Worldwide Response Center

HP 3000 APPLICATION NOTE #50

VFC's For Serial Printers

HEWLETT PACKARD

December 15, 1988
Document P/N #5959-9228

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

RESPONSE CENTER APPLICATION NOTES

HP 3000 APPLICATION NOTES *are published by the Worldwide Response Center twice a month and are distributed with the Software Status Bulletin. These notes address topics where the volume of calls received at the Center indicates a need for addition to or consolidation of information available through HP support services.*

Following this publication you will find a list of previously published notes and a Reader Comment Sheet. You may use the Reader Comment Sheet to comment on the note, suggest improvements or future topics, or to order back issues. We encourage you to return this form; we'd like to hear from you.

NOTICE

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

This document contains proprietary information which is protected copyright. All rights are reserved. Permission to copy all or part of this document is granted provided that the copies are not made or distributed for direct commercial advantage; that this copyright notice, and the title of the publication and its date appear; and that notice is given that copying is by permission of Hewlett-Packard Company. To copy otherwise, or to republish, requires prior written consent of Hewlett-Packard Company.

Copyright © 1988 by HEWLETT-PACKARD COMPANY

VFC's for Serial Printers

Programmable Vertical Forms Control (VFC) using PCL (Printer Control Language) seems to cause a lot of problems and misunderstandings. Most of the misunderstandings concern the use of the escape sequence which sets the VFC file.

Serial Interface printers are controlled by escape sequences. The drivers used for serial printers are designed to be very generic because of the large number and variety of serial printers which are available. Escape sequences may be used to override the default VFC of the printer.

The format of the escape sequence is

ESC&l##Wx....x

where

ESC	is the escape character or %33;
&l	is the PCL character used to start a page layout command. NOTE: l is a lower case L;
##	is the number of bytes in the VFC data string; This is twice the number of lines on the printed form;
W	is the terminating character;
x....x	is the actual VFC data string which contains 2 characters for each line in the VFC. The 2 characters are translated from a binary bit map of the VFC data.

To illustrate:

(Form length = 18 lines) * 2 = 36 (Byte count)

The escape sequence can be hard-coded into a program or EDITOR file, written to a spoolfile and merged with a data spoolfile using the SPOOK utility, or placed in the VFC and Initialization Menu of Workstation Configurator, an HP3000 software product. Please see the *256X Printer Family Technical Reference Manual (HP part number 02564-90905.)*

Escape sequences cannot be placed within an HP3000 HP-IB VFC download file. The HP-IB interface uses a different type of VFC file. More information on HP-IB VFC files may be found in section 8 of the *System Operation and Resource Management Manual (HP part number 32033-90005)* or *Application Note #23 VFC Files (HP part number 5958-5824R2709)*

CONVERT VFC DATA TO ESCAPE SEQUENCE FORMAT

VFC's are best understood by reviewing an example of how to implement them. For this discussion, the example consists of a three inch form at six lines per inch with several different channels defined for use on the form to be printed.

1 inch = 6 lines per inch X 3 inches = 18 lines; therefore this would be an 18 line form.

EXAMPLE FORM

```
------------------------------------------------------
line 1    Heading line 1
line 2    Heading line 2
line 3
line 4
line 5
line 6
line 7
line 8    First print line
line 9    Second print line
line 10
line 11   Third print line
line 12   Fourth print line
line 13
line 14
line 15   Fifth print line
line 16
line 17
line 18   Sixth print line
------------------------------------------------------
```

To visualize what this might look like on a carriage control tape, or using a VFC file (HP-IB), the following illustration is provided.

Channel

```
          0         1
          1234567890123456
line 1    1 1
line 2      1
line 3      1
line 4      1
line 5      1
line 6      1
line 7      1
line 8      1           1
line 9      11
line 10     1
line 11     11
line 12     1 1
line 13     1
line 14     1
line 15     1 1
line 16     1
line 17     1
line 18     111
```

NOTE

A "1" corresponds to a hole punched in the physical paper tape.

It is important to remember that a VFC is simply a template of where the print lines are. If the template is not used, or a wrong channel is chosen, the VFC will allow the detour. The program or print file is what controls where each line will print.

Notice that for each line channel 3 is selected; this allows a single space advance. A 1 could be placed in each of the columns and would still be valid since the program will only look at the column selected and advance to the next line that contained a 1 in the column.

