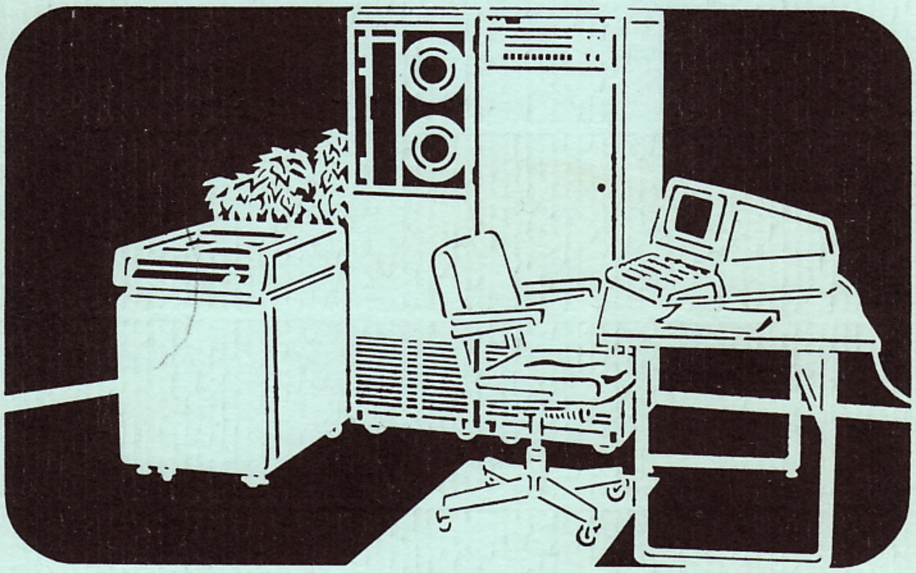
HP

general systems users group



# ABSTRACTS & PROCEEDINGS - 78

# PROCEEDINGS OF THE HP GENERAL SYSTEMS USERS GROUP SEVENTH INTERNATIONAL MEETING

October 30 thru November 3, 1978
Denver, Colorado

## TABLE OF CONTENTS

I. Introduction and Organization

II. Paper / Presentation Indices

A. By Session Code / Topic

B. By Author Name

C. By Author Company

III. Papers / Presentations

A. Computer Applications ............ Series "A"

B. Data Management .................. Series "B"

C. Machine Utilization .............. Series "C"

D. Installation Management .......... Series "D"

E. Programming Languages ............ Series "E"

F. Data Communications .............. Series "F"

G. System Development ............... Series "G"

H. System Peripherals ............... Series "H"

I. Special 'Round Table' Sessions ... Series "I"

J. Keynote Address

IV. Session Reviews

V. Conference Attendee Name List

VI. Exhibitor Name List

# Introduction to the Proceedings

## WELCOME ! ! !

Welcome to the proceedings of the seventh international meeting of users of the HP3000 computer system. This meeting was the first in a new era for our organization. We had formalized our structure and incorporated as a nonprofit organization, the HP General Systems Users Group (HPGSUG). This is to reflect our changing needs and also to more fully address the needs of the users of the HP General Systems Division computer products family.

The HPGSUG is an organization of individuals with the common interest and goal of maximizing the benefit they can obtain from the use of HP General Systems Division computer products. These benefits are realized through the contributed efforts of the members themselves, specifically through the contributed program library and meetings which provide the opportunity to 'interface and educate'.

## MEETING

The Goal of the 1978 meeting was to provide such opportunities. The format of this meeting was established with the foremost objective being to provide a flexible framework for users to benefit through the collective user experience. The schedule was set to run five days and the sessions to be mirrored in each half of the week with the vendor exhibit and special sessions on Wednesday.

## PROCEEDINGS

That brings us to this document. One goal of the meeting committee was to be able to provide the attendees with the printed proceedings at the time of registration. The result was a 'preprint' of this volume. We believe that the proceedings of the meetings provide a valuable source of information. It was with this in mind that we undertook the post conference task of assembling these materials. We hope this document will provide you with a guide by which to extract the utmost from this meeting and provide a reference to you in the times to come.

The Papers / Presentation section of the proceedings is organized around the 'series' classification by major topic. There are nine of these major topic series which include:

|  |  |  |
|---|---|---|
| Series | A | Computer Applications |
| " | B | Data Management |
| " | C | Machine Utilization |
| " | D | Installation Management |
| " | E | Programming Languages |
| " | F | Data Communications |
| " | G | System Development |
| " | H | System Peripherals |
| " | I | Special 'Round Table' Sessions |

Within each series the papers are numbered sequentially. Pagination follows the same organization...

```
A-11.03
*  *  *
*  *  *** page number within this paper
*  *
*  *** paper within this series or topic classification
*
*** series id or major topic classification
```

Please note that not all sequence numbers were used and that there may be gaps. For example Series H has papers H01, H03, H05 and H07.

ACKNOWLEDGEMENTS

The planning and organization of this conference has been an immense task which, without the support, effort and committment of all involved, might not have come to fruition. The papers contained herein and the sessions presented at the meeting are the volunteer efforts of that special breed of person, willing to share what they have learned with those who seek to learn. To these people, A very special 'Thank You!', for without your efforts the best organized meeting would have no basis.

Thank you Hewlett Packard and specifically Ralph Manies and Lynn Gardner for your committment, encouragement and support above and beyond the call of duty, in helping us to pull this meeting together.

As anyone who participates in the execution of such a 'special assignment' soon realizes, the support provided by one's family and employer is crucial to the success with which the completion may be measured. Our sincere appreciation to all.

Respectfully,

The HP General Systems Users Group
7th International Meeting Committee

Joyce B. Pleasants
Aurora Public Schools
International Meeting Director

David Appel
Information Resources Ltd.
Publications

Chuck Green
Adams County Schools
Vendor Exhibits

Don Phelps
Colorado Department of Highways
Activities

Bob Rice
Adams County Schools
Registration

C.R. Van Ausdall
Commercial Office Products Company
Program

# Index to Papers by Session Code / Topic

## Series A: Computer Applications

Computer Aided Instruction on the HP3000......................A-01
Diane Christopherson and Beverly Sheperd
Western Wisconsin Academic Computing Consortium

Computer Assisted Residential Energy Audit....................A-02
Dr. Neal H. Prochnow and Marlys Nelson
University of Wisconsin - River Falls

Decision Support System/Containerboard Logistics Management..A-04
Peter DiGiammarino and Richard Schwartz
American Management Systems, Inc.

Corporate Modeling-Financial Planning Systems................A-06
J. Gewecke
United Computing Systems

IDIMS - HP3000 Based Digital Image Processing................A-10
Paul Polk
E.S.L., Inc, Sunnyvale, CA.

Econonmetric Modelling on an HP3000..........................A-11
John Eaton
London Business School

CWF/3000: Computer Assisted Instruction and Training........A-12
Harold J. Peters
Educational Software Products

Statistical Inquiry and Retrieval with IMAGE.................A-13
Martin D. Jewel
National Spinal Cord Injury Data Research Center

On-Line Marketing Information................................A-14
George J. Neibergs
E.S.B. Exide Industrial Batteries

Computerized Word Processing.................................A-15
Martin Gorfinkel
Los Altos Research Center

Dollar-Flow: Financial Planning on the HP3000...............A-16
Jack Damm
The Palo Alto Group

Typist Oriented Fully Integrated Wordprocessing System.......A-18
Daryl A. Frame and Roger M. Goldmann
Comarco, Inc.

Graphics in Business.........................................A-19
Paul Cooper
Hewlett-Packard - Tulsa, Oklahoma

Experiences with MFG/3000....................................A-20
Ivan M. Rosenberg and John Doyle
System Design Associates and Vetter Corporation

SPSS/HP-An Update............................................A-21
Marlys Nelson and Nicholas Elliott
University of Wisconsin-River Falls

## Series B: Data Management

Adopting a Transaction Processor....................................B-01
Robin C. Wheeler
Domtar Construction Materials, Montreal, Quebec

IMAGE Data Base Design and Performance Measurement...........B-02
Orland J. Larson
Hewlett-Packard General Systems Division

IMAGE's Coming of Age: Data-Base Transformations.............B-03
F. Alfredo Rego
Instituto de Informatica y Ciencias de Computacion

Faster with Fast KSAM.................................................B-04
Stephen M. Butler
Paradise Valley Hospital

Information Management: An Investment for the Future........B-08
David C. Dummer
D.C. Dummer & Associates Limited

Sigma-Generalized Information System...........................B-09
Marcos A. X. de Carvalho
Promon Engenharia S.A.

Data Dictionary/Directory Driven Clinical Data System........B-10
Eric S. Herbel
Hoechst-Roussel Pharmaceuticals, Inc.

HP VIEW/3000-A Source Data Entry System......................B-12
Jutta Kernke
Hewlett-Packard General Systems Division

Factory Data Collection.................................................B-13
Linda Siener
Hewlett-Packard Data Systems Division

ASK/3000-A Necessary Complement to IMAGE/QUERY...............B-14
Yann Cordelle
COGELOG

## Series C: Machine Utilization

MPE Object Code Formats-USL and Program Files................C-02
Matthew J. Balander
The B & B Computer Company

HP3000/Optimizing On-Line Programs..............................C-03
Robert M. Green
Robelle Consulting Ltd.

On-Line Tape Library....................................................C-07
Pete Fratus
Futura Systems, Inc.

System Performance Measurement and Optimization.............C-10
Jim Squires and Ed Splinter
Hewlett-Packard Fullerton and Los Angeles

Using Extra Data Segments: Safe and Efficient..................C-11
Rick Ehrhart
Hughes Aircraft Company-El Segundo

## Series D: Installation Management

Installation Design and Operation Considerations.............D-01
Bill Gates
Longs Drug Stores, Inc.

Extended Operating Environment for Applications Programs.....D-02
Richard A. Bergquist and Steven M. Cooper
American Management Systems, Inc.

Beacon/Guardian: Solutions in Search of Problems............D-12
Wayne E. Holt
Whitman College

## Series E: Programming Languages

The Changing World of COBOL..................................E-04
Greg Gloss
Hewlett-Packard General Systems Division

Tips on Converting IBM FORTRAN Programs to the HP3000........E-08
Gary Anderson and Deepak Sinha
McMaster University

Writing SPL Routines Which Are Callable from BASIC...........E-09
Warren Kuehner
Hewlett-Packard - Englewood, Colorado

DEC/3000-An Example of Special-Purpose Language Design.......E-10
Matthew J. Balander
The B & B Computer Company

## Series F: Data Communications

Cutting Communications Costs with Statistical Multiplexors...F-03
Roger L. Evans
Micom Systems, Inc. Chatsworth, CA.

## Series G: System Development

Programming for Survival....................................G-03
Gerald T. Wade
Hewlett-Packard - Englewood, Colorado

Software Quality Control....................................G-07
C. Edward McVaney
J. D. Edwards & Company

An Approach to On-Line Applications Management...............G-09
Dennis Dinan and Glenn Entis
Morgan Guaranty Trust Company of New York

A Guide to Systems Development..............................G-10
John M. Grillos
American Management Systems, Inc.

Decision Tables - An Effective Programming Tool..............G-11
Daniel f. Langenwalter
Integrated Business Systems, Inc.

Microprocessor Based Product Design on an HP3000.............G-12
Jack C. Armstrong
Los Altos Research Center

## Series H: System Peripherals

UT200 RJE Subsystem.........................................H-01
Donald Klett
Sangamon State University, Springfield, IL.

HP3000/IBM1403 Coupling Extends System Utilization...........H-03
Ray Lorenz
Spur Products Corporation

Real-Time Instrument Interface System for the HP3000.........H-05
Gordon R. Symonds
Environmental Health Centre, Ottawa, Ontario

Disk Subsystems Software Considerations......................H-07
R. Brechtlein
CALCOMP

## Series I: Special 'Round Tables'

Organizing a Local Group of Computer Users...................I-13
Douglas J. Mecham
Hughes Aircraft Company - Fullerton

## INDEX BY AUTHOR

| | |
|---|---|
| Anderson, G. | Tips on Converting IBM FORTRAN Programs to the HP3000.........E08 |
| Armstrong, J. C. | Microprocessor Based Product Design on an HP3000...............G12 |
| Balander, M. J. | MPE Object Code Formats-USL and Program Files.................C02 |
| Balander, M. J. | DEC/3000-An Example of Special-Purpose Language Design........E10 |
| Bergquist, R. A. | Extended Operating Environment for Applications Programs......D02 |
| Brechtlein, R. | Disk Subsystems Software Considerations.......................H07 |
| Butler, S. M. | Faster with Fast KSAM.........................................B04 |
| Carvalho, M. | Sigma-Generalized Information System..........................B09 |
| Christopherson, D. | Computer Aided Instruction on the HP3000......................A01 |
| Cooper, P. | Graphics in Business..........................................A19 |
| Cooper, S. M. | Extended Operating Environment for Applications Programs......D02 |
| Cordelle, Y. | ASK/3000-A Necessary Complement to IMAGE/QUERY................B14 |
| Damm, J. | Dollar-Flow: Financial Planning on the HP3000.................A16 |
| DiGiammarino, P. | Decision Support System/Containerboard Logistics Management...A04 |
| Dinan, D. | An Approach to On-Line Applications Management................G09 |
| Doyle, J. | Experiences with MFG/3000.....................................A20 |
| Dummer, D.C. | Information Management: An Investment for the Future.........B08 |
| Eaton, J. | Econonmetric Modelling on an HP3000...........................A11 |
| Ehrhart, R. | Using Extra Data Segments: Safe and Efficient................C11 |
| Elliott, N. | SPSS/HP-An Update.............................................A21 |
| Entis, G. | An Approach to On-Line Applications Management................G09 |
| Evans, R. L. | Cutting Communications Costs With Statistical Multiplexors....F03 |
| Frame, D. A. | Typist Oriented Fully Integrated Wordprocessing System........A18 |
| Fratus, P. | On-Line Tape Library..........................................C07 |
| Gates, W. | Installation Design and Operation Considerations..............D01 |
| Gewecke, J. | Corporate Modeling-Financial Planning Systems.................A06 |
| Gloss, G. | The Changing World of COBOL...................................E04 |
| Goldmann, R. M. | Typist Oriented Fully Integrated Wordprocessing System........A18 |
| Gorfinkel, M. | Computerized Word Processing..................................A15 |
| Green, R. M. | HP3000/Optimizing On-Line Programs............................C03 |
| Grillos, J. M. | A Guide to Systems Development................................G10 |
| Herbel, E. S. | Data Dictionary/Directory Driven Clinical Data System.........B10 |
| Holt, W. E. | Beacon/Guardian: Solutions in Search of Problems.............D12 |
| Jewel, M. D. | Statistical Inquiry and Retrieval with IMAGE..................A13 |
| Kernke, J. | HP VIEW/3000-A Source Data Entry System.......................B12 |
| Kuehner, W. | Writing SPL Routines Which Are Callable from BASIC............E09 |
| Langenwalter, D. F. | Decision Tables - An Effective Programming Tool...............G11 |
| Larson, O. J. | IMAGE Data Base Design and Performance Measurement............B02 |
| Lorenz, R. | HP3000/IBM1403 Coupling Extends System Utilization............H03 |
| McVaney, C. E. | Software Quality Control......................................G07 |
| Mecham, D. | Organizing a Local Group of Computer Users....................I13 |
| Neibergs, G. J. | On-Line Marketing Information.................................A14 |
| Nelson, M. | Computer Assisted Residential Energy Audit....................A02 |
| Nelson, M. | SPSS/HP-An Update.............................................A21 |
| Peters, H. J. | CWF/3000: Computer Assisted Instruction and Training.........A12 |
| Polk, P. | IDIMS - HP3000 Based Digital Image Processing.................A10 |
| Prochnow, Dr. N. H. | Computer Assisted Residential Energy Audit....................A02 |
| Rego, F. A. | IMAGE's Coming of Age: Data-Base Transformations..............B03 |
| Rosenberg, I. M. | Experiences with MFG/3000.....................................A20 |
| Schwartz, R. | Decision Support System/Containerboard Logistics Management...A04 |
| Sheperd, B. | Computer Aided Instruction on the HP3000......................A01 |
| Siener, L. | Factory Data Collection.......................................B13 |
| Sinha, D. | Tips on Converting IBM FORTRAN Programs to the HP3000.........E08 |
| Splinter, E. | System Performance Measurement and Optimization...............C10 |
| Squires, J. | System Performance Measurement and Optimization...............C10 |
| Symonds, G. R. | Real-Time Instrument Interface System for the HP3000..........H05 |
| Wade, G. T. | Programming for Survival......................................G03 |
| Wheeler, R. C. | Adopting a Transaction Processor..............................B01 |

Index by Author's Employer

| | | |
|---|---|---|
| American Management Systems, Inc. | Peter DiGiammarino and Richard Schwartz | A04 |
| American Management Systems, Inc. | Richard A. Bergquist and Steven M. Cooper | D02 |
| American Management Systems, Inc. | John M. Grillos | G10 |
| CALCOMP | R. Brechtlein | H07 |
| COGELOG | Yann Cordelle | B14 |
| Comarco, Inc. | Daryl A. Frame and Roger M. Goldmann | A18 |
| D.C. Dummer & Associates Limited | David C. Dummer | B08 |
| Domtar Construction Materials, Montreal, Quebec | Robin C. Wheeler | B01 |
| E.S.B. Exide Industrial Batteries | George J. Neibergs | A14 |
| E.S.L., Inc, Sunnyvale, CA. | Paul Polk | A10 |
| Educational Software Products | Harold J. Peters | A12 |
| Environmental Health Centre, Ottawa, Ontario | Gordon R. Symonds | H05 |
| Futura Systems, Inc. | Pete Fratus | C07 |
| Hewlett-Packard - Englewood, Colorado | Warren Kuehner | E09 |
| Hewlett-Packard - Englewood, Colorado | Gerald T. Wade | G03 |
| Hewlett-Packard - Tulsa, Oklahoma | Paul Cooper | A19 |
| Hewlett-Packard Data Systems Division | Linda Siener | B13 |
| Hewlett-Packard Fullerton and Los Angeles | Jim Squires and Ed Splinter | C10 |
| Hewlett-Packard General Systems Division | Orland J. Larson | B02 |
| Hewlett-Packard General Systems Division | Jutta Kernke | B12 |
| Hewlett-Packard General Systems Division | Greg Gloss | E04 |
| Hoechst-Roussel Pharmaceuticals, Inc. | Eric S. Herbel | B10 |
| Hughes Aircraft Company - Fullerton | Douglas J. Mecham | I13 |
| Hughes Aircraft Company-El Segundo | Rick Ehrhart | C11 |
| Instituto de Informatica y Ciencias de Computacion | F. Alfredo Rego | B03 |
| Integrated Business Systems, Inc. | Daniel f. Langenwalter | G11 |
| J. D. Edwards & Company | C. Edward McVaney | G07 |
| London Business School | John Eaton | A11 |
| Longs Drug Stores, Inc. | Bill Gates | D01 |
| Los Altos Research Center | Martin Gorfinkel | A15 |
| Los Altos Research Center | Jack C. Armstrong | G12 |
| McMaster University | Gary Anderson and Deepak Sinha | E08 |
| Micom Systems, Inc. Chatsworth, CA. | Roger L. Evans | F03 |
| Morgan Guaranty Trust Company of New York | Dennis Dinan and Glenn Entis | G09 |
| National Spinal Cord Injury Data Research Center | Martin D. Jewel | A13 |
| Paradise Valley Hospital | Stephen M. Butler | B04 |
| Promon Engenharia S.A. | Marcos A. X. de Carvalho | B09 |
| Robelle Consulting Ltd. | Robert M. Green | C03 |
| Sangamon State University, Springfield, IL. | Donald Klett | H01 |
| Spur Products Corporation | Ray Lorenz | H03 |
| System Design Associates and Vetter Corporation | Ivan M. Rosenberg and John Doyle | A20 |
| The B & B Computer Company | Matthew J. Balander | C02 |
| The B & B Computer Company | Matthew J. Balander | E10 |
| The Palo Alto Group | Jack Damm | A16 |
| United Computing Systems | J. Gewecke | A06 |
| University of Wisconsin - River Falls | Dr. Neal H. Prochnow and Marlys Nelson | A02 |
| University of Wisconsin-River Falls | Marlys Nelson and Nicholas Elliott | A21 |
| Western Wisconsin Academic Computing Consortium | Diane Christopherson and Beverly Sheperd | A01 |
| Whitman College | Wayne E. Holt | D12 |

## COMPUTER APPLICATIONS

Series "A"

# COMPUTER AIDED INSTRUCTION ON THE HP3000

Diane Christopherson
Western Wisconsin Academic Computing Consortium
University of Wisconsin-River Falls
River Falls, Wisconsin 54022

Beverly Shepherd
Western Wisconsin Academic Computing Consortium
University of Wisconsin-River Falls
River Falls, Wisconsin 54022

The original CAI package was delivered with the installation of our HP3000 cx machine in August of 1975. An attached note from the sales representative said it would be a trivial task to convert the package to run on our machine. After three years of trial, error and much frustration, CAI is now running on an HP3000 Series II.

A good Computer Aided Instruction package should have three main goals. First, it should be an instructional tool to be used by educators as well as a learning experience for the students interacting with it. Secondly, good "courseware" must be available to allow for optimal utilization. Lastly, and most importantly, it must be easy to use for a person unfamiliar with computers. Thus, the teacher generates text for students without concerning himself with programming details. We feel that this CAI package meets these goals.

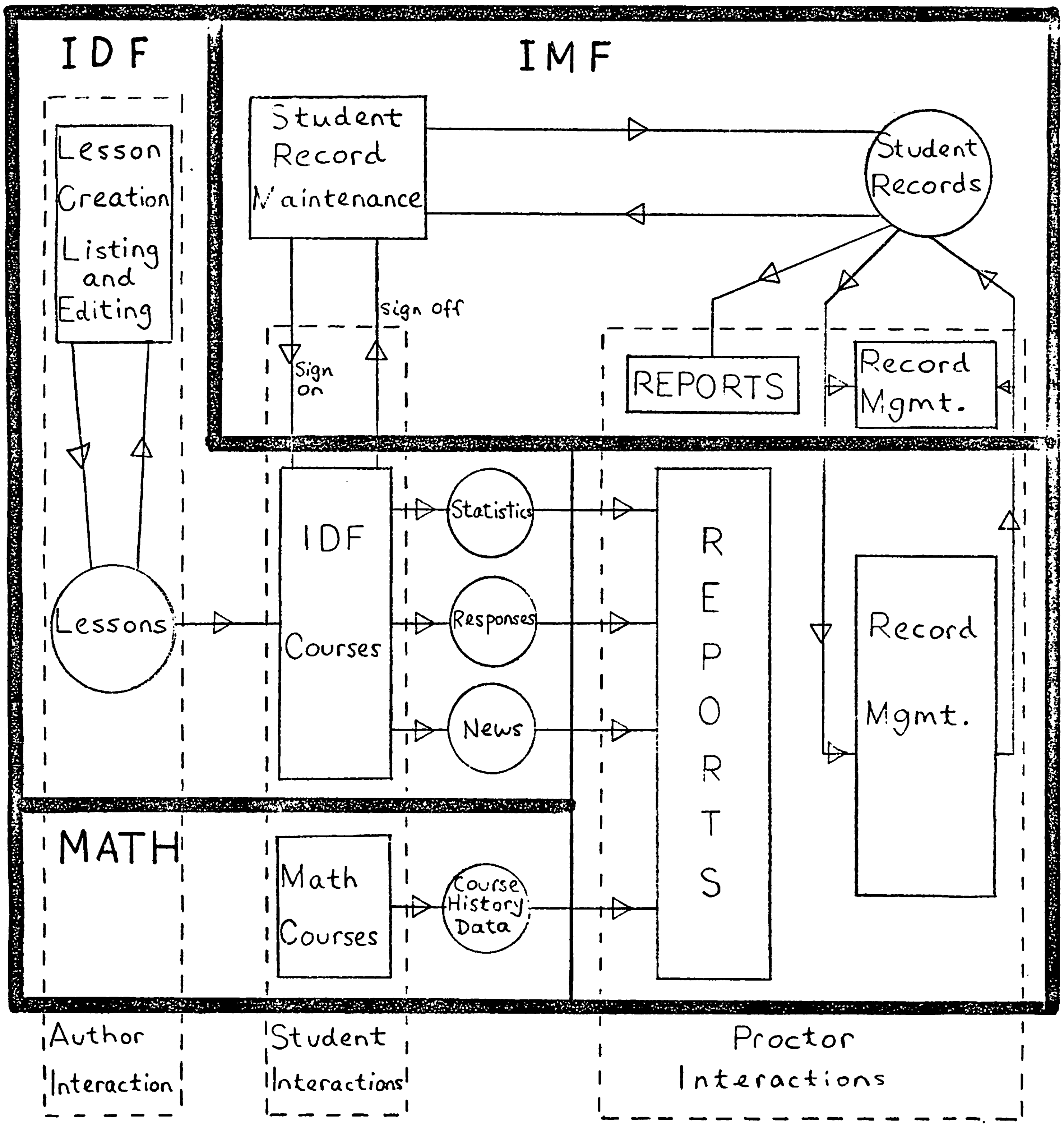
The CAI instructional package consists of three inter-related parts: IDF, IMF and Math Drill and Practice. A short description of each follows:

Lesson material is prepared through the Instructional Dialogue Facility (IDF) by the teacher who specifies the components of a course. These components include the questions to be asked with correct and incorrect anticipated answers, and the hints and clues that might be given -- together with the circumstances under which they are to be given.

The Instructional Management Facility (IMF) is a set of programs and files used to maintain Computer Assisted Instruction courses and make them available to the terminal user. The number of courses made available and the number of groups and students entered are limited only by the amount of storage available on a particular system.

The Mathematics Drill and Practice Program provides drill and practice in arithmetic fundamentals to supplement classroom instruction. Comprehensive records of all students using the program are maintained, permitting automatic individualized pacing of each student through the curriculum. The curriculum consists of six consecutive years of material -- with eac h year's material arranged in 24 consecutive blocks. A student's interaction with the course contains pretests, post-tests and review lessons in addition to the regular lesson. Depending on a student's progress, he/she is allowed to advance or review required material.

STRUCTURAL ORGANIZATION OF THE CAI PACKAGE



## IDF

The Instructional Dialogue Facility (IDF) enables educators to create and present computer assisted instruction (CAI) lessons without learning a programming language. The IDF guides them through each stage of the lesson creation, allowing the educators to concentrate on the instructional content of the lesson.

The program records student responses and compiles statistics to assist in evaluating and improving course material. Using these reports, the educator is able to immediately assess the effectiveness of the lesson that he has written and, if necessary, to modify the lesson using the editing facilities of the IDF. The educator can analyze the lesson either on a student-by-student basis (in which case he sees essentially a reconstruction of each student's interaction with the lesson) or on a group basis (where he sees statistical information of student performance).

Some other important features of IDF are:

1. The educator may allow the student to exit from an IDF lesson to use a simulated calculator capability or to use or write programs in BASIC. The student is then returned to his lesson exit point.

2. The educator and students are able to interact with IDF in English, German, French, Italian, Portugese, Spanish and Swahili.

3. Key word searches of two different types may be specified by the educator for answer processing: ordinary key word and context-sensitive (delimited) key word searches.

4. String or numerical answers are allowed; ranges may be specified for numerical answers, and there is an automatic provision for handling numerically-equivalent answers.

5. Students may request hints for problems if the educator has provided them.

6. Time limits and the number of times a student can try to answer a question may be specified.

7. A more sophisticated capability of IDF is in the use of as many as 12 counters. These counters can be incremented when a student's response matches a correct or wrong answer. The values of these counters can be used to provide branching to different IDF lessons.

A table showing how IDF guides an educator in lesson creation follows.

| PROMPT | | DEFINITION |
|---|---|---|
| Text | | Noninterrogative information displayed to the student before a question is asked. |
| Question | | A request for a student response. |
| Correct Answer Group ) | (1) | A collection of one or more answers the author considers correct. |
| Wrong Answer Group ) | (1) | A collection of one or more answers which the author regards as incorrect but which he suspects that a student may try. |
| or | | |
| Correct Answer Range ) | (1) | A range of numbers the author considers correct. If the student's answer is within this range, it is correct. |
| Wrong Answer Range ) | (1) | The author considers numbers in this range to be incorrect. |
| Reply To Correct (Or Wrong) Answer Group (Or Range) | (2) | The message the author wishes to have displayed to any student whose answer falls in a correct or wrong answer group or range. |
| Reply to Unexpected Answer | | The message the author wishes to have displayed to any student whose answer was not anticipated. |
| Failure Message | | The message the author wishes to have displayed to a student who has exhausted his permitted number of trials. |
| Hint | (3) | A hint to be given to students who request it. |

(1) Any number of groups or ranges is allowed.

(2) One reply is allowed for each group or range.

(3) Any number of hints are allowed.

IDF has the following editing capabilities:

1. Inserting, deleting, moving and appending sections;

2. Inserting, deleting or changing lines within a section;

3. Changing lesson branching;

4. Changing options.

## IMF

The Instructional Management Facility (IMF) is a set of programs and files used to present Computer Assisted Instruction, record student progress and provide a variety of reports for the teacher and the course author. The number of courses made available and the number of groups and students entered are limited only by the amount of storage available on a particular system.

Following is a list of the programs and a short description of what they do:

| Program | Description |
|---|---|
| ADMIN | Used to enter:<br>1. Each course in the IMF files;<br>2. School name, code word, group names;<br>3. Messages to be displayed to students;<br>4. Changes in the language. |
| PUPIL | Used to:<br>1. Enter students' names in the IMF files;<br>2. Enroll students in the CAI courses;<br>3. Change group assignments and session lengths;<br>4. Drop students from courses;<br>5. Obtain the Course Usage Report. |
| START | Asks students to sign in by typing their first name and identification number and the name of the course they wish to take. The course is then presented to the student. |
| OUTCOM | Used to:<br>1. Obtain IDF Statistics Reports;<br>2. View the IDF Response File;<br>3. Clear the Statistics and Response Files. |
| CNEWS | Used to:<br>1. List the News file;<br>2. Clear the News file. This file contains comments to authors from the IDSF. |

## HPMATH

HPMATH is a package that provides drill and practice in arithmetic fundamentals to supplement classroom instruction for elementary students. Lessons are drawn from a pool of  problem types (blocks) that cover such concepts as counting, addition, subtraction, multiplication, division, decimals, fractions and measurement. The students are assigned a starting year by their teacher; then they proceed through the course at their own skill level and pace.

The curriculum consists of six consecutive years of material -- with each year's material arranged in 24 consecutive concept blocks. Each block consists of a pretest, five drill lessons, and a post-test. The teacher enrolls a student at the beginning of any block in any year. If the student scores 100 per cent in the pretest of that block, then he will skip to the next block. Otherwise, he proceeds with the first lesson in the current block. His score on the pretest determines the level of difficulty at which he takes the first lesson. Within the curriculum block, the student's score on a main lesson determines the level of difficulty at which he takes the next main lesson. His score is determined by the number of correct answers he gives. The student's post-test score is recorded and used to determine whether he should review a curriculum block.

The system generates reports which keep the teacher informed of each student's progress. The following reports are available to the teacher:

1. Daily Report

This shows the following exceptional conditions:

i. Performance of a student below 60 per cent at level 1;
ii. A student skipping consecutive blocks;
iii. A student reaching a new concept block;
iv. Absences -- a student not taking the course.

2. Student Report

This contains post-test scores on blocks completed by the student, plus the number of times the block was reviewed.

3. Progress Report

This lists each student in the group, his exact position in the curriculum, and how many blocks he has completed to date.

4. Course History Report

This presents a statistical summary for the year (primarily of interest to school curriculum specialists) showing average and standard deviation of pre- and post-test scores.

5. Lesson Report

This provides the teacher with a sample lesson for any year, block and level.

6. <u>Usage Report</u>

This lists the students in alphabetical order with their identification numbers and the number of hours they spent using the math course.

7. <u>Roster Report</u>

This provides a permanent reference listing of students their identification numbers and their time limits.

IDF: A SAMPLE RUN

An example of a lesson created using the Instructional Dialogue Facility is shown below. The computer prompts the author for all of the structural elements in the natural order. For clarity, entries typed by the author are underlined.

```
SECTION # 1

OPTIONS? KEYWORD cr

OPTIONS? cr

TEXT:

   ? RECALL FROM OUR PREVIOUS LESSON THAT GEORGE II cr
   ? RULED GREAT BRITAIN FROM 1727 TO 1760, AND cr
   ? THAT DURING HIS REIGN BRITAIN HAD A SERIES cr
   ? OF WHIG PRIME MINISTERS. cr
   ? cr

QUESTION:

   ? WHO WAS THE FIRST PRIME MINISTER TO SERVE cr
   ? UNDER GEORGE II? cr
   ? cr

CORRECT ANSWER GROUP:

   ? WALPOLE cr
   ? cr

REPLY FOR THIS GROUP:

   ? THAT'S CORRECT; ROBERT WALPOLE WAS PRIME cr
   ? MINISTER UNDER GEORGE I FROM 1714 TO 1727; WHEN cr
   ? GEORGE II ASCENDED TO THE THRONE IN 1727, WALPOLE cr
   ? BECAME THE FIRST PRIME MINISTER UNDER HIS REIGN. cr
   ? cr

WRONG ANSWER GROUP # 1

   ? COMPTON cr
   ? cr

REPLY FOR THIS GROUP:

   ? NO—COMPTON WAS THE SECOND PRIME MINISTER TO cr
   ? SERVE UNDER GEORGE II; PLEASE TRY AGAIN. cr
   ? cr
```

**WRONG ANSWER GROUP # 2**

? PELHAM *cr*
? HOLLES *cr*
? CAVENDISH *cr*
? *cr*

**REPLY FOR THIS GROUP:**

? NO—HE DID IN FACT SERVE AS PRIME MINISTER UNDER *cr*
? GEORGE II, BUT WELL AFTER THE MAN I'M THINKING *cr*
? OF. PLEASE TRY AGAIN. *cr*
? *cr*

**WRONG ANSWER GROUP # 3**

? WALP *cr*
? *cr*

**REPLY FOR THIS GROUP:**

? I THINK YOU HAVE THE RIGHT ANSWER, BUT YOUR *cr*
? SPELLING IS WRONG; PLEASE TRY AGAIN. *cr*
? *cr*

**WRONG ANSWER GROUP # 4**

? *cr*

**REPLY TO UNEXPECTED ANSWER:**

? NOW STOP PLAYING AROUND; NO ONE BY THAT NAME *cr*
? EVER SERVED AS PRIME MINISTER IN GEORGE II'S *cr*
? TIME. PLEASE TRY AGAIN. *cr*
? *cr*

**FAILURE MESSAGE:**

? WELL, IT DOESN'T LOOK LIKE YOU'RE GOING TO GET *cr*
? THE RIGHT ANSWER ON THIS ONE, SO I'LL TELL YOU. *cr*
? THE MAN I'M THINKING OF WAS ROBERT WALPOLE; HE *cr*
? SERVED AS PRIME MINISTER UNDER GEORGE I FROM *cr*
? 1714 TO 1727, AND WHEN GEORGE II ASCENDED TO THE *cr*
? THRONE IN 1727, WALPOLE BECAME THE FIRST PRIME *cr*
? MINISTER UNDER HIS REIGN. *cr*
? *cr*

**HINT # 1**

? THE MAN I'M THINKING OF ALSO SERVED AS PRIME *cr*
? MINISTER UNDER GEORGE I. NOW TRY AGAIN. *cr*
? *cr*

## IMF: A SAMPLE RUN

An example of using the Instructional Management Facility appears below. It shows how to enter a school, group, course and student names into the IMF files.

```
ADMIN
CODE?MMMMMM
COMMAND?//HELP
TYPE CODE, SCHOOL, GROUP, COURSE, MESSAGE, RESET, LANGUAGE, OR PUPIL.
COMMAND?SCHOOL
  SCHOOL NAME?CAI HIGH
  SCHOOL NAME IS NOW CAI HIGH.

COMMAND?GROUP
  GROUP COMMAND?//HELP
  TYPE ENTER, CHANGE, REMOVE, OR LIST.
  GROUP COMMAND?ENTER
    NEW GROUP NAME?GROUP1
GROUP1 IS NOW ENTERED AS A GROUP.

    NEW GROUP NAME?GROUP2
GROUP2 IS NOW ENTERED AS A GROUP.

    NEW GROUP NAME?
  GROUP COMMAND?
COMMAND?COURSE
  COURSE COMMAND?//HELP
  TYPE ENTER, CHANGE, REMOVE, OR LIST.
  COURSE COMMAND?ENTER
    COURSE NAME?HISTORY
    CODE WORD?H1
    DOES IT HAVE A DEMO MODE?YES
    SPECIAL COURSE TYPE?IDF
    HISTORY IS NOW ENTERED AS A COURSE.

    COURSE NAME?MATH.PUB.CAI
    CODE WORD?MATH
    DOES IT HAVE A DEMO MODE?YES
    SPECIAL COURSE TYPE?MATH
    MATH.PUB.CAI IS NOW ENTERED AS A COURSE.

    COURSE NAME?
  COURSE COMMAND?
COMMAND?PUPIL
```

```
PUPIL

COMMAND?//HELP
TYPE ENTER, CHANGE, REMOVE, RESET, ADMIN, ENROLL, ALTER, DROP, OR LIST.
COMMAND?ENTER
  FIRST NAME?DIANE
  LAST NAME?CHRISTOPHERSON
  ENTERED WITH ID NUMBER 1000.

    COURSE NAME?MATH.PUB.CAI
    GROUP NAME?GROUP1
    STARTING YEAR?4
    ENROLLMENT COMPLETED.

  FIRST NAME?BEVERLY
  LAST NAME?SHEPHERD
  ENTERED WITH ID NUMBER 1001.

    COURSE NAME?HISTORY
    GROUP NAME?GROUP2
    ENROLLMENT COMPLETED.

  FIRST NAME?
COMMAND?LIST
  LIST BY ID, NAME, OR COURSE?ID
   PAGINATE?NO
    ID RANGE?1000,1001



-----


 CAI HIGH                  6 OCT. 1978         8:55 AM        PORT NO. 13

 ID LISTING                PAGE 1

                                   PREFERRED  TYPING  ENROLLED  USAGE
   ID         NAME                 LANGUAGE   MULT.   COURSE    (HRS)

  1000   CHRISTOPHERSON, DIANE    ENG         1      MATH.PUB.CAI    .0
  1001   SHEPHERD, BEVERLY        ENG         1      HISTORY    .0
 -----



    ID RANGE?
  LIST BY ID, NAME, OR COURSE?
COMMAND?//STOP
```

## HPMATH:  A SAMPLE RUN

```
>START
Upper case only?YES



----------




Please type your ID number and first name!  1000 DIANE

Is your last name CHRISTOPHERSON?YES

6 October 1978                    9:01                          Port 13




HELLO DIANE.  WE HOPE YOU ENJOY TODAY'S PROBLEMS.

M 6061

********** HERE WE GO !!!!! **********

 90
+ 5
---
 95
 16
+ 2
---
 18


  18
+ 10
----
  28


  69
+ 20
----
  89


 25
- 2
---
 23


 36
- 2
---
 34


  39
- 20
----
   8
WRONG, TRY AGAIN
  39
- 20
----
  19


  28
- 15
----
  13



 ENTER   <  OR  =  OR  >


6 + 1  <  3 + 5
```

```
8 + 6  >  1 + 3

  2^2 =  4

  6^2 =  36

 42
X 1
---
 42



   -----
3 / 2 4
    2 4   8
   ----
      0

3 X 8 = 24

LESSON OVER.  YOU ANSWERED 26 OUT OF 27 QUESTIONS CORRECTLY.
GOODBYE DIANE.




----------




Please type your ID number and first name:  //STOP
>EXIT
```

# COMPUTER ASSISTED RESIDENTIAL ENERGY AUDIT

DR. NEAL H. PROCHNOW
PHYSICS DEPARTMENT
UNIVERSITY OF WISCONSIN-RIVER FALLS
RIVER FALLS, WI 54022

MARLYS NELSON
WESTERN WISCONSIN ACADEMIC COMPUTING CONSORTIUM
UNIVERSITY OF WISCONSIN-RIVER FALLS
RIVER FALLS, WI 54022

The 1973 oil embargo focused this Nation's attention on its energy resources and precipitated increased costs for energy. The increased costs for energy resulted in monthly fuel bills for home owners which were twice that of previous years. A study in 1974 by the Department of Health Education and Welfare in cooperation with the Federal Energy Administration indicated that those hardest hit by the rising prices of home heating fuel were the poor who lived in single-family homes in the colder parts of the country and heated their homes with fuel oil. As a result the Community Services Administration (CSA) initiated in 1974 an energy conservation program via weatherization of homes. This program was delivered to low-income home owners by local community Action Agencies (CAA). It became apparent as a function of time that this program would be more effective if technical assistance could be provided to the weatherization crews, education to the home owners and management capabilities to the administrators of the program. As a result CSA provided funds for the development of a computer assisted energy audit management program. The program described in this paper is a spin-off of the program developed for CSA. The authors wish to thank Miriam Charnow at CSA National and Carl Saueressig at West CAP for their assistance and the personnel at West CAP for their cooperation in the development of the program.

The energy conservation awareness created by the 1973 oil embargo has somewhat dissipated. There are no long lines at service stations and there appears to be ample fuel for heating homes. The severe winter of 1976-77 is long forgotten and to some extent the glamour of solar energy and other alternate energy sources has attracted the general publics' interest instead of something as mundane as insulating a home. However, energy conservation must still remain as one of the most important objectives of any energy program for this country. The problem is to keep the concept of energy conservation in focus. Energy conservation requires a conscious effort and it seems as though the majority of adults react to it only via a catastrophe or a series of catastrophies; OPEC embargo, severe 1976-77 winter, etc. This is a difficult and traumatic way to maintain energy conservation awareness. Perhaps a more rationale way is to build the awareness in from the very beginning.

The United States has in place an elaborate education system and this system can be used to create an energy ethic. However, this mechanism still presents some problems. The energy problem is a social-cultural problem as well as a technical problem. Most teachers do not have the necessary information base to effectively build an energy ethic into their students. An even more important consideration is that most teachers are so overburdened with the existing curriculum that there may be little or no room for energy to be added. This is further compounded by the back to the basic demands currently being placed upon school curriculums. The time element also becomes a factor. It is perhaps most rationale to begin at the beginning, kindergarten, and create a process such that the ultimate result at grade 12 would be an energy conscious responsive individual. However, the developmental time and implementation time is too long for immediate results and may be too long to be sustained. Subsequently, there may be more cycles of energy catastrophe, reaction, etc. With all these factors in mind, it seemed apparent that if something was going to be done in the energy conservation area via students, it would have to compensate for all the factors indicated above. The interactive audit is a mechanism to do just that.

The audit is designed for use by students, grades 7-12. However, it can be used with college students and adults. Students using the audit are required to perform a series of activities. These activities can be coordinated very easily with the existing curriculum or can be done totally independent of the curriculum. The first activity requires the student to perform a structural assessment of his home with respect to energy efficiency. This assessment requires either a booklet or a data sheet as an explanatory device. West CAP is using the audit as a part of a youth energy conservation program and has developed a booklet and summary data sheet. Appendix A is a copy of the data sheet used by West CAP. The audit can be described most efficiently via a discussion of the questions involved in the assessment. Note that the fact that the student is required to perform an assessment of his home builds in an opportunity for parent-student-teacher interaction with respect to energy conservation.

The program can be used in a no instruction or instruction mode for data entry. In the instruction mode the entire question is printed by the computer, while in the no instruction mode only the line number and a question mark is printed. For all inputs the number entered is checked to see if it corresponds to reality where possible. Items two and three of Appendix A show sample data entries in each mode.

Question 1.: Date
As has been indicated some mechanism is needed to create awareness with respect to energy conservation. The mechanism in this case is the interactive capability of the audit. However, even the use of a computer is insufficient to maintain interest if the time to get results is too long. The time to enter the data and get a complete output ranges from 10 minutes for very few system users to 20 plus minutes if the system is saturated and the no instruction mode is used. Of this time,3 to 5 minutes are required to enter the data. In order to increase the flexibility the management portion of the program contains a file to temporarily store the answers needed to print the output. The status of this file is indicated when the user elects to use the audit program. If the file is not full (50 jobs currently allowed), the user can elect to enter the data and obtain his output up to 3 days later. If the file is full the user must obtain his output immediately after the data is entered. If this is impossible for him he need not initiate the program. The file is checked daily as a part of system back up and any data stored in excess of the three day limit is purged. The data is automatically purged after an output is obtained. The date is required to incorporate this feature.

Question 2.: YOU I.D. Number
The program is designed to store the energy conservation results of all the audits performed. In order to identify the audits performed by users on an individual basis a unique number can be assigned to identify their data. If this is not essential or desired the YOU I.D. number can be any number the user elects to enter.

Question 3.: House Number
The house number is computer assigned. This number is used to keep track of the total audits as well as allow the user to obtain his data at a later date.

Question 4.: Number of Home Occupants
The number of occupants is used to obtain an estimate of the hot water requirements.

Question 5.: Age of Home
This data is useful to State and Federal Agencies.

Question 6: Roof Condition
The condition of the roof is critical with respect to adding insulation to the attic.

Question 7.: Heating System Condition
The condition of the heating system is critical with respect to energy efficiency. Most if not all students do not have the skills to assess the condition of the heating system. However, the students can make the homeowner (their parents) aware of the fact that it should be checked. If the heating system has been serviced recently (2 to 3 years), it is considered to be in good condition; if not repairs may be needed. If it doesn't work it needs to be replaced.

