CRI

CURRENT AND RELEVANT INFORMATION

The Bi-monthly newsletter for RELATE, CREATE BUILDER, and GRAF Users.

Welcome to the New Improved Unexpurgated CRI Software Users Bi-monthly Newsletter (take a deep breath and continue).

Our goal is to keep you, our innovative and interesting customers, informed as to what interesting and innovative things are being done with our products, both by other innovative and interesting customers and by our own company (which is, without a doubt, interesting and innovative).

RELATE has been given an excellent workout through the years at customer sites, and we in customer support are constantly amazed at what can be done with our products. We welcome this opportunity to share some of these ideas and experiences with you. For your part, if you would care to share your concepts, we will be delighted to publish them here either anonymously or with credit.

Enjoy, and we'll see you in two months!

BUILDER LOOPING

When a CALL or SET SCREEN is done to a subroutine screen (a screen with no LAYOUT section), BUILDER will automatically loop forever through the ACTION or ENTER section of the called screen. This is a reasonable approach to the problems that could appear when a user has no visible method of control over the screen. User interface through a subroutine screen, then, can only be done by PROMPTs, as function keys cannot be accessed and the user cannot enter values into the screen.

This was incorrectly reported in the 4.4 PSR as a bug that would be removed in the 4.5 release. It was intentionally designed, and will remain, this way.

$CHANGED

The new $CHANGED option in 4.5 BUILDER was only noted in the documentation as a system-defined variable that returns TRUE if the value of any variable has been altered by the user.

In fact, it is also a function of the format:

```
$CHANGED(varexp)
```

Where "varexp" is either the name of a variable enclosed in quotes whose value is to be checked, or an expression whose value is the name of a variable whose contents are to be checked.

If no "varexp" is included, $CHANGED checks all variables in the current screen.

CHALLENGE (semi-outer join)

The question of semi-outer joins has been a fascinating one all along, and several approaches have been suggested for obtaining the desired results. None of them, however, seem to improve on the performance of the standard:

```
SELECT A.@,B.@ WHERE A.KEY=B.KEY
COPY TO TEMP
SELECT A.@ WHERE $COUNT (BY A.KEY
            WHERE A.KEY=B.KEY)=0
COPY TO TEMP
```

The semi-outer join solved here involves one main file, which may have zero or more matches in a secondary file for each record. One can get close to a single-step solution with a SELECT of the format:

```
SELECT A.@, TEXT =
            $IF ($COUNT (BY A.KEY WHERE A.KEY=B.KEY)=0,
                 "  ", B.TEXT)
WHERE A.KEY=$IF ($COUNT(BY A.KEY WHERE A.KEY=B.KEY)=0,
                A.KEY, B.KEY)
```

This does not generate the "all files may not be joined together" warning. It correctly obtains all matches, but generates multiples of the non-matches. This is because, in the event of a non-match, the files aren't joined and we obtain a cartesian product of file B with all non-matches in A. Adding a UNIQUE BY ‹something› obtains the result in most cases but makes it significantly slower than the old COPY.

How have you approached this problem? Good answers will be published here. Send your solution to:

> CRI, INC.
> 5333 Betsy Ross Drive
> PO Box 58004
> Santa Clara, CA 95052
> ATTN: Customer Support Department

ZONED NUMBERS

Although we seriously discourage the use of packed and zoned numbers, they sometimes just can't be avoided (darn!). When dealing with zoned numbers generated by software other than RELATE, beware of unsigned numbers! For example, RELATE has no way of distinguishing an unsigned 500 from a positive 500 when printing the data, but the internal storage is different. Hence, joins may appear not to work when actually you're trying to join an unsigned 500 to a positive 500.

The solution: force RELATE to treat the value as a signed number, by adding a positive zero. For example:

```
SELECT A.@,B.@ WHERE A.KEY+0=B.KEY+0
```

Presto fixo, everything works!

LIVE AND LEARN

A puzzle section for RELATE and BUILDER users. Our thanks to customers whose inquiries generated these puzzles.

1. I have two procedure files, A and B, as follows:

```
        A:  CLOSE
            OPEN FILE X
            CREATE FILE Y; FIELDS=(FLAG,I,2)
            ADD:I
            1
            //
            CREATE INDEX BY FLAG
            EXECUTE B

        B:  SET PATH Y
            CHANGE:S FLAG
            PRINT
            IF FLAG<>99
                 SET PATH X
                 SELECT @ WHERE FLAG>Y.FLAG
                 PRINT
                 CHANGE
                 EXECUTE B
            ENDIF
```

When I EXECUTE A, I will type 99 into FLAG when it eventually prompts me. Why do I get two consecutive prompts for FLAG, as follows, before it ever prints anything?

```
            FLAG [1]?99
            FLAG [99]?
```

What is an easy way to prevent this?

2. I'm opening an MPE file through RELATE. This file contains text, and roughly every other line is blank. In some cases, however, there are two consecutive blank lines. I want to find out how many blank lines there are which are immediately preceded by text (in other words, I don't want to count the second of 2 blank lines.) Here is my original attempt at a solution, which unfortunately always give me a result of zero. Why?

```
   OPEN FILE TEXT;TYPE=MPE;FIELDS=(TEXT,A,72),(LNUM,A,8)
   SELECT X=$COUNT (WHERE TEXT="  " AND $LAST(TEXT)<>"  ")
```

BUILDER EFFICIENCY

It helps the startup time for each screen if as much duplicate text as possible is eliminated.

METHOD 1	METHOD 2	METHOD 3
A; LENGTH=10	A,B,C; LENGTH=10	A, &
B; LENGTH=10		B, &
C; LENGHT=10		C; LENGTH=10

Method 1 declares each variable on a separate line, even though they have the same definition. Methods 2 and 3 lump similar declarations together (and method 3 still preserves the easier-to-scan one variable per line).