It is recommended that a "punch" (or a 1) be placed in channel 3 for all lines. This avoids slews when a channel is inadvertently accessed. In the example above, if the printer was on line 2 and channel 14 was selected, the printer would advance or slew down to line 8. This is the first line after line 2 where channel 14 is "punched".

The next step would be to convert the above VFC definition into the escape sequence format. The process gets tricky at this point. The VFC data is formatted for the escape sequence in reverse order for each line in the VFC. This means that a "punch" in channel 1 is included in the escape sequence after a "punch" in channel 16. The following example should help to illustrate.

```
          0         1
          12345678 90123456

line 1    1 1                          = %240  %000

This line would convert to the following

                  1         0
           65432109 87654321

line 1                    1 1          = %000  %005
```

The format of the print line is reversed. This bit pattern can then be converted to either an OCTAL, or an ASCII digit. ASCII conversion is accomplished by using the terminal CONTROL key which subtracts %100 from the character entered. If a "control A" is entered, (upper case A = %101) it will convert to a %001 or a Start of Header character.

Remember, the 16 channels are treated as 2 separate bytes of 8 bits each. Converting to octal requires the addition of an extra, high order zero on each byte.

Following the above example, this is illustrated as:

	Most Sig Byte								Least Sig Byte											
	msb															lsb				
	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1		OCTAL	ASCII	CTL
line 1								1						1		1	=	%001 005	SohEnq	AE
line 2								1						1			=	%001 004	SohEot	AD
line 3								1						1			=	%001 004	SohEot	AD
line 4								1						1			=	%001 004	SohEot	AD
line 5								1						1			=	%001 004	SohEot	AD
line 6								1						1			=	%001 004	SohEot	AD
line 7								1						1			=	%001 004	SohEot	AD
line 8			1					1						1			=	%021 004	DC1Eot	QD
line 9								1					1	1			=	%001 014	SohFF	AL
line 10								1						1			=	%001 004	SohEot	AD
line 11								1					1	1			=	%001 014	SohFF	AL
line 12								1				1		1			=	%001 024	SohDC4	AT
line 13								1						1			=	%001 004	SohEot	AD
line 14								1						1			=	%001 004	SohEot	AD
line 15								1				1		1			=	%001 024	SohDC4	AT
line 16								1						1			=	%001 004	SohEot	AD
line 17								1						1			=	%001 004	SohEot	AD
line 18								1				1		1		1	=	%001 026	SohSyn	AV

NOTE

A "1" is punched in each channel 9 because an %000 or a NUL cannot be sent as data.

Only one byte may be specified at a time in OCTAL, thus allowing a maximum of %377 which would place a 1 in columns 1 - 8 or 9 -16.

This VFC data can now be put into the escape sequence in an EDITOR file. This is done with the CHANGE command, or by using the CTL key with the appropriate letter.

CREATE THE ESCAPE SEQUENCE

The VFC file can be created using EDITOR or Workstation Configurator. There are four ways to use EDITOR to create the VFC file. When using EDITOR, keep in mind that the default record length is 80 and the VFC file may exceed this. Extend your record length if necessary.

Using EDITOR to Create VFC Files

Method 1: In EDITOR, there is no way to enter OCTAL numbers. Only ASCII characters may be entered. This can be accomplished in several ways. The first method involves placing extra characters in the columns you wish to convert to ASCII characters and using the EDITOR "CHANGE" command to convert them. For example, replace an "x" with an ASCII 01, or the SOH character by typing "CHANGEQ "x" TO '01".

```
/ADD 1
 1 ESC&l36Wabcdefghijkmnopqrstuvwxyz ABCDEFGHIJK
 2 //
/CHANGEQ "a" TO '01
/CHANGEQ "b" TO '05
/CHANGEQ "c" TO '01
          .
          .
          .
/CHANGEQ "K" TO '22
```

NOTE

The " ' " is the single quote and not the "prime" character (′). Additionally only one byte can be changed at a time.

