Worldwide Response Center

# HP 3000 APPLICATION NOTE #53

# Using Special Characters on 700/9x Series Terminals

HEWLETT PACKARD

February 1, 1989
Document P/N #5959-9245

# 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.*

# Using Special Characters on 700/9X Series Terminals

Special applications often require special characters which are not contained in the standard character sets (USASCII and Roman Extension Set). These are, for example, Greek letters or Roman numerals as well as mathematical special characters. The terminals of the series 700/92 and 700/94 offer the possibility of creating new customized characters and downloading them into the Terminal RAM. This Application Note demonstrates how relatively easy it is to create and make use of a new customized character. Two programs are included to illustrate the use of this feature.

## Defining the New Character and Downloading it into the Terminal RAM

Both "HP" and "EM220" modes offer the possibility of defining a maximum of 94 new characters by downloading character sets into the Terminal RAM. However, the definition, as well as the use, of these character sets is different for each mode. In contrast to the three ROM-resident character sets (USASCII, Roman Extension and Line Drawing Set), the downloaded characters get lost after every "POWER ON" or "HARD RESET" as well as when changing from one mode (HP and EM220) to the other (they are only stored in the Terminal RAM).

Both the Line Drawing Set and the customized, downloadable character set, can be redefined with only a minimum of effort. The standard width and height of a character cell is 9 X 14 bits. However, only a width of 8 bits can be used.

## Definition of a character

```
         --> character cell of 8 X 14 bits
         |
         |
    1 2 3 4 5 6 7 8  9  bit pattern  byte pattern** CHAR.-DOWNLOADED***
 1  . . . . . . . .  .  0000 0000X   30H 30H             00
 2  . . . . . . . .  .  0000 0000X   30H 30H             00
 3  . * * * * * * *  .  0111 1111X   37H 3FH             7?
 4  . * * * * * * *  .  0111 1111X   37H 3FH             7?
 5  . . * . . . . *  .  0010 0001X   32H 32H             22
 6  . . . * . . . .  .  0001 0000X   31H 30H             10
 7  . . . . * . . .  .  0000 1000X   30H 38H             08
 8  . . . . . * . .  .  0000 0100X   30H 34H             04
 9  . . . . * . . .  .  0000 1000X   30H 38H             08
10  . . . * . . . .  .  0001 0000X   31H 30H             10
11  . . * . . . . *  .  0010 0001X   32H 31H             21
12  . * * * * * * *  .  0111 1111X   37H 3FH             7?
13  . * * * * * * *  .  0111 1111X   37H 3FH             7?
14  . . . . . . . .  .  0000 0000X   30H 30H             00
    |                |
    |                |-> This column cannot be used for the
    |                    declaration of a character.
    |--> halfshift bit
         The leftmost bit in a scan line is used to select a
         halfshift for the subsequent dots in that line during the
         text treatment mode, which allows the characters to be
         defined with more precision.
```

** will be calculated as:
0111 --> 7h + 30h(const) = 37h

*** you obtain this value following the conversion of the hexa-decimal "Byte Pattern" into the correspondant graphics values.

**What does the ESCAPE sequence look like?**

```
ESC *y <w>w <h>h <d>d <t>t <e>e <ls>l <cc>n !<cd>!<cd>!<cd>!.....Z


<w>  = character cell width, default = 8
<h>  = character cell heigth, default = 14
<d>  = destination character set
        3 = Line Drawing Set
        4 = Downloadable Character Set (default)
<t>  = character set treatment
        0 = text treatment (default): allow halfshift
        1 = Line Drawing: extend definition to end of cell so as to
                              connect adjacent cells
<e>  = delete control
        0 = delete only the loaded characters (default)
        1 = delete all characters in the Downloadable Character Set
<ls> = character set to be loaded: pre-load of existing character
                                      set into the Downloadable Set:
        0 = no pre-load is done (default)
        1 = USASCII
        2 = Roman Extension
        3 = Line Drawing
<cc> = character code: initial character cell to  be loaded.
        33<=Start<=126
|    = seperates the individual character definitions
<cd> = 28 bytes for character definition;
        the escape sequence is terminated with a "Z"

ESC *y3d1t1e33n!00007?7?7?221008040810217?7?7?00Z
```