In a sample application with 75 variables, the time difference between method 1 and the other two methods varied between 2 and 10 CPU seconds (on a series 42) depending on the complexity of the declaration.

BUILDER-SCROLLING and ARROW KEYS

The BLDRTERM configuration for HPFKEY and HPCHARFKEY contains, in the initialization sequence, "Esc&s1A". This turns "XMIT FUNCTION" to YES or ON, enabling BUILDER to recognize the arrow keys.

The difficulty with this is that, when you go into DEBUG mode and scroll your screen up and down, BUILDER remembers these scroll requests and you can get some strange results. This can be remedied by either removing the XMIT ON (above) from the initialization sequence (in which case arrow keys will not register) or by using the user aid configuration keys to turn XMIT OFF (strap A) while you're in DEBUG mode and scrolling, then turn it back on.

In addition, BUILDER currently doesn't turn XMIT OFF when it exits – you can add the appropriate sequence ("Esc&s0A") to the Reset Sequence in BLDRTERM to cause it to do so.

FILE OPTIONS and PERFORMANCE

Notice the in 4.5 RELATE, several file options have been made available through the MODIFY FILE command (remember we told you to not abbreviate things in procedure files?.... How are all your MODIFY F's doing?). When a permanent RELATE file is created, it has the following defaults:

CLUSTER = 1st unary index (not $LINE)
COMPRESS = YES
CRASHPROOF = YES
DELETE = LOGICAL
SCAN = 10 (contrary to the manual, which says 3)

Each of these can have an impact on the performance of certain operations.

CLUSTER will make directed sequential access (eg: for a join or a print) in the order of the clustering index faster, but there is some overhead associated with attempting to determine where something should be physically located when records are added (eg: for a COPY) which may degrade the performance.

COMPRESS will take up slightly more time when doing any read or write if the data needs to be compressed or decompressed. Compression is designed to save disc space at the expense of CPU time, but in my trials here I found a difference of only one or 2 CPU seconds over 1000 records between compressed (with 3 mostly blank alpha fields). This could be because I'm saving disc reads by compressing my data. My best suggestion is to try it over large portions of your own data and see what your results are. A word on how much space your're saving: only 2 bits are used for any field which is entirely blank. Trailing spaces are removed from alpha fields that are more than zero or 20 characters long.

CRASHPROOF involves several different steps. The first main one, listed in the manual, is that updates are forced to disc for every record. This will have an obvious impact on batch processing (i.e. copy). Another main step is that the BREAK key is disabled while an update is taking place to prevent data corruption. In a copy of 300 records, removing CRASHPROOF from the output file reduced CPU usage from 16 seconds to 5 seconds. This time difference is a result of the FCONTROL MPE intrinsic. In the next release, CRASHPROOF will also control Intrinsic Level Recovery.

DELETE can be either physical or logical. Logical is quicker than physical, since the records are merely flagged, whereas physical actually removes the record and makes the space reusable. Physical deletes can take up to twice as long as logical deletes on a file with no indexes (how many of us have one of those?), but only about 20% more with 3 indexes. The more indexes you have, the closer the numbers will be. Logical deletes, although faster will require periodic REORGANIZES to recover the space.

SCAN indicates the relative frequency with which RELATE will spend extra time looking for "free" space in a file. The number is a percentage of adds and updates. The larger the number, the more likely RELATE is to find and re-use previously used space. If SCAN is set to zero, RELATE will not be able to refill any data blocks, until they have been completely emptied. Note that SCAN should be set to zero if DELETE = LOGICAL as substantial amounts of wasted space cannot typically be generated. An exception would be a file populated with sparse records with COMPRESS = NO and then substantially updated with COMPRESS = YES.

LIVE AND LEARN SOLUTIONS

1. The current index is by FLAG. Hence, RELATE reads serially through the file looking for the next, larger, value of FLAG. It prompts you for the first (and only) record, where the value of FLAG is 1). When you change the value to 99, and RELATE looks for a record with a value higher than 1.... it finds it (again)!

   Beware of changing key values! If the CHANGE command was placed inside a transaction (BEGIN and COMMIT) the problem would not occur.

2. The $LAST function only references records that are in any existing result. Since the condition includes "WHERE TEXT=" " AND", then the only records to be returned will have, at a minimum, blank text (plus whatever follows the AND). $LAST says: what is the value of the last record returned. Since all records that could be returned will have blank TEXT, it is never the case that TEXT is non-blank, so no records meet the full criteria. Additionally, in SELECT commands with joins the order of expression evaluation is difficult to determine and may change as the sizes and indexes on the files change.

   Be aware of the way $LAST operates.

RELATE^{tm}/3000

RELEASE NEWSLETTER - VERSION 4.5

TABLE OF CONTENTS

MENU SYSTEM

RELATE

The RELATE 4.5 release contains significant internal changes designed to increase the reliability and flexibility of the product.

In summary, the changes include the following:

√ * The elimination of RELATE index files by combining the index data with the actual records in the data file. Filenames may now be 8 characters long. MPE EOF no longer reflects RELATE EOF.

* More accurate distribution data is maintained on indexed fields.

* Constraints have been relaxed on the maximum number of records and indexes.

* Optional crashproof access is available for RELATE files.

* Optional data compression and primary key clustering is available for RELATE files.

* Improved multi-user support has been provided.

* Privileged mode files can now be created by Account Librarians.

√ * Fields can be added to RELATE files without copying the data.

* The CHANGE command will now function on files opened in SHARED mode.

* Files used in transactions no longer need to be locked.

* The SHOW SETS command has been added to enable interactive viewing of available IMAGE datasets.

INDEX FILE ELIMINATION

In order to facilitate implementation of crashproof access and data compression, the index file and the data file have been merged. Because of the significant internal organizational changes, a conversion operation must be performed on each file prior to being used under RELATE 4.5. The conversion will change the MPE file code on RELATE files from 635 to 638. The FIX FILE command has been provided to perform the conversion; this is described later in this document.

