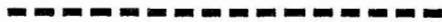


# THE DATA MANAGERS

NEWSLETTER OF THE DATA MANAGEMENT SYSTEMS SPECIAL INTEREST GROUP

November Vol. 5 -- No. 1 1982



Please address contributions to:

Paul D. Clayton, Editor  
c/o DECUS  
One Iron Way, MRO2-1/C11  
Marlboro, MA 01752

	Page
Fall Symposium Sessions .....	3-21
RMS-11 Internals .....	4-23
Additional FMS Options .....	4-21
Adabas-M .....	5-23
TECO Questions .....	6-12
Word Processor Feedback .....	8-11
10/20 Menu Response .....	5-21

**RECEIVED**  
DEC 03 1982

Copyright © Digital Equipment Corporation 1982  
All Rights Reserved

**ACCOUNTING DEPT.**

It is assumed that all articles submitted to the editor of this newsletter are with the authors' permission to publish in any DECUS publication. The articles are the responsibility of the authors and, therefore, DECUS, Digital Equipment Corporation, and the editor assume no responsibility or liability for articles or information appearing in the document. The views herein expressed are those of the authors and do not necessarily express the views of DECUS or Digital Equipment Corporation.

## Pac-Man game apes life: Needs cunning

## and digital dexterity

When the going gets tough, they used to say about Richard Nixon, the tough get phlebitis.

I'm not tough. I go out for Pac-Therapy.

It doesn't cure writer's block. It merely delays worrying about it.



Pac-Man is the latest video-game rage. It is a game about devouring, replete with psychosexual overtones. It appeals to those who find shoot-em-ups like Space Invaders or Asteroids unfathomable.

Pac-Man requires total concentration, cunning and digital dexterity — just like real life — and it only costs a quarter.

The player controls Mr. Pac, a little yellow thing with a clacking mouth who tries to eat all the dots on a maze-like screen while avoiding four other gizmos who chase Mr. Pac relentlessly.

When the player executes a certain move, the four enemy eaters turn blue and, for a few seconds, Mr. Pac becomes the aggressor, gobbling them for bonus points.

The attackers have cute nicknames provided by the game manufacturer, but no serious Pac-person knows or know, does not employ the Robert Extension, named after our bar's No. 2 player. Robert has been interviewed by three newspapers and a wire service.)

At the beginning of Donna's Detour, those little suckers are supposed to turn blue.

On the fifth key, she learned, they don't. They keep coming.

She died on the fifth key.

Now it is part of Pac-Lore. The underground is full of such stories.

The underground, for example, has found an arcade off South st. where they award free Pac-Man T-shirts for anyone who tops 100,000. The underground also reports that machines in New York cost 50 cents a play and those in some cities give five chances (instead of three) for your quarter.

The most chilling rumor is that the West Coast and Europe have been repro'd.

Reprogramming is the nightmare of the electronic gamesman. Every Pac-Man I've found in Philadelphia works exactly the same — for now — except the one across the street at Cavanaugh's Railroad Bar.

They got a new machine a month ago. I put in my quarter, tried my Cherry moves. I died. It was so bad, I had to go back to work.

It's taken dozens of quarters, but I'm developing new

cares from nicknames. The enemy has no face.

What the serious players do — I'm unofficially No. 6 in my favorite bar — is develop new systems and compare notes.

Notes, and then some.

Donna Shaw, a feature writer at The Bulletin, is No. 5 in the bar. She is a squealer. When Donna's on the machine, Pac-Man turns into a spectator sport.

The morning after, she shows off her calluses. They develop on the fourth finger, right hand, and the thicker they are, the better she's doing.

One day she came by the desk in high excitement. She had broken a blister! I examined it and offered congratulations, with a trace of envy.

She returned a few days later with a tale of triumph and horror in Chinatown.

It had been her Game of a Lifetime, breaking through the 100,000-point barrier. She had gone through the cherries, oranges, apples, limes (or are they bunches of grapes?) that designate each succeeding screen. She was up to the fifth housekey, meaning she had cleared the screen 16 times.

She swung into her Apple Move, a popular technique that begins with the apple screen and must be mastered to graduate from novice ranks.

She continued into Donna's Detour. (Donna, you should moves for the Cavanaugh repro. I'm up to 93,000, which is about 50,000 below my all-time game. (I died on the fifth key, too).)

I prefer bars to arcades, but that presents a problem. Two beers raise your competitive edge. After that, the little gizmos seem to get a lot smarter.

The drawback to arcades is teen-agers. They are very good. They also come with large, albeit silent, cheering sections. You know what they're thinking, and it can be intimidating.

If it's not teens, it's teasers. Some players tempt and torture those enemy gizmos forever, trying to line them up for a bonus kill. If you're playing against a teaser, you can go out for lunch between turns. When the teaser finally misses, the gizmos are so furious they take it out on you.

None of this makes any sense to those who live bleak and unfulfilling lives, bereft of Pac-therapy. To you, I pose the ageless question:

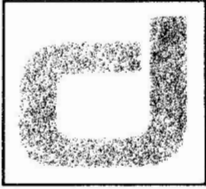
"Got four quarters for a dollar?"

*Ron Goldwyn*

Ron Goldwyn's column appears Sunday, Tuesday and Thursday.

Reprinted by permission of the Philadelphia  
Bulletin and the Register and Tribune  
Syndicate





**datacomp**  
**CORPORATION**

5 Echelon Mall  
Voorhees, NJ 08043  
(609) 772-9090

Philadelphia  
Birmingham  
Boston  
Cleveland  
New Haven  
Voorhees, NJ  
Washington

10/22/82

Sorry this issue took so long to get on the streets. I have changed jobs recently placing me into the "wonderful" (?) world of VAX 11/780 in a production shop.

At the present I'm without word processing capabilities so input from me will be brief and to the point.

Articles are needed for the next issue!!!!!!!!!!!!!!!!!!!!

If anyone has any questions or suggestions please call me at 609-772-9090, your input is needed.

*Paul D. Clayton*  
Paul D. Clayton  
Editor

New Address:

Datacomp Corp.  
5 Echelon Mall  
Voorhees, NJ  
08043

The Spring Symposium in Atlanta is passed, but a few moments reflecting might be worthwhile. The symposia in general was dominated by Digital Equipment's great leap into the personal/professional work station market with Rainbows, DECMATES and Professionals. Look for these to dominate the next few symposia. For the DMS Sig, Atlanta was a time to experiment and broaden horizons. Along with the VAX and Datatrieve Sigs we put on a group of sessions on Tuesday entitled VAX Information Management Theme Day. Begun with a keynote address by noted data base authority Leo J. Cohen, theme day was chock full of worthwhile management and technical oriented sessions. Look for other theme days in the future.

The Sig also took the opportunity in Atlanta to open its arms to issue oriented and application oriented interest areas. Two areas met with enough interest to create formal Working Groups. Data Management for Decision Support Systems and Data Management in the Scientific Environment are the new working groups and will be presenting a slate of sessions in the fall at Anaheim. We look forward to identifying other areas where the Sig ought to be concentrating efforts and I invite anyone having an idea to drop me a line.

In the area of Product Groups, which have been the traditional center of interest for the Sig, VAX II DBMS/CDD, FMS and TECO all exhibited great user enthusiasm. DBMS-II (PDP Systems) and RMS generated little or no user support. If you DBMS-II or RMS users are out there and want symposia activities, let's hear it from you.

Well so much for Atlanta. I'm sure Paul Clayton, our newsletter editor, has put together another useful issue. Let him hear your thoughts on that as well. L.A. will be upon us before you know it. December 6-10 are the dates. Send in your hotel registrations early. The deadline for session commitments (Call for Participation - CFP) is July 2, 1982. You are all invited to submit user papers.

All of our above mentioned Working and Product groups will be represented. Additionally, an Editors working group will try to form at Anaheim. As usual we expect some non-DEC vendors will be submitting generic sessions to enhance our knowledge of data management. In particular Herb Edelstien of International Data Base Systems, Inc. and Jim Starkey, the DEC Wombat, have promised to continue their entertaining Atlanta discussion on CODYSAL and Relational systems. Digital as usual will be out in full force. If you have any particular requests, get to me or any other steering committee member fast!

Look forward to seeing you all in Anaheim.



Sandy Krueger.

The Anaheim Symposium promises to be exciting and busy for the Data Management SIG. Many good sessions were submitted and every effort was made to schedule as many as possible. There is not as much time for lunch or dinner but fortunately there are only two nights dedicated to sessions.

Monday morning at 9:00 a DIGITAL V.P. will discuss Computer Trends in the 80's from a manufactures viewpoint. The PDP11 keynote speaker, early Monday afternoon, will discuss the impact of new DIGITAL products and the relationship of DECUS to these products. The DMS SIG opening, combined with the DATATRIEVE SIG, will be scheduled between these two global sessions and will pinpoint all the sessions relating to information processing across all operating systems.

Monday afternoon schedule includes:

1. VAX/FMS Overview: A presentation of the individual components of FMS.
2. VAX/FMS Forms Definition: Details on how to create form definitions from an application program.
3. TECO/VTEDIT Internals: An explanation of the programming structures internal to TECO.
4. DBMS-20 Structures: A technical session directed to DBMS-20 administrators and technical support personnel presented by DEC.
- ~~5. To Do or Not to Do: Rick Landau will discuss the conditions and advantages of file management vs data base management under various applications.~~
6. The Process of DBMS Evaluation: A case study of evaluating commercial DBMS products based on functional specifications and critical design features.
7. What is a Distributed Data Base?: A panel will discuss their experiences with sharing data from multiple computers.

Monday evening:

1. Virtual Record Locking in RMS Under RSX-11M/M+: A solution for allowing multiple users access to the same data base.
2. Integrating VAX/FMS, VAX-11 CDD, VAX-11 DTR, VAX-11 DBMS: DEC experts will be available during this session to help with any integration problems encountered with the above mentioned software.
3. FMS VI Internals and Magic: Tricks, magic and programming techniques for FMS VI, including an opportunity to examine source listings.

Tuesday's schedule includes:

1. Enhancements to Keypad EDT with Line Mode Commands Via Initializer: A user presentation on increasing the capabilities of EDT.
2. Real Time Automated Inventory Control with RMS: A description of RMS organization of data to efficiently process warehouse transactions.
3. Data Base Clinic: An opportunity for all users of DEC supported DBMS's to work out ideas and problems with developers and experienced users.
4. VAX/FMS Programming: Emphasis will be on programming techniques to aid in development of applications programs using FMS.
5. Why TECO is not an Editor: Comparison of TECO with other text processing languages.
6. VAX-11 DBMS Technical Tutorial: DBMS Development Group from DEC will describe the features of VAX-11 DBMS.
7. VAX-11 DBMS User Panel: Current VAX-11 DBMS users will present their application and use of DBMS.
8. Introduction to ADE for Non-Programmers: Session is aimed at explaining the capabilities and features of the Applications Development Environment software.
9. Implementation of a Data Base Editor: A panel will describe user written utilities which have proven useful at their sites.
10. Implementation of a Data Base Editor: A generic designed editor allowing full-screen, field oriented editing for quick retrieval and update of data base fields.
11. Knights of the Data Base Roundtable: Four knights will defend the virtues of relational vs flat file vs CODASYL vs data base machines.
12. RMS - Version 2 Product Panel: The overview will include new access methods, new functionality in standard utilities, and new utilities.