**Model Program to Redefine a Character in the Line Drawing Set**

```
$COPYRIGHT_DATE'27/10/1988'$
PROGRAM DOWNLOAD (OUTPUT);
(******************************************
*   This program will download a special  *
*   customized   character  and show how  *
*   you can activate this character.      *
******************************************)
var
   esc : char;   {escape    variable}
   so  : char;   {shift_out variable}
```

```
procedure download_char;
{ downloading a self-defined character }
{ in this case a Greek 'Sigma', also used as a math 'total' symbol }

{   CHARACTER CELL     BIT PATTERN  BYTE PATTERN     DOWNLOADED }
{ 1 . . . . . . . . .   0000 0000X   30H 30H               00    }
{ 2 . . . . . . . . .   0000 0000X   30H 30H               00    }
{ 3 . * * * * * * * .   0111 1111X   37H 3FH               7?    }
{ 4 . * * * * * * * .   0111 1111X   37H 3FH               7?    }
{ 5 . . * . . . . * .   0010 0001X   32H 32H               22    }
{ 6 . . . * . . . . .   0001 0000X   31H 30H               10    }
{ 7 . . . . * . . . .   0000 1000X   30H 38H               08    }
{ 8 . . . . . * . . .   0000 0100X   30H 34H               04    }
{ 9 . . . . * . . . .   0000 1000X   30H 38H               08    }
{10 . . . * . . . . .   0001 0000X   31H 30H               10    }
{11 . . * . . . . * .   0010 0001X   32H 31H               21    }
{12 . * * * * * * * .   0111 1111X   37H 3FH               7?    }
{13 . * * * * * * * .   0111 1111X   37H 3FH               7?    }
{14 . . . . . . . . .   0000 0000X   30H 30H               00    }

begin
  writeln(esc,'*y8w14h4d1t1e2165n!00007?7?221008040810217?7?00Z');
end;


procedure activate_new_charset;
begin
  writeln(esc,')X');
end;

begin  {MAIN}
 esc:=chr(27);
 so :=chr(14);
 writeln('THIS DEMO PROGRAM WILL DOWNLOAD A SPECIAL SELFDESIGNED');
 writeln('              C H A R A C T E R   S E T                ');
 writeln('       AND SHOW HOW TO WORK WITH THIS FEATURE          ');
 writeln('                                                       ');
 writeln('                    /_ ___                             ');
 writeln('                   / 7 /_7                             ');
 writeln('                      /                                ');
 download_char;
 activate_new_charset;
 write('      This is now the new character :');
 write(so,'A');
 end.
```