Because an index file is no longer required, RELATE filenames may now be eight characters long and may end with a "Q".

RELATE now performs its own blocking and unblocking, so all files are created with a record size of 1024 words. The MPE end-of-file on a RELATE file will no longer indicate the actual count of records in the file; instead, RELATE maintains this count in a control block within the file. The MPE end-of-file will always be at the end of the most recently allocated extent. Allocation of a new extent will immediately be followed by a forced disc update of the file label. Thus, the MPE end-of-file can never fall below the logical end-of-data within the file. This technique uses no more disc space than moving the end-of-file ahead record by record and reduces the number of updates on the file label to a maximum of 32 (once for each extent).

The elimination of the index file will also eliminate problems caused by storing, restoring, renaming or other MPE operations which were applied to the data file but not performed on the index file. The reduced file count will improve MPE performance during STORE and RELOAD operations and RELATE OPEN/CLOSE commands.

INDEX SUPPORT

Prior releases of RELATE had a limitation of 65535 records in each index file. This equated to between one and two million records in the data file (depending on the number of indexes defined and the size of each key). This restriction has been removed, allowing files containing gigabytes.

Up to 30 indexes may now be defined for each file. In addition, RELATE will now maintain distribution data on all indexes and all fields in each index. Prior releases could only maintain such information on indexes one through five and the first three fields in each index. The improved accuracy should allow the SELECT command to make better optimization decisions.

The speed of index creation (particularly UNARY indexes) has been improved.

CRASH PROOF ACCESS

Access to RETENTION=TEMPORARY or RETENTION=PERMANENT RELATE files can now be crashproofed. When crashproofed, RELATE cannot be aborted while the file is being updated. This will prevent file corruption in all cases but a system failure or ABORTJOB by the operator.

Crashproofing can be controlled with the newly added MODIFY FILE command. The crashproofing facility can be disabled or enabled on a file-by-file basis. Throughput will be better if this capability is disabled.

INVALID DATA DETECTION

Mechanisms that detect invalid data in RELATE files have been implemented now that RELATE 4.5 performs its own blocking and unblocking of data. Each block now contains its own block number and a block type designator. As each block is read, these indicators are checked. If an inconsistency is discovered, RELATE will issue a message and terminate the request. Errors of this type indicate that access may have been made to a RELATE file through non-RELATE routines, that MPE software has erroneously written data to an unopened file or that the hardware (particularly the disc controller) is failing.

RELATE also checks these indicators before updating a block; thus, it is not generally possible for RELATE to write erroneous blocks to a file.

DATA COMPRESSION

Data compression can now be enabled for RELATE files. When compression is enabled, RELATE will no longer store fields containing blanks or zeros. In addition, alphabetic fields containing more than 20 characters will be stripped of trailing blanks before being stored. Upon subsequent retrieval, the data record is expanded to its normal format.

Data compression allows closer packing of data, reducing both the number of disc I/Os required during processing and the amount of storage required for text data and sparsely populated files. Data compression should be enabled except in situations where the size of the compressed records (as described above) is subject to frequent large changes. In these situations, data compression may reduce performance by forcing updated records out of the block in which they were originally stored into overflow blocks. Subsequent access to this data will require an additional disc read.

DELETE and REORGANIZATION

RELATE 4.5 has the added option of either physically deleting data or logically deleting data. The delete mode can be controlled with the MODIFY FILE command. If records are physically deleted, the space used by the record can be reused by RELATE and the REORGANIZE command need no longer be used. If the DELETE mode is PHYSICAL, records cannot be RECOVERED. If records are logically deleted, the REORGANIZE command must still be used periodically to free the space occupied by the deleted records. The REORGANIZE command will no longer maintain the original record number sequence. The DELETE mode can always be changed from PHYSICAL to LOGICAL. The mode may only be changed from LOGICAL to PHYSICAL if no logically deleted records exist in the file.

FIELD ADDITION

Fields may now be added directly to RELATE files with the ADD FIELD command. The command logically adds the field to the structure without touching any data in the file. When a record is updated with data for the new fields, RELATE automatically allocates more space to the record.

PRIMARY KEY CLUSTERING

Clustering is a mechanism for placing logically related data physically close. Logically related data tends to be used as a group and generally in order. Common examples include the detail lines on an invoice and the invoices outstanding for a particular client.

In previous releases, RELATE clustered data around the line number index. As each record was added, it was placed physically close to the previous record. RELATE now has the capability of clustering data based on a user defined index. The index chosen should be the primary key for the file. When data is added to a file, RELATE will attempt to place the data near records containing similar values in the clustering key. Subsequent retrieval of the data in the order of the clustering index will require significantly fewer disc accesses. Since the frequency of reads versus writes is generally very high, throughput will increase.

MULTI-USER ACCESS

Concurrency control has been improved on all file types through the use of record checksums. In addition, logic has been provided for RELATE files that allows RELATE to correctly reposition itself during shared read access without the use of a long duration lock.

As records are read on shared files, RELATE constructs a checksum for the records. An update or delete of a record that is not guarded by a locked file causes RELATE to verify that the checksum on the record has not changed. If the checksums do not match, the operation is disallowed and RELATE will return an error indication. Because of memory constraints, this technique will only work correctly for 50 of the most recently accessed records (on shared files through an updatable path). Because of this safeguard, the CHANGE command will now operate on files opened with SHARED access.

The transaction processing module has been enhanced to use this technique. Files involved in a transaction no longer need to be locked prior to the start of the transaction. During a transaction, the 50 record limit no longer applies. A problem may develop with this technique if a partially COMMITed transaction must be ABORTed. It is possible for a transaction to discover (during the uncommit) that records that must be reset have been adjusted by another user. In this case, an error will result and the uncommit will be cancelled. If this situation cannot be tolerated, the file(s) participating in the transaction should be locked manually as in previous releases.