Question 8.: Home Temperature-Day and Night
The average temperature of the home is needed in order to obtain the heatloss. The average temperature is computed assuming a weighted average of the day and night temperatures; 1/3 for night and 2/3 for day. The design temperature of home is assumed to be 70$^{O}$F and the heatloss is adjusted at 3% per degree. The temperatures input can be determined by asking the homeowner or by measurement.

Question 9.: Degree Day Zone
The heatloss is determined on an annual basis using degree days. In most states degree day data is available as a function of geographical area. For example, the State of Wisconsin is divided into 11 degree day zones. Usually the degree day data is available for a 65$^{O}$F base. Recent data from the National Bureau of Standards (NBS) indicates that a 55$^{O}$F base may be more appropriate. This is consistent with UW River Falls fuel bill analysis and hence the 65 base data has been adjusted to 55 base data. All degree day data used should be adjusted to the 55 base. In addition a 242 day, September 15 to May 15, heating season is assumed.

Question 10.: Type of Fuel and Fuel Cost
Nine types of fuel are allowed; gallons of #1 oil, gallons of #2 oil, kilowatt-hours of electricity, cubic feet of gas, therms of gas, gallons of LP, tons of coal, gallons of kerosene, and full cords of wood. The energy content of the fuels and efficiencies assumed are documented in the program and are consistent with latest NBS data. The fuel cost can be input or if it is unknown an average value is assigned by the computer. The average values used for the cost of fuel as well as all other standards must be determined regionally and periodically updated by systems staff. The rationale on the design is to place responsibility for parameters on a systems individual rather than on a sub routine built for all regions for

all time. That is, decisions on prices, standards, etc. are best made by an informed individual rather than by a complex subroutine designed to guess the future.

Questions 11. to 14.: House Structure
The program has been field tested for 1,2 and 3 story homes. The data entered for these questions is used in the heatloss calculations as well as to provide checks on reality. For example, the wall area is calculated from the data of questions 12. to 14. and is used to check the data entered in questions 27. and 28. to ensure a reasonable value for the wall area. In addition, the first floor area is used to check the ceiling area and floor area for the attic and basement heatloss calculations.

Questions 15. to 17. and 20. and 21.: Basement Heatloss
An explanation of the methodology of how the basement heatloss is calculated is beyond the scope of this paper. What is needed is the type of basement and associated dimensions such as the floor area (from question 12.), perimeter, amount exposed above ground and composition of the floor with respect to R-value. The R-value of a material is a necessary concept and either the user must have a working knowledge of it or obtain it by reading accompanying material. It is a simple concept and poses no problem if a small amount of time is devoted to it. The program is most accurate for homes with basements followed by homes with crawl spaces and then walkout basement homes. The answers obtained for trailer homes skirted and unskirted are very crude. This is because there is very little data available to check the methodology for trailers.

Questions 18. and 26.: Infiltration-Foundation and Walls.
Infiltration is an important consideration in heatloss. What is required is to assess the extent of air leakage into or out of the home. This is usually done by rating the structural components 1,2 or 3; that is good, mediocre, or bad. The resulting heatloss is then computed by using infiltration functions for the various components. In this program infiltration functions have been derived for the foundation, crack length, the wall crack length and the window and door crack lengths. The wall crack length refers to the crack length associated with the window and door frames where they join the wall siding. The rationale is that the crack lengths to be considered must correspond to those places that leak and are caulked, weatherstripped or sealed in some fashion. A major source of infiltration is associated with opening and closing windows and doors and/or leaving them open. There is no way to account for this in this model and this loss is not related to the structure but to the occupants behavior.

Questions 22. and 23.: Ceiling Area and R-Value
The program allows for two types of ceiling areas. The rationale is that a reasonably large number of homes have an addition which

has a different ceiling R-Value. In all cases the R-Values entered are for structural components only and R-Values for air films, etc. are assigned internally in the program.

Questions 19., 24. and 25.: Windows and Doors
The windows and doors of a home can represent a large heatloss. What is required is the direction, area, an infiltration judgement and whether or not there is a storm. The direction is not critical because it is used only to identify the window or door. The degree day concept contains the heat gain or loss due to sunshine. The window and door areas are computed from length and width measurements. The criteria for judging the infiltration are included in the data summary sheet. The window or door either has a good tight fitting storm or it is considered not to have a storm. An allowance is made for windows of the same size, direction and condition. The infiltration functions have been designed for double hung windows and are approximate for other types of windows. Question 25. requires the user to indicate whether or not the windows are double hung. This information is used to check the validity of the infiltration functions.

Questions 27. and 28.: Wall Area and R-Value
As in the ceiling case, two wall areas are allowed to account for the fact that a large number of homes may have a part of the home with insulated walls and a part of the home with uninsulated walls. The area to be entered is the area associated with the living space. The program checks this entered area against the area computed from the earlier structural data. The R-values entered are for the wall materials only.

As can be seen from the brief description of the questions, the assessment activity can be used to enhance the existing curriculums in both the mathematics and science areas. In addition, energy concepts such as R-values can be added with a minimum of time.

The second activity of the audit consists of entering the data either via the instruction or no instruction mode. This can be an enhancing activity for a class which does not have usual access to a terminal or it can be just another program execution. Item four of Appendix A contains an example of the output. The output is reasonably self-explanatory and need not be discussed with the exception of the opening paragraph. While the computer community has the utmost confidence in output, the general public does not necessarily share this confidence, particularly if energy is involved. Therefore, a check on reality is built in via the total fuel bill. That is, the homeowners faith in the results increases as the computer guesses more accurately his total heating bill. This mechanism provides a reality check on the output.

The third activity can consist of the interaction of the student and the homeowner via the audit output. This interaction can be kept minimal or expanded depending upon the use of the audit and the time and energy of the participants.

The audit has the potential to be very useful depending upon its implementation. The program including the management aspects is reasonably large. It consists of 15 segments written in HP-3000 Basic and 1 segment written in HP-3000 SPL. All segments are less than 8K words. The temporary output file with a 50 job limit requires 4500 sectors. The management file on school use has a 100 sector limit and the information from the audit which is stored permanently occupies 1/2 a sector per job. The correct time to run the program ranges from 10 to 20 minutes plus depending upon the volume of users. The CPU time is about 5 seconds.

The program currently stores permanently the following information:

- YOU I.D. Number
- Age of the Home
- Type of Fuel
- Attic Insulation Standard
- Total Floor Area and Volume of Home
- Average Temperature of the Home
- Total Ceiling Area and Weighted R-Value
- Total Wall Area and Weighted R-Value
- Infiltration Judgement for the Walls and Foundation
- Average Infiltration Judgement for the Doors and Windows
- Window and Door Area
- Heatloss in Fuel Units and Dollars
- Potential Heat Savings in Fuel Units and Dollars
- Percent of the Heatloss Due to Infiltration and Conduction

While the implementation of the program would require very little effort from one HP-3000 to another, some training with respect to the energy and heatloss aspects would greatly enhance the effectiveness of the audit. Because the audit can generate data related to residential energy use, there is a possibility that Federal funds can be obtained to provide some training. If you are interested in implementing the audit please contact one of the authors of this paper.

## APPENDIX A

A sample data summary sheet is included to illustrate the technique used to increase the efficiency of data entry. This sample data sheet consists of one folded piece of paper but is displayed here as four separate sheets.

Sample data entries are enclosed in both the no instruction and instruction modes. For some entries errors have been made to illustrate the kinds of data checks built into the program.

A copy of the output is enclosed. Note the narrative form of the output. This enhances the ability of the homeowner to read and understand the results. The sample data entry in the instruction mode and the output required 22 minutes of connect time and 7 seconds of CPU time. There were 23 users and the data entry and output were obtained via DSLINE communication. In the no instruction mode with no DSLINE the typical data in output out connect time is less than 15 minutes with a CPU time of 4 to 5 seconds.

Housing & Energy
West CAP
525 Second Street
Glenwood City, WI 54731

VISIT
#1 ________
#2 ________

NAME OF HOMEOWNER ____________________ Telephone # ____________

| | | |
|---|---|---|
| 1. DATE: MONTH, DAY, YEAR (E.G. 1,10,78) | 1. ___,___,___ | 19. ________ DATA ON BACK |
| 2. Y.O.U. ID NUMBER | 2. ________ | 20. ________ |
| 3. Assigned House Number (computer assigned, if you wish to recall program record here) | 3. ________ | 21. ________ |
| 4. Number of Individuals Occupying Home | 4. ________ | 22. ___, ___ |
| 5. AGE OF HOME (YEARS) | 5. ________ | 23. ___, ___ |
| 6. CONDITION OF ROOF?<br>1=Tight Roof - no leaks, good condition<br>2=Needs Minor Repair - missing shingles, cracks, possible leaks<br>3=Needs Replacing - bad condition, major leaks | 6. ________ | 24. ___, ___ DATA ON BACK<br>25. ________ DATA ON BACK |
| 7. CONDITION OF HEATING SYSTEM?<br>1=serviced within 1 year<br>2=serviced 1-5 years - owner can recollect it's having worked better<br>3=can't recollect last date of service | 7. ________ | 26. ________<br>27. ___, ___<br>28. ___, ___ |
| 8. AVERAGE HOME TEMPERATURE (FARENHEIT)<br>Day Time ______, Night Time ______ | 8. ___, ___ | |
| 9. DEGREE DAY ZONE (SEE PAGE 2) | 9. ________ | |
| 10. FUEL NUMBER AND COST (SEE PAGE 2) | 10. ___, ___ | ________ |
| 11. NUMBER OF STORIES IN HOME | 11. ________ | TO CONDUCTION |
| 12. 1st FLOOR AREA (SQ.FT.), AVG. WALL HT. (FT.) | 12. ___, ___ | |
| 13. 2nd FLOOR AREA (SQ.FT.), AVG. WALL HT. (FT.) | 13. ___, ___ | ________ |
| 14. 3rd FLOOR AREA (SQ.FT.), AVG. WALL HT. (FT.) | 14. ___, ___ | ________ |
| 15. FLOOR EXPOSURE FACTOR<br>1=Basement<br>2=Crawl Space<br>3=Walkout<br>4=Skirted Trailer/Building on posts<br>5=Unskirted Trailer/Building on posts | 15. ________ | |
| Are the basement walls insulated? 1=Yes 2=No | ________ | |
| Walkout wall area (sq.ft.) and R-value | ___, ___ | |
| 16. FOUNDATION PERIMETER (FT) | 16. ________ | |
| 17. AMOUNT FOUNDATION EXPOSED ABOVE GROUND (inches)? | 17. ________ | |
| 18. FOUNDATION CONDITION NUMBER<br>1=no cracks or settling, very clean<br>2=minor cracks, loose morter<br>3=major cracks and settling | 18. ________ | |

| FUEL (circle one or more, enter major fuel on item 10). | FUEL COST | |
|---|---|---|
| 1= #1 Fuel Oil | ________ | (cents per gallon) |
| 2= #2 Fuel Oil | ________ | (cents per gallon) |
| 3= Electricity | ________ | (cents per kwhr) |
| 4= Natural Gas by Cu. Ft. | ________ | (cents per cubic foot) |
| 5= Natural Gas by Therm | ________ | (cents per therm) |
| 6= Bottled Gas (LP Propane) | ________ | (cents per gallon) |
| 7= Coal, Coke | ________ | (dollars per ton) |
| 8= Kerosene | ________ | (cents per gallon) |
| 9= Wood | ________ | (dollars per full cord, 4x4x8 Ft.) |

If fuel cost is unknown, computer will assign an average cost.

COUNTY DEGREE DAY ZONES    Enter on item 9

Chippewa, Barron, Polk = 4
St. Croix, Dunn, Pierce, Pepin = 7

CRITERIA FOR DETERMINING THE CONDITION OF DOORS, WIDOWS AND WALLS

Condition 1= DOORS, WINDOWS- good fit, no draft is felt, caulk and weatherstripping are in good shape.
WALLS- finish in good shape.

Condition 2= DOORS- loose or missing weatherstripping.
WINDOWS- loose fit, no weatherstripping, caulk and glazing cracked and missing in sections, storm open or cracked.
WALLS- caulking between wall and frames is cracked, shrunk or nonexistant.
WALL EDGE AT FOUNDATION- shows minor deterioration.

Condition 3= DOORS- large gaps between door and jamb. Door cracked, sagging on hinges.
WINDOWS- very loose, window glass broken or missing. Frame shows rotted areas.
WALLS- rotted areas, large gaps between wall and frames.
WALL EDGE AT FOUNDATION- show deterioration, missing siding and holes. Foundation has many cracks or holes.

| | |
|---|---|
| 19. BASEMENT WINDOWS: HOW MANY LINES FILLED IN? ENTER BASEMENT WINDOW DATA FROM REVERSE SIDE | 19. ________ DATA ON BACK |
| 20. CARPETED FIRST FLOOR AREA (SQ.FT) INSULATED 1=Yes 2=No R-Value if Yes | 20. ________ ________ ________ |
| 21. First Floor Area NOT CARPETED (SQ.FT.) Insulated 1=Yes 2=No R-Value if Yes | 21. ________ ________ ________ |
| 22. UNINSULATED CEILING AREA (SQ.FT.) AND R-VALUE | 22. ____, ____ |
| 23. INSULATED CEILING AREA (SQ.FT.) AND R-VALUE MATERIAL__________ DEPTH______________ | 23. ____, ____ |
| 24. WINDOWS: HOW MANY LINES FILLED IN? DOUBLE HUNG 1=Yes 2=No ENTER WINDOW DATA FROM THE REVERSE SIDE | 24. ____, ____ DATA ON BACK |
| 25. DOORS: HOW MANY LINES FILLED IN? ENTER DOOR DATA FROM THE REVERSE SIDE | 25. ________ DATA ON BACK |
| 26. WALL CONDITION NUMBER (1, 2 or 3) | 26. ________ |
| 27. TOTAL OUTSIDE WALL AREA INSULATED (SQ.FT.) AND R-VALUE MATERIAL______________________________ | 27. ____, ____ |
| 28. TOTAL OUTSIDE WALL AREA UNINSULATED (SQ.FT.) AND R-VALUE | 28. ____, ____ |

COMPUTER OUTPUT

| | |
|---|---|
| HEAT LOSS ______________ | FUEL COST ______________ |
| _____% DUE TO INFILTRATION | _____% DUE TO CONDUCTION |
| POTENTIAL SAVINGS HEAT LOSS ______________ | FUEL COST ______________ |
| SQ.FT. ______________ | VOLUME ______________ |

COMPUTATION SPACE

BASEMENT WINDOW DATA

See second page to determine condition number for windows and doors.

| Direction N,S,E,W | Width Inches | Height Inches | Condition 1,2 or 3 | Good Storm 1=Yes, 2=No | Number of Windows |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

WINDOW DATA

| Direction N,S,E,W | Width Inches | Height Inches | Condition 1,2 or 3 | Good Storm 1=Yes, 2=No | Number of Windows |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

DOOR DATA

| Direction N,S,E,W, | Width Inches | Height Inches | Condition 1,2 or 3 | Good Storm 1=Yes, 2=No |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |

```
UW - River Falls HEATLOSS Program : TUE, OCT  3, 1978, 11:16 AM

Enter two (2) digit code for desired option :
  01 Home energy audit                02 Output stored audit information
  99 Stop
 %01


Storage space on file is available at this time, you may be able to
store your audit information and receive output at a later time.




NEED INSTRUCTIONS (1=YES OR 2=NO) ? 2



 1. ? 10,3,78
 2. ? 2
 3.   100013
 4. ? 1
 5. ? 25
 6. ? 1
 7. ? 1
 8. ? 68,66
 9. ? 4
10. ? 1
??,100
BAD INPUT--RETYPE FROM ITEM 1
??1,100

* * ENTRY NOT REALISTIC * * FUEL COST FOR TYPE OF FUEL INDICATED MUST
BE BETWEEN 30 AND 65 CENTS INCLUSIVE.  PLEASE REENTER DATA.

    DO YOU KNOW THE APPROPRIATE FUEL COST (1=YES OR 2=NO) ? 2
11. ? 1
12. ? 543,8
15. ? 1
      INSULATED ? 2
16. ? 94
17. ? 6
18. ? 1
19. ? 0
20. ? 180
      INSULATED (1=YES OR 2=NO) ? 2
21. ? 363
      INSULATED (1=YES OR 2=NO) ? 2
22. ? 0,0
23. ? 543,24
24. ? 0,1

* * ENTRY NOT REALISTIC * *A HOME MUST HAVE AT LEAST ONE (1) WINDOW.
PLEASE REENTER DATA.
```

```
24. WINDOWS : NUMBER OF LINES FILLED IN (1-45) AND DOUBLE HUNG (1=YES
    OR 2=NO) ? 3,1
    ? W,30,50,1,2,1
    ? S,30,50,1,1,3
    ? W,30,60,1,2,1
25. ? 1
    ? E,31,84,1,1
26. ? 3
27. ? 0,0
28. ? 752,3


DO YOU WISH TO STORE INFORMATION UNTIL LATER (1=YES 2=NO) ? 1

Enter two (2) digit code for desired option :
  01 Home energy audit              02 Output stored audit information
  99 Stop
 %99


HEATLOSS RUN TERMINATED.


READY
#BYE
CPU=3. CONNECT=10. TUE, OCT  3, 1978, 11:24 AM
* * * GOOD-BYE FROM WACC I * * * A SERVICE OF UW-RIVER FALLS * * *
END OF SESSION
0:

:BYE

CPU=3. CONNECT=11. TUE, OCT  3, 1978, 11:25 AM
* * * GOOD-BYE FROM WACC II * * * A SERVICE OF UW-RIVER FALLS * * *
END OF SESSION
```

```
UW - River Falls HEATLOSS Program : TUE, OCT  3, 1978, 10:47 AM

Enter two (2) digit code for desired option :
  01 Home energy audit              02 Output stored audit information
  99 Stop
 %01


Storage space on file is available at this time, you may be able to
store your audit information and receive output at a later time.



NEED INSTRUCTIONS (1=YES OR 2=NO) ? 1

     THIS PROGRAM IS DESIGNED TO SAVE COMPUTER TIME.  THEREFORE ANSWER
THE QUESTIONS AS ACCURATELY AS POSSIBLE.  PLEASE READ THE BOOKLET CARE-
FULLY AND USE THE DATA SUMMARY SHEET.  IF YOU DO NOT KNOW AN ANSWER, DO
NOT GUESS THE ANSWER.  IF YOU ARE UNSURE AND HAVE A CHANCE TO ANSWER NO,
THE COMPUTER WILL ASSIGN A REASONABLE VALUE WHERE POSSIBLE.  YOU WILL BE
GIVEN ONLY  4  CHANCES TO ENTER THE CORRECT INFORMATION (WITH HINTS) BE-
FORE BEING TERMINATED.



 1. DATE (MONTH, DAY, YEAR) ? 10,3,78
 2. Y.O.U. I.D. NUMBER ? 2
 3. HOUSE NUMBER : 100012
 4. NUMBER OF INDIVIDUALS OCCUPYING HOME ? 1
 5. AGE OF HOME (YEARS) ? 25
 6. CONDITION OF ROOF : 1 = TIGHT ROOF  2 = MINOR REPAIRS NEEDED
    3 = NEW ROOF NEEDED  ? 1
 7. CONDITION OF HEATING SYSTEM : 1 = GOOD CONDITION  2 = NEEDS REPAIR
    3 = NEEDS REPLACEMENT ? 1
 8. AVERAGE HOME TEMPERATURE (FAHRENHEIT) : DAYTIME AND NIGHTTIME.
    SEPARATE EACH ENTRY WITH A COMMA. FOR EXAMPLE: 68,66 ? 70,70
 9. ENTER THE ZONE NUMBER WHERE HOME IS LOCATED ? 7
10. ENTER FUEL NUMBER ? 5
    DO YOU KNOW THE FUEL COST (1=YES OR 2=NO) ? 1
    ENTER FUEL COST ? 25
11. ENTER NUMBER OF STORIES ? 1
12. ENTER FIRST FLOOR AREA (SQ.FT.) AND AVERAGE WALL HEIGHT
    (FT.) ? 543,8
15. ENTER FLOOR EXPOSURE FACTOR ? 1
    ARE THE BASEMENT WALLS INSULATED (1=YES OR 2=NO) ? 2
16. ENTER FOUNDATION PERIMETER IN FEET ? 94
17. ENTER AMOUNT FOUNDATION IS EXPOSED IN INCHES ? 6
18. ENTER FOUNDATION CONDITION NUMBER ? 2
19. HOW MANY LINES FILLED IN FOR BASEMENT WINDOWS ? 2
    ENTER BASEMENT WINDOW DATA AS FOLLOWS : ENTER DIRECTION (N, S, E OR
    W), WIDTH (INCHES), HEIGHT (INCHES), CONDITION (1, 2 OR 3), GOOD
    STORMS (1=YES OR 2=NO), NUMBER OF WINDOWS OF SAME DIRECTION, CONDITION,
    ETC.  SEPARATE EACH ENTRY BY A COMMA.  FOR EXAMPLE : N,24,16,1,1,2
    ? E,30,12,1,2,1
    ? N,30,12,1,2,1
```

```
20. IS ANY OF THE FIRST FLOOR AREA CARPETED (1=YES OR 2=NO) ? 1
    ENTER CARPETED FIRST FLOOR AREA IN SQ. FT. ? 180
      INSULATED (1=YES OR 2=NO) ? 2
21. IS ANY OF THE FIRST FLOOR NOT CARPETED (1=YES OR 2=NO) ? 1
    ENTER FIRST FLOOR AREA NOT CARPETED IN SQ.FT. ? 363
      INSULATED (1=YES OR 2=NO) ? 2
22. ENTER UNINSULATED CEILING AREA (SQ.FT.) AND R-VALUE.  SEPARATE EACH
    NUMBER BY A COMMA.  IF THERE IS NO UNINSULATED CEILING AREA, ENTER
    A 0 FOR THE CEILING AREA AND A 0 FOR THE R-VALUE ? 0,0
23. ENTER INSULATED CEILING AREA (SQ.FT.) AND R-VALUE.  SEPARATE EACH
    NUMBER BY A COMMA.  IF THERE IS NO INSULATED CEILING AREA, ENTER A
    0 FOR THE CEILING AREA AND A 0 FOR THE R-VALUE ? 643,24

* * ENTRY NOT REALISTIC * *TOTAL CEILING AREA DOES NOT EQUAL INSULATED
PLUS UNINSULATED CEILING AREA (WITHIN 10%).  PLEASE REENTER DATA.

22. ENTER UNINSULATED CEILING AREA (SQ.FT.) AND R-VALUE.  SEPARATE EACH
    NUMBER BY A COMMA.  IF THERE IS NO UNINSULATED CEILING AREA, ENTER
    A 0 FOR THE CEILING AREA AND A 0 FOR THE R-VALUE ? 0,0
23. ENTER INSULATED CEILING AREA (SQ.FT.) AND R-VALUE.  SEPARATE EACH
    NUMBER BY A COMMA.  IF THERE IS NO INSULATED CEILING AREA, ENTER A
    0 FOR THE CEILING AREA AND A 0 FOR THE R-VALUE ? 543,24
24. WINDOWS : NUMBER OF LINES FILLED IN (1-45) AND DOUBLE HUNG (1=YES
    OR 2=NO) ? 7,1
    ENTER WINDOW DATA AS FOLLOWS : ENTER DIRECTION (N, S, E OR W), WIDTH
    (INCHES), HEIGHT (INCHES), CONDITION (1, 2 OR 3), GOOD STORMS (1=YES
    OR 2=NO), NUMBER OF SAME DIRECTION, CONDITION, ETC.  SEPARATE EACH
    ENTRY WITH A COMMA.  FOR EXAMPLE : N,28,53,2,1,3
    ? S,63,51,1,2,1
    ? N,2
        36,33,3,1,1
    ? N,48,33,1,1,1
    ? N,48,33,1,1,2
    ? E,54,33,1,1,1
    ? W,48,36,1,1,1
    ? W,39,33,2,2,1
25. DOOR DATA : HOW MANY LINES FILLED IN ? 2
    ENTER DOOR DATA AS FOLLOWS : DIRECTION (N, S, E OR W), WIDTH (IN
    CHES), HEIGHT (INCHES), CONDITION (1, 2 OR 3), STORMS (1=YES
    OR 2=NO).  SEPARATE EACH ENTRY WITH A COMMA.  FOR EXAMPLE : N,36,
    84,1,1
    ? S,36,84,2,2
    ? W,72,84,1,1
26. WALL CONDITION NUMBER ? 1
27. ENTER TOTAL OUTSIDE WALL AREA INSULATED AND R-VALUE ? 0,0
28. ENTER TOTAL OUTSIDE WALL AREA UNINSULATED AND R-VALUE ? 752,3


DO YOU WISH TO STORE INFORMATION UNTIL LATER (1=YES 2=NO) ? 2
```

```
TO DETERMINE THE ACCURACY OF THE COMPUTER RESULTS, COMPARE THE COMPUTED
FUEL BILL TO THE ACTUAL FUEL BILL.  THE CLOSER THE MATCH THE MORE ACCU-
RATE THE RESULTS.  THE POTENTIAL SAVINGS INDICATED ARE FOR THE FIRST
COMPLETE HEATING SEASON.  POTENTIAL DOLLAR SAVINGS ASSUME THAT THE PRICE
OF FUEL REMAINS THE SAME.  MATERIAL COSTS VARY A GREAT DEAL AND SHOULD
BE CHECKED LOCALLY.

* * * SURVEY QUESTION 8. * * *

AVERAGE HOME TEMPERATURE IS  70

YOU SHOULD LIVE AT A LOWER TEMPERATURE, 68 DEGREES FAHRENHEIT OR LESS.

* * * SURVEY QUESTION 7. * * *

HEATING EFFICIENCIES ASSUMED ARE : 65% OIL, LP AND NATURAL GAS, 100%
ELECTRIC, 60% COAL AND 50% WOOD.

YOUR FURNACE IS IN GOOD CONDITION - YOUR EFFICIENCY IS PROBABLY AS
STATED.

YOUR HEATLOSS IS    1111.7     THERMS GAS
ASSUMING THAT YOUR FUEL COSTS ARE  25.00 CENTS PER    THERM      YOUR
FUEL BILL SHOULD BE ABOUT $  277.91.
OF THIS HEATLOSS  26   % IS DUE TO INFILTRATION LOSSES
                  74   % IS DUE TO CONDUCTION LOSSES.

* * * INFILTRATION LOSSES SHOULD BE CONSIDERED FIRST * * *

* * * SURVEY QUESTIONS 19. & 24. * * *

WINDOW DATA : ONLY WINDOWS NEEDING WORK OR WITH TRIPLE GLAZED
############# SAVINGS GREATER THAN $2 ARE LISTED
HEATLOSS =       12.1     THERMS GAS    $  3.03 (CONDUCTION)    - BASEMENT
HEATLOSS =        1.4     THERMS GAS    $   .36 (INFILTRATION) - BASEMENT
HEATLOSS =      177.1     THERMS GAS    $ 44.28 (CONDUCTION)
HEATLOSS =       80.1     THERMS GAS    $ 20.02 (INFILTRATION)

                                                            POTENTIAL
                                        POTENTIAL POTENTIAL SAVINGS
                                        SAVINGS   SAVINGS   IF WEATHER
                                WEATHER IF WEATHER IF WEATHER STRIPPED
          WIDTH  HEIGHT STORM  STRIPPING STRIPPED  STRIPPED  AND TRIPLE
DIRECTION INCHES INCHES NEEDED NEEDED    (ONLY)    AND STORM GLAZED
------------------------------------------------------------------------
B-EAST      30     12    YES     NO      $   .01   $    .64  $    .94
B-NORTH     30     12    YES     NO      $   .01   $    .64  $    .94

  SOUTH     63     51    YES     NO      $   .12   $   7.24  $   9.94
  NORTH     36     33    NO      YES     $  4.72   $   4.72  $   5.72
  WEST      39     33    YES     YES     $  1.56   $   4.41  $   5.49
                                         -------------------------------
                                  TOTALS $  6.43   $  17.65  $  30.00

  2   BASEMENT STORM WINDOWS NEEDED
                USING ALUMINUM STORMS  AVG. RETAIL PRICE = $  24.00

 29   LINEAR FEET OF WEATHERSTRIPPING
      NEEDED FOR WINDOWS               AVG. RETAIL PRICE = $   2.97

  2   STORM WINDOWS NEEDED
                USING ALUMINUM STORMS  AVG. RETAIL PRICE = $ 100.00
```

```
* * * SURVEY QUESTION 25. * * *

DOOR DATA : ONLY DOORS NEEDING WORK ARE LISTED.
#########
HEATLOSS =     119.4    THERMS GAS     $ 29.86 (CONDUCTION)
HEATLOSS =     101.8    THERMS GAS     $ 25.44 (INFILTRATION)


                                             POTENTIAL  POTENTIAL
                                             SAVINGS    SAVINGS
                                     WEATHER  IF WEATHER IF WEATHER
          WIDTH  HEIGHT STORM  STRIPPING STRIPPED   STRIPPED
DIRECTION INCHES INCHES NEEDED NEEDED    (ONLY)     AND STORM
---------------------------------------------------------------
  SOUTH     36      84    YES      YES       $ 16.70     $ 23.06
                                              --------------------
                                      TOTALS $ 16.71     $ 23.07

 20   LINEAR FEET OF WEATHERSTRIPPING
      NEEDED FOR DOORS                     AVG. RETAIL PRICE = $    2.00
  1   STORM DOORS NEEDED
                   USING ALUMINUM STORMS   AVG. RETAIL PRICE = $   75.00

* * * SURVEY QUESTION 18. * * *

FOUNDATION INFILTRATION DATA:
##############################
HEATLOSS =      30.3    THERMS GAS     $  7.57 (INFILTRATION)

FOUNDATION NEEDS SOME WORK, PARTICULARLY CAULKING
CRACK LENGTH ASSOCIATED WITH SILL AND FOUNDATION IS     94 LINEAR FEET
  9   TUBES OF CAULKING MAY BE NEEDED *** AVG. RETAIL PRICE = $   13.50

POTENTIAL SAVINGS:     20.6      THERMS GAS      $     5.14

* * * SURVEY QUESTION 26. * * *

WALL INFILTRATION DATA:
########################
HEATLOSS =      59.8    THERMS GAS     $ 14.94 (INFILTRATION)

WALL CONDITION IS GOOD, NO WORK NEEDED; NO POTENTIAL SAVINGS.

CONDUCTION LOSSES SHOULD BE CONSIDERED SECOND.
```

```
 * * * SURVEY QUESTIONS 6.,22. & 23. * * *

CEILING DATA:
#############
HEATLOSS =        .0    THERMS GAS     $   .00 (UNINSULATED)
HEATLOSS =      55.8    THERMS GAS     $ 13.96 (INSULATED)

INSULATED CEILING AREA =    543  SQUARE FEET CURRENT R-VALUE =  24.0
ADDITIONAL INSULATION NEEDED TO ACHIEVE AN R-VALUE OF ABOUT R= 33
  2  SETTLED INCHES OF BLOWN CELLULOSE R=3.7/IN.
  8   30 LB BAGS OF CELLULOSE              AVG. RETAIL PRICE = $  40.00

***** OR *****

ASSUME ONLY 3 AND 6 INCH BATTS ARE EASILY AVAILABLE

  6  ROLLS OF 6.0 X 23 FIBER GLASS BATTS  AVG. RETAIL PRICE = $ 173.76

WITH THIS FIBER GLASS ADDED TOTAL R-VALUE = 43.0

VENTILATION REQUIREMENTS (ASSUMES NO VAPOR BARRIER EXISTS)
ABOUT  1.8 SQ.FT. INLET  AVG. RETAIL PRICE = $  3.75 USING ATTIC VENTS
ABOUT  1.8 SQ.FT. OUTLET AVG. RETAIL PRICE = $ 12.00 USING ROOF VENTS

POTENTIAL SAVINGS:      16.0     THERMS GAS     $    3.99

* * * SURVEY QUESTIONS 27. & 28. * * *

WALL DATA:
##########

* * * WALL INSULATION STANDARD USED IS R-11* * *
* * * TOTAL WALL R-VALUE STANDARD IS R-15  * * *

HEATLOSS =    287.4    THERMS GAS    $ 71.85 (CONDUCTION)

TOTAL WALL AREA =     752  SQUARE FEET
TOTAL WINDOW AND DOOR AREA =    159  SQUARE FEET
NET WALL AREA =    592  SQUARE FEET

100 % OF THE NET WALL AREA IS NOT INSULATED R-VALUE =   3.0
ADDITIONAL INSULATION NEEDED TO ACHIEVE AN R-VALUE OF ABOUT R=15
 3.5 INCHES OF BLOWN CELLULOSE R=3.7/IN.
 19   30 LB BAGS OF CELLULOSE              AVG. RETAIL PRICE = $  94.00

POTENTIAL SAVINGS UNINSULATED WALLS:    204.4    THERMS GAS    $  51.11
```

```
* * * SURVEY QUESTIONS 12.,15.,16.,17.,20. & 21. * * *

BASEMENT DATA :
#################
HEATLOSS =     186.4     THERMS GAS     $ 46.60 (CONDUCTION)


OF THE HEAT LOST TO YOUR BASEMENT  72% IS LOST THROUGHTHE EXPOSED PART
OF THE BASEMENT WALL, 18% IS LOST THROUGH THE PART NOT EXPOSED AND  9%
IS LOST THROUGH THE BASEMENT FLOOR.

IF YOU INSULATE THE BASEMENT WALLS, THE HEAT LOST TO THE BASEMENT IS :

INSULATION TYPE      CALCULATED HEATLOSS           POTENTIAL SAVINGS
---------------      THERMS GAS      DOLLARS      THERMS GAS      DOLLARS

   R- 4 WALL             134.0       $ 33.51          52.4        $ 13.10
   R-11 WALL             115.0       $ 28.75          71.4        $ 17.85


HOT WATER DATA:
###############

FOR THE    1      INDIVIDUALS OCCUPYING THE HOME
THE HOT WATER COSTS PER YEAR ARE ABOUT

          $  63   IF ELECTRIC AT $.03/KWHR
          $  20   IF NATURAL GAS AT $.25/THERM
          $  35   IF LP AT $.40/GALLON
          $  28   IF OIL AT $.47/GALLON


IF YOU WISH ADDITIONAL INFORMATION CONTACT:
WESTCAP HOUSING AND ENERGY, GLENWOOD CITY, WI. 715-265-4271.
UW RIVER FALLS, PHYSICS/ENERGY, RIVER FALLS, WI. 715-425-3196.
WISCONSIN OFFICE OF STATE PLANNING AND ENERGY
MADISON, WI. 608-266-6850.

##########################################################################

TOTAL POTENTIAL SAVINGS:    475.2     THERMS GAS     $118.80

##########################################################################

Enter two (2) digit code for desired option :
  01 Home energy audit              02 Output stored audit information
  99 Stop
 %09


HEATLOSS RUN TERMINATED.
```

# A DECISION SUPPORT SYSTEM TO ASSIST IN CONTAINERBOARD LOGISTICS MANAGEMENT

PETER DIGIAMMARINO AND RICHARD SCHWARTZ
AMERICAN MANAGEMENT SYSTEMS, INC.
SAN MATEO, CALIFORNIA

## ABSTRACT

Management decision-making must be responsive to changing economic conditions, dynamic internal business circumstances, and an expanding set of operational and strategic issues relevant to business policy-making. Decision-making in this environment requires timely access to accurate and complete information pertinent to specific business situations. Meeting this requirement is complicated by a narrowing time frame in which highly volatile data must be organized and analyzed before it can be assimilated into the decision-making process. Effective management of containerboard production and distribution requires an information system that supports decision-making in all phases of business activity: operational control, management control and strategic planning. The Brownboard Order And Rollstock Distribution System (BOARDS) integrates management science and operations research with advanced computer technology and human decision-making to support demand-based production planning in the corrugated shipping container industry.

American Management Systems, Inc., a management consulting and system development firm, developed BOARDS for the Shipping Container and Containerboard Marketing Division of Weyerhaeuser, a company principally engaged in the manufacture, distribution and sale of forest products. BOARDS operates on a dedicated Hewlett-Packard 3000 Series II computer with 512K bytes of main memory. Nine distinct IMAGE data bases constitute the system's foundation. Over one hundred BOARDS functions written in COBOL, SPL and FORTRAN have been aggregated into six integrated subsystems. Most of the system was installed in the fall of 1978; further development is underway and will be installed early in 1979.

## INTRODUCTION

The corrugated shipping container industry is highly sensitive to dynamic market and economic conditions. Increased competition, technological advances that have yielded attractive substitutes, and tightening environmental regulations are factors contributing to the rising production costs and unstable demand facing the industry. When the lumber business is prospering, the integrated forest products manufacturers (those companies that make lumber, brownboard and boxes) run their containerboard mills at full capacity in order to consume prodigious quantities of wood chips. Flooding the market with containerboard in periods of depressed box demand results in excess containerboard inventories and narrowing profit margins. Containerboard inventory distribution, in addition to supply considerations, is also an integral part of sales and profit performance.

The corrugated shipping container producers have every incentive to adjust operating policies and planning strategies to meet the challenge of achieving production capacity and improving profit margins. Alternative means of adjusting the market conditions are to control production of containerboard at the mill; discover ways to increase demand; or institute mechanisms for supply planning, demand forecasting, and mill scheduling which result in inventory levels that are responsive to actual consumer demand and that are economically justified.

## EXAMINING THE ALTERNATIVES

Mill production of containerboard typically runs in cycles; production of light-weight grades followed by heavy-weights and back to light-weights. Gradual increments in grade/basis weight production maintain an efficient cycle whereas irregular changes require substantial machine set up time, are disruptive, and drive up production costs. Extreme fluctuations in box demand do not coincide with an orderly, efficient production cycle. Satisfying uneven fluctuations in demand by increasing or decreasing production volume, while maintaining an efficient grade cycle, adds to the high operating cost of containerboard production. Therefore, production control mechanisms are expensive and undesirable. Even if production is responsive to demand, situations may still occur such that it would be cost effective to slow or shut down the mill; but generally only as a last resort.

Shipping container manufacturers have campaigned vigorously to increase demand for their product. As evidence of this, the integrated companies have chosen to enter markets for low volume, specialty containers. New foreign markets are also being explored. The continuous growth in demand for corrugated boxes that prevailed during the 1960's at 5.6% per year slowed to about 3% per year in the early 1970's and is optimistically expected to equal the growth of the U.S. economy in the long run (Business Week March 13, 1978). Efforts to generate increased demand for shipping containers continue, but are not a practical long-term solution to problems associated with containerboard inventory management under volatile market conditions.

A practical long-term strategy is to achieve improved management of existing facilities in response to consumer demand. The objectives are to minimize inventory stockpiling and to avoid lost sales due to poor inventory distribution. This is accomplished if each corrugated box shop is supplied with, or has access to, sufficient and specific quantities of brownboard needed to satisfy existing and forecasted customer orders. Containerboard Logistics Management is responsible for achieving this demand/supply balance. The difficulty associated with establishing demand-based production and planning policy is that large quantities of information must be assimilated into the logistics management decision-making process. Fluctuations in demand must be accurately monitored for each box shop and market region; up-to-date inventory levels at the box shops and the mills, and quantities in-transit must be known at all times; and mechanisms to process, track and assess the status of containerboard orders must be in place.

A system that provides this information must support containerboard logistics decision-making in all phases of management activity: operational control, management control and strategic planning. Within each type of activity, the system must support decision-making that ranges from structured (such as order entry, inventory control, consumption forecasting, and pricing) to semi-structured (such as mill scheduling, simulating policy decisions to set target weeks of containerboard inventory at the box shops, and demand/supply planning analysis).

## A FRAMEWORK FOR CONTAINERBOARD LOGISTICS DECISION MAKING

The usefulness of such a system extends beyond the realm of management control. Day-to-day operational support to the box shops and containerboard mills is also achieved. Order entry, containerboard production, shipment, invoice and receipt transaction processing facilitate ongoing activities at distributed sites while simultaneously supplying a central pool of management information. Decisions to contract for additional containerboard orders or to purchase additional containerboard are based on better and more complete information by providing box shops direct access to the system.

A system to manage a demand-based production strategy must be capable of accurately capturing and monitoring consumption forecasts and converting those forecasts into orders that can be filled economically. Access to weekly box shop consumption forecasts, current on-hand and in-transit quanities, and outstanding orders provides containerboard logistics management with the information required to determine the grade and quantity of brownboard to order for each box shop in the upcoming weeks. Mill production rates, warehouse inventory levels, and trade agreement balances provided by the system are also used by management to help make economical decisions when placing orders.

A system to capture, store, process, and report information in a timely and usable manner must be responsive to dynamic market conditions to justify its use in any market. It is intuitively appealing to posit that a well organized demand-based production strategy will function well under any market conditions, although it is especially cost effective in tight markets. Using the system to simulate alternative supply planning strategies in accordance with projected economic circumstances further contributes to its usefulness in diverse market situations.

American Management Systems, Inc., a management consulting and systems development firm, has developed the Brownboard Order And Rollstock Distribution System (BOARDS) for the Shipping Container and Containerboard Marketing Division of Weyerhaeuser, a company principally engaged in the manufacture, distribution, and sale of forest products. BOARDS provides information services for all phases of containerboard logistics management activity: operational control, management control, and strategic planning. The functions of BOARDS map well into the Framework for Management Information Systems developed by Gorry and Scott Morton (Gorry and Scott Morton 1971) from Anthony's taxonomy of management activity (Anthony 1965) and Simon's continuum of managements' approach to decision-making (Simon 1960). Figure 1 shows the array of decision-making activity supported by BOARDS within this framework.

To the extent that BOARDS handles routine processing needs (e.g., order entry, inquiry and update, and inventory control) it is a conventional management information system. These functions are structured in the sense that they are based upon well understood, easily automated processes and require limited human intervention. The rest of BOARDS supports decision-making that is much less mechanical. Order sourcing, mill scheduling, box shop inventory replenishment and demand/supply planning are examples of functions that require relevant information to be processed and acted upon differently depending upon conditions that are so dynamic and diverse that they cannot be effectively automated. In these cases, BOARDS is designed to assist the decision-maker by providing rapid access to timely information in the format most appropriate to the situation at hand. As the first two columns in Figure 1 depict, some BOARDS functions assist institutional decision-making; i.e., decision-making required to manage everyday business situations. Conversely, some BOARDS functions aid in ad-hoc decision-making as shown in the third column of Figure 1 (Donovan and Madnick 1977). BOARDS is, therefore, more than a conventional management information system and is appropriately termed a Decision Support System (Keen and Scott Morton 1977).

