE FIRST NONBLANK BYTE IS ENCOUNTERED NEXT 8 BYTES
C     ARE READ AND SAVED.
C
C     ****************************************************************
C
      NK = NK + 2
      NN = -1
      NM = 2
      DO 77 I5 = 1,16
      DO 78 I6 = 1,4
      CALL ILBYTE (LB,ABUF(I5,3),I6-1)
      IF (NM.EQ.3) GO TO 79
      IF (NN.EQ.-1 .AND. LB.EQ.Y'20') GO TO 78
 79   NN = NN + 1
      CALL ISBYTE (LB,NABUF(NM,NWLINE(NK)),NN)
      IF (NN.EQ.3 .AND. NM.EQ.3) GO TO 83
      IF (NN.EQ.3) THEN
          NM = NM + 1
          NN = - 1
      ELSE
      ENDIF
 78   CONTINUE
 77   CONTINUE
C
C     ****************************************************************
C
C                    **  PAGE DISPLAY **
C
C     THIS PART OF THE PROGRAM SKIPS FIRST 3 CRT LINES. AFTER 3
C     LINES, IF IF BYTE 1 OF ABUF(1,L) IS NOT A DIGIT OR IF FIRST
C     TWO BYTES OF ABUF(1,L) ARE BLANKS NO NUMERICAL CONVERSION IS
C     REQUIRED SO THE LINE IS PRINTED AS IT IS. IF THE LINE IS BLANK
C     IT IS NOT PRINTED. OTHERWISE NUMCON SUBROUTINE IS CALLED.
C
C     ****************************************************************
C
 83   DO 100 CRTROW=1,27
      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 121
```

```
      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 121
 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
      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     ****************************************************************
C
C                    ** DETERMINE FUNCTION STATUS **
C
C     THIS SECTION OF THE PROGRAM CHECKS THE TWO FUNCTIONS OF BOTH
C     THE STATUS FIELDS FOR THE STATUS. IF THE STATUS IS A 1 IT
C     PRINTS AN '*' ADJACENT TO THE FUNCTION.
C
C     LOCATE - WORD NUMBER OF ABUF
C     LBYTE  - BYTE NUMBER OF WORD LOCATE
C
C     ****************************************************************
C
 119  DO 801 I = 1,3,2
         IF(POBUF(I,PRMROW).LE.0) GO TO 801
      STPTR = POBUF(I,PRMROW)/8 + 1       ; status word pointer
      IF(TYPSAV(1,STPTR).GE.0) GO TO 801 ; data active ?
      DO 802 J = 1,3,2
         JJ = 2*(I-1) + J
```

```
         IF (BITBUF(JJ,PRMROW).EQ.0) GO TO 802
       STWRD = BLC02(2,STPTR)                ; get status word
       BITPTR = BITBUF(JJ,PRMROW)            ; get bit pointer
          IF(BITPTR.LT.0) THEN
             BITPTR = -BITPTR
             STWRD = NOT(STWRD)
          ELSE
          ENDIF
       STAT = BTEST(STWRD,BITPTR)            ; find status bit
C
C      LOCATE THE SYMBOL FOR STATUS
C
       LOCATE = 8
       IF(STAT) THEN
         GO TO 803
       ELSE
         GO TO 802
       ENDIF
 803   IF(JJ.EQ.1) THEN                      ; status field 1, function 1
         LOCATE = 8                          ; word no 8
         LBYTE = 0                           ; byte 0
       ELSEIF (JJ.EQ.3) THEN                 ; status field 1, function 2
         LOCATE = 10
         LBYTE = 0
       ELSEIF (JJ.EQ.5) THEN                 ; status field 2, function 1
         LOCATE = 10
         LBYTE = 1
       ELSEIF(JJ.EQ.7) THEN                  ; status field 2, function 2
         LOCATE = 12
         LBYTE = 1
       ELSE
         GO TO 802
       ENDIF
       CALL ISBYTE(Y'2A',ABUF(LOCATE,CRTROW),LBYTE) ; print '*'
 802   CONTINUE
 801   CONTINUE
         IF(BFPTR.EQ.0) GO TO 118
       IF(TYPE.EQ.0) GO TO 117
       GO TO 804
C
C      ***************************************************************
C
C                    ** PRINT LINES OF CRT PAGE **
C
C      ***************************************************************
C
 121   WRITE(5,122) (NABUF(I,PRMROW),I=1,4),(ABUF(I,CRTROW),I=1,16)
 122   FORMAT(1X,20A4)
       CRTLINE = CRTLINE+1
       GO TO 100
 118   WRITE(5,96) (NABUF(I,PRMROW),I=1,4),(ABUF(I,CRTROW),I=1,3),
      1      (ABUF(I,CRTROW),I=4,6),(ABUF(I,CRTROW),I=7,16)
 96    FORMAT(1X,20A4)
       CRTLINE = CRTLINE+1
       GO TO 100
 804   WRITE(5,86) (NABUF(I,PRMROW),I=1,4),(ABUF(I,CRTROW),I=1,3),
      1       RESULT,(ABUF(I,CRTROW),I=7,16)
 86    FORMAT(1X,4A4,3A4,2X,A9,1X,10A4)
       CRTLINE = CRTLINE + 1
       GO TO 100
```

