DR 1196

LA-6789-MS
Informal Report

UC-32

Issued: May 1977

RSX-11M Versus RSX-11D: Compatibility and Conflict

Jeffrey M. Gallup*
Edward G. Lieberman*
Sally Shlaer

*Lawrence Berkeley Laboratory, University of California, Berkeley, CA 94720

los alamos
scientific laboratory
of the University of California
LOS ALAMOS, NEW MEXICO 87545

An Affirmative Action/Equal Opportunity Employer

UNITED STATES
ENERGY RESEARCH AND DEVELOPMENT ADMINISTRATION
CONTRACT W-7405-ENG. 36

This work supported by the US Energy Research and Development Administration, Division of Physical Research.

Printed in the United States of America. Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161
Price: Printed Copy $3.50 Microfiche $2.25

CONTENTS

RSX-11M VERSUS RSX-11D

Compatibility and Conflict

by

Jeffrey M. Gallup, Edward G. Lieberman,
and Sally Shlaer

ABSTRACT

It is our observation that, as RSX-11M and RSX-11D continue to diverge, more and more users are finding it necessary to run both systems for similar ranges of applications. This situation can impose an unacceptable cost in software development and maintenance. Compatibility in utilities, system files, system features, and code at source and object level offers a realistic first step to reduce this burden.

This report presents a discussion of the incompatibilities uncovered in converting a large volume of software from one operating system to the other, as well as the techniques developed to circumvent these problems. Code conversion and long-term software maintenance are discussed with respect to specific system incompatibilities. Finally, management and software engineering tools are discussed as they apply to the conversion and maintenance effort.

I. HISTORY AND MOTIVATION FOR THIS STUDY

Both the Lawrence Berkeley Laboratory (LBL) and the Los Alamos Scientific Laboratory (LASL) support a number of experimental research stations utilizing PDP-11 computers to perform data acquisition with CAMAC instrumentation. These installations span a broad range of complexity, from small single-purpose, single-processor systems to large and sophisticated multi-purpose, multi-processor systems. During the last several years, there has been a persistent trend towards the latter type of configuration. It is our observation that this trend has been accompanied by an increasing duplication of effort in the preparation of application codes to run on these large systems. Moreover, the fiscal burden imposed by the need for individually tailored, high-performance software is severe and becomes increasingly difficult to justify when the application systems are or can be made quite similar.

Due to these considerations, substantial interest developed at LBL in a general-purpose data-acquisition system and in "intelligent" front-end hardware capable of handling increasing data rates. Accordingly, LBL proceeded with in-house conceptual design and specification while concurrently surveying other national laboratories. Most of LBL's requirements were met by a system known as 'Q'[1] which was in operation at LASL. This system was chosen as a starting point, and with the aim of achieving an even more sophisticated system for use at both laboratories, an inter-laboratory collaboration was subsequently begun.

The Q system comprises approximately two million bytes of source, command, and documentation files. When the collaboration began, Q represented an investment of seven man-years in software and four man-years in hardware. Furthermore, LASL had substantial field experience with the Q system. The need for the collaborative effort arose to a large extent from the fact that the two laboratories are oriented towards different operating systems: Digital Equipment Corporation's RSX-11D,[2] the system for which Q was designed and implemented, predominates at LASL; Digital's RSX-11M[3] predominates at LBL.

The collaboration began with a systematic evaluation of the conversion effort required to obtain a Q system running under 11M while retaining a functionally equivalent Q system running under 11D. (In designing Q, LASL had, in fact, given consideration to having it run eventually on small 11M systems; however, despite substantial expertise in 11D, the familiarity of LASL personnel with 11M was primarily at the conceptual level.) No substantial difficulty was anticipated in converting the 47 nonprivileged tasks and three support libraries, which were generally uncomplicated and contained many modules coded in FORTRAN. Of primary concern were the eight privileged tasks which were categorized as follows: Two privileged only to map over the I/O page, two others privileged but not device handlers (one straightforward, one complex), four device handlers. The device handlers were assessed as follows: One trivial, one reasonably simple, two (which share a substantial amount of common code) extremely complex. Because each laboratory had considerable expertise in at least one of the two systems (11D or 11M) in the areas of system data structures and internal operations, we were able to make a reasonable estimate of the difficulty of converting the privileged tasks. Since LASL had a strong interest in providing Q under 11M, the decision was then made by the two laboratories to commit their joint resources to the conversion effort. One should note that this was to be a "wrong way" conversion in that DEC had originally advertised 11M as upward-compatible to 11D at the object level for nonprivileged tasks.

II. SOFTWARE ENGINEERING CONSIDERATIONS

We define "software engineering" as consisting of the concepts, procedures, and utilities that provide for orderly, reliable, and reproducible development, distribution, and maintenance of software systems. These concepts extend from specification documents through coding standards all the way down to decisions on such small points as generation of trace-back code, schemes for filing listings, and generation of long or short load maps. The conversion project raised a number of procedural and organizational issues which have been resolved through engineering "ground rules." This framework allowed the elimination of numerous small, tedious ad hoc decisions. Our primary goal was to complete the conversion rapidly and to achieve a pair of highly supportable systems over a long project lifetime--say five years. The ground rules are aimed toward that end and are motivated by both technical and organizational considerations.

The alternative of converting all present and potential Q users to a single one of the RSX-11 systems was not open to us: The various machines that presently run Q at LASL also employ a great deal of other application code concurrent with Q. This application code is specific to each machine, is not centrally maintained, and uses features unique to RSX-11D. The same situation prevails at LBL, except that there the locally generated code uses RSX-11M specific features.

A. Single System Rule

> RULE 1: There will be a single Q system. Variants for RSX-11M and D are allowed only when no feasible alternative can be found. Internal (system-dependent) variations must be transparent to users of the system.