## BOARDS OVERVIEW

The distribution of containerboard is illustrated as a cyclical flow in Figure 2. Contemporary containerboard distribution systems are driven by mill production and containerboard supply capabilities. With BOARDS, however, the cycle starts with consumption forecasts provided by the box shops. This input is based upon actual sales agreements, outstanding orders, seasonal trends, and prevailing market conditions. Containerboard Logistics (C/L) management has the option of scaling forecasts up or down, either across all box shops and products or by individual box shop, based upon its

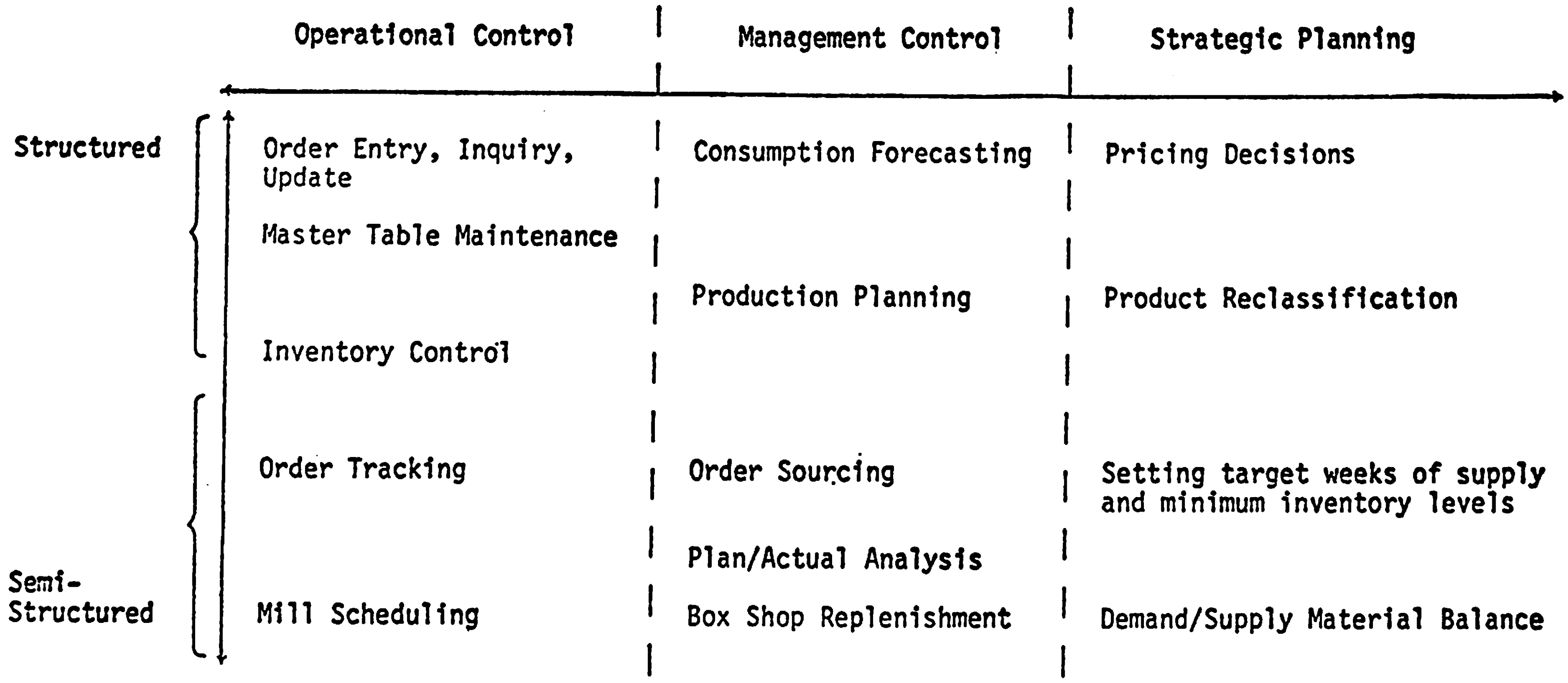


Figure 1

BOARDS-supported Decision Making Activity

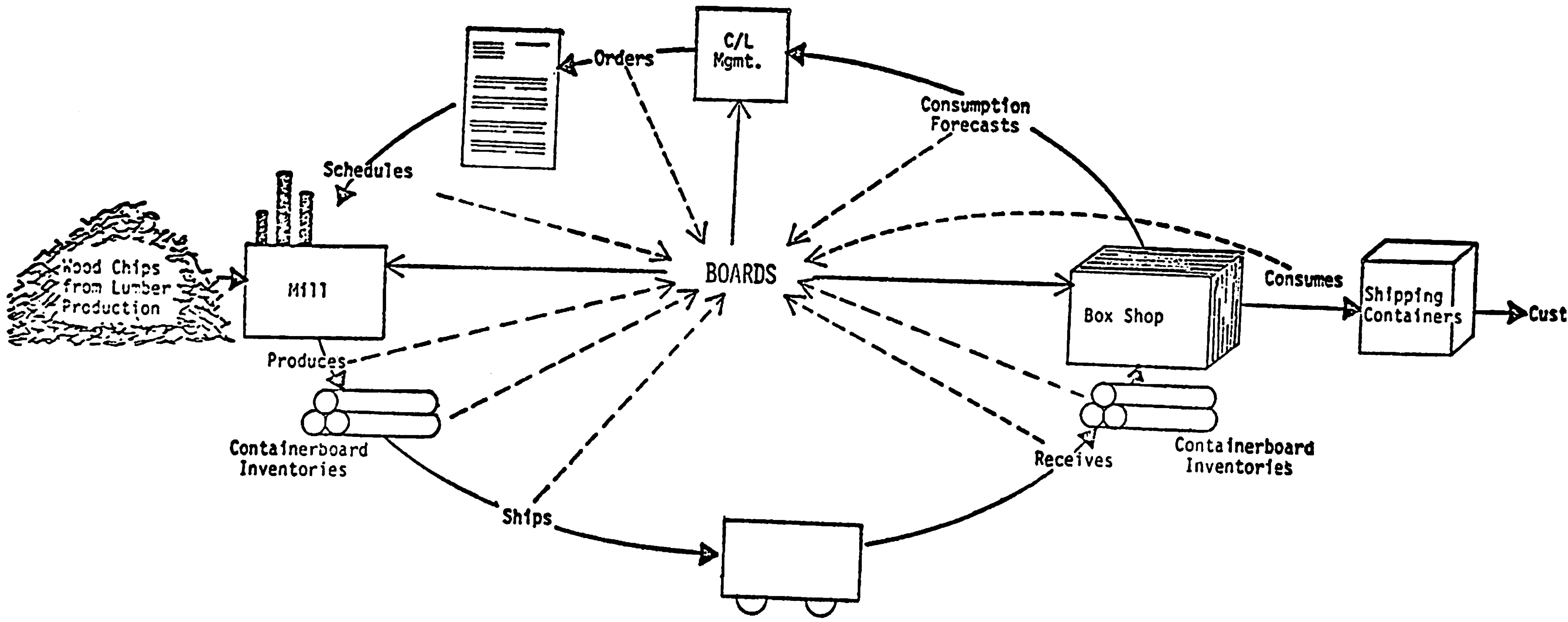


Figure 2
Containerboard Distribution System
and Information Flow

perception of market trends and prevailing or projected economic circumstances. Based upon these forecasts, and knowledge of unfilled orders and current inventory levels, C/L generates mill orders for containerboard. Taking into account current mill production schedules, in-transit times, and production capacities, these orders are then sourced and scheduled at the mills. Mills produce what is ordered, accumulating inventories until shipments are made. The containerboard is then in-transit until it is received by the box shop where it is stored and utlimately consumed.

BOARDS serves two fundamental purposes in this distribution cycle, also shown in Figure 2. Each activity in the cycle is recorded by BOARDS as it compiles a reservoir of information and assists in reconciling operational discrepancies. Correspondingly, BOARDS is a source of information. The mills may examine their production backlog, and the box shops can determine how much brownboard is in-transit, how much is yet to be scheduled, and so forth. C/L may request aggregate or detailed information concerning the current status of containerboard flowing through the system.

BOARDS operates on a dedicated Hewlett-Packard 3000 series II computer with 512K bytes of main memory. Nine distinct IMAGE (Hewlett-Packard's data base management system) data bases constitute the system's foundation. Over one hundred BOARDS functions are aggregated into six integrated subsystems: Order Processing, Production Planning and Mill Scheduling, Inventory Control, Trade Tracking, Planning and Material Balance, and Mill Price Difference Reporting as depicted in Figure 3. BOARDS supports management decision-making at Weyerhaeuser headquarters in Federal Way, Washington, and at 31 box shops and four mills located across the country. BOARDS has added flexibility since it interfaces directly with various computer systems located at the four mills and indirectly via magnetic tape transfer with existing computer systems running on Honeywell equipment at company headquarters. Figure 4 shows an overview of BOARDS hardware configuration and telecommunications network.

A detailed example of two integrated BOARDS functions, Box Shop Replenishment and Mill Scheduling, illustrates the system's potential to monitor and control demand-based production.

## BOX SHOP REPLENISHMENT

The Box Shop Replenishment (BOXREP) function, as shown in Figure 5, retrieves weekly consumption forecasts from the Planning Data Base, current on-hand and in-transit quantities from the Inventory Data Base, and outstanding orders from the Order Master Data Base. This data is used to automatically determine the grade and quantity of containerboard to order for each Box Shop in the upcoming weeks. The C/L Box Shop Service Representative interactively specifies input parameters which include whether to use economic adjustment factors that account for recent changes in the market, whether to use target or minimum inventory levels, and report specifications. In-transit times between the mills and the box shops are also used in computing order quantities and shipment dates. Default parameters are supplied from the BOARDS Master Table Data Base upon user request. The results of the computations are stored on the Want-to-Ship Data Base. Quantities stored in this data base represent the amount of containerboard that should be placed on order by box shop and week for the next six weeks in order to meet projected consumer demand.

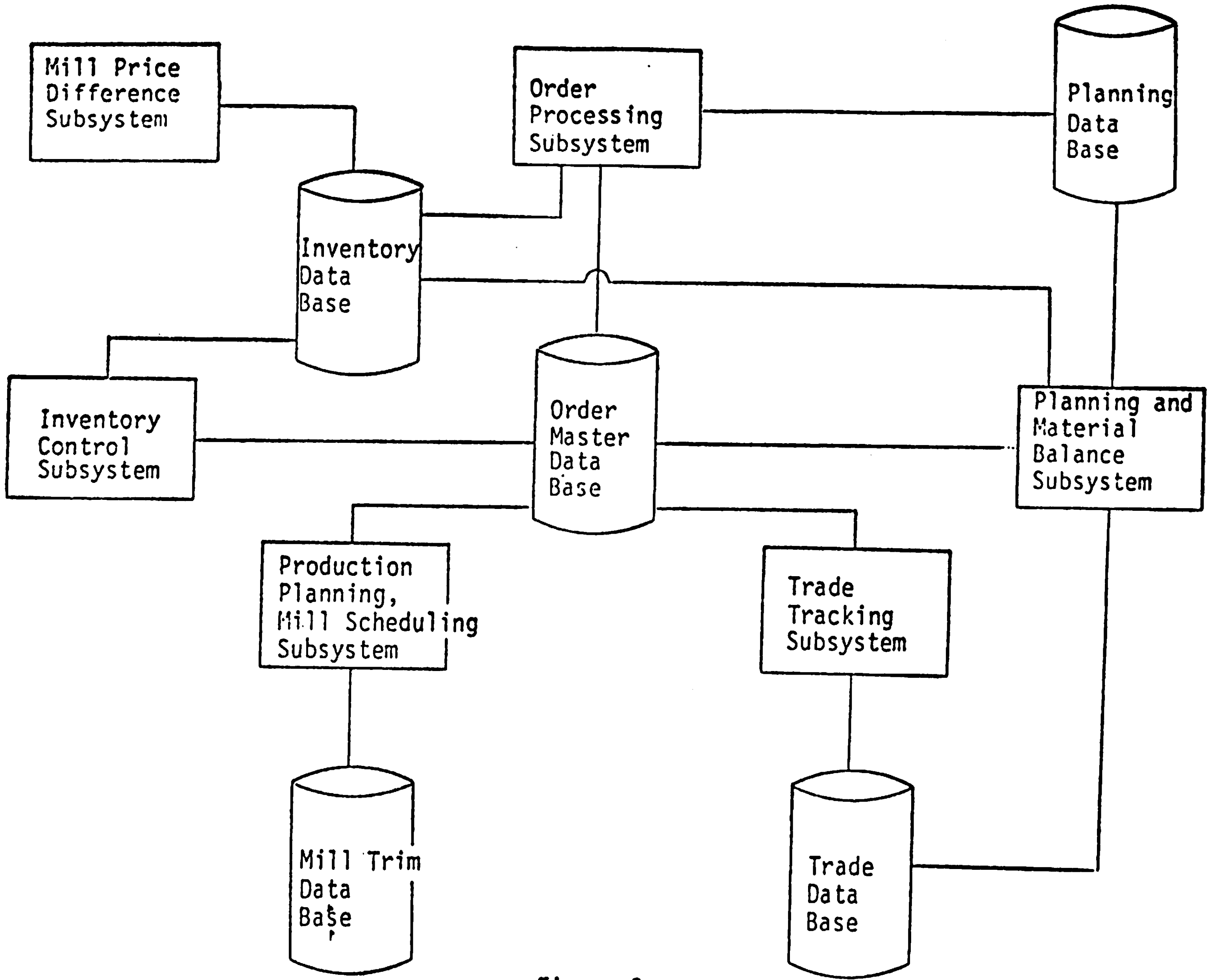

Mill Price Difference Subsystem
Order Processing Subsystem
Planning Data Base
Inventory Data Base
Inventory Control Subsystem
Order Master Data Base
Planning and Material Balance Subsystem
Production Planning, Mill Scheduling Subsystem
Trade Tracking Subsystem
Mill Trim Data Base
Trade Data Base

Figure 3

BOARDS Subsystems and Data Bases

(four auxiliary and system data bases not shown)

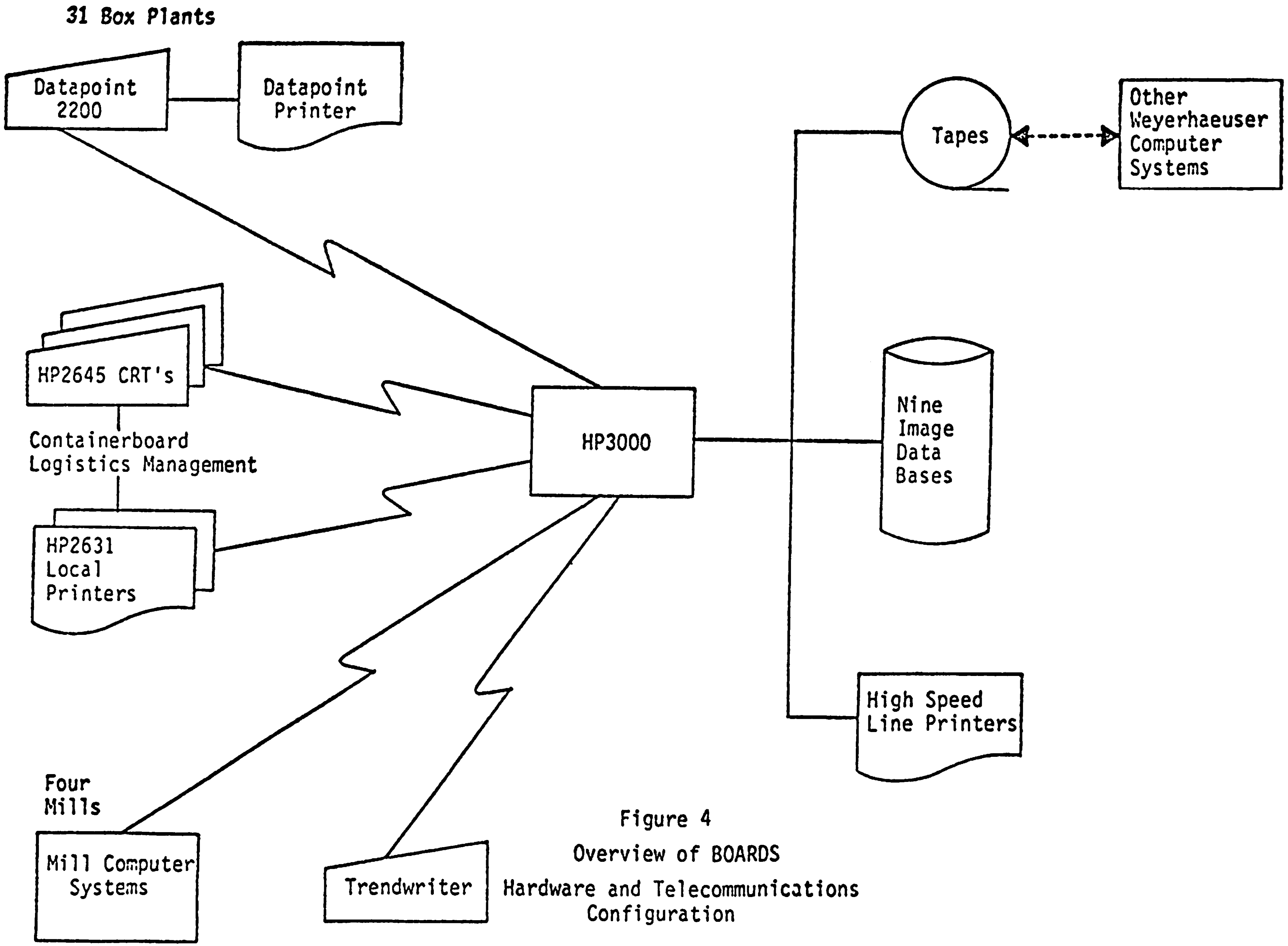


Figure 4

Overview of BOARDS

Hardware and Telecommunications Configuration

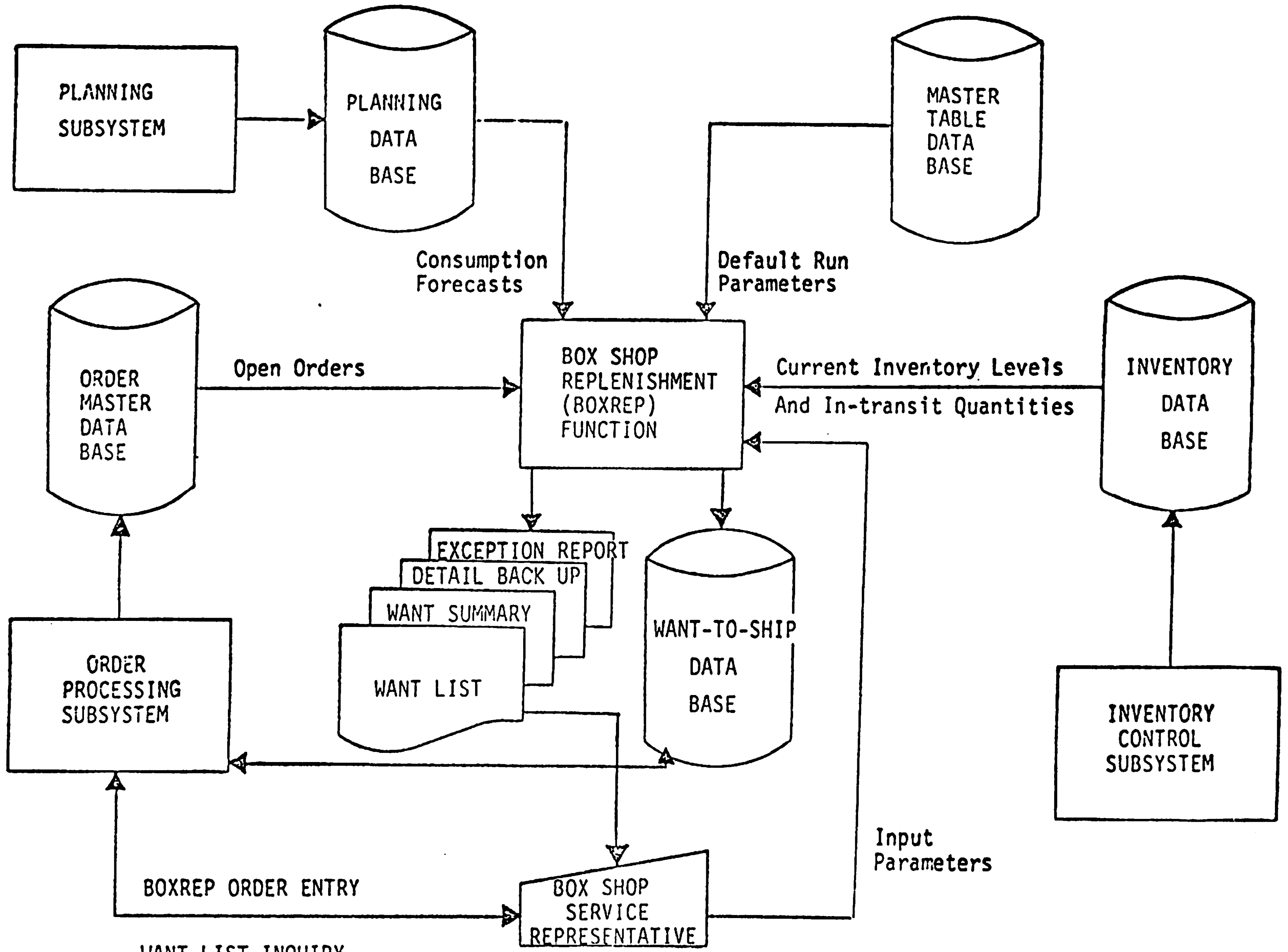


Figure 5

BOX SHOP REPLENISHMENT FUNCTION

BOXREP results can be examined in any of three ways. The first output is in the form of standard reports. The reports show Want-to-Ship quantities in varying levels of detail and aggregation. BOXREP also produces an exception report that shows current and projected crisis conditions by week for each box shop and product (for example; inventory levels are projected to fall below minimum, or expected receipts are greater than capacity). The second mechanism for inspecting Want-to-Ship quantities shows a box shop's proposed shipments in the form of a purchase order on a CRT. The information is displayed, can be revised and, optionally, converted into an open purchase order. The quantities stored on the Want-to-Ship Data Base are automatically adjusted to account for containerboard that is placed on order. Lastly, C/L management or users at the box shops can examine the current contents of the Want-to-Ship Data Base. A BOARDS inquiry function interactively displays the proposed shipment quantities by box shop, product and ship week, adjusted for orders already placed.

Proposed shipments are never automatically converted to open orders, since many non-quantifiable factors affect the decision to place an order. Surrounding economic conditions can drastically alter the usefulness of proposed shipments. Therefore, manual involvement is critical. Support for this sort of semi-structured decision-making is what distinguishes BOARDS from conventional management information systems. The utility of BOXREP is in <u>assisting</u> order generation rather than in <u>automating</u> it.

While BOXREP is engaged in information retrieval, calculations and reporting, the user is free to execute other BOARDS functions since the BOXREP processing is performed in batch mode. BOXREP can also be run iteratively to test the impact of alternate consumption and inventory adjustment factors, policy variables or target inventory levels without interfering with the active contents of the Want-to-Ship Data Base. When a desirable set of calculations has been achieved the data base can then be updated. BOXREP logic is configured such that several BOXREP processes can be under way simultaneously as long as only one is updating the data base. This flexibility has been provided to enhance BOXREP's utility as a planning facility in addition to its use as a production system.

## MILL SCHEDULING

Mill scheduling, as shown in Figure 6, consists of three functions: Trim Input, Trimming, and Mill Order Maintenance. Containerboard Logistics' mill schedulers use these functions to specify the quantity, sequence, and date in which ordered containerboard is to be produced at each mill. Their objectives are to maximize production efficiency and minimize trimming loss. The scheduler uses the Trim Input function to interactively select unscheduled purchase orders from the Order Master Data Base for production at the mills. The retrieved orders are displayed on a CRT as a "Trim Input". The scheduler can modify the input as desired to obtain a tentative set of orders to be scheduled together at a mill. Once satisfied with the Trim Input, the scheduler specifies a trim algorithm, either the linear programming

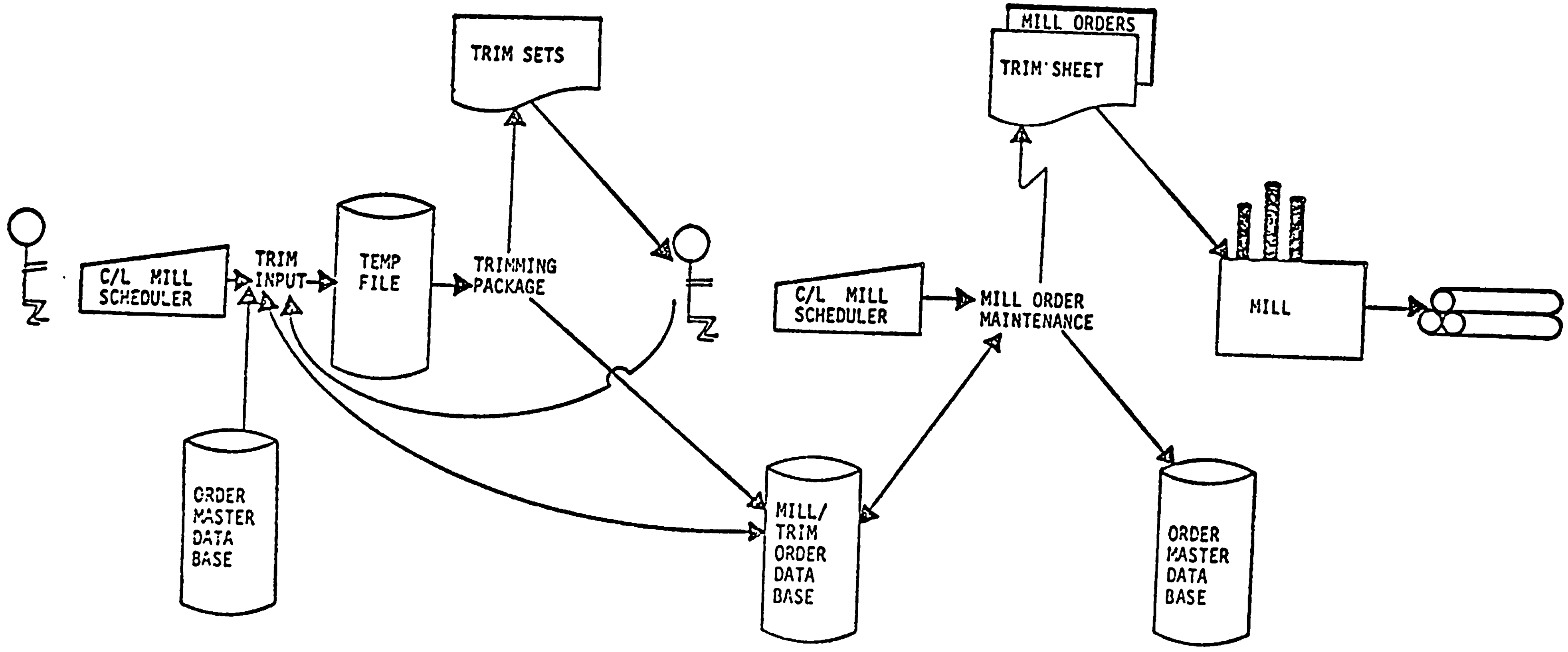


Figure 6
BOARDS MILL SCHEDULING

model or the heuristic program, and the Trim Input is passed to the Trimming function via a temporary file. The Trimming function executes the selected trim algorithm, stores the results in the Mill/Trim Order Data Base, and prints the results, referred to as a Trim Set. The Trim Set shows the near optimum positions in which to place cutting knives in order to trim the maximum number of ordered widths per containerboard roll and minimize the trim loss.

The Mill Order Maintenance function converts a Trim Set into a Mill Order. If the Trim Set is satisfactory, the scheduler can interactively convert it by assigning a production date and sequence number, and requesting that the production schedule be forwarded automatically to the mill. Alternatively, the scheduler can manually adjust the Trim Set to account for factors not considered by the trim algorithm. Lastly, the scheduler can discard the Trim Set altogether. In this case, the scheduler either modifies the original selection criteria or compiles an entirely new set of criteria to construct a new Trim Input and begin the trim cycle again. Note, once more, the critical role of human involvement in BOARDS supported decision-making. The scheduler is better prepared than a totally automated process to deal with day-to-day peculiarities and priorities. Therefore, the Trim Set is never automatically converted into a Mill Order.

It may require several iterations before the scheduler arrives at a satisfactory production schedule. Although the objectives are well defined, it is technically infeasible to completely automate the scheduling process. BOARDS provides utility in that it enables several input combinations to be analyzed in the form of production schedules. The limiting factor is time. To examine every possible set of input combinations is tedious, time consuming, and costly. BOARDS facilitates this complicated decision-making process by enabling more alternatives to be considered through functions that are efficient, fast, and easy to use.

## CONCLUSION

BOARDS provides decision support in all phases of containerboard logistics management: operational control, management control, and strategic planning. BOARDS is distinguished from conventional management information systems in that it integrates automated processes of management science and operations research with routine data processing and human decision-making. Advanced technologies, in the form of data base management and distributed on-line user access, are brought together to form a system that satisfies the long term information processing needs of Weyerhaeuser's containerboard logistics management. Its goal is to control containerboard inventories and to improve inventory distribution by supporting demand-based production planning in an industry highly sensitive to market conditions and economic circumstances.

## REFERENCES

Anthony, R. N. Planning and Control Systems: A Framework for Analysis, Harvard University Graduate School of Business Administration, Boston, 1965.

Donovan, John J. and Madnick, Stuart E. "Institutional and Ad-hoc Decision Support Systems," Data Base, Vol 8, No. 3, Winter, 1977.

Gorry, A. and Scott Morton, M.S. "A Framework for Management Information Systems," Sloan Management Review, Fall, 1971.

Keen, Peter G. W. and Scott Morton, M.S. Decision Support Systems: An Organizational Perspective, Addison Wessley, Boston, 1978.

Simon, H. A. The New Science of Management Decision, Harper & Row, New York, 1960.

## CORPORATE MODELING DESIGN & IMPLEMENTATION OF FINANCIAL PLANNING SYSTEMS FOR THE HP 3000

FORESIGHT is a computerized financial analysis, planning and modeling language that addresses the numerous problems faced by managers. It was the first User-oriented interactive financial planning language to be placed on a computer system. It has been in use for over a decade and has been continually upgraded and modified in responce to changing business trends and computer technology.

FORESIGHT is a part of United Computing Systems, Inc., Business Information Products and is backed by its parent company, United Telecommunications, Inc. with assets of $3.5 Billion. As a consulting oriented firm, UCS-BIP has extensive technical, training and consulting staff to assist the user through numerous regional offices - Atlanta, Houston, Kansas City, Los Angeles, Minneapolis, New York City, San Diego and San Francisco.

FORESIGHT's command vocabulary is based on everyday business language. Data entry is simple and previous computer experience is not required. Your attention is focused on the report, not the computer.

In addition, FORESIGHT can be used at any origanizational level, from the smallest cost center to the largest multi-national corporations. It was developed for business people by business people. It readily adopts to your particular requirements and can accommodate each departments unique operation. Management reports or financial models can be easily adjusted to reflect changing business conditions or organizational evolutions.

FORESIGHT can be purchased or leased for your HP 3000.

Planning for the future, whether that future is the next quarter, year or 5 or 10 year period, is a basic concern of every business manager. Management has to make decisions on corporate analyses, planning and control. All within limited time periods. And - not only is the manager expectedto have access to the required information, but he is expected to be able to report on it, and present it in an understandable form.

Management has need to test assumptions in relation to planning, control or financial models without risking capital or resources.

The ability to explore alternatives, to ask "What if?" is more important today than ever before.

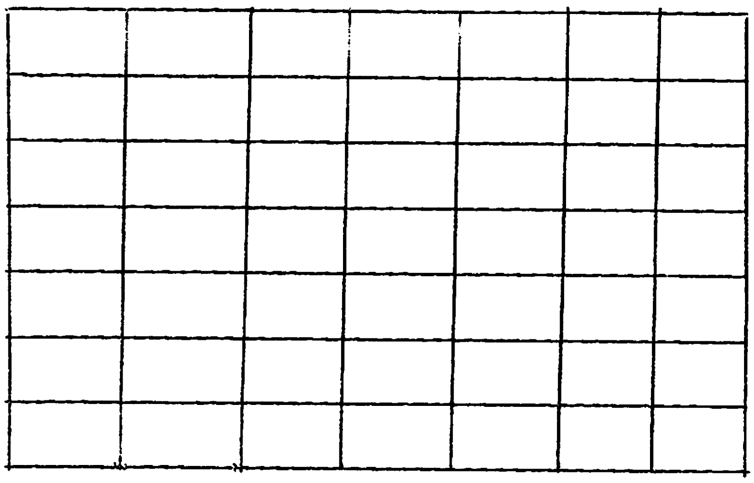
-What if my sales should increase by 15% ?

-What if material costs should rise 5% ?

-What if I should have to make a substantial capital outlay in the third quarter?

Before we look at an example, let's take a minute to understand how information is handled using Foresight. Foresight is based on the concept of a matrix. A matrix being made by the intersection of a series of lines and columns (both variable). The intersection of which houses a data element (either directly input of calculated). The intersection of a line and column we call a cell.

Specifications - 5500 cells
100 columns
1000 lines



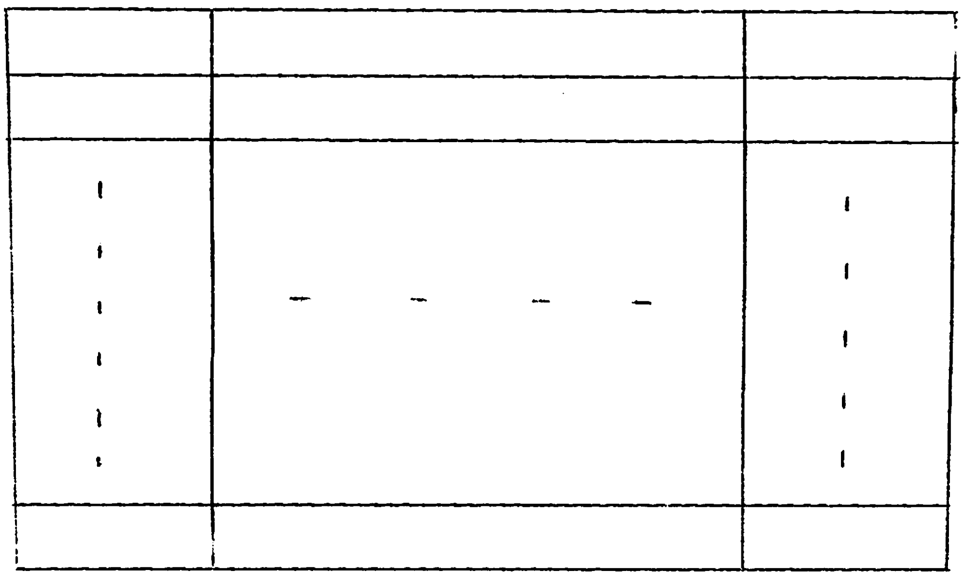
To this matrix, a number of descriptive fields are added:

January 1, 1977 Eastern Division

1977 Planned Product Sales
Cost of Goods Manufactured
Profit and Loss Statement

Month end
January
1977

Total
Fiscal Year
1977



Let's look at an example which will illustrate some of these needs. To help with the illustration, we will look at the Willett Manufacturing Company which has two product lines (we'll call them A and B). Willett markets these products to both the commercial and residential housing markets. With the exception of sales which is segmented into two geographical divisions, all functions are centralized within the company (attached).

The three individuals involved in this planning cycle are the:

- Director of Marketing
- Manager of Production
- Director of Finance

As a marketing driven company, the director of marketing has developed a sales forecast which is submitted on a monthly basis for the period January to December.

Eastern Region

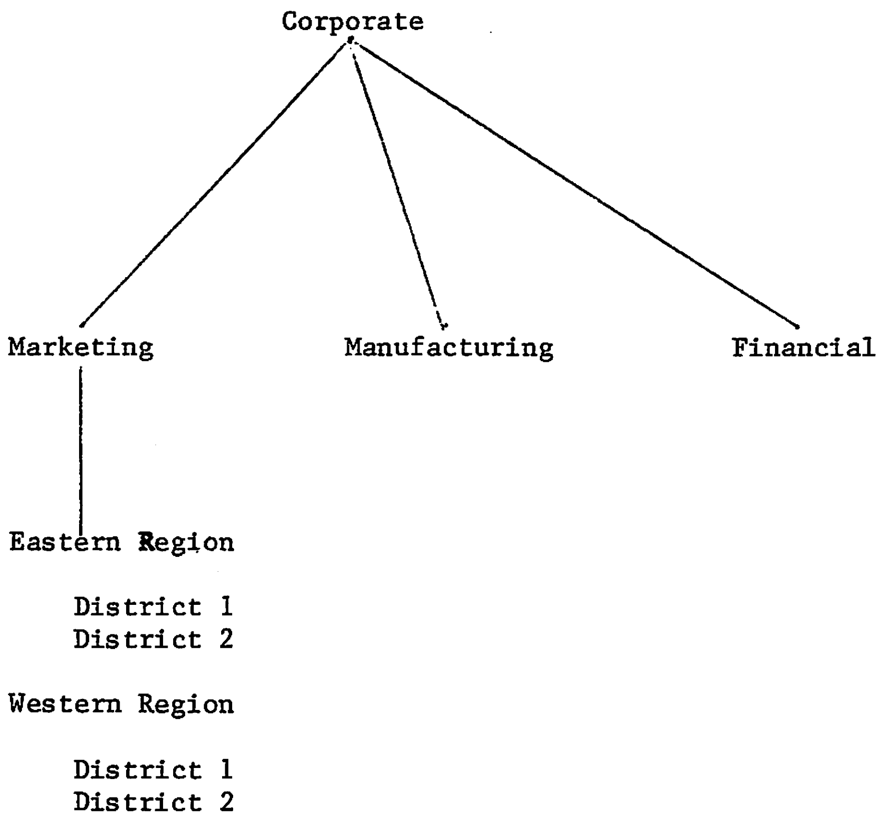
District 1

| | | | | | | |
|---|---|---|---|---|---|---|
| Product A | 135 | 175 | 225 | 265 | 310 | 310 |
| | 265 | 225 | 225 | 225 | 175 | 135 |
| Product B | 105 | 95 | 93 | 95 | 115 | 125 |
| | 135 | 135 | 135 | 120 | 115 | 155 |

District 2

| | | | | | | |
|---|---|---|---|---|---|---|
| Product A | 180 | 180 | 180 | 180 | 180 | 180 |
| | 180 | 180 | 180 | 180 | 180 | 180 |
| Product B | Incrementing by 5% per month based on a 1976 year end average of 100 units per month. | | | | | |

WILLETTS MANUFACTURING COMPANY



In addition, marketing has determined the price per unit for product A will be \$1475 per unit; for product B - \$1700 per unit with a price increase to \$1800 per unit starting in July.

The first statement we need to look at is planned product sales (Exhibit 1). We have taken input from the Director of Marketing and produced a report based on the various line items and calculations he wished to see.

Based on these results the Manager of Manufacturing determined the necessary lead time for production. The Director of Finance concurred for the indirect cost and overhead. From this input, a cost of goods manufactured report was produced (Exhibit 2).

At the Finance Director's level, most of the detailed information is not required. So his profit and loss report is produced (Exhibit 2) using only those totals he wishes to see and including additional corporate level items of interest (Depreciation, General and Administrative expenses, Bonuses).

Now that we have a full set of basic reports for the Eastern Division, what about the Western Division? As both divisions have the same organizational structure, the only change from the Eastern division is in the sales estimates -

District 1

| | | | | | | |
|---|---|---|---|---|---|---|
| Product A | 160 | 210 | 270 | 320 | 370 | 370 |
| | 320 | 270 | 270 | 270 | 210 | 160 |
| Product B | 100 | 90 | 90 | 90 | 105 | 120 |
| | 130 | 125 | 130 | 110 | 110 | 145 |

District 2

| | | | | | | |
|---|---|---|---|---|---|---|
| Product A | 190 | 190 | 195 | 195 | 195 | 225 |
| | 235 | 275 | 275 | 260 | 260 | 250 |
| Product B | Incrementing by 10% per month based on a 1976 year end average of 100 units per month. | | | | | |

A similar set of reports is produced (Exhibits 4-8).

From these two sets of divisional outputs, a series of higher level reports can be easily attained.By the use of the Foresight conso.lidate and merge command a consolidated profit and loss in statement produced (Exhibit 9).

At higher corporate levels a variety of additional reports may be required using the existing data base of information. A comparative profit and loss statement is produced using the consolidate and select command (Exhibit 10). At each higher level, additional logic an be added to specified results extracted from the original data base utilizing Foresight's format file capability.

For the final presentation to the President of the Willett Manufacturing Company, the Finance Director decided to use Foresight's andvanced report writer capability to produce a more finished report (Exhibit 11).

PRESENTATION TITLE: IDIMS - HP3000 Based Digital Image Processing

INDIVIDUAL(S) NAME(S): Paul Polk

ADDRESS: ESL Inc.
P.O. Box 6000
Sunnyvale, CA 94086

ABSTRACT:

The field of digital image processing has grown dramatically in the last six years, largely due to the increasing use of NASA's LANDSTAT satellite for resource inventories, mineral and petroleum exploration, and urban planning use. This presentation will describe ESL's HP3000 based image processing system - IDIMS; some of the above mentioned applications areas ESL has been involved with; and the special purpose peripherals ESL has interfaced to the HP3000. The special peripherals include an interactive color display, an array processor, 300 Mbyte disks, electrostatic printer/plotters, and high speed, high density tape drives (1600/6250 bpi, 75 ips).

# ECONOMETRIC MODELLING ON AN HP 3000

John Eaton
London Business School

## INTRODUCTION

Econometric modelling, particularly of national economies, has become something of a growth industry during the 1970s. The current publically available models of national economies grew out of research projects in universities and other research - based organizations in the late 1960s and early 1970s. This development was made possible by several factors, of which one of undoubted significance was the advent of the electronic computer in the social sciences. For econometric modellers, computers removed the tedium and effort in storing and manipulating the data required, in estimating the relationships that make up the model, and in solving the model for forecasting and simulation purposes.

This paper discussed the implementation of a large econometric model of the UK economy on the HP 3000 system at the London Business School. Though the discussion in this paper is entirely in terms of modelling national economies, modelling techniques such as these are finding increasing application in many large organizations. Thus many large corporations maintain econometric models of the market environment in which they operate, in most cases taking as input the outputs from the large national models of the type discussed in this paper. The purpose of such models from the corporations' view is to enable them to evaluate alternative strategies in the simulated environment, with respect to factors determined by their decisions, and those determined externally by government or their competitors.

The next section provides a general introduction to econometric modelling, and the one following introduces the LBS model of the UK economy. The remaining three sections then discuss the implementation of this model on the HP 3000, in terms of, an on-line database of macro-economic variables;the model estimation process; and the model solution process for forecasting and simulation purposes.

## ECONOMETRIC MODELLING OF A NATIONAL ECONOMY

Econometric models, like any other mathematical models, are intended as mathematical repreentations of the significant characteristics of the system being modelled. In the case of a model of a national economy, the relationships that go to make up the model represent the flows or levels of real and financial resources within the economy, and between it and other economies. The relationships may be of three types. Accounting identities, which are of little technical interest, but which are necessary to complete the system; technical relationships, which usually represent institutional or administrative activites (for example, computation of tax yield, given information as to rates, income groups, etc); and finally, behavioural equations, which represent the behaviour of some specific economic activity (for example, the level of imports of fuel, or the rate of inflation). In general, it is these behavioural equations which require the modellers time, effort and skill, though in the majority of current models they comprise less than half of the equations in the model.

Within each behavioural equation in the model the variables appearing on the right-hand side of the equation (the independent or explanatory variables) determine the systematic variation in the variable on the left-hand side of the equation (the dependent variable). It is axiomatic that in a behavioural equation there will be some residual, random variation in the dependent variable which cannot be explained by any combination of systematic variables which have a meaningful economic interpretation. Thus an econometric model is composed of stochastic, statistical relationships; it is not an exact model (though for forecasting and simulation purposes it is usually treated as though it were).

A variable which appears as the dependent variable of one equation may well appear as an independent variable in another equation in the model. Econometric models are usually therefore systems of simultaneous equations, though often they may be broken down into one or more blocks of simultaneous and recursive equations.

Furthermore, it is highly likely that some of the equations will be non-linear in the variables, so that for forecasting and simulation purposes we are faced with the solution of a system of non-linear simultaneous equations. It is impossible to model the entire economic environment; some factors are not modelable (for example, determination of government policy variables), or are infeasible for modelling by an individual group (for example, the world economy, though I will return to this example later). The basic aim is to model those factors which are significant in terms of the movement or management of the national economy. In order to be able to solve the model for future periods, we must have values of these variables determined outside the model, and of the parameters of the equations that go to make up the model. The variables whose values are determined by solution of the model are known as variables endogenous to the system. Those variables whose value must be known in order that the system may be solved are known as exogenous. The exogenous variables may be further sub-divided into those that are purely exogenous (typically government policy variables), and those, known as predetermined variables, which are defined as equal to the value of endogenous variables in periods previous to that for which the model is currently being solved.

The original, and still the major purpose in constructing models of a national economy is to facilitate the evaluation of alternative government policies for the management of the economy. Forecasting initially entered the picture as a means of model validation; that is, to demonstrate that the model is capable of accurately replicating the systematic behaviour of the real economy that it purports to represent. In an on-going context, forecasting has a crucial role to play in keeping the model on track. But the rationale for econometric modelling as opposed to other forecasting methods is that it enables us to consider and assess alternative government policies on a consistent, rigourous basis. Thus, although the public face of the major modelling groups appears in the regular forecasts produced by their models, in practice, much more effort and interest will be found in the alternative strategies for government policy that are evaluated.

Building a model in the first place is relatively simple as compared with the challenge of keeping it operational on an on-going basis, learning from it, and developing it as the economy itself and economists' understanding of it change. In an important sense, modelling a national economy can be viewed as a continuous exercise in learning by doing. And it is an important part of the make-up of a modeller that he is able to learn from the environment he is modelling, and adapt his perception of the workings of the economy and his model of it accordingly. Some would argue that there was a time at the beginning of the 1970's when econometric modellers were in danger of being swept away by their use of computer technology, and of becoming over confident in the capabilities of themselves and their models. However, the oil-crisis and resultant world inflation provided a salutory shock to modellers everywhere, not only about the tru capabilities of their models, but also about some of the perhaps overlooked characteristics of the environment they were attempting to model.

## THE LBS MODEL OF THE UK ECONOMY

The parentage of the LBS model goes back to the mid-1950's, when one of the leading American modellers, Professor Lawrence Klien of the University of Pennsylvania, visited Oxford University. Klien gathered together a small team of British graduate students, with whom he constructed the first econometric model of the UK economy, using annual data. Valuable though this first model was as a research exercise, possibly its greatest contribution lay in high-lighting what needed to be done in order to construct a fully operational model of the UK economy. One of Klien's team on this first model was Jim Ball, who in 1965, was appointed Professor of Economics at the then newly founded London Business School (in 1971 becoming Principal of the School). Since that first exercise, he had focussed his research efforts on assembling the basic building blocks for an operational UK model based on quarterly data.

Shortly after joining the LBS he assembled a new model, from which the first forecasts were published towards the end of 1966. Since this time the LBS model has been continuously maintained and developed, and has been used to produce forecasts of the UK economy on a regular basis. The project received a major boost in 1967, when it was funded for the first time by the Social Science Research Council, whose generous support has continued to the present day. In 1976 a consortium of twelve major UK organizations was formed to support the Centre for Economic Forecasting at the LBS, which took over responsibility for the model, with Terry Burns as its first director. In addition to participation in forecast meetings with members of the staff of the Centre, members of the consortium also gained access to the data base, estimation programs and forecasting system maintained by the Centre. Thus, in addition to the ten full-time and four part-time staff of the Centre, who are engaged in maintaining the data-bank and the model, production of the basic control forecasts, and a range of research studies using the model, there is also a group of practising economists and forecasters making use of the system within their own organizations.

The LBS model from which the first forecasts were published in 1966, comprised only 25 equations; the current model is made up of over 300 equations. Although this is not as large as some of the models of the US economy, size is not necessarily a criteria of excellence. An operational model of a national economy does have to grow to a considerable size, simply in order to be able to adequately represent the detail of government policy options (i.e. detailed tax rates, expenditure patterns, financial behaviour, and so on). But increasing size brings with it problems of comprehension and management of the model, as the larger the model, the more unlikely it is that any single person is able to understand all the interactions of the complete model.