Method 2: This could also be accomplished by changing the specific column to the ASCII character required. It is important to specify the starting and stopping column or the CHANGE command will act as a column INSERT. For example, if column 7 is changed to an ASCII 01 you would type "CHANGEQ 7/7 to '01".

```
/ADD 1
 1 ESC&l36Wabcdefghijklmnopqrstuvwxyz ABCDEFGHIJ
 2 //
/CHANGEQ 7/7 TO '01
/CHANGEQ 8/8 TO '05
/CHANGEQ 9/9 TO '01
          .
          .
          .
/CHANGEQ 43/43 TO '22
```

Method 3: This could also be input by not entering the letters, but simply changing the specific column to the desired ASCII character. In the above example the "abc...HIJ" would not be entered and the CHANGE command would be used as follows:

```
\CHANGEQ 7 to '01
\CHANGEQ 8 to '02
        .
        .
        .
```

Note: The *HP3000 EDITOR Manual HP (part number 03000-90012)* discusses the CHANGE command on page 3-2 referencing STRING data in in paragraph 3-6.

Method 4: Another method used to enter ASCII characters in EDITOR is to place the terminal in DISPLAY FUNCTIONS mode, type in the "ESC&lbyte countW" followed by the ASCII characters using the control key with the corresponding letter as mentioned above.

Control key has been pressed for each character past the "W"

ESC&l36WAEADADADADADADQDALADALATADADATADADAV

This would display the corresponding ASCII character, i.e., the SOH for CTL A.

Using Workstation Configurator to Create VFC Files

In Workstation Configurator, the VFC data in the Initialization Menu may be entered in several different ways. These are as follows:

1) Entering data as a decimal number, i.e. 13 for a carriage return.

2) As an OCTAL number (one byte only) by preceding the number with the "%" sign.

3) As a two or three character mnemonic such as BS or DC1 see Appendix B of the *Workstation Configurator Referencing Manual (HP part number 30239-90001).*

4) By entering the control characters with the up-arrow or circumflex character preceding the character, i.e. a backspace would be a ^H.

5) By entering the actual ASCII character within single quotes, i.e. 'A' would equate to an OCTAL 101.

Each character entered, with the exception of multiple ASCII characters within the single quote, MUST BE separated by commas. An example would be:

ESC,'&16W',%101,'ABC',BS,13

Workstation Configurator could also be used for this example but the second field of the Initialization Menu has a maximum of 120 characters. Since it is easy to create a VFC file that exceeds 120 characters, the user would be forced to hard code the escape sequence into the application or append it to to the data file with EDITOR. There are some other aspects about Workstation Configurator discussed below.

An example in Workstation Configurator may be input several ways. Following is an example of one way:

ESC,'&136W',SOH,EOT,^A,^D,^A,01,04,%2,%4,....%1,%26

The VFC initialization strings input in the Workstation Configurator may be sent to the printer in one of two ways. Either the Term Type file with the associated VFC file may be configured directly to the printer in the I/O configuration on the HP3000, or by using the ENV= parameter of the file equation with the Term Type file specified. It should be noted that if the ENV= parameter is used, the associated Term Type file that has been configured with the printer will still be sent to the printer first and the Term Type file specified with the ENV= parameter will be sent next. This should not be a problem unless the user is expecting to see only the escape sequences that were specified in the Term Type file specified with the ENV= parameter.

More detail is included in the *Workstation Configurator Reference Manual (HP part number 30239-90001).* In particular pages 2-4, 2-5, and 13-12 thru 13-14 help clarify VFC and data entry.

SPECIAL CONSIDERATIONS

2235 - Rugged Writer does not support VFC's

Vertical Forms Control is not handled the same on all HP printers. If a process works on one printer, do not assume it will work the same on all printers.

The HP2235 or Rugged Writer was designed to work with Personal Computers. It does not have the same features as the HP2934. It will not recognize VFC controls.

BACK ISSUE INFORMATION

Following is a list of the Application Notes published to date. If you would like to order single copies of back issues please use the *Reader Comment Sheet* attached and indicate the number(s) of the note(s) you need.

