REALITY RELEASE 2.0

TABLE OF CONTENTS

RELEASE 2.0

1.0 INTRODUCTION

Release 2.0 includes a new firmware set, new monitor software, several changes in some of the system software, and an updated RPG.

The changes in the firmware were mainly to give it the added capabilities necessary to operate the multidisc monitor. Other changes were made to correct the loss of time problem with the real time clock, improve response to powerfail, correct a bug in the TST instruction, add necessary microcode to allow upgrading to the 2613 communication controller, and to add three new instructions for use at the monitor level of programming for diagnostics.

The new monitor software makes it possible to run Reality with more than one disc drive. Not only does this allow additional disc storage space but it produces a speed enhancement by allowing overlapped seeks for drives on the same controller and in addition, overlapped transfers for drives on different controllers. The software can handle one or two drives per controller with one through four disc controllers. Also provided is a new scheduling algorithm for the disc resource which can be tuned by the user. Finally, the software now has the ability to configure itself at the time a bootstrap is performed so the same tape may be used with any hardware configuration that is currently supported.

The new system software provides the ability to do multiple reel file saves and restores with or without tape labels. File restore frame limits may now be optionally supplied at the time of the file restore. Some problems in the 1.2 print spooler have been corrected. A few new verbs have been added: WHO, BLOCK-TERM, T-RDLBL, BLOCK-PRINT, and :MSETUP. The time which prints in page headings, logon and logoff messages, etc., has been modified to include seconds. The overflow handler has been modified to correct a problem which occurs when the overflow table becomes full.

This release contains version 2 modification level 9 of RPG. It replaces version 1 modification level 10 which was on release 1.2. Tables and arrays, inverted print, sorting, and the "nolist" option are not yet supported with this release. This is still considered a preproduction release of RPG.

1.1 Updating to Release 2.0

1.1.1 General Steps

1. Do a file save of the current system with no coldstart and no abs on the front of the tape.
2. Switch firmware if not already using 2.0 firmware.
3. Run Multidisc Diagnostic on all disc drives.
4. Do a coldstart and file restore (abs and files) from 2.0 SYS-GEN tape.
5. Log on to SYSPROG and do a file restore from the tape created in step 1.
6. Log on to SYSPROG and perform a SYS-UPDATE from the 2.0 SYS-GEN tape.

1.1.2 Detailed Steps

1.0 Log on to SYSPROG
1.1 Mount a scratch tape with write ring and place drive online.
1.2 Do a file save as follows:
Key in FILE-SAVE ⓡ
Respond to ABS DUMP SPECS? with A ⓡ
Respond to DISK INIT SPECS? with F0-m ⓡ
where m is the maximum FID for your system which is covered in section 2.5.1 (It is important to use the "maximum" see step 6.5.11.)
1.3 Wait until file save is complete.

2.0 Skip to 3.0 if you are already running with 2.0 firmware.
2.1 Depress write protect switches on all disc drives
2.2 Turn key on front panel to "off" position
2.3 Remove old firmware and replace with new firmware
2.4 Hold master Reset down and turn key to "on" position.
2.5 Remove write protect from all disc drives.

3.0 See separate instructions for running Multidisc Diagnostic.

4.0 Read section 2 of this document thoroughly before continuing.
4.1 Mount 2.0 SYS-GEN tape and place drive online.
4.2 Place sense switches 1,2, and 4 down.
4.3 Depress Clock, Reset, INT.
4.4 Answer the questions on the terminal on line zero according to section 2 of this document.
4.5 Reply to OPTIONS (X/A/AF/F)= with AF ⓡ
4.6 When abs section is complete the message FRAME LIMITS= will appear. Reply with a ⓡ .

4.7 Wait until file restore is complete.

5.0 Log on to SYSPROG.
5.1 Mount tape created in step 1 and place drive online.
5.2 Key in :DLOAD ⓡ (Note that the colon is part of the verb name.)
5.3 Reply F ⓡ to OPTIONS (X/A/AF/F) =
5.4 Replay ⓡ to FRAME LIMITS =
5.5 Wait until file restore completes.

6.0 Log on to SYSPROG.
6.1 Mount 2.0 SYS-GEN tape and place drive online
6.2 Attach to the mag tape unit (key in T-ATT ⓡ)
6.3 Bypass the file save portion of the tape (key in T-FWD ⓡ)
6.4 Load the new SYSPROG M/DICT (key in T-LOAD MD (N) ⓡ)
6.5 Execute the SYS-UPDATE PROC. (key in SYS-UPDATE ⓡ)

The following message must appear. If not, the system is not loading properly.

THIS PROC SHOULD ONLY BE EXECUTED TO UPDATE TO RELEASE 2.0. ENTER YES TO DO SO

6.5.1 Any answer other than YES ⓡ will cause an immediate exit to TCL.

An answer of YES ⓡ should cause the following display:

6.5.2 UPDATE ERRMSG FILE? (Y/N)

The answer of Y ⓡ will first clear and then load the ERRMSG file and produce the following message:

NOW LOADING ERRMSG FILE

6.5.3 UPDATE NEWAC? (Y/N)

The answer of Y ⓡ will first clear and then load the NEWAC file and produce the following message:

NOW LOADING NEWAC FILE

6.5.4 UPDATE OSYM? (Y/N)

The answer of Y ⓡ will first delete and then create a new OSYM file with modulo and separation 53,1 and then load the OSYM file and produce the following message:

NOW LOADING OSYM FILE

6.5.5 UPDATE PSYM? (Y/N)

The answer of Y ⓡ will first delete and then create a new PSYM file with modulo and separation 37,1 and then load the PSYM file and produce the following message:

NOW LOADING PSYM FILE

6.5.6 The MPSYM file will then be automatically created with modulo and separation 17,1. It will then be loaded from the tape producing the message:

NOW LOADING MONITOR PSYM

6.5.7 LOAD USER-MODES? (Y/N)

The answer of Y ⓡ will cause the USER-MODES dictionary and file to be created with modulo and separation 1,1 and 7,1 respectively. The file will then be loaded resulting in the following message:

NOW LOADING USER-MODES DICTIONARY
NOW LOADING USER-MODES FILE

6.5.8 LOAD BLOCK-CONVERT FILE ? (Y/N)

The answer of Y ⓡ will cause a dictionary type file to be created with a modulo and separation 7,1. The file will then be loaded producing the following message:

NOW LOADING BLOCK-CONVERT FILE

6.5.9 UPDATE SYSTEM-MODES? (Y/N)

The answer of Y ⓡ will first clear and then load the SYSTEM-MODES dictionary and file and produce the following messages:

NOW LOADING SYSTEM-MODES DICTIONARY
NOW LOADING SYSTEM-MODES FILE

6.5.10 DO YOU WANT RPG OBJECT TEXT SPACE ASSIGNED? (Y/N)

The answer of N ⓡ will take you to step 6.5.13.

The answer of Y ⓡ to the preceding question will generate the following request:

6.5.11 HOW MANY FRAMES DO YOU WANT?

The response to this question should be determined by the user on the basis of the following ratio:

1000 source cards requires approximately 50 frames for object text storage.

Thus an answer of 200 ⓡ would assign object text space for about 4000 RPG source cards.

Your system will be in an initial condition at this point if all the steps have been followed carefully, especially step 1.2. All object text that existed is now gone and all RPG source programs must be recompiled. This is because version 1 RPG object cannot be executed by the version 2 RPG run time executive.

The RPG files will now be created, cleared, and loaded. Any files already in existence will cause error message
413 THE FILE NAME ALREADY EXISTS IN THE MASTER DICTIONARY. This is perfectly allright since the clear file will set files already in existence to the same state as those which are newly created.

6.5.12 Progress of the RPG dictionary and file load will be indicated by the following comments:

NOW LOADING RPG-OP FILE
NOW LOADING DICT RPG-ERROR
NOW LOADING RPG-ERROR FILE
NOW LOADING DICT RPG-OBJECT
NOW LOADING RT-HALTS
NOW LOADING RPG-MAP FILE
NOW LOADING RPG-DUMP-MSG FILE

6.5.13 End of job will produce the following message:

SYSTEM LOAD IS COMPLETE

The COLD-START PROC will now be entered automatically. See documentation on page XVIII-6 of the April '74 revision of the Reality Computer System Reference Manual for instructions.

1.1.3 UPDATE-ACCOUNT PROC

The UPDATE-ACCOUNT PROC should be executed prior to using any of the accounts. This is done by logging on to SYSPROG and keying in UPDATE-ACCOUNT ⓡ . The PROC will then prompt you for account name with the message:

ACCOUNT NAME

If you key in ⓡ only, the PROC will exit and print DONE. Anything else is assumed to be an account name which has a Q entry in the SYSPROG M/DICT. The account is updated by copying all entries from NEWAC into the account's M/DICT. You will then be prompted with:

DO YOU WANT THE ACCOUNT UPDATED FOR RPG (Y/N)

A response of Y creates Q entries for the files RPG-MAP and RPG-DUMP-MSG and then prompts for the next account name. A response of N simply causes the prompt for the next account name. Any other response causes the question to be repeated. An account should only be updated for RPG if it was previously set up for RPG using the SETUP-RPG PROC of Release 1.0, 1.1, or 1.2.

2.0 REALITY CONFIGURATION

2.1 Introduction

With release 2.0 the software configuration may be set up in two ways to match the hardware configuration of the system. The first way is to preset the configuration before a coldstart tape is created using the verb :MSETUP which is imbedded in two different procs, SYS-GEN and RE-GEN. The other way is to use any coldstart tape and reconfigure the software at the time the coldstart is performed. Whether the preset configuration is used or the reconfiguration is used is determined by the sense switch setting at the time the coldstart is being performed.

2.2 Presetting the Configuration

The actual presetting operation is performed by the MSETUP program which is described elsewhere. The proc, RE-GEN, is used to call the MSETUP program so a user need not refer to documentation everytime the procedure is necessary. RE-GEN prompts the user for all of the information which is necessary to preset the configuration. To use the RE-GEN proc it is first necessary to log on to SYSPROG and then simply key in RE-GEN(T). The proc first prompts you with the question: ARE MONITOR MODES LOADED? (Y/N) only Y or N will be accepted as legal responses. If the response is N, the following modes are loaded from SYSTEM-MODES : MBOOT, MBUFFERS, MMONITOR, MMONITORX, MMONITORY/N1, MMONITORY/N2, MMONITORZ, MSETUP0, MSETUP1, PIB0, PIB1, PCB0, ABSD, and MSETUP. The proc next prompts for all of the configuration parameters.

NUMBER OF DISC CONTROLLERS (only 1,2,3, or 4 is valid.)

NUMBER OF DISCS PER CONTROLLER (only 1 or 2 is valid)

CORE SIZE (only 16,24,32,40,48,56, or 64 is valid)

MAXIMUM FID (only 9743 through 155903 is valid)

NUMBER OF ENTRIES IN IOQ TABLE (only 2 through 8 is valid)

NUMBER OF DISC READS BEFORE REMOVAL FROM IOQ (any value 4 through 127 is valid)

NUMBER OF COMMUNICATION LINES (only 1 through 32 is valid)

2.3 Using Preset Configuration

To use the preset configuration on a coldstart tape, simply perform a coldstart in the same way as on previous releases.

2.4 Reconfiguring Software at Coldstart Time

To reconfigure software on a coldstart tape that was preset to any arbitrary configuration, it is necessary to put sense swtich 2 down as well as 1 and 4. All of the same questions pertaining to configuration parameters are asked at this time as are asked when the RE-GEN proc is used. The first question is DO YOU WANT TO RUN THE DISC DIAGNOSTIC (Y/N). A response of Y gives control to the disc diagnostic covered in separate documentation. A response of N causes the configuration questions to be asked. Any other response causes the same question to be repeated. The last question asked is IS CONFIGURATION CORRECT (Y/N). A response of Y will cause the rest of the coldstart tape to be read and the OPTIONS(X/A/AF/F)= message to appear. A response of N will cause all of the configuration questions to be asked again.

