HP RPG/iX Reference Manual
Software Update Notice

900 Series HP 3000 Computers

HP Part No. 30318-90016
Printed in U.S.A. 1995

E0395

Printing History

New editions are complete revisions of the manual. Update packages, which are issued between editions, contain additional and replacement pages to be merged into the manual by the customer. The dates on the title page change only when a new edition or a new update is published. No information is incorporated into a reprinting unless it appears as a prior update; the edition does not change when an update is incorporated.

The software code printed alongside the date indicates the version level of the software product at the time the manual or update was issued. Many product updates and fixes do not require manual changes and, conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one-to-one correspondence between product updates and manual updates.

First Edition	December 1988	30318A.00.00
Second Edition	October 1989	30318A.00.04
Third Edition	November 1993	30318A.00.10
Update 1	March 1995	30318A.00.14

Preface

This update notice contains new information that should be added to the *HP RPG/iX Reference Manual*. Place the new pages included in this update at the end of Chapter 10 in your manual.

10a

The RPG Interface to VPLUS

VPLUS Communication Area Access

New COMAREA Enhancement

A new enhancement is now available for VPLUS application users, with RPG/iX version A.00.14 and later. This enhancement now allows read/write access to the VPLUS Communication Area (COMAREA). In the past this area was managed only by RPG, consistent with the philosophy of making all lower level I/O transparent to the user. One problem with this approach is that, for certain errors on VPLUS intrinsic calls, the application aborts with no possibility of recovery. With terminals and printers now being attached to systems over networks where noise may be a problem, this is no longer acceptable.

The new enhancement is enabled for your application by placing a

```
$CONTROL VPLUSCOM
```

record prior to your H-specification record. If you make no other changes to your program, it will continue to run as before, with the exception of certain errors on return from VPLUS intrinsic calls (COMAREA status word not equal to zero). In general, these errors are on intrinsic calls that access the terminal or printer, and will be discussed in more detail later. If an error does occur in this case, your program will not abort, and unexpected things could happen. To take care of this situation, read on.

The VPLUS COMAREA is specified to be either 60 or 85 16-bit words in length (see Chapter 6 of the *HP VPLUS Reference Manual*). It contains a mix of binary integer, double integer, logical, and ASCII data. RPG/iX allocates a space of 100 16-bit words for this, located in a special area of the run-time data space (the additional space is there just in case VPLUS ever expands the COMAREA). This area is zeroed out prior to running a VPLUS application. When RPG/iX calls VOPENTERM to begin everything, it makes the following initial settings:

- COMAREA length (word 3) to 85 (this is arbitrary, but allows 3075/6 terminals to be used)
- Language (word 2) to 5 (this specifies PASCAL, but works equally well for C, in which RPG/iX is written)
- LABEL'OPTION (word 10) to the value specified in in the workstation File Specification column 50
- FORM'STORE'SIZE (word 39) to the value specified in the workstation File Specification FORMSDL continuation record (if used)

$CONTROL VPLUSCOM

When the RPG/iX compiler detects the statement $CONTROL VPLUSCOM it sets a flag used to control certain items, both at compile time and at run time. It first creates two new reserved-word variables, *VC and *VSTAT. *VC is declared to be a 200 character alpha-numeric array of 1 character per element. Do NOT enter a File Extension Specification for this variable. *VSTAT is declared to be a 6 digit numeric variable with 0 decimal places.

Note

If you try using these items without the $CONTROL statement, you will get a compile-time error. Since these are declared for you, all you need to do is use them. No other declarations are necessary.

Reserved word *VC

The new reserved word *VC is initialized by the compiler to point to the start of the COMAREA. Thus, *VC,1 and *VC,2 point to the left and right bytes of the status word, for example. Since RPG does not have an internal binary data type, management of the COMAREA by the user can be somewhat tricky. This will be discussed later.

