LIVING IN AN RPG/3000 ENVIRONMENT: ALTERNATIVE APPROACHES TO ENTRY AND UTILIZATION OF RPG ON THE HP 3000

DUANE R. SCHULZ
DIRECTOR OF DATA PROCESSING
M. A. BIOPRODUCTS
WALKERSVILLE, MARYLAND

INTRODUCTION

This discussion is presented with the intended goal of developing a fuller understanding of the problems associated with utilizing the HP3000 in an RPG programming environment, reviewing several alternative approaches to the traditional batch report generation uses of RPG. Other commonly unsolved problems, such as techniques for entry and maintenance of RPG source code on the 3000 and the use of RPG as an interactive language, will be discussed. Emphasis will be placed on the transition in conceptual approaches to data processing as the 3000 is used more fully by RPG-oriented users.

BACKGROUND

As we all know, during the past 15 years several events have occurred in the data processing marketplace which have placed computing power within the reach of medium-to-small companies: some of the most notable were the announcement of the IBM System/3 series, and later the IBM Systems/32 and /34. The migration of these systems into the world marketplace in the past decade has also led to the widespread use of RPGII, a vendor-proprietary programming language initially designed to convert card images to formatted output. Because of the market strategy which has been used by vendors of small computers featuring RPG as the only high-level programming language, RPG and RPG programmers have, justifiably or not, inherited a reputation as unsophisticated, unprofessional strangers to the data processing community (a complete examination of the social and professional evolution of the RPG programming community, seemingly as a nearly separate pocket of the data processing population, would require more time than I wish to devote to it here). Regardless, the RPG-oriented machine hit with great force: though RPG shops are usually

small, no less than 25,000 IBM System/3s were shipped in the first 5 years of the product's life, and the newer machines, especially the System/34, have hit with much greater force than even the Sys/3. Though the use of RPG is not as visible as COBOL and Company, its use is extremely widespread and popular, and will remain so.

Now, as users of the more common RPG machines are experiencing more and more difficulty with poor system capabilities, seemingly endless conversions, and dead-ends to attempted shifts to more appropriate use of systems technology, the flexibility and seeming open-ended capabilities of time-shared minicomputers are looking a great deal more appealing. With the failure of IBM to produce the promised System/38, and our successful conversion to the "mini" enviromnent, the migration to the Hewlett-Packard 3000 as an RPG machine will undoubtedly occur at a rapidly increasing rate. And, as we have all discovered, HP offers an excellent RPG and operating software, tied to hardware flexibility and state-of-the-art engineering most of us are very surprised to find. Nonetheless, I don't believe the average (or in HP's case, above average) System Engineer knows quite what to do with an all-RPG, green-to-the-minicomputer-marketplace, installation. After the installation of the 3000, several problems present themselves, demanding solution before any serious long-range software plan is developed to facilitate movement towards the appropriate use of the HP3000 as it is intended. The most immediate problems are: How will you enter and maintain RPG source code on the 3000, and how will you change your approach to the use of RPG on the 3000?

RPG ENTRY AND MAINTENANCE: 5 OPTIONS

Most RPG users who walk into an HP environment begin with two problems: they code in a symbolic, column-oriented language which still reflects the cards it was developed on, and they are probably used to a vendor-supplied Silver Spoon (pronounced: SEU, On-line CCP/RPG Maintenance, etc.). In addition, most RPG systems come equipped with a library maintenance utility (pronounced: Silver Spoon Number Two) and possibly a "structured programming" capability designed to cut redundancy of code. IBM's Auto Report is a nice example of the latter. HP offers None-Of-The-Above, a new concept to most RPG users, so our first task on the 3000 will be to develop a strategy for handling this problem. Though the options are endless, I would like to review 5 relatively productive options for entry and maintenance of RPG on the 3000. They are:

a) Removable media
b) RPGWIZ
c) RPG/ENTRY
d) EDITRPG
e) In-house solutions

REMOVABLE MEDIA:

Probably the most traditional solution to facilitate entry of RPG on the 3000 is the use of some offline device to place your source code on removable media. This would include 80- or 96-column cards, floppy disc, or even mag tape. The removable media may be submitted directly to the RPG compiler, or placed on disc via the FCOPY utility or even a non-written library maintenance program. The advantage of this approach is the ease of formatting cards or floppy disc output through drum cards or format records. It is also comforting to some to be able to correct compiler errors by plucking the appropriate card and replacing it with a new one. Nonetheless, the hardware and drivers necessary to handle removable media are costly and in most cases unavailable from HP, so this solution is not particularly cost effective. Secondly, this type of approach is quite different in orientation than the use of the HP as an online system. If we cannot find a way to enter our source code without cards or diskettes, what kind of systems will we later offer our users?