It is important to note that all communication between the computer and the operator is done on the terminal on line zero. Also, since the communication is being handled entirely by the monitor and not by the much more sophisticated firmware/virtual software combination, the communication is handled in a much more simplistic manner. None of the normal control characters have their usual effect (i.e., control H does not backspace etc.) All responses are terminated with a non-numeric character which is not considered as part of the response unless it is the character "X". If the character "X" is used to terminate a response, the previous question is asked again.

2.5 Further Explanation

2.5.0

Most of the questions which concern the configuration parameters are self explanatory, but a few of them are not or have additional considerations which may not be obvious.

2.5.1

Maximum FID can be determined from the following table: (size of system is in megabytes)

Size of System	5	10	15	20	30	40	60	80
Maximum FID	9743	19487	29231	38975	58463	77951	116927	155903

If a number other than those above is entered, one of two things will happen. If the number is larger than that allowable for the size of the system, the monitor will later allow a virtual process to reference a FID which does not exist which will cause that process to go into an infinite loop trying to reference that FID. The break key will get you out of such a loop in most cases. The other possibility is that the number is smaller than allowable for the size of the system. In this case a virtual process will later break with an illegal FID error when referencing a perfectly good FID. The only way to cure either of these problems is to coldstart the system again and supply a valid maximum FID.

2.5.2 The number of entries in the IOQ table can be 2 through 8. This is one of the parameters with which the user can tune his system. The IOQ table determines how many virtual processes can access the disc during the same time period. If a process does not have an entry in the table and the table is full, then that process must wait until another process has been removed from the table before he can be placed there in order to satisfy a frame fault. How many frame faults a process is allowed before he is removed from the table is covered in 2.5.3.

The number of entires should be determined by the number of discs and the amount of core in your system. The following table is meant to be used only as a guide in setting up the number of IOQ entries. Users may find that one entry more or less may give them better response time depending on the use of the system.

		Amount of Core in 1024 Byte Multiples						
		16	24	32	40	48	56	64
No. of Discs	1	2	2	3	3	4	4	5
	2	2	3	4	4	5	5	6
	3	-	3	4	5	6	6	7
	4	-	-	5	6	6	7	8
	6	-	-	-	7	7	8	8
	8	-	-	-	-	8	8	8

The dashes in the above table represent unbalanced systems for which no number exists that would be the correct number of entries in the IOQ.

2.5.3 The number of disc reads before removal from the IOQ is synonymous with the number of frame faults that will be satisfied for a process before it is removed from contention for the disc. This number can range from 4 to 127. It should be chosen based on the number and type of processes a user will be running. Higher numbers give the edge to processes requiring many frames such as sorts, RPG compile and execution, and assemblies. Lower

numbers give the edge to processes requiring few frames such as data entry or inquiry. Numbers in the range of 16 to 25 work quite well in a mixed process environment.

2.5.4 The number of communication lines is determined by adding the number of communication ports to the number of phantom processes, such as the print spooler. For example, if you have an 8-way and a 4-way communication board, that is 12 communication ports. If you are going to use the print spooler but no user written phantom processes, that's 13 communication lines all together.

2.6 Technical Details

The new coldstart tape contains 37 frames instead of 32 frames. The five new frames are MSETUP0, MSETUP1, MMONITORY/N2, DISC-DIAG, and DISC-MSG. MSETUP1 contains a table which has the preset configuration parameters as well as all of the messages which are used to ask the operator questions concerning reconfiguration. MSETUP0 contains the code which communicates with the operator and which actually does the configuration. The program uses the values in the table in MSETUP1 to decide how to configure the system. The reconfiguration is just a matter of changing the values in the table before the configurator is activated. MMONITORY/N2 is the same as MMONITORY/N1 except that it is set up to use two discs per controller instead of one disc per controller and is in a different frame. MMONITORY/N1 is overlayed by the configurator if two discs per controller is selected.

The bootstrap program MBOOT, which is read in by the firmware, first reads in the twelve monitor frames which includes MSETUP0, MSETUP1, MMONITORY/N2, DISC-DIAG and DISC-MSG. Control is then transferred to MSETUP0. If sense switch 2 is down, the program first determines whether the disc diagnostic is to be executed. If it is, MMONITOR and MMONITORX are overlayed by DISC-DIAG and DISC-MSG and control is passed to DISC-DIAG. If the disc diagnostic is not to be executed, the table in MSETUP1 is set up by communicating with the operator before the configurator is activated. If sense switch 2 is up, the table of configuration parameters is used as it comes in from the tape. Control is then passed back to MBOOT which reads the 24 virtual frames from the tape, overlaying MSETUP0, MSETUP1, MMONITORY/N2, DISC-DIAG and DISC-MSG so still only 16K of memory is needed for the coldstart even though there is 18.5K of data on the tape.

3.0 FILE RESTORE FRAME LIMITS

File restore frame limits may now be entered at the time of the file restore. Just prior to the files portion of the restore (i.e., after the abs portion has completed or skipped), the message, FRAME LIMITS=, will appear on the terminal on which the file restore is being done. A response of ⓡ will cause the prestored limits on the tape to be used. A response of n-m ⓡwill cause n to be used as the base of the SYSTEM dictionary and m to be used as the highest frame which can be allocated to files. Four audits are performed on these numbers: n must be greater than or equal to 32 plus the PCB FID of the last communication line set up for the system; m must be greater than n; there must be a "-" between n and m; and no other characters may be entered after the m. If any of these audits fail the message, FRAME LIMITS, is repeated. The prestored limits may be used after any number of audit failures by responding with ⓡ to the repeated message.

If the prestored limits are used, the lower limit is checked to be certain that it is greater than or equal to 32 plus the PCB FID of the last communication line set up for the system. If it isn't, then the default is used which is the number the lower limit was being checked against.

The following table shows the relationship between the number of communication lines set up for a system and the lowest legal base FID for the SYSTEM dictionary:

No. of Lines	Base FID	No. of Lines	Base FID
1	544	17	1056
2	576	18	1088
3	608	19	1120
4	640	20	1152
5	672	21	1184
6	704	22	1216
7	736	23	1248
8	768	24	1280
9	800	25	1312
10	832	26	1344
11	864	27	1376
12	896	28	1408
13	928	29	1440
14	960	30	1472
15	992	31	1504
16	1024	32	1536

4.0 TAPE I/O ROUTINES

The following describes the changes in the magnetic tape I/O routines; the main changes being the ability to have multiple-reel tape files, and to have tape labels. These capabilities have been incorporated into the FILE-SAVE/ FILE-RESTORE, and the T-DUMP/T-LOAD processor.

4.1 Labels

A tape label may be written at the beginning of each tape reel; the label may consist of up to sixteen characters, and the system adds the time, date, and the reel number. Labels are specified when invoking either the FILE-SAVE or T-DUMP processor; in the case of a FILE-SAVE, the label is written after the cold-start section of the tape, if specified, is written. T-DUMP will cause the label to be written only if the tape is at the load point; it will be ignored otherwise.

The tape label is read by the FILE-RESTORE and T-LOAD processors. The former will always read the first record presented to it (when an 'A', 'AF', or 'F' option is entered); the latter will attempt to read the label only if the tape is at the load point. In the case of unlabeled tapes, the processor will read the first tape record, determine that the tape is unlabeled, and backspace the tape by one record before continuing.

Labeled tapes are created as follows:

1) T-DUMP

T-DUMP ..... HEADER "xxxxxx" ⓡ

The data enclosed in double-quotes following the 'HEADER' connective is used as the label. Omitting the 'HEADER' connective causes unlabeled tapes to be generated.

2) FILE-SAVE

:DDUMP xxxxx ⓡ

The data following the :DDUMP' verb is used as the label. If no data appears, unlabeled tapes are generated.

Notes: The characters ' " ^] [\ should not appear as part of the label. Labels can only be up to 16 characters; data beyond 16 characters will be truncated.

4.2 Multiple Reel Tape Files

When a reel reaches the End-of-tape marker, the system will issue a 'tape rewind' command, and print a message requesting the next reel of tape. An asterisk is printed as a prompt for input. When the next reel has been mounted, the process may be continued by entering a control-shift-O character; all other entries will be ignored. (The process is roadblocked on a READ instruction, and will therefore echo up to eleven non-control characters before re-issuing the asterisk prompt; any control character other than the O^{cs} will cause an asterisk prompt immediately). If the tape unit is not ready or non on-line, the asterisk will be re-issued, and another O^{cs} character should be entered when the tape is ready. The prompts cannot be suppressed, and the input to the asterisk prompt cannot be "stacked" by a PROC.

4.3 Messages Relating to Multiple-Reel Files:

1. MOUNT REEL # xx ; LABEL = label time date *

   The requested tape is to be mounted, and a control-shift-O character typed when the tape unit is ready. If unlabeled tapes are in use, the "LABEL =" etc. part of the message will be suppressed. "xx" is the two-digit hexadecimal reel number required.

2. INCORRECT REEL # *

   This message is returned when the reel number on the labeled tape does not match the requested reel number (or if the first tape mounted is not reel #1). Mount the correct tape and type an O^{cs}. This message cannot appear if unlabeled tapes are in use.

3. INCORRECT LABEL *

   Self-evident; action as above

4. REEL #1 WAS UNLABELED *

   A labeled tape has been mounted, when the first tape reel was unlabeled. Action as above.

5. REEL #1 WAS LABELED *

   Converse of (4) ; action as above.

5.0 PRINT SPOOLER MODIFICATIONS

Two bugs have been corrected. One was a timing problem involving the linkage between the current frame to be printed and the previous frame that was printed and released back to the overflow chain. The result was that once in a while the printing process would get into an illegal condition causing a break to the debugger thus hanging the spooler. The other bug caused the spooler to modify a PIB status other than its own when running on lines 0 through 7 or 16 through 23.

Two improvements have been made. If a person forgets to detach from the line printer after having been attached to it, LOGOFF will now automatically detach the process when the person logs off. The other improvement concerns starting the spooler. Previously any account which had the :START-SPOOLER or :INIT-SPOOLER verbs could start the spooler. Now only accounts with SYS2 privileges are allowed to do this. Also, it was possible to start the spooler on a line that didn't exist for a system. It was possible in that way to modify eight bytes in the SYSTEM dictionary or the SYSPROG M/DICT which could have extremely bad effects. Now only lines with a PCB FID lower than the base of the SYSTEM dictionary are legal candidates on which to start the spooler. An attempt to start the spooler on an illegal line will result in error message 330 ILLEGAL LINE NUMBER. Error message 330 has been changed slightly to cover this case as well as the case of using :INIT-LINES to try to set up more lines than is allowed.

Also some modifications have been made in the documentation on the spooler which appeared in release 1.2. The revised version appears here in total.

5.1 Revised Print Spooler Documentation

5.1.1

Whenever the spooler is awakened to print a file, printing will start immediately after the first frame of data has been accumulated. The word "print" has a new definition because the output device is no longer just the line printer; the device is now assignable. There might not be any device assigned, in which case the file will be deleted when "printed".

The "P" option with TCL-II verbs and the LPTR connective with ENGLISH verbs formerly used to output to the line printer, now means output through the spooler.

5.1.2 DEVICE ASSIGNMENT - The default device is always the line printer. Device assignment is on a user line-by-line basis. Each user can issue an ASSIGN verb at any time, which will remain in effect until a new ASSIGN verb has been issued for that line or until LOGON resets it.

The spooler can spool to multiple devices simultaneously. The options for ASSIGN are as follows:

H - Hold the file on disc. The device is the disc and the file is retained for future printing.

T - Output to tape from wherever the tape is currently positioned. Write an End-of-File (EOF) mark when a close command is received (TCL-II closes all files).

S - Suppress line printer printout.

C - Output to the terminal connected to the communication line on which the spooler is running.

N - No spooling to disc. Output is line-at-a-time directly to the line printer. If N is set, all other options are ignored.