Reserved word *VSTAT

The reserved word *VSTAT may be used as an ordinary 6-digit numeric variable, but has special properties. For certain actions (SHOWMSG, SHOW, RDTERM, SHODATA, PRINTX, CLRMSG) the COMAREA status word is copied to this item on return from its VPLUS intrinsic call. The COMAREA status word (the first 16-bit word of the COMAREA) contains the result of a VPLUS intrinsic call. If the intrinsic executes correctly, this value will be 0. If it fails, the value returned will be a binary integer equal to an error number indicating the reason for the failure. It is the user's responsibility to test *VSTAT immediately after the EXCPT statement that initiates the action. An example will be shown later.

If *VSTAT is non-zero, you must set it to zero before taking any other action (except possibly calling VERRMSG using INTR/IPARM operators - an example of this will also be given later). To reset the status word to 0 do "Z-ADD0 *VSTAT ". This will set both *VSTAT and the COMAREA status word to 0. ONLY the Z-ADD operator will do this. If you Z-ADD a non-zero value to *VSTAT, *VSTAT will be set to this value but the COMAREA status word will still be set to zero.

VPLUS Intrinsics

You may now call VPLUS intrinsics directly from your RPG program using the INTR/IPARM operators. On return from the intrinsic call, *VSTAT is updated with the COMAREA status word. You must test this yourself to verify correct operation. This may only be done if you have specified $CONTROL VPLUSCOM in your program, it is a workstation application, and the intrinsic begins with the letter 'V' (a potential limitation, but OK for now).

If HP releases a new intrinsic beginning with 'V', and you use it in your VPLUS application, *VSTAT would still be updated from the status word in the COMAREA, but it would be meaningless for the new call (unless,of course, it is a new VPLUS intrinsic).

Due to the philosophy of managing lower-level I/O for the user, RPG/iX takes care of calling VOPENTERM and VCLOSETERM. Since these intrinsics also communicate with the terminal, it is possible they can also fail, with the resultant abort. With this enhancement, RPG now tests the status word on return from the call, and if non-zero will retry the intrinsic up to 5 times, with a one second pause between each retry, before aborting. This is done regardless of whether or not you have specified the $CONTROL VPLUSCOM statement. It is also done for the intrinsic VSHOWFORM where it is called from certain error routines not accessible to the user.

Modified Actions

A list of the RPG actions, the intrinsics called by them, and the behavior under the enhancement is shown below.

Some VPLUS intrinsics that fail may not be recoverable; i.e. there is some system failure that caused the problem. The response to these has not been modified. A program abort may still occur if the status return from an intrinsic call is non-zero.

SHOMSG — Calls VPUTWINDOW, VSHOWFORM. If VPUTWINDOW fails, it will put the error message in the window and still call VSHOWFORM. On return from the latter, the status word is placed in *VSTAT. If it is non-zero, the error message will be placed in the window. It then returns to the user with no abort. The user must test *VSTAT and decide what to do. WINDOWENH in the COMAREA may also be modified, as in PUTMSG above.

SHOW — Calls VSHOWFORM. Modifies COMAREA item SHOWCONTROL bit 9, depending on the value in the output buffer column 7 (blank or 'P' - the preload option). On return from the intrinsic, it copies the COMAREA status word to *VSTAT. If it is non-zero, it puts an error message in the window. It then returns to the program with no abort. The user must test *VSTAT and decide what to do.

RDTERM — Calls VREADFIELDS. The COMAREA item LOOK'AHEAD may be modified depending on the value in the output buffer, column 8. On return from the intrinsic, it copies the COMAREA status word to *VSTAT. If it is non-zero, it places an error message in the window. It then returns to the user with no abort. The user must test *VSTAT and decide what to do.

Chapter 13 of the *HP RPG/iX Reference Manual* notes that it also calls VGETBUFFER. This is actually done by the READ operation following the RDTERM action. There is no change from the previous behavior for this call.