Wednesday's schedule:

1. VAX-11 Common Data Dictionary: A technical presentation on managing the CDD of interest to DATATRIEVE and DBMS users.
2. Practical Usage of VAX-11 CDD: Panelists will present their experiences with CDD.
3. Data Management Tools in the Scientific and Engineering Environment: Panel members will show how their particular data management tools have either improved productivity or provided new capabilities.

- 4. Comparing INDENT and FMS: With both available on VMS and RSTS, this session will address the strengths and weaknesses of each product.
- 5. Is DBMS-10 Dead?: An opportunity for DBMS-10 users to organize their concerns about the direction and commitment of DEC toward their product.
- 6. RMS-10/20 Data Structures & Performance Comparison: Included will be comparative test data on the performance of RMS, DBMS and ISAM systems on TOPS-10/20.
- 7. ~~Fourth Generation Languages: to cover the functions required to establish a high productivity programming environment for developing application software.~~
- 8. Data Base Performance: Outlines a methodology for evaluating data base application performance.
- 9. Data Base Modeling and Analysis Techniques: A presentation illustrating a method of viewing a problem area and devising a representation of data which is more effective.
- 10. ~~Data Base Analysis, Design and Implementation: Introduces an integrated design methodology for development of a management information or decision support system.~~
- 11. VAX/FMS Advanced Programming: Techniques that utilize the most sophisticated features of VAX/FMS.

Thursday's session:

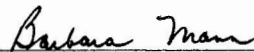
- 1. MIS Strategies for Decision Support Systems: This session will describe Decision Support Systems and their differentiation from traditional business application systems.
- 2. VAX-11 DBMS Data Base Design and Tuning: A description of the factors affecting performance of VAX-11 DBMS applications.
- 3. Data Base Security: Discussion of security with emphasis on future features of VAX-11 DBMS.
- 4. On-Line Updating with VAX-11 DBMS: Addresses the problems and considerations for effective design necessary for on-line updating of a data base.
- 5. RSTS/INDENT Hints & Kinks: Designed to help you solve your own problems or questions about using INDENT.
- 6. Future Trends in Data Dictionary: A discussion of a high level data dictionary and the problems and benefits of using it.

- 7. Data Base Administration - Controls and Standards: A how-to session describing considerations and procedures for standards implementation.
- 8. ~~VAX Applications Management Approach: to cover a new approach for terminal application development.~~
- 9. User-Friendly VAX Interface to a High Performance Back-End Relational Data Base Processor: A successful implementor will present this session.
- 10. Future Trends in Interactive Display Methods: How data capture and manipulation on CRT terminals will be handled in the future.
- 11. Benchmark of VAX-11 DBMS and SEED CODASYL Data Base Management Systems: Presents the benchmark criteria, system performance and performance of each DBMS executing a common data base schema.
- 12. Testing and Tuning the VAX-11 DBMS for Order Processing End-of-Day: Will report the results of tests conducted in a single stream, single user environment and will include recommendations for tuning VAX-11 DBMS.
- 13. Implementing a Very Large DBMS-20 Data Base: This paper will offer guidance for any site implementing a data base.
- 14. DBMS for the RT-11 User: Implementing the benefits of DBMS software on small DIGITAL systems.

Friday's sessions include:

- 1. DMS SIG Closing: An opportunity to evaluate the week and help define needed sessions for next symposium.
- 2. ~~VAX Application Management Approach: A Technical follow up to overview session offered earlier in the week.~~

In addition, there will be working group meetings for the DMS related product groups and subgroups. These are open meetings and all interested people are urged to attend those of interest. Your continued support and ideas are necessary for maintaining quality sessions. I know the Anaheim Symposium will be worthwhile and I hope many DECUS members will have the opportunity to attend.

  
Barbara Mann  
DMS Symposia Coordinator

A Brief Overview of RMS-11 Internals

BY

Phil Stephensen-Payne  
c/o Systime Ltd.,  
432 Dewsbury Road,  
LEEDS LS11 7DF,  
England. Tel: (0532)-702211  
October 1980.

INTRODUCTION

RMS-11 forms a very important part of many applications and systems on PDP-11 processors running RSX-11M, RSTS/E or VMS and yet there seems to be very little documentation on the internals of the file system, particularly compared with the wealth of information available on its older cousin FCS, while the new documentation that was released with RMS-11 V1.8 is vastly superior to that provided previously, it still does not contain enough information for the user to determine exactly what the format of his files on disk is, or even exactly how much space in his program is required. This document aims to fill in some of those gaps (with emphasis on RSX-11M) and to provide enough basic information for the interested user to work from.

The information for this document has been garnered from six places:-

- a) RMS-11 User's Guide, No. AA-D538A-TC, dated March 1979.
- b) RMS-11 Macro-11 Ref. Manual, No. AA-H683A-TC, dated March 1979.
- c) RMS-11 Design & Logic Manual dated July 1977.
- d) The macros supplied with RMS-11 V1.8 in RMSMAC.MLB.
- e) Dumps of various test files.
- f) The RMS-11 Prototype ODL file RMS11.ODL.

In general information contained adequately in the first two (hereafter referred to as RUG & RMR) will not be repeated on the assumption that every user has access to them. Odd pieces of information garnered from c) & d) have been included even when I am unsure as to their meaning or relevance.

The document is divided into five major sections with four appendices at the end amplifying and illustrating the preceding material:-

- Chapter I. RMS-11 File Layouts.
- Chapter II. RMS-11 In-Core Table Layouts.
- Chapter III. RMS-11 Macros.
- Chapter IV. RMS-11 Subroutines.
- Chapter V. RMSBCK/RMSRST Tape Formats.
- Appendix A. Formatted Dump of an RMS-11 Indexed File.
- Appendix B. Some additional RMS-11 Design/Logic points.
- Appendix C. Some RMS-11 Design/Logic Algorithms & Flowcharts.
- Appendix D. RMS-11 Mnemonic Glossary.

Table of Contents

INTRODUCTION .....	2
RMS-11 File Layouts .....	3
File Header .....	3
Sequential Files .....	5
Relative Files .....	6
Indexed Files - Prologue .....	8
Indexed Files - Bucket Headers .....	14
Indexed Files - Record Headers .....	16
RMS-11 In-Core Table Layouts .....	20
User-Visible Tables .....	20
Internal Tables .....	37
RMS-11 MACROS .....	54
General .....	54
Field Definition Macros .....	55
Field Initialization Macros .....	56
Pool Space Definition Macros .....	59
Dynamic Field Access Macros .....	60
File and Record Operation Macros .....	61
Others .....	62
Debugging System .....	63
Macro Cross-Reference .....	66
Doing without the Macros .....	69
RMS-11 Subroutines .....	74
RMSBCK/RMSRST Tape Formats .....	91
Formatted Dump of Indexed File .....	93
Some Additional RMS-11 Design/Logic Points .....	113
RFA .....	113
Record ID .....	113
VFC Print Control Information .....	114
The 4th File Type .....	115
Bucket Splitting .....	115
RMS-11 Interlocking Algorithms .....	116
Subprogram Interfaces and Calling Sequences .....	120
String Collating Sequence .....	120
Unusual Conditions Treatment Philosophy .....	121
Some RMS-11 Design/Logic Algorithms & Flowcharts .....	122
Bucket Split .....	122
Record Insertion .....	123
Functional Flowcharts .....	134
RMS-11 Mnemonic Glossary .....	138

APPENDIX B

Some Additional RMS-11 Design/Logic Points

The overall design/structure of RMS-11 is fairly well documented in RUG, but a few points have come up either from the investigations above or from the RMS-11 Design & Logic Manual which might be worth mentioning separately. Those taken from the manual are in general quoted verbatim and have not been checked.

a) RFA

Curiously enough the RFA is never actually defined anywhere (that I could find). However from the structure defined above and a little experimentation it is fairly easy to deduce that:-

RFA Word 1 = 1st word of VBN of bucket holding RRV  
RFA Word 2 = Record ID of RRV in bucket  
RFA Word 3 = 2nd word of VBN of bucket holding RRV

It is interesting to note that even using RFA access, RMS must still search through the relevant bucket sequentially to find the requested record.

b) Record ID

The record ID is a fairly interesting field. As seen above in the file structures it is only one byte long and hence has a maximum range of 0 - 255 (decimal). Given that record IDs can never be reused (or you would have a new record with the same RFA as an old deleted one - which is illegal) this means that during its ENTIRE HISTORY no bucket may hold more than 255 records. If the last ID is used, then as records are deleted from a bucket more and more space is left unused!

It is also interesting to note that it was originally intended to handle this problem. The Design Manual entry on the LID field in the Bucket Header states:-

"When the contents of BSNID are greater than the contents of BSLID or is zero then there is no 'next' available ID. When this condition occurs the bucket is scanned to find the largest contiguous range of unused IDs and BSNID and BSLID are updated to describe that range. This procedure is deferred until we must do it (i.e. we wish to store a record in the bucket), and by updating BSNID and BSLID with the ID of a record removed from the bucket when it is appropriate to do so."

No mention is made of how the RFA problem would be overcome by this method. It is also interesting to note a legacy of those ideas remains in the module R3BSRT which currently contains only the code:-

```
CLR 10(R4)
SEC
RTS PC
```

Even when believing such ID 'reuse' was possible, it was still recognized that the method had limitations as the RMS-11 Design & Logic Manual also states:-

"The combination of large buckets and a very volatile file (or one in which the records are inserted largely in inverse order of Primary Key value) can lead to inefficient space utilization: once RRVs (or vestigial headers from deleted records) have used up the 255 valid ID values in a bucket, any remaining space

cannot be used and the bucket becomes static. Occasional rebuilding of such highly volatile files may be worthwhile in some cases, but this will invalidate any old RFA values which the user may have stored."

c) VFC Print Control Information

The RMS-11 Design & Logic Manual mentions the concept of a 'two byte print control field' in records with FB\$PRN set in F\$RATT and FB\$VFC in F\$FORG, even though it is not currently supported by RMS or FCS. The idea was:-

"Byte 0 is print control information to act upon before the record data is output to a unit record device & byte 1 is print control information to act upon after the record data has been output to a unit record device.