The ASSIGN parameters are input in free form format. Blanks and unrecognizable characters are ignored. Here are some examples of the ASSIGN verb:

ASSIGN
(Resets all the option flag bits the same as LOGON does. Output is spooled to the line printer only and is not held).

ASSIGN N T C
(Output line-by-line to line printer. Ignore T and C).

ASSIGN H T S C
(Hold the file on disc, output to tape, suppress line printer printout, and output to console connected to the communication line on which the spooler is running).

ASSIGN S
(Any file will be deleted when it is next "printed").

5.1.3 HOLD FILE - The spooler has two main queues. The first is a 32 entry queue which contains information on each currently open print file. An open file is one that has not been closed and is still having frames of print output added to it.

The second queue contains entries for up to 32 closed files; i.e., those files that are ready for printing. After a file has started printing it is normally purged from the second queue unless it is flagged as a Hold file. A Hold file will occupy one of the 32 queue entries until it is printed without the hold option being set by ASSIGN. When a Hold file is entered in the print queue it is

assigned a decimal entry number which is printed as follows:

ENTRY NO. 7

A Hold file is kept on the disc until released. A verb to interrogate and edit the hold queue entries is the PRINT-QUE verb. The verb is input without any parameters. A series of messages will then be output, requiring a response. They are as follows:

ENTRY 5	(Queue entry flagged as hold)
LIST FRAME (Y/N)	("Y" reply will list first Frame of the file on your terminal)
PRINT (Y/N)	("Y" reply will print the file according to the presently set ASSIGN parameters. If H is not set the file will be released as it is printed. If only the parameter S is set then the file will be deleted.)
ENTRY 7	(Next queue entry flagged as hold).

After the entire queue has been searched for hold entries the message

[283] END OF PRINT QUEUE

is output to the user.

5.1.4 TAPE FILES - If the "T" option is used on an ASSIGN, then the file spooled will be written to magnetic tape in 500 byte records, starting wherever the tape is positioned. After the last record is written, an End-of-File (EOF) mark is written.

The magnetic tape is never automatically rewound because files may be "stacked" on the tape.

The user must mount the tape (with write-ring), attach the tape unit, and rewind or position the tape where the file is to be written. The tape must be left attached because the spooler will automatically write on the tape without attaching. This means that the spooler will use the tape in this situation even if another process is currently using it. It is the user's responsibility to see that this does not occur.

5.1.5 SUPPRESS LINE PRINTER - The "S" option on an ASSIGN will suppress line printer output. Normally, the "S" is used in conjunction with "T" and/or "C" options. However, if the "S" appears alone, the file will be effectively deleted since the output device is null and the frames of the file are released.

5.1.6 CONSOLE OUTPUT - The spooler can spool output to a hard-copy terminal or CRT connected to the communication line assigned to the spooler. The "C" option will enable console output. The spooler will output the data regardless of whether a console is actually connected. The line printer parameters (lines per page, etc.) will apply to the console.

Note that the spooler will hang permanently if the "C" option is used and an asynchronous communications channel (2613 or 2614) does not exist for the spooler line. The first character output will cause the spooler to wait forever for an interrupt.

5.1.7 NO SPOOL OPTION - The "N" option specifies no spooling. The tape test, etc., will be bypassed. The data is output a line-at-a-time directly to the line printer. All other options are ignored. This option should normally only be used when the disc is getting full and space is a problem or for forms alignment.

The "ASSIGN N" causes an automatic attachment to the line printer the first time WRTLIN is called. The user must use the P-DET verb to detach the line printer when finished. The line-at-a-time mode is a special mode where the user usually wants to gain exclusive control of the printer for a time in order to print several files or do forms alignment.

If the line printer is attached to another line the first time WRTLIN is called, the following message will be output to the user console over and over until the other line is detached:

LPTR ATTACHED TO ANOTHER LINE

The line-at-a-time mode and the normal spooler mode will blank out the user buffer from OBBEG through OB each time WRTLIN is called unless the NOBLNK option is used when calling WRTLIN.

5.2 Spooler Verbs and PROCS

5.2.1

FORM n m - FORM is a verb which is used for forms alignment on a Hold File queue entry n. The first m lines are output to the line printer. The message

AGAIN? (Y/N)-

is output after each m lines. The response "Y" will output the m lines again. The only restriction is that the m lines must be on the first frame of the file.

The forms alignment uses the line-at-a-time feature of the spooler. The present ASSIGN bits are saved before the alignment and restored afterwards. The spooler routine PPUT is called to output each line.

The PPUT routine will attach the line printer when it is available. The FORM routine detaches from the line printer after "N" is typed in response to AGAIN?

Four error messages can be output with FORM:

[284] INPUT ENTRY MUST BE 1-32
[285] ENTRY IS NOT A HOLD FILE
[286] NO. OF LINES MISSING
[287] FILE BUSY BEING PRINTED

5.2.2

P-ATT- The P-ATT is a verb which attaches the user to the line printer. If the user is attached, only the TCL prompt comes back. If the printer was already attached to another communication line, the following message is returned:

[289] LINE PRINTER ATTACHED TO LINE X

5.2.3

P-DET - The P-DET detaches the user from the line printer if the line is presently attached and an exit taken to TCL.

The LOGOFF routine will automatically detach a line from the line printer if it is attached. A user can cause the spooler to wait if he attaches to the line printer and never detaches or logs off.

5.2.4

P-STAT - P-STAT performs a printer status. One of four messages will be output:

1) *PRINTER READY*
2) *LPTR OFF-LINE*
3) *LPTR PRINTING*
4) *LPTR CABLE OFF*

5.2.5 LOAD-SPOOLER - is a PROC which loads eight frames of the spooler from SYSTEM-MODES. This usually followed by a START-SPOOLER PROC. The LOAD-SPOOLER does not destroy the print queue containing Hold files. However, the START-SPOOLER PROC will clear the Hold file queue if the reply is 'Y' to the reset question. The disc space will not be released in this case.

5.2.6 KILL - This verb unconditionally aborts the current spooler job in execution or the next one to be executed. A Hold file will be preserved while a normal print file will have its frames released.

5.2.7 P-ATT-KILL - The P-ATT-KILL verb unconditionally detaches the line printer from whatever line is currently attached to it. The spooler attaches and detaches from the line printer at the beginning and end of each print file. Thus, a user can normally get attached between spooler print files.

The spooler assumes that it remains attached to the printer once it becomes attached. The spooler unconditionally outputs to the line printer each frame without testing to see if it is still attached.

P-ATT-KILL should only be used under the SYSPROG account and only when a line has inadvertantly remained attached to the line printer.

5.2.8 PRINT-TAPE "string" - This is a PROC which does the following:

1) Attaches to the tape unit.
2) Rewinds the tape
3) Attaches to the line printer
4) Prints one tape file. An optional string can be specified as a starting point.
5) Detach the line printer.
6) Detach the tape unit.

The rewind command can be removed from the PROC if a manual rewind is used. Thus, a user could print multiple files stacked on the tape.

An assembly language routine is called to print the tape. It also attaches to the line printer unit and will spin until attached. The

attach in the PROC will give a message if another user is presently attached to the line printer or mag tape.

5.2.9 EJECT n - The line printer will be attached, n pages ejected and the line printer detached. The number n cannot exceed 10 or else the message [286] NO. OF PAGE EJECTS GREATER THAN 10 will be output to the console and an exit taken to TCL.

5.2.10 PRINT-QUE - Used to interrogate and print hold files (as previously described).

5.2.11 PRINT-HOLD n "string" - The Hold file queue entry n will print according to the current ASSIGN options. The Hold file will be deleted unless the H option is set. For example, to copy a Hold file to magnetic tape without releasing the Hold file, use ASSIGN T H followed by PRINT-HOLD.

Three error messages can be output at this point:

[284] INPUT ENTRY NO. MUST BE 1-32
[285] ENTRY IS NOT A HOLD FILE
[287] FILE BUSY BEING PRINTED

If the option string is used, a search for the string will be made and the output started at the beginning of the string. If the string is not found, the following error message is printed:

[290] STRING NOT FOUND IN HOLD FILE

Thus, a Hold file can be reprinted starting at any point.

5.3 SUMMARY OF SPOOLER ERROR MESSAGES

5.3.1 SPOOLING TO LPTR -

5.3.1.1 *LPTR OFF-LINE*

Line printer is powered off, the off-line switch is in the off-line position, or the paper has broken. The message is printed just as the file is queued for printing. The spooler will process the file as normally, then it will spin and wait for the line printer to go ready.

5.3.1.2 *LPTR PRINTING*

The line printer was busy at the time of status. Another spooled file is being printed. This message does not come out every time because it is timing dependent on line printer state. Thus, the line printer may be printing without getting the message.

5.3.1.3 *LPTR CABLE OFF*

The line printer connector is loose or the cable is bad. The spooler will spin and wait for good status.

5.3.1.4 *LPTR ATT. TO ANOTHER LINE*

The line printer is attached to another line. The spooler will wait until the printer is free before printing. This message is output when the print file has been closed and is ready for printing.

5.3.2 SPOOLING TO MAG TAPE -

5.3.2.1 MAG TAPE ATTACHED TO ANOTHER LINE TYPE (A) TO ABORT OR (G) TO TRY AGAIN (A/G)

The user should go to another terminal and detach the tape unit (or hit break and detach by using debugger) and type 'G' to try again.

The 'A' response will release all the spooled frames and exit to TCL.

5.3.2.2 MOUNT WRITE RING ON MAG TAPE AND TYPE C/R TO GO

This message allows the user to put a write ring on the tape without aborting the job and losing spooled output. Merely, mount the write ring, put the tape at load point and type carriage return to try again.

5.3.3 MISC ERROR MESSAGES

5.3.3.1 *PRINT QUE FULL*

The 32 entry queue of closed print files is full. The spooler will do a release-quantum for 40 times, then try to put an entry in the queue again. Consequently, the message will be repeated until the entry is made.

This queue also holds all of the hold files. Some of the hold files may have to be deleted. It is not recommended to leave hold files on the disc for any long periods of time. Five hundred print characters (including blanks) occupy one frame on the disc.

5.3.3.2 *OPEN FILE PRINT QUE FULL*

The 32 entry queue of open print files is full. These are files that have not been closed and are not ready for printing. Exiting to TCL will close all the files for a line. Loading the program OPNPF from SYSTEM-MODES will clear the queue.

This should only be done when loading the spooler. The LOAD-SPOOLER proc will re-load OPNPF.

5.4 THINGS TO WATCH OUT FOR

5.4.1 Hold file queue entries are destroyed if the answer "Y" is given to the START-SPOOLER (PROC).

5.4.2 One characteristic of the "instant printing" feature of the spooler is that first print file to be printed after the spooler is activated will cause all other subsequent print files to wait on the first one. For example, if a user is the first one to get his printer output and he hits the break key and waits, all the new print files will wait until he either types "END" to the debugger or lets the program run to completion.

Typing "END" will force a close of all the outstanding print files for that user and the partially completed last frame of output will be printed as well as all the other print files (in the order they were queued with other users).

6.0 NEW SYSTEM PROGRAMMER FILES

MPSYM is a new file which contains the permanent symbols used in assembling the monitor software.

BLOCK-CONVERT contains the specifications for printing blocked characters as described elsewhere.

RPG-MAP is a new file necessary for the proper operation of an RPG compilation to create an object map.

RPG-DUMP-MSG is used to create a formatted RPG object program dump when the RPG-DUMP proc is used.

7.0 VERBS

7.1 WHO

WHO is a new TCL-I verb which returns the line number and the account name to which you are logged on. The line number is computed by subtracting 512 from your PCB FID and dividing by 32. The account name is obtained by looking your PCB FID up in the ACC file and returning attribute one. If not found "UNKNOWN" is returned as account name.