The checksum verification will be used by the Host Language Interface routines if the following sequence is used:

```
POINT
READ
processing, possibly including points and reads on other records
REPOINT
UPDATE or DELETE
```

or:

```
POINT
READ
processing, with no intervening points or reads
UPDATE or DELETE
```

If a second READ is introduced before the UPDATE or DELETE the validation will only apply to the period of time between the second READ and the UPDATE and the DELETE, not from the first READ.

New HLI calls, RDBREPOINT and BDBREPOINT, have been added to allow the previous sequence.

PRIVILEGED FILES

Files created by an Account Librarian can be specified to be privileged files. Privileged files will block users' attempts to access the data through system utilities or programs.

SERVER PROCESS

When RELATE is installed, there will now be a SERVER job that runs continually on the system. RELATE requires this process in order to run. This significantly improves the speed and accuracy of updates made to files opened by multiple users.

PRIVILEGED MODE

In order to implement privileged files and the SERVER process, the account and group in which RELATE is installed must now have PM (privileged mode) capability. All uses of PM are documented by HP or used extensively by MPE. The release is presently running on Q-MIT, T-MIT, MPE IV, MPE V/P, and MPE V/E. In addition, safeguards beyond those suggested in the MPE documentation have been included in RELATE.

CONVERTING FILES FROM 4.4 TO 4.5 FORMAT

RELATE/3000 release 4.5 utilizes a new internal organization for RELATE files. The new organization supports data compression, crash proof access, improved multi-user support and larger file sizes. Before RELATE 4.5 can access RELATE files, the files must undergo a conversion. A FIX FILE command has been implemented to perform the conversion.

The FIX FILE command will convert a fileset at a time. The fileset can represent at most all files in one account. For each file, RELATE will open the existing file, create a new file with the same structure, copy the original file, apply any indexes, purge the old file, and save the new file. The command will display status information as each file is converted.

The conversion process should only be run after the files being converted have been placed on tape (by a STORE or SYSDUMP). If an error should occur during the conversion the particular file will remain unconverted. The backup copy of the storage could be restored and normal operations could proceed using RELATE 4.4.

Conversion at test sites have indicated that 2 concurrent sessions performing the conversion on a series 40 machine yields the best throughput. Up to 4 such sessions can be run on a series 64.

HELP MODE REMOVAL

The HELP command will no longer place the user in a special HELP mode. Instead, all of the desired options should be listed on a single command line.

HLI PERFORMANCE IMPROVEMENTS

The Host Language Interface has been streamlined to improve its performance. This improvement will benefit all users of the HLI in addition to BUILDER, MENU, and DATAGUIDE.

SHOW SETS COMMAND

The SHOW SETS command will allow the user to list the names of datasets available in an open IMAGE database. Previously, this was possible only at the time the database was accessed.

SYNTAX CHECKING

CREATE now more accurately describes what the error is (as in "The XX keyword is duplicated" instead of "Duplicate not allowed") and where the error occurred (pointing to missing punctuation).

REPORT OPTIONS ENHANCEMENT

The REPORT OPTIONS command has had several new keywords added which will affect the following:

VERTICAL SPACING: The SPACING keyword allows the detail lines of the report to be double-, triple-, etc. spaced.

BLANK LINE SUPRESSION: The NOBLANKS keyword will suppress the printing of any line in the body of the report in which all the fields contain blanks.

GLOBAL COLUMN HEADING UNDERLINES: All field headings can now be underlined with a single UNDERLINE keyword.

NOFOOTING: Footings for single data lines can be suppressed with the NOFOOTING keyword.

RESERVING DATA LINES ON PAGE BREAKS: The HEADDATA and FOOTDATA keywords allow the user to specify that a certain number of data lines must always appear on the same page as the group heading or footing.

APPLICATION BUILDER

Major functional enhancements have been made to BUILDER to further simplify the creation of applications.

In summary, the changes include the following:

* A GLOBAL screen has been added, which defines globally available function keys and variables and provides application initialization.
* Range checking and initialization can now be performed in declarations.
* Function key labels are now supported.
* Processing states have been added to simplify application bookkeeping.
* Additional functions and built-in variables have been added.
* The actions of VERIFY and function keys and the operation of VARIABLE sections and REQUIRED variables have been clarified and standardized.
* A RECORD REPOINT command has been added to allow full use of RELATE's checksum capabilities for files open in SHARED mode.

GLOBAL SCREEN DEFINITIONS

A GLOBAL screen can now be declared in the first application file. The screen can contain an INITIAL section, a DECLARATION section, and FUNCTION sections. The INITIAL section is executed before the first screen in the file is accessed. The entries in the DECLARATION section become globally accessible variables. The FUNCTION sections can be invoked from any screen in the application.

DECLARATION ENHANCEMENTS

The DECLARATION section now supports the keywords INITIAL and RANGE. The INITIAL keyword must be followed by the initial value of the variable. When the screen is first drawn this value will be placed into the variable. A RANGE may also be specified for the variable. The range can specify combinations of unique values or ranges of values. If discrete values are specified, and the prefix in the variable uniquely identifies one of the values, the system will automatically complete the content of a variable. These new keywords are also available on the MODIFY VARIABLE statement.

FUNCTION KEY LABELS

BUILDER applications can now make use of function key labeling. The configuration dialogue has been enhanced to allow entry of the start and end sequences required to place text into the labels. On terminals which do not contain positions for labels, the labeling can be turned off or can be made to use the message line.

The text for the labels is obtained from the FUNCTION section header. Any globally declared function label can be overridden by a local function label. If function labels are not defined, the areas are not changed.