RPGWIZ:

Undoubtedly the most common of all solutions to the RPG entry problem is RPGWIZ, a series of DEL screens controlled by an RPG/DEL program, WIZ. RPGWIZ was developed by Craig Jester at Hewlett-Packard in Rockville, Maryland, and is available in the HPGSUG Contributed Library. RPGWIZ allows entry of RPG code with relative ease through Block Mode utilization of an HP-264X series terminal. The WIZ program initially displays a screen for entry of the $CONTROL Statement for the program being entered, also allowing setup of the program name and line numbers, both of which are duplicated into all statements. A HEADER/CONTROL Statement screen is then displayed and used to produce the H-specifications for the program. Next, the FILE specifications screen is displayed, and program entry begins. With the use of Block Mode, it is relatively difficult to enter values into the wrong columns of RPG source records. WIZ allows movement from one specification type to another, and uses soft keys for field toggling in the INPUT and OUTPUT Screens. RPG specifications are written to a KSAM work file, allowing retrieval and correction of a statement entered earlier during the session. Another attractive feature of RPGWIZ is that the lower half of each form displays statements entered just before the statement now being entered, allowing the programmer to look back through the previous 8 to 10 statements and facilitate corrections. Once program entry is terminated (using the f8 soft key), the KSAM RPG work file is copied, sequentially-by-key, into the final sequential source file, using another contributed program, WIZ2. Once this is done, RPGWIZ can no longer be used to modify the program: modification is then most commonly done with EDIT/3000.

Installation of RPGWIZ should be done by users with a good knowledge of the RPG/DEL interface, and the use of DEL forms. The problem of multiple users executing RPGWIZ at the same time can be solved by building a set of KSAM work files, and developing a User Defined Command (UDC) which includes all steps of the RPGWIZ entry process and will allow the user to specify the desired KSAM work file and the name of the final source file to be created. It will take a few days to be totally comfortable with RPGWIZ (as is true with the following two options), but it can be a very productive and useful tool for RPG programmers. Unfortunately, it has one major drawback: DEL is no longer a part of the HP3000 software offerings; virtually all new HP customers who use an HP screen formatter will use VIEW.

RPG/ENTRY:

At this moment, several of us are working on, or have completed, an RPG entry routine which uses VIEW screens for formatting RPG being entered into the 3000. To avoid confusion, I will review only one of these, RPG/ENTRY, which I developed in 1979. RPG/ENTRY is available in the 1980 San Jose tape swap, and has been given to the HPGSUG Contributed Library for 1980 distribution in release 07. RPG/ENTRY consists of nothing more than a VIEW form file containing one form for each RPG specification type, including one for all comments. Relatively extensive FORMSPEC edit specifications are included, allowing for quicker specification entry through the use of justification and pre-compiler type editing. An ASCII representation of a typical RPG/ENTRY screen is included in appendix A.

RPG/ENTRY does not interface to any user-written program. It is designed to be used with HP ENTRY/3000 utility. Execution of RPG/ENTRY is done by running ENTRY.PUB.SYS and specifying the RPG/ENTRY form file. Specification sequencing is controlled by using the soft-key functions defined for ENTRY; as with RPGWIZ, I first display $CONTROL and HEADER/CONTROL screens. Out-of-sequence forms selection is handled by indicating the specification type desired, then pressing ENTER and f6 (this writes blank records to the Batch file and necessitates my stripping program). RPG/ENTRY is designed for speed and productivity, and relies on the features of ENTRY for control of the entry process. Correction or review of prevously entered statements is done by using Browse mode. One entry is terminated, the ENTRY batch file is written to a KSAM work file, stripped, and resequenced to the final sequential RPG source file.

Implementation of RPG/ENTRY should be done carefully according to the RPG/ENTRY documentation. Again, I must stress the fact that RPG/ENTRY is designed for the entry, not modification, of new RPG source code. The most commonly used tool for maintenance of existing RPG source code is still EDIT/3000, which may be enhanced by using the more advanced EDITOR commands and capabilities.

EDITRPG:

To complement RPG/ENTRY or RPGWIZ, I have developed another tool, EDITRPG, which allows RPG programmers to use the text editor in conjunction with any HP CRT for easy manipilation of existing RPG source files (EDITRPG is available along with RPG/ENTRY). EDITRPG is a group of USE files containing EDITOR "Q" commands for formatting the terminal screen. To use EDITRPG, evoke the EDITOR and enter "USE RPGEDIT". This file loads soft keys with USE commands for each RPG statement type. Text in the file to be modified (unless new code entry is to be done). To modify an existing statement, find it, then enter "USE MOD", which outputs a columnar mask on line 20 of the screen and sets on memory lock. The editor "M" command is then used to do the actual modification; the MOD mask simply makes column orientation and viewing of the corrected statement simple and reliable. For listing and addition of new statements, the soft keys are used to select the appropriate statement format. For instance, to add input statements, f6 should be pushed, followed by the "ADD" command. The soft key will execute another USE file, which sets tabs, explains tab settings, labels the specification type, outputs a columnar mask, and sets memory lock at line 18 on the screen.