CORERR	Calls VSETERROR, VSHOWFORM, and VREADFIELDS. If VSETERROR fails, it will put an error message in the window and continue with VSHOWFORM. On completion of either VSHOWFORM or VREADFIELDS, the COMAREA status word is copied to *VSTAT. If there is no failure, the final *VSTAT value is from VREAD-FIELDS. If either fails, an appropriate error message is placed in the window, and control returns to the user with no abort. The user must test *VSTAT and decide what to do.
SHODTA	Calls VPUTBUFFER, VSHOWFORM. If VPUTBUFFER fails, it puts an error message in the window. On return from VSHOWFORM, the COMAREA status word is copied to *VSTAT. If it is non-zero, an error message is placed in the window. Control then returns to the user with no abort. The user must test *VSTAT and decide what to do.
PRINT	Calls VPRINTFORM. On return from the intrinsic, the COMAREA status word is copied to *VSTAT. If it is non-zero, we also place an error message in the window. Control returns to the user with no abort. The user must test *VSTAT and decide what to do. If an error has occurred and the user has specified closing the spoolfile after each print, the close will not be done.
CLRMSG	Calls VPUTWINDOW, VSHOWFORM. Clears the window message buffer. If the RPG output buffer column 7 = 'I' it calls VSHOWFORM. On return, the COMAREA status word is copied to *VSTAT. If it is non-zero, an appropriate message is placed in the window. Control then returns to the user with no abort. The user must test *VSTAT and decide what to do.
CHMODE	Calls VTURNOFF. Puts the return status into *VSTAT. If it is non-zero, it puts a message into the window, and the mode is not changed. Control returns to the user with no abort. The user must check *VSTAT and decide what to do.
BLMODE	Calls VTURNON. Puts the return status into *VSTAT. If it is non-zero, it puts a message into the window, and the mode is not changed. Control returns to the user with no abort. The user must check *VSTAT and decide what to do.

Actions Not Modified

The actions that have not been modified are:

CHGNXT	Does not call any intrinsics. May modify COMAREA items NFNAME, REPEATAPP, FREEZAPP, depending on content of record.
GETNXT	Calls VGETNEXTFORM. If not in browse mode, calls VPUTBUFFER, blanking out the VPLUS data buffer.
PUTMSG	Calls VPUTWINDOW. If the output buffer column 7 is not blank, the COMAREA item WINDOWENH is modified.
BADFLD	Calls VSETERROR. May modify COMAREA item WINDOWENH depending on the value in the output buffer column 14.
PUTDTA	Calls VPUTBUFFER.
INIT	Calls VINITFORM.
EDITS	Calls VEDITFIELDS.
NUMERR	No intrinsic calls. Sets the event code to 9. A following READ operation places the number of fields that failed a VPLUS or user edit operation into the Input Specification field allocated for this value.
GETDTA	No intrinsic calls. Sets the event code to 10. A following READ operation calls VGETBUFFER. No change from previous behavior for this call.
FINISH	Calls VFINISHFORM.
WRTBAT	Calls VWRITEBATCH. Sets the COMAREA item DELETEFLAG to false prior to the call. On return, if not in browse mode and there is no error, the COMAREA item RECNUM is incremented.
PREV	Calls VREADBATCH. If not in browse mode, will set the COMAREA item CMODE to 'browse'.
REREAD	Calls VREADBATCH.
NEXT	Calls VREADBATCH.
RESUME	No intrinsic calls. Sets the COMAREA item CMODE to 'collect', and restores RECNUM and NFNAME to the values saved during a PREV action.
DELETE	Calls VWRITEBATCH. Sets the COMAREA item DELETEFLAG to true.
RDBTNU	Calls VREADBATCH. Sets the COMAREA item RECNUM to the record to be read.
GETFLD	No intrinsic calls. Sets the event code to 12. A following READ operation calls VGETFIELD. No change from previous behavior for this call.
PUTFLD	Calls VPUTFIELD.
LOADFM	Calls VLOADFORMS.
UNLDFM	Calls VUNLOADFORM.