The format of each byte is as follows:

Bit 7	Bits 6-0	Meaning		
0	0	No carriage control		
0	count (1-127)	"count new lines" (CR/LF)		
Bit 7	Bit 6	Bit 5	Bits 4-0	Meaning
1	0	0	ASCII C0 Set	ASCII Char to Output (CR, FF, etc.)
1	0	1	ASCII C1 Set	ASCII Char (8 bit code) to output
1	1	0	Code (0-63)	Device specific code
1	1	1	-	Reserved

d) The 4th File Type

Every now and again reference is made in the RMS-11 Design & Logic Manual to a fourth file type, for which the modules would be called R4xxxx and so on. (Some vestigial remains exist in entry points such as \$CLO4E.) When referred to it is usually given the reference 'DIRECT' but it is never mentioned what it was to have been.

e) Bucket Splitting

RUG (5-17/18) gives a detailed breakdown of all the overheads of Bucket Splitting. It is possibly worth quoting a comment in the RMS-11 Design & Logic Manual that follows the  $(n**2+11n+20)/2$  derivation:-

"For completeness, one should mention that there is one slightly worse case of splitting. When a record is inserted into the middle of a bucket, it is possible that it will be so large that it will fit neither with the set of records which have lower Primary Key values nor with the set which have higher Primary



Key values. In this case, a 'three-bucket-split' occurs: the lower-valued records remain in the original bucket, the new record gets a bucket of its own, and the higher-valued records are placed in the third bucket; the Level 0 horizontal chain and the rest of the index are updated accordingly. One should realise that a three-bucket split can occur only in the Primary Index, and only at Level 0."

#### f) RMS-11 Interlocking Algorithms

This is covered in two areas in the RMS-11 Design & Logic Manual as follows:-

"One might reasonably inquire why it is necessary to return to the Root of the index and work down for each level of index updating. This is a consequence of the fact that others may be reading, or even writing to, the file while this potentially lengthy updating process is going on. Some form of interlocking must be employed to prevent one user from following a not-yet-updated pointer in the index which no longer correctly represents the just-updated bucket to which it points, to prevent multiple writers from simultaneously accessing the same bucket for update (if this were allowed to happen, the update caused by the first writer to finish would be over-written by the second), etc.

RMS-11's 'top-down' approach guards against this in the following manner. First, on every excursion downwards through the index (for both read and write operations), the Root bucket is read and locked against access by any other user. The appropriate pointer is followed down to the next index level, and the corresponding bucket at that level is read and locked. Now the Root bucket is released and the process is repeated: the appropriate pointer is followed down another level, the target bucket is read and locked, and the previously-locked bucket is released (unlocked). In this manner one proceeds down to the Data Level of the index, finishing with only the target Data Level bucket locked; if at any point a non-Data Level target bucket is discovered to be locked by someone else, the existing higher-level lock is released and RMS returns to the Root to try again, since the algorithm guarantees that the other user's lock will be released fairly soon.

This completely describes interlocking on \$GET/\$FIND operations. On write operations, the interlocking process continues in a more involved manner:

Upon reaching the Data Level, the insertion is performed. If no splits occur (and there are no Alternate indexes) the operation is complete, and the Data Level bucket is released. Otherwise, a lock is retained on the Data Level bucket containing the new (or updated) record until all index updating is done (including Alternate Indexes, if any): this guarantees that no other user will be able to access the affected record in any manner until all necessary index corrections have occurred.

Assuming a Data Level split has occurred, RMS returns to the top of the index and steps down to Level 1 using the same 'lock next/unlock previous' method described above. Again, if any target bucket is discovered to be locked by someone else, RMS returns to the Root to try again. When Level 1 is reached, the target bucket is updated; at this point, only the Level 1 target bucket and the Level 0 data bucket are still locked.

Should a split occur in the Level 1 target bucket, the lock on that bucket is retained while RMS again returns to the Root and steps down, as before, to Level 2. Once the Level 2 target bucket has been successfully locked, the Level 1 bucket is released (but the Data Level data bucket lock remains). Should a split occur on Level 2, the Level 2 target bucket lock is retained until the Level 3 bucket which points to it has been successfully locked, and so on.

The consequences of this algorithm are the following:

1. Since the bucket containing the data record is locked throughout the index update(s), no race conditions can develop wherein a second user modifies a record (and the resulting index entries) before the original index entries have stabilized.
2. Any second user stepping down through an index which is being updated will encounter a lock on any bucket which is not correctly described by the index level above it. This second user must then unlock the previous bucket and return to the Root to try again. Meanwhile, however, the task which was doing the updating will also be attempting to come down through the index to update the bucket which the second user has just released. The possibility that a second collision will occur at this bucket is very slight, and dependent on timing; in any case, no real deadlock can occur, and the update will eventually succeed.
3. There is never any need to retain locks on any new buckets created by splitting above the Data Level, since the algorithm guarantees that there will always be locks higher in the index which prevent vertical access to these new buckets until the entire index has been suitably modified. In the event of non-fatal corruption in the index, RMS does make provision for using the horizontal pointers in Level 1 and above to work around the corrupted area. Even in this case, however, the updating of the level in question will be complete before access can occur (due to locks at that level), and since all lower levels have already been appropriately modified the path downwards from that point will be valid.

4. The maximum number of buckets which must ever be simultaneously locked is 4: the bucket containing the actual data record, the most recently updated index bucket, and the two locks which must be used in the process of stepping down through the index (this will later be true for the Alternate index updates as well). On \$GET/\$FIND operations, only the two 'stepping down' locks will be needed. Since a Buffer Descriptor Block (BDB) is required for each simultaneously-locked bucket, this establishes the minimum number of BDBs required for read and read/write operations."

and:-

#### "Locking Modes.

When a file is opened, the locking mode of the file will be determined from the user's file-access fields (FAC, SHR). These modes determine how much locking is necessary for future record operations. The possible locking modes are:-

1. Exclusive access - no locking is done.
2. Read-delete access - indices are not locked, data records are locked.
3. Write access - indices and data buckets are locked.

Note that the inclusion of \$DELETE in mode #2 implies that all deletions are 'fast' - i.e. no bucket compression is done. This type of space reclamation is done only when necessary during record insertions.

#### Horizontal Locking (Forward).

Forward horizontal index locking is done in a similar manner to downwards locking. If the target bucket is already locked, backtracking is necessary. The task must release the current bucket and attempt to lock it again. When this is done, the horizontal pointer is fetched again and a second attempt to lock the next bucket can be made.

#### Horizontal Locking (Backward).

Backward horizontal locking is a potential in only one situation: an RRV must be deleted or modified. In this case the 'locker' may 'block' until the lock is granted. This is because all forward locking is done only with backtracking, so that deadly embrace is not possible. Since backward locking is done when bucket compression is a possibility, it is advantageous to give priority to such a process.

#### Secondary Index Locking.

Secondary indices are generally locked in a manner similar to primary indices. However, there are some minor complications which are listed below:

1. When a record is inserted, the primary index is used to determine the correct data bucket in which it should go. Next, the secondary index must be traversed in order to find the correct index data bucket into which the new index data record should go. After insertion of the index data record, the index may have to be locked from the top again if bucket splits occur. Therefore, the worst case is that the index is locked all the way down for each index level which splits.
2. The Data bucket remains locked during all updating operations on secondary indices. Therefore, other users may not access the Data bucket until the entire updating operation is completed. This is to avoid the situation where a secondary index has a key value which is different from the corresponding data record key value.
3. During secondary index updating, each index is locked in numerical order of key of reference, and is unlocked immediately after modification is complete. Since RMS-11 will not support the combination of 'no duplicates' and 'key value may change' attributes on secondary keys, updating operations will not have to be 'unwound'.
4. If a data bucket split causes the target record to be moved into the newly acquired bucket, then the new bucket must be locked also. This is not required for primary index operations because, since the index which points to the new bucket is locked, no access to the bucket is possible.

#### Alternate Techniques

Another scheme for locking of secondary indices is still under consideration. Instead of locking the index on each level, this alternative would lock the entire index whenever a modification to it was to be done. Because the 'coarseness' of this locking scheme is obviously much greater than the first technique, simultaneous modification of a particular secondary index would not be possible. However, this limitation may be offset by the following considerations:-

1. Locking is potentially a very expensive operation. Successively to lock several multi-level indices in which each index may have to be locked successively from the root is an extremely painful operation.
2. Contention for secondary indices will be less than for the primary index. (Note that this does not mean that secondary indices will be used less, only that the contention at any point in time will be less).
3. All modifications to secondary indices are known 'a priori'. No locking across user calls is required.

g) Subprogram Interfaces and Calling Sequences

"In general the following conventions will be followed for all internal components of RMS-11:

1. All calls are made via the \$CALL\$ macro which generates a JSR PC.
2. Unless noted otherwise by a routine ALL registers are preserved across a call.
3. R5 is used to point to the USER's input structure (FAB, RAB etc.).
4. R4 is used to point to the primary SYSTEM input structure (IFAB, IRAB etc.).
5. R3 is used to point to the system impure area.
6. The C-bit is used as a success or fail flag.
7. No condition code bits with the exception of the C-bit will be used to convey input, output or status information between routines.
8. All inputs, outputs and status information is conveyed either in structures (FAB, IRAB etc.) or in Registers."

h) String Collating Sequence

"The internal representation of ASCII characters on PDP-11 systems is 7-bit ASCII. The string compare routine of RMS-11, however, will perform a full 8-bit unsigned compare per character. RMS-11 will not perform any 'clear bit 7' code on input or output operations. This will allow the support of user binary byte strings, the KANA character set used in Japan, and in the future 8-bit ASCII when defined, without RMS-11 modifications since the true collating sequence is lowest character = 0, highest character = 255."

i) Unusual Conditions Treatment Philosophy

"The unusual condition treatment philosophy can be summarized as follows:

1. If the operation can proceed, but at a decrease in process/access time efficiency then continue, but note if possible.
2. If the operation can not proceed then abort the operation. Give the user the appropriate error status. Take necessary actions to insure the integrity of the file, and of the system data structures.

3. The insuring of file integrity does not mean that the file is in the most efficient form for access, but does mean that the file can be accessed.
4. The basic method of insuring file integrity during output operations is to perform the operation so that if the system fails or a hard or logical error is detected the following conditions are met in their order by priority:
  - a. No loss of data currently in the file.
  - b. New or Updated record can be found via the Primary access path.
  - c. Efficiency of random access to the new or updated record via the primary access path.
  - d. The efficiency of the secondary access paths.
  - e. The correctness of the secondary access paths."

)

.

.

)

)

)

.

.

)

Additional Functions for the FMS-11 Form Driver  
 Legare Coleman  
 Philip Morris  
 Richmond, Virginia

Ed. - This letter is on the Fall 1981 RSX/IAS SIG tape on account [347,101] and is available for general use. No gaurentees are made to its working, try it for yourself.