I have found EDITRPG to be an extremely powerful yet simple tool for RPG modification and entry. Potential users should be sure that their terminal is equipped with a tab-setting capability (HP264X or HP2621) and be familiar with its use. Entry of new RPG via ENTRYRPG should only be done by experienced programmers who are familiar with the columnar layout of RPG specifications. A simulation of an EDITRPG screen format is included in appendix A.

IN-HOUSE SOLUTIONS:

Though all of the above options will lead to some solution of the RPG entry and maintenance problem, there will be no solution which is ideal for your environment. A final option, one which led to all of the above solutions, is to do it yourself. One of the most important differences with the HP3000 environment is that users have the freedom to create their own systems software. It is constantly amazing to note how many people immediately reject the HP3000 because no Silver Spoons are available. Rather than consider this problem a problem, here is a strong opportunity for all of us to get acquainted with the 3000 at a level we've not previously had access to. Given the availability of languages and contributed software, there are several feasible approaches to the RPG entry and maintenance problem. For instance, all of the afforementioned solutions could be tailored and enhanced to fit specific needs. A full-capability library maintenance system could be developed to simplify program development and documentation capabilities. An interpretive RPG entry precompiler could be written. Again, the successful use of the 3000 in an RPG environment will depend upon

the capability to generate creative solutions, rather than rely on others for answers. By installing an HP3000, we have entered a situation which will require much stronger internal site support.

USING RPG ON THE HP3000: CONCEPTUAL APPROACHES

Once a tool for RPG program maintenance has been selected or developed, RPG users must determine how they will use RPG in developing new systems on the 3000. Again, there are two basic alternatives: find a standard batch RPG approach to systems design, or explore as many new uses of RPG as can be found before settling on which system capabilities will be made available to users. When faced with the need for interactive, on-line system capabilities, most HP3000 RPG users begin using other languages which are more suited to on-line activities. I feel that a blanket rejection of RPG as an interactive language is a mistake. Because it is a cycle-driven language, there are unquestionable limitations to the flexibility of RPG in an on-line environment: most systems tasks are handled by the RPG compiler. This does not mean that RPG programmers have no control over files and devices - we are simply used to using RPG in a fashion which allows the system to do most of the work. Given expertise with all the capabilities of RPG, we have the tools necessary to develop an interactive environment using RPG as the primary language. The remainder of this discussion will cover three basic alternative approaches to using RPG as a primary applications language. We know that disc I/O, report and output formatting, and basic file handling are easily and quickly handled by RPG; we will attempt here to discuss only how RPG can be used to handle on-line data collection and inquiry.

RPG/DEL:

For DEL users, the HPGSUG Contributed Library includes RPG/DEL, written by Paul Grazulis of Hewlett-Packard in Dayton, Ohio, which allows RPG programs to call up DEL forms and conrol terminal I/O. The terminal is defined as a variable-length, update primary or demand file. Use of the demand file option allows reading of the terminal at will; output can be accomplished via the EXCPT calculation. Terminal handling is done by preceding output literals with a two-digit function code defining the action required. The 09 subfunction allows output of escape sequences, giving the RPG programmer complete control of the terminal (an HP264X series terminal must be used), including turning block mode on and off as desired. RPG/DEL programs may, by definition, be used for on-line input editing, control of general processing programs, and random inquiry/master file maintenance. Any data structure, including direct file, KSAM, or IMAGE data sets, may be utilized and accessed by RPG/DEL (the RPG/IMAGE interface is another topic which would

require another paper to adequately evaluate, but it seems to be the weakest RPG/3000 capability at present). Probably because it is an unsupported software product, RPG/DEL continues to be under-utilized, though it is an especially powerful and reliable tool for using RPG in an interactive environment.