7.2 POVF

POVF has been modified to calculate the number of frames in each contiguous block. There are two ways to use POVF. If POVF x ⓡ is keyed in where x is any character, the linked chain is not counted. In this case the first line of output will contain only the FID of the first frame in the chain. If POVF ⓡ is keyed in, the linked chain is counted and the number of frames in the chain is displayed in parenthesis on the first line of output. Regardless of which way POVF is used, every line of output after the first is of the format m n (p) where m is the first frame of a contiguous block, n is the last frame of that block and p is the number of frames in the block.

7.3 TIME

The time now prints in the form HH:MM:SS.

7.4 :MSETUP

:MSETUP is a TCL-I verb which allows a user to preset the configuration parameters before the creation of a coldstart tape. This verb is called by the procs RE-GEN and SYS-GEN. It is called from TCL as follows:

:MSETUP cn,dn,cs,fmax,ioq,imax,ncl

where the parameters are decimal numerics

cn	Number of disc controllers; must be 1 through 4.
dn	Number of discs per controller; must be 1 or 2.
cs	Core size in kilobytes; must be 16 through 64 and a multiple of 8.
fmax	Maximum FID; must be 9743 through 155903.
ioq	Number of entries in IOQ table; must be 2 through 8.
imax	Maximum number of disc reads before removal from IOQ table; must be 4 through 127.
ncl	Number of communication lines; must be 1-32.

Parameters that are to remain unchanged need not be specified (may be null). If a range error occurs, processing is terminated

and the message:

[329] PARAMETER RANGE ERROR AT : xxx

will be returned.

7.5 :START-SPOOLER , :INIT-SPOOLER

These verbs have been modified to check the line number to be sure it is valid before starting the spooler on that line. See section 5.0 for more details.

7.6 BLOCK-PRINT, BLOCK-TERM

These verbs will print characters in a 9 by n block form on either the line printer or the user's terminal respectively. Any ASCII character may be printed. Each word in the input line will cause a carriage return and three line feeds prior to printing the word in block form. Any word containing single quotes (') must be contained within double quotes ("), and vice versa. The surrounding quotes will not appear in the output.

The program uses a file named BLOCK-CONVERT to create the blocked characters. A BLOCK-CONVERT file already exists which contains the conversion specifications for all printable ASCII characters (no lower case alphas, however). With this file characters will be printed as 9 by 12 to 9 by 20 blocks. If it is desired to change the way any character is printed, it is necessary to change the corresponding item in the BLOCK-CONVERT file. The item-id of the item is the character to be converted. Each item in the file must consist of exactly ten (10) attributes; the first must specify in decimal the number of horizontal bytes in the blocked character to be output, (i.e., n of the 9 by n block mentioned above). The second and subsequent attributes provide the conversion specification. These attributes contain one or more values; each value except the last is separated from the preceeding a value mark (VM), hex "FD". The first character of the first value in each attribute must be "C" or "B". These signal respectively the output matrix line of the blocked character begins with a character or a blank. Immediately following must be the number of characters or blanks in decimal. The presence of a value mark indicates a switch from character to blank status or vice versa and must be followed by the number of bytes to be output. The process continues until the attribute mark at the end of the current line. An example of BLOCK-PRINT appears on the next page.

```
TTTTTTTTTTTT   HHHH     HHHH   IIIIIIIIIIII    SSSSSSSSSS
TTTTTTTTTTTT   HHHH     HHHH   IIIIIIIIIIII    SSSSSSSSSSS
    TTTT       HHHH     HHHH       IIII        SSSS
    TTTT       HHHHHHHHHHHH       IIII        SSSS
    TTTT       HHHHHHHHHHHH       IIII        SSSSSSSSSS
    TTTT       HHHH     HHHH      IIII         SSSSSSSSSS
    TTTT       HHHH     HHHH      IIII              SSSS
    TTTT       HHHH     HHHH   IIIIIIIIIIII    SSSSSSSSSS
    TTTT       HHHH     HHHH   IIIIIIIIIIII     SSSSSSSS

IIIIIIIIIIII    SSSSSSSSSS                  AAAA      NNNN    NNNN
IIIIIIIIIIII   SSSSSSSSSSS                 AAAAAA     NNNNN   NNNN
    IIII       SSSS                       AAAAAAAA   NNNNNN  NNNN
    IIII       SSSS                      AAAA  AAAA  NNNNNNN NNNN
    IIII       SSSSSSSSSS               AAAA    AAAA NNNNNNNNNNNN
    IIII        SSSSSSSSSS              AAAAAAAAAAAA NNNN NNNNNNN
    IIII             SSSS               AAAAAAAAAAAA NNNN  NNNNNN
IIIIIIIIIIII   SSSSSSSSSSS              AAAA    AAAA NNNN   NNNNN
IIIIIIIIIIII    SSSSSSSSS               AAAA    AAAA NNNN    NNNN

                EEEEEEEEEEEE  XXXX    XXXX
                EEEEEEEEEEEE   XXXX  XXXX
                EEEE            XXXXXXXX
                EEEEEEEEEE       XXXXXX
                EEEEEEEEEE        XXXX
                EEEE              XXXXXX      ----------
                EEEE              XXXXXXXX    ----------
                EEEEEEEEEEEE     XXXX  XXXX
                EEEEEEEEEEEE    XXXX    XXXX

   AAAA       MMMM       MMMM   PPPPPPPPPPP   LLLL          EEEEEEEEEEEE
  AAAAAA      MMMMM     MMMMM   PPPPPPPPPPPP  LLLL          EEEEEEEEEEEE
 AAAAAAAA     MMMMMM   MMMMMM   PPPP    PPPP  LLLL          EEEE
AAAA  AAAA    MMMMMMMM MMMMMMMM PPPP    PPPP  LLLL          EEEEEEEEEE
AAAA    AAAA  MMMM MMMMMM MMMM  PPPPPPPPPPPP  LLLL          EEEEEEEEEE
AAAAAAAAAAAA  MMMM  MMM  MMMM  PPPPPPPPPPP   LLLL          EEEE
AAAAAAAAAAAA  MMMM       MMMM  PPPP          LLLL          EEEE
AAAA    AAAA  MMMM       MMMM  PPPP          LLLLLLLLLLLL  EEEEEEEEEEEE
AAAA    AAAA  MMMM       MMMM  PPPP          LLLLLLLLLLLL  EEEEEEEEEEEE
```

For example an "X" might be sepcified as follows:

```
001   7                 blocked character is 7 bytes wide
002   C1] 5] 1          output 1 char, 5 blanks, and 1 char
003   B1] 1] 3] 1] 1    output 1 blank, 1 char, 3 blanks, 1 char, 1 blank
004   B2] 1] 1] 1] 2    output 2 blanks, 1 char, 1 blank, 1 char, 2 blanks
005   B3] 1] 3          output 3 blanks, 1 char, 3 blanks
006   B2] 1] 1] 1] 2    output 2 blanks, 1 char, 1 blank, 1 char, 2 blanks
007   B1] 1] 3] 1] 1    output 1 blank, 1 char, 3 blanks, 1 char, 1 blank
008   C1] 5] 1          output 1 char, 5 blanks, 1 char
009   B7                output 7 blanks
010   B7                output 7 blanks
```

Words to be blocked cannot have more than nine (9) characters, and in addition, the total number of bytes (including three (3) blanks separating each blocked character in a word cannot exceed the current line length set by the last TERM verb.)

The following error message will be produced if the corresponding error occurs.

[520] NO DATA FOR BLOCK OUTPUT (A string of characters to be blocked did not follow the verb)

[521] TOO MANY CHARACTERS IN WORD TO BLOCK (more than nine characters specified in a word)

[522] BLOCK CONVERT FILE MISSING OR IMPROPERLY DEFINED

[523] BLOCK OUTPUT WOULD EXCEED PAGE WIDTH (see discussion above)

[524] INPUT CHARACTER 'x' IS NOT IN BLOCK CONVERT FILE

[525] INPUT CHARACTER 'x' IS IMPROPERLY FORMATTED IN BLOCK CONVERT FILE

7.7 T-REW - will now return control to TCL immediately after issuing the "rewind" command to the tape, that is, it will not wait for the completion of the rewind operation. As a result, the message #96 (BOT) will not appear when a T-REW is entered.

7.8 T-RDLBL n - will read and store the label from tape reel number "n" ("n" in hexadecimal), if the tape is at load point. This verb must be used to initialize the internal label storage area, and is needed under either of the following conditions:

a) Data is to be written to a tape (T-DUMP, etc.) starting at other than the load point of reel number one.

b) Data is to be read from a tape (T-LOAD, SEL-RESTORE, etc.) starting at other than the load point of reel number one.

The label data is maintained during a logon session, and therefore, T-RDLBL need be used only once when working on any particular multi-reel tape set. As an example, suppose there is a three-reel file-save, and it is desired to start a SEL-RESTORE from reel #2 (since SEL-RESTORE can be started in the middle of the file-save):

```
   Reel 1                Reel 2                Reel 3
BOT . . . . . .EOT BOT . . . .EOT BOT. . . . . .EOF
                     start
```

If tape #2 reaches EOT, the system must be able to prompt for reel #3; therefore, it is imperative that prior to the SEL-RESTORE, reel #2 is mounted at load point, and the command T-RDLBL 2 r is used to initialize the label save buffer.

7.9 :INIT-LINES

This verb has been modified so it no longer changes the object code of PIB0 and PIB1. It used to zero the end of PIBs bit in the object code as well as in the current PIBs. This was done so the system would be set up for the proper number of lines the next time a bootstrap was performed. With the new method of configuration, this is no longer necessary.

Error message 330 which used to be "ILLEGAL NUMBER OF LINES" has been changed to "ILLEGAL LINE NUMBER" to cover the case of :START-SPOOLER and :INIT-SPOOLER.

8.0 REALLOCATION OF PROCESS WORK AREAS

The executable frames from 1 to 399 are rapidly being exhausted. In order to continue expanding the system software without using the user's area from 400 to 511, it will be necessary to move the process work area. In order to do this with a minimum of disruption to user routines which might use the fact that the FID of the PCB of line zero is 512, a new subroutine has been written to supply the FID. The name of the subroutine is GPCB0 and it returns the FID of line zero's PCB in the accumulator.

It is currently anticipated that the PCB FID for line zero will be another of the configuration parameters that can be entered when a system is coldstarted. In this way users can control how many executable frames are available for their own use.

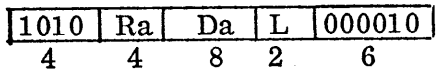
9.0 FIRMWARE MODIFICATIONS

The interrupt structure of Reality was modified to cause more rapid recognition of powerfail and to correct the loss of time problem with the real time clock. This new structure allows the testing of internal interrupts regardless of whether the computer is currently in virtual or monitor mode. Thus a panel interrupt or powerfail is recognized immediately at RNI. Also internal interrupts can now be recognized after any single byte has been processed in the multibyte processing instructions MIID, MIIT, MIIR, and SCD. Since the real time clock interrupt is an internal interrupt, it too gets immediate response and thus cures the loss of time problem. However, it was necessary to change the monitor to check for clock counter overflow. Although the clock counter can now be updated in both virtual and monitor mode, the clock counter overflow can only be accepted as an interrupt while in virtual mode. A side effect of the new interrupt scheme is that the STEP switch on the front panel of the CPU now functions the same as the INT switch when the system is running.

9.1 Other changes to the firmware were made to allow the multi disc operating system to function. The monitor can now attach to a buffer that is I/O busy without getting a monitor error. The automatic disarm on an external interrupt was removed from the firmware and is now being done by the monitor. This was necessary to allow the disc to run on interrupts since the command being sent by the firmware was not compatible with the disarm command for the disc controller. A virtual process is now deactivated on an interrupt from the disc (assumed to be any device address X'10' through X'17'). This is necessary since the virtual process may have a register which is attached to a buffer that the monitor will use during the interrupt processing for the disc.