## Example for the Declaration of a Company Logotype and the Use of Characters Newly Declared under VPLUS

```
$COPYRIGHT_DATE'27/10/1988'$
PROGRAM DOWNLOAD (OUTPUT);
(*******************************************
*   This program will download a special   *
*   customized  character  set and show    *
*   how to activate it.                    *
*******************************************)
var
   ESC : CHAR;
   SO  : CHAR;
   CR  : CHAR;

procedure download_chars;
{ downloading self-defined charset }
type mstr = string[29];
var buffer : string[1094];
    esc1   : string[1];
    escseq : string[20];
(*********************************************************)
(*  VARIABLE DECLARATION FOR THE DOWNLOADED CHARACTERS  *)
(*********************************************************)
    c1 ,c2 ,c3 ,c4 ,c5 ,c6 ,c7 ,c8 ,c9 ,c10  : mstr;
    c11,c12,c13,c14,c15,c16,c17,c18,c19,c20  : mstr;
    c21,c22,c23,c24,c25,c26,c27,c28,c29,c30  : mstr;
    c31,c32,c33,c34,c35,c36,c37              : mstr;

(*********************************************************)
(*   INTRINSIC - DECLARATION                             *)
(*********************************************************)
  procedure print; INTRINSIC;

begin
    escseq:='*y8w14h4d1t1e0148n!';
    c1 :='070?1?3?7?????????????????!';
    c2 :='????????????????????????????!';
    c3 :='?????????????<?8?8?0?0>0>0>0!';
    c4 :='?????<?080000000000000000000!';
    c5 :='?=<00303030707070?0?0?0?1?1?!';
    c6 :='?<?8?8?8?8?0?0?0>0>0>0>0<0<0!';
    c7 :='3?07000000000000000000000000!';
    c8 :='??????1?03010000000000000000!';
    c9 :='????????????7?3?070703030101!';
    c10:='>0?0?8?<?>??????????????????!';
    c11:='????????????????????????????!';
    c12:='<0<0808080808000000000000000!';
    c13:='0000000000000000000000000000!';
    c14:='1?1?3?3?3?3?7?7>7>7>?<?<?<?8!';
    c15:='<0<0<0??????????07070?0?0?0?!';
    c16:='000000<3>3>3>3<7<7<7<78?8?8?!';
    c17:='000000??????<0<0<0<0<0808080!';
    c18:='000000?<?>?>?>?<?<?<?<?8?8?8!';
    c19:='????7?7?7?3?3?3?1?1?1?1?1?!';
    c20:='????????????????????????????!';
    c21:='00000000000080808080<0<0<0<0!';
    c22:='0101010103030303070707000000!';
    c23:='?8?8?8?0?0?0?0>0>0>0>1000000!';
    c24:='3?3?3?7>7>7>7>7<?<?<?<000000!';
    c25:='0?0?1?1?1?1?3?3?3?3?7<7<7<7<!';
    c26:='010103030303????????00000000!';
    c27:='?8?0?0?0?0>0>0>0>0<000000000!';
    c28:='1?1?1?1?1?1?1?1?3?3?3?7?7?7?!';
    c29:='??????????????????7?3?1?0?07!';
    c30:='>0>0?0?8?8?<?<??????????????!';
    c31:='0000000000000000<0?0?<??????!';
    c32:='000000000000000000000000?0?>!';
    c33:='0000000001010101030303030707!';
    c34:='?8?8?8?8?8?0?0?0?0>0>0>0>0<0!';
    c35:='000000000000000000071???????!';
    c36:='01010303031?1???????????????!';
    c37:='???????????????????>?<?8?0>0Z';

    ESC1:=#27;
    buffer:=esc1+escseq+c1+c2+c3+c4+c5+c6+c7+c8+c9+c10;
    buffer:=buffer+c11+c12+c13+c14+c15+c16+c17+c18+c19+c20;
    buffer:=buffer+c21+c22+c23+c24+c25+c26+c27+c28+c29+c30;
    buffer:=buffer+c31+c32+c33+c34+c35+c36+c37;
    print(buffer,-1094,208);
end;


procedure first_line;
 begin
   WRITE(ESC,'&a26c22Y');
   WRITE(SO,'0','1','1','2','3','4','5',' ','6','7','8','1','1');
   WRITELN(SO,'9');
 end;


procedure second_line;
 begin;
   WRITE(ESC,'&a26c23Y');
   WRITE(SO,'1','1',':',';','<','=','>','?','@','A',' ','B','1');
   WRITELN(SO,'1');
 end;


procedure third_line;
 begin
   WRITE(ESC,'&a26c24Y');
   WRITE(SO,'1','1','C','D','E','F','G','H','I','J',' ','K','1');
   WRITELN(SO,'1');
 end;


procedure fourth_line;
 begin
   WRITE(ESC,'&a26c25Y');
   WRITE(SO,'L','1','1','M','N','O','P','Q',' ','R','S','1','1');
   WRITELN(SO,'T');
 end;
```

```
procedure activate_new_charset;
begin
  writeln(esc,')X');  (* SELECTION OF ALTERNATE CHARACTER SET *)
  first_line;
  second_line;
  third_line;
  fourth_line;
end;

begin  {MAIN}
 ESC:= CHR(27);
 CR:= CHR(13);
 SO:= CHR(14);
 writeln('THIS DEMO PROGRAM WILL DOWNLOAD A SPECIAL SELFDESIGNED');
 writeln('                C H A R A C T E R   S E T              ');
 writeln('       AND SHOW HOW TO WORK WITH THIS FEATURE          ');
 writeln('                                                        ');
 download_chars;
 activate_new_charset;
end.
```

### How can you determine if the download was successful?

You can easily determine if a download is successful by enabling the Terminal Test. The newly created character will be positioned where the redefined character is placed, (position depends on the selected character position in the escape sequence sent to the terminal).

### Problems which may arise when Downloading an Extensive Escape Sequence

A typical problem occurs when only a portion of the characters were loaded into the Terminal RAM. This problem may arise, when the 'WRITE' or 'WRITELN' procedures are used with Pascal. After 80 characters are downloaded, a <CR> <LF> is sent automatically, which breaks a longer escape sequence and causes the partial download. To avoid this, the preceding program example has used 'PRINT-INTRINSIC'.