It was our decision that reprogramming, as a one-time investment, was acceptable to meet Rule 1. Some reprogramming has, in fact, been necessary to achieve compatibility in nonprivileged tasks. Privileged tasks have required extensive and, in some cases, extremely difficult redesign and reprogramming. Although we consider conditional assembly to be a reasonable technique for handling and minimizing variations, it is applicable in only a few situations.

The engineering basis for the "single system" rule is simply that by limiting the number of system-dependent modules and, hence, the total inventory of code, a much higher quality of code will be achieved and maintained. A single system can be expected to exhibit a more coherent character than two which are maintained in parallel incarnations. Moreover, only a single set of documentation must be maintained. Finally, the chance of divergence due to human oversight is minimized.

The "single system" rule leads to the observation that a phased conversion is both possible and highly desirable. The basic steps in the phased conversion are:

1. Convert all nonprivileged code so that it can run under either operating system. This code can now be validated by use under the original system.

2. Convert system-dependent modules and test under the target system.

Since Q must run under both systems, our effort stopped at Step 2. For applications in which an irreversible conversion is possible, Step 3 is available:

3. Relax restrictions on nonprivileged code to allow use of the full power of the target operating system.

B. File Compatibility

A logical outgrowth of Rule 1 is

RULE 2: Files must be compatible at all possible levels in order to minimize maintenance and storage requirements. Listing files are not exempt from this requirement.

There are various classes of compatibility which deserve separate discussion. In general, file compatibility at some level is defined operationally: A compatible file will function in exactly the same way under each operating system. For any given file, compatibility can only be established by incorporation in both systems.

1. MACRO Source Files

Example 1: This file is entirely incompatible:

```
.MCALL FIX$C
FIX$C  TSKNAM
```

The FIX directive is not supported by 11M and the FIX$C MACRO is not defined in RSXMAC.SML for 11M. The listing and object files produced by MACRO[4] under the two systems differ. This file is incompatible at source, object, and listing level. Furthermore, it will produce run-time incompatible tasks.

Example 2: Compatibility dependent on host system:

```
.MCALL  QIO$C
QIO$C   IO.ATT,1
```

RSXMAC.SML for 11M causes a 12-word DPB to be generated, whereas 11D generates a 6-word DPB. The file is source-compatible, but, if assembled under 11D, not run-time compatible. If assembled under 11M, it is object, listing, and run-time compatible.

Example 3:

```
START:  MOV   R0,R1
        .END  START
```

This file is compatible at source, object, list, and run levels.

Files such as that shown in Example 3 are clearly the most desirable and, in fact, can often be achieved by severely restricting the system capabilities they exploit. Since the MACRO assembler on both systems employs the last-encountered definition of a macro, we developed a file which, when assembled as a prefix file to any MACRO-coded module, guarantees the compatibility at source,

object, listing, and run level of any error-free assembly. Moreover, the prefix file MD.MAC flags areas of contention.

It should be noted that a MACRO file which is compatible at source, object, and list level may, in spite of all these precautions, cause faulty task-building under certain circumstances. Even a successful task-build does not guarantee identical behavior in execution. This will be discussed in more detail in Sections III, IV, and V.

2. FORTRAN Source Files. FORTRAN is defined to be a system-independent language. In the spirit of Rule 1, the goal is to have 11M and 11D systems produce identical .OBJ and .LST files; in this manner, incompatibilities are eliminated at the FORTRAN object level. The expeditious method to obtain this goal is to use the same compiler on both systems which is, in fact, done. Run-time incompatibilities exist when FORTRAN routines invoke system-dependent support and interface routines from the system library. This incompatibility is resolved only by careful manual screening of FORTRAN source code or resultant load maps. SYSLIB differences are discussed in Section IV.

3. Object Libraries. Determination and enforcement of compatibility for DEC-supplied system libraries is dependent on the availability of machine-readable source code. Access to this material is costly and release-dependent. Further complications are discussed in Section D.

Non-DEC object libraries present a different situation:

RULE 3: Libaries for use by nonprivileged tasks must be made object and run-time compatible.

This was achieved by careful review of all library material and is enforced by the prefix file for MACRO-coded modules. We note again that only careful and knowledgeable coding and review will produce run-time compatible code from FORTRAN. This is an especially sensitive point for user libraries since thorough testing of a library module on both systems is often prohibited by time constraints.

C. Automation

RULE 4: The production of the system must be entirely automated through batch and command files at all levels.

Command files guarantee reproducible results and minimize operator error. The assembly or compilation of each module requires a precise command string specification, as does each task-build. Accordingly, about 300 command and batch files are used to build Q. The uniform and reproducible facility for production of Q is a necessary software engineering tool for maintaining project integrity. Rule 4, while seemingly innocuous, raises a number of compatibility issues:

1. Task-images are never compatible since the header is system-dependent, as are SYSLIB and probably SYSRES. We chose to distinguish between the pairs of task images (with their associated load maps) on the basis of a UIC convention to be described in Section VI. This implies that there must be a pair of indirect command files for each task to be built which, for nonprivileged tasks, differ primarily in UIC assignments of output files.

2. Command files to task-build SGA's differ between the two systems in partition-assignment of the SGA.

3. Batch and indirect MCR command files, while essentially similar, are not interchangeable. Again, these files occur in pairs: a Batch file for 11D and an indirect MCR file for 11M.

4. Where conditional assembly is used, the command file for producing the assembly must select some appropriate system-dependent prefix file other than MD.MAC to produce code for the selected target system.

To aid in maintenance of the command files, a scheme was developed which allows the production of the command files to be automated. A utility program named OST was written for just this purpose; its functional character will be described in Section VI.

D. System Modifications

RULE 5: DEC-supplied components must be used without modification.