This is a correction file to the FMS-11 form driver (FDV Version 01.00) to add additional functionality. The RSX utility SLP should be used to apply this correction to the FDV.MAC source file.

ADDED FUNCTIONALITY:

One word is added to the FMS impure data area. The first byte of this word contains the first line number on the screen on which the current form is displayed. The second word contains the last line number on the screen used by the current form. This word may be addressed by using the symbol I\$FLLL as an offset into the impure area.

Program function keys PF3 and PF4 are changed to function as user controlled program function keys if the cursor is not positioned in a scrolled area when they are depressed. The keys function exactly like the numeric key pad when it is in program function key mode. PF3 returns an "R" and PF4 returns an "S" to the user program if the keys are depressed while the cursor is not in a scrolled area. If the cursor is in a scrolled area when PF3 or PF4 is depressed, the keys function exactly as described in the FMS-11 documentation.

Three new function codes are added to form driver calls.

FC\$GVA this call may be used by the user program to obtain the current setting of the video attributes of any field on the current form.

FC\$SVA this call may be used by the user program to dynamically set new video attributes for any field on the current form.

CAUTION: 1- the original video attributes are not saved and must be restored by the user program if desired.

2- if the video attributes of an indexed field are changed, only the single field specified by the index value specified in the call is changed on the screen. however, if cntrl w is depressed while any field in an array has its attributes changed, all fields in the array are repainted with the attributes specified in the last call which affected a field in the array.

FC\$NFI This call may be used to find the I.D. (name) of the first field on a form, or the I.D. (name) of the field following the field specified in the call. This call is useful in determining the names or order of fields on a form.

The new function calls and resulting status codes are documented in detail in comments in the following lines, which should be put into their own file. This would then allow 'SLP' to do its job with the source code.

```
[7,47]FDV.MAC;2/AU:72./-BF=[30,10]FDV.MAC;1
-/FC$SLN:./,./; HLC001/
FC$GVA:./BLKW ;GET VIDEO ATTRIBUTES
FC$SVA:./BLKW ;SET VIDEO ATTRIBUTES
FC$NFI:./BLKW ;GET NEXT FIELD I.D.
-/I$DISP:./,./; HLC001/
I$FLLL:./BLKW ;FIRST & LAST LINE NUMBER
I$RES:./BLKW ;UNUSED RESERVED SPACE
-/DISPAT:./,./; HLC001/
-/FSCLN:./,./; HLC001/
.WORD FGVATT ;GET VIDEO ATTRIBUTES
.WORD FSVATT ;SET VIDEO ATTRIBUTES
.WORD FNFID ;GET NEXT FILE I.D.
-/INITFM:./,./; HLC001/
-/80$:./,./; HLC001/
MOV R1,I$FLLL(R5) ;SAVE FIRST & LAST LINE NUMBERS
-/FGETAL:./,./; HLC001/
-/20$:/+1,./; HLC001/
CMPB #'R,RO ;PF3 ?
BEQ 30$ ;IF EQ, YES - TREAT AS FT$KPD
CMPB #'S,RO ;PF4 ?
BEQ 30$ ;IF EQ, YES - TREAT AS FT$KPD
-/FGETA:./,./; HLC001/
-/#FT$KPD:/+1,./; HLC001/
CMPB #'R,RO ;PF3 ?
BEQ 20$ ;IF EQ, YES - TREAT AS FT$KPD
CMPB #'S,RO ;PF4 ?
BEQ 20$ ;IF EQ, YES - TREAT AS FT$KPD
-/INPUT:./,./; HLC001/
-/#ALTKEY:/+1,./; HLC001/
CMP R1,#EXSCUP ;PF3 ?
BEQ 50$ ;IF EQ, YES - TREAT AS FT$KPD
CMP R1,#EXSCDN ;PF4 ?
BEQ 50$ ;IF EQ, YES - TREAT AS FT$KPD
-/EXSCUP:/+2,./+1,./; HLC001/
TST I$LPTR(R5) ;LOWER RIGHT ?
BEQ ALTJMP ;IF EQ, YES - NOT A SCROLLED AREA
BIT #D1$SCR,D$ATT1(R4) ;IN A SCROLLED AREA ?
BEQ ALTJMP ;IF EQ, NO
-/.ENDC:./,./; HLC001/
ALTJMP: CLC
JMP ALTKEY
-/EXSCDN:/+2,./+1,./; HLC001/
TST I$LPTR(R5) ;LOWER RIGHT ?
BEQ ALTJMP ;IF EQ, YES - NOT A SCROLLED AREA
BIT #D1$SCR,D$ATT1(R4) ;IN A SCROLLED AREA ?
BEQ ALTJMP ;IF EQ, NO
-/5$:./,./; HLC001/
.IFF
BR ALTJMP
-/ESCSEQ:./,./; HLC001/
-/.DSABL:./,./; HLC001/
```

```

.PAGE
.SBTTL
; FUNCTION: FGVATT - GET VIDEO ATTRIBUTES
;           RETURN THE VIDEO ATTRIBUTES FOR THE SPECIFIED FIELD.
; INPUT:   ARG - POINTER TO ARGUMENT LIST.
;           REQ - POINTER TO REQUIRED ARGEMENT LIST.
;           NAM - POINTER TO 6-BYTE ASCII FIELD NAME.
;           NUM - FIELD INDEX.
; OUTPUT:  VAL - LOW 4 BITS CONTAIN THE CURRENT VIDEO ATTRIBUTES OF THE
;           SPECIFIED FIELD.
; STATUS:  FE$FLD - SPECIFIED FIELD DOES NOT EXIST.
;
FGVATT:
CALL    FNDFLD      ;LOCATE THE FIELD
MOV     D$VATT(R4),F$VAL(RO) ;MOVE ATTRIBUTES
BICB   #360,F$VAL(RO) ;ISOLATE LOW 4 BITS
RETURN ;RETURN TO CALLER
.PAGE
.SBTTL
; FUNCTION: FSVATT - SET VIDEO ATTRIBUTES
;           SET THE VIDEO ATTRIBUTES OF THE SPECIFIED FIELD AND REPAINT
;           THE FIELD WITH THE NEW ATTRIBUTES.
; INPUT:   ARG - POINTER TO ARGUMENT LIST
;           REQ - POINTER TO REQUIRED ARGUMENT LIST.
;           NAM - POINTER TO A 6-BYTE ASCII FIELD NAME.
;           NUM - FIELD INDEX
;           VAL - LOW 4 BITS CONTAIN THE NEW VIDEO ATTRIBUTES FOR THE
;           SPECIFIED FIELD.
; OUTPUT:  NONE
; STATUS:  FE$FLD - SPECIFIED FIELD DOES NOT EXIST.
;
FSVATT:
CALL    FNDFLD      ;LOCATE THE FIELD
BICB   #17,D$VATT(R4) ;CLEAR PREVIOUS ATTRIBUTES
BICB   #360,F$VAL(RO) ;ISOLATE NEW ATTRIBUTE BITS
BISB   F$VAL(RO),D$VATT(R4) ;SET NEW ATTRIBUTES
BIC    #IS$CLR,(R5)
CALL    PRTRSP      ;REPAINT FIELD
RETURN ;RETURN TO CALLER

```

```

.PAGE
.SBTTL
; FUNCTION: FNFID - GET NEXT FIELD I.D.
;           DETERMINE THE I.D. (FIELD NAME AND INDEX NUMBER) OF THE NEXT
;           FIELD AFTER THE FIELD SPECIFIED. IF THE FIELD NAME SPECIFIED
;           CONTAINS AN "*" IN THE FIRST POSITION, THE I.D. OF THE FIRST
;           FIELD IS RETURNED. IF NAM=0, THE I.D. OF THE CURRENT FIELD IS
;           RETURNED.
; INPUT:   ARG - POINTER TO ARGUMENT LIST.
;           REQ - POINTER TO REQUIRED ARGUMENT LIST.
;           NAM - POINTER TO A 6-BYTE ASCII FIELD NAME.
;           NUM - FIELD INDEX.
; OUTPUT:  NAM - POINTER TO 6-BYTE ASCII FIELD NAME OF THE NEXT FIELD ON
;           THE FORM AFTER THE ONE SPECIFIED.
;           NUM - FIELD INDEX OF THE NEXT FIELD.
;           LEN - FIELD LENGTH
;           VAL - FIELD TYPE FLAGS
; STATUS:  FE$FLD - SPECIFIED FIELD DOES NOT EXIST.
;           FE$NOF - NO FIELDS DEFINED FOR CURRENT FORM.
;           FE$IFN - NO NEXT OR CURRENT FIELD.
;
FNFID:
MOV     F$NAM(RO),R3 ;R3 = A(FIELD NAME)
BEQ    1000$         ;IF EQ, CURRENT FIELD
CMPB   #'*,(R3)     ;FIRST FIELD DESIRED ?
BNE    1100$         ;IF NE, NO - FIELD SPECIFIED
CALL   ANYFLD       ;CHECK FOR ANY FIELDS ON FORM
BR     1200$         ;GO TO COMMON CODE
1000$: MOV    I$LPTR(R5),R3 ;R3 = A(LNCL FOR CURRENT FIELD)
BEQ    1300$         ;IF EQ, NO CURRENT FIELD
BR     1200$         ;GO TO COMMON CODE
1100$: CALL   FNDFLD ;LOCATE THE SPECIFIED FIELD
CALL   NXTLC        ;FIND THE NEXT FIELD
BCS    1300$         ;IF CS, NO MORE FIELDS
1200$: MOV    L$FDES(R3),R4 ;R4 = A(FDES ENTRY)
MOV    R4,F$NAM(RO) ;F$NAM = A(FDES)
ADD    #D$FID,F$NAM(RO);F$NAM = A(FIELD NAME)
MOV    D$RLEN(R4),F$LEN(RO) ;F$LEN = FIELD LENGTH
MOV    D$ATT2(R4),F$VAL(RO) ;F$VAL = FILED TYPE FLAGS
MOV    (R3),R3 ;R3 = LINE & COLUMN
SUB    @D$LNCL(R4),R3 ;R3 = FIELD INDEX - 1
INC    R3 ;R3 = FIELD INDEX
MOV    R3,F$NUM(RO) ;F$NUM = FIELD INDEX
RETURN ;RETURN TO CALLER
1300$: JMP    ILGFNC ;RETURN ERROR