SET STATE STATEMENT

Processing states have been added which allow dynamic switching of DECLARATION, INITIAL, FUNCTION and ENTER sections. A new statement (SET STATE) has been added to allow the state to be assigned. When a state is set, the DECLARATION section corresponding to the state is processed, any function keys defined for the state which include labels are drawn, and the INITIAL section for that state (if found) is executed. When the screen is completed, the ENTER section corresponding to the current state is used.

BUILT-IN VARIABLES and FUNCTIONS

A number of additional built-in variables and functions have been implemented:

$CHANGED	Returns 1 if any variables on the current screen have been changed by the user.
$SCREEN	Contains the name of the current screen.
$STATE	Returns the name of the current state.
$TIME	Returns the time of day.

DATE HANDLING

BUILDER will now perform comparisons between dates as dates, not as character strings.

In addition, several date handling functions have been added to BUILDER.

RECORD REPOINT

The RECORD REPOINT command has the same syntax as the RECORD POINT command. Its function is similar except that it is used to relocate a previously read record to preserve the multi-user checksum. This command is only useful for access to an updatable path open in SHARED mode. It verifies that no other user has altered the record since it was read before doing an UPDATE or DELETE.

PROGRAMMING LANGUAGE INTERFACE

Four new routines have been provided to allow direct reading and updating of BUILDER variables and partitions from a subroutine invoked with the CALL PROCEDURE command. BLDRGETVAR, BLDRPUTVAR, BLDRGETCUR, and BLDRPUTCUR make it simple to pass parameters between a BUILDER application and a program.

MANUAL EXAMPLES

The usage examples in the manuals have been significantly expanded.

RECORD NEXT

The RECORD NEXT command has been added to allow quick repositioning of the file pointer without reading data.

GRAF

Major internal enhancements have been made in this release of GRAF. The original device drivers have been replaced by Graphics Kernal System (GKS) routines. GKS is a two dimensional ISO (International Standards Organization) standard which is pending ANSI (American National Standards Institute) approval. GKS will allow us to develop drivers for all HP devices as well as those from third party manufacturers.

The following additional devices can now be used with GRAF:

HP2700
HP2648
QMS1200 (Quality Micro Systems Lasergraphics 1200)

Additionally, support has been enhanced for:

HP7470
HP7475
HP7580
HP7585
HPIB plotters connected to the HP150
HPIB plotters connected to remote terminals through an RS232 interface

CPU usage during plotting has been reduced significantly.

DRAW NEWPAGE

A DRAW NEWPAGE command has been added to allow forms advance on those devices that support such a feature.

SET SPEED

The SET SPEED command has been changed to accept a value ranging from 1 to 100, which represents the appropriate percentage of the maximum speed available on a device.

MENU SYSTEM

A number of minor enhancements have been made to MENU:

PERFORMANCE

A number of frequently called routines have been sped up. This should be particularly noticeable upon entry to screens which display field names.

REPORT/PAGE HEADINGS

Page headings and footings and report headings and footings may now be placed on reports. They may contain the current date, the page number, and several lines of text or the values of fields.

SHARED MODE ACCESS

The SHARED access mode can now be specified when files are opened.

RELATE™/3000

Release Newsletter 4.55A

RELATE/3000
BUILDER
GRAF
CREATE
MENU
DATAGUIDE

CRI INCORPORATED
Advanced Solutions for Management Problems

TABLE OF CONTENTS

RELATE

BUG FIXES

Most known bugs have been fixed, unless noted in the attached PSR.

WHERE/FOR NUMERIC TO ALPHA COMPARISON

A WHERE or FOR clause containing a comparison of an alpha field to a numeric constant, for example:

```
FOR ZONE=2
```

where ZONE is an alpha field containing some numbers and some text data, used to ignore certain types of data in certain operations because the comparisons were tested as alpha rather than numeric. Now, however, the comparison will be treated as numeric, which will cause ARITHMETIC errors on any non-numeric data. This will generally cause the result to be invalid.

To avoid arithmetic errors, either remove non-numeric data from the field or alter the comparison:

```
FOR ZONE="2"
```

USER DEFINED FUNCTIONS (BUG #1396)

The workaround to this bug was to alter your code so that PARMS subscript 7, items N+2 and N+4 referenced PARMS(7) rather than PARMS(7).(0:8). Code will now work as originally documented, so any altered code referencing PARMS(7) must reinstate PARMS(7).(0:8).

LEVEL= ON SELECT (BUG #1309)

Data entry level for all calculated constants in a SELECT command is now assumed to be zero. Therefore, if the copying of the result causes a new file to be created, the field in that file will also have a level of zero.

ENHANCEMENTS

SYSTEM COMMAND

Two new options have been added to the SYSTEM command.

$MODE=openmode

This sets the default mode for opening files in the current session of RELATE. This can be any open mode as described for MODE in RELATE's OPEN FILE command. This overrides the system default.

$MULTIPLE[:Y|:N]

In previous releases, a backslash ("\") included as part of a Relate command was always used by Relate to indicate the beginning of a new command on the same line; for example:

```
OPEN FILE XYZ\SET INDEX NAME
```

This prevented the use of the backslash as text; for example:

```
SELECT TEXT="CHOOSE A\B\C"
```

This new option allows the user to turn off the use of the backslash as a separator for multiple commands:

```
SYSTEM $MULTIPLE:N
```

:I IN HLI

If any RELATE command is passed to the RELATE call with a :I switch (CREATE FILE, CHANGE, ADD, REPORT, CREATE VIEW), all text passed through additional RELATE calls is expected to be valid input for that command, up to and including the response to terminate input. The question being posed by the system can be determined by looking at word 18 of the cursor.

This will also allow input to be included in BUILDER code, so that, for example, SUBMIT REPORT is no longer required for generating a report.

SHOW INDEX:S

The :S suppresses the display of all information about the index except for the list of fields in the index.

EDITOR ACCESS

If a FILE equation for RDBEDIT exists, using :EDITOR from RELATE will invoke the specified editor. If an invalid program is specified, HP's editor will be used.