There were three reasons for declaring this rule:

1. The Q system is distributed to other users with whom we may have no close contact. The only assumption that can be made is that the underlying system is at least as rich as that distributed by DEC. This rule forbids our alteration of RSXMAC.SML and SYSLIB and so prevents most naming conflicts that could arise if the user had elected to modify either of these files.

2. Q should be relatively insensitive to release changes; that is, conversion from release to release must involve only our own code. This is especially important for RSX test sites and, in fact, both laboratories involved had at least one system acting as test sites for RSX-11M V3 and RSX-11D V6.2 during the conversion project.

3. Finally, we wanted to be able to isolate bugs quickly to DEC-supplied components or to our own work and thereby avoid any awkward finger-pointing sessions.

However, even as we made Rule 5, we realized that it would have to be broken: Modifications to the 11M executive would be required in order to cope with the directive weaknesses to be described below.

E. Program Development and Maintenance

RULE 6: Errors and incompatibilities must be detected as early as possible in the development cycle.

Our weapons for attack here are: (1) the prefix file for enforcing compatibility of MACRO-coded files, (2) parallel command files: An error detected in one command file generally implies an error in the partner file, and (3) thorough understanding of at least one of the operating systems and most of the application system on the part of every member of the team. This knowledge, a full set

of telephone numbers and sufficient travel support from both laboratories have allowed us to communicate every error and incompatibility on an early-warning basis.

III. SYSTEM DIRECTIVE DIFFERENCES

Of the 52 system directives supported by RSX-11D V6A, we find 41 of these incompatible with RSX-11M V2. The following is a summary of these discrepancies together with our evaluation of their impact on the total system. The strategy employed for dealing with these differences revolves around the previously mentioned prefix file MD.MAC, which is assembled with every MACRO-coded component of Q. This prefix file guarantees object compatibility at the price of decreased capability or increased module length. We found it necessary to modify the 11M executive in order to strengthen a few critical directives. These modifications are noted in this section. We note that the prefix file makes it possible to support the modified 11M capabilities.

The RSX-11M directives are an austere subset of those available in RSX-11D. Repeatedly, a deficiency in an RSX-11M directive forces one to consider alternative directive sequences and design strategies; these are often ruled out by other directive deficiencies and system architectural differences.

A. Directives Available in Only $S Form

The directives ASTX, DSAR, DSCP, ENAR, ENCP, EXIT, GSSW, SPND, and WSIG are available only in the $S form in 11M. As the manuals indicate, for these directives the $S form is always shorter and executes at least as fast as the $C form. We consider the lack of a $C form to be a nuisance at worst and have arranged the prefix file so that $C forms are acceptable (although $S directives are actually generated). We note that although some $C forms will work in 11M, there is no reason to suppose that will be the case in future releases; hence, they are excluded. The prefix file flags the $ form of the directive as illegal. The lack of the $ form in 11M prohibits the writing of table-driven code involving these particular directives.

B. CSRQ: Cancel Scheduled Requests

In 11M, the task issuing the "cancel scheduled requests" directive cannot specify the requestor task name; all scheduled requests for the target task are cancelled. The prefix file accepts the 11M form only, flagging the 11D extension as illegal. This is an acceptable solution for our system since the code is not sensitive to the strong form of the directive at this time. However, we have seen situations where the directive would be essential; for instance, a task which runs periodically to perform a centralized function for a series of other tasks (take data A once a second for task A, take data B once a minute for task B,...) could be controlled effectively only through this missing function. This deficiency is judged to be serious in a real-time system.

C. RQST, RUN, SCHD, SYNC, and EXEC

If the partition and/or priority for the target task are specified in a RUN or RQST MACRO, they are processed by 11D but ignored by 11M. This denies the 11M system the possibility of configuring a memory load to fit a particular partitioning scheme on the target system; under 11D, one can attempt to run a

task in partition A if it exists, falling back to partition B if the first request is not successful. This scheme allows one to write nonprivileged code which adjusts itself to the configuration of the target system. The inability to assign priority at run time makes it virtually impossible to centralize control of multiple functions in a single real-time task which executes operations according to their relative importance in the overall application scheme. The prefix file forces the 11M form of the RQST and RUN directives, flagging the 11D extensions as illegal. The missing options are considered to be serious deficiencies in a real-time environment; the design impact is far-reaching in the application code. System-imposed impairment of one's ability to centralize control and distribute functionality is unfortunate.

SCHD and SYNC differ from RUN only in the presentation of scheduling data. Though some feature and utilization differences exist, they are considered relatively minor.

EXEC (Execute task contingent on memory availability) is not supported in 11M. In 11D, EXEC is used in preference to RUN when tight control of partition utilization is required. This technique is often used to avoid "deadly embrace" situations. The absence of this capability in 11M necessitated redesign of some intertask relationships. Without the EXEC directive, system stalls must be discovered by other means; and unfortunately, many appealing schemes (SGA communication, Send and Request, etc.) are ruled out by other system incompatibilities. A console message is appropriate when an application-critical task cannot be executed provided that the presence of an operator can be guaranteed and that an appropriate recovery sequence can be initiated from the terminal. Time-critical operations generally cannot be handled in this manner.

D. GPRT: Get Partition Parameters

The partition flags word returned to the issuing task differs in form between the two systems:

11M	bit 0=0	System-controlled partition
	bit 0=1	User-controlled partition
11D	bit 0=1	User-controlled partition
	bit 1=1	Occupied-user-controlled partition
	bit 2=1	System-controlled partition
	bit 3=1	Active-system-controlled partition

The meaning of bits 2 and 3 in the flags word in 11D is in question; that given above is quoted from the directive description in the manual.[2] It is far more likely that these bits are copied from the TPD (and that also is implied by the directive description), in which case bit 2, if set, indicates that the partition is time-shared and bit 3 indicates whether or not the time-shared partition is active.