9.2 A hitherto undocumented instruction has been fixed so it works properly. The TST instruction used to cause unpredictable results. It now works properly. The form of the instruction is as follows:

TST (Test and Set arithmetic condition flags)

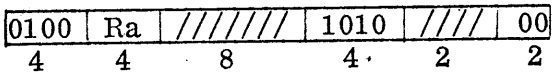


The contents of (Ra, Da) is tested and the arithmetic condition flags (i.e., ZROBIT and NEGBIT) are updated appropriately. The instruction may be used with a half tally, a tally, or a double tally. L is '00', '01', or '10' respectively depending upon the type of operand.

9.3 The changes to the firmware in the area of communication handling was chiefly to allow the software to distinguish between a break key and the other types of unusual status that can be generated by the 2613 communication controller. With the 2614 all unusual status was treated as a break key.

9.4 Three new instructions, HLD, ECS, and ESS have been added to the firmware for use in monitor mode for diagnostics. The forms and functions of these new instructions are as follows:

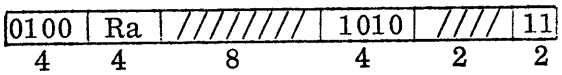
9.4.1 HLD (Halt and Display)



Halts the CPU and gates the eight-bit literal addressed by register RA to the A bus where it can be displayed in the eight least significant indicator lamps of the system panel by depressing the Data select switch.

This instruction is restricted to Monitor level code.

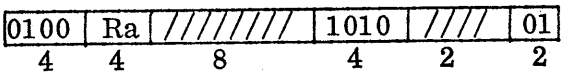
9.4.2 ECS (Enter Console Command Switches)



The status of the eight low-order console command switches is placed in the eight bit byte addressed by register RA. If a switch is on, the corresponding bit in the byte addressed by register RA is set to one. If a switch is not set, the corresponding bit will be set to zero.

This instruction is restricted to Monitor level code.

9.4.3 ESS (Enter Sense Switches)



The status of the four console sense switches is placed in the four most significant bits of the eight bit byte addressed by register RA. The status of a switch is one when the switch is set. The four low order bits are set to one.

This instruction is restircted to Monitor level code.

9.5 The MIIT instruction has changed slightly by virtue of the fact that internal interrupts now cause the instruction to be discontinued and restarted. However, since the reference manual does not describe the execution of the instruction properly for the old firmware, the entire instruction is redescribed below.

MIIT (Move Incrementing Incrementing under Tally Control)

0110	Ra	Rb	0100
4	4	4	4

The contents of the lower half of the accumulator (T0) is read into internal hardware registers. Ra and Rb are each incremented by one and then the byte at c(Ra) replaces the byte at c(Rb). Next the internal hardware registers are decremented. If the resulting value is not zero the increment and move is repeated. If, during the execution of the instruction, an internal interrupt occurs or if the move crosses the frame boundary of a linked frame, the current contents of the internal hardware registers are stored in T0. For this reason the contents of T0 at the conclusion of the move are indeterminable. If T0 is initially zero no operation is performed.

10.0 MODIFICATIONS TO OSYM

10.1 The three new instructions in the firmware have been added to OSYM. They are as follows:

```
HLDR
  P
  G,4,4,16   4,A2;2,X'00A0'

ECSR
  P
  G,4,4,16   4,A2;2,X'00A3'

ESSR
  P
  G,4,4,16   4,A2;2,X'00A1'
```

10.2 Also two errors in OSYM have been corrected:

was	is
BDLEZHHL M BZ:HHLN (*),3	BDLEZHHL M BZ: (*),3
BDLZHHL M BZ:HHLN (*),1	BDLZHHL M BZ: (*),1

These changes were made so forward references can be made to labels in these two instructions.

11.0 SYSTEM SOFTWARE AND SUBROUTINES

11.1 GPCB0 (4,ABSL)

Functional Description

Returns the FID of the PCB for channel zero in the accumulator.

Input Interface

None

Output Interface

D0 Contains FID of PCB for channel zero. High order bits are zero.

Subroutines Called

None

11.2 BLOCK-LETTERS

Functional Description

This program provides the block letter capability (see BLOCK-TERM and BLOCK-PRINT documentation). In addition to its use at the verb level, it may be called as a subroutine (DEFM 2,290). It will format a string of words on the terminal on the printer and return to the caller.

Input Interface

ZBIT if set direct output to the terminal; if not set, direct output to the printer.

IS points one character prior to the first character to be output; end of data is indicated by the character pair SM, Z, if a segment mark is present in the string not followed by a "Z" the string must be terminated by a switch block (SB) X'FB' (see TCL-I interface).

PAGSIZ maximum number of lines per page.

OBSIZE maximum number of characters on each output line.

The following functional elements are used and not restored.

SCZ, SC1 REJCTR, CO, PAGINATE, BASE, MODULO, SEPAR, CTR16, CTR17, SR5, SR6, SR7, SR8, SR9, SR10, SR11, SR12, SR13, SR14, SR15, SR16, SR17, SR18, SR19, SR20, SR21, SR22, CTR18, AFEND, LPBIT, ZBIT

Subroutines called: RETIX, GBMS, NEWPAGE, CVTNIR, WRTLIN

Error conditions: see verb write-up on BLOCK-TERM and BLOCK-PRINT.

11.3 RDLABEL (2, TAPEIO-II)

May be called once by any program to read the label off reel #1; if the tape is labeled, the label is stored in the save area; if not, the "unlabeled tapes in use" flag is set. If the tape is not at the load point, no action is taken. No input interface.

SEGMNT (3, TAPEIO-II)

Used by the FILE-RESTORE processor, to de-block a data from a file save tape. An entire "segment" returned, built up from as many tape records as necessary.

11.4 NEXTTAPE (4, TAPEIO-II)

Used internally by TPWRITE & TPREAD to message the process when an EOT condition is reached, and to setup for the next tape reel. Should not be used by user-programs.

11.5 RDLABELX (5, TAPEIO-II)

As for RDLABEL, except that no check is made to see if the tape is at the load point. This routine is used by the FILE-RESTORE processor (ABSL), to read a label even though the tape may be positioned past the load point. May be used by user-programs, though the implications of doing so should be kept in mind.

11.6 SAVE (6, TAPEIO-II)

This routine saves R13, R14, R15 in internal save areas (defined in TAPEIO-II), and sets up R13 to displacement X'1A6' in the testing control block for use by other routines.

11.7 RESTORE (7, TAPEIO-II)

Restores R13, R14, R15 from save areas in TAPEIO-II.

11.8 RDLABELY (8, TAPEIO-II)

Processes the T-RDLBL verb

11.9 WTLABEL (2, TAPEIO-III)

May be called once by any processor to write a label on reel #1; no action is taken if the tape is not at load point. The label (if any) passed as an input parameter is written to the tape, with the current time and date, and reel number one, added. The label is also stored in the label save buffer.

Input Interface:

IS	R	Points one before the label data, which must be terminated by any standard system delimiter. The label cannot be greater than 16 characters; it will be truncated to 16 if it is. If a null label is submitted, no label is written to the tape, and the "unlabeled tapes in use" flag is set.

Output Interface

IS	R	Points to delimiter terminating label, or to 16 bytes beyond the input position if none is found.

Label save area initialized.

11.10

Internally used by NEXTTAPE

WTLABELX (4, TAPEIO-III)

As for WTLABEL, except that no check is made to see if the tape is not set to load point. This routine is used by the FILE-SAVE processor (ABSD), to write a label at the current position of the tape. May be used by user-programs, though the implications of doing so should be kept in mind.

11.11 Programming notes on Tape I/O Labels

Label format:

L...label data ...	time	date	reel#		
(SM)	(VM)	(AM)	(AM)	(SM)	

The label is stored in the quadrenary control block (PCB+3); displacements to various elements are as below:

Byte Displacement	Type	Description
1A5 (Bit 0)	B	"unlabeled tapes in use" flag
1A6	T	Reel number
1A8	L	Label save buffer (46 bytes)
1D6	L	Label write/read buffer (30 bytes)

Since the tape-IO routines are non-reentrant, internal storage is utilized in TAPEIO-II when an EOT condition is handled by the tape write or read subroutines.

12.0 RPG

12.1 The RPG version 02, Modification Level 09 is substantially improved over the prior release. Among the features now available are subroutines (including the calculation op-codes BEGSR, EXSR, and ENDSR); look-ahead records on input specifications , compiler object maps, output field conditioning and "AND" conditioning of output records, forward defined calculation factors, MVR calculation op-code, support of negative numeric literals, extension of compilation possibilities to large source programs (exceeding 1600 source statements).

The following features are still not available (but will be soon):

Tables, arrays and the calculation op codes LOKUP and XFOOT

*PLACE on output

spread card processing

DSPLY and DEBUG op-codes

packed and binary input and output

use of sorted files

suppression of compiler printed listing

12.2 The use of the COMPILE proc (as opposed to the :COMPILE verb) is still recommended but more than one compilation may be performed under a single COMPILE statement, e.g.

COMPILE PROGA PROGB ⓡ

will cause both programs to be compiled before control is returned to TCL.

12.3 A cross reference of symbols and program branch points is now available. The following rules apply:

If the listing option (col 11 on the control card [card type H]):

1. = blank, a cross reference of the program branch points is produced;
2. = T, all cross reference is suppressed
3. = C, both branch points and symbols are cross referenced

Note: selection of the third option will increase the compile time; option 2 will have minimal time effects.

12.4 Source program items cannot exceed 32,267 bytes (approximately 1000 source cards). If a larger program is needed, the compiler will logically link from one source item to another under the following restrictions.

1. The last line in each item except the last item must be an "item link"; the format is given below.

2. The first line of the source program may not be an item link.

3. Source items subsequent to the first one may consist of only an item link.

The format of the item link is as follows:

1. Columns 1-3 must be "= = ="

2. The name of the next source item must immediately follow (no blanks may intervene).

3. Comments may appear on the same line but must be separated by at least on blank;

4. The total size of the line must not exceed seventy nine (79) characters

Smaller programs may be sectioned if desired using the above rules.

12.5 A dump facility is now available. The dump may be executed in two different ways:

A. A dump of an object program may be obtained using the proc RPG-DUMP.

B. Execution dumps may be obtained by hitting the break key and typing (assuming SYS2 privilege) G 351.D.

Ordinarily, the dump will consist of the control information, file data, calculation data, variable storage information, and the data in hexadecimal format. However, there are options to suppress some or all of the data.

H suppresses control data
F suppresses all file data
I suppresses detail input data
O suppresses detail output data
C suppresses all calculation data
D suppresses detail calculation detail data
T suppresses total calculation detail data
S suppresses subroutine calculation detail data
V suppresses variable storage data
X suppresses the hexadecimal dump

If you went to dump an object program, type the proc name followed by this object program name; e.g.

:RPG-DUMP PROGA ⓡ

The options, if any, should be entered in parameters and follow the program names.

Most users will probably never be interested in an object program dump, but if a run-time abort (or other strange condition occurs), you should provide a dump when reporting the problem to Microdata. One of two procedures may be followed:

I. If the account executing the RPG-I program has SYS2 priviledges (and the "START FIRST CYCLE" message has been output), exit the debugger with G351.D ⓡ

II. If the account does not have the appropriate privileges exit the debugger with END ⓡ ; and immediately execute the dump by typing EXEC-DUMP ⓡ . Note: this verb must be executed immediately; this means no other action after the END.

III. If the first cycle message has not appeared, use the RPG-DUMP proc.

In either of the first events, the message

DUMP OPTIONS?

will appear. All of the above options may be specified (without commas or spaces) and, in addition, the terminal characteristics may be changed (to change forms, for example) by specifying the new line size followed by "-" followed by the new page size. E.g., 120-58 indicates print lines of 120 characters and 58 lines per page. Note: both the line size and page length must be specified.

The dump has been designed to anticipate illegal situations, but aborts may occur. In that event, type END ⓡ and execute a normal RPG-DUMP. The dump may not execute properly on a version 01 object program.