Note #	*Published*	*Topic*
1	2/21/85	*Printer Configuration Guide (superseded by note #4)*
2	10/15/85	*Terminal types for HP 3000 HPIB Computers (superseded by note #13)*
3	4/01/86	*Plotter Configuration Guide*
4	4/15/86	*Printer Configuration Guide - Revised*
5	5/01/86	*MPE System Logfile Record Formats*
6	5/15/86	*Stack Operation*
7	6/01/86	*COBOL II/3000 Programs: Tracing Illegal Data*
8	6/15/86	*KSAM Topics: COBOL's Index I/O; File Data Integrity*
9	7/01/86	*Port Failures, Terminal Hangs, TERMDSM*
10	7/15/86	*Serial Printers - Configuration, Cabling, Muxes*
11	8/01/86	*System Configuration or System Table Related Errors*
12	8/15/86	*Pascal/3000 - Using Dynamic Variables*
13	9/01/86	*Terminal Types for HP 3000 HPIB Computers - Revised*
14	9/15/86	*Laser Printers - A Software and Hardware Overview*
15	10/01/86	*FORTRAN Language Considerations - A Guide to Common Problems*
16	10/15/86	*IMAGE: Updating to TurboIMAGE & Improving Data Base Loads*
17	11/01/86	*Optimizing VPLUS Utilization*
18	11/15/86	*The Case of the Suspect Track for 792X Disc Drives*
19	12/01/86	*Stack Overflows: Causes & Cures for COBOL II Programs*
20	1/01/87	*Output Spooling*
21	1/15/87	*COBOLII and MPE Intrinsics*
22	2/15/87	*Asynchronous Modems*
23	3/01/87	*VFC Files*
24	3/15/87	*Private Volumes*
25	4/01/87	*TurboIMAGE: Transaction Logging*
26	4/15/87	*HP 2680A, 2688A Error Trailers*
27	5/01/87	*HPTrend: An Installation and Problem Solving Guide*
28	5/15/87	*The Startup State Configurator*
29	6/01/87	*A Programmer's Guide to VPLUS/3000*
30	6/15/87	*Disc Cache*
31	7/01/87	*Calling the CREATEPROCESS Intrinsic*
32	7/15/87	*Configuring Terminal Buffers*
33	8/15/87	*Printer Configuration Guide*
34	9/01/87	*RIN Management (Using COBOLII Examples) (A)*
34	10/01/87	*Process Handling (Using COBOLII Examples) (B)*
35	10/15/87	*HPDESK IV (Script files, FSC, and Installation Considerations)*
34	11/01/87	*Extra Data Segments (Using COBOLII Examples) (C)*
36	12/01/87	*Tips for the DESK IV Administrators*
37	12/15/87	*AUTOINST: Trouble-free Updates*
38	1/01/88	*Store/Restore Errors*
39	1/15/88	*MRJE Emulates a HASP Workstation*
40	2/01/88	*HP 250 / 260 to HP 3000 Communications Guidelines*
41	4/01/88	*MPE File Label Revealed - Revised 6/15/88*

42	7/15/88	*System Interrupts*
43	7/15/88	*Run Time Aborts*
44	8/01/88	*HPPA Pathing Conventions For HP3000 900 Series Processors*
45	8/15/88	*Vplus & Multiplexers*
46	8/15/88	*Setting Up An HPDesk/HPTelex For The First Time*
47	9/15/88	*Customizing Database Data Items & Changing Passwords in JCL Files*
48	11/15/88	*Printer Configuration (Revision #4)*
49	12/01/88	*Configuring DATACOMM Products Into MPE*
50	12/15/88	*VFC's For Serial Printers*

READER COMMENT SHEET

World Response Center Supports
HP 3000 Application Note #50: VFCs For Serial Printers
RC Questions & Answers (December 15, 1988)

We welcome your evaluation of this Application Note and attached RC Questions & Answers Sheet. Your comments and suggestions help us to improve our publications. Please explain your answers under Comments, below.

	AppNote	RC Q&A
Is this publication applicable to your site?	☐ Yes ☐ No	☐ Yes ☐ No
Are the concepts and wording easy to understand?	☐ Yes ☐ No	☐ Yes ☐ No
Would you like to see additional Notes on this subject?	☐ Yes ☐ No	☐ Yes ☐ No

Back Orders/Comments/Suggestions for future Application Notes:

This form requires no postage stamp if mailed in the U.S. For locations outside the U.S., your local HP representative will ensure that your comments are forwarded.