TERMINAL OUTPUT

LABEL, PRINT, and REPORT, when used from the HLI (or BUILDER), no longer automatically prompt the user to align the forms. The application programmer should now control this.

MESSAGE CATALOG TEXT

All text in Relate's message catalog (RDBCAT and RDBECAT) has been formatted in upper and lower case, instead of all upper, to more closely resemble English sentences. This affects not only error messages but also informational messages such as "5 lines printed."

EXECUTE FILE

To be more compatible with the Ada version of the RELATE product, the EXECUTE command now recognizes the syntax:

```
EXECUTE [FILE] filename
```

900 SERIES SUPPORT

This release will run on HP Series 900 machines.

PARTITION/CURSOR

WORD 19	The word length of the record in the current path, including two words for $LINE if it exists on the current path.
WORD 29-30	Upgraded to also include the number of records output on the last Create report.
WORD 41-42	Doubleword quantity of arithmetic errors (overflows, underflows, etc.) ignored during the last RELATE command.
WORD 46.(9:1)	=1 if last command (or any command in a procedure file invoked by the last command) set $CANCEL:Y.

HLI REVISION LOCK

The first RDBINIT/RDBINITX call from the HLI (or during initialization of the RELATE process during BUILDER startup) will verify that the version of the HLI being used matches the current installed version of RELATE. If they do not match, error 50 is returned, RELATE is not initialized, and no other HLI calls can be made (including RDBERROR). We suggest that all programs be modified to check for error number 50 in the first word of the cursor and take appropriate action (ie.terminating the program).

RDBLIST CALL IN HLI

A new call, of the format:

```
RDBLIST (cursor,rows,columns,text,cctl)
```

has been added to allow data from the user's application to be placed into the RDBLIST file. The OPEN RDBLIST command must be executed prior to this call. The text is broken into the specified number of rows and columns with the indicated carriage control (matching FWRITE's control) for each record. A BASIC call is also provided:

```
BDBLIST(cursor,text[,cctl])
```

RELATE REFERENCE MANUAL

Manual update pages are included for the above enhancements. Update pages for corrections or significant clarifications in the manual are also included.

BUG FIXES

Most known bugs have been fixed, unless noted in the attached PSR.

OBSOLESCENCE

SET OPTION INTERFACE will no longer be supported.

ENHANCEMENTS

TIMEOUTS

TIMEOUT sections can now be included in each screen and in the GLOBAL section. TIMEOUT sections begin with a delimiter line of the format:

```
delimiters TIMEOUT
```

TIMEOUTS are set with the command:

```
SET OPTION TIMEOUT=seconds
```

The TIMEOUT value applies only to the current screen or, if the SET OPTION is in the GLOBAL INITIAL section, to the first screen in the application.

When the user fails to press a key for the indicated number of seconds, including responding to PROMPTS, the TIMEOUT section in that screen will execute. If there is no TIMEOUT section in that screen, the GLOBAL TIMEOUT section will be executed. If there is also no GLOBAL TIMEOUT section, an error will be generated.

To turn off TIMEOUT checking once it has been turned on, issue

```
SET OPTION TIMEOUT=0 or SET OPTION TIMEOUT=NONE
```

VARIABLE OPTIONS

Two new variable options can be referenced in either a DECLARATION section or MODIFY VARIABLE command.

ECHO=YES|NO|ON|OFF

If this option is NO (or OFF), then anything typed into this variable will not display on the screen, whether the variable is in a LAYOUT or a PROMPT. The default is YES (or ON). This can allow endusers to type a password without it appearing on the screen.

SUB_ENHANCE=enhlist

This determines the enhancement for a variable at the current value of $SUBSCRIPT. It is most useful for array variables, although non-array variables are assumed to have a subscript of zero. If used in a DECLARATION section, subscript zero of the specified variable will have its enhancements changed. BLINK, INVERSE, HALFBRIGHT, UNDERLINE, NONE, or any valid combination can be used.

SET OPTIONS

Three features have been added to the SET OPTIONS command.

SET OPTION TIMEOUT=seconds

This is described in the TIMEOUTS discussion.

SET OPTION VARIABLE=LOCAL|GLOBAL

Determines whether variables that are not listed in a screen's LAYOUT or DECLARATION section, but are used in the screen processing, should be LOCAL or GLOBAL by default. If this command is not used, variables are considered to be GLOBAL. Two examples of how this can affect code:

1. A variable XNAME is declared in SCREEN1 and a CALL SCREEN SCREEN2 is performed. SCREEN2 does not list the XNAME variable in its LAYOUT or DECLARATION, but does a RECORD READ on a file containing a field called XNAME. If VARIABLES=GLOBAL, then a value will be read into XNAME. If VARIABLES=LOCAL, the value of XNAME will remain unchanged.

2. Same setup as before, but now SCREEN2 contains the statement:

```
IF XNAME="FRED"
```

If VARIABLES=GLOBAL, the value of XNAME will be used to compare. If VARIABLES=LOCAL, a syntax error will occur since the XNAME variable has not been defined in SCREEN2.

SET OPTION MESSAGE=CLEAR|KEEP

CLEAR causes anything displayed in the message line to be cleared as soon as a tab occurs or RETURN, ENTER, or a function key is pressed. KEEP causes anything displayed in the message line to remain until something else is placed there. The new setting remains in effect for all screens until it is explicitly changed. KEEP can prevent function key labels from being displayed on terminals where the labels are displayed on the message line. The default is CLEAR.

ERASE SCREEN

The ERASE SCREEN command clears the entire display screen, so that the user will be viewing an entirely blank screen. The LAYOUT is redrawn the next time an automatic or explicit REFRESH occurs.