The computer system requirements of an econometric model can be considered in three parts. A data base containing the historical data from which the relationships in the model are to be estimated, and two software packages, a statistical estimation program and a model solution/simulation program. Originally, the LBS system was implemented on a large-scale batch-processing machine (an IBM 360/65, located at University College, London University), and accessed by means of an rje terminal. However, this did not provide ideal support for either modellers or forecasters. In addition to the rje link to the 360/65, the LBS has operated an HP 2000 timesharing system since 1972 primarily providing a student computing service. In the mid-1970s a simple forecasting system was implemented on this system, with two objectives in mind. First, to examine to what extent on-line access and interactive processing might better support the modellers and forecasters needs. And second, to aid in the determination of the true computer system needs of the modelling and forecasting system. Nevertheless, although it was clear that the system was more supportive of the modellers and forecasters needs than a batch system, the HP 2000 could not cope with the workload. In 1977, the LBS installed an HP 3000 Series II to provide computing services to all its research activities, including the Centre for Economic Forecasting. The entire modelling and forecasting system has now been implemented on the 3000, exploiting the on-line interactive capabilities of the system.

## THE MACRO-ECONOMIC DATA BANK

The macro-economic data base provides modellers and forecasters with access to the basic raw material for their studies; the time-series recording the detailed movements in the national economy. At its simplest such an on-line data-bank is a source of data for information purposes, providing the answers to queries as to the level or movement in particular variables, or the input for tabular or graphic reports. However, its crucial role is to provide the raw material for the modelling and forecasting processes, for which it must be interfaced with the estimation and solution/simulation systems.

Three macro-economic data-banks are implemented on the HP 3000 at the LBS the major one being that containing quarterly data from the mid-1950s to date on slightly more than 1000 economic variables relating to both the UK and world economy. In addition there are smaller data-banks containing annual and monthly data respectively on the major variables. These data banks are structured as KSAM files with a standard structure applied to other data-banks at the LBS, in particular, a data bank containing stock market data on all UK quoted companies since the mid-1950s. The detailed structure of the LBS KSAM data banks is set out in Appendix A.

The main data bank is updated on-line as new data is released by government and other agencies. Unfortunately, in the UK the government's statistical service is unable to provide data immediately in machine-readable form (only about a month late), so that the majority of the data is entered manually from press releases. A small suite of programs has been developed (in Fortran, using the KSAM intrinsics) to carry out the standard data management functions on-line or in batch as appropriate. Thus, functions to enter update, modify, and list specific variables may be carried out on-line; whilst functions such as to copy, list or transform the entire data bank are carried out as batch jobs. The variables that are used in the equations of an econometric model are rarely published in that form. Instead they are formed as functions of published variables, in some cases very simple ones (such as products or ratios), in other cases much more complex. However, these functions are programmable, so that the updating of a particular published variable, may in turn lead to the automatic updating of several other variables in the data bank which are specified functions of the original variable.

## ESTIMATING THE RELATIONSHIPS OF THE MODEL

As discussed earlier, an econometric model is made up of a large number of economic relationships. The process of initially estimating and then maintaining these relationships is a continuous one, since the economy itself, the modellers and other economists' perception of its behaviour, and the statistical tools available to the modeller are continuously changing, albeit slowly in most instances. Given the size and complexity of the national economy, most successful models have begun as top-down exercises, starting simply in order to find out where a more detailed understanding would be most useful. But ultimately the models grow to a considerable size through the necessity to model in detail the impact of government policy variables, so that alternative policy strategies can be evaluated. In general, the modeller will begin with relatively simple relationships, and then incrementally introduce more detail and complexity in order to try to cope with its observed inadequacies; it is a classical example of a man-machine interactive system.

Current economic theory will provide the modeller with initial ideas as to which variables might enter the relationship, but gives much less guidance as to the precise functional form of the relationship, and, in particular, to the distribution of the effects of the independent variables over time. There are relatively few instances where it is believed that the effect of a change in an independent variable on the dependent variable in an equation is totally complete within the current time period. It is usually suggested that there will be carry-over effects over several time periods, and indeed, one of the major reasons for constructing econometric models is to examine in detail the time path of the response of the economy to changes in government policy variables. Thus, for example, the effects on consumption patterns of a change in the rate of a sales tax will not be seen solely in the period in which the change is made, but will be distributed over several subsequent time periods (hence the technical name of distributed lags to refer to this generic problem). Clearly, the significance and magnitude of these carry-over effects dependents on the unit time period of the data used in the model. Thus, if the relationships in the model were estimated using annual data, then we would expect that there would be only relatively small carry-over effects, as compared with a model estimated using monthly data. Most macro-econometric models of national economies are estimated using quarterly data.

The basic model-building process is dependent on a high degree of man-machine interaction, which is provided much more successfully on the HP 3000 than on the previous batch system. The model-building, estimation process is both iterative and highly suitable for interactive processing. It is iterative in that the modeller will begin with his initial relationship, which he will then refine, develop and evaluate by estimating many more equations, the precise format of each successive equation being heavily influenced by the results obtained from the estimation of previous equations. It is interactive in that interspersed with the actual estimation of the parameters of an equation, the modeller is likely to manipulate the variables in a variety of ways, and to want varying amounts of detail presented to him. An interactive econometric estimation program (ISEA, Interactive Software for Econometric Analysis) has been implemented on the HP 3000 to provide powerful and detailed support to the modeller. ISEA interfaces with the LBS KSAM data banks to allow the modeller to interactively load data for analysis, to which a wide range of specific data transformations may be applied prior to the estimation process. The relationships in an econometric model are typically estimated using classical regression techniques, which the modeller can apply using ISEA, with interactive control over the specification of the model to be estimated (including specific options tailored to econometric applications, such as estimation subject to polynomial (Almon) distributed lags, first-order autoregressive error terms, linear restrictions on the coefficients, and so on), and over the content and volume of output from the estimation process presented at the terminal. Optionally, large volumes of output may be directed to a lineprinter. It is also possible to interactively estimate the parameters of a Box-Jenkins ARIMA time-series model using ISEA. The objective in the design of ISEA has been to enable the modeller as far as possible to maintain a continuous meaningful man-machine dialogue, in terms that the econometrician can readily comprehend. A brief description of the capabilities of ISEA is given in Appendix B.

## MODEL SOLUTION AND SIMULATION

As indicated earlier, in general econometric models will comprise a system of non-linear simultaneous equations, which have to be solved for the endogenous variables in each forecast or simulation period. In order to be able to solve the model in the future, values will be required for those variables which are purely exogenous to the model (values for the predetermined variables will either be the appropriate historical value of the relevant endogenous variable, or its forecast value for a previous period). There are commonly two categories of variables which are likely to be purely exogenous to a model: those whose value is determined either directly or indirectly by the government, central bank, or some other economic regulatory agency; and those measuring the overall economic environment, such as world output, trade, prices and so on. The major models for each country now tend to explicitly model those variables which directly influence their interaction with the economies of other countries (typically, domestic prices, trade prices, money supply, interest and exchange rates). Indeed, there is a project in existence (project LINK, based with Klien at the University of Pennsylvania, and funded by the IMF, NSF and other agencies), to put together on a consistent basis econometric models for individual countries and major regional groupings, and to solve them simultaneously to produce consistent forecasts of the current and capital account trade items, output, inflation interest and exchange rates for the world as a whole. This system is now operational, and the LBS model provides the UK model for the project.

Prior to commencing a forecast or simulation exercise with an econometric model, it is standard practice to adjust it, if necessary, for a variety of potential disturbance factors. First, the individual equations are examined in order to see if the estimated values of the error term (the residuals) over the recent past exhibit any significant systematic behaviour. Although according to statistical theory the expected value of the error term is zero, in practice it is often the case that for a variety of reasons examination of any specific period will demonstrate a significant discrepancy from this expected value. It may well be thought that any observed systematic pattern will continue into the future forecast period, and should be explicitly allowed for. Second, even in the best model, there are some systematic factors whose influence is inadequately accounted for by the structure of the model, but concerning which some reasonable ad-hoc estimate may be made for the future. Finally, it is often the case that the modeller has prior knowledge of some significant event that will occur in the forecast period, but which is not explicitly included in the model. For any or all of these reasons it is usual for the modeller to make adjustments to the structure of the model in the forecast period. This is done by making adjustments to the intercept or constant term of the behavioural equation determining the value of the endogenous variable that it is desired to adjust (hence they are known as constant adjustments). Values for any non-zero constant adjustments must then be specified before forecasting can commence. In practice there tends to be more uncertainty over the appropriate values for the constant adjustments since they are more influenced by the subjective assessment of the modeller.

Given values for the exogenous variables and the constant adjustments for the forecast period, then the model may be solved for the values of the endogenous variables for each forecast period. As the model is usually a system of non-linear simultaneous equations it is not possible to compute an analytic solution to the model, and it is necessary to adopt an iterative numerical procedure, starting with an initial estimate of the solution for the forecast period, then hopefully converging to a solution. In general, the solution of systems of non-linear simultaneous equations can pose considerable problems in terms of successful convergance to a consistent solution. However, these problems do not arise in practice in the solution of econometric models, for two reasons. First, most econometric models are only mildly non-linear, with only a few exponential equations being the norm. Second, it is always the case with econometric models that we possess a good initial estimate to the solution for the current period with which to begin the iterative sequence (at worst, we can use the solution for the previous time period; in practice, we often have an existing solution for the present period from a previous forecast run, with only slightly different values for exogenous variables or constant adjustments). Thus it is a general experience of econometric modellers that the computationally simple Gauss-Siedel algorithm will always generate a solution to such a model, normally in less than 20 iterations per period.

Forecasting, or simulating alternative policy strategies, is a highly iterative process. The forecaster will set his initial estimates of the exogenous forecasts and constant adjustments, compute the forecast, evaluate it in some way, modify the exogenous forecasts and/or constant adjustments, compute another forecast, and so on. Because of the simultaneous nature of the models being used, it is normal practice to only make one or two changes each forecast run, so that their effect can be clearly seen (if many changes were made in one run it would not be possible to infer the effects of an individual change). To some extent this process is open-endedand continuous in that the adequacy or otherwise of a particular forecast is ultimately largely dependent upon the subjective assessment of the forecaster. In the context of the LBS model, major forecasts are published three times a year (in January, May and October), with short monthly ones in the intervening months.

The actual implementation of the Gauss-Siedel algorithm for solving the model is as a batch job, as there is nothing to be gained from the forecaster's point of view in interacting with the algorithm. However, from forecast run to forecast run, the forecaster wishes to make a series of small usually incremental changes to the exogenous forecasts and/or constant adjustments. A set of initial exogenous forecasts and constant adjustments is agreed by the staff of the Centre as a 'control' set; the forecaster then creates an Editor file on-line which records his changes to the basic control files (which are structured as KSAM files). He then 'streams' a job which solves the model, using the control exogenous forecasts and constant adjustments modified by the contents of his Editor file. The options available in the processing of the Editor file not only include simple one-for-one changes in value, but also more complex operations such as incrementing all values from a certain date by a constant absolute or percentage factor. The forecast values for the endogenous variables are placed in a KSAM file for future reference; in theory, the forecaster could interrogate this file on-line to see if it was worth printing full forecast details. In practice, the amount of data that most forecasters want to see before they are able to make such a decision is rather large, so the usual option is to print a standard set of national accounts tables on the lineprinter. A typical five year (20 period) forecast takes about 100 cpu seconds on the HP 3000.

In addition to the standard forecasting and simulation operations, the solution system is also used for more research-orientated applications to determine characteristics of the system as a whole. Typical applications of this type would be, solution of the model for within-sample periods to determine the inherent error characteristics; computation of the dynamic multipliers for the major government policy variables, both singly and in combination (the multipliers give the proportionate change in endogenous variables for an appropriate unit change in one or more policy variables, all else being held constant); and computation of stochastic rather than deterministic forecasts, in order to evaluate the likely error bounds for the forecast (this involves setting the error terms in the model to a random value drawn from an appropriate probability distribution). Further, a major exercise is now underway to attempt to compute optimal policy strategies given some desired pattern of activity in the economy for future periods, using optimal control techniques drawn from control engineering.

## APPENDIX A

KSAM file structure used at LBS for time-series data-bases:

| Bytes | Description | Variable type |
|---|---|---|
| 1-8 | Variable code | character*8 |
| 9-44 | Variable title | character*72 |
| 45-46 | Frequency of measurement per year | integer |
| 47-50 | Start date | integer*4 |
| 51-54 | End date | integer*4 |
| 55-56 | Number of observations | integer |
| 57-62 | Variable type | character*6 |
| 63-64 | Code number | integer |
| 65-66 | Lag length | integer |
| 67-68 | Transformation number | integer |
| 69-70 | 1st transformation parameter | integer |
| 71-72 | 2nd transformation parameter | integer |
| 73-76 | 3rd transformation parameter | real |
| 77-80 | Not used at present | |
| 81-800 | Up to 180 observations (or multiple thereof) | real |

## APPENDIX B

ISEA commands (version 6.4, 2 October 1978):

| | | |
|---|---|---|
| AUTO | Computes autocorrelation coefficients | |
| BOXJ: | Computes the parameters of a seasonal Box-Jenkins ARIMA model: | |
| | Output options: | |
| | AUTO: | autocorrelation coefficients of residuals |
| | PAUT: | partial autocorrelation coefficients of residuals |
| | PRES: | print actual, fitted, residuals and percent error |
| | SRES: | save, in the ISEA data matrix, the fitted, ratio of actual/fitted, or the fitted + and - one or two standard errors |
| | GRAF: | plots residuals or actual and fitted at terminal |
| | VCOV: | variance-covariance matrix of estimated coefficients |
| | BCOR: | correlation matrix of the estimated coefficients |
| | FORE: | compute a forecast, and (optionally) save in the ISEA data matrix |
| | MODL: | prints standard output on the lineprinter |
| CORR: | Computes simple correlation coefficients | |
| DATA: | Enters data into the ISEA data matrix, | |
| | T: | from the terminal |
| | K: | from an LBS KSAM file |
| | F: | from a file 'kept' by ISEA |
| | E: | from an Editor file |
| DIM: | Re-dimensions the ISEA data matrix (the default is 300 observations on 20 variables; this may be increased to 500 observations and 50 variables, so long as the product of the two does not exceed 8000). | |
| EDIT: | Edits the data in the ISEA data-matrix: | |
| | ACOL: | adds a column |
| | CCOL: | changes a column |
| | AROW: | adds a row |
| | CROW: | changes a row |
| | COBS: | changes an observation |
| | AFOR: | adds date from a CEF KSAM forecast file |

| | | |
|---|---|---|
| GRAF: | Plots a variable at the terminal, either across or down the page. | |
| KEEP: | Keeps the ISEA data matrix in a private sequential file | |
| LPNT: | Prints the contents of the ISEA data matrix on the lineprinter | |
| MEAN: | Computes means and standard deviations of variables | |
| MISS: | Specifies a value to indicate missing observations | |
| NAME: | Prints the names of the variables currently in the ISEA data matrix | |
| PAUT: | Computes the partial-autocorrelation coefficients | |
| PLOT: | Plots up to three series on an HP 7203 graph-plotter | |
| PRNT: | Prints the contents of the ISEA data matrix | |
| REG: | Specifies a regression equation (default is OLS), options: | |
| | OLS: | Ordinary Least Squares |
| | TSLS: | Two-stage Least Squares |
| | RSLS: | OLS subject to linear restrictions |
| | ALMN: | Almon distributed lags |
| | AUTO: | Include 1st order-autoregressive term |
| | NCON: | Supress constant term |
| | LOG: | Print true statistics for a log equation |
| | WGHT: | Weighted least squares |
| | DSCT: | Discounted least squares |
| | STRT: | Start date for sub-sample |
| | END: | End date for sub-sample |
| | Output options: | |
| | AUTO: | autocorrelation coefficients of residuals |
| | PAUT: | partial autocorrelation coefficients of residuals |
| | PRES: | print actual, fitted, residuals and percent error |
| | SRES: | save, in the ISEA data matrix, the fitted, ratio of actual/fitted, or the fitted + and - one or two standard errors |
| | GRAF: | plots residuals or actual and fitted at terminal |
| | VCOV: | variance-covariance matrix of estimated coefficients |
| | BCOR: | correlation matrix of the estimated coefficients |
| | FORE: | compute a forecast, and (optionally) save in the ISEA data matrix |
| | MODL: | prints standard output on the lineprinter |

|  |  |
|---|---|
| SMTH: | Exponential smoothing (Winters method) |
| TRAN: | Compute any of the following transformations: |

|  |  |
|---|---|
| LAG: | lag the observations on a variable |
| LOG: | natural logs |
| DIFF: | differencing |
| MSUM: | moving sum |
| MAVE: | moving average |
| ADD: | sum two variables |
| SUBT: | subtract one variable from another |
| MULT: | multiply two variables |
| DIVI: | divide one variable by another |
| RECP: | reciprocal of a variable |
| POWR: | raise to a power |
| EXP: | e to the power of a variable |
| CSUM: | cumulative sum |
| TIME: | time trend |
| MULC: | multiply by a constant |
| ADDC: | add a constant |
| SUBC: | subtract from a constant |
| ABS: | absolute value |
| SUMV: | sum several variables |
| EXS: | simple exponential smoothing |
| DEV: | positive, negative (or both) deviations from zero, mean or a specified value |
| DUMY: | form a dummy variable |
| SDUM: | form seasonal dummy variables |

# CWF/3000: A COMPLETE SYSTEM FOR COMPUTER ASSISTED INSTRUCTION AND TRAINING

HAROLD J. PETERS
EDUCATIONAL SOFTWARE PRODUCTS
9 GEORGETOWN CIRCLE
IOWA CITY, IOWA

## What is CW?

CW stands for "coursewriter", which is an authoring language for computer-assisted instruction (CAI). Figure 1 shows a sample of student interaction with a CAI lesson that was written with CW. While the subject matter can of course vary greatly, this example is typical of the sort of tutorial dialogue that can readily be written with the CW language.

Figure 2 shows the CW code corresponding to the sample student interaction in Figure 1. The two letter "op-codes" are almost self-explanatory; QU: question, CA: correct answer, TY: type, WA: wrong answer, UN: reply for unexpected answer. This example illustrates the principal advantage that CW, or any other CAI authoring language, offers over a general purpose programming language -- that is: implicit branching. If the student's answer does not match the argument of the CA, then processing automatically branches around the TY associated with the CA and on to the next implied comparison, the first WA.

## FIGURE 1

```
Most nouns ending in "ch" form plurals
by adding "es".
For example,
           "church" becomes "churches",
           "lunch" becomes "lunches", etc.
What is the plural of "torch"?
torchs
"torch" ends in "ch".  Please try again.
terches
I think you've spelled "torch" wrong.
torches
Good!  Try another.
```

Figure 1. A sample of student interaction with a CAI course written in CW.

FIGURE 2

```
QU     Most nouns ending in "ch" form plurals
       by adding "es".
       For example,
                  "church" becomes "churches",
                  "lunch" becomes "lunches", etc.
       What is the plural of "torch"?
CA     torches
TY     Good! Try another.
WA     torchs
TY     No, "torch ends in "ch".  Try again.
WA(L)    t&ches
TY       I think you've spelled "torch" wrong.
         Please try again.
UN     No. The answer is "torches".  The ending "es"
       is added for the plural because "torch" ends
       in "ch".  Try another one.
BR     PR
```

Figure 2. CW code for the sample student interaction given in Figure 1.

Again, if there is no match a branch is made automatically to the next implied comparison, etc. If on the other hand, the student's answer does match the argument at one of the implied comparisons, then the corresponding TY (and/or other so-called "minor" op-codes) is executed and all further comparisons are skipped over in an automatic branch to the next question.

All this implicit branching does save a lot of busy work on the part of the author and lets him concentrate on the higher-level structure of the lesson. It clearly does not absolve the author of all programming tasks but it does make his job a lot less tedious.

First appearing in the early 1960's, CW has to be considered one of the authentic pioneering languages for CAI. Many CAI languages have come along since the introduction of the first version of CW. And this leads to legitimate speculation as to why this "old" language still manages to survive. Two reasons appear most prominent. First, CW is extensible: the short-comings in its original design can be circumvented by the use of its user-written function feature. Any programming function within the capabilities of the underlying software system can in principle be invoked by a CW course through use of the user-written function feature. Within a CW course, the function call is simply

```
FN MYPROG
```

where MYPROG is the name of the user-written function to be invoked. More will be said regarding user-written functions in a later section.

The second factor contributing to the longevity of CW is that because of its early existence on the most popular equipment, i.e., IBM, a great deal of courseware has been created in CW over the years, and the easiest way for newcomers to CAI to get off to a running start has been to tap into that large reservoir of courseware by getting a CW system themselves. They then in turn create more courseware and so the bandwagon goes on. It was precisely this second attractive feature of CW -- the large accumulated base of courseware -- that originally drew the interest of Hewlett Packard -- and leads to the next question:

## What is CWF?

CWF stands for Course Writing Facility, Hewlett Packard's name for its emulation in BASIC of IBM's CWIII (the version of CW current in the early 1970's). CWF was originally developed for the HP2000 timesharing system, and some 15-20 CWF systems were sold prior to HP's withdrawing it from the market, at least partly in anticipation of the obsolescence of the HP2000 itself. Essentially all features of CWF in its HP2000 implementation have been retained in the HP3000 version, so the description of those features is deferred to the next section.

## What is CWF/3000?

Building upon its HP2000 predecessor, CWF/3000 provides on the HP3000 essentially all the course authoring, course presentation, and student-keeping capabilities of IBM's Coursewriter III, version 3 (CWIII), plus a significant additional feature <u>not</u> offered by CWIII. In CWIII, the extensibility described earlier is achieved through user-written functions programmed in IBM that such functions can be written directly in BASIC.

CWF/3000 consists of four major subsystems: course authoring, course translation, student usage, and record keeping/reporting, each described in turn below.

Course Authoring

CWF/3000 authors use a set of programs headed by CWEDIT to enter, revise and edit new course material, using the CW language op-codes such as shown earlier in Figure 2. Some thirty op-codes, many of which may be modified in several ways, offer wide flexibility to the author. For example, the CA op-code in simplest form accepts exactly one correct answer:

```
CA torches
```

With an "L" modifier, a variety of answers can be accepted through usage of "don't care" characters. For example,

```
CA(L)  t&ches
```

will accept "torches", "touches" or "txyzches" or any other "word" beginning with "t" and ending with "ches", as correct. Another modifier,"W", allows more specific alternative correct answers. For example,

```
CA(W) torches lamps candles
```

accepts these three and only these three as alternative correct answers. And if all this flexibility is not enough, the author can resort to a user written function, e.g.,

```
FN ANSWER
```

and rely on his own BASIC program, ANSWER, to achieve <u>any</u> type of answer processing he may desire that can be implemented within the broad generality of BASIC. The experienced CWF/3000 author moves flexibly between CW and BASIC, taking advantage of the best features of each as the occasion demands. Very roughly, CW is superior in dialog-intensive work such as in tutorials, and BASIC is superior in computation-intensive tasks such as simulations.

## Course Translation

In principle, any CWIII course developed on IBM equipment can be converted via CWF/3000 conversion programs to CWF/3000 compatible form and then used, revised and edited just like any other CWF/3000 course. In practice, virtually every conversion we have seen attempted has succeeded -- eventually. The biggest problem -- when there have been problems -- arises with courses that make extensive use of special display characteristics. The special display features available on the system on which the course was originally offered may not be available on the target system. And even if they are, they are likely to be implemented differently so that considerable conversion attention may be necessary. But whatever effort may be required, it is almost invariably true that starting with someone else's courseware and modifying it in whatever ways necessary to meet one's own needs remains a far superior method for getting usable courseware up and running than by starting from scratch on one's own. So it remains the case that the principal source of attraction in CWF/3000, as in its predecessors, is the ability it gives the user to most easily tap into a large existing base of CW courseware. The 1978 Index to Computer-Based Learning[1] lists 311 instructional modules developed in CWIII that in principle should be readily convertible to CWF/3000. In addition, the Index lists 720 modules written in BASIC, which are compatible with CWF/3000 in important ways, as described in a later section.

## Student Usage

CWF/3000 incorporates HP's Instructional Management Facility, IMF, to handle student sign-on, sign-off and some of the student record keeping. Hence a student taking a CWF/3000 course enters through the IMF program START in the following standard fashion:

```
RUN START.PUB

What is your ID number and first name? 1000,Jimmy
Is your last name Carter? Yes
Course name?  ENG4
30 September 1978        14:49        Port 7

Welcome, Jimmy to session number 3 of English 4.
In our last session....
```

Whenever the student signs off by typing //STOP or //SIGN OFF, or is signed off automatically by the instructional program, the CWF/3000 system saves his restart information so that he may begin his next session where he left off in the last one, or wherever else the author may wish to designate.

## Record-Keeping/Reporting

In addition to student restart information, as already described, many other types of information relating to student usage of a CW course can be recorded and later reported. Information concerning essentially any aspect of student interaction with the course material can be saved by one means or another.

Perhaps most common is simply keeping track of correct answers in a session, reporting this to the student at sign-off time, and possibly recording the number for reporting to the instructor later, or maintaining a cumulative day-to-day record of scores for benefit of both the instructor and student. Another common type of information saved is the number of times each registered student has accessed a course and how much cumulative time the student has spent on the course.

During the development phase of a course it is especially important to accumulate all <u>unanticipated</u> student answers, i.e., all answers for which no tailored replies had been prepared and for which only the "reply to unanticipated answer" (the argument of the UN op-code) is displayed to the student. Frequently, these unanticipated student answers will suggest that parts of the course need further development work, including perhaps new explanatory passages, or at least a broader array of model answers and corresponding tailored replies. Most of the record keeping and reporting functions that have been mentioned require that students be <u>registered</u> for courses, and that the courses themselves first be entered into the IMF record keeping system. Some examples of instructor or proctor interaction with the IMF programs are shown in Figure 3.

FIGURE 3

A. Entering a course

```
RUN ADMIN.PUB
CODE?MMMMMM
COMMAND?COURSE
  COURSE COMMAND?ENTER
    COURSE NAME?REX
    CODE WORD?R1
    DOES IT HAVE A DEMO MODE?YES
    SPECIAL COURSE TYPE?CW
    REX IS NOW ENTERED AS A COURSE.
    COURSE NAME?//STOP
DONE
```

B. Enrolling a student

```
RUN PUPIL.PUB
CODE?MMMMMM
COMMAND?ENTER
  FIRST NAME?JOE
  LAST NAME?SWARTZ
  ENTERED WITH ID NUMBER 1018.
    COURSE NAME?REX
    GROUP NAME?MS NOEL
    HISTORY FILE OF REX INITIALIZED TO 9 STUDENTS.
    AREA NUMBER?1
    USER GROUP NUMBER?1
    ENROLLMENT COMPLETED.
  FIRST NAME?//STOP
DONE
```

Figure 3. Some examples of instructor or proctor interaction with IMF record keeping programs under CWF/3000.

As a final note regarding recording keeping, we should point out that the CWF/3000 record keeping facilities are available for BASIC language courses as well as CW courses. It is typically the case that computer-based instructional materials written in BASIC have not used _any_ kind of student record keeping functions. Typically instructors have no information as to which students have used the modules nor as to how well they have performed. It is certainly difficult to assess the instructional value of CAI materials in this case. And refinement of the modules, or development of new materials must be based on guesswork rather than hard data. Once again, the record keeping and report facilities of CWF/3000 provide a ready solution to these problems for both BASIC and CW courses.

## Some Technical Considerations

CWF/3000 has not been optimized for the HP3000. As of September, 1978, only the student driver programs have been compiled so that they can run as object code rather than under the BASIC interpreter. This provides good performance for up to 15-20 students running most CW materials.

Some technical problems have delayed compilation of the authoring programs, which means that some authoring functions execute much more slowly than desired.

Beyond simple compilation of the various programs comprising CWF/3000, it is clear that substantial further optimization could be achieved through redesign of the file structures, which still suffer from design constraints imposed by the early

HP2000 series time sharing systems. These and other refinements to CWF/3000 await the indication of further interest in the system by HP3000 users.

## Availability

CWF/3000 has been developed by and is available from (under a license agreement) Educational Software Products. Requests for further information should be directed to the author, at that organization.

# STATISTICAL INQUIRY & RETRIEVAL
an IMAGE application

MARTIN D. JEWEL
NATIONAL SPINAL CORD INJURY DATA RESEARCH CENTER

## BACKGROUND

The National Spinal Cord Injury Data Research Center (NSCIDRC) - a division of Good Samaritan Hospital in Phoenix, Arizona - is supported in part by a grant from the U.S. Department of Health, Education & Welfare through the Rehabilitation Services Administration. NSCIDRC's goal is to provide access to a national repository of data relative to spinal cord injured persons for the purpose of improving the care and treatment thereof, and reducing the length of hospital stay and associated costs.

Since spinal cord injury is a sudden traumatic shock and extremely expensive, the costs are often borne by society in the form of taxes and insurance premiums. Helping a patient to achieve his most productive status as quickly as possible gives him a psychological boost, reduces the drain on family and personal resources, and decreases the cost to society.

## SYSTEM FLOW

The source of patient data is the eleven Regional Model SCI Systems (see appendix 1). Data is extracted from hospital records, physicians' statements, patient interviews and bills for various types of equipment and services. This information is compiled by medical record personnel and transcribed onto pre-printed forms. The forms are assembled into batches upon completion, logged, and then forwarded to Phoenix. Generally, a batch represents a weeks work. (see Figure 1).

After receiving the batched forms at NSCIDRC, the forms are logged in on the HP3000 and sorted by new entries and updates (see Figure 2). The new entry data is keyed into the computer via an HP2645 video terminal using a general purpose data entry program designed to create a transaction file. We do not use DEL, having found it inadequate for our needs.

Processing at the Regional SCI Center

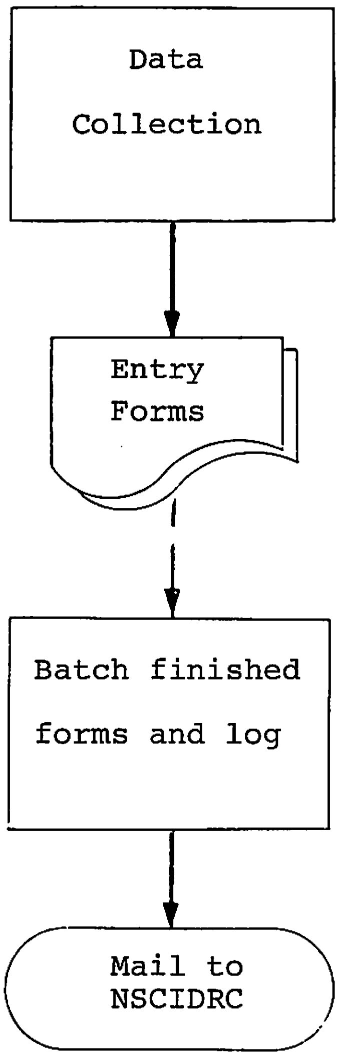


Figure 1.

Processing at NSCIDRC

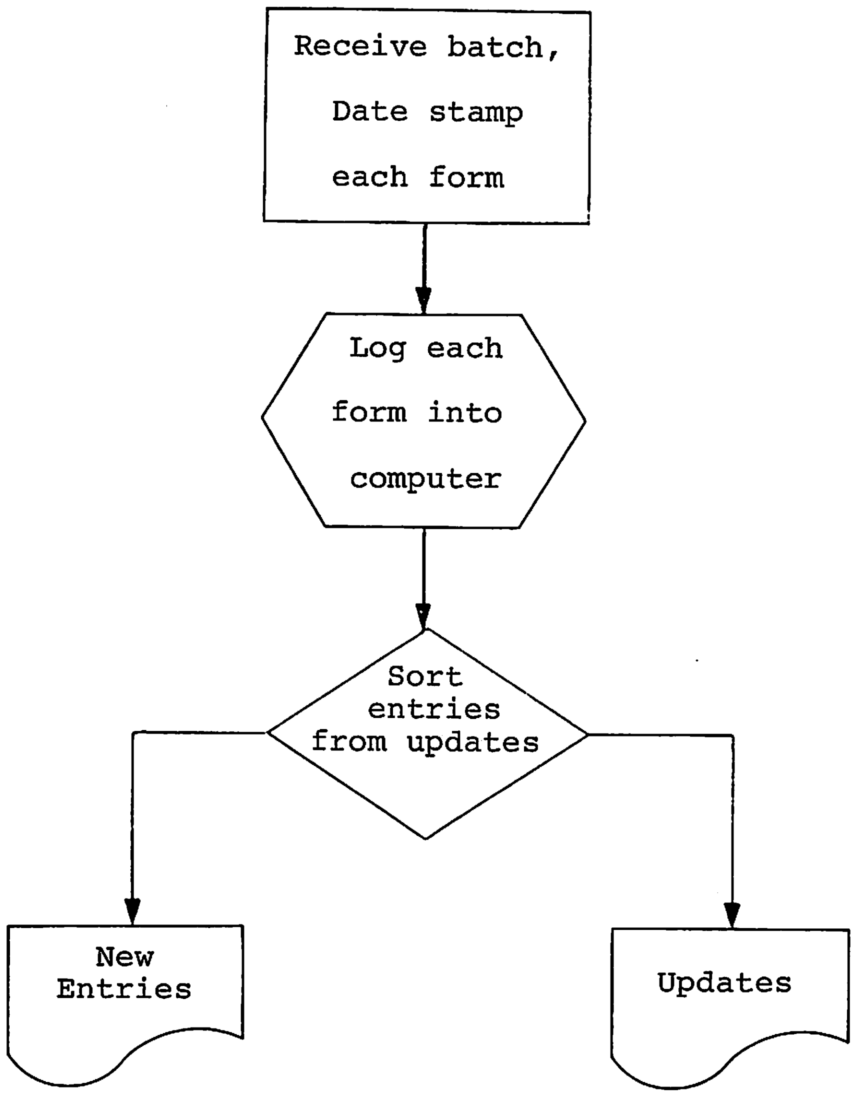


Figure 2.

At appropriate periods, usually twice a day, a data base posting program is executed as a batch job stream to add the data to the data base (see Figure 3). We do not enter data directly to the data base.

Twice monthly, a data quality audit program is run to produce a discrepancy list which is forwarded to the Regional SCI Centers for corrective measures. Resultant updates are then sent to NSCIDRC as described above. The updates are posted to the data base on-line. Data verification is done after entry/posting and updating.

Other data base management tools are shown in Figure 4. The selective dump provides a hardcopy of patient data as it exists in the data base. The Form 2 Follow-up produces a tickler report of those forms which are due during the next quarter, and an expediting list of in-process and past-due forms.

We have grown increasingly confident of the relative cleanliness of the data base. We now average fewer than 2 discrepancies per 100 forms. With the many checks for validity and logical interrelationships between variables, the error rate is approximately 1 in 40,000.

## DATA FLOW TIMING

Patient information comes to NSCIDRC at the completion of the initial hospitalization and rehabilitation period, and again at each subsequent anniversary of injury. There may be a lag time of up to three months while data is extracted from case records and various professionals respond to requests for information. Typically, after three to six months after the end of a calendar year the data for that year is complete, clean and ready to be analyzed.

The form, on which the initial data is submitted, is referred to as Form 1 and is complete in itself. The annual follow-up data is reported on a Form 2. Each Form 2 may have one or more Hospitalization forms (Form H) attached, if the patient was admitted to a hospital during that year. Thus, a particular patient will have a single Form 1, and, depending on the number of years after injury, one or more Form 2's. Each Form 2 may, in turn, have one or more Form H's attached.

## DATA BASE MANAGEMENT BACKGROUND

Initially, data was stored on an IBM System/32 which had many hardware and software limitations. It was limited to producing RPG reports and had no data base capabilities. In addition, the volume of data was not sufficient for

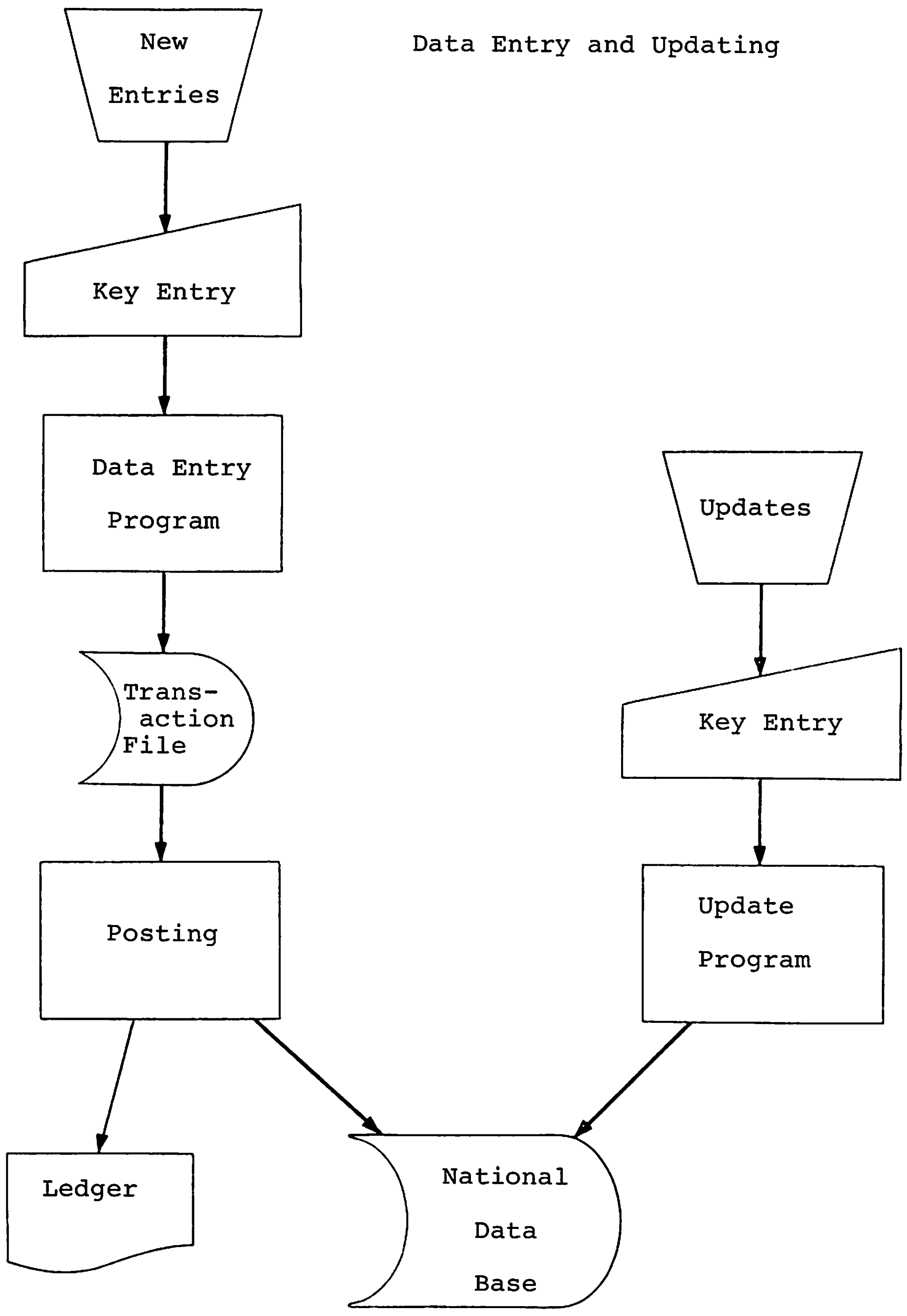

New Entries
Data Entry and Updating
Key Entry
Data Entry Program
Trans-action File
Posting
Ledger
Updates
Key Entry
Update Program
National Data Base

Figure 3.

Data Base Management Tools

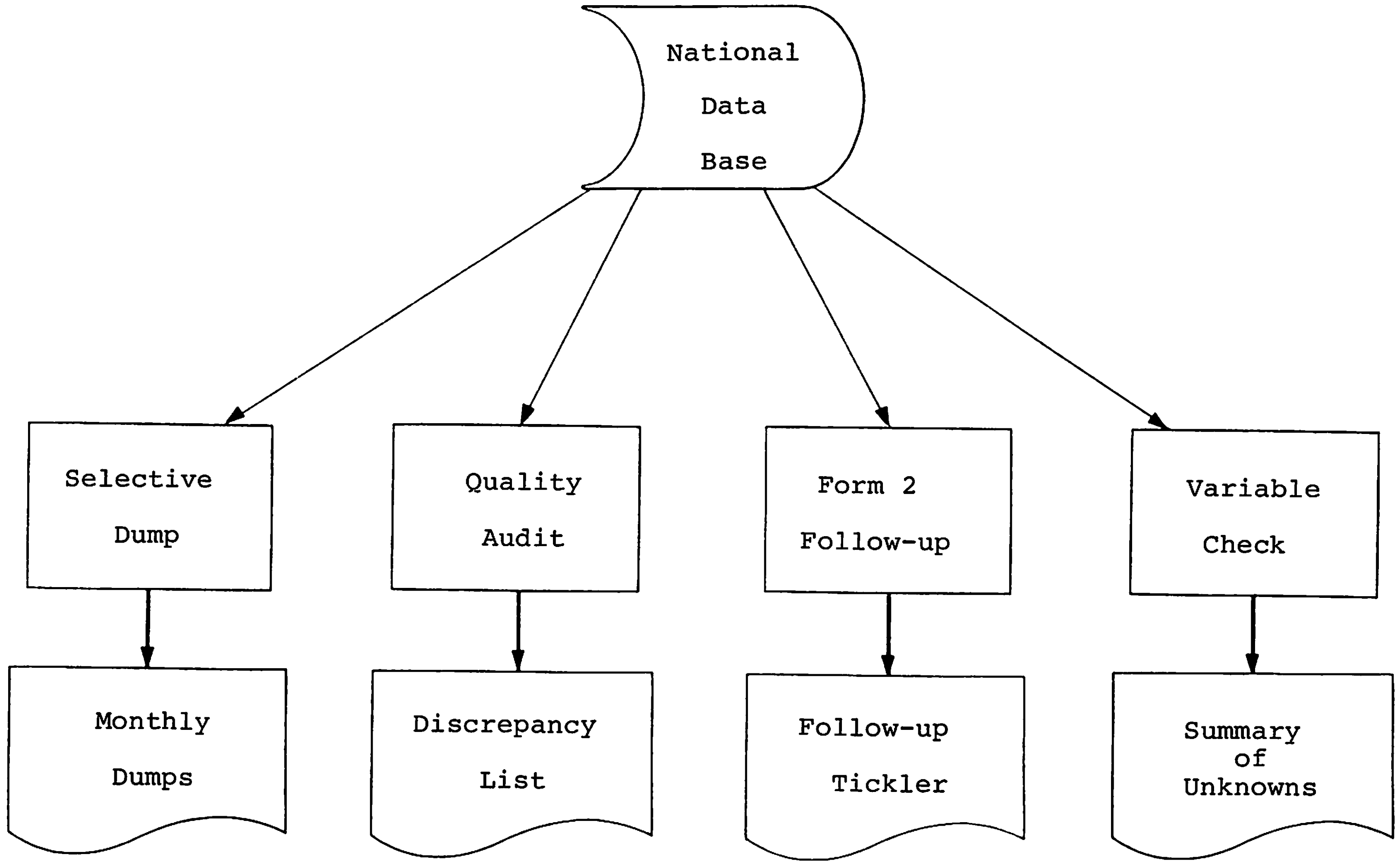


Figure 4.

analysis. As a result, there were no useful precedents for statistical inquiry and analysis.

With the growth in data volume, as well as the desire to perform statistical analyses and to make the data available for analysis via remote terminal, a hardware and software evaluation was conducted of available systems in the $100-150K price range. The HP3000 was selected as the best system in that price range. Selection was based on ease of use, multi-lingual capabilities, and data base management software in a time-shared and batch oriented system. NSCIDRC's data processing facilities are outlined in Appendix 2.

We elected to use IMAGE and, at least initially, QUERY. For our beginning efforts at inquiry into the data base and to check our conversion, QUERY was, to say the least, very handy to have. However, in establishing NSCIDRC's data bases under the HP-3000 IMAGE data management facility, it was obvious that the HP QUERY program, although a good general approach to on-line inquiry, was not adequate for our needs. As a result, NSCIDRC Computer Services undertook to design and implement a full function program that would serve all of our data bases and provide efficient interactive access to the data with our special needs for security, ease of use, and statistical analysis in mind. INQUIRY is the result of that effort.

In order to better comprehend NSCIDRC's data base management needs, let's examine the data base structure.

## DATA BASE ORGANIZATION

NSCIDRC's data base structure was designed to anticipate the need for a variety of possible access modes. The present schema structure is outlined in Figures 5 & 6. Because the schema was designed to allow use of QUERY, certain inefficiencies (which may be obvious to the experienced HP3000 user) were introduced. We will address these aspects at a later point.

We utilized Automatic Masters because we anticipated access to the data base from a variety of directions: patient number, center, number of hospitalizations, anniversary year, etc. We have learned from experience that the only Masters we need are File-key (center, patient number and anniversary year, combined) and Center. The reason for this is that data is generally selected on the basis of a combination of logical selection criteria applied to several different variables or data items.