FROM: Date _____

Name _____

Company _____

Address _____

_____

_____

FOLD FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 95, MT. VIEW, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Application Note / RC Q & A Comments
Hewlett-Packard North American Response Center
100 Mayfield Avenue
Mail Stop 37MA
Mt. View, CA 94043

FOLD FOLD

HEWLETT PACKARD RESPONSE CENTER QUESTIONS & ANSWERS

HP 3000 Questions Commonly Received by the North American Response Centers

Q. I have received several notifications of INP RAM Dumps and INP LOGn files. Is my INP broken?

A. Just because you have received an INP RAM Dump does not mean your INP is broken. These dumps can be caused by many things. To determine if the problem is hardware, try the following:

```
:FILE INPDUMP=INPLOGnn.PUB.SYS        <<THE LOG FILE>>
:FILE INPLIST;DEV=LP,1                <<DEFER THE OUTPUT LISTING>>
:RUN INPDPAN.PUB.SYS,FULLDUMP         <<FORMAT THE INPRAMDUMP>>
:RUN SPOOK5.PUB.SYS                   <<GO LOOK AT DUMP>>
>SHOW                                 <<GET FILE NUMBER>>
#ONNN xxxxxxxxxxxxxxxxxxx             <<OUTPUT FROM SHOW CMD>>
>TEXT NNN                             <<TEXT IN DUMP>>
>FIND@"LAST CS ERROR"
     this will print your last cs error code
>LIST 140/LAST                        <<GET FAILURE CODE>>
     this will print the FAILURE CODE
```

If FAILURE CODE is non-zero (ie:0004) you had a software, not a hardware interrupt (ie:noise on the line). You should check your modem(s), cable(s) or have your line checked.

If FAILURE CODE is zero (ie:0000), additional tests are needed to determine the cause of the failure. Contact Hewlett-Packard if you require assistance.

* * * * *

Q. How can I pass character and numeric data from my Cobol (74 or 85) main program to a FORTRAN (77) subprogram?

A. If you are using FORTRAN (77) version A.00.09, all that is necessary is to specify the directive " $FTN3000_66 ". Example A demonstrates the usage of $FTN3000_66.

For all other versions of FORTRAN (77) you must pass the length of the character field in the Cobol program. Please refer to Example B.

more...

NOTE

If you fail to perform either of these steps, you will probably receive the following message when you try to PREP the program.

```
*** ERROR *** Subpgm-Name,MAINCONTROL00'
ERROR #50 INCOMPATIBLE NUMBER OR PARAMETERS
PREP FAILED DUE TO SEGMENTER ERROR. (CIERR 621)
```

EXAMPLE "A"

```
IDENTIFICATION DIVISION.
PROGRAM-ID. COBFTN.
AUTHOR.  HP RESPONSE CENTER.
*THIS COBOL PROGRAM CALLS A FORTRAN 77 SUBPROGRAM PASSING
*4 VARIABLES (2 INTEGER & 2 CHARACTER).
*THE KEY TO THIS WORKING IS TO SPECIFY THE LENGTH ON THE
*CHARACTER FIELDS.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  HP3000.
OBJECT-COMPUTER.  HP3000.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  INT1                          PIC S9(4) COMP VALUE 0.
01  INT2                          PIC S9(4) COMP VALUE 1.
01  CHAR1                         PIC X(10) VALUE SPACES.
01  CHAR2                         PIC X(10) VALUE SPACES.
PROCEDURE DIVISION.
MAIN-CONTROL.
     DISPLAY "MAIN-PROGRAM  ".
     MOVE ALL "A" TO CHAR1.
     MOVE ALL "B" TO CHAR2.
     MOVE 1         TO INT1,
                       INT2.
     CALL "FTNSUB" USING @CHAR1, @CHAR2, INT1, INT2.

     STOP RUN.

0 ERRORS, 0 QUESTIONABLE, 0 WARNINGS

    DATA AREA IS %000213 WORDS.
    CPU TIME = 0:00:03.  WALL TIME = 0:00:05.
```

more...

EXAMPLE "A"

```
$FTN3000_66

C      FORTRAN 77 subprogram called by Cobol Main program
C      passing 4 variables.
C      (2 integer & 2 character data types)
C
       Subroutine FTNSUB (HCHAR0,HCHAR1,HINT0,HINT1)
C
C      HINT's are Integer variables
C      HCHAR's are Character variables (10 char's long)
C
       INTEGER HINT0, HINT1
       CHARACTER*10 HCHAR0,HCHAR1

       WRITE (6,*) HCHAR0
       WRITE (6,*) HCHAR1

   10  CONTINUE
C
       WRITE (6,*) HINT0
       WRITE (6,*) HINT1
C
       Return
       End

NUMBER OF ERRORS =     0   NUMBER OF WARNINGS =    0
PROCESSOR TIME   0: 0: 3   ELAPSED TIME      0: 0:13
NUMBER OF LINES =     22
```

more...