12.6 The RPG-OBJECT item has been expanded to include, for successful compiles only, the date of compile and the beginning and ending time of the compilation. For example,

:LIST RPG-OBJECT 'SAMPLE' ⓡ

will display the pointer entry for the example.

12.7 Object programs may be deleted with the proc DEL-OBJ. The example program may be deleted by typing DEL-OBJ SAMPLE ⓡ

Note the object program name must be specified.

12.8 In the event of an abnormal compiler abort, the following procedure should be followed:

1. Note the abort address.

2. Type G240.7 ⓡ

   This should cause the following message to appear followed by a dump of the return stack. If not, type END ⓡ to exit to TCL.

   R01 PREMATURE END OF COMPILATION
   XXX.XXX XXX.XXX XXX.XXX XXX.XXX

3. Note the addresses in the return stack.

4. The compiler will exit to TCL after transmission of the return stack dump.

12.9 Files

12.9.1 RPG II,version 02, modification level 09 requires several files for its operation. There should be only one copy of each of the following files in a given system:

RPG-OP (contains needed data to compile calculation op-codes)

RPG-ERROR (contains compiler error messages)

RPG-OBJECT (contains object text data and provides the interface between the compiler and the run-time execution).

RPG-MAP (contains data for producing object maps)

RT-HALTS (contains run time halt messages)

RPG-DUMP-MSG (contains formats for the formatted object dumps)

12.9.2 The D (dictionary) entry for each of these files should appear in the SYSPROG (system programmer) account.

12.9.3 Each user who will compiler RPG II programs must have a Q (synonym) entry for the files RPG-ERROR and RPG-OP in his master dictionary.

12.9.4 Each user who will execute and/or compile RPG II programs must have a Q (synonym) entry for the file RPG-OBJECT in his master dictionary.

12.9.5 Each user who will compile RPG II programs must have a file named RPG-PROGRAMS to hold the source program.

12.9.6 Each user who will execute RPG II programs must have a Q (synonym) entry for the files RT-HALTS and RPG-DUMP-MSG.

12.9.7 The system error message file (ERRMSG) must contain the RPG compiler and utility error/information messages. These messages are distinguishable by the presence of the character 'R' as the first character of the three character item name.

12.10 Verbs and Procs

Each user who will compile RPG II programs must have the COMPILE proc and the :COMPILE verb. Each user who will execute RPG II programs must have the EXEC proc and the :EXEC verb, and should have the RPG-DUMP proc as well as :RPG-DUMP and EXEC-DUMP verbs.

13.0 USER-MODES

The file, user-modes, on account SYSPROG contains some assembly language routines of general interest. These include the PROC extensions developed by INFACT, inc., a cursor control program for the standard Microdata terminal, and a lock/unlock extension to procs.

Note: These modes are distributed by Microdata but they are not part of the standard Reality software nor are they covered under the standard software maintenance. All of the programs have been extensively tested; however, any problems should be directed to Microdata and the problems will be passed on to the program author.

The names of the modes in the USER-MODES file may be listed from the SYSPROG account with:

LIST USER-MODES ⓡ

13.1 Loading the Modes

The modes may be loaded using the following:

MLOAD USER-MODES * ⓡ

Note: that all of the modes object code is located in frames 400 and above and the user is cautioned to check for possible frame assignment conflicts with any other user written assembly programs. Modes may be selectively loaded using:

MLOAD USER-MODES 'name' ⓡ

13.2 Using the CURSUR mode

This mode is called from a PROC and provides the capability to position the terminal cursor and output text and special characters to the terminal. The calling sequence is:

```
:
:
U01A6
  control list
```

Control List Options (Separated By Commas)

(Col#,Row#)	Col# = 0-79, Row# = 0-23
"-TEXT-"	Output Line of Text
B	Ring the Bell
C	Clear the Screen
INN	Output Integer Char. NN
XNN	Output Hex Char. NN

NOTE - Once Each Control Element Has Been Recognized, The Cursur Control Prgram Will Skip To The Next Comma (Or End Of The Proc Line). Thus, Commentary May Be Included In The Line For Better Readability.

If The Last Character Of a Cursur Control Line Is A Comma (,) Then Additional Cursur Control Elements May Be Written On The Next Line Of The Proc. This Process May Continue For Many Lines.

Example:

```
U01A6
BELL,CLEAR,(10,10),"*" MARK CENTER,
X12 TURN PUNCH ON,
(0,2),"THIS IS A COMMENT "
```

13.3 Using The LOCK/UNLOCK Mode

This mode provides a simple method to limit the execution of a critical section of a proc (such as a file update) to only one user at a time. The PROC user exits are:

U0191	to	Lock
U1191	to	Unlock

Once a lock is set by a user, other users attempting to set the lock will be suspended until the lock is removed (unlocked) by the original user. A long lockout delay will be indicated by the suspended user's terminal displaying a '#' character periodically Note that if a PROC programming error or an abort (or BREAK) prevents the UNLOCK from being executed, subsequent calls to LOCK will never complete. To clear this situation, the following verb should be added to the SYSPROG master dictionary:

```
      UNLOCK
001   P
002   2191
```

Executing this verb will either unlock the lock status or display a message identifying which terminal line last set the lock. Executing UNLOCK on that line will restore normal operation.

13.4 Using The INFACT Modes

The INFACT modes consist of:

TRANS2
KERNEL2
POOL2
CALC2
FORMAT2

The complete description of features, use, and examples follows.

REALITY User Modes -- Version 2.1 Enhancements

1. SUMMARY

The four user modes described in INFACT Report 74-1 have been revised to provice additional facilities. The modes have been repackaged to provide additional space, and a fifth frame, POOL2 (419) is now required. Some internal routines have been moved from frame 418 to the new frame 419 with a long term objective of moving all internal routines to Frame 419.

2. NEW FACILITIES

Since the initial Version 2 release, the following new facilities have been added to the INFACT user modes:

2.1 Right Justification of conversion format items. In FORMAT2, conversion format items (indicated by an *) are converted as before. In addition the V/TYPE dictionary entry is analysed during the conversion process. If the field is R, RA, or RN the converted data item is right justified in a field whose width is specified by the V/MAX dictionary entry.

2.2 Formatted Output to the Line Printer. A new format item, L, has been added to FORMAT2. The L-format item routes output to the line printer instead of the user's terminal. Output to the line printer is initiated by the L-format item and is terminated by exit from the FORMAT2 mode.

2.3 Placement of text in the Primary Input Buffer.

A new TRANS2 entry point has been added so that constant information can be placed into the Primary Input Buffer. The calling sequence is:

```
U41AD
any-text
(normal return)
```

The argument, "any-text", replaces the current item in the primary input buffer; that is, the item pointed to by the IB register. Note that, if "any-text" contains blank separators, the effect is to shift subsequent PIB entries to the right. If "any-text" is null, the effect is to delete an item from the PIB. A blank value for any-text is not recommended, since it could lead to surprising results.

2.4 Optional Replacement of Blanks with Backslashes. Target designations (see Report 74-2, paragraph 4) can now specify that embedded blanks in an attribute be replaced by backslash (SB) characters. A correlated change in FORMAT2 scans the output buffer and replaces backslash (SB) characters by ASCII blanks. This option is requested by prefixing a target designator by the letter "B". Thus, the following are now legitimate target designators:

BA BP BS BT BV BV

2.5 Verify Non-null Attribute. A new target designator, VA, has been implemented. If the requested attribute is null, the error return is taken; if it is non-null, the normal return is taken.

2.6 Space Suppression on Exit from FORMAT2 or TRANS2. The target designator for FORMAT2 and TRANS2 can now be:

-) T carriage/return, line/feed after last output
-> T+ suppress carriage/return, line/feed.

2.7 TRANS2 movement of multivalued fields. Except for the case when the target designator is W, TRANS2 now moves multi-valued attributes just as if they were ordinary fields; that is, the VM and SVM marks are moved along with the rest of the data characters.

2.8 Proc-to-proc Link. A new entry point, U91A2 has been added to permit a proc to link to a successor proc.

3.0 CHANGE IDENTIFICATION

Significant coding changes are identified in the maintenance record of each mode and are flagged in the assembly listings. The following table shows the modules affected by each change:

CALC2	FORMAT2	KERNEL2	POOL2	TRANS2	Change
---	(A)	---	---	---	Right justified output.
(B)	(B)	(B)	(B)	(B)	General Code Cleanup.
---	(C)	---	---	---	Formatted Output to the Line Printer.
---	---	---	---	(D)	Constant data to the PIB.
---	(E)	---	(E)	(E)	Swap blanks and backslashes.
---	---	---	(F)	(F)	Verify non-null amc.
---	(G)	---	(G)	---	Optionally suppress CR/LF.
---	---	---	(H)	(H)	Permit TRANS2 to move multi-valued flds
---	---	---	(J)	---	Clean up P-target code.
---	---	(K)	---	---	Fix bug in PCLIP.
---	---	(L)	---	---	Proc-to-proc link.
---	(M)	(M)	(M)	---	Final release clean up and de-bugging
---	---	---	(N)	(N)	Reduce size restriction in GETNXT

REALITY User Modes

1. Introduction. This report describes a set of assembly language modules which provides additional facilities for data access and manipulation, processing logic, arithmetic and logical computation, and report formatting on the Microdata REALITY system. The routines are installed as User Modes and are invoked by the U command in the REALITY Procedure language. Two modules, KERNEL2 and POOL2, include subroutines which are used by the other modules. Therefore, these modes are a prerequisite to installation of the other user modes. The remaining modules are independent of each other, although the functions they provide are complementary.

2. Summary. The following modules are available for immediate delivery:

2.1 KERNEL2 is a set of general purpose utility routines which includes n-way branching on output buffer values, suppression of printing of system-oriented messages, deletion of values from the primary or secondary output buffers, a facility for padding input values with leading zeros, a limited subroutine capability within a Procedure, and the ability to to from one prod to another. KERNEL2 is a required module since it contains subroutines which are used internally by the other user modes.

2.2 CALC2 is a computation module which operates on polish strings developed in an output buffer. Arithmetic, logical, and relational operations are provided and the resukt of a computation can be routed to the terminal, to the primary input buffer, or to either output buffer.

2.3 FORMAT2 is an output formatting module which permits data items to be displayed in a formatted report. Output values can be literals, attribute values, or attribute values which are converted as specified in their corresponding V/CONV dictionary value.

2.4 TRANS2 is a generalization of the T-conversion available in the basic REALITY system. Each of the values necessary for a file reference (file-name, item-id, attribute mark count) may be shown explicitly in the calling sequence, or may be located in the primary input buffer or either output buffer, with its location shown by a pointer in the calling sequence. The value retrieved from the file can be routed to the terminal, to a private file, or to either output buffer.

2.5 POOL2 consists exclusively of subroutines used internally by the other user modes.

3. Distribution. The INFACT User Modes will be distributed as source modules which can be assembled and loaded on your REALITY System. The normal frame assignments are given below. They can be modified prior to shipment, if desired.

Frame No.

Dec	Hex	Module
400	190	CALC2
402	192	FORMAT2
418	1A2	KERNEL2
419	1A3	POOL2
429	1AD	TRANS2

4. Modifications. Corrections and modifications will be distributed as necessary in the form of source modules.

5. Documentation. INFACT Report No. 74-2 presents general information, definitions, and conventions which apply to all modules. The modules themselves are described in the following INFACT reports:

Report No. 74-3	KERNEL2
Report No. 74-4	CALC2
Report No. 74-5	FORMAT2
Report No. 74-6	TRANS2

Concepts and Definitions

1. Concepts. The INFACT User Modes operate within the framework provided by REALITY Stored Procedures. The Stored Procedure concept is extended in that the data source for an operation can be the user data-base, the primary or secondary output buffers, the primary input buffer, or a special file which is uniquely associated with each port. The target for an operation can be either output buffer, the primary input buffer, the users terminal, or the private file. Figure 1 depicts the various possible input/output relationships.