ARRAY TERMINATION

The EXITARRAY command has been provided to allow the forced early termination of an ARRAY loop. Normally, an ARRAY loop will terminate only after the highest element of an array has been referenced, or after no array variable has been referenced. In either case, the final loop continues executing commands until the ENDARRAY is reached. Setting $SUBSCRIPT explicitly to a high value does not, by design, terminate the loop; $SUBSCRIPT will be reset to its proper value when the ENDARRAY is reached.

EXITARRAY causes the rest of the ARRAY loop to be ignored and execution to continue with the command immediately following the first ENDARRAY after the EXITARRAY. **We suggest that this command only be used after careful consideration, as**

most BUILDER commands automatically alter their operation slightly to deal with operations occurring within an array loop. Call CRI Customer Support for more information.

FUNCTIONS and BUILT-IN VARIABLES

$RANDOM (seed)

Calculates a random number. If a non-zero number is fed to this function, it will initialize the $RANDOM function seed for additional calls. If the same non-zero number is fed to $RANDOM, it will initialize an identical series of "random" numbers. It is suggested that the first call be something like $RANDOM ($TIME). Additional calls should use 0 (zero) as the number passed. $RANDOM(0) returns a random number between zero and .999...

$JCW(jcwexpression)

Returns the value of the JCW whose name is a result of the expression. The JCW is set with MPE's SETJCW command. For example, if the command SETJCW TERMTYPE=3 had been issued, the "3" can be retrieved with:

```
X:=$JCW("TERMTYPE")
```

or:

```
JCWNAME:="TERMTYPE"
X:=$JCW(JCWNAME)
```

$INFORMATION

Contains any text included with the INFO= parameter. For example:

```
:RUN BUILDER.PUB.RELATE45;INFO="STARTUP"
```

would place the text "STARTUP" into $INFORMATION.

$READ (parexpr, fldexpr, keyexpr [,keyexpr [,...]])

Returns the value(s) of the indicated field from the record designated by the key expression in the current path of the indicated partition. Parexpr is an expression that should evaluate to the name of an existing partition containing a current path. Fldexpr is an expression that should evaluate to the name of a field in that path. Keyexpr is an expression containing the value to be searched for in the current index.

If more than one record is found to match the given key values, the values for the requested field will be taken from all qualified records and appended together, with a space between values, up to a length of 1500 characters. Trailing blanks will be removed from each value before the append is performed.

The $READ will read all qualified records and leave the record pointer positioned at the record after the last qualified record.

Example:

```
*** INITIAL
        NOTE  each of these files contains the fields ERRNUM,
        NOTE SEQUENCE, and TEXT. There may be several records,
        NOTE with different sequence numbers, for each error number.
```

```
        CREATE PARTITION FRENCH
                OPEN FILE FRMSG
                SET INDEX ERRNUM, SEQUENCE
        CREATE PARTITION GERMAN
                OPEN FILE GERMSG
                SET INDEX ERRNUM, SEQUENCE
        PROMPT "WHAT LANGUAGE?", LANG

    *** DECLARATION
        LANG; LENGTH=10; UPPER; RANGE="GERMAN", "FRENCH"
        PARTITION_NAME; LENGTH=15

    *** ENTER
        IF some sort of error
                NOTE This error is application error number 2750
        SCROLL $READ(LANG, "TEXT", 2750)
                NOTE Or, to do substitution on the message:
                NOTE SCROLL $TEMPLATE($READ(LANG,"TEXT", 2750))
```

FILE INTERFACE

To make access to specific fields in a file easier by eliminating the need to rename variables or fields, the VARIABLES= clause has been added to the RECORD READ, RECORD UPDATE, and RECORD ADD commands. The syntax is:

```
;VARIABLES=varname|(varname,fldname)[,varname|(varname,fldname)][,...]]
```

varname:
Optional. If specified by itself, the varname must be both a valid BUILDER variable and a valid fieldname in the current file. If specified with a fldname, the varname must be a valid BUILDER variable which will be associated with the field specified by fldname.

fldname:
Optional. The fldname must be a valid fieldname in the current path. The same fieldname is allowed to appear more than once.

If the VARIABLES= option is used, only those BUILDER variables listed will have values read into them.

For example, to read from the field PARTNO into the variable PNO, from the field DESC into the variable DESC, and ignore all other fields in the file:

```
RECORD READ;VARIABLES=(PNO,PARTNO),DESC
```

GLOBAL LINEDRAW

The GLOBAL delimiter line has the format:

```
*** GLOBAL [LINEDRAW=linechar]
```

The character specified will be used as the linedrawing character for all screens unless a different LINEDRAW character is specified on the LAYOUT delimiter line.

EDITOR ACCESS

If a file equation for RDBEDIT exists when DEBUG mode is invoked, choice 1 for EDITOR will immediately bring up that editor; for example:

```
:FILE RDBEDIT=QEDIT.PUB.ROBELLE
```

If an invalid program is specified, HP's editor will be used.

QEDIT™ is a trademark of ROBELLE Consulting Ltd.

HLI CHANGES

Enhancements to the HLI also affect BUILDER. See RELATE enhancements:

:I in HLI (also see below)
TERMINAL OUTPUT
HLI REVISION LOCK

:I IN BUILDER

See RELATE enhancement ":I IN HLI".

If a :I is used in BUILDER, it does not alter the standard process of BUILDER first performing substitution, then determining if the line is a valid BUILDER command,on every line. Only a double slash ("//") will be passed directly to RELATE if a :I sub-prompt is pending. This means that, if you want to use text that would duplicate a BUILDER command as input to a command with :I, it must be enclosed in quotes ("). For example:

```
ADD:I
BREAK
```

will cause BUILDER to break into DEBUG mode, but

```
ADD:I
"BREAK"
```

will use the text as field data.

Use extreme caution if embedding BUILDER commands within responses to a :I command. Specifically, changing or purging the current partition can have serious repercussios (translation: don't do it).