When the partition name is omitted, 11D returns the size of the partition of the calling task, while 11M reports the size of the sub- or task partition. Furthermore, the partition base address is returned by 11D while 11M returns the task partition base address. Thus, the size and base address returned for a system-controlled partition differ for 11M and 11D. The implementation incompatibility within this directive reduces its utility to zero. Perhaps differing directive names and identification codes should have been used.

E. GTSK: Get Task Parameters

The buffer contents returned by the directive differ as follows:

word	11M	11D
4,5	Undefined	Name of task to which ATL is accounted (requestor)
11	Undefined	CPU type
12	Undefined	STD flags
15	Size of task address space	Size of read-write address space

There appears to be no method for a nonprivileged task to determine the CPU type under 11M. Lack of this capability forces increased administrative work for a compiler within the Q system, as the compiler generates code appropriate to the CPU model.

The lack of the requestor information in 11M sharply curtails design options in multi-task subsystems. This deficiency forces one to use the SDAT directive in order to allow the requestor to identify himself; this quickly leads to the difficulties described in the receive directives where actual data transmission is also involved.

F. CMKT: Cancel Mark Time

Under 11M, all mark-time requests are cancelled; under 11D, one has the option of cancelling selectively according to a match on event flag number or AST address. This capability is essential to control execution of a task which is executing two asynchronous processes. We extended 11M to support selective cancel mark time on event flag or AST address match, and this is reflected in the prefix file.

G. DECL: Declare Significant Event

DECL is essentially a compound directive: Test event flag, set flag, declare significant event, and report flag polarity prior to issuance of the directive. 11M uses the $S form with no event flag. 11D takes all of the three standard directive forms with an optional event flag. The prefix file generates the $S form for either $S or $C if no event flag is specified and simulates the 11D capability with 11M-compatible directives if an event flag is specified. The $ form is declared illegal.

H. WTLO: Wait for Logical OR of Event Flags

Under 11D, this directive can be used to wait for any combination of local and global event flags in any or all groups, while in 11M the directive allows one to wait for a combination of flags in only a single group. The prefix file simulates the 11D capability by an awkward combination of 11M compatible forms. The overhead in 11M for simulating the 11D capability is so high (WSIG is required) that simulation can be used only in selected cases. Redesign of the application code is an expensive but preferred method for realistically dealing with the deficiency in this directive.

I. DSAR/IHAR: Disable/Inhibit AST Recognition

These two directives are actually the same in both systems but bear different names. The 11D version of RSXMAC.SML cross-defines the macros. In 11M, only the name DSAR is used and only the $S form is supported. Since 11M does not cross-define the MACROs, both definitions are supplied by the prefix file.

J. ALUN: Assign Logical Unit Number

Two problems exist with this system directive. In 11D, ALUN can deassign a LUN, whereas no similar facility exists in 11M. The prefix file detects this case in the use of the directive and flags it as illegal. The second problem is one of system architecture. In 11M, one may use operator-induced logical device assignments. The logical device assignments made at a terminal apply to all tasks whose TI assignment points to that terminal. Since ALUN searches the local device assignment tables before the system device tables, its effect under 11M can be somewhat unpredictable, especially for devices such as SY. There is a similar but lesser effect under 11D, since only the less-used system-wide REDIRECT controls device assignments. The proper use of ALUN, particularly for disks, must therefore be determined on a case-by-case basis.

K. GLUN: Get Logical Unit Information

The first device characteristics word returned by the system in response to this directive has minor differences:

11M	bits 6-12	Reserved
11D	bits 6-8	Not defined
	bit 9	Software write-locked
	bit 10	Input spooled
	bit 11	Output spooled
	bit 12	Pseudo-device

The flags byte has similar differences:

11M	200	
11D	sum of:	200 Handler resident
		100 Load and record
		40 Device off-line

L. QIO and QIOW: Queue I/O and Queue I/O and Wait

11M does not presently support QIOW, and this facility is expected in Version 3. In the meantime, the prefix file simulates QIOW by a combination of 11M compatible forms.

An ordinary QIO in 11M ignores the I/O priority parameter if specified and always generates a six-word parameter list. The prefix file flags an error if priority is specified in the MACRO-coded directive and always generates a full six-word parameter list. This could cause difficulties in the conversion of code since an 11D driver has access to DPB length and, by deduction, to the length of the parameter list. This information can be used to determine the precise nature of the requested function. DEC-supplied device drivers in 11D make use of the fact that a request issued by the system is flagged with a DPB size of zero.

Under 11D, if no I/O priority is specified, the task priority is used as the I/O priority which dictates the position the I/O request will take in the handler queue. Under 11M, an I/O request is always queued according to task priority. Again, the incompatibility arises from system architectural differences. 11D applications which depend on the I/O priority effect can, therefore, be difficult to convert to 11M, even if comprehensive redesign is done.

M. RCVD and RCVX: Receive Data and Receive or Exit

Under 11D, these directives can be used to do a selective receive of data sent by a specified task; under 11M, the directive will cause the next data-packet queued to the receiver to be received. We found it necessary to modify the 11M executive to support selective receive in order to allow a dialogue between two tasks to complete before another is initiated. The prefix file is used to support the stronger form of the directive MACRO.

A few other incompatibilities in the receive directives should be noted: In 11D, the V-bit is set on directive completion if the sender is privileged, whereas, under 11M, a receiver cannot determine the sender's privilege state. We note that the V-bit feature is seldom used in 11D. Furthermore, if an 11D receiving task is multi-user, its TI assignment must match that of the sender for the directive to return data. This capability is lacking in 11M; consequently, the use of send/receive as a communication technique for multi-user task clusters is effectively precluded. Support for these facilities must be provided at the executive level if the capabilities are required.