```



FMS-11 Offset Definitions

Legare Coleman  
Philip Morris  
Richmond, Virginia

Distributed at Fall '81 DECUS Symposium

Impure Area Header

Name	Offset	Length	Description
I\$IMPA	0	0	BEGINNING OF IMPURE DATA AREA
I\$FDST	0	2	STATUS FLAGS IF FDV IN CONTROL, IMPURE AREA SIZE
I\$FMST	0	2	IF PROGRAM IN CONTROL:
			NAME VALUE MEANING
			IS\$ALT 1 OPERATOR HAS MODIFIED
			DATA IN CURRENT FIELD
			IS\$HLP 2 FIELD LEVEL HELP
			MESSAGE DISPLAYED
			IS\$INS 4 CURRENT FIELD IS BEING
			PROCESSED IN CHARACTER
			INSERT MODE
			IS\$\$SGN 10 N/A
			IS\$DEC 20 N/A
			IS\$MED 400 FORM IS MEDIA RESIDENT
			IS\$DSP 1000 INITIAL FORM DISPLAY
			IN PROGRESS
			IS\$HFM 2000 HELP FORM IS CURRENTLY
			DISPLAYED
			IS\$SCR 4000 CURRENTLY IN SCROLLED
			AREA
			IS\$CLR 10000 ?
			IS\$LST 20000 MESSAGE DISPLAYED ON
			LAST LINE
			IS\$ERR 40000 PROGRAM ERROR
			IS\$NMS 100000 NO MESSAGE IF ERROR
I\$ILEN	2	2	IMPURE AREA SIZE IN BYTES
I\$ALLC	4	2	IMPURE AREA REQUIRED BY CURRENT FORM
I\$SVST	6	2	STATUS FLAGS (I\$FDST) IF PROGRAM IN CONTROL
I\$STAT	10	2	A (USER'S TWO WORD STATUS BLOCK)
I\$LLIN	12	2	LINE NUMBER OF LAST LINE ON SCREEN NOT IN MESSAGE
			AREA
I\$LCOL	14	2	COLUMN NUMBER OF LAST COLUMN ON SCREEN
I\$VATT	16	2	CURRENT VIDEO ATRIBUTES
I\$LVID	20	1	LAST LINE VIDEO ATTRIBUTES
I\$ADVO	21	1	ZERO IF ADVANCE VIDEO OPTION NOT PRESENT, DEVICE
			ATTRIB. BYTE IF ADVANCE VIDEO IS PRESENT
I\$STKP	22	2	CALLER'S STACK POINTER
I\$BEND	24	2	A (LAST BYTE OF TERMINAL I/O BUFFER)
I\$BPTR	26	2	A (TERMINAL I/O BUFFER)
I\$FCHN	30	2	FORM LIBRARY LUN
I\$FIXD	32	0	LENGTH OF FIXED AREA
I\$BADR	32	2	A (FORM LIBRARY BUFFER)
I\$BSIZ	34	2	FORM LIBRARY BUFFER SIZE

Name	Offset	Length	Description
I\$FORM	36	6	NAME OF BASE LEVEL FORM
I\$CFRM	44	6	NAME OF FORM CURRENTLY DISPLAYED
I\$HLPF	52	6	NAME OF NEXT LEVEL HELP FORM
I\$NFLD	60	2	NUMBER OF FIELDS ON FORM (EACH ENTRY IN AN
			INDEXED FIELD IS COUNTED AS A FIELD)
I\$LNCL	62	2	A (LINE/COLUMN TABLE - LNCL)
I\$FDES	64	2	A (FIELD DESCRIPTOR TABLE)
I\$NDAT	66	2	A (NAMED DATA BUFFER)
I\$RESP	70	2	A (RESPONSE BUFFER)
I\$CURP	72	1	CURRENT LINE NUMBER
I\$CURC	73	1	CURRENT COLUMN NUMBER
I\$ROFF	74	2	OFFSET INTO FIELD BUFFRE IN RESPONSE BUFFER
I\$FOFF	76	2	OFFSET INTO PICTURE FOR THIS FIELD
I\$LPTR	100	2	A (LNCL TABLE ENTRY FOR CURRENT FIELD)
I\$LINE	102	2	OFFSET FROM FIRST LINE ON SCREEN TO FIRST LINE OF
			FORM
I\$FBLK	104	2	FORM LIBRARY CURRENT RELATIVE BLOCK NUMBER
I\$NBYT	106	2	NUMBER OF BYTES REMAINING IN FORM LIBRARY BUFFER
I\$FSIZ	110	2	NUMBER OF BYTES IN FORM
I\$FADR	112	2	A (NEXT BYTE IN FORM LIBRARY BUFFER)
I\$DLN1	114	6	DUMMY LNCT ENTRY TO PROCESS LEFT PART OF FIXED
			DEC FIELD
I\$FXD1	120	2	(LAST WORD OF I\$DLN1) A (DUMMY FDES FOR LEFT PART
			OF FIXED DEC FIELD)
I\$DLN2	122	6	DUMMY LNCT ENTRY TO PROCESS RIGHT PART OF FIXED
			DEC FIELD
I\$FXD2	126	2	(LAST WORD OF I\$DLN2) A (DUMMY FDES FOR RIGHT
			PART OF FIXED DEC FIELD)
I\$FDS1	130	20.	DUMMMY FDES FOR LEFT PART OF FIXED DEC FIELD
I\$FDS2	154	20.	DUMMMY FDES FOR RIGHT PART OF FIXED DEC FIELD
I\$PATN	200	2	SET TO 052525 TO INDICATE THAT THE IMPURE AREA
			HAS BEEN INITIALIZED
I\$DISP	202	2	?
I\$HLEN	204	0	LENGTH OF IMPURE AREA HEADER

Line/Column Table

Name	Offset	Length	Description
L\$LNCL	0	2	LINE AND COLUMN ON SCREEN WHICH CONTAINS THIS
			FIELD. BYTE 0 CONTAINS THE BINARY LINE NUMBER AND
			BYTE 1 CONTAINS THE BINARY COLUMN NUMBER
L\$RESP	2	2	ADDRESS OF AREA WHICH CONTAINS THE DATA CURRENTLY
			DISPLAYED IN THE FIELD. THIS DATA COULD BE THE
			FILL CHARACTER, DEFAULT VALUE, VALUE OUTPUT BY
			PROGRAM, OR VALUE INPUT BY OPERATOR
L\$FDES	4	2	ADDRESS OF FIELD DESCRIPTOR TABLE ENTRY WHICH
			DESCRIBES THIS FIELD
L\$CLSZ	6	0	SYMBOL WHICH DEFINES THE SIZE OF EACH ENTRY IN
			THIS TABLE

Field Description Table

Name	Offset	Length	Description
D\$ATT2	0	2	FIELD ATTRIBUTES FLAGS
			NAME VALUE MEANING
			D2\$VRT 20 VERTICAL TAB
			D2\$DEC 40 FIXED DECIMAL
			D2\$ZFL 100 ZERO FILL
			D2\$TAB 200 AUTO TAB
			D2\$DIS 400 DISPLAY ONLY
			D2\$RTJ 1000 RIGHT JUSTIFIED
			D2\$REQ 2000 RESPONSE REQUIRED
			D2\$FUL 4000 MUST FILL
			D2\$NEC 10000 NO ECHO
			D2\$SPO 20000 SUPERVISOR ONLY
D\$FID	2	6	ASCII FIELD NAME
D\$LNCL	10	2	ADDRESS OF ENTRY IN FIELD LOCATOR TABLE
D\$VATT	12	1	VIDEO ATTRIBUTES FLAGS
			NAME VALUE MEANING
			DV\$UND 1 UNDERLINE
			DV\$REV 2 REVERSE VIDEO
			DV\$BLD 4 BOLD
			DV\$BLK 10 BLINK
			DV\$DWD 20 DOUBLE WIDE
			DV\$DHW 40 ?
			DV\$GRA 100 GRAPHICS CHAR. SET
D\$CLRC	13	1	CLEAR CHARACTER
D\$ATT1			
D\$2ATT	14	2	FIELD ATTRIBUTES FLAGS
			NAME VALUE MEANING
			D1\$ARY 40 ARRAY
			D1\$SCR 100 SCROLLED AREA
			D1\$COM 200 ?
			D1\$SNM 400 SIGNED NUMERIC
			D1\$ALN 1000 ALPHA-NUMERIC
			D1\$ALP 2000 ALPHABETIC
			D1\$NUM 4000 NUMERIC
			D1\$MIX 10000 MIXED PICTURE
D\$RLEN	16	2	MAXIMUM RESPONSE LENGTH
D\$FXLN	20	1	LENGTH OF FIELD PICTURE IN BYTES
D\$PLEN	21	M	FIELD PICTURE
D\$PICT	21+M	1	LENGTH OF DEFAULT VALUE IN BYTES
	21+M+1	N	DEFAULT FIELD VALUE
	21+M+N+1	1	LENGTH OF HELP TEXT IN BYTES
	21+M+N+2	0	HELP TEXT

DBMS-10/20 Menu Responses:

Item 1: Provide an integrated Data Dictionary

This is under active investigation as part of a longer range strategy for information management on Large Systems. We are studying the nature of an Information Resource Directory facility which is oriented toward both productivity in automated systems and tracking of resources for the enterprise. We can make no commitments at this time, but will be informing customers as the strategy evolves.

In the shorter term, we see a need for a means of describing data, common to several system processes. Customers will probably see an evolving capability, starting with simple descriptions.

Item 2: Provide an inquiry language which is simpler and easier to use than IQL.

We are currently investigating changes to IQL which will make it easier to use in the data base environment. In the long term we are in the process of evaluating our overall information management strategy. Query language capabilities are clearly a central issue.

Item 3: Move the DBCS out of the user address space.

This item is under active investigation internally to define the scope of the work involved. We would like to do this one also, but it's not necessarily straight-forward.

Item 4: Provide a transaction processor similar to the Univac "TIP" product. Support fixed preamble with user responsibility for the remainder of the message.

We currently have no plans in this area for work directly on large systems. We believe that the correct solution to this problem for large systems customers involves dedicated resources for the transaction processor, with application systems capabilities either local or remote from the concentrator. Corresponding data base capabilities must also exist locally or remotely. We believe there will be a corporate solution to this problem. There is work, however, in the area of the COBOL-8x standard, which specifies a COMMUNICATIONS region. This facility includes message sub-queues between processes, and may solve some of the need in this space.

Item 5: Provide DBMS Page inventory information to improve store operation performance when searching for space for a record.

This is an item that was originally on the wish list for version 6. It's not difficult, but was voted (by customer poll) lower on the list of desired features than others and was consequently dropped. We are currently in the planning phases for the next release, and will include this as a requested item on the desired feature list. Obviously we can't make any promises until we have been through prioritization and scheduling.

Item 6: Allow store option which would inhibit connection to an automatic set.

We'll consider it. This is the first time we've heard of it, but it's a good suggestion. This is also a good time for suggestions. An alternative possibility is the use of data base procedures to conditionally store or remove from set memberships (or do virtually anything else).

Item 7. Speed up the processing of the invoke command.