RPG/VIEW:

With the announcement of VIEW, Hewlett-Packard unveiled their first block mode RPG terminal interface (again, the HP264X series must be used, until the 2621 series terminal supports block mode). As with RPG/DEL, the terminal is defined as an update file with a special device class name, but the resemblance stops there. First of all, instead of using a separate relocatable library, the VIEW interface intrinsics are handled by the RPG compiler. Through a series of special output codes, and through READ and EXCPT calculations, the RPG language is handled similarly to other "real" languages, with calls to VIEW intrinsics at will. The calculation frame of an RPG/VIEW program requires much more organization than normal batch RPG codes (though this exercise is probably good for RPG-only programmers), and the learning curve required for implementation of RPG/VIEW is significant. As compared to RPG/DEL, the terminal is under the complete control of VIEW; no escape sequences are allowed, so it is important to have the proper education regarding the use of VIEW and VIEW form files. RPG/VIEW applies in generally the same areas as RPG/DEL, but because it is an HP-supported product, it is certain that major enhancements to the capability will appear as the RPG/3000 user base grows.

INTERACTIVE RPG:

The block mode RPG capabilities discussed above make the assumption that: all users can afford and wish to use HP264X series terminals; block mode is the most effective treatment of on-line I/O; RPG alone is not sufficient to handle on-line communication. None of these is necessarily true; I have found that the use of just a few basic RPG techniques are sufficient to completely control on-line operations:

1) Operate in character mode: Handle I/O as up-to-80 character strings and manipulate these strings within the program with MOVE/calculation loops. This will allow use of any type of terminal.

2) Handle I/O with two files: One defined as Input Demand with a device class of $STDIN, the other as an Output file with a device class of $STDLST. This will allow elimination of file equations for execution; I/O operations may be handled at will entirely through the use of READ and EXCPT calculations.

3) Rely as heavily as possible on KSAM files and IMAGE data sets for data retrieval and updating. Again, this offers more speed and control over I/O operations.

4) Structure all calculation operations into basic subroutines. Use EXSR statements to perform I/O tasks, as well as the more traditional uses of subroutine-oriented calculation logic.

5) Use SETON and SETOF in conjunction with the above calculation structure. This, used with all the points above, will prevent the RPG cycle from controlling program execution, which is the key to on-line RPG.

6) Place message text, desired escape sequences (if you are not yet familiar with the use of terminal escape sequences as output literals, it is a capability well worth pursuing), and other desired output control characters into a compile-time array. Used with a variable array index and subroutines, output can be handled with 2 output specifications.

This approach, tied with an effective RPG maintenance system and UDC files for ease of operation, provides us with very quick, easy to support, RPG programs for use in an on-line environment. No high-level systems software or applications programming maintenance is required - just the MPEIII file system.

CONCLUSION

To briefly summarize, RPG users have been offered a unique opportunity: we have the opportunity to show our vendor the true capabilities of a language which has been considered to be predominantly oriented towards batch processing. In fact, given a good degree of professional curiousity and a high level of technical competence, an interactive RPG shop can be one of the most interesting and creative HP3000 environments. Fortunately, Hewlett-Packard has shown a very strong desire to enter the RPG community, and as the events in the marketplace place more RPG-driven HP3000's on the map, our capability to educate the new HP community as to the alternatives they may choose from will be more and more critical.

APPENDIX A
LIVING IN AN RPG/3000 ENVIRONMENT

A SAMPLE RPG/ENTRY SCREEN:

```
-------------------------------------------------------------
              RPG/VIEW CALCULATION SPECIFICATIONS
-------------------------------------------------------------
                    Line Number-------->[50150]
                    Statement Type----->[C]
                  ---------------------------------
                    Level/Subr Ind----->[  ]
                    Indicators--------->[ 01][N99][   ]
                    Factor 1----------->[FIELDA    ]
                    Operation---------->[MULT ]
                    Factor 2----------->[.01       ]
                    Result Field------->[PCT   ]
                    Result Fld. Length->[  5]
                    Decimal Positions-->[4]
                    Half Adjust-------->[ ]
                    +/-/0-blank Inds--->[  ][  ][98]
                    Comment------------>[CALCULATE PCT.  ]

                    Program Name------->[SAMPLE]
-------------------------------------------------------------
```

A SAMPLE EDITRPG EDIT MASK:

```
                  CALCULATION STATEMENT FORMAT
Tab Assignments:
 1. Spec. Type (06)   5. Operation     (28)   9. Decimal Pos.    (52)
 2. Indicator 1(09)   6. Factor 2      (33)  10. Hi/+ Indicator  (54)
 3. Indicator 2(12)   7. Result Field  (43)  11. Lo/- Indicator  (56)
 4. Factor 1   (18)   8. Length        (49)  12. Eq/0 Indicator  (58)
 ------------------------------------------------------------------
          11111111112222222222333333333344444444455555555556666666666
 12345678901234567890123456789012345678901234567890123456789012345678
 ^    ^   ^  ^      ^         ^    ^         ^     ^   ^   ^  ^

/A
   58 C   01N99   FIELDA    MULT .01       PCT     54     98
```