If an error occurs while a :I response is expected, DEBUG mode will cancel any additional requests for the :I command and proceed as normal.

We suggest that :I commands not be used in DEBUG mode as the prompt/response method can be confusing.

APPLICATION BUILDER REFERENCE MANUAL

The Application Builder manual has been reprinted in whole. Many minor modifications and clarifications have beenmade, new features have been documented, and:

The "Development Commands" BREAK, TRACE, SHOW VARIABLES and SHOW PARTITIONS have been intermingled with the rest of Builder's commands rather than set aside in their on section.

Section 2 has been renamed "Terminal Interface" to better reflect its content.

Additional discussion on running Builder has been added to the newly titled "Running Builder" discussion in Section 3.

The ARRAY VARIABLES discussion in Section 3 has been altered slightly for more clarity.

An additional example of GLOBAL VARIABLES has been added to Section 3.

A new page about RECORD POINTERS has been added to Secton 3.

An observation about efficiency has been added to the STATES discussion in Section 3 and to the Performance checklist in Section 5 (old Section 6).

The Execution Processing Loop in section 3 has been clarifed.

Additional text and/or examples have been added in section 4 to RELATE Commands, CLEAR VARIABLE, PROMPT, RECORD REPOINT, RECORD RESET, SET CURSOR, and SET STATE.

A summary of all SECTION syntaxes has been added to the command summary in Appendix A.

Additional discussion on configuring function keys has been added to Appendix B.

Appendix D now discusses and clarifies Builder logging and offline uses of scripts.

GRAF

BUG FIXES

Most known bugs have been fixed, unless noted in the attached PSR, and as follows:

HP2680A

Was erroneously listed as a supported graphics device. It has been removed from the documentation.

3-DIMENSIONAL GRAPHICS

Worked incorrectly in the DRAW LINE command. It is not currently possible to implement. The third dimension is not supported for the DRAW LINE command.

ENHANCEMENTS

SPOOLED HP7550

MPE version V G.A3.01 supports the HP7550 plotter as a spooled device. To reference this from GRAF, use device HP7550S. When a PLOT is done to this device, the plotfile will be spooled and any DRAW NEWPAGEs will result in formfeeds.

FONT TYPE 5

A non-proportionally spaced font, numbered 5, is now supported in all places where a font number can be specified.

RANGES ON DRAW COMMANDS

To be consistent with RELATE data commands and CREATE's REPORT command, ranges can now optionally be included at the beginning of all DRAW commands to select only records in the current index containing the indicated values. See the RELATE Reference Manual for discussions on ranges.

LABEL WIDTH

The DRAW LABEL command will now allow a negative width to be specified for a label. A positive width normally specifies the desired width of each character. A negative width specifies the desired width of the entire label. For example:

```
DRAW LABEL "3 UNITS WIDE"; SIZE=-3, .2
```

DRAW LINE

The thickness of a line can now be specified for the DRAW LINE command:

```
DRAW LINE TO=x,y[;LINETYPE=[linetype][,thickness]]
```

The thickness is an expression indicating how much thicker than a standard line the line will be. For example, a thickness of 1.5 makes it one and a half times the standard thickness; a thickness of .5 means half the standard thickness but may be rounded up to 1 (standard thickness) on plotters that can't draw thinner lines.

DRAW MARKER

The size of a marker can now be specified:

```
DRAW MARKER [[markertype][,scale];] AT =x,y
```

The scale indicates how large the marker should be in relation to a standard marker; for example, .5 is half the size and 2 is twice the size.

PLOT COMMAND

Three new options are allowed on the PLOT command:

```
[;WINDOW=surface]
[;FRAME=surface]
[;ORIENTATION=PORTRAIT|LANDSCAPE]
```

The first two allow the user to generate a large plot once, and then plot portions of the plotfile, using WINDOW, and/or plot it in different locations, using FRAME.

On most graphic devices, the X-axis is along the bottom, wider side of the device and the Y-axis runs along the left, shorter side. This is ORIENTATION=LANDSCAPE, the default. ORIENTATION=PORTRAIT moves the Y-axis to the top, wider side and the X-axis to the left side (rotates the drawing clockwise by 90 degrees).

GRAF REFERENCE MANUAL

The GRAF manual has been reprinted in whole. Several minor modifications and clarifications have been made, and new features have been documented. Appendix C, "Driver Error Codes", is obsolete and has been removed.

CREATE

BUG FIXES

Most known bugs have been fixed, unless noted in the attached PSR.

ENHANCEMENTS

The number of records used in a report can now be returned. If the report is run interactively, the number of records is displayed at the end of the report. The number is also placed into words 29-30 of the cursor/partition.

CREATE REFERENCE MANUAL

No updates are required for the manual.

MENU

Most known bugs have been fixed.

DATAGUIDE

Most known bugs have been fixed.

SUMMARY

This maintenance and enhancement release of RELATE/3000 and its options, numbered 4.55A, supersedes release 4.51A (and release 4.51C of Dataguide).

Your release packet should include the following:

1. 4.55A Release tape, containing fixes for most known bugs in all products, and enhancements for Relate, Builder, and Graf.
2. 4.55A Installation instructions
3. 4.55A Demonstration instructions
4. This release newsletter
5. Product Status Report listing known problems in 4.55A (SAVE THIS-WE'LL BE SENDING UPDATES)
6. 4.55A Final Release Report listing all problems from 4.51A that have been closed for the 4.55A release
7. If you are a Beta Test Site for 4.55A, a Beta Site Status Report
8. Blank Enhancement Request Forms to be used to request new features of CRI products
9. Training schedule
10. Update pages, part #HPR-RF01A, for the RELATE manual part #HPR-RF01
11. New GRAF manual, part #HPG-RF02
12. New BUILDER manual, part #HPB-RF02
13. BUILDER manual errata sheet #HPB-RF02a