This is also under consideration. We recognize the start-up costs involved for small operations on large sub-schemas, particularly for user-written, data base specific utility programs. We are looking at what is involved in a re-design of the BINDing operation. This may have to wait one more release, as we interested in a short next release.

## ADABAS-M, A "VALUE-BASED" DBMS FOR PDP-11 AND VAX COMPUTERS

Arthur C. Curtis  
and  
David Keller  
Software AG of North America  
Reston, Virginia

ABSTRACT

ADABAS-M is a full facility data base management system designed specifically for use with the Digital PDP-11 and VAX-11 series of computers. It will function under the RSX-11M, RSX-11M-PLUS, IAS and VAX/VMS operating systems. ADABAS-M is a "value-based" DBMS utilizing the relational data model (implemented via inverted lists) and incorporates a Data Dictionary to control and manipulate data bases. The ADASCRIP-T-M query language, a report writing facility, and a complete set of utility programs to interactively create and maintain data bases are included with the product.

Philosophy of ADABAS-M Implementation

The basic set of software development tools required to build applications from simplistic batch all the way up to sophisticated online MIS systems usually include several or all of the following:

- o data base management system
- o data dictionary
- o end-user query language
- o report writer
- o high-level applications programming language
- o screen/forms manager

Data base management (i.e. the data and its access paths) is the basis of organizational and application information systems. Virtually every online or batched transaction in such systems will involve one or more data base accesses, either (1) to retrieve from existing data for display or validation, or (2) to add new data to existing data bases. The data base is therefore the foundation of user applications. All other software tools can be evaluated in terms of getting data in and out of the data base: ease of use, reliability, flexibility and performance.

The basis for a data base manager is its data model. This model determines how data is to be accessed, both logically and physically. The three classic models are usually labelled the hierarchical, the network and the inverted (including relational). One of the problems associated with traditional data base implementations has been the selection of a data base management system that was suited for an organization's initial data base but not flexible in modelling evolving information requirements. For example, an organization may have initially selected a hierarchical DBMS that provided an easy implementation of the bill of material application. As the organization evolved however, new information requirements necessitated contorting the hierarchical model to support a network or perhaps even a "flat file" or inverted approach. Today, state-of-the-art data base management software provides for all three data models. How then does one determine which of these packages best meets data modelling requirements now and in the future? The key words are "now" and "in the future". Built-in flexibility is required. Beneath the three logical data models is a physical data structure implementation. This implementation is what will provide an organization either with flexibility or built-in obsolesence.

Given today's hardware and software, there exist only two traditional ways of implementing data structures. These two architectures are pointers and values (relations). "Pointer-based" systems make use of physically embedded pointers to effect relationships between data records while "value-based" systems use the actual data values in the records to do so. This simple distinction should not be underestimated. Remember, selection of data base models on which to build applications will ultimately depend upon the flexibility of the data structures supporting the desired data relationships.

In a pointer-based DEMS, both programmers and end-users ultimately end up navigating record-by-record through the data base in order to get information. Such systems are also called "designer" systems because of the burden of knowledge about the the data base design that is forced on users in order to be able to access data. It is required that the system designers spend considerable effort in designing the data base because once it is implemented there is little flexibility for changing the hard-wired pointers which effect the data relationships (and hence the access paths).

"Value-based" data base managers provide for flexible growth. This has been reflected in the data base research which addresses the problem of accomodating growth and complexity in the number of applications, in addition to human engineering for both data processing and end-users. The relational or value-based models provide flexibility and ease of use. Value-based data bases are easy to design and conceptualize because all users, including systems analysts and programmers, deal with "flat" files or two-dimensional tables. Because there are no chained pointers, such systems are easier to implement. Such data bases allow for non-procedural or non-navigational access to data. Instead of accessing record-by-record, as in a pointer-based DEMS, a user or programmer deals with collections of records that have selected values, leaving any navigation to the DEMS.

Maintenance is easier in a value-based system because the data access paths are independent of the physical records. Records are accessed by reference to their data values rather than by the existence of pre-determined physically linked pointers. Thus the addition or deletion of new access paths does not physically affect the data base records. Because the data structures are independent of program logic, changes that must be made to applications are minimized through the use of non-navigational data access. Reliability of the data base is enhanced and the data base is easier to recover due to the absence of physical pointers which may be corrupted if the operating system, machine or data base manager fail when pointer maintenance is taking place. In a value-based system, there can be no broken chain pointers because there are no chains or pointers. One of the major benefits of a value-based system is that files are logically interdependent (by means of inherent values) but physically independent (by lack of embedded pointers). Files may grow, new files and relationships added, and files may be distributed across a network without requiring physical structural changes to those files or related files.

All of these characteristics of a value-based DEMS contribute to:

- o simplicity in data base design
- o easier programming
- o faster transaction processing
- o ease of application and file maintenance
- o modular data base growth
- o ease and speed of recovery

ADABAS-M is a highly sophisticated, value-based DEMS specifically designed for Digital VAX and PDP-11 computers. It is an efficient implementation of proven DEMS techniques and is written in Macro assembler for maximum efficiency and performance.

#### An Overview of ADABAS-M For VAX and PDP-11 Environments

ADABAS-M consists of a Nucleus, a set of Data Dictionary and general purpose Data Base Administrator (DBA) utilities, a data base query (ADASCRIP-T-M), a report facility and an application program "CALL" interface routine.

#### The Nucleus

The Nucleus is a reentrant task (or VMS detached process) which is multi-tasked and multi-threaded. That is, it can be used simultaneously against multiple data base files by multiple users, where each user may be using the query/report facility, or a batch or interactive application program. In addition, the data base administrator (DBA) can use the general DBA utilities while the data base is active.

User tasks are processed within the ADABAS-M Nucleus by priority. Like an operating system, the task with the highest priority and with all required resources available, is processed. When a task becomes blocked, (e.g. waiting on I/O to complete) the next highest priority task is processed. Within a priority class, a round robin scheduler insures equitable Nucleus processing time. Note that this facility allows the DBA to give query users higher priority than batch jobs.

The Nucleus is responsible for data base security and integrity, retrieval and formatting of data, and editing and storing of data. All data base activity is routed through the Nucleus.



The Utilities

Interactive DBA utilities enable the DBA to define, control and maintain data bases and the run-time environment for the Nucleus. Using the utilities, the DBA may:

- o interactively create and maintain file, record and logical record view (USERVIEW) definitions within the Data Dictionary (analogous to the CODASYL schema and sub-schema definition)
- o load new files at high-speed or merge records from sequential files with existing files
- o open, close and lock files for access/update
- o control access to data base files via file passwords manipulation
- o save, restore and regenerate a data base to recover from disk and/or soft system failures
- o obtain Data Dictionary information about any or all data base files
- o prioritize and otherwise affect the job mix within the Nucleus

ADASCRIP-T-M

ADASCRIP-T-M is an adhoc query facility which uses English-like language to enables users to retrieve, display, update and delete data held in an ADABAS-M data base. ADASCRIP-T-M will be discussed further in the 'Data Base Commands' section.

Report Writer

ADABAS-M includes a full-feature report writer that is easy to use. The report writer includes many powerful features, such as: loop control, control breaks, sorting, computation, edit masks, and built-in functions including MIN, MAX, TOTAL, AVERAGE, SUM, and COUNT.

The CALL Interface

ADALNK, a data base interface routine, enables user-written programs to call ADABAS-M and request execution of ADABAS-M data base commands. Although somewhat similar to the CODASYL data manipulation language (DML) commands, ADABAS-M's command set is non-navigational, with the commands being higher-level ("value-based"), and as such are less procedural and easier to use. User programs can be written in any language supporting the RSX/IAS or VAX/VMS calling convention, such as Pascal, C, COBOL, FORTRAN, BASIC-PLUS-2, etc. This interface will be discussed further in the 'Data Base Commands' section.

System FilesSYSDIC

The System Dictionary (SYSDIC) is created when ADABAS-M is generated. On initialization, it only contains information about itself (which in turn is an ADABAS-M data base file). As user data base files are subsequently added to the data base, their File, Data and Userview definitions (analogous to the CODASYL schema and sub-schema concept) are added to this Data Dictionary file.

The information contained in SYSDIC is used to determine the structure and status of user data base files. In addition, SYSDIC contains the Control Dictionary which, at start-up time, defines the ADABAS-M run-time Nucleus and environment parameters for the current session or execution.

Only those parts of the SYSDIC Data Dictionary in use, are in memory at any one time. All system and user file information is "aged" in memory using a least-recently-used (LRU) algorithm.

The implementation of ADABAS-M uses an integrated data dictionary. The data base manager (Nucleus, etc.) could not function without the Data Dictionary information. It is an active Data Dictionary also in its efforts to minimize memory requirements, because only those dictionary definitions which are in active use are read into memory

SYSWRK

The System Work File (SYSWRK) is controlled by the Nucleus and is used as a temporary scratch file and buffer area. At any one time, it may contain information about the status of various data base users (threads), collections of bit strings representing intermediate results from users' data base commands, and DBA utility line editor data.

SYSLOG

The System Log File (SYSLOG) is a continuously reused sequential disk file containing information necessary to enable a "warm" restart to recover data bases after the various types of system failure. SYSLOG contains "before" and "after-images" of all update activity for user and system files. It also contains 'clean-points' which indicate that the data base is logically correct, and therefore restartable, at that point.

SYSDMP

The System Dump File (SYSDMP) is used to capture a snapshot of the system at the time of data base failure or if the DBA uses the SNAP utility. This dump provides technical support personnel with the necessary Nucleus internal information to aid in remote problem diagnosis and resolution.

User Data Base FilesDefinitions

Data Base - A data base is collection of data base files required by an application or group of programs.

File - A user data base file is a collection of related records which are to be treated as a unit. A data base may contain one or more files. These files need not be related to each other.

Record - An ADABAS-M file consists of a number of data records. Each record contains related information about an entity represented in the file. A file may contain one or more record types.

Internal Sequence Number (ISN) - As a record is added to an ADABAS-M file, an ISN is assigned to it. An ISN is a binary number that uniquely identifies that record within a file. ISNs are used by ADABAS-M during search operations and are analagous to the CODASYL data base "key" concept.

Field - A record consists of a number of "named" fields. Each field is given a unique field name when the record type is defined to the Data Dictionary. Fields in a personnel file might be NAME, AGE, SALARY, etc. Although the field names would be the same for each file record, the field values could differ from one record occurrence to another.

Descriptor (or key) - Fields in a data base file can be designated as "descriptors". Descriptors enable complex but flexible search requests to be issued. ADABAS-M automatically maintains inverted lists for descriptors and uses these lists to process records in random or sequential order based on descriptor value, and to process collections of records based on Boolean combinations and/or ranges of descriptor values.

ADABAS-M does not access any data records while finding a collection based on Boolean criteria. This process is effected entirely by forming bit map masks of ISN lists obtained from the inverted lists. The benefit is having retrieval statements operate at almost memory speeds against the inverted lists to identify those records meeting a user's selection criteria.