**NOTE**

The terminal screen flashes for each downloading escape sequence it receives. To minimize these screen flashes, you can send all characters to be downloaded in one escape sequence by stringing the characters serially at the end of the sequence. In this context, however, you have to take into account that the characters defined in one escape sequence string are filed successively in a character set, so that a downloaded character cell initializes all subsequent characters.

## Use of Customized Characters under VPLUS/3000

User-defined characters can be used in VPLUS/3000 without difficulties. However, the following should be taken into account before starting:

1) Which character set shall be redefined?
2) Which characters of the selected character set shall be redefined?
3) Note that form designing requires a previous download of the modified character set.
4) Make sure that programs using self-created characters have performed a download before.
5) The newly created characters are only loaded in the RAM, so that a Hard Reset would reinitialize the RAM with the standard character sets.

## Special Considerations

The newly created character set can also be used with a terminal configuration of 132 columns/line. This, however, does not apply when creating VPLUS forms. In this case the line width will automatically be reset to 80 columns/line, because every time VPLUS is called, Intrinsic VOPENTERM is used fixing the standard line width at 80 columns/line (hard-coded).

Another restriction is that when copying the terminal screen to the printer, the customized characters will not be printed because only ASCII code, and not the character cell itself, is transmitted.

## Conclusion

The possibility of creating self-defined characters opens a variety of applications to the user. In this context, a particular significance is attached to the possibility of making special characters available to the user, which are typical and essential for his special field.

## References

For supplementary information on this subject, please refer to the *HP 700/92, HP 700/94 Reference Manual* (Part Number 5957-9982) and the *HP 700/92, HP 700/94 Users Manual* (HP Part Number 5957-9971).

## 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* |

# READER COMMENT SHEET

**Worldwide Response Center Support**
**HP 3000 Application Note #53: Use of Customized Special Characters on 700/9x Terminals**
**RC Questions & Answers (February 1, 1989)**

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 received the following error:**

**BLOCK NUMBERS ARE OUT OF SEQUENCE DBPUTERR -66 on DBLOAD.**

**What does this error mean?**

**A.** This error is caused by a problem with the pointers in the dataset you are unloading. The most common cause is a RESTORE of a single detail dataset without its corresponding master, prior to the DBUNLOAD. The following steps will correct the problem.

RESTORE a complete copy of the database from a good tape.

DBUNLOAD this copy (see the *TurboIMAGE Reference Manual*, Part Number 32215-90050).

ERASE the database with DBUTIL (see the *TurboIMAGE Reference Manual*).

DBLOAD the database (see the *TurboIMAGE Reference Manual*).

This will clean up the database and reset all pointers.

* * * * * *

**Q. Why am I receiving a different record length between FO SETS and FO datasetname?.**

**A.** When you perform a FORM sets, the "Entry Length" is the total record length of all the items in the dataset. If you are using a password which does not have access to a particular data item (or group of items) within that dataset, an FO datasetname will not return any data for that particular item. Hence, the total record length for an FO datasetname will be smaller than the total record length for that particular dataset on an FO sets.

Therefore, in order for the record lengths to be equal, you must have access to all items within each dataset.

more...

**Q. When a program uses the QUIT or QUITPROG intrinsic and is compiled with a native mode compiler on a Precision Architecture machine, it does not flush the remainder of a job stream when the intrinsic is called to end the program. Why?**

A. The QUIT and QUITPROG intrinsics work differently when called from Native Mode than they do when called from Compatibility Mode. Programs in CM continue to work exactly as they did on the classic 3000; any parm will flush the remainder of the jobstream. However, in NM, there is now greater flexibility. A positive parameter continues to flush the jobstream, but a negative parameter will not flush the remainder of the jobstream.

* * * * * *

**Q. A VPLUS application is hanging when the SELECT key is pressed. This only happens on terminals with a SELECT key. The program is checking LAST-KEY in the VPLUS COMAREA which is being set to zero even when the SELECT key is pressed instead of the ENTER key. What can be done to work around this?**

A. Checking the LAST-KEY field of the COMAREA is a good way to find out which key was hit, but on some terminals, when the SELECT key is pressed the LAST-KEY field is not properly changed. The best way to code around this problem is to check the COM-STATUS to ensure that it is zero before checking LAST-KEY.