Changing Data in COMAREA

As mentioned previously, reading or changing data in the COMAREA is somewhat tricky because it contains a mix of data, and RPG does not have an internal binary data type. The method chosen to handle this is to declare the array *VC as an alphanumeric array of one character per element.

To read the data, you must do so one character at a time. There are no conversion routines to make binary data readable (like DSPLY), but you can check the data using the COMP or TESTB operators. Alphanumeric data in the COMAREA (form names) can be displayed one character at a time. To modify binary data, you must create the desired bit pattern yourself using the BITOF and BITON operators.

Example 1

Suppose you want to set the window enhancement to "B". Because this is alphanumeric, it is easier. You need to modify the right byte of COMAREA word 8, which is *VC,16. The operation here is:

```
C                   MOVE "B"        *VC,16
```

To set the COMAREA length (word 3) to 60, note that the binary equivalent of 60 is '0000000000111100' (see any good computer science book for information on decimal to binary conversion). To set the length, clear and then set bytes 5 and 6. The operations here are:

```
C                   BITOF"01234567"*VC,5
C                   BITOF"01234567"*VC,6
C                   BITON"2345"    *VC,6
```

Note that "MOVE 0 *VC,5" or "Z-ADD0 *VC,5", etc. will not work. MOVE places an ASCII '0' into the left byte of the 'length' word, and Z-ADD yields a compile-time warning 6063 (result field must be numeric) and does not produce code for the operation.

Example 2

As another example, suppose you want to set retries to 8 instead of the default of 4. This is word 55 (bytes 109 and 110) of the COMAREA. The bit pattern for byte 110 is '00001000'. However, the result field is only 6 characters long, and cannot hold the byte reference *VC,110. Use a variable with a short name such as X to hold the value of the index.

```
C                   Z-ADD109        X       30
C                   BITOF"01234567"*VC,X
C                   Z-ADD110        X
C                   BITOF"01234567"*VC,X
C                   BITON"4"        *VC,X
```

Example 3

Here are some coding examples. Suppose you are in a noisy environment and you would like to protect the SHOW action by retrying it five times with a 3 second pause between retries, in the event of a VSHOWFORM failure. The following example shows how to do this:

```
$CONTROL VPLUSCOM
H
                          .
                          .
C                     Z-ADD0         X       20
C                     Z-ADD3         TIME    40
C                     SETON                     80
C           AGAIN     TAG
C                     EXCPT
C           *VSTAT    COMP 0                    9191
C  N91               GOTO AOK
C                     SETOF                     91
C           X         IFEQ 5
C                     GOTO ABORT
C                     END
C                     ADD  1         X
C                     INTR PAUSE
C                     IPARM          TIME
C                     Z-ADD0         *VSTAT
C                     GOTO AGAIN
C           AOK       TAG
C                     SETOF                     80
C                       .
C                       .
C           ABORT     TAG
C                       .
C                       .
OTERMINALE        80
O                                    6 "SHOW  "
```

Example 4

As another example, suppose you want to do your own batch file operations. You want to check the COMAREA status on a VWRITEBATCH call, and display an error message if it is non-zero. (Note: if you have a form on your screen while executing the following code, it would mess it up.) The code for doing this follows:

```
$CONTROL VPLUSCOM
H
                        .
                        .
E                    BUFR        1 72
                        .
                        .
C                     Z-ADD0         *VSTAT
C                     INTR VWRITEBATCH
C                     IPARM          *VC
C           *VSTAT    COMP 0                     9191
C   91                EXSR WRBBAD
                        .
                        .
CSR         WRBBAD    BEGSR
C                     SETOF                      91
C                     Z-ADD72        BUFLEN  40
C                     Z-ADD0         ACTLEN  40       no. chars returned
C* do not clear the comarea status word.  we need it here.
C                     INTR VERRMSG
C                     IPARM          *VC
C                     IPARM          BUFR
C                     IPARM          BUFLEN
C                     IPARM          ACTLEN
C* display the message.
C           BUFR      DSPLY
C* pause 10 seconds to allow it to be read.
C                     Z-ADD10        TIME    40
C                     INTR PAUSE
C                     IPARM          TIME
C* reset status to zero.
C                     Z-ADD0         *VSTAT
C                     ENDSR
```