Data Base Structure

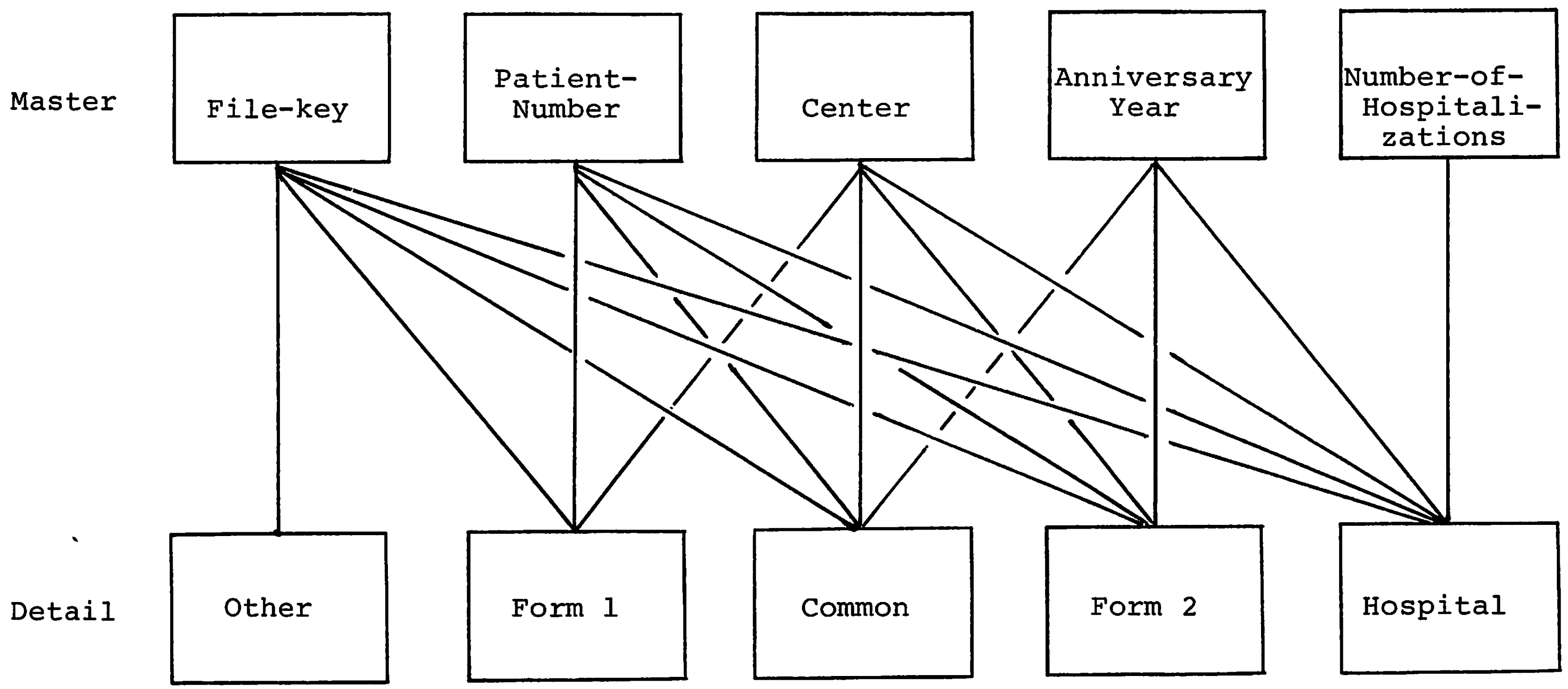


Figure 5.

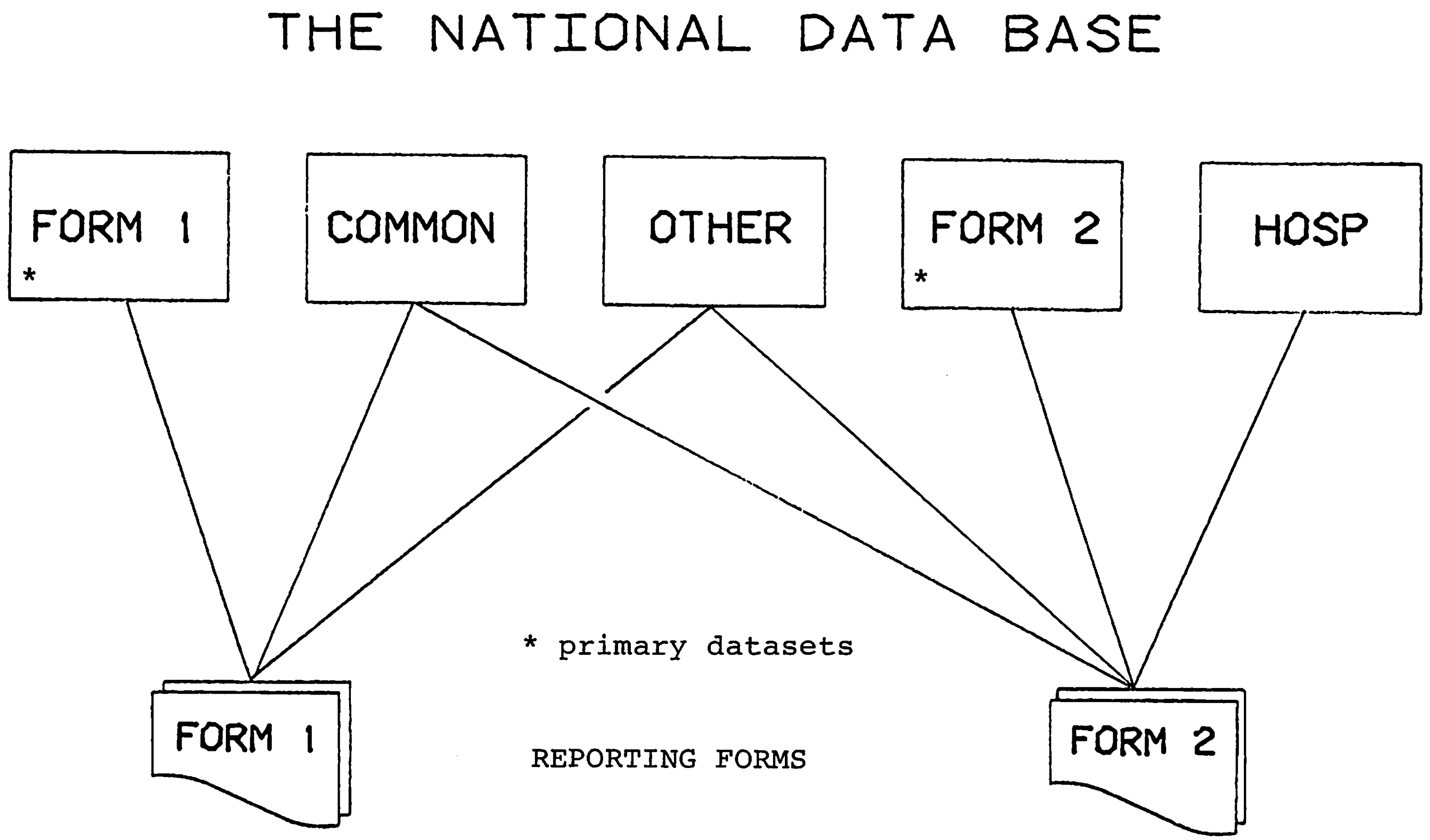

THE NATIONAL DATA BASE
FORM 1 *
COMMON
OTHER
FORM 2 *
HOSP
* primary datasets
FORM 1
REPORTING FORMS
FORM 2

Figure 6.

At NSCIDRC, the data bases are accessed in one of the following ways:

* Serial, by primary dataset
* Chained read, by Center
* Calculated read, by File-key.

INQUIRY utilizes the three methods above, plus directed read.

## RETRIEVAL REQUIREMENTS VERSUS QUERY

A brief look at the features of HP's QUERY program is appropriate here. These features are as follows:

* Interactive English-language commands, like FIND and LIST

* Single data set access

* Command features such as:
  - Find command accesses any single data set for selection of data by logical comparisons
  - List command, with access similar to Find, produces columnar listings of desired variables; some column headings will be truncated
  - Report command allows flexible output report formats, including sorted details, column totals, and register manipulation (calculate averages, etc.)
  - Report commands may be repeated to display other variables (but List may not)

* Update command features:
  - Direct access to a particular variable by name
  - Add, replace or delete a data record
  - Global replacement of a specific variable

* Execute from a command file

* Execute pre-written procedures for data selection, reports, etc.

While we liked QUERY's English-like command-driven style, we required multiple-dataset access. We wanted to be able to relate the entry forms and variables to the datasets so that the user need not be concerned with data base structure.

We wanted a List command that did some simple, automatic formatting, without losing vital header information. Since we were interested in statistical inquiry, we wanted elementary statistics on all listed numeric fields. Further, we wanted to be able to save a selected population and to extract from the data base as a separate file desired variables based on the subset population.

While QUERY must create an internal "tag" file of pointers to selected data, it is probably in an extra data segment. In any case, it cannot be saved, nor is it accessible to the user.

In addition, QUERY provides no simple means of creating an output disk file of selected variables from the subset population. The listing file can be equated, but that is a messy and undesirable workaround, particularly for the unsophisticated user.

## DESIGN OF THE INQUIRY PROGRAM

The following outline summarizes the features of the INQUIRY program:

* Multiple data set access

* EDITable directory file contains indices to relate data sets, forms, and groups of variables; output formats for the List function; field type and position for the Find function

* Selection of variables by number as on the data entry forms

* Command features such as:
  - Find command accesses all data sets in the specified form
  - Find can pseudo-chain across form boundaries, that is, it can access both Form 2 and its corresponding Form 1.
  - Find can locate cases of multiple occurrences of a value (e.g. those patients with 2 spinal fusion operations)
  - Find allows use of parenthetic notation:
    ```
    F NATL1 V104 < 7 AND &
     (V120=" 030" OCC 2 OR V130=" 030" OCC 2)
    ```
  - Temporary Tag files created by the Find command define the population and may be saved and later recalled as required; both positive and optional negative tagfiles can be created
  - List command, with access similar to Find, produces neat columnar listings of desired variables; for numeric variables, produces elementary statistics at the end of the list; details may optionally be sorted, or suppressed; variables may be decoded thru automatic tabular look-up for more readable listings
  - List and Output File commands may be repeated to display other variables from the same population, and subsets from the population may

be Listed or Output to a File via the IF option which allows further selection from the "tagged" population

- Output File may contain up to 20 variables and is compatible with the input requirements of SPSS and LISA, statistical packages available on the system
- Frequency command produces a table of frequencies, cumulative frequencies, and cell counts, along with statistical totals.
- Transform function allows creation of a pseudo-variable for use in List, Frequency, and Output File commands.

* Execute from a command file

* Command termination on Control-Y

The relating of datasets, forms and variables was solved by the use of a driver directory file. All access to the data bases uses the directory file indices which are core-resident, except for the variable descriptors' portion which, because of its size, is accessed by a binary-search routine.

The use of a private directory file as a driver has many important implications. The data base may utilize single-byte fields, odd-length fields, multiple-occurring fields. Fields may be redefined. Thus a date may be accessed in its entirety as YYMMDD, or just the year as YY, while occupying only 6 bytes of space. The only limitation on redefinition is that search-keys must be uniquely defined, although even they may be redefined for purposes of data manipulation within the program. The directory file is not a privileged file and is quite separate from the data base. Therefore, it may be edited and modified as required without necessarily affecting the data base or its schema.

Because INQUIRY is designed for use by relatively unsophisticated users who are familiar with the forms and the patient data - but not data bases or programs - it assumes a set of operational defaults. These defaults may be over-ridden by a simple command. Among the defaulted options are: choice of single- or double-spaced listings, "noisy" or "quiet" mode (in which various messages are suppressed for the experienced user), and a lookup feature in which coded data is decoded for more readable listings (e.g. sex code of 2 becomes "Female").

All commands are accepted in both full English as well as shorthand, e.g. "FIND" or simply "F". Simple error messages and help messages are provided. Syntax is entirely free-form, with continuation lines and multiple commands on a single line allowed.

## A CLOSER LOOK AT THE INQUIRY PROGRAM

INQUIRY is written in modular fashion in COBOL. While not strictly structured, it is functionally top-down in design. Although a large program, it has been carefully designed to minimize swapping and maximize execution efficiency. Its stack size of 3000 is necessary primarily because of an integral sort statement. Program segments responsible for the data base I/O are self-contained so that no segment transfer takes place until it has completed its task. As an example, the Find command is set up in one segment and executed in a second segment which retains complete control until that command is finished.

Data base I/O transfers are performed in "all-items" (i.e. full record) mode. The extraction of bytes is performed entirely by the program rather than by IMAGE intrinsics. This is the key to the odd-length field access, and the ability to handle multiple occurrences.

## USING INQUIRY IN THE ANALYTICAL PROCESS

In practice, INQUIRY can be used to peruse the data base or to extract a data matrix to test a tentative hypothesis. The user will usually save his tag file which contains binary record pointers to the population selected. These pointers are used to perform directed reads in IMAGE. If the user should decide to extract a second group of data items, the tag file previously created provides rapid access to the same population. Figure 7 outlines the basic inquiry and analysis flow.

INQUIRY can be used interactively, although it is often more appropriate to create a Stream file for batch execution and return later for the results. This is because the time between responses to user commands may range from a few seconds to several minutes, depending upon the access mode, number of datasets and records accessed, and the overall system load at the time.

A sample Job stream with annotations is shown in Figure 8. A sample listing and frequency table are shown in Figures 9 and 10.

Our user base is spread over the United States and thus connect time and telephone charges can be expensive for some users. We have established alternate methods for the user with local analytical capabilities. The user can run INQUIRY and create a data matrix of selected variables from a tagged population. He may then elect to use SPSS or LISA on the NSCIDRC HP3000.

Inquiry and Analysis

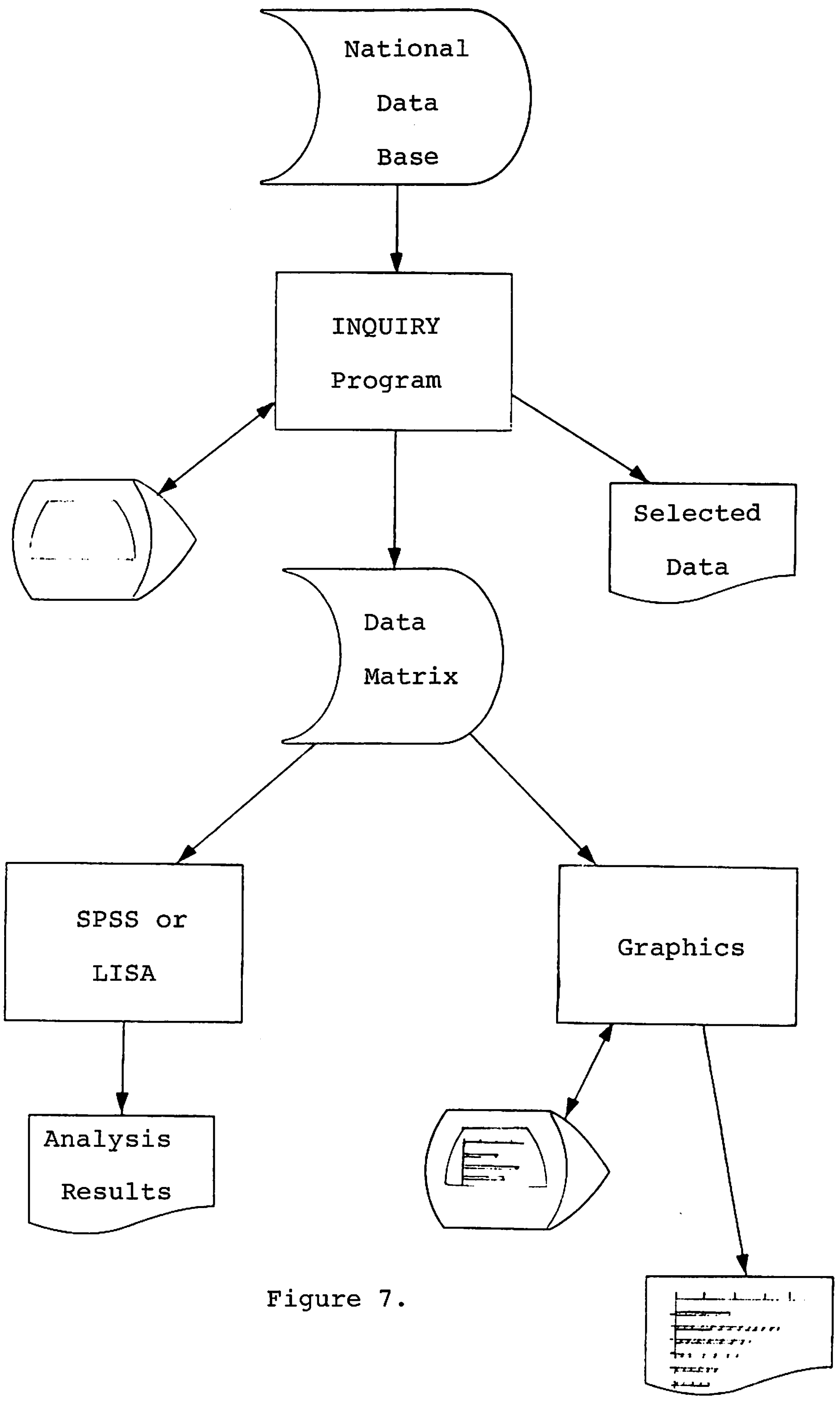


Figure 7.

```
!JOB MARTY.JEWEL/PASS
!RUN INQUIRY.MARTY.JEWEL
B NATL;M A T;M QUI;OL
                  specify which data base; access
                  all data; create a tag file;
                  use "quiet" mode; output to the
                  line printer
F NATL1 V103=1 AND V132D<5 (V119=7070 OR V129=7070)
                  in Form 1, select patients meeting
                  certain criteria
FR V107 15;S SORETAG
                  create a frequency table of variable
                  V107 (age) in 15-year groups;
                  save the tag file as SORETAG
T COST77
                  recall a previously saved tag file
TRAN V161+V166T
                  establish an equation for a
                  pseudo-variable (transformation)
LT V161 V166T TRAN
                  List totals only for the variables
                  shown, including the pseudo-
                  variable, TRAN
TRAN V160+V164+V169T+V170T
LT V160 V164 V169T V170T TRAN V172
OF CSTMATRX V161 V162 V163 V166T V167T &
 V168T TRAN V172 V132D V103
                  create an Output File called
                  CSTMATRX containing 10 variables
                  as columns, each patient being
                  represented as a row; from the
                  population given by tag file
                  COST77; "&" means continuation
E
!EOJ
```

Figure 8.

If the user has a terminal with a local storage capability (i.e. tape cartridge, diskette, etc.), the user can simply use FCOPY to transfer the data matrix to his terminal storage medium. Then the user can dial his local computing facility and feed in the data for statistical analysis.

For the user whose time constraints are more flexible, or who requires a large quantity of data, the data matrix (or the entire Regional Center data) can be dumped to a magnetic tape and mailed to the Center.

```
SELECTED CASES
AS AN EXAMPLE

132D           108            109            131
NEUR  IMPAIR   SEX            RACE           DAYS INJURY
                                             DISCH

Para  Compl    Female         Caucasian         96
Para  Compl    Female         Latin Amer.      111
Para  Compl    Male           Amer. Indian     112
Para  Compl    Male           Caucasian         58
Para  Compl    Male           Caucasian         69
Para  Compl    Male           Caucasian         97
Para  Compl    Male           Caucasian         99
Para  Compl    Male           Caucasian        157
Para  Compl    Male           Caucasian        167
Para  Compl    Male           Latin Amer.      126
Para  Incomp   Female         Caucasian         90
Para  Incomp   Female         Caucasian        135
Para  Incomp   Male           Caucasian        111
Para  Incomp   Male           Latin Amer.      132
Quad  Compl    Female         Amer. Indian     153
Quad  Compl    Female         Amer. Indian     216
Quad  Compl    Male           Amer. Indian     132
Quad  Compl    Male           Caucasian        154
Quad  Compl    Male           Caucasian        176
Quad  Compl    Male           Caucasian        179
Quad  Compl    Male           Caucasian        189
Quad  Compl    Male           Caucasian        238
Quad  Incomp   Female         Caucasian        127
Quad  Incomp   Male           Amer. Indian     133

READ =
        192
SELECTED:
   24

SUMS:
                                                     3257
MINIMUM:
                                                       58
MAXIMUM:
                                                      238
RANGE:
                                                      180
MEAN:
                                                   135.70
STD DEV:
                                                    43.89
SUM OF SQUARES:
                                                   486305
```

Figure 9.

```
AGE AT INJURY
IN 15-YEAR GROUPS

107
AGE


COUNT =
   31

 CELL VALUE    FREQUENCY   CUM. FREQ  CELL COUNT

    0 TO    14        16.13       16.13          5
   15 TO    29        41.94       58.07         13
   30 TO    44        12.90       70.97          4
   45 TO    59        12.90       83.87          4
   60 TO    74        12.90       96.77          4
   75 TO    89         3.23      100.00          1

SUMS:
        973
MINIMUM:
          5
MAXIMUM:
         75
RANGE:
         70
MEAN:
       31.38
STD DEV:
       19.57
SUM OF SQUARES:
    42031.00


MODE:
         15 TO        29
MEDIAN:
         26
```

Figure 10.

## THE FUTURE OF THE DATA BASE

In the near future, we plan to reorganize the data base in order to eliminate the undesirable space-wasting aspects mentioned earlier. We estimate that the reorganization, while losing some compatibility with QUERY, will save approximately 13% of the disk space currently used. In addition, eliminating the Common dataset will reduce the number of accesses by 25 to 50 percent, depending upon the variables accessed.

The reorganization will take into account:

* Odd-length fields
* Multiple-occurring fields
* Redefinition of fields, including search keys
* Elimination of the Common dataset by expansion of the Form 1 and Form 2 datasets

Just how do we plan to implement the reorganization? A conversion program will write consolidated records to a tape, eliminating the Common dataset and the unnecessary bytes in various fields. Then we will purge the old data base, create the new root file and data base allocation. We will then sort the tape by file-key and utilize another special program to load the data base from the sorted tape.

Since we are also modifying the data base syllabus (definitions) and adding some new variables, the conversion program will have to translate some data to new values. This will result in a somewhat more complex program, but will allow us to do the job in a single pass.

The structure of the planned data base is shown in Figures 11 and 12 (compare with Figures 5 and 6).

## SUMMARY

Spinal cord injured patient data from eleven Regional SCI Centers is submitted to the National Spinal Cord Injury Data Research Center in Phoenix, Arizona. The data is entered into an IMAGE data base on NSCIDRC's Hewlett-Packard 3000 using custom-designed software. The use of custom software for inquiry and retrieval was necessary in order to access multiple datasets at one time. It was also needed to relate entry forms and variables to the data base structure for user ease and convenience.

The INQUIRY program provides elementary statistics as well as the ability to save a subset population for later recall. It may also be used to create an data matrix as an output file for further analysis.

An added advantage of the INQUIRY program is the space savings which result from freedom from the usual limitations of the HP QUERY/3000 program.

Planned Data Base Structure

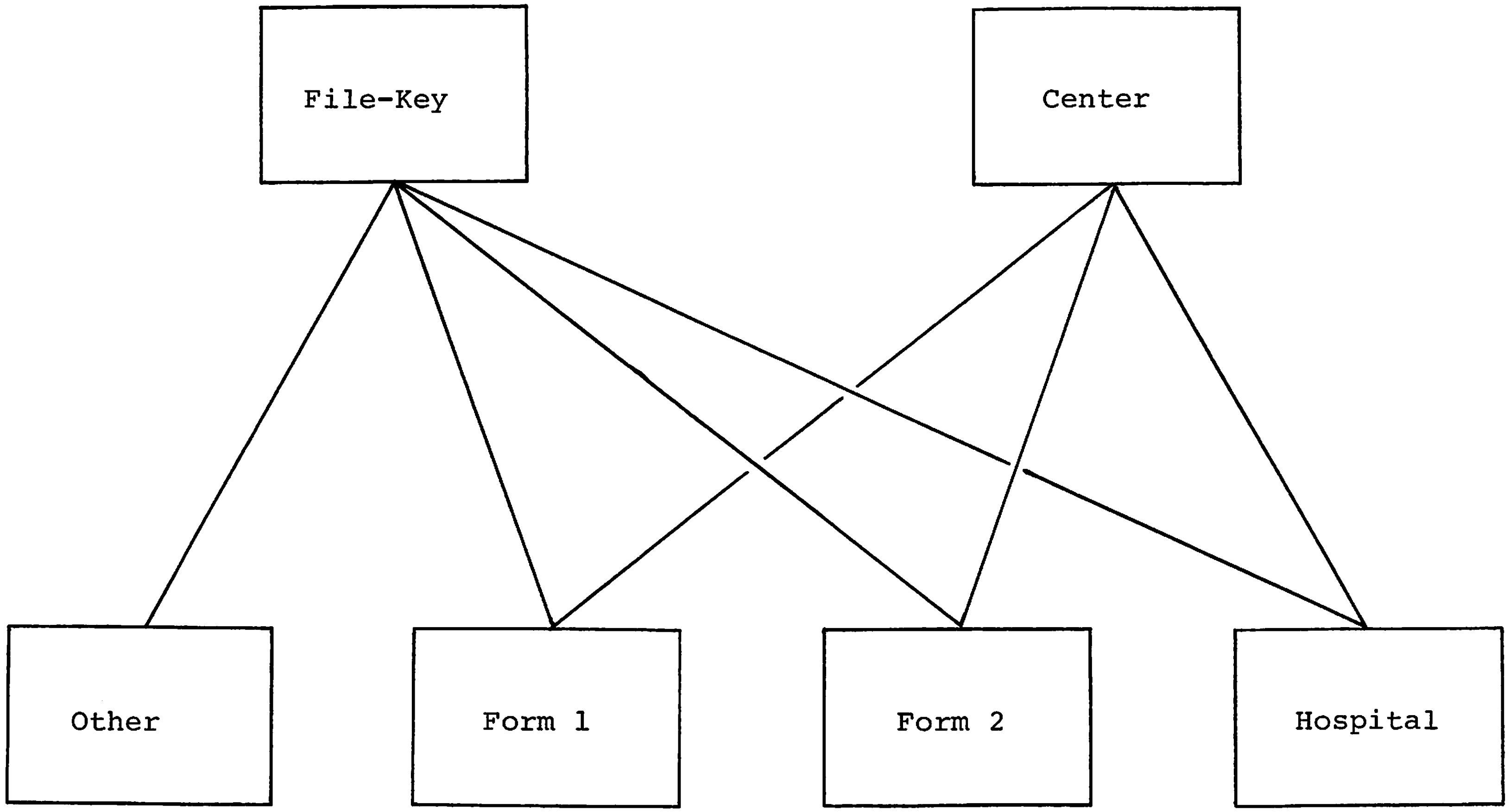


Figure 11.

THE NATIONAL DATA BASE

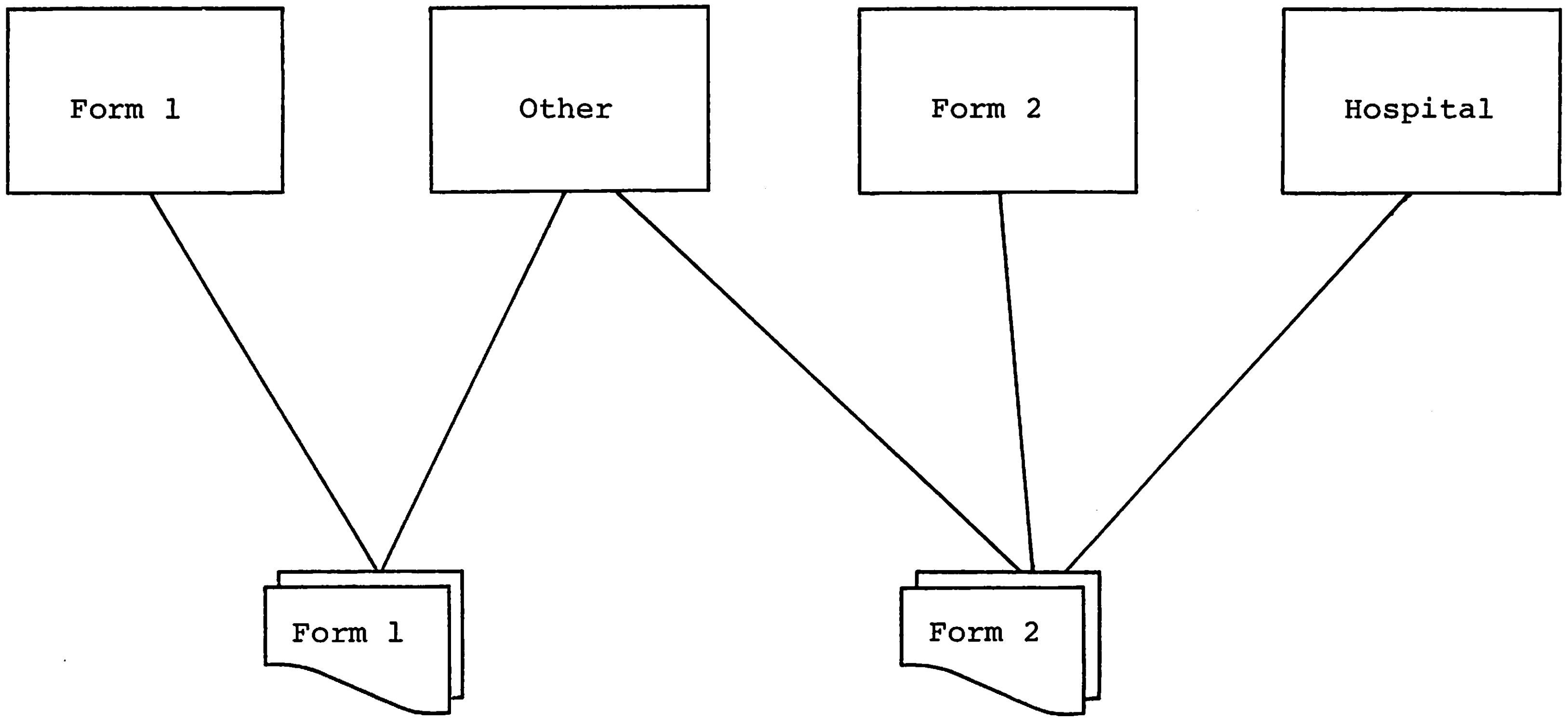


Figure 12.

The National Spinal Cord Injury Model Systems' Project is sponsored in part by the Rehabilitation Services Administration, Department of health, Education and Welfare. The following are participating institutions:

University of Alabama; Birmingham, AL

Good Samaritan Hospital - St. Joseph's Hospital; Phoenix, AZ

Santa Clara Valley Medical Center; San Jose, CA

Craig Hospital; Denver, CO

Northwestern Memorial Hospital - Rehabilitation Institute of Chicage; Chicago, IL

Boston University Medical Center; Boston, MA

University of Minnesota Hospital; Minneapolis, MN

Institute of Rehabilitation Medicine, New York University; New York, NY

Texas Institute for Rehabilitation and Research, Baylor University; Houston, TX

Woodrow Wilson Rehabilitation Center; Fishersville, VA

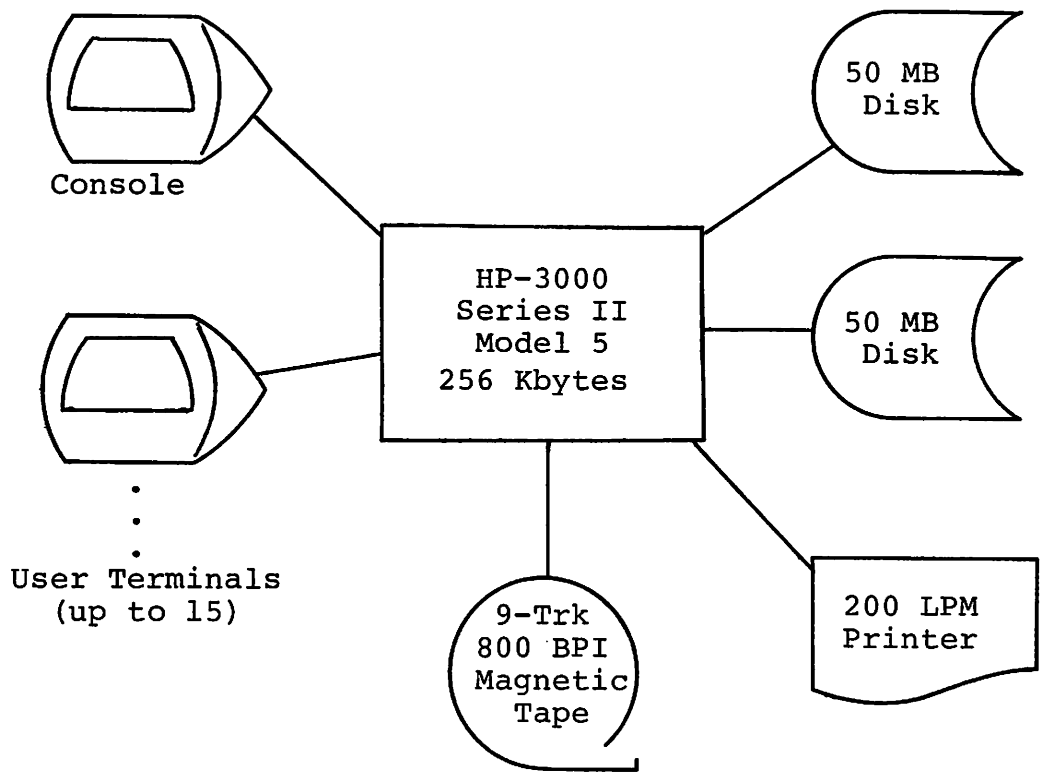
University of Virginia; Charlottesville, VA

University of Washington; Seattle, WA

N S C I D R C

The National Spinal Cord Injury Data Center

Data Processing Facilities


Console
50 MB Disk
HP-3000 Series II Model 5 256 Kbytes
50 MB Disk
User Terminals (up to 15)
9-Trk 800 BPI Magnetic Tape
200 LPM Printer

<u>Software Support</u>:

Languages: COBOL, FORTRAN, BASIC, SPL (extended ALGOL)

Data Base Facilities: IMAGE (data management system)
QUERY (inquiry and reporting)

Libraries: DEL (Data Entry Library)
Scientific Library

Operating System: MPE III (Multi-Processing Executive III)

# ON-LINE MARKETING INFORMATION

GEORGE J. NEIBERGS
E.S.B. EXIDE INDUSTRIAL BATTERIES

## BACKGROUND

Exide Industrial Battery is a division of ESB Ray-O-Vac which is a wholly owned subsidiary of INCO Corporation of Canada. ESB was founded in 1888 to supply batteries to Philadelphia Electric. The Industrial Battery Division has its headquarters in Horsham, Pa., just outside Philadelphia, and manufacturing plants located in Sumter, S.C., Richmond, KY. and Raleigh, N.C..

The primary products we manufacture include industrial batteries for lift trucks and uninterrupted power supplies with batteries for large computer installations and process control industry where backup power is required. In Raleigh, we manufacture the chargers for batteries and uninterrupted power supplies. The products are sold by our own sales force from 18 locations in the U.S. There are 34 Service Centers where the field inventory is kept.

In the past five years, there have been major changes in this Division. New products have been introduced and an old plant was closed in Philadelphia. Essentially, the Division has been undergoing change and revitalization.

One of the areas identified as requiring substantial change was in the development of new operations and systems at the Division. The old systems were primarily oriented toward accounting applications and were run in batch mode resulting in data that tended to be rather stale - showing what happened, rather than allowing anticipation of problems and opportunities.

Division Management decided that large benefits would be realized in the development of a Marketing Information System. A task force of users was then formed to define the detail system requirements. All major functional areas of the Company participated in this task force. Their efforts gave birth to a system we càll Direct Order Entry System or "DOES". "DOES" encompasses the ESB-Exide Data Collection and Order Management System. It was the intent of Management to improve the order process from closing of the sale to customer delivery. The by-product of "DOES" is improved information flow to management as well as reduction in administrative costs.

The end product of the task force was a designed requirement document which delineated all the elements a Marketing Information System should contain. This document was written by the Systems Department and extensively reviewed by the user task force.

It was determined that in-house expertise was insufficient for the installation of a major on-line order entry system. Both software and hardware proposals were solicited from outside vendors. The proposal was sent to about 12 vendors which included software and hardware combinations such as DEC, Data General, Hewlett-Packard, and Corporate Computer Center - the latter consisting of NCR and Data General. The major determinant in the selection of HP as the vendor was the availability of the high level languages COBOL, RPG, FORTRAN, and their IMAGE data base software.

The design and development phase began in June 1976 with delivery of the CX System to the software house for development. The project was scheduled for completion by the end of 1976. However, the project began slowly due to changes in personnel at the software house, and because of changes that we were making. After several setbacks, the project finally took off in November, 1976, and the system was delivered to our Horsham location in August, 1977.

Within two weeks of the system's delivery, we began entering orders on the system and started printing our invoices for a small segment of our business. Within a couple of months, it became obvious that the CX could not handle the workload nor could it perform all the functions we desired of the system. An upgrade was made to the Series II in February, 1978.

We currently have the HP-3000 Series II with 512 k memory, (1) HP7905 15 megabyte disk, (1) HP7920, 50 megabyte disk and (1) 47 megabyte disk. There are 14 on-line order entry and inquiry terminals, 4 terminals for system development and one on-line terminal printer. We also have a 600 line per minute printer and a card reader. This upgrade improved the CRT response time to an acceptable level.

## SCOPE

Marketing Information System "DOES" controls the order from initial receipt to final invoicing of the product.

1. The order entry covers three distinct products and meets the order entry needs of multiple market places. It can be as simple as entering a catalog number or as complex as actuallly designing the final configuration of a total battery installation.

2. Reservations can be made against batteries and chargers located in one of the 34 Service Centers or in transit to the Service Centers. The salesmen can inquire of product availability at the plants in transit, and at all the service stations, by calling the Customer Service Center in Horsham.

3. Assignments of firm customer orders are made against stock orders that are being produced at our plants. This is not an automatic function. An operator has to review the CRT display and then select the order she wants assigned

based on its need date and availability.

4. Shipments are entered and invoices are requested each day. In addition, each order is checked on the CRT against the customer purchase orders.

5. Customers can call and make inquiries on the status of their orders. Multiple keys are used for simplifying the order search. Since this is the most frequently used function, we designed it two ways; one is part of the "DOES" system; the other is a stand-alone inquiry routine. Since "DOES" requires heavy overhead to run, we hope to reduce the resources required.

6. Each day operational reports such as invoices, acknowledgements and shipping documents are printed for mailing the next day. In addition, various documents are printed to control paper flow at the Division. We have reduced the number of internal copy order distributions from 48 to 12.

7. End of the week reports are prepared on gross profit regarding orders received, discount analysis and freight allowances given.

8. Monthly reports consist of a sales backlog. Future plans include a profit backlog.

9. The users are writing their own QUERY programs for on-line inquiry of backlogs, order assignments and the amount of business generated from a given customer. The QUERY is used to make ad-hoc reports for management. The only complaint is that they can't cross data sets.

## DESIGN CONCEPT

The system is divided in two major parts; the on-line, which is available from 8:30 a.m. to 5:00 p.m. and the batch phase, which is run from 5:30 p.m. to about 8:00 p.m. We would not start the on-line function unless there has been a successful completion of the batch process the night before. This separation of functions insures that we do not have problems with cut-off dates.

There are three data bases on the system: Open Order, Product and Customer. The Open Order data base duplicates all the products and customer information for each order. The Customer data base, in addition to containing customer information, contains reservations and availability of orders at the plants. The product data base consists of all the product information and freight rate tables. These data bases were selected in order to prevent excessive contention on one data base due to the requirement of locking at the data base level. Therefore, there's no contention between entering orders and product availability inquiries.

The languages used are FORTRAN for the on-line application, COBOL to extract information from the data base and RPG to prepare various reports on the extracted program. A special screen monitor is used to control the writer and reach to and from CRT's, and to call application routines. Block mode (page) is used to transfer data from the screen to the computer at 2400 bd. All transactions that update the data bases are logged on a single transaction file. A special recovery program is used to rebuild the data base by applying the logged transactions to the most recent back-ups. The on-line module consists of 62 code segments. The largest is 6700 words, the average is 2000 words. The total size of the on-line module is 124k words. The data stack is approximately 8k words of which approximately 5.3k words are the global area (FORTRAN common).

Communication with the CRT's, updating of data bases, and editing of input data are centralized in three of the 62 segments of the "DOES" manipulation. As a result, more than 90% of the time required to process a transaction is spent in one of these three segments. Since we are dealing with a re-entrant (or code-sharing) system in the HP-3000, this allows a great deal of efficiency to be gained as these three segments tend to remain in memory. Thus, we have decreased the amount of memory we would normally require, and decreased the terminal response time.

The batch program consists of about 40 different programs and each program performs a specific function. No attempt was made to utilize one program for multiple functions. This concept has allowed us to make changes to extract and report with minimum impact on other programs.

## SPECIAL FEATURES

1. Every night we notify 18 sales offices of the number of orders we have received from them, any reschedules and special shipments that are authorized on their location. The achieved objective was to reduce the incoming phone calls since all the information was available at the sales offices.

2. In addition, each of the CRT operators can transmit administrative messages to respective sales offices or a general message to all of the sales offices over the TWX network by utilizing a "DOES" function.

3. A special subroutine was written to estimate freight which is based on Ship to Zip Code, weight and FOB point. If the customer requests the freight invoice, we utilize this subroutine to prepare him an invoice at time of shipment.

4. Our field offices have the capability to TWX orders into a central TWX machine where a paper tape is generated for subsequent batch update of the data base.

## COMMUNICATIONS

1. Every night we transmit shipping documents and bill of ladings to Data Point to our plants using IBM 3780 protocol.

2. Every night a specially formated file is transmitted using IBM 3780 protocol to Western Union Data Center in Virginia for distribution of daily newspaper over the Western Union TWX network.

3. The HP is also used for transmission of information to our IBM 148. The protocol used is 2780.

## FUTURE PLANS

We are currently in a phase where we are building our data bases by order only. The next phase will be to utilize this information for management reports.

## ACKNOWLEDGEMENTS

I'd especially like to thank Chuck Aigen, now of R. Schriver Associates, for his many novel design concepts and almost all the on-line programming, and also Paul Schatschneider, the User Project Manager, who implemented "DOES" making it a reality.

## BIOGRAPHY

George J. Neibergs is Manager, Information Systems for ESB-Exide in Horsham, Pa. Mr. Neibergs was formerly with Eaton Corporation as Materials Manager. Prior to this, he was with GTE Sylvania where he held various managerial positions in Manufacturing.

A graduate of Rensselser Polytechnic Institute, Mr. Neibergs holds a B.S. in Mechanical Engineering and a M.S. in Management Engineering.

# Computerized Word Processing

a Presentation for the
7th International Meeting of the
HP General Systems Users Group
Denver Colorado, November 1978

by
Martin Gorfinkel
Los Altos Research Center
339 South San Antonio Road
Los Altos, California, 94022 USA

## Description

A Computerized Word Processing System captures documents in machine readable form as they are first typed. The machine readable form allows changes to be made in a document without retyping. Formatting instructions are stored with the document or can be changed after input of the document to allow output in various formats - again without retyping.

Capabilities vary greatly from one system to another. Some key items are: ease of handling large documents; ease of integrating the word processing with other computerized functions; the variety of output and input devices available for use with the system; and the ease of use of the system.

## Advantages

Material is typed manually only once; the time and errors involved in retyping is eliminated. Reports from other computerized systems can be automatically integrated into the word processing product. (Financial tables and mailing addresses from a data base are two examples.) A greater variety of format between documents and consistency within documents is easier to attain. Collaboration in writing and editing among several authors is facilitated, particularly when the authors are in different locations with access to the same computer.

The dis-incentive to make last minute corrections or improvments to a document is removed. The final copy is produced quickly, neatly, and without introduction of new errors.

A well designed system will be easy for clerical and secretarial staff to learn and to use. The system should adapt well to both intensive and casual use.

## Features

Features which can provide additional benefits to the user include: incorporation of one document within another; formatting of numeric tables and calculation of totals on those tables (saves proof reading the numbers); inclusion of the current date as a document is printed; flagging lines changed or inserted; automatic generation of an index and/or table of contents; and an encryption scheme to protect the confidentiality of sensitive documents.

## Sample Applications

Program Development and Documentation
Technical Documentation
Proposal and Contract Writing
Letter Writing and Mass Mailings
Financial Report Writing

# DOLLAR-FLOW: FINANCIAL PLANNING ON THE HP3000
(users write their own programs)

By Jack Damm, Principal, The Palo Alto Group, Sunnyvale, Calif. (408) 735-8490

Good afternoon. I am going to talk about financial planning on the HP3000 with the Dollar-Flow planning language. My discussion will focus on three areas: 1) What financial planning is, and why there is a need for computerized planning; 2) Design considerations for "friendly" user-oriented applications; and 3) How the language Dollar-Flow is used for applications like profit planning.

## THE NEED FOR FINANCIAL PLANNING

First, let's start with two questions: What is financial planning? And why is it necessary? Financial planning is making decisions about allocating the scarce resources of an organization so as to best achieve its goals. In the private sector, this usually means how best to allocate money and people to achieve profitability goals. In the public sector, it may mean how best to allocate people and dollars to provide a desired level of service. The main idea here is that the resource is scarce and, as a manager, hard decisions have to be made about how to use it. More specifically, financial planning is setting budgets, making pricing decisions, and estimating future demand for products and services, in order to achieve profit and/or performance goals.

Why is formal planning necessary? First, of course, because a scarce resource (typically money) is involved. If we had enough money for everything, then we could simply raise our salaries and retire early. Secondly, it is very important to have general agreement within an organization about how goals are to be achieved. No assumptions should be made without clearly stating and documenting them. With a good financial plan, trouble signs can be spotted earlier and corrective action taken sooner. Businesses which fail to plan effectively are the best illustration of the need for planning.