N. RCVS and SDRQ

Receive or suspend, RCVS, is not supported by 11M and cannot be simulated if the receiving task is of lower priority than the sender. The combination RCVD / SPND is effective if the sender has LOWER priority.

The partner directive in a dialogue, SDRQ, Send and Request or Resume, has a similar problem: The directive is not supported by 11M and accurate emulation depends on the sender being of HIGHER priority. Catch-22. Both directives are declared illegal in all of their various forms by the prefix file.

For these directive pairs, event flag coordination may be used to the extent that application redesign is possible. In the absence of an "inhibit task switching" directive, there is no alternative but to consider each use of these directives on a case-by-case basis.

Alter priority (ALTP) would be considered if it were available in 11M. This lack of options drastically restricts a program structure in which two tasks engage in a send-resume/receive-suspend dialogue.

O. Variable Length Send and Receive

The following directives exist only in 11D:

VSDA	Variable length send data
VSDR	Variable length send and request or resume
VRCD	Variable length receive data
VRCS	Variable length receive or suspend
VRCX	Variable length receive or exit

Long strings of data must therefore be sent in a series of 13-word packets with the nonstandard selective receive used by the target task as described in section M above. A user-supplied protocol must be followed in order to detect end-of-data reliably.

The prefix file declares all variable-length send and receive directives to be illegal; simulation is possible only in highly controlled code taking into account the relative priorities of the tasks and the structure of the data. We consider the absence of this capability in 11M to be extremely serious, particularly since both operating systems offer poor support for inter-task communication.

P. Further Directives Missing in 11M

The directives ALTP (alter priority), FIX (fix task in memory), UFIX (unfix), DSBL (disable execution of task), ENBL (enable execution of task), and GCOM (get SGA parameters) do not exist in 11M in Version 2. Where these directives are employed in 11D applications, they must be designed out of the application architecture. Alternately, some of the operations these directives imply may be invoked at the console level. Increasing operator burden requires substantial justification (tasks must ask the operator for a FIX) and is, of course, inappropriate for an unattended application.

IV. TASK-BUILDER AND SYSLIB DISCREPANCIES

We compared full entry-point listings produced by the librarian for SYSLIB for RSX-11D V6A and RSX-11M V2 at base level 12. There are approximately 160 modules in each of these libraries, not including the FORTRAN run-time system. Of these 160 modules, we found discrepancies (other than length or IDENT) in 57 modules. The inconsistencies, in addition to task header differences, make it mandatory to use the proper task builder with its matching SYSLIB to get useable output. Many of these appear to be minor, but a proper evaluation cannot be made without study of the source listings. Those we can comment upon are as follows:

1. As one would expect, in the RSX-11M SYSLIB there are no entry points to interface to capabilities that do not exist in the system. For example, CANOBY is the RSX-11D entry point to interface to selective cancel scheduled requests; this entry point does not exist in 11M. One would also expect that library modules that interface FORTRAN code to the weaker 11M directives would ignore arguments which are not meaningful in 11M, particularly when these arguments are optional in the 11D counterpart. We have not been able to confirm this notion, but find support for it in length differences we see in such library modules. This is the real source of the difficulty in establishing compatibility for FORTRAN-coded tasks: Proper operation can be confirmed only at run-time.

2. The modules which define the FCS offset symbols are different. There appear to be two different names for some offsets; not all are cross-referenced. This may arise, in part, from the fact that the release dates on the compared libraries differ by more than a year. Some differences are clearly traceable to spooling as 11M and 11D FDB's differ at this point. The significance of other differences is unknown.

3. 11M has a module in SYSLIB that defines the directive DPB offsets (A.LULU, Q.IOPL, etc.); 11D does not. This is a problem when transporting code

from M to D but not from D to M; these symbols can always be defined at assembly time by invoking the appropriate directive MACRO with the symbol $$$GLB defined. We were fascinated to learn that object modules are, therefore, not upward-compatible even at 11M V2.

4. The SYSLIB module QIOSYM defines I/O function and status codes. Different symbols are defined in the two libraries. Many, but not all, of these symbols are associated with networking; it is hoped that differences we note are a reflection of the different development level of the two libraries and that this module will be compatible at V6.2 and V3.

5. The nonprinting terminal control characters CR, LF, HT, VT, etc. are defined in SYSLIB for 11D but not for 11M.

6. $DSW is defined in SYSLIB for 11D, internally by the task builder for 11M.

V. FEATURE DIFFERENCES

A. Message Output Handler MO

MO does not exist as a device handler in 11M, although there is a subroutine package available to do message formatting (SYSLIB modules EDTMG and EDDAT). A set of subroutines with the same names is available in 11D but has not been documented for D users to our knowledge; we assume it is the same package. The lack of MO in 11M is a problem only for existing 11D code. We have used the prefix file to declare all MO MACROs illegal.

B. Line Printer Spooling

RSX-11M spooling is much like that used in RSX-11D V4A. File output ("PUT$") to the printer is not automatically intercepted; utilities neither uniformly attach the printer nor does the spooler when it is employed. Mixed listings are a common occurrence. For "single job" systems, this is rated as a nuisance problem; for true multi-user systems, especially those in networks or with remote terminals, this type of spooling is totally unacceptable.

The 11D spooler is a vast improvement, but there are significant holes in that printing system also. PUT's to the line printer are automatically intercepted and spooled, QIO's are output directly. The 11D spooler does not attach the line printer. The consequences of this are most clearly revealed under Batch operations and so will be discussed in Section C.

PRINT$, the MACRO used to send a list file to the spooler, generates significantly different code in the two systems. The 11D spooler supports forms, copy, and deletion control; the 11M spooler has none of these features. For these reasons, we have declared PRINT$ illegal and are still looking for a simple and sensible way to handle these incompatibilities.