Using the VPLUS Environment

Once RPG/iX has opened the workstation, you can use the INTR and IPARM operators, *VSTAT, and *VC to write your VPLUS applications using the VPLUS intrinsics. If you use a mix of RPG actions and intrinsic calls, you must be aware that the actions in general do more than just call the intrinsics. For example, if you do a SHOW action, RPG will update the COMAREA word 34 (SHOWCONTROL) bit 9 (PRE-LOAD) based on the value in column 7 of the Output Specification record for SHOW (blank, 'P', or invalid). This would modify anything you put in that spot prior to the action.

A problem that currently exists in the RPG VPLUS interface is that, if you wish to do an EDITS action on data in the VPLUS buffer following an RDTERM action, you lose the ability to use the f1-f8 function keys. In other words, if you respond to RDTERM with f1-f8, do the EDITS, and then do a READ TERMINAL, the F1-F8 indicators do not get set.

A Complete Sample Program

The following program uses the new enhancement which overcomes the problem described above. The forms file is not presented. If you wish to run this program, you would need to create your own forms file with appropriate fields.

```
* This program shows how to use the new VPLUS enhancement,
* both in accessing the COMAREA and in using INTR/IPARM oper-
* ators.  If you enter data and press ENTER, editing data as
* needed, the data will be copied to MPEFILE.  If you press
* f1-f7,no data is read.  Instead the form is
* redisplayed, and you may enter different data.  If you press f8,
* the program terminates.
* To do the EDITS operation, do the following:
* Do a SHOW followed by a RDTERM action, and then use the new
* capability to call VFIELDEDITS.  Then access the COMAREA to
* see if f8 was the last key pressed, and if so terminate the
* program.  If f8 was not the last key pressed, check the
* 'numerrs' value; if it is non-zero, try again, with an error message
* in the window.  If no errors are found, the program continues.
*
$CONTROL VPLUSCOM
H
F*************************************************************
F*               File Specifications                         *
F*************************************************************
FTERM    UD  V     110            WORKSTN   L3B
F                                                KFORMS VPFORMS
FMPEFILE O   F      72            DISC
E*************************************************************
E*               Array Specifications                        *
E*************************************************************
E                    ERM     1   1 79
E                    MSG        79  1
I*************************************************************
```

```
I*               Input Specifications                           *
I***************************************************************
ITERM    AA  01   1 C0   2 C0   7 C1
I       OR       1 C1   2 C0   7 C1
I                                       1   2OEVENT
I                                       3  17 FORM
I                                      18  21OLENGTH
I                                      22  27ODIGIT
I                                      28  33 ALPHA
I*
I* Define function key f1, f2, f3, f4, f5, f6, f7, f8.
I*
ITERM    BB  11   1 C0   2 C1
I       OR   11   1 C0   2 C2
I       OR   11   1 C0   2 C3
I       OR   11   1 C0   2 C4
I       OR   11   1 C0   2 C5
I       OR   11   1 C0   2 C6
I       OR   11   1 C0   2 C7
I       OR   18   1 C0   2 C8
C***************************************************************
C*               Calculation Specifications                    *
C***************************************************************
C                     BITOF"01234567"X       1
C           START     TAG
C                     SETOF                     0111
C                     SETOF                     18
C                     EXSR GETNXT
C                     EXSR SHOW
C                     EXSR EDITS
C   18                GOTO ENDPGM
C   11                GOTO START
C                     EXCPT          RECOUT
C                     GOTO START
C           ENDPGM    TAG
C                     SETON                     LR
C***************************************************************
C*               Subroutine GETNXT                             *
C*   INIT   = Initialize fields in current form                *
C***************************************************************
C           GETNXT    BEGSR
C                     MOVE "GETNXT"  ACTION  6
C                     EXCPT          ACTOUT
C                     MOVE "INIT  "  ACTION
C                     EXCPT          ACTOUT
C                     ENDSR
C***************************************************************