Let me offer one last reason why planning is important. For many companies, planning is a necessity because of the complexity of their operations. A typical manufacturing company may purchase thousands of parts for use in a vast array of products, and assemble them in many different locations. They cannot wait until there is no money in the till to decide that its time to raise prices. And our current rate of inflation makes this an even more important consideration.

## THE TYPICAL PLANNING PROCESS

Okay, let's assume that you accept the need for financial planning. So what's the big deal? Well let's look at the typical planning process and I'll show you.

First, planning involves lots of numbers. And these numbers change often. Financial planning involves projections into the future and is a very uncertain process. When you're uncertain, then you have to do contingency planning. Play "what if" games. What if sales are 20% higher than planned? What if the cost estimates are too optimistic? What if our product sales mix is different? Because of uncertainty, alternative plans are necessary, increasing the amount of work required to plan several times over.

And that's not all. The attempt to reach a targeted objective such as profit adds to the work. It may take several passes before all of the budgets combined with the sales estimates, cost estimates, and so forth, sum up to the desired results. The task soon becomes monumental.

The following is not an uncommon occurrence: You work many hours preparing budgets and doing sales forecasts. With a board meeting just a few days away, you finish your plan. The company president takes one look at the results of the combined numbers and gives it back, requesting a 15% cut in the budget. You prepare a revised budget, repeat all of the calculations, this time under increasing pressure to get the job done fast. The day before the board meeting, marketing revises the forecast. All of the budgets must be revised again. And now it is getting late into the evening the day before the meeting. Everybody is getting tired. After a few more iterations, exhaustion sets in. The planning process finally ends. With a good plan? No, with exhaustion. Does this seem like a doomsday tale? It's not. I've see this happen many times. No wonder people dread budgeting time.

Combine the sheer effort required to effectively plan with the requirements for a good plan: It must be TIMELY. In a dynamic, growing company, a plan must reflect todays expectations, not yesterday's. It must be ERROR FREE. Late-night, reworked plans suffer from simple calculation errors. Errors due to using the wrong set of estimates, because they keep on changing. Imagine the embarrassment of a summation error. And with all this, the plan must remain FLEXIBLE. I worked on a profit plan for a company a few years ago which added an entire product line between iterations of the plan. And finally, when you are all done, a good plan must be WELL DOCUMENTED. What factors were used for overhead? What was the basis for the final sales figure? How was a particular number calculated? All too often, there is little documentation on how a plan was actually prepared.

To summarize: A typical financial plan involves lots of numbers, which change often. The need for many iterations makes this process time consuming and exhausting. At the same time, the plan must be timely, error free and well documented. In short, good financial planning is not easy.

## WHAT IS THE BEST WAY TO PLAN?

Given that this is the nature of planning, what is the best way to plan? How can it be done with a minimum of difficulty? Traditionally, there have been two ways of planning. Planning by hand (and calculator) and planning using the computer. Let's take a look at both of these methods and evaluate the pluses and minuses of each.

Preparation of plans manually has several drawbacks. First, because of the amount of data involved and the number of iterations, it is slow and time consuming. After many iterations, accuracy becomes a problem. The wrong estimates may be used, particularly if they keep changing. Calculation errors seem to increase with each iteration. And documentation is usually not very good.

On the other hand we have financial planning on the computer using the traditional programming languages like BASIC, FORTRAN, or COBOL. Once set up, a model written in one of these languages will run on the computer in a matter of minutes or seconds. Great! But here's the catch. The model will run very quickly once it has been set up, but it may take months to get it developed. And you need a programmer. Let's see what can happen. You start your plan well in advance of the next budgeting cycle. With six months lead time you give a precise set of specifications to an enthusiastic programmer who dutifully sets about coding your model. At the end of the first three months, he comes back to you with his first try. You patiently point out where the model is not consistent with the specifications, settle on a set of revisions, and the model is reprogrammed to your satisfaction. All set, right? No. As you begin using the model, the company president starts to change his mind (even though he reviewed the original specifications). Add a decimal place here, another line item there. Why aren't all twelve columns of data on the first page? Frustration.

What is the moral of our story? Programming a planning application with the traditional programming languages lacks flexibility. The programmer needs lead time to set up the application and has difficulty in reacting to short term changes. How about adding another division to a multi-divisional company? Try changing every format statement in the model in an hour. And add to that the bother of documentation.

To summarize, manually prepared plans can be flexible, but they take a long time to do and lots of effort, especially if several passes are done. They often lack documentation. Planning with traditional programming languages takes too long to set up, is inflexible, and requires the services of a programmer.

## PROBLEM ORIENTED LANGUAGES

Let me digress for a moment. For several decades now, computer scientists have been searching for a "universal" programming language. ALGOL? PL/I? APL? The search goes on. Each has its merits, each its disadvantages. But these "procedure oriented" languages have one thing in common: You have to be a programmer to use them. And it is altogether too easy to include bugs in even the simplest of programs. As long as there is a programmer acting as middleman between the user (or analyst) and the computer there are going to be communication problems. Maintenance problems. Resource and priority problems.

What's the answer? A planning oriented application language which incorporates the good aspects of traditional programming, but eliminates the problems. Where plans can be set up and revised easily, without having to be a programmer. What I am describing here is one example of another class of programming languages, "problem oriented" languages. Languages which have been designed to provide solutions in a general way to classes of problems. Simple enough to be used by non-programmers. Easier to debug. Self-documenting. QUERY is an example of a problem oriented language. It provides access to IMAGE data bases in a fashion simple enough to be used by non-programmers. Dollar-Flow is a problem oriented language, designed as a tool for non-programmers who want to set up tabular planning reports.

Financial planning is an area well suited to problem oriented languages. There is a considerable amount of generality in what planners do, although no two plans are the same. A financial plan typically involves mathematical operations on rows and columns of numbers. With well defined rules for the calculations. And the burden of planning in any other way give the financial planner considerable incentive to try new approaches.

This is a good start. But we still have to get the planner onto the terminal and communicating with the computer. How is this done? By giving him an effective tool. One which is both friendly and enables him to get the job done in a way that he understands.

## DESIGNING FRIENDLY SYSTEMS

This leads us to the next point: What makes a system "friendly"? How can a system be designed so the novice or non-computer type feels comfortable with it? I offer here a few of my ideas and techniques for developing friendly systems.

### SIMPLICITY

Keep the system simple at all cost. Do not let the internal structure on the computer dictate how a system looks to the user. Let him express his ideas in his own terms. For example, the original design for the Dollar-Flow language was based on a set of documentation which I prepared for a group of accounting types. This documentation described the workings of a particular customized model on a line by line basis. I figured: What could be a better set of design specifications for a language than actual documentation? As you document your model you are also writing your program! Another example. Dollar-Flow re-orders calculation rules automatically. Thus, line 1 on a report can reference data on line 10, which, in turn, can reference data on line 20. Dollar-Flow automatically figures out the proper sequence for calculations (calculate 20, then 10, then 1) without any intervention by the user.

It is important that the application be self documenting. For example, Dollar-Flow is a menu driven system. At each step of operation, the user knows his alternatives. There is little need for a "pocket guide" to the language. This is not to say that there is no need for manuals. A good manual is important. But it is a fact that few people actually read manuals. The less a system forces a user to read the manual, the more usable it will be.

Not only should the user be told what his alternatives are, the system should also help him to choose the proper response. Throughout the Dollar-Flow prompts, the most likely response is shown in brackets as the "default" response. In some cases, he can use the default response without bothering to even understand the question! For example, the prompt:

```
USE STANDARD OVERALL REPORT FORMAT (<Y>,N,W-WIDE PAGE)?
```

In one brief prompt, the user can see his options and pick one. A simple carriage return will cause the system to use the default response. And his entire report format is set up. No PRINT USING or FORMAT statements. Very simple. And it can be changed easily. As the user becomes more familiar with the language, he can begin to exercise more options. With an 'N' response, Dollar-Flow leads the user through a review of the many formatting alternatives. Report formatting can even be done on a trial and error basis. Start off with the standard format, then change the column width or number of decimal places shown as needs require.

As I already mentioned, the design for the Dollar-Flow calculation rules was based on a set of user oriented documentation. Ask a user to describe how the values on the report are to be calculated in his own terms. With the addition of a few quote marks here and there, he has already written a program in the Dollar-Flow language. Self-documenting languages not only save the effort required for documentation, but make debugging much easier.

One last comment about simplicity. Save the user concerns about internal structure through structure independent (or data base) approaches to data relationships. One of the beauties of QUERY is that the user doesn't have to concern himself with all of the details of the data base to get a simple report. In Dollar-Flow, all reports are programs, all saved programs are files, and all save files contain reports. To reference data on a saved Dollar-Flow report, simply indicate the line name and the report save file name:

```
MARKETING BUDGET = 'BUDGET' OF 'MKTG';
```

There is no need for the user to know how the data is stored or even which line on the 'MKTG' report is the 'BUDGET' line which he is using.

## ERROR HANDLING

Okay, so let's say you have implemented a simple system. Does this mean that users won't make mistakes? Of course not. In fact, the friendlier a system is, the greater the likelihood that the users will not be computer types. So, keep in mind that "to err is human, to forgive is good systems design." Of course, you must edit all inputs. But then use a friendly approach when the user has made an error. Because Dollar-Flow is menu driven, simple typing errors cause the system to repeat the prompt. Errors of a more complex nature, such as where a report is referenced but does not exist, generate intelligible error messages. Along with each error message give a message number. And provide a glossary with the documentation which gives even greater detail on the possible cause of the problem.

At the same time that it is informative, a system should help the user to work around problems. For example, in the case of an invalid report reference in Dollar-Flow, the user can interactively specify a different report name, or values, or zeroes. He can also indicate that computation should cease after a scan for further errors. Again, unless a particular error is extremely serious, warn the user and proceed (with his permission). Another example. As far as the mathematician is concerned, division by zero gives unworkable results. In Dollar-Flow, division by zero yields 'invalid' numbers (which print as asterisks), but doesn't stop computation. It's amazing how much more satisfying a user finds a report filled with asterisks than just a list of error messages. At least he can look at the format to see if it's to his liking.

If you must tell the user that he has made an error, tell him as early as possible. One of the most enlightened things done by the MPE operating system is to edit the job card image when a job is being streamed from an interactive session. It sure is better to find out right away than waiting for the job to begin execution to find out that a simple error has been made on the JOB statement. Report development in Dollar-Flow is completely interactive. If a user is setting up a report and he enters a calculation rule with invalid syntax, the system responds with a message immediately, and permits him to edit his error (not unlike the BASIC interpreter). It is not necessary to go into the computation step to find many errors.

## MAINTENANCE AND SUPPORT

Let us assume that as an enlightened designer of friendly systems you have now designed and implemented your masterpiece. Are you done? Of course not. This is only the first step. There are two more important aspects which are critical for good, friendly systems: Continuing improvement and good support. Let me talk about continuing development first. No system is great on the first try. I am a believer in the iterative approach to systems development, if you can afford it. I am not talking about sloppy design. I am talking about the tremendous wealth of ideas that you can get from your users, AFTER you have implemented a system. Try to be receptive to the suggestions of your users (even if they are infeasible). Never give a critical user the impression that you think he has just offered a bad idea. Go out of your way to solicit ideas from your users. If the situation merits it, get involved in several of their applications. You can learn about ways the system is being used that you never thought about. Ways in which its use may be awkward. Which messages are more annoying than useful. Which features are badly needed. I send periodic questionnaires to my users (some of them even respond). This helps to prioritize new features. And users group meetings are a great boon to information flow.

How should this wealth of new ideas be integrated into an already developed system? Carefully. Do not rush a new version of a system out to users just because they need a particular feature. You must let a new version of a system be "burned in" first by a test site. Software bugs cost you credibility. Once lost, credibility is very difficult to reestablish, so reliability is extremely important. After all, would a user prefer a system with the bells and whistles he wants but doesn't work, or one which works with a few less features?

Speaking of bugs and user suggestions leads me to the question of support. There is nothing more frustrating to a user than to get 95% of the way to his computer solution only to be stopped by the application package he is using. For any reason. If you can afford to do it, good support pays great dividends. Dollar-Flow is supported in an "on-line" fashion. This means that if a user has a problem, he picks up the telephone and calls. If his problem is with an existing report, I may even log onto his system and take a look at that report. This kind of support not only helps to find and eliminate system problems quickly, but I also find out about areas where the documentation may be confusing (or incorrect). Where another feature might simplify the users application. In short, on-line support can be another source of good ideas from users.

Let me summarize these techniques for creating friendly systems. First, KEEP IT SIMPLE. Try to think like the user instead of a computer expert. Use his terms. Assume that he won't read the manual. Try to make it self-explanatory. Second, be INFORMATIVE but FORGIVING with your error handling. Edit all inputs, but don't bother the user with minor errors. When the application merits, CONTINUING ENHANCEMENT will make a much more usable system. Respond to user suggestions. But exercise good judgment in the trade-off between adding new features and degrading SYSTEM RELIABILITY.

## PROFIT PLANNING

I am not going to take too much time on the last part of my talk. I am just going to show you a few sample reports prepared using Dollar-Flow. At the risk of violating my agreement not to make a sales pitch, I invite you to visit the PALO ALTO GROUP's booth on Wednesday for a demonstration of Dollar-Flow in action.

Let me first describe to you the typical company profit planning cycle and the environment in which a planning tool like Dollar-Flow is used. The typical Dollar-Flow user is the accountant or company controller who is responsible for preparing the reports. Not a programmer. Most users are working on in-house HP3000 systems. With access to CRT's and a system line printer nearby. Reports are written interactively, and manual inputs are also entered via the terminal. Usually, reports are printed on the CRT for review then saved when the user is satisfied with the report. If hard copy is desired, the reports can be routed to the line printer. For generating large numbers of reports, the "batch command mode" is used, where with very little terminal input a large number of reports can be generated.

Profit planning typically begins with a preliminary sales forecast. Preliminary. Sales forecasts always change. And at the last minute, too. Often the sales forecast is done on a product-by-product basis for the first year or so, then combined with overall dollar sales projections further in the future. The near term unit forecasts are sometimes adjusted based on an overall dollar figure. The forecast is iterated several times. To make a change, the product manager just runs Dollar-Flow, inputs whichever figures have changed, pushes a few buttons, and the new sales forecast is ready. Since many parts of the profit plan depend on this sales forecast, the typical plan is usually set up with reports referencing the sales forecast report. If the figures are changed on the sales forecast, these changes will be automatically reflected on the other reports the next time they are run. Some manufacturing companies even use a multi-level sales forecast step, where a build plan (or production plan) is generated from the sales forecast.

Meanwhile, departmental budgets are prepared. Some Dollar-Flow users centralize the budgeting function and only distribute budget worksheets to each department or location. This is usually done if there are only one or two budget iterations. On the other hand, some of my customers distribute the budget preparation, with each location setting up its own budget in Dollar-Flow. In this case, figures can be input to Dollar-Flow, changes can be made, and several iterations of the budget can be done all in a matter of minutes. And budget consolidations are fun! With a few simple commands to Dollar-Flow, a whole series of budgets can be consolidated into a departmental or divisional budget. When changes are made to the low level budgets, they automatically are reflected on the consolidated budget the next time its run.

The profit/loss projection is next. Using the data from the sales forecast, the build plan, and the budgets, and adding factors for items like sales discounts and returns, a pro forma operating statement is prepared. Often, the bottom line (profit) on this report determines what (if any) changes need to be made to the budgets. With a flexible tool like Dollar-Flow, a financial executive can even do sensitivity analysis: What if sales are 20% lower than forecast? What if our discount schedule is more aggressive and our volume is larger?

Some companies that rely on substantial amounts of debt to finance their operations combine the profit/loss projection with a cash flow projection. This is because interest paid (an item of expense on the profit/loss statement) has an impact on the amount of money required to run the business. This determines the level of borrowing, which, in turn, affects the amount of interest which is paid. Dollar-Flow, and most good financial planning languages, can solve the "simultaneous equations" this circular logic represents, and determine a level of debt and debt service which are consistent with each other. This is far more difficult when done manually.

Another procedure which is laborious when done by hand is the aging of accounts receivable and accounts payable projections. Using Dollar-Flow, once the rules for aging have been set up, a change in the sales forecast or the build plan will automatically be reflected in new receipts and payables projections.

And, finally, some companies prepare pro forma balance sheets as the last step in their profit planning cycle. This is not necessarily the way all companies plan. Or even the way all Dollar-Flow users plan. In fact, many Dollar-Flow users are not even responsible for profit planning. Instead, the system is used for a wide variety of ad hoc applications involving calculations on rows and columns of numbers. It is even used as a design tool for systems which will later be hard-coded in COBOL, FORTRAN, or BASIC.

Some of the other applications of Dollar-Flow that I am aware of include:

Product pricing. Comparing alternative prices for a single product (the plotting capability is great for comparisons). Or comparing profit percentage across an entire product line. Financial ratio analysis. Comparing selected financial ratios against industry standards or company objectives. Capital budgeting. Rates of return and discounted cash flows can be calculated easily using built-in financial functions.

Performance reporting. Variance reports showing actual budgets or profits versus plan. How sales are doing against target. (One Dollar-Flow user generates 500 graphs every quarter showing product line sales performance for every branch of every distributor who markets his products!)

## SUMMARY

Let me leave you wish a few parting thoughts. Financial planning is not an easy process. Figures change. The whole approach to a plan may change. And you need your results yesterday. Traditional systems design and programming methods are not going to be effective in this kind of situation. Use a better approach. With a friendly, problem oriented planning language like Dollar-Flow, applications nightmares can become applications successes.

# Considerations for a Typist Oriented, Fully Integrated Wordprocessing System

by

DARYL A. FRAME
and
ROGER M. GOLDMANN

COMARCO, Inc.
227 W. Hueneme Rd.
Oxnard, California 93030
(805) 488-6441

Wordprocessing systems are the glamour products in today's office equipment market. However, until very recently, Wordprocessing has for the most part been ignored by Data Processing managers and personnel because they did not recognize that they could offer service to departments which by nature perform labor intensive, repetitive tasks. Typical departments which have seemed unlikely candidates for Data Processing services are Office Administration and Office Services.[1]

It is extremely doubtful that this situation can continue much longer. Without the help of today's latest computer technology, these formerly non-Data Processing departments will become buried in the avalanche of paperwork which now passes through them (Fig. 1). In some organizations, Data Processing personnel are already being called upon to assist in selecting Wordprocessing equipment or software. This really should not be too surprising, since the technologies used in Data Processing and Wordprocessing are too similar to ignore.

A few years ago, the typical office accounted for only a fraction of the total company budget, however, this is no longer true. Greater demands are being placed on office personnel and therefore higher wages are required. This has a tendency to increase the cost of all office services (Fig. 2). Without new methods, the future prospects reveal more increases in cost without any appreciable increase in productivity (Fig. 3).

On the other hand, costs within the Data Processing field appear to be declining (Fig. 4).[2] The needs of formerly non-Data Processing departments cannot be ignored. As more and more demands are placed on them, the need to apply technological expertise will be mandatory.

However, there is a significant problem. Based on a qualitative survey conducted by Starch INRA Hooper, Inc. during May and June of 1977, "corporate decision-makers are neither sufficiently knowledgeable nor comfortable" with the concept and application of Wordprocessing and they feel that "while the industry *talks* the system concept, it actually *sells* pieces of equipment."[3]

## THE PROBLEM

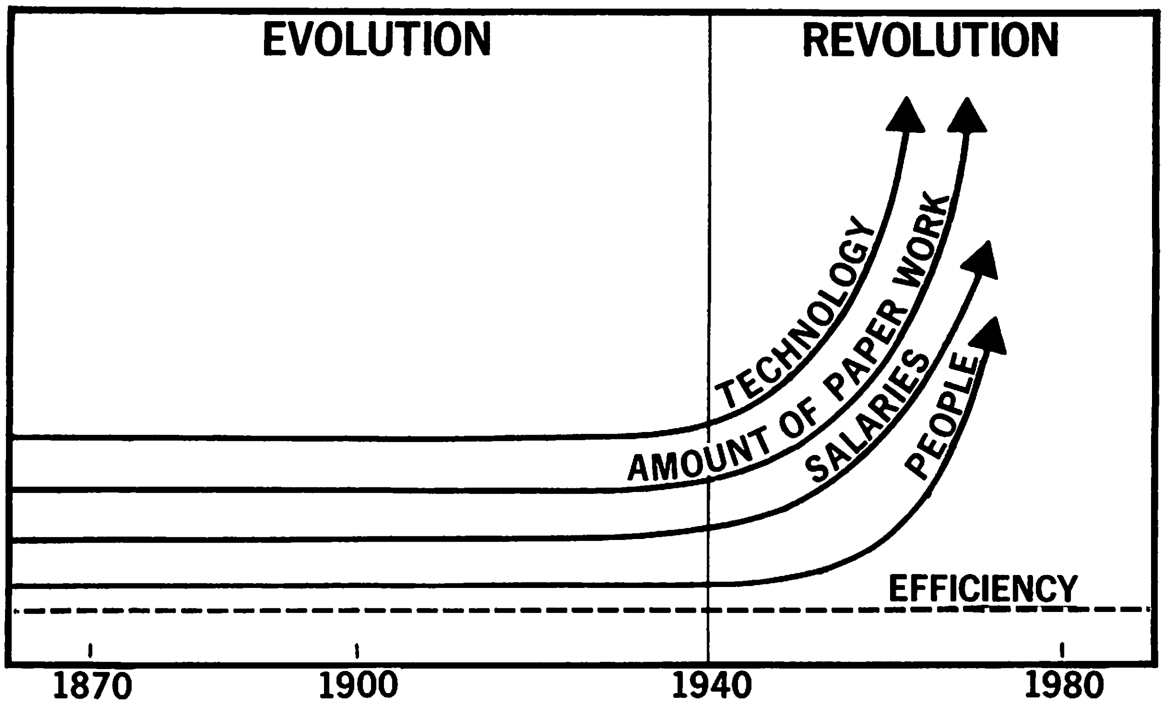


Fig. 1

## OFFICE COSTS

- WERE 20% TO 30% OF TOTAL
- NOW 40% TO 50%
  - — DEMANDS FOR MORE INFORMATION
  - — PAPER EXPLOSION
  - — RISING SALARIES
- AVERAGE SECRETARIAL SALARY 68% HIGHER
- AVERAGE LETTER COST 40% MORE THAN 10 YEARS AGO

Fig. 2

## FUTURE TREND

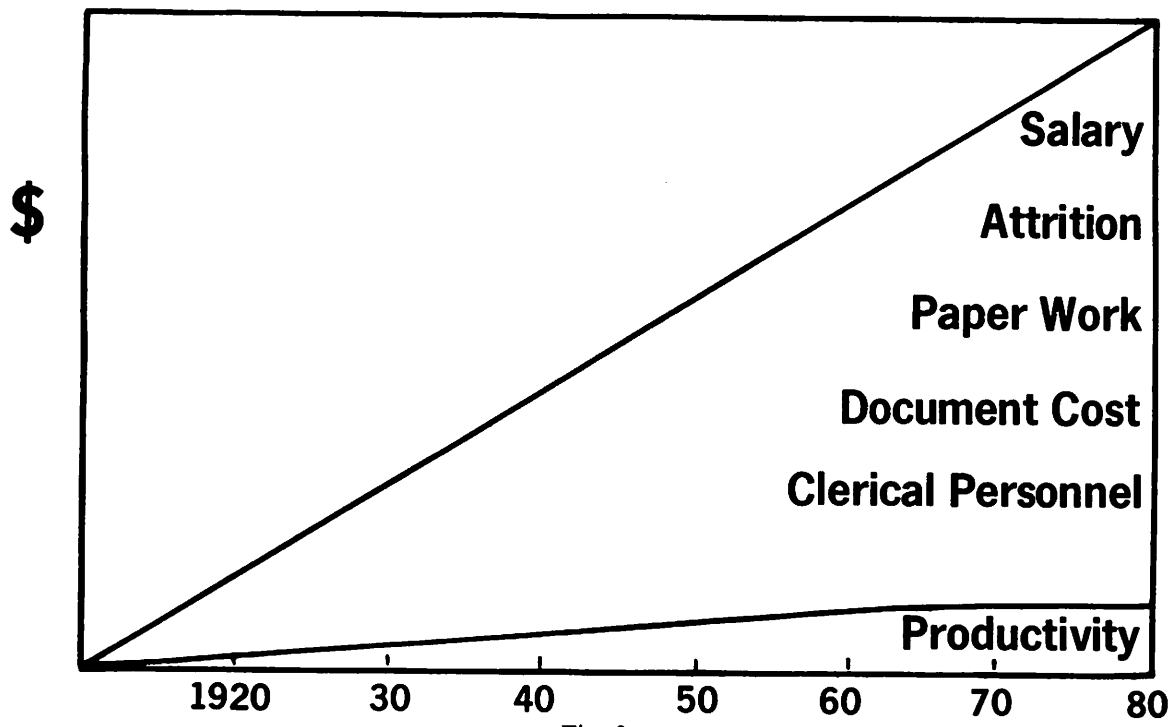


Fig. 3

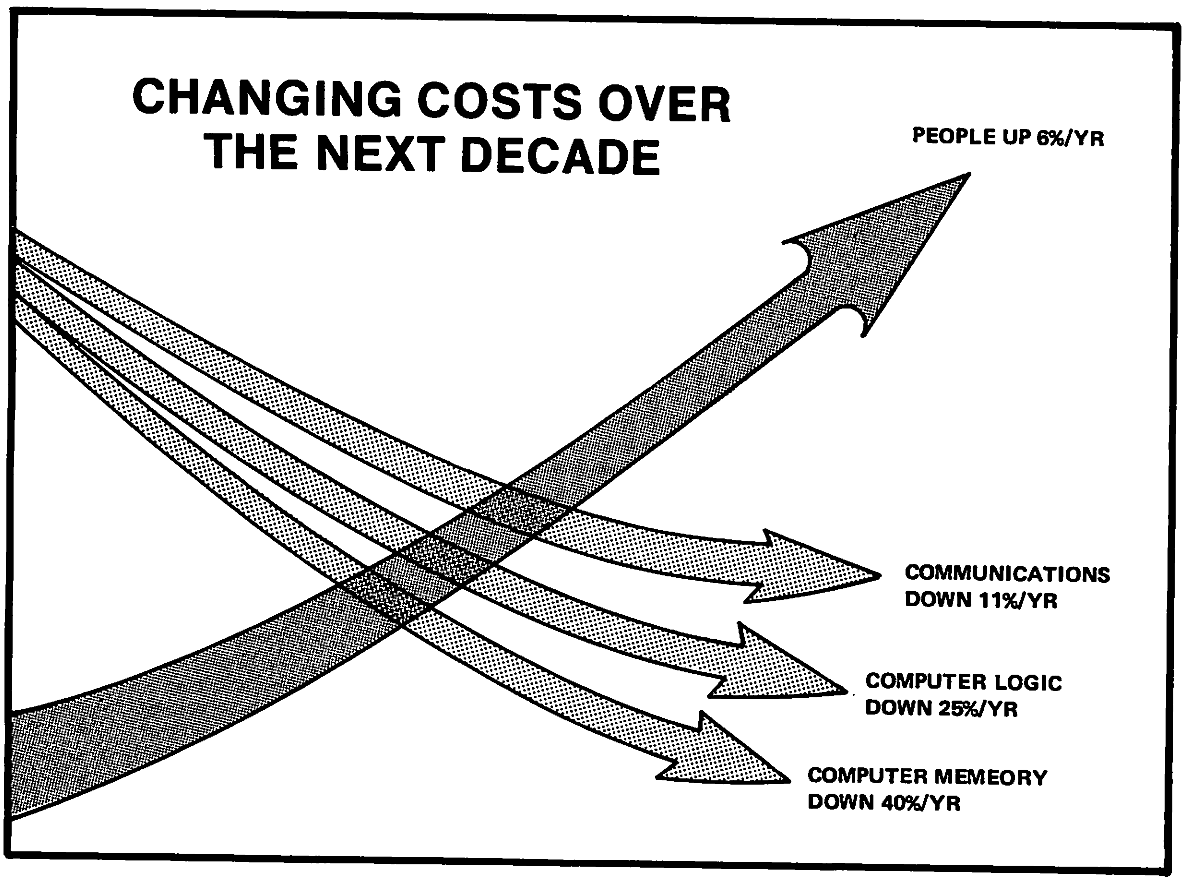


Fig. 4

When you consider that there is not even an industry accepted definition for "Wordprocessing", it comes as no surprise that management level personnel would be somewhat bewildered. At the risk of being presumptuous, I will attempt to define Wordprocessing as a "Work Cycle". This work cycle includes dictation, formatting, typing, text manipulation, proofing, revising, and printing. Based on this definition, Wordprocessing or the processing of words, exists in every office, whether it is automated or not.

Automated Wordprocessing, though, has a number of distinct advantages over the manual method. Cost saving is perhaps the most important and obvious need. It has been demonstrated time and again that Wordprocessing can return a substantial savings. One recent example appeared in the September 1978 issue of Datamation magazine.[4] The cost per page using the manual approach was listed at $11.24, whereas, with automated Wordprocessing, the cost was $6.42. This represents a savings of $4.82 per page or nearly 43%. In addition to these savings, there are real savings in time when using automated Wordprocessing. Typewriter prepared documents were estimated at 75 minutes per page to produce the first draft, coordination draft, and final copy.[4] With Wordprocessing, however, the same steps required only 27 and one half minutes per page, a savings of 47 and one half minutes per page or 63%. With today's higher paid office personnel, such savings in time represent significant savings in money. As a result, the need to hire additional personnel may be deferred due to greater individual productivity (Fig. 5).

You can appreciate that while Dataprocessing costs in 1973 were $26 billion, those of office administration processing were $42 billion.[5] This indicates that a great deal of work can be done to automate the modern office.

## THE SOLUTION

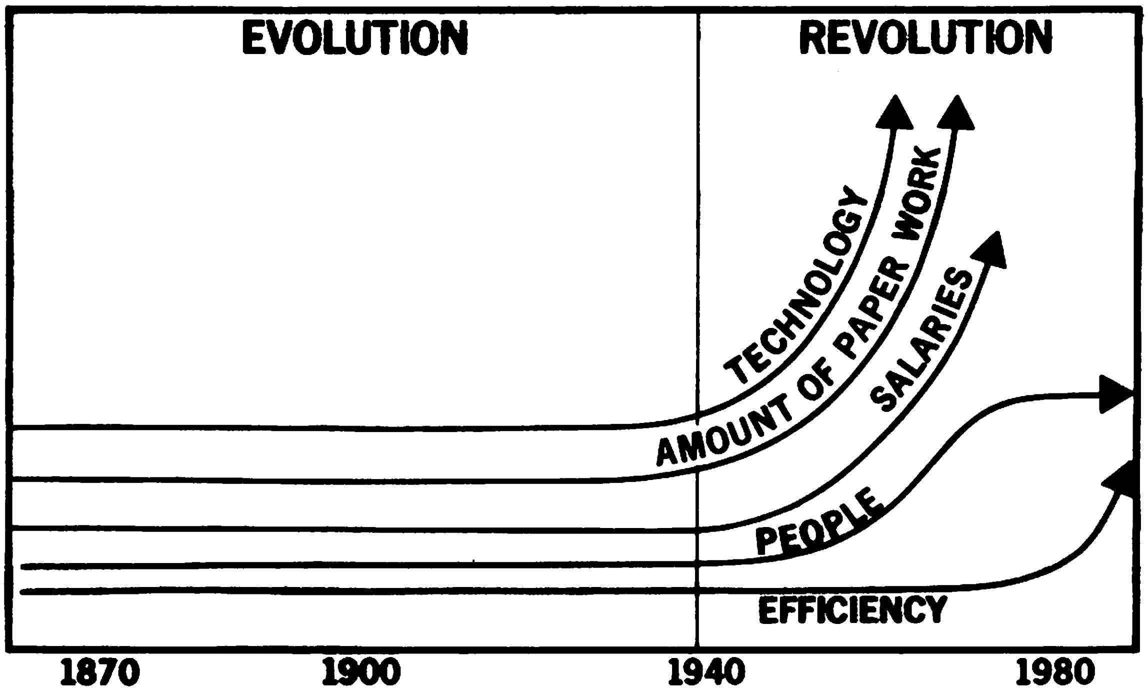


Fig. 5

Consider also the quality and speed of a good Wordprocessing system. Not only does it produce error-free originals, but every character is captured and stored, preserving the document for future use and thereby shortening the turn-around time for revisions. Every revision cycle is apt to require progressively less and less time with every output copy being flawless.

Today's Wordprocessing marketplace has every imaginable type of product available. These range from simple standalone hardware units to massive software systems which run on the largest of main-frame computers. The technology is such that whether you are generating single page letters or complex technical publications, systems are available.[6] Your organization may or may not already be involved with one or more of these systems, but in choosing a future system, it is essential that it be an extension of techniques with which the users are already familiar.

The fact is, there are only four (4) basic types of Wordprocessing systems available today.[7] An understanding of these categories will help in evaluating the pros and cons of your existing/future systems.

1. Standalone Hardcopy-
   This includes all types of automatic typing devices that use magnetic media to capture and store what has been typed. Although the single-unit cost is modest, these systems cannot be integrated for several operators to share resources. Their capacity is limited and they are very inflexible. These units are generally used for short documents.

2. Standalone Video Display-
   This equipment possesses a video display capability ranging from part of a line to a full 66-line legal-size page. At times this type of equipment has text editing capabilities and may include some Data Processing capabilities. A number of products in this category are direct crossovers from the intelligent terminal portion of the Data Processing industry. Again, while the capabilities of this type of equipment is greater than for standalone hardcopy units, it is by nature non-shareable between users and still limited in capacity and flexibility.

3. Time-Shared Services-
   This is an arrangement in which one or more terminals (which may be owned or leased) are hooked into a service company's computer to provide an economical alternative for users who occasionally require sophisticated capabilities, or who are taking a tenative first step toward Wordprocessing technology.

4. Shared-Logic Processing-
   Several independent stations are linked to a single central processing unit for increased capability and storage at a lower cost per station. This is the type of system we might envision with an HP 3000 at the hub.

Few organizations set out to procure a computer based on their Wordprocessing needs. Instead, they quite often have an established Data Processing department running payroll, general ledger, cost accounting or other "standard" computer applications. Let's examine the benefits of adding a Wordprocessing system to this already existing hardware.

First of all, Wordprocessing allows greater use of existing equipment, which may be idle part of the time anyway. Because we are very cost conscious, this type of extra duty from hardware already in place rarely meets with criticism. And since few, if any, equipment changes would be required, no interface problems are encountered in dealing with another hardware vendor either. A great deal of time and effort is generally consumed just familiarizing with new equipment. So, it is a real asset if the end users do not have to be retrained on a new piece of equipment when a Wordprocessing system is installed and they are more likely to respond favorably to new systems implemented on existing equipment.

When looking at Wordprocessing then, what are important considerations? Two main areas of concern are paramount:

1. It must be typist oriented.
2. It must be fully integrated.

At Comarco, we have been producing text editing software for the U. S. Navy for a number of years. When we undertook development of WORDWRIGHT©, our general purpose commercial Wordprocessing system, we placed these two items at the top of our list of requirements. Data Processing personnel can get along fine with "computer" terminology. Secretaries and typists though do not (and do not need to) speak "computerese". They simply do not respond well to such systems. In terms of being fully integrated, we felt that it was important that the user not always be changing gears, so to speak. You no doubt have experienced the frustration of constantly skipping around from one "sub-system" to another. Because this situation is common, some corporate decision-makers apparently have remained cautious about Wordprocessing.

This is not to say that systems which are not typist oriented or fully integrated cannot be forced upon them, but in all fairness, wouldn't it be best to allow the dictates of the end user to determine which product will be selected? This process begins with defining the task to be performed by the equipment and software.[7] Second, consideration should be given to the organization's internal operations and established procedures. "Because of the normal resistance to change, any new technology introduced should, where possible, be an extension of techniques the user is already familiar with."[6] Not just the managers, but the people who will actually be using the system should be consulted and their views explored in depth.

Only after the above steps have been completed should an investigation of what is available be made or a design for an in-house developed system started. If you are looking at purchasing or leasing, bear in mind that special features, not available on all Wordprocessing packages, may be required to provide properly composed technical reports. For instance, automatic alignment of decimal points for tabular data, right-hand justification or automatic hyphenation. You might also require a dual pitch feature for ten or twelve characters per typed line inch.

Does the system provide for true document management? The ability of a Wordprocessing system to track existing documents and catalog names, dates and current status eliminates the associated manual bookkeeping. So, in evaluating whether the system is truly integrated, this is one area not to be overlooked. Some organizations maintain a very large text base including many documents not frequently referenced. Due to a limited amount of on-line storage, it is essential that the Wordprocessing system include some sort of archival and retrieval capability. Again, this feature may not be available on all systems.

All documents require formating. Formatting includes the setting of margins, tab positions, indentation, page lengths, etc. A big asset in a Wordprocessing system is the ability to create a standard set of formats—formats which are flexible, easy to define and use. Once defined, these formats insure style uniformity from document to document. Some Data Processing oriented systems do not allow for stored formats. Instead, the user must embed formatting directives directly into the text. This may be acceptable for some applications, but you will find that non-dataprocessing oriented secretaries and typists will not be able to achieve the speed in inputting text for which they are being paid. A system which provides the ability to utilize stored formats reduces operator keystrokes, detail coding of text and formatting errors. It is at the user level that a typist oriented Wordprocessing system must work well.

Since many organizations produce "form" letters, at least two features must be considered. 1) Does the system provide an integrated name and address directory which can be incorporated into the body of the text? In providing this capability, is text composed around variable length inserts? 2) Are user definable prompts available for a fill-in-the-blank effect? Both of these items are valuable options which can speed form letter preparation and at the same time eliminate the need for the user to browse through the text looking for the locations of the changes.

In addition to the ability to input text with efficiency and speed, is the need for rapid error free original hard-copy. A system should provide for output on various devices. For those who require high quality character printing the system should interface to daisywheel or cup type terminal printers. When such quality is not required, such as preliminary drafts, the system should be able to drive a high speed line printer. Some users of Wordprocessing are interested in producing high quality camera ready copy. This requires the addition of a photocomposer to the basic system configuration. However, as you can see from this paper, the quality far surpasses anything which can be produced on the standard daisywheel printer or even the system line printer. The ability of a Wordprocessing system to prepare text for output to a photocomposer is quite rare on a hardware configuration like the HP 3000. If your organization's requirements include this feature, you will be able to take advantage of such things as variable character spacing and a choice of multiple character fonts and sizes.

Automated Wordprocessing has remained relatively simple to use. Systems have been introduced which vary from basic editors to WORDWRIGHT©, which is sophisticated and comprehensive. Which one you choose will be a matter your organization will have to consider carefully. One thing is for sure—Wordprocessing is a very dynamic field. The time is right for automated Wordprocessing!

## REFERENCES

1. Wohl, Amy D.,"What's Happening in Word Processing",DATAMATION,April 1977, pp.65-74.
2. Burns, J. Christopher,"The Evolution of Office Information", DATAMATION,April 1977,pp.60-64.
3. WORDPROCESSING-A Survey of Perceptions and Attitudes of Top Management by Philip F. Shannon,NEWSWEEK,New York,1977.
4. Carls,C. B.,"Getting Ready for Word Processing's Second Generation", DATAMATION,September 1978,pp.139-144.
5. Strassman, Paul A.,"Stages of Growth",DATAMATION,October 1976,pp.46-50.
6. SURVEY OF COMPUTER-ASSISTED WRITING AND EDITING SYSTEMS by P.I. Burnam, Technical Editing and Reproduction Ltd.,London,1977,pp.48.
7. Winkler, Michael W.,"A Framework for Selecting the 'Right' Word-Processing System",Technical Communication,First Quarter 1978,pp.14-17.

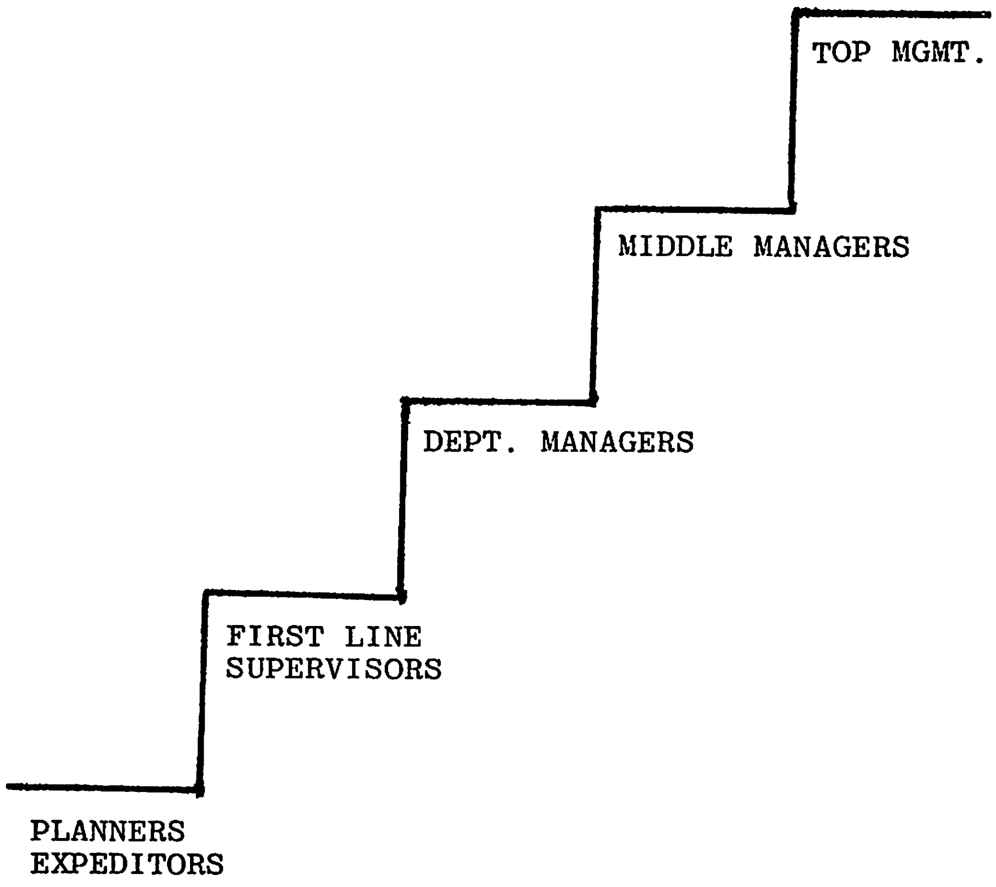
# GRAPHICS IN BUSINESS

PAUL COOPER
SYSTEMS ENGINEER
TULSA, OKLAHOMA

Although there is no graphics software language currently supported on the HP3000, many products are offered to give the 3000 user excellent business graphics capability. One of these products, the HP2648A, especially lends itself to business graphics because of the terminal oriented nature of the HP3000.

My purpose here is to explore some ways of applying graphics to business applications. In the past, graphics have been primarily used in scientific and engineering applications because graphics technology was developed for those kinds of jobs. The business data processing area is on the brink of a new era in data reporting through the use of graphics.

Looking at the users of information output from business systems, a hierarchy of job positions can be seen as shown in the following chart:



MANAGEMENT REPORTING

STAIRSTEPS

This hierarchy will be referred to throughout this presentation as possibilities for graphics applications are explored.

Of prime consideration for a planner/expeditor type person is "How is the output of my department doing with respect to schedule?". In the past, reports have been available to show at any given instant, what the schedule number of output units are versus actual output. Typically, there have been time lags between reported data and actual output data. These time lags make scheduling very difficult. In addition, data of this type ignores the time continuum needed to know where one stands with respect to remaining schedule and time.

With a simple graph, one can easily tell at a glance not only where one stands today but also over a time continuum. The fact that the output is coming from a terminal oriented system allows the user the ability of reporting this information on a "pseudo-real-time" basis, thus eliminating the confusing time lags between reported and actual.

Figure 1 would be a good example of a planner's tool from a batch type system. The planner should expect to get this type of information the day after the output actually takes place. The list would point out that 181 units were output as of the end of M-Day 13, but it would not indicate how the production line is doing so far during the current day.

It may also be a good example of a planner's tool from an online system. In this case, the information would be much more timely and therefore much more useable. The information should be expected to be available at any hour of any M-Day, and should be expected to reflect output already finished during the current day.

Graph 1 shows this same planner's tool graphically. Note that not only does the graph indicate production to date, but also gives an excellent trend line to let the planner know what needs to be done between now and the end of the production period. It answers questions like, "Do I need to recommend overtime to management", or "Do I have a problem with material flow?".

Notice also, that during the current day an indication is given as to whether the ahead/behind trend is being impacted by the current day's production. This indication is given by the slope of the graph line during the current day.

While this type of detail information is important to the planner, the first level supervision may well be more interested in more long-term type information. Again the same progression of data reporting can be seen at this level. Figure 2 is an example of year to date type information. The same kind of "timely information" parameters apply to this reporting level.

In addition, some modelling is in order at this level to indicate to management what the year end production will be if projected at current production levels. There are a number of good forecasting methods which may be used to extend the production line out through the end of the year or beyond. This kind of information is essential for good manpower planning and for good material control.