```
 117  WRITE(5,84) (NABUF(I,PRMROW),I=1,4),(ABUF(I,CRTROW),I=1,3),
     1       RESULT,(ABUF(I,CRTROW),I=7,12),PERCENT,SAVWRD
 84   FORMAT(1X,4A4,3A4,2X,A9,1X,6A4,A9,A9)
      CRTLINE = CRTLINE + 1
 100  CONTINUE
      GO TO 126
C
C     ****************************************************************
C
C     IF NUMBER OF LINES PRINTED IS LESS THAN 22 THIS SECTION WILL
C     PRINT BLANK LINES TO MAKE IT 22 SO THAT PAGE STARTS NEAR THE
C     TOP OF THE SCREEN
C
C     CRTLINE - COUNTS HOW MANY LINES HAVE BEEN PRINTED
C
C     ****************************************************************
C
 123     DO 124 J = 1,3
         WRITE(5,125) (ABUF(I,J),I=1,16)
 125     FORMAT(1X,16A4)
      CRTLINE = CRTLINE + 1
 124     CONTINUE
 126  IF(CRTLINE.GE.22) GO TO 127
      DO 128 J=CRTLINE,21
      WRITE(5,129) (CLBUF(I,J),I=1,20)
 129  FORMAT(1X,20A4)
 128  CONTINUE
 127  CRTLINE = 0
      GO TO 902
 81   STOP
      END
```

APPENDIX B

DATA TYPES AND FORMULAS

Let us assume that

Value (x) = (Value of 16 least significant bits of BLC02 (2, x))* MBLIST (1, x) + MBLIST (2, x)

The value of parameters (result) for different types are calculated as follows, and then changed to appropriate formats.

Type O:

Result = Value (pointer)

Type 1:

Exponent
(= 0 if Value (pointer) < 0.2
(= 1.07* Value (pointer) –2.0 if value (pointer) < 2.0
(= 0.398* Value (pointer) –1.64 if value (pointer) <7.4
(= 1.06* Value (pointer) –6.54 otherwise

then Result = 10 ** exponent

Type 2:

Result = 10 ** Value (pointer)

Type 3:

Result = –10 ** Value (pointer)

Type 4:

Result = Value (pointer) * 10 ** Value (pointer +1)

Type 6:

Result = BLC02 (2, pointer)

Type 7:

Result = 2 * (value of the complement of 24 least significant bits of BLC02 (2, pointer))

Type 8:

Result = Value of the 4 least significant bits of BLC02 (2, pointer)

Type 10:

Result = Value (pointer)

Type 18:

Result = (Value of 12 least significant bits of BLC02 (2, pointer)) * MBLIST (1, pointer)

Type 19:

Result = BLC02 (2, pointer)

For negative pointer:

If pointer = –8

Result = Value (127) – Value (142) – Value (143) – Value (144)

If pointer = –9

Result = Value (214) + Value (215) – Value (220)
– Value (216) – Value (217) – Value (434)

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. C. M. Jones
11. R. C. Juras
12. M. J. Meigs
13-15. S. W. Mosko
16. D. K. Olsen
17. B. A. Tatum

EXTERNAL DISTRIBUTION

18. Dr. A. M. Hasanul Basher, School of Engineering Technology, South Carolina State College, Orangeburg, SC 29117
19. J. T. Crockett, Jr., Director, Faculty and HBCU Programs, ORAU, Energy Bldg., Room OB-9, Oak Ridge, Tennessee 37831
20. Assistant Manager, Energy Research and Development, DOE-OR
21-22. Office of Scientific and Technical Information, P. O. Box 62, Oak Ridge, TN 37831

END

DATE
FILMED
01/22/92