2. Conventions.

2.1 Character Set Conventions. Two characters have special meanings to the INFACT User Modes; the percent sign (%) flags a reference to a value in the primary input buffer, and a sharp (#) flags a reference to a value in the current output buffer (primary, if ST OFF, or secondary, if ST ON). These characters should not be used as the initial character of a file-name, item-id, or attribute-name.

2.2 File-Name Convention. The private files used by the INFACT Modes have names of the form: nnP, where nn is the port number to which the terminal is currently attached.

2.3 General Naming Convention. A good general rule, which will avoid name clashes with the present modules, and with any future extensions, would be to avoid using non-alphanumeric characters as the initial letter of a file-name, item-id or attribute-name.

3. Definitions. The following definitions are used throughout the detailed descriptions of the INFACT User Modes.

3.1 Current Output Buffer. The primary output buffer if it has been selected by a ST OFF command; the secondary output buffer if ST ON was the last buffer selection command.

3.2 Alternate Output Buffer. The secondary output buffer if ST OFF was the most recent buffer selection command; the primary output buffer if ST ON last was issued. In other words, the alternate output buffer is the buffer which is currently inactive.

3.3 Indirect Reference. An argument in a TRANS2 or FORMAT2 calling sequence of the form:

%nn references a value in the primary input buffer
or #nn references a value in the current output buffer.

3.4 Private File. A pre-allocated file which is associated with the port to which the user terminal is attached. Private Files are working storage which is unique to a session; they are usually essential if multiple users are to execute the same stored procedure.

4. Target Designation. A consistent notation is used by all Modes to identify the target, or destination for a process.

A is the alternate output buffer.
P is the primary input buffer.
S is the current output buffer.
T is the user terminal.
W is a work file, the sessions private file.
V verifies item, but does not produce output.

4.1 Substitute Blank Option. Any of the target designators may be prefixed by the letter "B" to specify that embedded blanks should be translated to substitute blank (backslash) characters. This is an extremely useful facility when the target is P, S, or A.

4.2 Verify Attribute Option. VA is a valid target as well as V. VA verifies that the attribute is non-null.

4.3 Carriage Return/Line Feed Option. When the terminal is the specified target, carriage return/line feed can be suppressed by appending the character "+" to the target designator.

4.4 Summary of target designators. The following are all valid target designators.

A	BA		
P	BP		
S	BS		
T	BT	T+	BT+
W	BW		
V	BV	VA	BVA

KERNEL2 User Notes

1. Functions. KERNEL2 has nine user-oriented entry points which provide a variety of operations within a stored procedure. In addition, it contains two subroutines which are called internally from other INFACT Modes. The user entry points are:

Entry Point	Name	Function
0	NWAY	N-way branch on value in current output buffer.
1	NLDZ	Pads secondary input value with leading zeros.
2	SCLIP	Deletes the last item, or all items from the current output buffer.
3	CALL	Limited subroutine capability within a stored procedure.
4	RECALL	
5	RETURN	
6	HUSH	Alternates the system LISTFLAG. Can supress system oriented messages.
9	CHAIN	Proc to proc transfer of control
A	PCLIP	Deletes all values from the primary input buffer, from the current position to the end.

The following entry points are reserved for use by other INFACT modules. They cannot be called from a stored procedure.

7	PMAKE	Places port number in BMSBEG.
8	INDIR	Resolves indirect references.

2. Module Use and Calling Sequences.

2.1 Entry Point = 0. NWAY. Calling sequence is:

```
a-1     U01A2
a       index arg1 arg2 ..... argn
a+1     ...return here if buffer (index) = arg1
```

```
a+2       ...return here if buffer (index) = arg2
...       ...
a+n       ...return here if buffer (index) = argn
a+n+1     ...return here if no match with argument list
```

NWAY compares the specified value in the current output buffer with each argument in the calling sequence. If the value matches an argument, control is returned to the corresponding statement in the procedure. If no match is found, control returns to the statement following the nth statement following the call.

Example:

```
001  PQ
002  0..DEMONSTRATION OF N-WAY BRANCH
003  0..ENTER YES, NO, OR MAYBE
004  ST ON
005  H1
006  IN
007  A
008  H3
009  U01A2
010  2 YES  NO  MAYBE
011  G 100
112  G 200
013  G 300
014  X.YOU WERE SUPPOSED TO ENTER YES, NO, OR MAYBE
015  100 X.STACK(2) WAS "YES"
016  200 X.STACK(2) WAS "NO"
017  300 X.STACK(2) WAS "MAYBE"
```

The second value in the secondary output buffer is tested. If it is YES, control passes to line 011. If it is NO, control goes to line 012. If it is MAYBE, control is passed to line 013. A value other than YES, NO, or MAYBE would return control to line 014.

2.2 Entry Point = 1. NLDZ. Calling sequence is:

```
a         U11A2
a+1       ndigits
```

where ndigits is the total number of digits desired.

NLDZ pads the current value in the secondary input buffer with enough leading zeros to give a total of ndigits characters, and returns the resulting string to the secondary input buffer.

Example:

```
002  IN
003  U11A2
004  3
```

Input values of 1, 01, or 001 will all be replaced by 001.

2.2 Entry Point = 2. SCLIP. Calling sequence is:

```
a      U21A2
a+1    flag
```

If flag = 0, all values are deleted from the current output buffer. If flag = 1, the last value in the current output buffer is deleted.

2.3 Entry Points 3, 4, and 5. CALL, RECALL, and RETURN. These entry points provide a limited subroutine capability within REALITY stored procedures. CALL flags a statement for later transfer of control. RECALL returns control to the statement immediately following the latest call, while RETURN passes control to the first statement following the first GO statement following the latest CALL. The routines have no arguments so the calling sequences are simply:

```
U31A2    CALL
U41A2    RECALL
U51A2    RETURN
```

Example:

```
001  PQ
002  C.THIS PROC ILLUSTRATES THE LIMITED SUB-ROUTINE
003  C.CAPABILITY PROVIDED BY INFACT'S U31A2, U41A2,
004  C.AND U51A2.
005  G 200
006  100 IN
007  IF A = YES U51A2
008  IF A = NO U51A2
```

```
009  IF A U41A2
010  X..I GIVE UP!
011  200 U31A2
012  OENTER YES OR NO +
013  G 100
014  X..THANK YOU..
```

Statements 006 through 010 are the subroutine to be called. Statement 005 is a branch around the subroutine. (Note that it is desirable to place subroutines near the front of a PROC to reduce the overhead associated with transferring to them.) Statement 011 flags itself as the base for return of control. Statement 012 prints the message "ENTER YES OR NO", and then statement 013 transfers control to the subroutine. The subroutine accepts input, and if the value "YES" or "NO" is entered, transfers control to statement 014 via U51A2. If no response is entered (carriage return), the subroutine exits typing "I GIVE UP". For any other value, control is transferred via U41A2 (RECALL) to statement 012. The prompt is reissued, and the cycle repeats until one of the required responses is elicited.

2.4 Entry Point = 7. HUSH. This entry point alternates the print control switch between ON and OFF and thus can suppress system-oriented messages. Assuming the primary and secondary output buffers have been set up for a B/ADD, the following will suppress the system message "XYZ UPDATED".

```
010 U61A2    turns the print switch OFF
011 P        invokes the B/ADD process
012 U61A2    turns the print switch back ON
```

2.5 Entry Point = A. PCLIP. The calling sequence is simply UA1A2. In the following example, if three or more values are entered in response to the prompt on line 002, the calls to UA1A2 will clip the primary input buffer back first to two values and then to one value.

```
001  PQ
002  0ENTER MULTIPLE WORDS +
003  IP
004  DO
005  S3
006  UA1A2
007  S2
008  DO
009  UA1A2
010  DO
011  XDONE
```

2.6 Entry Point = 9. CHAIN. This entry point transfers control to the specified PROC without affecting the contents of the Primary Input Buffer. Both output buffers, however, are cleared. The following example shows how to use CHAIN.

```
001  PQ
002  ST OFF
003  HNEW.PROC
004  P91A2
```

Statements 2 and 3 set up the successor proc name in the primary output stack. Statement 4 then transfers control to "NEW.PROC". Note that statement 4 is a P91A2, not a U91A2.

CALC2 User Notes

1. Function. CALC2 performs arithmetic on a polish string which has been constructed in the current output buffer. The string is delimited on the left by an arbitrary character which is specified in the CALC2 calling sequence, and on the right by the character "?". CALC2 evaluates the string expression, deletes the entire string, including delimiters, and routes the calculated arithmetic value to the designated target.

2. Principles of Operation. CALC2 implements a stack arithmetic machine within the REALITY frame of reference. The character string passed to CALC2 consists of operands (numerical values) and operators (see below). Each operand implies a "LOAD STACK" command which places the value on the top of a push-down stack. Each operator processes one or two values from the top of the stack, and returns the result to the stack. Operators are either unary (the ¬ operator) or binary (all other operators). A unary operator replaces the top value in the stack with the result of operating on that value. A binary operator processes the top two stack values and then 1) deletes the top value from the stack, and 2) replaces the second value with the result of the operation. The string

   2 3 + 6 * ?

would result in the following operations:

String Character	Operation	Stack	Values
2	LOAD	S1	2
3	LOAD	S1	3
		S2	2
+	ADD	S1	3 + 2 = 5
6	LOAD	S1	6
		S2	5
*	MULTIPLY	S1	6 * 5 = 30
?	ASSIGN	S1	30 assign to target

3. Calling Sequence. CALC2 is invoked by the following calling sequence:

```
        a       U0190
        a+1     left-end-delimiter  target
```

where left-end-delimiter ::= character which defines the beginning of the string

and target ::= A ! P ! S ! T (See INFACT Report: 74-2)

4. Operators. The following operators are recognized by CALC2.

4.1 Arithmetic Operators. Integer arithmetic is provided for the following operators:

+	add
-	subtract
*	multiply
/	divide

4.2 Logical Operators. In the following operations, any value = 0 is FALSE, and any other value is TRUE:

&	logical and
!	logical or
¬	logical not

4.3 Relational Operators. In the following operations, the result is 1 if the relationship is TRUE, 0 otherwise:

<	less than
<=	less than or equal
=	equal
¬=	not equal
>=	greater than or equal
>	greater than
¬<	not less than
¬>	not greater than

5. Examples. The following PROC will evaluate the expression (A + B) (A - B), where the integers A and B are entered at line 003 separated by a blank. The result is printed at the terminal. At the point where CALC2 is entered (line 014), the secondary output buffer contains the string

```
: A B + A B - * ?
001   PQ
002   OENTER A AND B
003   IP
004   ST ON
005   H:
006   A1
007   H
008   A2
009   H +
010   A1
011   H
012   A2
013   H - * ?
014   U0190
015   : T
016   X
```

The following commands will place a zero in the second position in the primary input buffer:

```
a     S2
a+1   H: 0 ?
a+2   U0190
a+3   : P
```

FORMAT2 User Notes

1. Function. FORMAT2 provides general formatting capabilities for output from REALITY stored procedures. Output values can be literals, attribute values, or attribute values which are converted as specified by their corresponding V/CONV dictionary entry.

2. Calling Sequence. The entry point for FORMAT2 is GFORM and the calling sequence consists of multiple lines which specify data sources and format control for fields in the output image. The calling sequence is terminated by a special format item which consists of the characters "-> T" (note that there must be a blank between the > and the letter T). The calling sequence has the general form:

```
U0192
format-item
format-item
......
......
-> T
normal return
```

There are two kinds of format items; those which control horizontal or vertical spacing, and those which fetch (and optionally convert) values or which output constant information.

3. Spacing Format Items. There are three spacing format items: X, which controls horizontal spacing: S, which controls vertical spacing: and P, which skips to the top of the next page. FORMAT2 interfaces with the existing REALITY pagination code.

3.1 Horizontal Spacing. The format item Xnn places nn blanks in the current output line. If nn is omitted, no spaces are produced.