C. MCR Batch vs Indirect MCR

The full indirect MCR with question-and-answer facilities supplied with 11M has far more capability than MCR Batch provided for 11D. Unfortunately, indirect MCR does not collect listings as does Batch. Again, on a multi-user

system, collated listings are extremely useful; if remote terminals are in use, the collated listings with banner pages as produced by Batch are a requirement.

Just as indirect MCR can go through multiple levels of indirect files, so can Batch. This is accomplished by inserting a "$MCR BAT filespec" line in the batch input stream. We refer to this as "layered Batch" and make extensive use of the facility in management of the development disks. One should be aware that a layer of Batch cannot simply be substituted for a level of indirect MCR, for every Batch job will go to completion before the next is initiated. A line containing "$MCR BAT filespec" is complete when ...BAT has read the specified file and entered the job file in BPR...'s queue. In contrast, indirect MCR executes every command completely as soon as it is encountered, doing further look-up in indirect files if required. Hence, the order of execution will differ in the two systems, and this must be carefully taken into account.

The utility programs available under 11D output certain of their error messages via QIO's to TI, the calling terminal. When the utility is run under the normal unlayered Batch, this "TI" is the pseudo-device "BP," and BP is redirected by BPR... to the TI from which ...BAT was called. Now if ...BAT is called from BPR..., which is what must occur in a layered Batch, the TI of ...BAT is BP, so that BP is now redirected to itself. The error messages intended for the "terminal" are simply rejected by the system ("handler not resident") and are forever lost. One can, however, redirect BP to a useful physical device in the batch stream itself at all layers below the first in order to preserve the error messages. The line printer would be a natural choice, but because the messages are output through QIO's rather than PUT's, they then appear at bizarre spots in the line printer output. An SPR has been filed on the most elementary aspect of this problem (that error messages should be explicitly routed to the batch log in accordance with the spooling protocol of the system), and we are pleased to report that the problem has been fixed in IAS (!).

D. Multi-user Tasks

Under 11D, a multi-user task consists of a single copy of the read-only code plus a separate copy of the read-write memory for each active user; in 11M, a full copy of the multi-user task is required for each active user. This probably has little design impact but could tend to make some 11M systems overly core-hungry.

E. System Global Areas

Under 11M, an SGA is always resident; under 11D, an SGA is present in core only when required by a loaded task. This allows SGA's to be used as easily maintained and accessed scratch files under 11D. Furthermore, since symbols can be defined globally in SGA's, they are far superior for many uses. This incompatibility has extremely heavy design impact.

An 11D SGA which has been declared to be a COMMON area (/CM) is rewritten to disk whenever it is no longer bound in core by a task, thereby automatically providing a disk copy of potentially critical data. The same data preservation facility is available under 11M only by means of a manual SAVE of the entire system.

F. Logical Device Assignments

RSX-11D does not have 11M's logical device assignment capability. The closest simulation we have been able to achieve is to generate in the system the

pseudo-devices IN (input), OU (output), and LS (listing) and, by convention, to refer to only these devices in command files. The simulation is complete when, under 11D, the pseudo-devices are redirected to physical devices. Since redirect affects the entire device assignment structure of the system, disastrous results can occur if more than one user is referencing the pseudo-devices in an uncoordinated fashion. Furthermore, there is no mechanism within 11D which allows one to determine if a redirected device is in use; hence, only strict adherence to operating conventions (remote terminals may not redirect the pseudo-devices, redirection must be coordinated with all logged-on users, etc.) makes this style of operation practicable. We have found it to be both hazardous and difficult to schedule and control, but the utility of pseudo-devices for large program development jobs is judged to be so high that the risks are acceptable. We consider the lack of logical device assignments to be a most serious deficiency in RSX-11D since it detracts from the multi-user nature of the system.

G. Terminal Handler Inconsistencies

The differences in the two multi-terminal handlers are most conspicuous at the operator level, but the handling of unsolicited input (type-ahead in 11D vs "hot MCR" in 11M) has an effect on user code: We have a set of terminals where the user code must issue a write to turn on a cursor; in its own good time, the terminal will generate interrupts and characters giving the cursor position. To receive the position characters, the user task must do a read from the terminal. If the characters are returned before the read is issued, 11M's MCR will receive them with disconcerting results both to MCR and to the user code. Under 11D, the position characters go into the type-ahead buffer and are then picked up by the correct user task. The proper procedure is for the user task to attach the terminal before initiating the sequence, then issue an immediate read on the terminal. This solution admits a race condition in 11M since one has no guarantee that the read is truly immediate. We give this as an example of the class of problems that have plagued us since the beginning of the project. It demonstrates the impact that the lack of coordination between two systems has on non-privileged code.

H. Mag Tape

RSX-11M has no file-structured mag tape but does support it at the QIO level. In 11D terms, 11M supports mag tapes only as foreign volumes. 11M does not, however, recognize a MOUNT command for mag tape, even as a non-file-structured volume--an operation that is required by 11D. Hence, 11D tasks that do internal tape mount/dismount have required minor changes. We have yet to develop a mechanism to regain the small amount of protection afforded by 11D.

The mount/dismount problem is not unique to mag tape: No foreign volume (i.e., disk) is mountable under 11M. Again we have a case where uncoordinated system design has produced two conceptually different operations masquerading under the same name.

I. SYSGEN Memory Allocation

SYSGEN for 11M always places the executive and system dynamic memory at the bottom of physical memory. Under 11D, the executive and node pool can be placed anywhere in memory by appropriate choice of SYSGEN parameters. When the hardware configuration of a system includes extra processors or peripherals (some of

which are manufactured by DEC) that address a subset of physical memory, careful placement of the executive code is vital both to make the best use of system resources and, in some cases, to ensure the integrity of the monitor itself. This lack of flexibility in 11M is sorely missed.