If the SELECT key is pressed, COM-STATUS will be set to 154 which is 'UNKNOWN ESCAPE SEQUENCE'. COM-STATUS is a more accurate way to ensure that a proper key was pressed. Once COM-STATUS is zero, then check LAST-KEY. This will work even on terminals which exhibit the problem with LAST-KEY.

* * * * * *

**Q. When doing a READ FORM in BUSINESS BASIC, I cannot read the display only fields. Why?**

A. BUSINESS BASIC does not allow you to read display only fields. Currently, the only way to do this is to use VPLUS intrinsics for all of the forms handling in the program.

more...

Q. When reading records from a file using the FREAD intrinsic in a COBOLII program, every record read returns a condition code of CCG, meaning end-of-file. Because data is being read correctly from the file, the condition code should be CCE. Why is the condition code being set to CCG?

A. Condition codes are not saved after each call to an intrinsic. This means any statements that are executed AFTER the intrinsic call, but BEFORE the condition code test, could change the value of the condition code. Testing of condition codes should occur IMMEDIATELY after each call to an intrinsic.

* * * * * *

Q. How can a subprogram residing in a user's logon group or account SL be called from COBOLII rather than the program resident SL?

A. COBOLII allows two ways to pass the name of the subprogram that is being called: as a literal or as an identifier, (i.e. "SUB1" or WS-SUB-NAME). If a literal is used, the program resident SLs are searched, LIB=G starts with the group SL, Lib=P with the account SL. If an identifier is specified, the logon SLs are searched. COBOLII can not determine an identifier at PREP or RUN time, so it uses a LOADPROC which is higher overhead.

**NOTE**

As of COBOLII version A.01.06, adding the ON OVERFLOW statement makes the call to the subprogram act as if an identifier has been specified.

* * * * * *

Q. The "WHO" intrinsic is used to programmatically determine user capabilities. When the capabilities returned by the intrinsic are checked, it does not indicate if a user has Programmatic Sessions (PS) capabilities. Why?

A. The manual is incorrect: bit 10 in the first word indicates Programmatic Sessions and bit 10 in the second word is reserved for MPE. If you notice, bit 10 in the first word is set (indicating PS capabilities) rather than bit 10 in the second word.

For further information regarding the "WHO" intrinsic refer to the *MPE Intrinsic Reference Manual*, Part No. 32033-90007.

more...

**Q. Is there a way to access the VPLUS comarea that is being used by TRANSACT?**

A. TRANSACT allows you to modify the comarea with the "SET OPTION(VPLS)". In order to use this, define a character array in the program and use child items for the comarea items to be modified. A "SET(OPTION) VPLS=itemname" is done, bringing the comarea into the TRANSACT data register; it can then be modified as required. A "RESET(OPTION) VPLS" is then done in order to write the changed data back to the comarea. TRANSACT statements that refer to VPLUS forms, such as GET(FORM), cannot be executed between the SET(OPTION) and the RESET(OPTION), or an error will result. For more details and an example, refer to the SET verb description in the *TRANSACT Reference Manual*, Part No. 32247-90001, Section 6 -Transact Verbs.

NOTE: VPLUS intrinsics, such as VPRINTFORM, can be called and pass the TRANSACT comarea by use of the VCOM(formname) parameter of the PROC verb. This references the VPLUS comarea used by TRANSACT to reference the "formname" form. See Section 6 -Transact Verbs under "PROC".

**Q. The FORTRAN77 compiler is returning the error:**

**Error 147 ILLEGAL INITIALIZATION**
**when compiling. How can this be fixed?**

A. This is usually caused by one of three things:

A. A variable is initialized in a program unit in which initialization is not permitted.

B. A data value is not constant.

C. A subscript or substring expression in a DATA statement is not constant.

Use the "LIST ON" compiler directive to look at the source code which generated the message. Compare this to the above examples.

* * * * * *

**Q. How can I tell if my PASCAL reference manual is up to date?**

A. Information about manual versions and updates can be found in the "Catalog of User Documentation" section of the Communicator 3000 for the most currently released MIT.

For example, as of V-DELTA-3, the latest edition of the PASCAL Reference Manual (part number 32106-90001) was published on 10/83.