Repeating Group - A group of fields that can occur multiple times within a record is called a repeating group. Repeating groups are either single-field or multi-field. That is, they may have one, or more than one field per group occurrence (or member). See figure 1 for an example of repeating groups.

The concept of repeating groups is a very powerful feature which several other relational DBMS models do not directly support. As a direct result, these DBMS models cause an explosion of flat files or relations which could have very easily been stored as repeating groups within a record. The indirect result of lack of repeating groups is an explosion in redundancy necessary for these relational model implementations to effect relationships across tables.

Any field in a repeating group may be specified as a descriptor and many repeating groups may be defined for a record. Repeating group members defined as descriptors may be manipulated by the DML commands 'First', 'Last', 'Next' and 'Previous' member.

Uerview - When processing data from a file, the user or programmer normally will only be concerned with a subset of the fields in each record. This subset is called a Uerview (or subschema). A Uerview is created by the DBA and stored in the Data Dictionary. In addition to the name of the Uerview, this definition also specifies the fields to be used, their order, length and format. ADABAS-M requires that a Uerview be used to access any fields of records in one of its files. At run-time, ADABAS-M maps the Uerview's logical record onto that of the corresponding physical record (run-time binding as opposed to compilation-time binding).

Uerviews enable "data independence". That is, changing the physical order of fields in a record does not affect existing programs. If a new field is added to record, those programs that do not have the field defined as part of their Uerview(s) are also not affected. No recompilation, re-linking and subsequent debugging of existing user programs is thus ever required for such data base changes.

Uerviews may also be used to effect forms and screen management, as they may be designed to be displayed, and can contain text messages. Non-printable ASCII characters can also be embedded in Uerviews. These may be used to format the display using terminal advanced video features, cursor or forms control, etc. Escape sequences for both DEC and non-DEC terminals can be used as necessary. The benefit of such capability is to have formatted information rather than just raw data displayed immediately on a user's screen.

Phonetics - The use of phonetics for data retrieval or selection reduces the number of errors caused by spelling variations. This is accomplished by storing the phoneticized information in a compressed form. Any string field may be defined as 'Phonetic'. The ADABAS-M phonetics routine implements the highly regarded New York State Identification and Intelligence System (NYSIIS) phonetics algorithm and is especially accurate in phonetic matching of Hispanic surnames.

Maximum flexibility is provided by such features as:

- o multi-data base support
- o multiple concurrent retrieval and update users per data base
- o 100+ files per data base
- o up to 16 million records per data base file
- o up to 255 fields per data base record
- o up to 128 bytes per field
- o data types of ASCII numeric, ASCII and binary string, dates, RAD50, floating point and integer
- o up to 32 descriptor keys per record
- o multi-volume file support; e.g. a data base file may be comprised of up to 8 Files-11 files spread across multiple packs, even of diverse device types

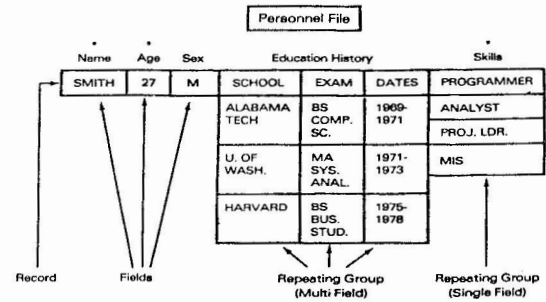


Fig. 1 A user file record layout

• = Fields Designated as Descriptors

File Structure

Each user data base file in an ADABAS-M data base consists of an Associator and a Data Storage Area. Figure 2 illustrates the structure of an ADABAS-M file.

The Associator - This contains the access structures for the file. For each descriptor field in the file, the Associator contains an inverted list to map each key value to the list of ISNs (internal IDs) of the records in which that value occurs. An ISN is mapped to the corresponding physical record address by the Nucleus via use of that file's Address Converter, which contains one entry per record in the file. The Address Converter is the only place the Associator has to be altered in case a record is moved in Data Storage, no matter how many descriptors are defined for the file. ADABAS-M allows any data item or field to have an associated index. However, the storage overhead which usually accompanies inverted lists is reduced by the use of forward and trailing key compression. These densely compacted keys minimize access times, and save valuable disk and memory storage buffer space.

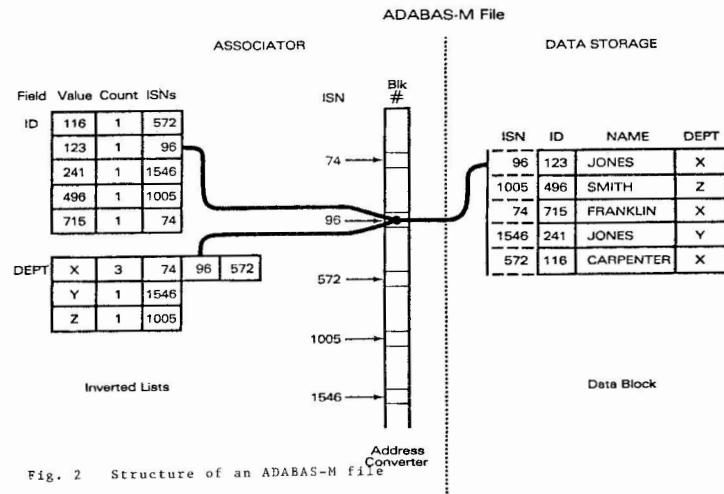


Fig. 2 Structure of an ADABAS-M file

The Data Storage Area - The Data Storage Area contains the data records for the file. Records are stored as variable length compressed strings. A number of records may occupy a single block (or sector), or a single record may span a number of blocks. Field values are represented in the smallest number of bytes (words for RAD50 fields) required to maintain their accuracy. No space is reserved for empty, or null, fields and repeating group members.

The Data Storage Area consists of one or more extents, where an extent is a Files-11 file. When a user file is created, the DBA defines to the Data Dictionary its device and number of blocks. After some time, this extent may fill. The DBA can then add further extents, possibly on other disk devices. Up to 8 data extents are allowed, and each could be on a different disk device. The benefit of such an approach is that a data base file need not be unloaded in order to increase the number of records in that file, and extra disk storage need not be allocated until actually required.

#### Utilities

Utility programs exist for the DBA to define, modify and maintain an ADABAS-M data base. In this section, those utilities used for File, Data and Userview definition will be discussed. Other utilities will be discussed in the 'Inter-task Communications' and 'Backup and Recovery' sections. All ADABAS-M utilities are fully interactive with HELP facilities at each level of operation.

#### The FILE Utility

The FILE utility enables the the characteristics of a file to be defined, modified or listed through the integrated Data Dictionary. They include the file name, its optional access and update passwords, file and Data Dictionary status, number of records and descriptors, and Data Storage Area extent locations and sizes.

#### The DATA Utility

The DATA utility allows the DBA to define and update the characteristics of the data to be held in a file, including the name of each field, its type, format, and descriptor characteristics. A descriptor may be defined to have data or null values, or both, to be included in the inverted list in its file's Associator. The DATA utility has a line editor which may be used to define/modify data definitions. Alternately, a data definition may be loaded from a disk file which was created by a text editor such as EDT or SOS.

Data types include string, integer, ASCII numeric with trailing or overpunch sign, binary, floating point, RAD-50 and date. Date fields allow the user to specify a date in MMDDYY (or MM-DD-YY) format. ADABAS-M stores this internally as YYMMDD so that searches may be done on date ranges.

#### The VIEW Utility

The VIEW utility enables Userviews to be defined. A Userview describes the fields, their order, format and size to be entered or retrieved by a user. Userviews may include repeating groups. The VIEW utility has a line editor which may be used to define/modify Userviews. Alternately, a view definition may be input from a disk file.

A number of Userviews can be defined for a file, and any number of users may share a Userview concurrently. The information contained in a Userview definition is referenced by the ADABAS-M Nucleus at run-time when a program requests data from a record. Userview information is read into memory from the Data Dictionary with the more active Userviews kept in memory to minimize reads from the Data Dictionary.

#### The LOAD and RLOAD Utilities

LOAD is used to populate an empty file, while RLOAD is used to merge records into an existing file. Both require a load Userview which defines the layout of the sequential input file records being loaded. Not all input file record fields need be loaded, and their order can be changed as necessary before a load is performed. LOAD is a high-speed loader.

#### Inter-task Communication and Data Transfer

A "thread" is a communications link between a user task and the ADABAS-M Nucleus. This link is established when the user task (or ADASCRIP-T-M) issues an OPEN THREAD command. A 1500 byte subroutine called ADALNK is linked with applications programs to enable the program to communicate with the Nucleus. The ADALNK interface routine does not require the dedicated use of any of the application program's APRs on a PDP-11. A user program passes parameters indicating the requested data base action by CALLING ADABAS. This transfers control to the ADALNK interface routine which in turn communicates with the Nucleus. RSX/IAS SEND/RECEIVE directives (VMS temporary mailboxes) are used for the initial handshake, then all data transfer is done at high speed via a pseudo-device driver (for a software virtual device IT!) This approximately 1 KB device driver is memory resident and controls all inter-task communications between the ADABAS-M Nucleus and its users.

Data Base Commands

Commands exist to open and close files, find record collections, to randomly position by descriptor or ISN value, sequentially process in descriptor or ISN sequence, produce histograms (frequency counts of records by key value) and to display, update, and delete records and repeating group members.

The abbreviated command syntax is the same for both ADASCRIP-T-M users and application programs. A more English-like syntax with 'noise' words is also available to the ADASCRIP-T-M user.

Security and Protection

ADABAS-M is designed to automatically protect the data base from system software and hardware failures. Privacy is secured through the prevention of unintentional or unauthorized access to data within the data base. ADABAS-M uses file passwords to prevent unauthorized data base access. Access and update passwords may be changed online by the DBA. The DBA may selectively make one or more data base files available only for reading or updating for the duration of a particular data base session.

A user can only access data fields that have been defined in his/her Userview(s). Userview creation and modification is controlled by the DBA.

ADABAS-M automatically checks and validates field forms using Data Dictionary information and safeguards against field truncation.

Records must be 'held' by a user before they can be updated or deleted. This is done using the Hold Record (HR) command or the hold modifier on another command, such as DRH, which displays and holds a record. The record "lock" is removed when the Release Record (RR) command is issued or another record from the file is retrieved.

Backup and Recovery

Recovery/restart capabilities are crucial to the success of most data base applications from simplistic batch systems all the way up to sophisticated online systems. DEMS-based systems, like online systems, must therefore be designed with failure in mind. Consequences of hardware or software failure are highly visible and magnified under DEMS, in part due to the integrated nature of data and file sharing under DEMS implementations. Recovery/restart is not a feature to be added to DEMS applications after they have been implemented. Systems must be designed to take advantage of whatever facilities are available to guarantee both the logical and the physical integrity of the data base, and hence the success or failure of the application system.