VI. MANAGEMENT OF JOINT SYSTEMS

A. Management Problems

In the early stages of the Q conversion project, management of the joint M and D systems was not considered to be a severe problem. For instance, it was thought that parallel sets of Batch and indirect MCR command files would be maintained, as well as parallel task-build command files. However, as the conversion effort progressed, it became increasingly apparent that unless countermeasures were taken, system management problems would overwhelm the technical development aspects of the project. Of particular concern are the following:

1. The joint Q system is maintained on six RK05 disk packs and requires about 150 command files for M and a like number of command and Batch files for D. A complete M/D Q system contains about 1600 files. This is an enormous amount to keep organized.

2. Because of the simultaneous development of the Q system under D at one laboratory and M at the other, maintenance of parallel sets of command and Batch files soon becomes tedious and problematical.

3. Procedures for each laboratory to incorporate system modifications originating at the other lab must be straightforward and efficient. Reducing the number of files that must be kept in phase is a reasonable first step here. Hence, we conceived of a class of "primitive files"--the set of files from which all files in the system can be obtained. The primitive file set contains only source and "skeleton" files, as will be described below.

4. In accordance with our "Automation Rule," it must be possible to generate (i.e., compile, task-build, etc.) by mechanical procedures a complete system using only the primitive files.

B. UIC Conventions

To make the joint system manageable, we first laid down a set of UIC conventions. Files usable only with D go into UIC's of the form [abc,2de] (lower-case letters are octal digits); corresponding files usable only with M go into UIC's of the form [abc,1de]; files applicable to either D or M go into UIC's of the form [abc,de].

A simple example of a UIC set in which we have maximum file compatibility is shown below. The files with extensions SKC and SKT are skeleton files. The primitive files are A.SKT, A1.FTN, A2.FTN, F4P122026.SKC and TKB122026.SKC.

```
[122,6]
A.SKT
A1.FTN      A2.LST
A1.LST      A2.OBJ
A1.OBJ      F4P122006.SKC
A2.FTN      TKB122006.SKC
```

```
[122,106]
A.BLD       F4P122006.CMD
A.MAP       TKB122006.CMD
A.TSK

[122,206]
A.BLD       F4P122006.BIS
A.MAP       TKB122006.BIS
A.TSK
```

C. Skeleton File Concepts

A skeleton file is analogous to a command or Batch file, but each line contains information for both an M command line and a D Batch or command line. These may differ in both syntax and UIC usage, e.g.

```
M:  TKB @IN:[122,106]ABC.BLD
D:  $MCR TKB @IN:[122,206]ABC.BLD
```

Skeleton files are used to generate all Batch and command files in the system. In the example of the last section, [122,6]A.SKT is used to generate [122,106]A.BLD and [122,206]A.BLD. Similarly, the CMD and BIS files are generated from [122,6]F4P122006.SKC and [122,6]TKB122006.SKC.

To implement parallel command file generation, we devised a straightforward syntax for skeleton files (discussed below), so that TECO MACROs could be used at once to translate skeleton files into M or D files. To automate the creation of command and Batch files, more than simple translation is required, however. It is necessary to have a utility which also performs indirect operations. Consider then running this utility on a skeleton Batch/indirect MCR command file to generate (for use by M) an indirect MCR command file, which references other indirect MCR command files, which, in turn, reference utility (particularly TKB) command files, which may further reference .ODL files, and having all these referenced files automatically created from skeleton files (the old versions having been automatically deleted). The situation is analogous for D, with the identical skeleton files being used to generate corresponding Batch, command, and .ODL files. In either case, a single keyboard command causes all of the file generation mechanics to occur; and subsequently, a single MCR command line causes all of the generated files to be utilized by Batch/indirect MCR, TKB, etc. The procedure for "building a disk" (by which we mean generating all desired M or D .OBJ files, .TSK files, etc., starting from only skeleton and source files) has, in fact, been automated into a two-step procedure:

1. Initiate processing of a master skeleton file, which, in turn, causes all other referenced skeleton files to be processed.

2. Initiate Batch or indirect MCR on the master Batch or indirect MCR command file generated by step 1.

The structural relationship of the referenced skeleton files is that of a tree: The master skeleton file branches out to (i.e., references) other skeleton files, which, in turn, branch out to others, etc. The processing of the master skeleton file results in the creation of an isomorphic image tree of Batch/indirect MCR command files and utility command files. The processing can,

in fact, be initiated at any node in the skeleton file tree, yielding a corresponding subtree of the image tree. Analogously, utilization of the image tree by Batch, indirect MCR, or a utility may be initiated at any node.

D. Skeleton File Syntax and Processing

The program which does the processing of skeleton files is called OST--for "OSTeopathic Translation Program." We now discuss briefly the syntax it recognizes and give some examples of the operations it performs. In a skeleton file, portions of a line pertaining to M are delimited by '#,' and portions pertaining to D are delimited by '%.' Thus, the skeleton file line

```
%$MCR %PIP IN:[122,%2%#1#06]ABC.BLD;*/DE
```

is translated into

```
M:  PIP IN:[122,106]ABC.BLD;*/DE
D:  $MCR PIP IN:[122,206]ABC.BLD;*/DE  .
```

For convenience, OST was made to recognize '%xxx#yyy#' as an abbreviation for '%xxx%#yyy#.' Furthermore, end-of-line terminates any construction. We now have a simple example of the usage of OST:

```
File 'IN:[1,1]OSTDISK.CMD':

     IN:[1,%2#1#01]DISKBUILD.SKC

File 'IN:[1,1]DISKBUILD.SKC':

     %$JOB/NAME=OSTDEMO/MCR
     %$!#;#COMMENT
     %$MCR %PIP OU:[122,%2#1#02]ABC.TSK;*/DE
(*)  %$MCR %TKB @IN:[122,%2#1#06]ABC.SKT

File 'IN:[122,6]ABC.SKT'

     OU:[122,%2#1#02]ABC/-FP=IN:[122,%2#1#06]ABC
     /
     LIBR=%OTSCOR#SYSRES#:RO    %;FORTRAN OTS
     //
```