Another topic of great importance to first line supervision and also to department managers is that of productivity. One measure of productivity is "dollars per manhour". The method I've chosen to work with here is number of dollars transferred from a production department versus the amound of labor applied. Going back to our earlier example, I've applied an arbitrary price to the units in the graph. In addition, I've established an arbitrary standard time for producing one of the units. This new data is represented in chart form in figure 3. Although this is a very simplified example of productivity measurement, the following conclusion can easily be reached: Data of this type can be understood much more simply and easily when it is presented graphically. Again, the graph which results from this data shows a trend over a time continuum. This trend cannot be readily gleaned from the same data presented in the traditional columnar form.

Efficiency is another topic of importance to management at this level. Efficiency is derived by comparing actual time spent on a unit to a predetermined standard time. Efficiency is usually shown as a percentage, which is easily understood as a number and does not need to be depicted graphically. But managing efficiency cannot be accomplished by a snapshot of any one particular time period. Here again, a group of percentages spread over time is needed to spot trends. With a computer, the current day's efficiency can be reported with respect to past days, and probably more important, a moving average can be derived and displayed quickly. This kind of information is very useful for spotting trends in efficiency to enable management to be responsive to developing problems. Figure 4 shows data of this type.

Middle and top level management are interested in more long-term information. Managers at this level are interested in topics like "What are my projected sales over the next 24 months?" or "How is my business doing compared to others in this industry?". Using forecasting algorithms, the computer can easily be made to project future trends based upon past performance and expected future trends in the market place. Adding graphics to these projections produces a pictoral view of future trends, making the data very easy to understand. The whole idea of making projections (modelling) is to enable management to set a direction for future business. Questions about manpower, material, and financial requirements can be answered in this manner. It is incumbent upon data processing systems to report this information in any easy to understand and readily useable form. Graphics capabilities make this requirement easy to fulfill.

Figure 5 is an example of forecasted sales in the form one might expect to see from conventional systems. The corresponding graph makes the data much more digestable and easier to comprehend at a glance.

Figure 6 is an example of trend analysis. This type of data answers the question about how my business is doing with respect to the rest of the industry. Industry data is available from Dunn and Bradstreet or the Securities and Exchange Commission. A manager can place his company in perspective over time to see how the company is performing and whether the performance is improving or degrading over time.

To this point we've looked at reporting to all five levels of management through the use of linear graphs. A short mention should be given to the ease of generating these graphs on an HP2648. The fact is that the HP2648 has an autoplot function imbedded in firmware within the terminal. The autoplot function is complete with a menu to describe the X and Y axes. Once the axes are described, the terminal will draw the graph, with tic marks and an optional grid. Data for the graph can be obtained either from the display or from the HP3000. With this versatility, linear graphs become very easy to generate.

The next area of importance to business graphics is generating other types of general purpose report forms. One example of this type of report is a pie chart. Unlike linear graphs, there is no firmware driven menu to generate a pie chart. However, with a little imagination, a simple program can be written to accomplish this feat. Appendix 1 to this report is a source listing of a Fortran program which generates a pie chart. Going through the program, one can see that the terminal graphics are controlled through the use of "escape sequences". The sequences are easily edited into a source program or data file through the use of the editor. This program is included here to demonstrate the relative ease of generating business graphics, and is intended to stimulate the imagination.

Over the past few years, the data processing industry has seen a migration of trends. From batch systems to interactive systems; from main frame processing to distributed processing. The industry has also seen a shift in the way people think about computers. The computer is losing its' shroud of mystery and is becoming a tool to aid businessmen accomplish goals and plan for the future. Data reporting through graphics is a natural extension to these trends and opens a whole new methodology for data reporting.

FIGURE 1

NOV. '78 ABC PRODUCTION TOTAL SCHEDULE: 330

| M-DAY | SCHEDULED | CUM. | ACTUAL | CUM. OUT |
|---|---|---|---|---|
| 1 | 15 | 15 | 14 | 14 |
| 2 | 15 | 30 | 18 | 32 |
| 3 | 15 | 45 | 13 | 45 |
| 4 | 15 | 60 | 12 | 57 |
| 5 | 15 | 75 | 15 | 72 |
| 6 | 15 | 90 | 8 | 80 |
| 7 | 15 | 105 | 10 | 90 |
| 8 | 15 | 120 | 8 | 98 |
| 9 | 15 | 135 | 10 | 108 |
| 10 | 15 | 150 | 5 | 113 |
| 11 | 15 | 165 | 14 | 127 |
| 12 | 15 | 180 | 16 | 143 |
| 13 | 15 | 195 | 8 | 151 |
| 14 | 15 | 210 | | |
| 15 | 15 | 225 | | |
| 16 | 15 | 240 | | |
| 17 | 15 | 255 | | |
| 18 | 15 | 270 | | |
| 19 | 15 | 285 | | |
| 20 | 15 | 300 | | |
| 21 | 15 | 315 | | |
| 22 | 15 | 330 | | |

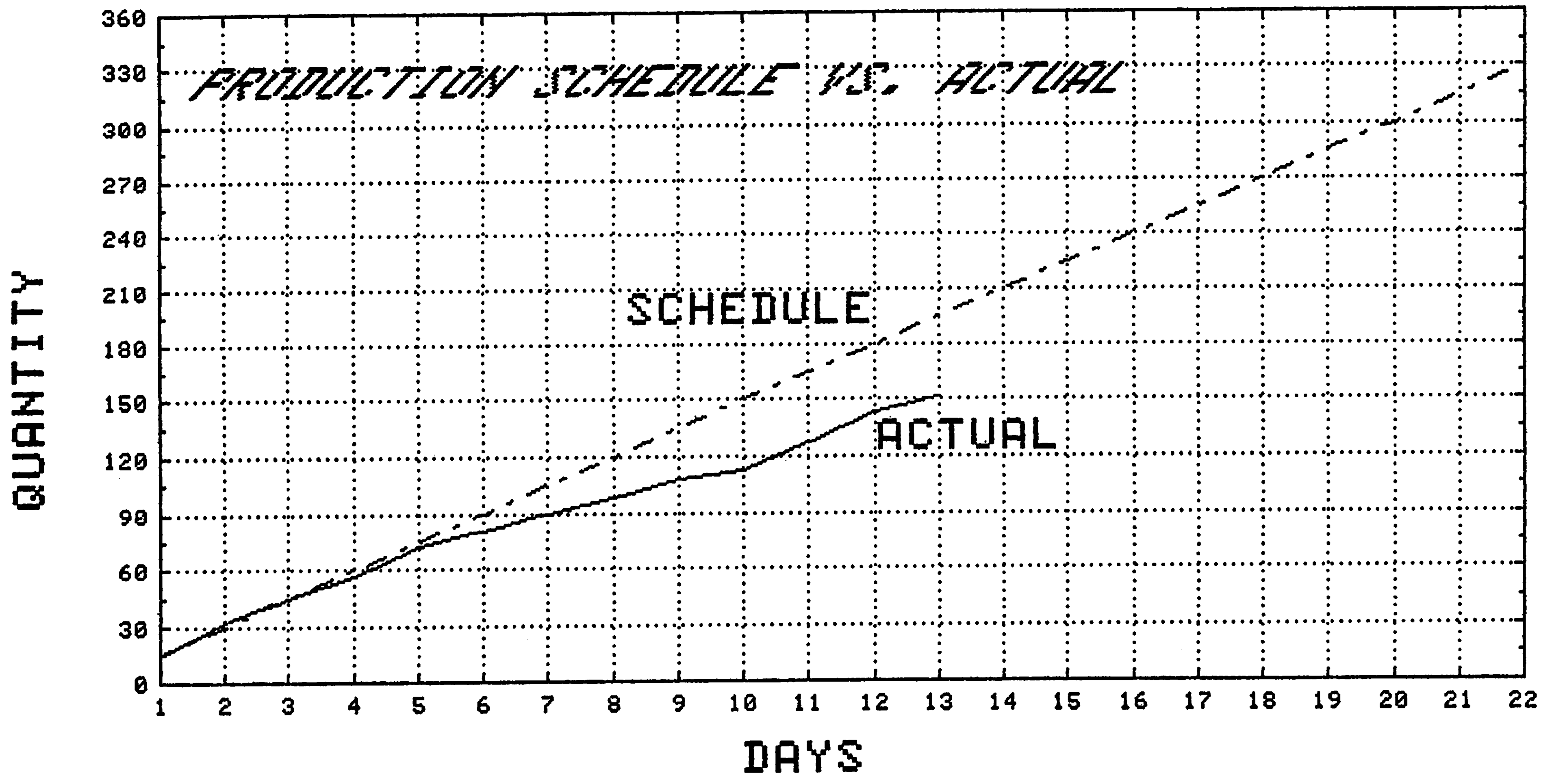

PRODUCTION SCHEDULE VS. ACTUAL
QUANTITY
360
330
300
270
240
210
180
150
120
90
60
30
0
SCHEDULE
ACTUAL
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
DAYS

FIGURE 2

1978 ABC PRODUCTION TOTAL SCHEDULE: 3960

| MONTH | SCHEDULED | CUM. | ACTUAL | CUM. OUT |
|---|---|---|---|---|
| 1 | 330 | 330 | 325 | 325 |
| 2 | 330 | 660 | 340 | 665 |
| 3 | 330 | 990 | 298 | 963 |
| 4 | 330 | 1320 | 308 | 1271 |
| 5 | 330 | 1650 | 240 | 1511 |
| 6 | 330 | 1980 | 290 | 1801 |
| 7 | 330 | 2310 | 270 | 2071 |
| 8 | 330 | 2640 | 310 | 2381 |
| 9 | 330 | 2970 | 288 | 2669 |
| 10 | 330 | 3300 | | |
| 11 | 330 | 3630 | | |
| 12 | 330 | 3960 | | |

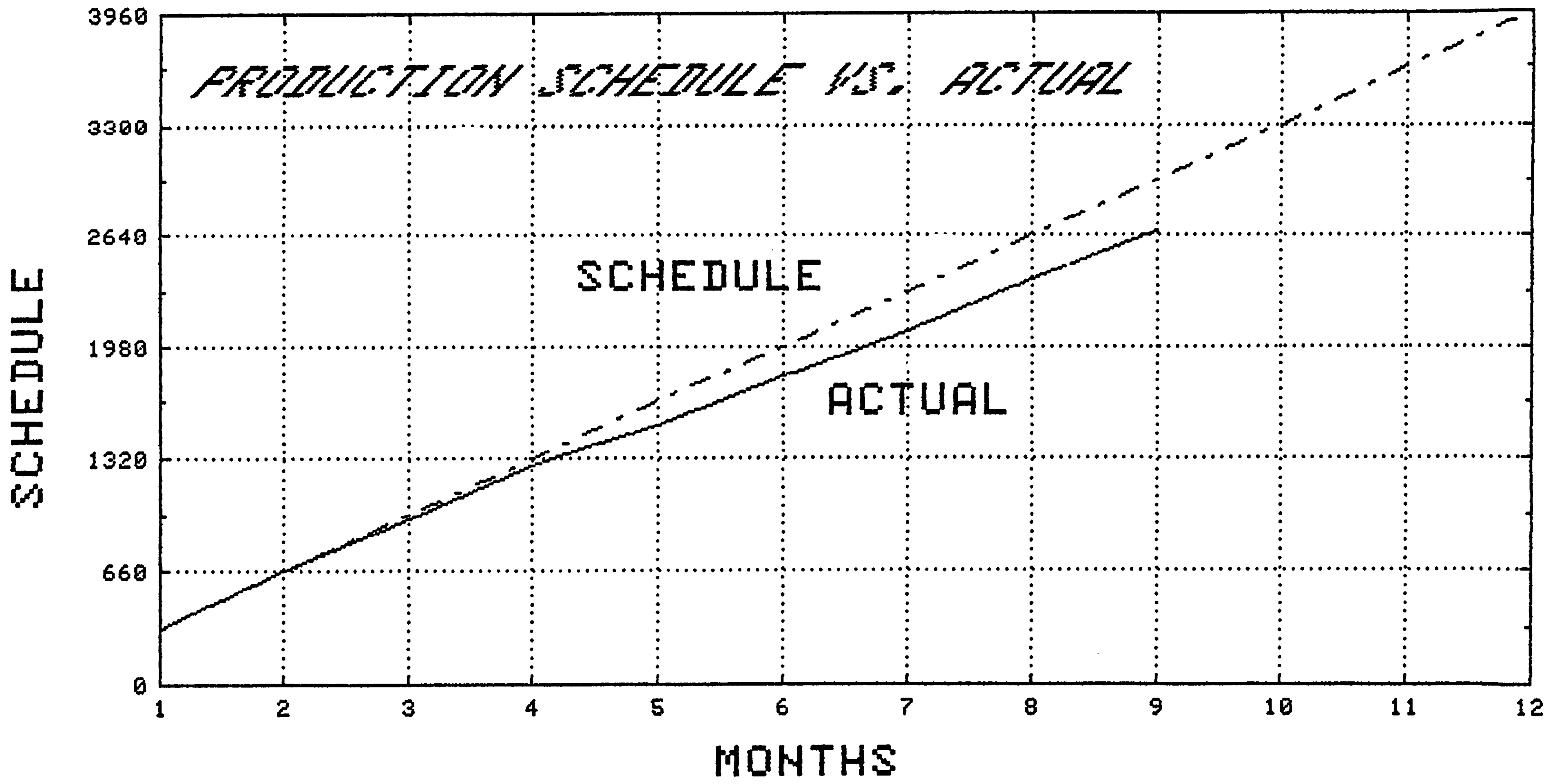

PRODUCTION SCHEDULE VS. ACTUAL
SCHEDULE
ACTUAL
SCHEDULE
MONTHS
3960
3300
2640
1980
1320
660
0
1
2
3
4
5
6
7
8
9
10
11
12

FIGURE 3

STANDARD COST PER UNIT: $200.00

STANDARD TIME PER UNIT: 7.5 HOURS

| MONTH | SCHEDULED OUTPUT | ACTUAL OUTPUT | STD. TIME APPLIED | ACTUAL TIME APPLIED | STD. $ OUT | ACTUAL $ OUT | STD. $/ HOUR | ACTUAL $/ HOUR |
|---|---|---|---|---|---|---|---|---|
| 1 | 330 | 325 | 2475 | 2480 | 66000 | 65000 | 26.67 | 26.2 |
| 2 | 330 | 340 | 2475 | 2510 | 66000 | 68000 | 26.67 | 27.1 |
| 3 | 330 | 298 | 2475 | 2540 | 66000 | 59600 | 26.67 | 23.5 |
| 4 | 330 | 308 | 2475 | 2540 | 66000 | 61600 | 26.67 | 24.3 |
| 5 | 330 | 240 | 2475 | 2470 | 66000 | 48000 | 26.67 | 19.4 |
| 6 | 330 | 290 | 2475 | 2462 | 66000 | 58000 | 26.67 | 23.6 |
| 7 | 330 | 270 | 2475 | 2470 | 66000 | 54000 | 26.67 | 21.9 |
| 8 | 330 | 310 | 2475 | 2520 | 66000 | 62000 | 26.67 | 24.6 |
| 9 | 330 | 288 | 2475 | 2536 | 66000 | 57600 | 26.67 | 22.7 |
| 10 | 330 | | 2475 | | 66000 | | 26.67 | |
| 11 | 330 | | 2475 | | 66000 | | 26.67 | |
| 12 | 330 | | 2475 | | 66000 | | 26.67 | |

A-19.15

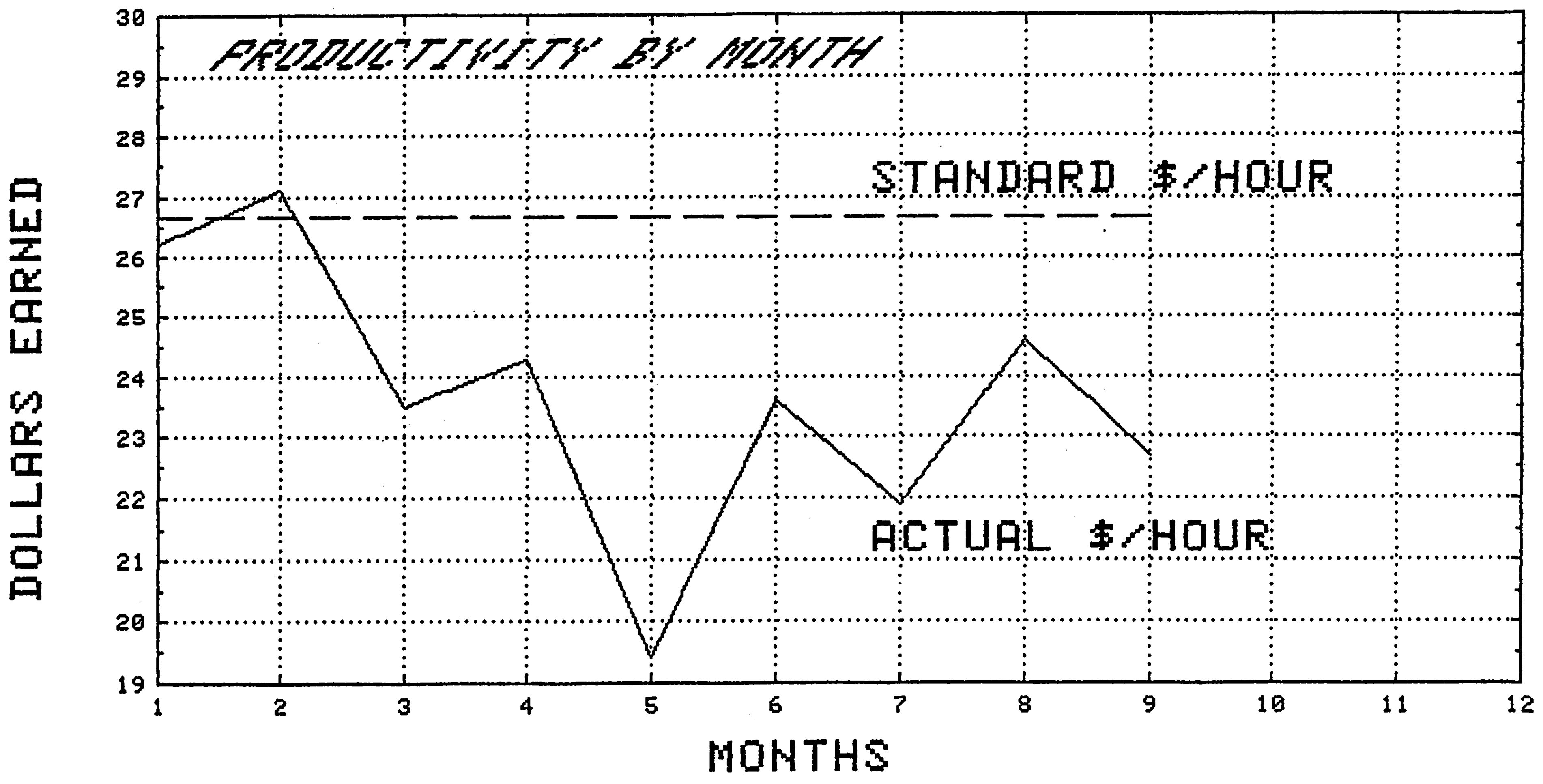

PRODUCTIVITY BY MONTH
STANDARD $/HOUR
ACTUAL $/HOUR
DOLLARS EARNED
MONTHS
30
29
28
27
26
25
24
23
22
21
20
19
1
2
3
4
5
6
7
8
9
10
11
12

FIGURE 4

NOV. '78 ABC EFFICIENCY

| M-DAY | HOURS EXPENDED | HOURS EARNED | % EFFICIENCY | MOVING AVERAGE |
|---|---|---|---|---|
| 1 | 104 | 105 | 100.9 | 100.9 |
| 2 | 112 | 135 | 120.5 | 111.1 |
| 3 | 108 | 98 | 90.7 | 101.2 |
| 4 | 102 | 90 | 88.2 | 100.4 |
| 5 | 112 | 112 | 100.0 | 100.3 |
| 6 | 88 | 60 | 68.2 | 95.8 |
| 7 | 80 | 75 | 93.8 | 95.6 |
| 8 | 88 | 60 | 68.2 | 92.6 |
| 9 | 88 | 75 | 85.2 | 91.8 |
| 10 | 48 | 38 | 79.2 | 91.2 |
| 11 | 104 | 105 | 100.9 | 92.2 |
| 12 | 108 | 120 | 111.1 | 94.0 |
| 13 | 72 | 60 | 83.3 | 93.3 |
| 14 | | | | |
| 15 | | | | |
| 16 | | | | |
| 17 | | | | |
| 18 | | | | |
| 19 | | | | |
| 20 | | | | |
| 21 | | | | |
| 22 | | | | |

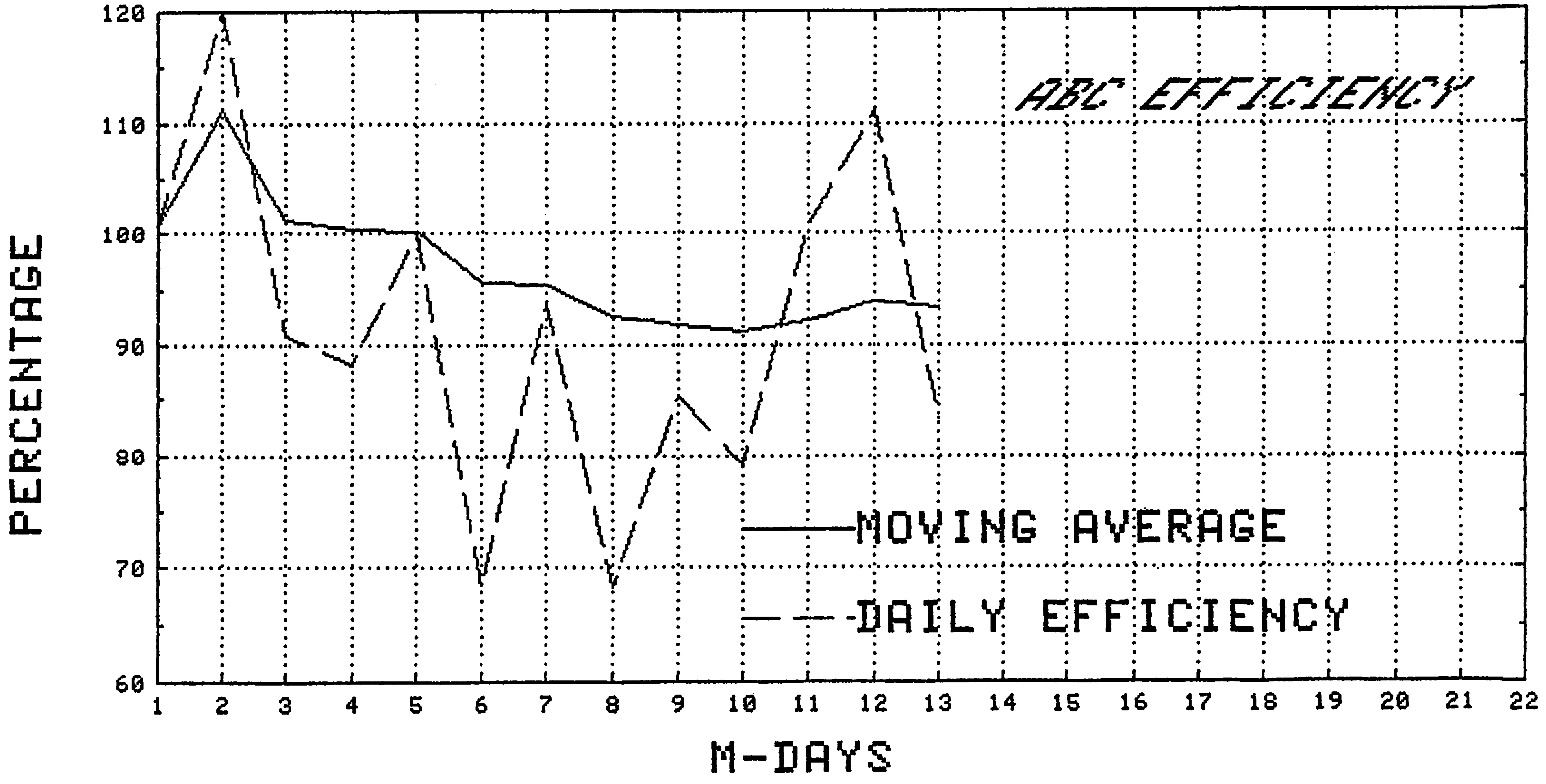
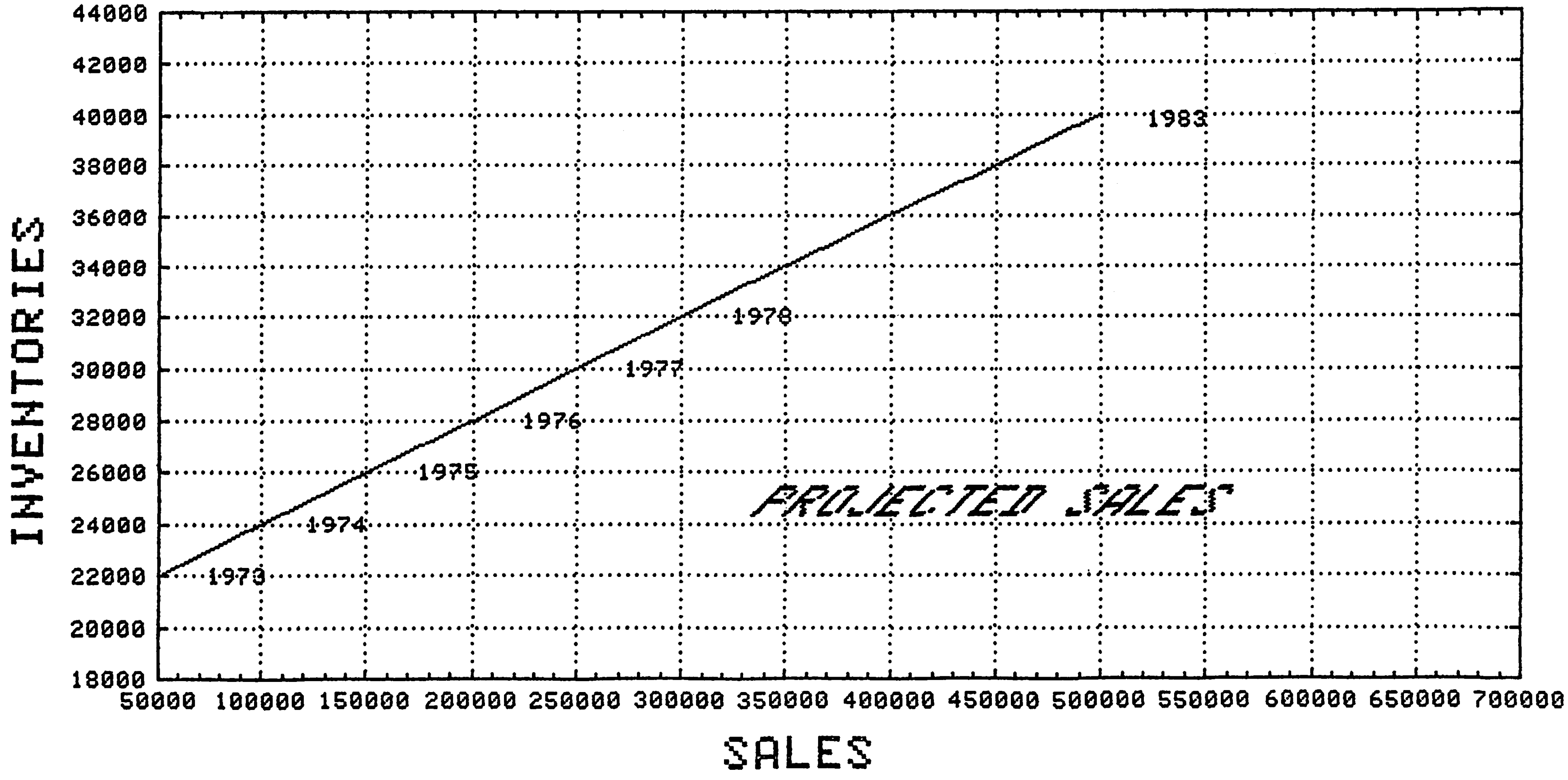

ABC EFFICIENCY
PERCENTAGE
120
110
100
90
80
70
60
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
M-DAYS
MOVING AVERAGE
DAILY EFFICIENCY

FIGURE 5

FORECASTED SALES AS A FUNCTION OF INVENTORY

| YEAR | SALES | INVENTORY |
|---|---|---|
| 1973 | 50,000 | 22,000 |
| 1974 | 100,000 | 24,000 |
| 1975 | 150,000 | 26,000 |
| 1976 | 200,000 | 28,000 |
| 1977 | 250,000 | 30,000 |
| 1978 | 300,000 | 32,000 |
| 1979 | 350,000 | 34,000 |
| 1980 | 400,000 | 36,000 |
| 1981 | 450,000 | 38,000 |
| 1982 | 500,000 | 40,000 |

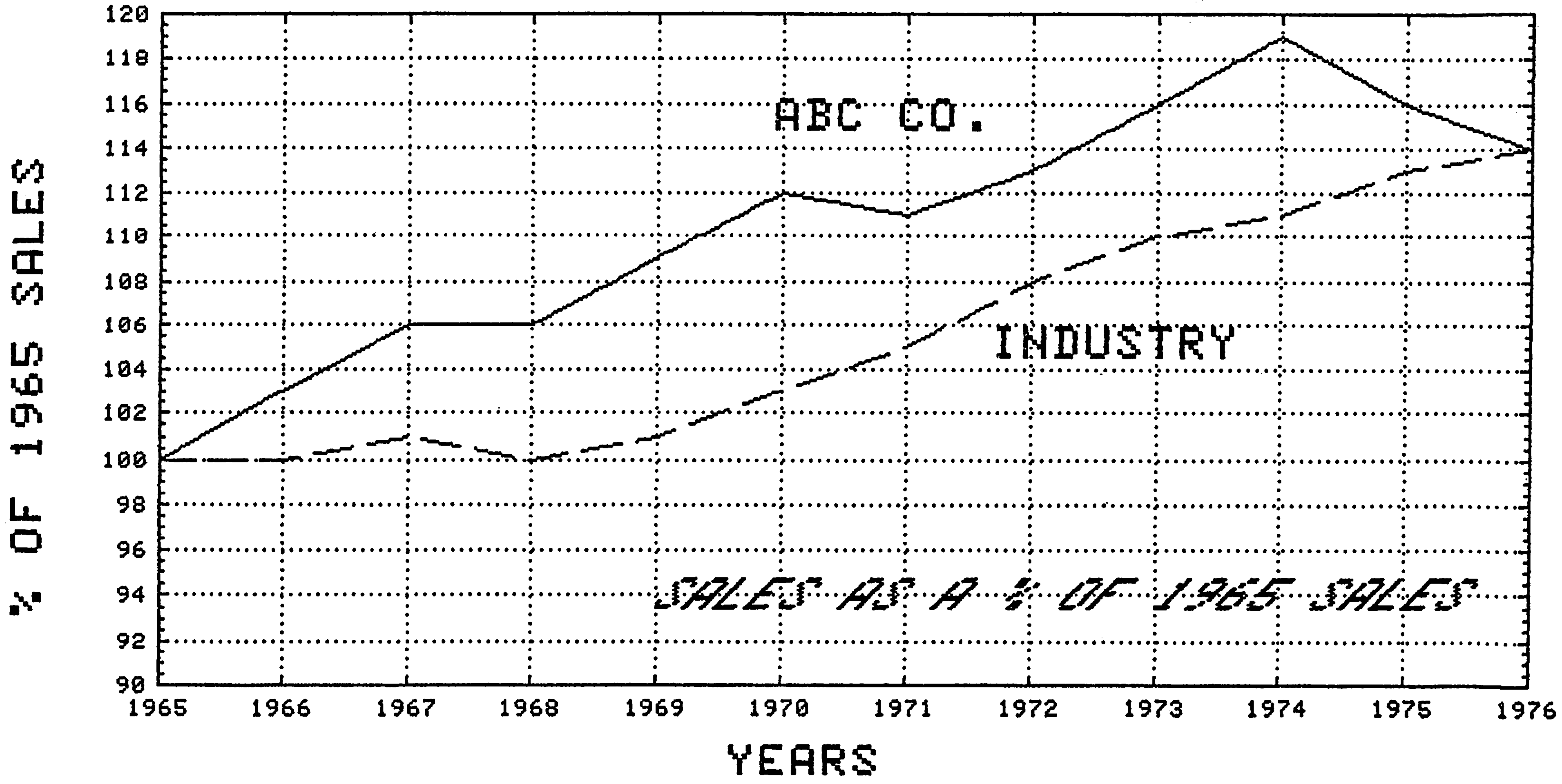


INVENTORIES
44000
42000
40000
38000
36000
34000
32000
30000
28000
26000
24000
22000
20000
18000
50000
100000
150000
200000
250000
300000
350000
400000
450000
500000
550000
600000
650000
700000
SALES
1973
1974
1975
1976
1977
1978
1983
PROJECTED SALES

FIGURE 6

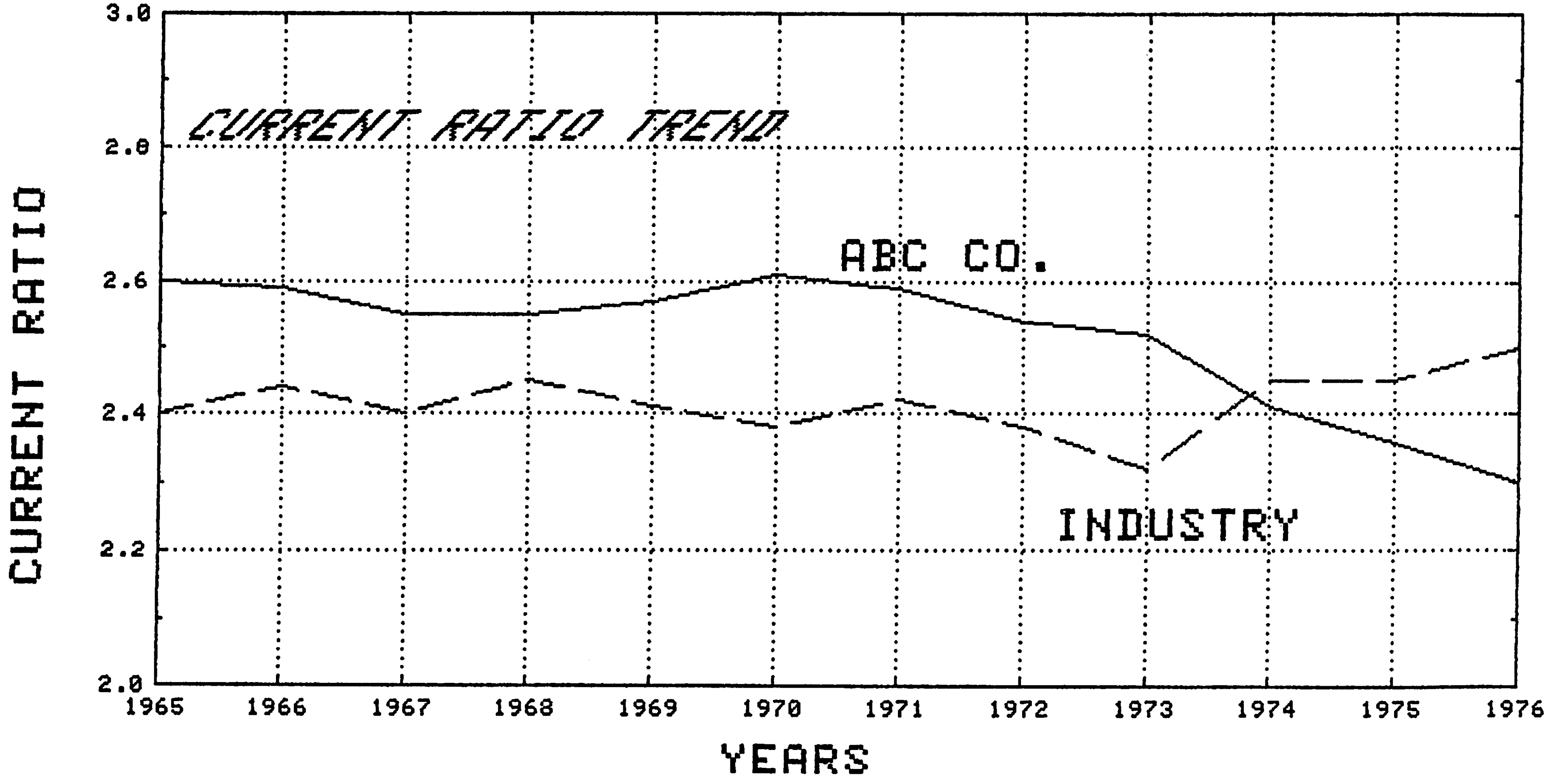
TREND ANALYSIS

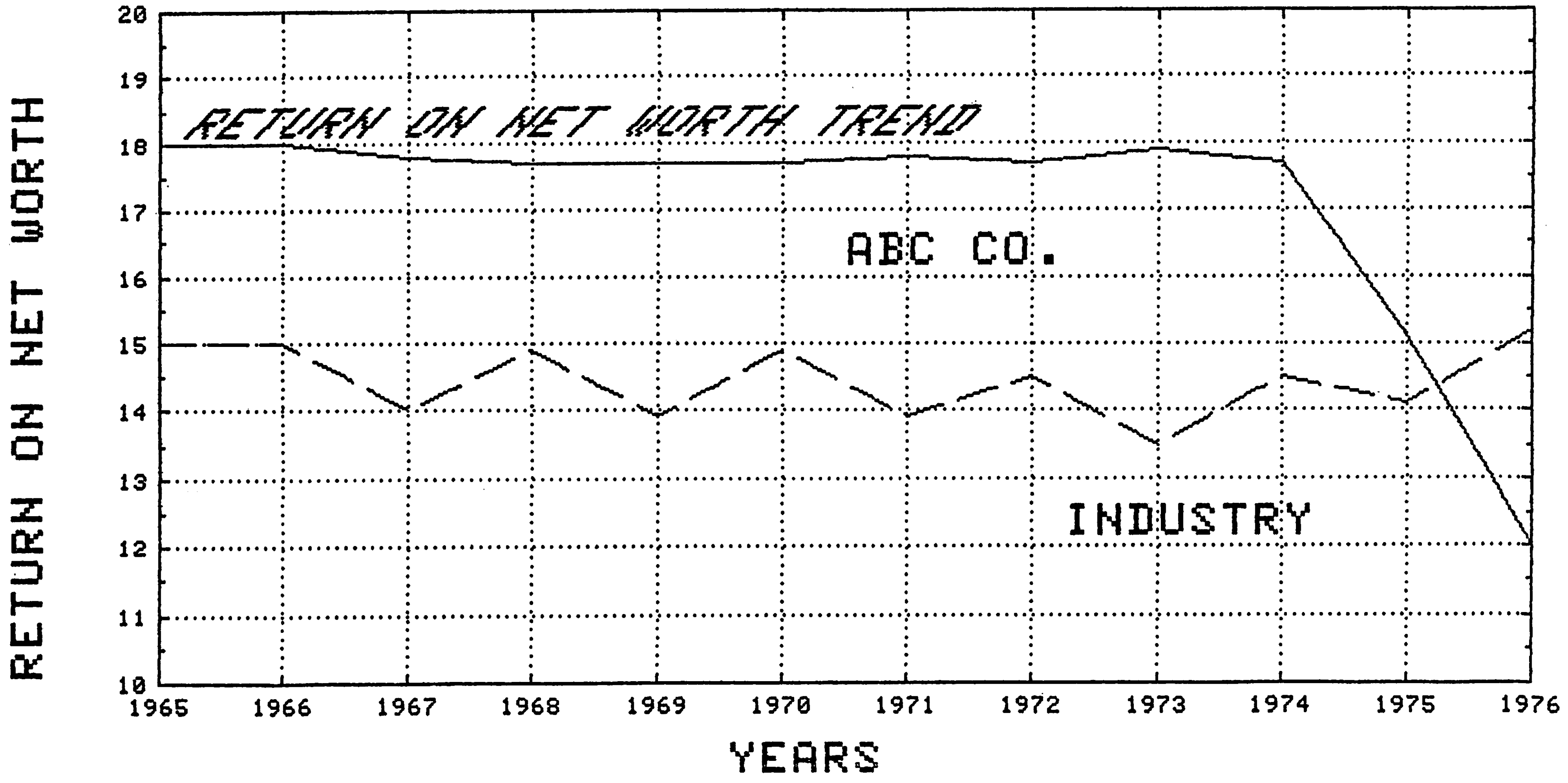
| | Sales as a % of 1965 Sales | | Current Ratio | | Return on Net Worth | |
|---|---|---|---|---|---|---|
| | ABC | INDUSTRY | ABC | INDUSTRY | ABC | INDUSTRY |
| 1965 | 100 | 100 | 2.60 | 2.40 | 18.0 | 15.0 |
| 1966 | 103 | 100 | 2.59 | 2.44 | 18.0 | 15.0 |
| 1967 | 106 | 101 | 2.55 | 2.40 | 17.8 | 14.0 |
| 1968 | 106 | 100 | 2.55 | 2.45 | 17.7 | 14.9 |
| 1967 | 109 | 101 | 2.57 | 2.41 | 17.7 | 13.9 |
| 1970 | 112 | 103 | 2.61 | 2.38 | 17.7 | 14.9 |
| 1971 | 111 | 105 | 2.59 | 2.42 | 17.8 | 13.9 |
| 1972 | 113 | 108 | 2.54 | 2.38 | 17.7 | 14.5 |
| 1973 | 116 | 110 | 2.52 | 2.32 | 17.9 | 13.5 |
| 1974 | 119 | 111 | 2.41 | 2.45 | 17.9 | 14.5 |
| 1975 | 116 | 113 | 2.36 | 2.45 | 15.1 | 14.1 |
| 1976 | 114 | 114 | 2.30 | 2.50 | 12.0 | 15.2 |
| 1977 | | | | | | |



ABC CO.
INDUSTRY
SALES AS A % OF 1965 SALES
% OF 1965 SALES
YEARS
120
118
116
114
112
110
108
106
104
102
100
98
96
94
92
90
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976

A-19.22

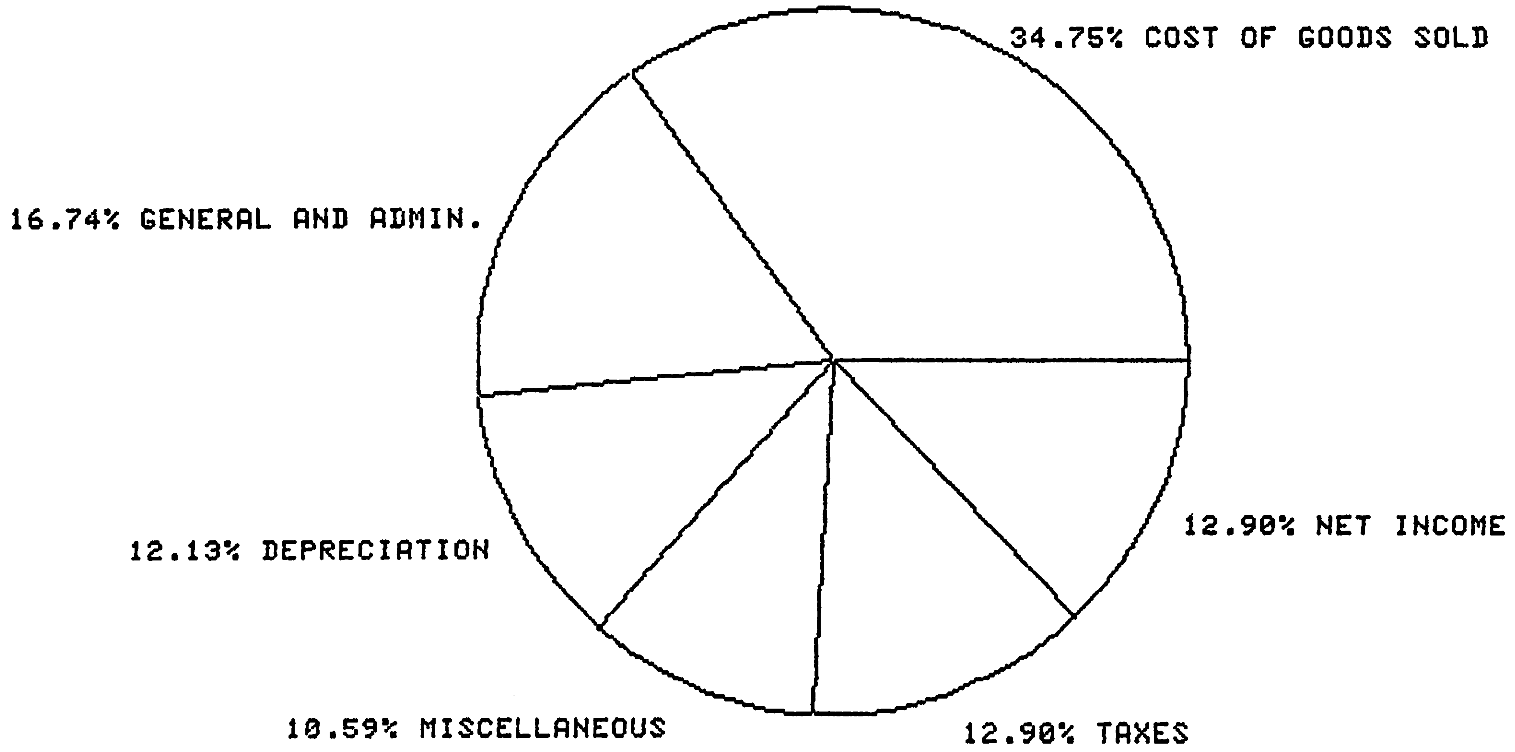

CURRENT RATIO TREND
CURRENT RATIO
3.0
2.8
2.6
2.4
2.2
2.0
ABC CO.
INDUSTRY
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
YEARS