EXAMPLE "B"

```
IDENTIFICATION DIVISION.
PROGRAM-ID. COBFTN.
AUTHOR.  HP RESPONSE CENTER.
*THIS COBOL PROGRAM CALLS A FORTRAN 77 SUBPROGRAM PASSING
*4 VARIABLES (2 INTEGER & 2 CHARACTER).
*THE KEY TO THIS WORKING IS TO SPECIFY THE LENGTH ON THE
*CHARACTER FIELDS.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  HP3000.
OBJECT-COMPUTER.  HP3000.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  INT1                          PIC S9(4) COMP VALUE 0.
01  INT2                          PIC S9(4) COMP VALUE 1.
01  CHAR1                         PIC X(10) VALUE SPACES.
01  CHAR2                         PIC X(10) VALUE SPACES.
PROCEDURE DIVISION.
MAIN-CONTROL.
     DISPLAY "MAIN-PROGRAM  ".
     MOVE ALL "A" TO CHAR1.
     MOVE ALL "B" TO CHAR2.
     MOVE 1         TO INT1,
                       INT2.
     CALL "FTNSUB" USING @CHAR1,/10/, @CHAR2,/10/, INT1, INT2.

     STOP RUN.

0 ERRORS, 0 QUESTIONABLE, 0 WARNINGS

    DATA AREA IS %000213 WORDS.
    CPU TIME = 0:00:03.  WALL TIME = 0:00:05.
```

more...

EXAMPLE "B"

```
C     FORTRAN 77 subprogram called by Cobol Main program
C     passing 4 variables.
C     (2 integer & 2 character data types)
C
      Subroutine FTNSUB (HCHAR0,HCHAR1,HINT0,HINT1)
C
C     HINT's are Integer variables
C     HCHAR's are Character variables (10 char's long)
C
      INTEGER HINT0, HINT1
      CHARACTER*10 HCHAR0,HCHAR1

      WRITE (6,*) HCHAR0
      WRITE (6,*) HCHAR1

   10 CONTINUE
C
      WRITE (6,*) HINT0
      WRITE (6,*) HINT1
C
      Return
      End

NUMBER OF ERRORS =     0   NUMBER OF WARNINGS =    0
PROCESSOR TIME   0: 0: 3   ELAPSED TIME      0: 0:13
NUMBER OF LINES =     22
```

****

Q. When I use FORTRAN77 to read a file, how do I prevent the MODIFY DATE from being changed?

A. FORTRAN 77's file open defaults to input/output access. To avoid changing the MODIFY DATE, issue a file equation specifying ACC=IN before opening the file In FORTRAN 77 the OPEN verb will open the specified file or the READ will open the file which is equated to FTNnn where nn is the FORTRAN unit number in the READ statement. Tested on Version A.00.10.

more...

Q. How do I get output from a Pascal subroutine when called from COBOL?

A. Pascal's predefined file for $STDLIST is OUTPUT. The problem is that $STDLIST is already opened by the COBOL main. To use this file for output a Pascal REWRITE needs to be done against another file of text declared locally to the procedure. When a rewrite associates outfile with $STDLIST, writes to outfile will output to the screen.

```
$Subprogram$
PROGRAM DUMMY_OUTER_BLOCK(INPUT,OUTPUT);
PROCEDURE PASWRITE;
VAR OUTFILE : text;
BEGIN
  REWRITE (OUTFILE,'$STDLIST');
  WRITELN(OUTFILE,'THIS IS A LINE');
END;
BEGIN
END.
```

This routine is then compiled into the same USL file as the COBOLII main and is called via CALL "PASWRITE".

****

Q. How do I clear the screen of my HP terminal using COBOLII?

A. A simple escape sequence can be used to home the cursor then clear the screen.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. CLEAR-THE-SCREEN.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 CLEAR-SCREEN.
   05 ESC1      PIC X(1) VALUE %33.
   05 HOME      PIC X(1) VALUE "h".
   05 ESC2      PIC X(1) VALUE %33.
   05 CLEAR-S   PIC X(1) VALUE "J".
PROCEDURE DIVISION.
100-CLEAR-ROUTINE.
   DISPLAY CLEAR-SCREEN.
   STOP RUN.
```

This program fragment can be also be used as a subprogram with the addition of $CONTROL DYNAMIC and a GO BACK statement instead of a STOP RUN.