C*               Subroutine SHOW                               *
C*   SHOW  = Display current form, initial data, any           *
C*           messages.                                         *
C*   RDTERM= Read input from terminal to data buffer.          *
```

```
C*************************************************************
C           SHOW      BEGSR
C  N90                MOVEAERM,1     MSG
C  N90                EXSR SHOMSG
C                     MOVE "SHOW  "  ACTION
C                     EXCPT          ACTOUT
C                     EXSR RDTERM
C                     ENDSR
C*************************************************************
C*               Subroutine SHOMSG                           *
C*   DISPLAY MESSAGE, ANY NEW DATA.                          *
C*************************************************************
C           SHOMSG    BEGSR
C                     SETON                    54
C                     EXCPT
C                     SETOF                    54
C                     ENDSR
C*************************************************************
C*               Subroutine EDITS                            *
C*    EDITS = Perform edits on fields in current form.       *
C*    FINISH= Perform final processing on current form.      *
C*    GETDTA= Write data in data buffer to user program.     *
C*************************************************************
C           EDITS     BEGSR
C                     SETOF                    90
C           AGAIN     TAG
C   11                SETOF                    11
C   18                GOTO ENDEDT
C   90                EXSR SHOW
C* this is where the new VPLUS enhancement begins.
C* note: when testing the comarea "lastkey" and "numerrs" values,
C*       I have assumed the left byte of the 16-bit word is zero.
C*       In general, this will always be true (lastkey could be
C*       -1 for 3075/6 terminals, and numerrs would need more than
C*       256 fields on the screen to overflow into the left byte).
C                     INTR VFIELDEDITS
C                     IPARM          *VC
C* check comarea status. better be zero.
C           *VSTAT    COMP 0                   1818
C   18                GOTO ENDEDT
C* if comarea lastkey = 8, abort
C                     TESTB"4"       *VC,12        18
C   18                GOTO ENDEDT
C* else if lastkey  0, turn on indic. 11
C* note: if so desired, we could put in a specific test for
C*       each function key.
C           *VC,12    COMP X                   1111
C* if numerrs is not equal to 0, try again
C           *VC,14    COMP X                   9090
C   90                EXSR ERRMSG
C   90                GOTO AGAIN
```

```
C                     SETOF                     90
C                     MOVE "FINISH"  ACTION
C                     EXCPT          ACTOUT
C                     MOVE "GETDTA"  ACTION
C                     EXCPT          ACTOUT
C                     READ TERM                    H1
C           ENDEDT    TAG
C                     ENDSR
C*************************************************************
C*               Subroutine RDTERM                           *
C*   RDTERM = Read input from terminal to data buffer.       *
C*************************************************************
C           RDTERM    BEGSR
C                     MOVE "RDTERM"  ACTION
C                     EXCPT          ACTOUT
C                     ENDSR
C*   ERRMSG = put the error msg in the window for retry.
C           ERRMSG    BEGSR
C                     Z-ADD79        BUFLEN  40
C                     Z-ADD0         ACTLEN  40
C                     INTR VERRMSG
C                     IPARM          *VC
C                     IPARM          MSG
C                     IPARM          BUFLEN
C                     IPARM          ACTLEN
C*
C                     INTR VPUTWINDOW
C                     IPARM          *VC
C                     IPARM          MSG
C                     IPARM          ACTLEN
C                     ENDSR
O*************************************************************
O*               Output Specifications                       *
O*************************************************************
OTERM    E                ACTOUT
O                         ACTION     6
O        E        54
O                                     6 "SHOMSG"
O                                     8 "79"
O                                     9 "J"
O                         MSG       88
O*
OMPEFILE E                RECOUT
O                                     6 "Digit:"
O                         DIGIT     12
O                                    20 "Alpha:"
O                         ALPHA     26
O                                    72 "VPFORMS"
**
```