Journal File

The System Log File (SYSLOG) contains before- and after-images of all records added or updated. Record deletions are also recorded. When an update (add, delete or modify) is done, it may not be immediately written to the data base file on disk, in order to reduce disk I/O. Periodically, ADABAS-M flushes its buffers and writes a 'clean-point' to the log file. This marks a place where the data base is logically and physically intact, and can be restarted.

Auto-restart

When ADABAS-M is started-up, the system log file is checked to see if ADABAS-M was shut down gracefully. If not (because of hardware or software failure), transactions that took place after the last 'clean-point' are "backed-out" and reapplied so that the index and data structure integrity is not compromised.

System Backup and Recovery

When the system log file fills, the ARCHIVE utility is used to back it up to tape or disk so that the log file can be reused. This can be done on-line. The complete data base environment can be captured periodically using the SAVE utility. During system SAVE, the data base is available in "read-only" mode. To recover a data base, the SAVED system is restored using the RESTORE utility and the ARCHIVED log images are applied in a forward direction using the REGENERATE utility.

## OPERATING SYSTEM INTERACTION

The shared, reentrant ADABAS-M Nucleus is a non-privileged task that runs in 28 - 38 KW on a PDP-11 or a recommended VMS working set of 128 pages minimum on a VAX. It is therefore subject to the same operating system conditions as any other task or process. The inter-task driver requires 1000 bytes; ADASCRIP-T-M uses a minimum of 24KB. User programs have the 1500 byte ADALNK module as part of their task (process) space.

All system and user file extents appear as sequential FILES-11 structure files and can be manipulated by operating system utilities such as PIP or VMS COPY. If these extents are defined to reside on logical devices, they may also be redirected in the event of a disk drive failure.

The DBA "console" terminal is available for other work while ADABAS-M is running.

### Summary

Data base management (the data and its access paths) is the foundation of application systems. ADABAS-M is a complete DEMS supportive of hierarchical, network and inverted/relational data models. Through "value-based" architecture, ADABAS-M provides a flexible, easy-to-use, and efficient technique for the development of user-oriented application systems. With an extensive array of data base maintenance utilities, an application program interface, Data Dictionary, query/report writer, security, backup and recovery facilities, ADABAS-M supports the data management and processing requirements of a diverse group of VAX/PDP-11 users.



4303 Santa Barbara  
Columbia, Missouri 65201  
July 11, 1982


TECO Special Interest Group  
c/o D.E.C.U.S.  
One Iron Way, MR2-3/E55  
Marlboro, Mass. 01752

Dear Sirs:

I am a zealous user of TECO. My hardware consists mainly of LSI-11/23's running RT-11, occasionally RSX-11. I would like to re-link my TECO to run as a foreground job under RT-11, however, I discovered that my TECO object library is incomplete. I am writing to learn if the TECO object library is available from you.

As a macro programmer, I would be very interested in reviewing the routines from which TECO is built. I understand that the source is not distributed usually so not to encourage variations. I would be amenable to a gentleman's agreement not to distribute or alter code. On the other hand, I only really desire a listing to review. Please respond as to which if any of these are acceptable to you. Thank you.

Sincerely,



Don Delwood, M.D.

# MICROSOUND COMPANY

● BOX 1153  
● LOOMIS, CALIFORNIA 95650

28 July 1982

DECUS

ATTN: TECOEXPERT  
One Iron Way, MRO2/E55  
Marlboro, MA 01752

I would appreciate your forwarding this to one of the  
TECO experts. Return this with comments if you wish.

Problem is with V36. I am unable to load VTEDIT.TEC as  
I do with V35:

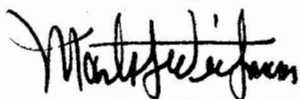
After loading TECO, I try EIVTEDIT\$\$ (as in 35)  
I get \*MEM Memory Overflow.

System is LSI-11 and I have both V3 and V4 of RT-11 with  
same results on either one. Another user suffers same  
problem. Possibly my files are incorrect. I have:

	V35	V36
TECO.SAV	51	50
VTEDIT.TEC	23	32

V35 is so slow I still use V28. I know there is KED and  
I use it too but I can hardly just discard an old friend.

Sincerely,



Martin J. Weitzman  
TECOPERSON



*National Museum of Natural History · Smithsonian Institution*

WASHINGTON, D.C. 20560 · TEL. 202- 357-2938

March 25, 1982

Mr. Paul D. Clayton  
Republic Management Systems  
One Neshaminy Interplex, Suite 306  
Trevose, Pa. 19047

Dear Mr. Clayton:

I just received the February issue of the DMS SIG Newsletter, and was delighted to find a section on word processing. NMNH has been using a WS 200-48 system for about a year with great success, and we are anxious to learn from the experiences of other users. We are new to the DEC family, and need guidance on how to go about discovering other WPS users, perhaps through DECUS. Do you have time to talk with me about this by phone in the near future?

DEC seems to think we are using the system in creative and unusual ways. They may be just flattering us because we may be a big customer in the future, but if we do have some good ideas that others can use, we would be happy to contribute them to the newsletter.

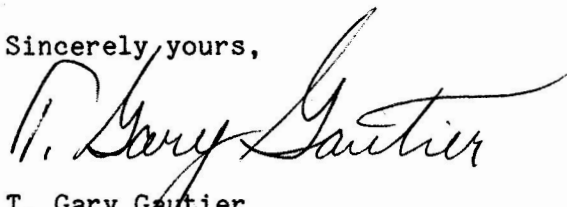
For Example, would you be interested in writeup on a method for automatically numbering list processing records? My method requires connecting two host ports (eg. HOST2 and HOST3) with a cable. I then send a document to HST2 as if printing, and receive it on HOST3 using CX set to HD. The 'printout' with its page numbers is thereby transferred to a document. To consecutively number records in a list, I insert print commands and page breaks so each record starts a new page and the page number 'prints' at the top of the page in the first field of the record. With some creative thinking, the same basic procedure could be used for numbering lines and paragraphs, which Eisenberg (newsletter article) said could only be done by brute force. The method has its drawbacks (eg. underlining gets messed up), but may be of some use to other WPS users. I hope DEC makes the method obsolete by providing a math package for WS200's, but I'm not holding my breath.

On another matter, we wish to send WPS documents to non-DEC photocomposers to publish scientific papers, but are hindered because WPS won't send underlining (extensive in our manuscripts) through CX. To get around this in a recent test, we flagged each underlined word with '@' characters, but that required extra typing and wouldn't be a good long-range solution. Our problem would easily be solved if WPS would let us do search and replace operations on the invisible

- 2 -

underlining commands, but it won't. Do you know of any solution? Could OS-8 replace the invisible commands with visible ASCII characters, for example? We can't predict what brand of photocomposer will be used next because of government printing regulations, so we pretty much have to depend on using CX and visible ASCII characters. DEC just keeps running us around in circles on this problem.

Sincerely yours,



T. Gary Gautier  
Chief, ADP Program

## Fall '82 DECUS Symposium

### AIRPORT TRANSPORTATION

Luxe Livery Service, Inc. has offered to provide our attendees Airport transportation to and from the Los Angeles Airport at a discounted fee. The total cost for our attendees will be fifteen dollars (\$15.00) each way excluding driver gratuity.

You can arrange to be met at the airport outside your airline by a uniformed driver who knows your name and where you are going. The driver will load your baggage and take you directly to your hotel.

#### FROM LOS ANGELES AIRPORT

After claiming your baggage please proceed to the center island, across from the baggage claim area. The van service is only permitted to park to load passengers. If you do not see Luxe Livery's green van in your area, watch for them in traffic - they may be circling the airport. To be recognized by the driver detach and wave the green portion of your transpass as he approaches. In the event you do not make contact with your driver within 10 minutes, please phone (800) 422-4267 or (714) 558-1413 and ask to speak with the dispatcher.

#### TO LOS ANGELES AIRPORT

Call (714) 558-1413 after 7:00 p.m. the evening before service to confirm your reservation and pick-up time. Meet the van at the hotel's main entrance. Be prompt! The drivers are not permitted to wait more than 10 minutes. Remember, Airport check-in lines are usually quite long.

#### SPECIAL NOTES:

1. Acquire a transpass from the DECUS office. A special mailing will be done for all preregistered attendees and DECUS Leadership. Anyone else wishing to receive the transpass must call the DECUS Office at (617) 467-4875.
2. Each attendee is required to have their own transpass. Therefore, when requesting transpasses from the DECUS Office please state the correct number you need.
3. Phone Luxe Livery Service and make an advanced reservation. The phone number for outside California is (800) 854-8171 and within California (800) 422-4267.
4. The transpass must be presented to the driver at the time of transfer to be eligible for the discounted rate.
5. The transpass has no value in itself, cannot be bought or sold, and must be used in conjunction with CASH ONLY at the time of transfer.
6. Please pass this information on to your SIG/LUG members so they can also benefit from this offering.



**DECUS**

DIGITAL EQUIPMENT COMPUTER USERS SOCIETY  
ONE IRON WAY, MRO2-1/C11  
MARLBORO, MASSACHUSETTS 01752

BULK RATE  
U.S. POSTAGE  
PAID  
PERMIT NO. 129  
NORTHBORO, MA  
01532

ASSOCIATE



**MOVING OR REPLACING A DELEGATE?**

Please notify us immediately to guarantee continuing receipt of DECUS literature. Allow up to six weeks for change to take effect.

- ( ) Change of Address
- ( ) Delegate Replacement

DECUS Membership No.: \_\_\_\_\_

Name: \_\_\_\_\_

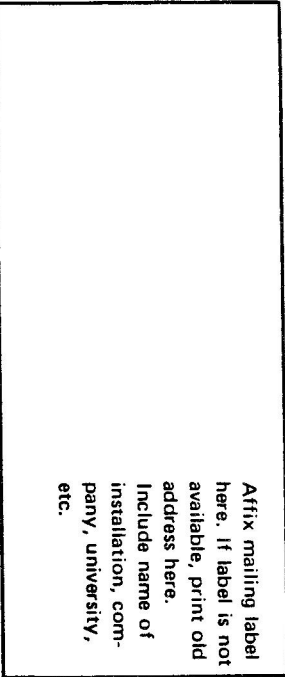
Company: \_\_\_\_\_

Address: \_\_\_\_\_  
\_\_\_\_\_

State/Country: \_\_\_\_\_

Zip/Postal Code: \_\_\_\_\_

Mail to: DECUS - ATT: Membership  
One Iron Way, MRO2-1/C11  
Marlboro, Massachusetts 01752 USA



Affix mailing label here. If label is not available, print old address here. Include name of installation, company, university, etc.