3.2 Vertical Spacing. The format item Snn spaces vertically nn lines after printing the current output line image. If nn is omitted, 1 line is skipped.

3.3 Eject to New Page. The format item P presents a top-of-form character and resets the REALITY page counter and line counter to 1. It also blanks out the heading which is pointed to by PAGHED.

4.0 Data Format Item. There are three data format items; V, which fetches and outputs a value: *, which fetches a data value, converts it according to the dictionary V/CONV field, and then places it in the output image: and H, which places the associated hollerith text in the output image.

4.1 Value. A value format item specifies the file-name/item-id/amc from which the data is to be fetched. It may also specify the column number in the current output line in which the value is to be displayed.

Vnn file-ref item-ref amc-ref

or V file-ref item-ref amc-ref

In the above, file-ref, item-ref, and amc-ref can be explicit references, or indirect references to values in the primary input buffer or in either output buffer. Output begins in column nn of the current output line. If nn is omitted, output begins in the "next" column (a current column pointer is maintained within FORMAT2).

4.2 Conversion. A conversion format item is similar to a value format item, with the following important exceptions:

1) The value is referenced by name instead of amc, and
2) conversion is performed, as specified by the dictionary V/CONV entry, before output. If no conversion is specified in the dictionary, this item is functionally identical to

a V format item. It has the possible advantage (for maintenance purposes) that the attribute is referenced by name, rather than amc.

*nn file-ref item-ref field-ref

or * file-ref item-ref field-ref

In the above, file-ref, item-ref, and field-ref can be explicit attribute names, or indirect references to names in the primary input buffer or either output buffer. Output begins in column nn of the current output line. If nn is omitted, output begins in the next column.

4.2.1 RIGHT JUSTIFICATION. If a dictionary has a V/CONV entry, the V/TYPE dictionary entry is also analyzed. If the entry is R, RA, or RN the converted data item is right-justified in a field whose width is specified by the V/MAX dictionary entry.

4.3 Hollerith. The H format item places hollerith text in the output image. The following three forms are resognized:

1. Hnn string
2. Hnn %m
3. Hnn #m

The first form places "string" in the output image. The second form places the value located in the m th position of the primary input buffer in the output image. The third form places the value located in the m th position of the current output buffer in the output image. In all cases, output begins in column nn of the output image unless nn is omitted, in which case, output begins in the next column.

5. Line Printer Selection. An L format item routes output to the line printer instead of to the terminal. The terminal is reselected for output on exit from a FORMAT2 call. The format item consists of the letter L, with no parameters.

6. Terminator. Since the calling sequence to FORMAT2 is of variable length, a special mark is required to identify the end of the calling sequence. The characters -> T have been selected (read "assign to terminal"). The T may optionally be followed by a plus sign to indicate suppression of the final carriage return/line feed sequence.

TRANS2 User Notes

1. Function. TRANS2 is a user mode which generalizes the REALITY translate facility and makes the result of a translation available for additional processing. Input to TRANS2 is a triple (file-name item-id amc) which specifies the data value to be fatched. Each component of the triple can be a direct reference (that is, it appears explicitly in the calling sequence), or an indirect reference (that is, it points to a value in the primary input buffer or the current output buffer). The calling sequence also specifies the destination, or target, for the translated value. The target can be the user terminal, the current output buffer, the primary input buffer, or a record in a private file associated with the port to which the user terminal is currently attached. Secondary entry points provide for the definition of the private file, and the subsequent fetching of values from it.

2. Calling Sequences. The primary entry point is GETAT. The calling sequence is:

```
a       UO1AD
a+1     file-ref item-ref amc-ref target
a+2     error-return
a+3     normal-return
```

where:

```
file-ref ::= file-name  ! indirect-ref
item-ref ::= item-id    ! indirect-ref
amc-ref  ::= amc        ! indirect-ref

indirect-ref ::= #integer ! %integer
```

(Note: #integer references a value in the current output buffer; %integer references a value in the primary input buffer)

```
target ::= T = user terminal
           S = current output buffer
           A = alternate output buffer
           P = primary input buffer
           V = verify existence only (no output)
          VA = verify non-null amc
           Witem-id = user work file
```

All target designators can be prefixed by the letter B to designate replacement of embedded blanks by back-slashes.

TRAN52 has four additional entry points:

GETNXT: U11AD
Witem-id target
end-of-file return
normal return

SEENXT: U21AD
Witem-id target
end-of-file return
normal return

PRIVFILE: U31AD
target
normal return

HOLLPIB: U41AD
string
normal return

GETNXT get the next item from the record identified by (priv-file item-id) and returns it to the target. Next in this sense, means the next value in a multi-valued field. If no values remain, control passes to the PROC statement immediately following the argument list. Otherwise, control passes to the second statement following the argument list. SEENXT operates in a similar way, with the important difference that GETNXT advances the counter which determines the "next" field, while SEENXT does not.

PRIVFILE determines the port number to which the user terminal is currently connected, generates a file-name of the form nnP (where nn is the port number), and returns this file-name to the specified target.

HOLLPIB replaces the current item in the Primary Input Buffer by the character string which follows the call. Note that, if "string" has multiple blank-separated words, the effect is to right shift the following items in PIB. The current PIB item can be specified by use of S statements preceeding a call to HOLLPIB.

3. Error returns. Two classes of errors are recognized by TRAN52; those which terminate execution of the proc, and those which do not.

Error/Message	Action
invalid file-name	error return
invalid item-id	error return
72 VALUE "target" IS MEANINGLESS	terminate proc
201 "file-ref" IS NOT A FILENAME	terminate proc
13 DL/ID IS MISSING	terminate proc

4. Using TRANS2. The following examples assume the files and data shown on page 5. The examples are intended to highlight some of the features of TRANS2 and do not necessarily represent a practical use of the REALITY System.

4.1 Dynamic Allocation of a Private File.

```
001  PQ
002  ST OFF
003  HCREATE-FILE (
004  U31AD
005  S
006  H 1,1 1,1)
007  PP
008  X
```

Assuming that the port number to which the terminal is currently connected is 7, the primary output buffer will contain the following text after the execution of line 006:

```
CREATE-FILE (07P 1,1 1,1)
```

4.2 Using GETAT, SEENXT, and GETNXT.

```
001  PQ
002  1 OENTER CUSTOMER NO.+
003  IP                    accept input to prim. ip buf.
004  U01AD                 call TRANS2 to get the open
005  CUST %1 3 WWORK       invoice list..to private file
006  G 200                 no such customer
007  U01AD                 call TRANS2 again
008  CUST %1 1 T           print cust. name on terminal
009  XERROR.9              should never happen
010  100 U21AD             call SEENXT to print
011  WWORK T               the invoice number
012  G 1                   no more invoices
```

```
013  U11AD                call GETNXT; move invoice no.
014  WWORK S              to current output buffer
015  XERROR.15            should never happen
016  U01AD                call GETAT to print invoice
017  INV #1 2 T           amount at the terminal
018  XERROR.18            invoice record missing
019  U21A2                clear current output buffer
020  0                    (see Report 74-3)
021  G 100                get next invoice
022  200 OCUSTOMER NOT FOUND
023  G 1                  try again
```

5. Private File Items Created by TRANS2. As explained earlier, GETAT can create an item in the user's private file for later use by GETNXT and SEENXT. The item has two attributes. The first is a four digit counter which specifies the next value to be returned. The second is the multi-valued field referred to in the call to TRANS2.

6. Sample Files and Data. The example in section 4.2 of this report assumes the following files and data:

DICT CUST:	CUST...	A/AMC	V/CONV	V/TYP	V/MAX	V/MIN
	ACCT-NO	0		LN	5	5
	NAME	1		L	20	5
	ADDR	2		L	25	10
	INVCS	3		LN	3	3
DICT INV:	INV.NO	0		LN	3	3
	DATE	1	D	L	11	11
	AMT	2	MD2	RN	7	3

```
:COPY CUST * (T)

    12345
001 SAM JONES
002 ANYWHERE,USA
003 102 110 127

    54321
001 CHARLEY SMITH
002 NOWHERE, USA
003 013 103

:COPY INV * (T)

    013
001 2199
002 1234

    102
001 2200
002 2345

    103
001 2210
002 4321

    110
001 2220
002 5432

    127
001 2249
002 3333
```

REV	EO	DESCRIPTION	DWN	CHKD	APPD	DATE

PREPARED DATE	TITLE				
CHECKER	REALITY MULTIPLE DISC DIAGNOSTIC External Specification Consists of program part numbers 11392-01, 11396-01, 11390-01		TM Microdata IRVINE, CALIFORNIA		
ENGINEER					
APPD					
APPD			A		
APPD					
RECORDS	IDENT CODE		DWG SIZE	SHEET 1 OF 5	REV

Form 3-132-1

3.5 ENTER DISC DEVICE ADDRESS will appear next on the terminal. Two characters will be accepted. If they are not 14, 15, 16 or 17, the message will be repeated.

3.6 UNIT NUMBER (0/1) will be printed next. Any response other than 0 or 1 will cause the message to be repeated.

3.7 IS DENSITY 100 TPI? (Y/N) is printed next. A response of Y causes the program to assume 203 cylinders and a response of N 406 cylinders. Any other response causes the message to be repeated.

3.8 DO YOU WANT TO FORMAT ONLY? is next. A response of Y causes the single operation of 'format' to be performed on the entire disc drive defined by device address and unit number. This takes about 3 minutes on a 100 TPI drive and about 6 minutes on a 200 TPI drive. This mode of operation should not be used on a virgin pack since it will not rectify any data check code errors. A response of N will cause the next message to be printed. Any other response will cause the message to be repeated.

3.9 DO YOU WANT DATA COMPARE? is printed if the response to the previous message was N. A response of N causes four operations to be performed on the entire disc drive: format, read, write and verify. This mode of operation takes about 12 minutes on a 100 TPI drive and about 24 minutes on a 200 TPI drive. This mode of operation is recommended for virgin packs. A response of Y causes an additional read into a second buffer and a byte for byte compare of both buffers. This mode of operation takes about 30 minutes on a 100 TPI drive and about an hour on a 200 TPI drive. This mode of operation is recommended for drives or controllers with suspected problems.

4.0 Control

4.1 Sense switch 2 is used to control looping on errors. If sense switch 2 is up, the diagnostic will loop indefinitely on an operation which produces an error status. If the switch is down no operation is repeated. ; The diagnostic must be run with sense switch 2 down on a virgin pack to get past the data check code errors on the first read. Note that the format only mode never loops on errors. To restart the diagnostic before it is complete, depress INT to halt the CPU then RESET, INT with sense switches 1 and 4 up to restart the diagnostic.

1.0 Introduction

This is a stand alone Reality program which is read in from a coldstart tape. The diagnostic consists of two modes, DISC-DIAG which contains all of the code and DISC-MSG which contains all of the disc and communication controller control strings plus all messages which are output to the operator.

This diagnostic provides the ability to format and establish good data check codes on virgin packs; to reformat disc packs which have already been used but have become unformatted without destroying any data; to verify that the read, write, and verify operations work correctly; and to verify that the data is transferred correctly. It also checks the interrupt capability of the disc controller and verifies that the controller interrupts on the proper device address. The diagnostic can be used with device addresses X'14' through X'17' on either unit 0 or 1.

2.0 Creating the Tape

2.1 Mount a scratch tape with write ring and place the tape drive on line.

2.2 Log on to SYSPROG

2.3 FILE-SAVE
(ABS DUMP SPECS?)C
(DISK INIT SPECS?)X

3.0 Using the Tape

3.1 Mount the tape and place the tape drive on line

3.2 If this is a running system make sure everyone with additional work space has logged off and core has been flushed.

3.3 Depress INT,RESET,INT with sense switches 1,2, and 4 down.

3.4 DO YOU WANT TO RUN THE DISC DIAGNOSTIC (Y/N) will appear on the terminal on line zero of device X'18'. A response of Y will cause the next message. A response of N gives control to the system configurator.