The MCR Command 'OST @IN:[1,1]OSTDISK results in the creation of the following files, after old versions have been deleted (note the translation of file types, e.g., .SKC to .CMD (M) or .BIS (D)):

```
(M)  File 'IN:[1,101]DISKBUILD.CMD':

     ;COMMENT
     PIP OU:[122,102]ABC.TSK;*/DE
     TKB @IN:[122,106]ABC.BLD

(M)  File 'IN:[122,106]ABC.BLD':

     OU:[122,102]ABC/-FP=IN:[122,106]ABC
     /
```

```
LIBR=SYSRES:RO
//
```

(D) File 'IN:[1,201]DISKBUILD.BIS':

```
$JOB/NAME=OSTDEMO/MCR
$!COMMENT
$MCR PIP OU:[122,202]ABC.TSK;*/DE
$MCR TKB @IN:[122,206]ABC.BLD
```

(D) File 'IN:[122,206]ABC.BLD':

```
OU:[122,202]ABC/-FP=IN:[122,206]ABC
/
LIBR=OTSCOR:RO     ;FORTRAN OTS
//
```

In an M system, the MCR command '@IN:[1,101]DISKBUILD' causes task-building to occur as delineated by the two M files shown above. In a D system, the MCR command 'BAT IN:[1,201]DISKBUILD' yields analogous results using the two D files shown above.

In processing the line marked (*), OST recognizes the presence of '@' in conjunction with the skeleton file type .SKT ("skeleton task-build"). The following actions pertaining to M are then taken (those pertaining to D are analogous):

1. Line (*) is translated into 'TKB @IN:[122,106]ABC.BLD' and output to file IN:[1,101]DISKBUILD.CMD.

2. The file specifier 'IN:[122,106]ABC.SKT' is extracted from line (*) and the owner number hundred's digit is dropped to yield filename IN:[122,06]ABC.SKT. This skeleton file is then opened for input; all existing versions of IN:[122,106]ABC.BLD are deleted, and a new file with this name is created.

3. After construction of IN:[122,106]ABC.BLD is completed, processing of skeleton file IN:[1,1]DISKBUILD.SKC is resumed.

VII. CONCLUSIONS

We wish to emphasize that this report represents only a preliminary study of 11M/11D compatibility problems. New difficulties come to light almost daily, and we find our perspectives continually evolving as we develop new techniques to deal with them. Much more work remains to be done: A systematic comparison of the two SYSLIB's would undoubtedly be revealing. I/O status codes returned by comparable drivers, cross-referenced by function code, is an untouched area. The terminal handlers merit special study. A systematic survey of differences perceived by an operator at a terminal would make it clearer to us why we personally experience a certain level of confusion as we move back and forth from one operating system to the other. The two FOR compilers have not been fully explored, but we have mercifully restrained ourselves from detailing for you the extra dimension of compatibility problems we find in the M vs D/FOR vs F4P matrix. Many areas that we have probed are still producing surprises. Nonetheless, we feel that our experience should be made available both to users

contemplating a similar conversion and to those who are now selecting one of these two operating systems for the first time.

In summary, then, we find the intersection of RSX-11M and RSX-11D to be small indeed. Since RSX-11D is now considered to be a mature and stabilized system (in the sense that no significant development work is anticipated), one might hope that development resources could now be assigned to 11M. This effort, if properly and sensibly directed, could then be used to provide a feasible migration path for those 11D users who are attracted to new 11M capabilities such as PLAS. We therefore hope that 11M will be extended toward achieving the fundamental system strength and real-time capabilities that 11D has today rather than toward new "add-on" features intended to make the system marketable in even more diverse areas.

Our specific recommendations are as follows in priority order:

1. Implement the remaining RSX-11D executive directives in RSX-11M. Essential directives are:

   a. Cancel mark time by AST address or by event flag.

   b. Receive from specified sender.

   c. "Send and Request or Resume" and "Receive or Suspend" compound directives. (We note that an enable/disable context switching would allow emulation of the RSX-11D forms.)

   d. Variable length send and receive directives.

   e. Wait for Logical Or on group 4 (Event flags 1-64).

2. Implement the RSX-11D handling of System Global Areas in 11M.

3. Form a SYSLIB that is execution-level compatible. Supply a pair of secondary concatenated object module files to reflect RSX-11M and RSX-11D features and differences.

4. Implement indirect MCR (11M format and functionality) in RSX-11D. This includes logical device assignments.

5. Implement substantial line printer spooling capability with deletion control under RSX-11M.

6. Allow user-specified placement of the RSX-11M executive in physical memory.

Item 1 is in our opinion of extreme urgency and importance. We note that Item 4 enables RSX-11D users to make a more orderly transition to RSX-11M.

ACKNOWLEDGMENTS

The authors wish to thank Martin Kellogg for his technical assistance in unravelling the true source of some of the incompatibilities described in this report. Martin's clear understanding of system processes proved an invaluable reference throughout the conversion project.

REFERENCES

1. "Data Acquisition Program Q," Los Alamos Scientific Laboratory Group MP-1 internal report, July 1976.

2. "RSX-11D Executive Reference Manual," Digital Equipment Corporation, DEC-11-OXERA-B-D, May 1975.

3. "Introduction to RSX-11M," Digital Equipment Corporation, DEC-11-OMIEA-A-D, May 1974.

4. "IAS/RSX-11 MACRO-11 Reference Manual," Digital Equipment Corporation, DEC-11-OIMRA-A-D, December 1975.

☆ U.S. GOVERNMENT PRINTING OFFICE 1977—777-018/69