RETURN ON NET WORTH TREND
ABC CO.
INDUSTRY
RETURN ON NET WORTH
YEARS
20
19
18
17
16
15
14
13
12
11
10
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976

HP3000
GRAPHICS

DOLLAR DISTRIBUTION



MON, OCT 30, 1978, 3:54 PM

# APPENDIX 1

```
 L 1/50
$CONTROL USLINIT
      PROGRAM PIE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC
CCC
C       THIS PROGRAM GENERATES A PIE CHART FROM DEFAULT DATA
C       HARDCODED IN THE PROGRAM, OR FROM DATA INPUT FROM THE
C       OPERATOR'S TERMINAL.
C
C       WRITTEN:    OCTOBER, 1978
C                   DEVELOPED FOR SSR'S CONTRIBUTION TO THE
C                    HP3000 USER'S GROUP IN DENVER, COLORADO.
C
C                   PAUL COOPER
C                   SYSTEMS ENGINEER
C                   HEWLETT PACKARD COMPANY
C                   TULSA, OKLAHOMA
C                   (918) 665-3300
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SYSTEM INTRINSIC DATELINE
      CHARACTER*8 HP,GR
      CHARACTER*2 IYES,IBLK1
      CHARACTER*20 LAB(11)
      DIMENSION IDUMY(10,11)
      CHARACTER*27 DATE
      DIMENSION NUMCHARS(11),POSLAB(11)
      DIMENSION VALUES(11),ANGLE(11),IBLK2(1),PERCT(11)
      CHARACTER*5 CLEAR,SIZE1,SIZE2
      CHARACTER*4 INIT,GRTXON,GRTXOF,ALPOF
      CHARACTER*5 COMP
      REAL THETA,RAD
      EQUIVALENCE (IDUMY,LAB)
      EQUIVALENCE (IBLK1,IBLK2)
CCC
C  BEGIN NOW TO INITIALIZE CHARACTER STRINGS AND VARIABLES.
C
      HP=" HP3000 "
      GR="GRAPHICS"
      ALPOF="␛*dF"
      IBLK1="  "
      LAB(1)="DOLLAR DISTRIBUTION "
      LAB(2)="COST OF GOODS SOLD  "
      LAB(3)="GENERAL AND ADMIN.  "
      LAB(4)="DEPRECIATION        "
      LAB(5)="MISCELLANEOUS       "
      LAB(6)="TAXES               "
      LAB(7)="NET INCOME          "
      LAB(8)="                    "
      LAB(9)="                    "
```

```
   L51/100␍
␊   51            LAB(10)="                    "␍
␊   52            LAB(11)="                    "␍
␊   53            VALUES(1)=2210.␍
␊   54            VALUES(2)=768.␍
␊   55            VALUES(3)=370.␍
␊   56            VALUES(4)=268.␍
␊   57            VALUES(5)=234.␍
␊   58            VALUES(6)=285.␍
␊   59            VALUES(7)=285.␍
␊   60            VALUES(8)=0.␍
␊   61            VALUES(9)=0.␍
␊   62            VALUES(10)=0.␍
␊   63            VALUES(11)=0.␍
␊   64            CLEAR="␛*daZ"␍
␊   65            INIT="␛*mR"␍
␊   66            COMP="␛*m3A"␍
␊   67            SIZE1="␛*m1M"␍
␊   68            SIZE2="␛*m2M"␍
␊   69            GRTXON="␛*dS"␍
␊   70            GRTXOF="␛*dT"␍
␊   71      CCC␍
␊   72      C␍
␊   73      C  THE NEXT TWO VARIABLES ARE THE X AND Y CENTER OF THE PIE␍
␊   74      C␍
␊   75            IXCTR=360␍
␊   76            IYCTR=180␍
␊   77            RAD=130.␍
␊   78            RRAD=140.␍
␊   79            INUM=6␍
␊   80      CCC␍
␊   81      C  INITIALIZE TERMINAL AND SET COMPLIMENT MODE.␍
␊   82      C␍
␊   83            WRITE(6,1)CLEAR,INIT,COMP␍
␊   84       1    FORMAT(1X,3(A5))␍
␊   85      CCC␍
␊   86      C  PROMPT OPERATOR FOR VALUE OVERRIDE␍
␊   87      C␍
␊   88            WRITE(6,21)␍
␊   89      21    FORMAT(1X,"VALUES FOR SALES, COSTS, ETC. WILL BE DEFAULTED")
␊   90            WRITE(6,22)␍
␊   91      22    FORMAT(1X,"DO YOU WANT TO OVERRIDE THESE DEFAULTS?")␍
␊   92            READ(5,23)IYES␍
␊   93      23    FORMAT(A2)␍
␊   94            IF(IYES.NE."YE")GO TO 25␍
␊   95      CCC␍
␊   96      C   IF WE'RE HERE WE NEED NEW LABELS FOR THE PIE CHART␍
␊   97      C   AS WELL AS NEW NUMBERS FOR THE SUBPARTS (PIECES).␍
␊   98      C␍
␊   99            DISPLAY "INPUT THE LABELS PROMPTED FOR:"␍
␊  100            DO 49 I=1,11␍
␊/␑
```

```
L101/150
  101               LAB(I)="                    "
  102         49    VALUES(I)=0.
  103         50    FORMAT(A20)
  104         91    DISPLAY "INPUT NUMBER OF PIECES IN WHOLE"
  105               DISPLAY "(NOT MORE THAN 10)"
  106               READ(5,*)INUM
  107               IF(INUM.LE.10)GO TO 90
  108               DISPLAY "GIMME A BREAK!  THE NUMBER CAN'T BE MORE THAN 10"
  109               GO TO 91
  110         90    WRITE(6,53)
  111         53    FORMAT(" INPUT TITLE FOR PIE CHART:")
  112               READ(5,50)LAB(1)
  113               DO 60 I=2,INUM+1
  114               JJ=I-1
  115               WRITE(6,51)JJ
  116         51    FORMAT(/," INPUT CHARACTER STRING FOR LABEL",I3,":  ")
  117               READ(5,50)LAB(I)
  118               WRITE(6,52)JJ
  119         52    FORMAT(/," INPUT VALUE FOR LABEL",I3,":  ")
  120               READ(5,*)VALUES(I)
  121         60    CONTINUE
  122               DO 61 I=2,INUM+1
  123         61    VALUES(1)=VALUES(1)+VALUES(I)
  124         CCC
  125         C  FIND OUT THE # OF CHARACTERS IN EACH LABEL
  126         C
  127         25    DO 110 I=1,11
  128               ICNT=0
  129               DO 109 J=1,10
  130               IF(IDUMY(J,I).EQ.IBLK2(1))ICNT=1
  131               IF(ICNT.EQ.0)GO TO 109
  132               NUMCHARS(I)=J-1
  133               GO TO 110
  134         109   CONTINUE
  135               NUMCHARS(I)=10
  136         110   CONTINUE
  137         CCC
  138         C  COMPUTE THE ANGLES FOR THE PIE CUTS HERE.
  139         C  THE FORMULA IS THE RATIO X IS TO 360 AS PART IS TO TOTAL.
  140         C
  141               ANGLE(1)=0.
  142               DO 28 I=2,INUM+1
  143               ANGLE(I)=(360*VALUES(I))/VALUES(1)+ANGLE(I-1)
  144               POSLAB(I)=((ANGLE(I)-ANGLE(I-1))/2)+ANGLE(I-1)
  145               PERCT(I)=(VALUES(I)*100)/VALUES(1)
  146         28    CONTINUE
  147         CCC
  148         C  DRAW THE CIRCLE USING POLAR COORDINATES!
  149         C
  150               DISPLAY " ␛h␛J"
/
```

```
   L151/200␍
⊦  151            WRITE(6,2)␍
⊦  152        2   FORMAT(1X,"␛*pa")␍
⊦  153            DO 4 I=0,360,5␍
⊦  154            THETA=I/57.295779␍
⊦  155            IX=IXCTR+RAD*COS(THETA)␍
⊦  156            IY=IYCTR+RAD*SIN(THETA)␍
⊦  157            WRITE(6,3)IX,IY␍
⊦  158       3    FORMAT(1H+,I4,1X,I4," ")␍
⊦  159       4    CONTINUE␍
⊦  160      CCC␍
⊦  161      C␍
⊦  162      C  COMPUTE AND DRAW THE PIE CUTS FOR THE VALUES␍
⊦  163      C␍
⊦  164            DO 35 I=1,INUM␍
⊦  165            THETA=ANGLE(I)/57.295779␍
⊦  166            IX=IXCTR+RAD*COS(THETA)␍
⊦  167            IY=IYCTR+RAD*SIN(THETA)␍
⊦  168      C␍
⊦  169      C DRAW THE PIE CUT␍
⊦  170      C␍
⊦  171            WRITE(6,31)IXCTR,IYCTR,IX,IY␍
⊦  172      31    FORMAT(1X,"␛*pa",I4,","I4," ",I4,",",I4,"Z")␍
⊦  173      35    CONTINUE␍
⊦  174      CCC␍
⊦  175      C  DRAW LABELS␍
⊦  176      C␍
⊦  177            WRITE(6,204)GRTXON␍
⊦  178            DO 200 I=2,INUM+1␍
⊦  179            THETA=POSLAB(I)/57.295779␍
⊦  180            IX=IXCTR+RRAD*COS(THETA)␍
⊦  181            IY=IYCTR+RRAD*SIN(THETA)␍
⊦  182      CCC␍
⊦  183      C  POSITION THE PEN␍
⊦  184      C␍
⊦  185            WRITE(6,203)IX,IY␍
⊦  186            IF(POSLAB(I).LE.90.OR.POSLAB(I).GE.270)WRITE(6,201)␍
⊦  187            IF(POSLAB(I).GT.90.AND.POSLAB(I).LT.270)WRITE(6,202)␍
⊦  188      201   FORMAT(1H+,"␛*m1Q")␍
⊦  189      202   FORMAT(1H+,"␛*m7Q")␍
⊦  190            KK=NUMCHARS(I)␍
⊦  191      200   WRITE(6,205)PERCT(I),(IDUMY(J,I),J=1,KK)␍
⊦  192      205   FORMAT(1X,"␛*l",F5.2,"%",1X,10A2)␍
⊦  193      204   FORMAT(1H+,A4)␍
⊦  194      203   FORMAT(1H+,"␛*pa",I3,",",I3,"Z")␍
⊦  195            CALL DATELINE(DATE)␍
⊦  196            IX=360␍
⊦  197            IY=10␍
⊦  198            WRITE(6,203)IX,IY␍
⊦  199            WRITE(6,207)SIZE2␍
⊦  200      207   FORMAT(1X,A5)␍
⊦/␄
```

```
   L201/LAST␍
␊  201                WRITE(6,208)␍
␊  202       208      FORMAT(1H+,"␛*m4Q")␍
␊  203                WRITE(6,206)DATE␍
␊  204                IY=339␍
␊  205                WRITE(6,203)IX,IY␍
␊  206                J=1␍
␊  207                KK=NUMCHARS(1)␍
␊  208                WRITE(6,209)(IDUMY(I,J),I=1,KK)␍
␊  209       209      FORMAT(1X,"␛*l",10A2)␍
␊  210       206      FORMAT(1H+,"␛*l",A27)␍
␊  211                IX=61␍
␊  212                WRITE(6,203)IX,IY␍
␊  213                WRITE(6,301)HP␍
␊  214                WRITE(6,301)GR␍
␊  215       301      FORMAT(1H+,"␛*l",A8,/)␍
␊  216                WRITE(6,204)GRTXOF␍
␊  217                DISPLAY " ␛h␛*dF"␍
␊  218                STOP␍
␊  219                END␍
␊/␑␛Z
```

# EXPERIENCES WITH THE

# MANUFACTURING PACKAGE MFG/3000

By

Ivan M. Rosenberg
General Partner
Systems Design Associates
P.O. Box 1144
San Luis Obispo, CA, 93406

John Doyle
Plant Manager
Vetter Corporation
1150 Laurel Lane
San Luis Obispo, CA, 93401

A paper presented to the 7th International Meeting of the HP General Systems Users Group, October 30 - November 2, 1978, Denver, Colorado.

TABLE OF CONTENTS

Page

Abstract . . . . . . . . . . . . . . . . . . . . . . . . ii

I. Introduction . . . . . . . . . . . . . . . . . . . . . 1

II. System Installation . . . . . . . . . . . . . . . . . 3

III. MFG/3000 From A User Point of View . . . . . . . . . 4

IV. MFG/3000 From A System Administrator Point of View . 10

V. Implementation of An Order Processing System . . . . 24

VI. Potential and Recommended Enhancements . . . . . . . 28

VII. Conclusion . . . . . . . . . . . . . . . . . . . . . 30

Acknowledgments . . . . . . . . . . . . . . . . . . . . . 31

Figures . . . . . . . . . . . . . . . . . . . . . . . . . 32

# EXPERIENCES WITH THE MANUFACTURING PACKAGE MFG/3000

## I. Introduction

MFG/3000 is a collection of three software modules that form an integrated MRP-based manufacturing support package. It is oriented to manufacturers who manufacture and assemble discrete parts, where a primary goal is to minimize inventory investment, yet maintain adequate and timely supplies for production and customer orders. On-line use is encouraged through the use of highly simple menu and data entry screens on CRT terminals, although batch input is available.

MFG/3000 is structured on IMAGE/3000 data bases, and permits the use of QUERY for ad hoc and customized reports, as well as some emergency data base "fixing." The on-line programs utilize DEL/3000, thus requiring HP 264X terminals.

The relation between the three software modules is shown in Figure 1. Engineering Data Control (EDC) maintains descriptive, cost, and planning information about all parts, and bill-of-material workcenter, and routing information about the manufacturing operation. In addition, engineering change information regarding future changes in a bill-of-material and miscellaneous remarks about a part or bill-of-material may be stored. The Inventory and Order Status (IOS) module maintains records of planned and actual inventory issues and receipts. Work and purchase orders are entered directly into the system. Upon request, IOS traces through the relevant bills to determine and allocate all needed parts for a work order (which can refer to only one fabricated part). At this time the user can determine if all needed parts are or will be available. At the proper time, based on specified lead times, pick lists are generated. The input resulting from the actual pick operation creates a reduction in inventory level and a stock activity record. Likewise, receipts create a historical stock activity record. Such records can be accessed on-line and are purged on a periodic basis.

Finally, the Material Requirements Planning (MRP) module is a planning tool to help management balance current and anticipated demand for a part with current and anticipated supply for the same part.

HP recommends that there be at least two management positions associated with the implementation and operation of MFG/3000. The User Trainer is responsible for educating users as to the proper procedures for using the system, including on-line transactions, management policies, and use of printed reports. The System Administrator is responsible for the proper operation of MFG/3000 itself, including data base integrity and security, maintenance, back-up, and modifications as required.

This paper will discuss the experiences of one of the earliest users of MFG/3000, first from a user point of view (Section III), and then from a System Administrator view point (Section IV). Perceptions gained during implementation of an order processing system are shared in Section V. Enhancements, potential and recommended, are discussed in Section VI.

## II. System Installation

Vetter Corporation is a manufacturer of motorcycle accessories and employs approximately 250 people at an Illinois facility and approximately 125 at a west coast California plant. It has experienced a high growth rate over the last few years and has had a continuing problem relative to materials required for the production and support of its products. In order to solve these problems, a program of investigating computer systems for inventory management was begun during the summer of 1977.

In the fall of that year, Systems Design Associates (SDA), a computer consulting firm, was engaged to perform a market survey and analysis and to make recommendations regarding feasible computer systems. Among the criteria for the system were:

1. Reasonably powerful data base management system software
2. On-line query capability into the data base
3. Ability to handle multiple data bases
4. Ability to handle up to 24 on-line CRT terminals simultaneously, with at least three background batch streams
5. Availability of adequate maintenance and vendor support
6. Availability of a reasonably powerful MRP application package
7. Capacity for future expansion

Primarily because of the on-line data base query capability, there were few feasible systems to consider.

The HP 3000 and MFG/3000 were studied during the first months of 1978, followed by a purchase of an HP 3000 Series II Model 6 on March 22. Besides MFG/3000, this system included 320K of main memory, two 50 MB discs, a 600 LPM line printer, a 1600bpi tape drive, the MPE-II operating system, the IMAGE data base system, QUERY, DEL for terminal management, and COBOL. The 3000 arrived on May 2 and was installed on May 10. The industry specialist from the Los Angeles office installed the EDC package on May 12, and the following week on May 19, he installed the IOS package. On June 9, the MRP package was installed. The system has been in continuous operation since that date. During the fall of 1978 an upgrade was made to a Series III with one megabyte of main memory and a 120 MB disc was ordered. In addition, the new operating system MPE-III was installed in September, 1978. A 9600 baud lease line connects the Illinois plant with the California computer facility.

After system selection, SDA was contracted to perform facilities management and MFG/3000 administration during installation and until system operation stabilized and Vetter personnel could be properly trained. Turnover of system management was completed during the latter part of summer, 1978. Currently, SDA is designing and supervising the implementation of a comprehensive Order Processing System—including order entry, accounts receivable, warranty picking, shipping, and inventory allocation—which is discussed in Section V.

## III. MFG/3000 From A User Point of View

The EDC package is somewhat of a misnomer for production description. It could be more accurately described as a definition module, and more specifically, the definitions are all defined in terms of manufacturing or production criteria. It is broken down into three primary divisions:

1. Part definition—Data elements directly tied to a part number. Structure definition, which is the relationship between a number of parts which describe a finished product or sub-component.

2. Planning and control function for purchased parts as well as fabricated goods.

3. Routing and work center section to describe work centers and activities which occur at the work centers to produce finished goods and sub-assemblies.

The IOS package handles inventory, including stock locations, quantities on hand, and other essential elements necessary to track and maintain accurate inventory records. The second portion of this relates to orders—purchase orders and their associated vendor information, and work orders, which are essentially in-house purchase orders and associated pick lists and materials requisitions to drive the work order system.

The MRP package is very complete, allowing multi-purpose policies and scheduling techniques. It is a re-generative type in that it is a batch type program which is run once a week in our plant. It is essentially driven by the IOS and EDC packages; and if those two have been brought up in a complete manner, the MRP package simply strips data from both and produces a series of reports which control and schedule both in-house work orders and out-of-house purchases.

Implementation from the standpoint of the users is reasonably defined by the structure of the system. The EDC package must be brought up first and item data entered so that it may be transferred in a batch job to IOS in order to allow the warehousing people to begin to work on their inventory counts. We have found that at our site, with the item data set complete and outstanding purchase orders entered on the IOS data base, the warehousing functions can begin by receiving the outstanding purchase orders and by commencing cycle counts which are provided by the system. This will allow purchasing people to begin to become familiar with entering purchase orders on the system which have been initiated by their old manual inventory control system.

Essentially, this allows a small step toward bringing them out of their old system into an automated one and reduces the trauma of grossly exposing them to all its facets at the same time. The warehousing people at this point are simply exposed to receiving goods on the computer and cycle counts. At this point, two parallel lines develop—the warehousing group can do physical inventory and load the counts on the computer and verify the inventory by additional cycle counts to guarantee the accuracy of their data, and the person responsible for developing the EDC structure information can now begin to load the structure data required to define the assemblies and sub-assemblies required in the plant operation. This data is necessary prior to the development of any internal work order situations. At this point, the purchasing people have been exposed to the system and are loading their purchase orders. The receiving department is receiving goods acquired from those purchase orders. Incoming inspection of those goods may be implemented at any point in this cycle, depending on the local needs. At the same time, the warehousing people are becoming familiar with the cycle counts and details of the system necessary to maintain their inventories.

The final step in the process is—with the structure information loaded—a final review is required of the EDC information necessary to define inventory control and purchase part definition. This information relates to lot sizes, lead times, and policies, etc. It is extremely important that they be related to the real situations in this plant and that they be applied with good inventory management goals in mind and good purchase policies. This is important because of the fact that the MRP program uses this item data in calculating inventory purchases and demand. The system simply emulates the buying decisions and the manufacturing decisions which have been attached to their component part. In essence, poor purchasing information will be followed by the computer and implemented just as effectively by the computer as it would by an individual.

If the EDC control portion has been implemented properly and with all due regard for economy and control and the IOS portion implemented with accurate inventories and outstanding purchase orders, the final step is to load a master schedule useable by MRP to create the materials requirement plan for the plant. The key in developing a master schedule is, of course, to weigh the needs of the sales forecast with the inventory and cash goals provided by a finance group to provide a realistic schedule within the capabilities of the physical plant which the MRP program can implement to drive purchase order and work order requirements.

One final issue is that the operational departments of the company be structured for control and economy with the master schedule with its inputs from sales forecasting and finance driving the MRP, which provides further specific direction for purchasing and production. It is necessary that the operational departments of the plant be structured so that a scheduling department or production inventory control department be intimately tied to the forecasts, etc. The computer and the MFG/3000 package will take the master schedule demand and provide information which may be directly inputted into purchasing and production scheduling. Conflicts in the schedule and the master plan must be resolved by the scheduling, purchasing, and production departments. In some cases, the master schedule must be revised due to the inability of purchasing and production to accomplish the goals. On the other hand, the system also provides quantitative feedback to anticipate and measure capacity limitations. These limitations, if pointed out in advance, may, at the option of management, be resolved to meet the master schedule if its requirements are paramount to these defined limitations.

Our feeling after using the package for approximately six months is that it provides extremely useful and flexible tools for purchasing, warehousing, and production to accomplish their objectives. It provides qualitative tools for management to evaluate operational departments in terms of inventory value, inventory turns, back order analysis, shortage analysis, and materials requirements. These may be used by the operational departments to evaluate their own members and by management to evaluate their operating departments. The MFG/3000 package is entirely materials-oriented, and its biggest limitation is simply that it encompasses the materials portion of the manufacturing plant's operational system. It provides only slight support for labor and routing information and provides little or no support for the financial department. The package may be enhanced best by further development of product costing, job costing for the materials area, and the development of a master schedule system to ease the development of master schedules and inventory control. Obviously, to encompass all the elements of the manufacturing plant, an accounts payable package tied to the purchase order portion of the IOS package is desirable; and on the other side of the master schedule module, an order entry and accounts receivable package would greatly round out the whole system. Of course, standard accounting functions, such as general ledger, etc., should be provided to tie the whole system together.

In summary, the package has been found to be extremely easy to use and is easily implemented by the operational departments. The design is relatively simple and straight forward so that it solves a large number of the operational problems of the manufacturing plant without embroiling the departments in embellishments which detract from the objective of this system—which is to increase the efficiency and operational effectiveness of the manufacturing plant.

In many MRP-oriented systems, the goal of increased efficiency is obscured by "bells and whistles," which reduce the effectiveness of the computer-oriented system. The MFG/3000 package, in a simple, straightforward approach, has addressed the materials problems of the manufacturing plant in a very successful manner. We strongly recommend that Hewlett-Packard continue with this approach and encompass the above-mentioned additional areas to provide an inclusive package for the manufacturing plant.

## IV. MFG/3000 From The System Administrator Point of View

A. Installation. As discussed in the previous section, installation was a fairly non-traumatic process. Vetter had the advantage of implementing first at the California plant which was in the process of being established and is considerably smaller than the Illinois plant. In addition, since the computer system was on-site, communications problems were not involved. Implementation of each package for the Illinois plant followed California installation by about one month. Both installations pointed out the need for careful planning, particularly during the loading of the EDC data base. For the System Administrator, early establishment of the default values for the various data items associated with parts is very important. Although direct use of QUERY can be used for some simple changes to large numbers of parts, such updates must be done extremely carefully. Later, QUERY was used indirectly for massive updates by using it to create a transaction file containing all needed changes for EDCMAINT. Such a technique can also be used to more easily request reports pertaining to large numbers of parts, instead of requesting the report for each individual part number through EDC. For example, one can request a bill-of-material for all parts meeting certain criteria. It is recommended that forms similar to the EDC screens be designed to simplify and control initial data entry.

As is typical, most errors are discovered during system use, so one can expect to detect most of the errors in EDC during the early months of IOS use.

Another problem was encountered during the installation of the Illinois plant. Although MFG/3000 is installed by HP assuming a MGR.MFG3000 user and account, with two plants two different accounts are needed (each plant has its own MFG/3000 data base). In addition, to save disc space and CST's, we decided that only one copy of MFG/3000 programs would be stored for use by both plants. With all programs stored in one plant's accounts, the other plant needs the work files, its own data base, the forms files, and its own copy of the JCL (or STREAM) files. These files are modified as follows:

1. The PGM= parameter of the EDC3000 and IOS3000 screens (the first screens of the modules) must be changed to a fully qualified program name in the "program" account. This is easily done through FORMAINT.

2. All JOB user and account names must be changed to that of the plant.

3. All RUN statements in JCL and those of on-line users must be changed to fully qualified program names.

4. The access security on the program groups must be changed to ANY for execute. A similar change may have to be made in the account in which the programs reside.

Adequate security between plants is maintained since file references default to the log-on account, and this also causes our recommendation that these courses be attended only by people who were already educated regarding computers and manufacturing, thus reducing the chance they would degenerate into introductory courses.

We found the quality of the courses, instructors, course materials, and documentation to be very high. Each course lasts two to three days, and much material of practical worth is covered. We had an advantage in that the implementation of each module was actually accomplished before the corresponding course was attended, but that just increased its worth to us since problems we had encountered and "tuning" issues could be discussed. There was considerable lab work, although some was simply rote. There is a significant advantage in having a "live" terminal in front of each student to test functions as they are discussed.

The system specialists in MFG were also (and continue to be) invaluable, as problems occur particular to our installation. Telephone calls average one or two a week, and we are visited at least once a month. This frequency is expected to diminish over the next half-year.

C. Customization. Vetter purchased the object code version of MFG. The source code version costs considerably more and is not maintained by HP. Despite this, MFG offers some customization abilities.

Screens may be modified under certain conditions. Alternatives may be added to menu screens, including branching to user-written routines. Fields that exist in the standard MFG data base may be added to data entry (FMT) screens and non-key fields may be deleted from those screens. The order of existing fields may not be changed. Fields added to the data base by the user may not be added to the screens.

Data retrieval (RET) screens, for all practical purposes, may not be modified except to move fields (maintaining the same order) and to change protected fields.

The new user may become somewhat confused by MFG's use of the NEXT= parameter of a DEL screen, which usually indicates the next screen in a sequence. In MFG it indicates the previous screen, and is used to "back up."

Field editing may be changed within the different alternatives offered by MFG (such as alphanumeric, numeric, etc.) as long as the new editing is more restrictive. The size of fields may be diminished.

We have modified the JCL streams often, particularly with the MRP module. Up to 15 levels of planning are available; Vetter uses only seven. Since the analysis routines are performed by chained STREAM commands, one need only change the last STREAMs of the last level desired, e.g., MRP2007J, to chain instead to the last jobs of the MRP process (MRP2100J and MRP2400J). In addition, MRP2400J must be modified to turn the FILE and SORT statements for unused levels into comments. Finally, the FILE statements in the last MERGE are likewise deleted. Thus, to add another level of planning is a relatively simple process of removing comment indicators.

It is difficult, if not impossible, to modify MFG off-line reports. QUERY has proved to be an invaluable tool in this regard. In addition to the technique of producing a transaction file described above, QUERY is extensively used directly for producing a multitude of periodic and ad hoc reports. Purchasers should be cautioned, however, not to permit any except trained technical personnel to use QUERY for updates. It is very easy to get the data base in such a state that the pointers, etc., are in very bad shape. Additionally, updates lock the data base for a long time. In general, avoid using QUERY for anything except retrieval.

D. Security. Probably one of the weakest aspects of MFG is its security provisions. There are essentially four levels of security: the MPE account structure, file lockwords, MFG originator numbers, and data base passwords.

The MPE password security is of limited value since it must be known by large numbers of plant personnel to enable them to use the system. Lockwords on user programs suffer from the same problem, although all our System Administrator routines are so protected.

Originator numbers must be entered by the MFG user before access to most on-line functions. Some data retrieval and report requests do not require originator numbers. These numbers, ranging from 0 to 99, are assigned to individuals and groups and are used for distributing the Transaction Register Report of EDCMAINT and for controlling the functions that may be performed using each screen. An originator may have any of three capability levels:

1. None

2. Modify

3. Add/Delete (which includes Modify)

Although access rights are assigned through MFG by fields, we have found it more practical to regard the capabilities to be attached to screens since to effectively use a screen, one must have the same capability for all fields on that screen. Any particular originator must have Add/Delete capability for all fields associated with him, not just a subset.

Originator numbers do have some limited value in protecting against inadvertent but unauthorized behavior, but they have almost no value in protecting against intentional misuse. Originator numbers are restricted to only 100 alternatives and are printed on so many reports that acquiring one seems a very easy process. We have recommended to HP that the originator number be alphanumeric. Although we were told that we could have 100 originators as long as we had only 20 variations of capabilities, we discovered that only 20 originators are allowed, regardless of capabilities.

The system is installed with a system administrator originator number with complete capability (in fact, there are numerous originators existing on the originally installed system which might be deleted). It may be best not to have any originator with such capability, and only create it when needed. At the very least, the standard administrator number should be changed since it is published in HP documentation.

A major criticism of the security is the data base protection. Any user with the logon password and the data base password can easily access (and update) the data base using QUERY. Unfortunately, there is one master password for write access to <u>all</u> fields in the data base. This password is common to all three MFG data bases, is the same for all MFG installations, is not at all cryptic, and even worse is published in some HP literature! For the purchaser with dial-up capability, a major vulnerability exists with regard to the security of the data base. We have recommended to HP that a unique password be established for each installation.

E. Production Operation. There are many batch jobs that must be STREAMed to maintain the system. Data entry in EDC does not directly update the data base, but rather appends a record to a transaction file which is later used by the batch job, EDCMAINT, to perform the updating. EDCMAINT also produces the off-line reports. IOSO400J strips certain information from the EDC data base and updates part information in the IOS data base. Other batch jobs update the Edit Tables (which define screen field editing and originators), delete parts from the IOS data base, etc. In addition, we have created job streams for some QUERY reports.

Most of these batch streams require exclusive access to the data base for correct operation. EDCMAINT is particularly tricky, since it repeatedly requests and gives up exclusive access. If someone logs into EDC between such accesses, EDCMAINT can abort in the middle. It is not a big problem to restart it in the aborted place, but it must be done with care and is an inconvenience. In general, all batch jobs are run after working hours and/or during lunch. There is a slight modification to the EDCMAINT jobstream that will prevent logons during its execution; but this poses some additional problems if an abort occurs for other reasons. And, since the MFG programs are shared by both data bases, this prevents users of the other data base from running EDC. One then tries to insure that all users of MFG are logged off prior to STREAMing EDCMAINT. A potential solution is to create two users, e.g., USREDC and USRIOS, one for each module. Another simpler alternative is to ask users to append their name and module in the HELLO command, e.g., :HELLO IVANEDC, MGR.MFG3000. A :SHOWJOB then indicates who is logged on. Without this additional information, one only has the QUIET indication on the SHOWJOB as a guide.

It was originally intended that EDCMAINT would be run only once a day. However, during the first few months after installation, changes were so frequent that four runs in a day was not uncommon. With more users on now and a more stable data base, there are two EDCMAINT runs per day, at Noon and after 5:00 p.m., for each plant. There is a time zone difference between the two plants. Thus, because of the extensive nature of EDCMAINT, users at one plant can experience a noticeable degradation in response time when EDCMAINT is being run for the other plant.

To reduce such degradation, one may remove from EDCMAINT the job steps which update data sets for which the System Administrator is certain no transactions exist. For example, Vetter does not yet use the workcenter or routing data sets. Clearly, such a modification must be done with care.

In addition, STREAMing IOS0400J (which updates the IOS part information from EDC) has been appended to the end of the EDCMAINT jobstream so the IOS data base is always current with the EDC data base. All scheduled off-line reports are then run after the completion of EDCMAINT.

It is very important that the System Administrator monitor the EDCMAINT runs since we experienced frequent aborts and erroneous data entries during the first months of operation. The aborts were not due to program bugs, but rather to something wrong with the input data or data base. The JCL comments and abort messages are very extensive and helpful. However, it is important to insure that EDCMAINT runs to completion before permitting users to initiate EDC again. The Transaction Registers should be distributed promptly to the originators after being reviewed by the System Administrator for serious or frequent errors. We tried to insure that all users completely understood how to interpret the Transaction Register Report and that they kept all copies on file.

In particular, during the initial months of use, the System Administrator should monitor the sizes of the data sets of each data base. This may be easily accomplished using the DBSTATS routine. Capacities should be adjusted so the data sets are approximately 70% full (or less).

Periodic DBUNLOAD and DBLOAD of a data base can also contribute to improved response times. In a multiple disc installation, it is preferable to place the root file on one disc and the data sets on the other, thus reducing disc contention.

F. Multi-Plant Operation. The modifications necessary to the JCL, etc., to maintain only one copy of the programs in a multi-plant environment has been discussed in Section IV.A. Three problems remain for the remote plant:

1. To direct printed output to the remote line printer

2. To be able to defer the printing of certain reports for the loading of special forms, for large reports, etc.

3. To be able to view the JCL listings resulting from a STREAMed job to check on the occurrence and reason for an abort, etc.

Initially, we installed a "minispooler" provided informally by HP. This routine solved the first problem, but not the other two. Understandably, frustration was high at the remote site.

In the fall of 1978, the minispooler was replaced with the RSPOOL package of DataCon of Oregon, resulting in the solution of all three problems indicated above.

The JCL for the remote site's MFG system must be modified in the following manner:

1. Add the OUTCLASS = LP,1,1 clause to all JOB statements. This specifies 1 copy and an OUTPRI of 1.

2. After the FILE statement for the report file, a set of about 10 lines must be added to initiate and control the execution of the routine SPOOLCOM. This connects the report file to the remote printer, specifies the number of copies, and permits the report file to be held after printing.

3. After each execution of SPOOLCOM, there must be inserted a set of lines which initiate and control the execution of RPOOL. This sets such parameters as lines/form, number of lines between forms, etc.

If the OUTFENCE of the system is set to 1 or greater, the JCL of the remote plant will not print on the system printer, but will be held in the spool files. The remote plant then uses SPOOK to look at the JCL files for a solution to the third problem indicated above. Since SPOOK permits access only to the JCL of the logon account, such users do not have access to the JCL of other accounts. SPOOK permits the user to list the job numbers of the JCL currently in the spool file, which then may be used to display the JCL of the job. The users must periodically purge the JCL files in the spooler so that it does not fill.

If the parameters of the RSPOOL and SPOOLCOM executions are set appropriately, the printing of a report at the remote site may be deferred. SPOOLCOM may then be used on-line to alter file specifications in order to initiate printing when ready or to delete a report file.

We have experienced great satisfaction with this arrangement. Essentially, it grants to the remote site all the power (and then some) of the on-site installation.

G. Backup. One of the most important jobs of the System Administrator is scheduling and supervising backup procedures.

Vetter does a partial backup each night, with a full system backup once a week. Currently, the MFG data bases are not separately backed up using the DBSTORE program, although this will probably be implemented shortly since restoring from the SYSDUMP tape is less reliable and slower.

The transaction file of EDC provides some roll-forward capability, since the last five transaction files are automatically saved. However, this requires that a copy of the data base corresponding to its state prior to the EDCMAINT run be available. If EDCMAINT is run twice a day and backup is performed only once, roll forward can be done from only two or three backup copies, not five. However, it is possible to modify the EDCMAINT JCL so that more than five transaction files are saved.

IOS automatically provides a journaling of all transactions which affect the data base. The documentation is very unclear as to how to control this capability. In summary, the logging in IOS is <u>always</u> running. The only control the user has is to which device the logging is directed, disc or tape. Normally, we log to disc. Twice a day, STARTLOG is run to dump the current contents of the disc log file to tape, which also directs any future log transactions to the tape. After the disc file has been dumped, STOPLOG is run to direct future logging to the disc. We insure that no one is logged into IOS during the time STOPLOG is run. Besides insuring that the log file is large enough to contain the number of transactions likely to be entered between tape dumps, that is all there is to it. There is a utility to display the current number of log entries in the file so that it can be monitored. Our volume is easily accommodated on a 1200' reel.

At the current time, IMAGE does not provide any journaling capability; thus roll-forward and roll-backward are not available outside of the MFG facilities.

Another backup capability may be provided by the HP 2645A terminals themselves if they have the tape option installed. If the computer system is unavailable, or data entry takes place over a slow speed data line, the data may be loaded off-line through the terminal onto a tape cassette, then dumped to MFG in a rapid fashion. In order to prepare for this process, store the MFG screens on a tape cassette, (by displaying a screen, then in local mode copying to tape). Later, in LOCAL mode, display the selected MFG data entry screen by copying from tape to the screen. Put the terminal in format mode using CTRL $f_4$. Enter the data normally. Pressing the ENTER key will copy the unprotected fields to the tape and clear the screen. Data entry may then continue in a similar fashion. Conclude by releasing format mode using CTRL $f_5$.

In order to enter the stored data, log on to MFG and display the appropriate data entry screen. Copying a record from tape to the screen and then pressing ENTER will cause MFG to read the screen in the normal manner.

## V. Implementation of an Order Processing System

SDA and Vetter are currently implementing a comprehensive order processing system (OPS) to interface with MFG/3000. The overall structure of the system is shown in Figure 2. It consists of the following elements:

1. Part Element—To maintain descriptive and availability information about each part that may be sold.

2. Dealer Element—To maintain information about each customer.

3. Order Entry Element—For the entry, review, modification, and reporting of information about orders.

4. Pick List Element—To produce pick lists on a selected basis.

5. Shipping Element—To produce invoices and other shipping documents.

6. Accounts Receivable Element—To maintain accounts receivable information relative to orders and dealers.

7. Warranty Element—To maintain information about the product and the end purchasers.

The Part Element operates much like IOS in that it strips relevant information from the EDC and IOS data bases and puts it into the OPS data base. This continues the MFG policy of maintaining data base separation between major functions. Because IMAGE permits locking only at the data base level (but see Section VI), separate data bases permit an acceptable response time in a multiple user, on-line updating environment at a cost of duplication of data and increased storage requirements.

In addition to the obvious on-line functions, there will be numerous reports (well over 20) provided.

The major contribution of this system will be timely indications of the ship date of an order through consideration of planned receipts of finished goods and shipping rate limitations. We expect to be able to predict ship dates within a day's accuracy at the time of order acceptance. Information concerning planned receipts will be acquired directly from the master production schedule driving MRP or from the work orders and purchase orders of IOS. Thus, any change in production schedules will be immediately reflected in changed order ship dates.

During this implementation, we realized that although MFG and IMAGE will permit the user to add data fields to the end of data sets, this was a dangerous practice. Future versions of MFG could very well expand the number and size of data fields. The addition of custom data fields could prevent the easy updating of MFG, requiring special programs to unload and load the data base (since the new HP fields would have to be "inserted" between the standard and the custom fields), and the custom programs would also have to be changed. This was another major reason that we decided to implement a separate OPS data base.

There are four interfaces between the two systems:

1. Moving part and inventory information from MFG to OPS
2. Acquiring the master production schedule (MPS)
3. Acquiring the shop calendar
4. Updating the inventory counts as a result of shipments

Acquiring part information is a very easy task. The Part Element has produced the side benefit of insuring that the EDC data bases of the two plants contain the same information about the same part number.

We have not yet finalized the method of acquiring the MPS. This is relatively difficult to acquire directly from IOS. We considered having each plant manager create an EDITOR file, from which we would generate appropriate transactions to IOS for MRP to OPS. However, this seems too complicated. Our judgment now is to simply scan the IOS data base for work orders for finished goods and create corresponding allocation data records in the OPS data base. Since the factory works directly from these work orders, we would be using the best information available. This would free the plant manager to use the input of a MPS to IOS as a planning and "what if" tool. Finally, it encourages the plant managers to plan their production schedule out to the horizon of shipment planning.

MRP generates a ship calendar which we plan to use directly for assigning order ship dates. In this manner, we can easily take into account holidays and weekends.

Finally, since IOS provides no method (other than stock adjustments) for drawing down finished goods inventory, the OPS system must create a transaction file of stock adjustment requests that a MFG batch job uses to update the inventory levels in IOS. In addition, OPS generates an Issued Goods Report for cross-checking IOS stock activity records.

All of these interfaces have been relatively simple and easy to implement. It appears that MFG lends itself very well to supporting user-written custom programs. The required MFG documentation is very adequate and clear.

We have experienced two problems with the use of DEL in this implementation. Since DEL does not permit field-level reads and writes, the entire contents of the unprotected fields must be transmitted, even if only a subset of the fields is desired. In a remote terminal environment, this tends to radically increase communications line load, and if the terminal is slow, response time can suffer. Our only current solution is careful design of screens.

The other limitation appears to be in FORMAINT. The OPS employs a few screens where there are a considerable number of enhanced, unprotected fields. We have discovered that FORMAINT (and DEL) will accept a maximum of 1920 characters, including the ESCAPE sequences for the fields. Since there may be as many as 10 characters associated with a field just for the enhancements, this limit can rapidly be reached. Although the 2645A terminal may support a form, FORMAINT may not. Our only current solution is to give in and reduce the number of fields.

## VI. Potential and Recommended Enhancements

Enhancements from the user point of view have been discussed in Section III. This section will focus on improvements from a System Administrator view.

With the announcement of the new IMAGE with record-level locking, we can expect the merging of the three MFG data bases into one. This should produce significant improvements in storage requirements and some improvement in user response time (since interactive programs will no longer be locking the entire data base). Some operational problems (such as exclusive access aborts) may be reduced. The batch programs that move information from one data base to another will be eliminated. EDC may be converted to an on-line updating system.

Because of the problems associated with MFG updating, we plan to maintain a separate OPS data base, but record-level locking will produce very significant operating improvements in order processing. The impact of VIEW/3000 as a substitute for DEL is unknown at this time due to the lack of documentation. However, the ability to do field level read/write would reduce communication loads considerably. We would also like the size limit on forms to be at least increased, if not eliminated. A significant improvement to DEL would permit the selective modification of a form, without re-entering all the edit specifications. In addition, it would be nice to simply state that no edit specs existed for the entire form, thus skipping laborious entering of X's.

Although MFG provides some journaling, we would like to see the logging from IOS and EDC better coordinated and similar. Roll-back abilities would be nice. Eventually, it would be best to have such recovery facilities built into IMAGE so user-written programs could take advantage of them.

The OPS would benefit by a better method of entering the master production schedule than as a series of work orders.

However, the two major limitations of MFG/3000 are its security provisions and the high number of CSTs it demands. The security provisions were discussed in Section IV. Despite the large size of the Vetter system, it frequently runs out of CSTs, particularly during a COBOL compile. This is because both COBOL and MFG make high demands on this limited resource. In a multi-plant environment, we have doubts that we could oerate if a copy of the MFG programs were required for each plant. This limitation has forced us to violate the generally accepted HP3000 practice of small routines in the implementation of the OPS, with its attendant problems of long compiles and decreased system performance. We expect that MFG's high use of CSTs will be alleviated in future updates.

## VII. Conclusion

Both as user and system administrator, we have found MFG/3000 to be a well-designed, reliable, well-documented, and "friendly" system, with a few relatively minor limitations. Installation proceeded remarkably smoothly and rapidly. Users became quickly familiar with the system. The addition of custom systems that interface with MFG is straightforward. And, lest we forget, it also has resulted in improved plant operation and customer service.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the assistance of many HP and Vetter personnel who have contributed their creativity and knowledge to the running of MFG/3000 and the ideas in this paper. In particular, we thank Margie Coolidge, DP Manager, and Mark Laslo, Operator, both of Vetter Corporation in California, and John Scribner and Dave Hesser of Hewlett-Packard.

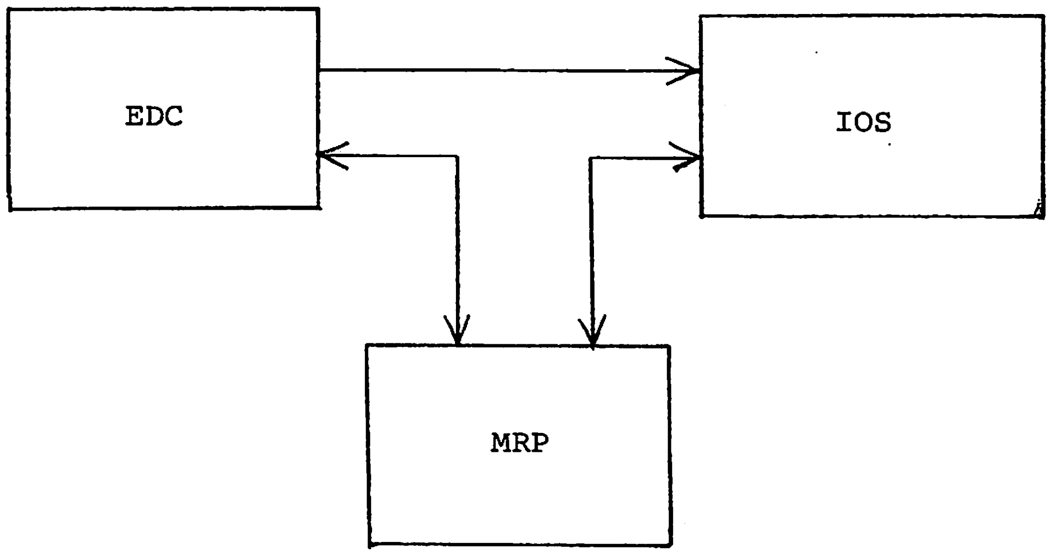


FIGURE 1. MFG/3000

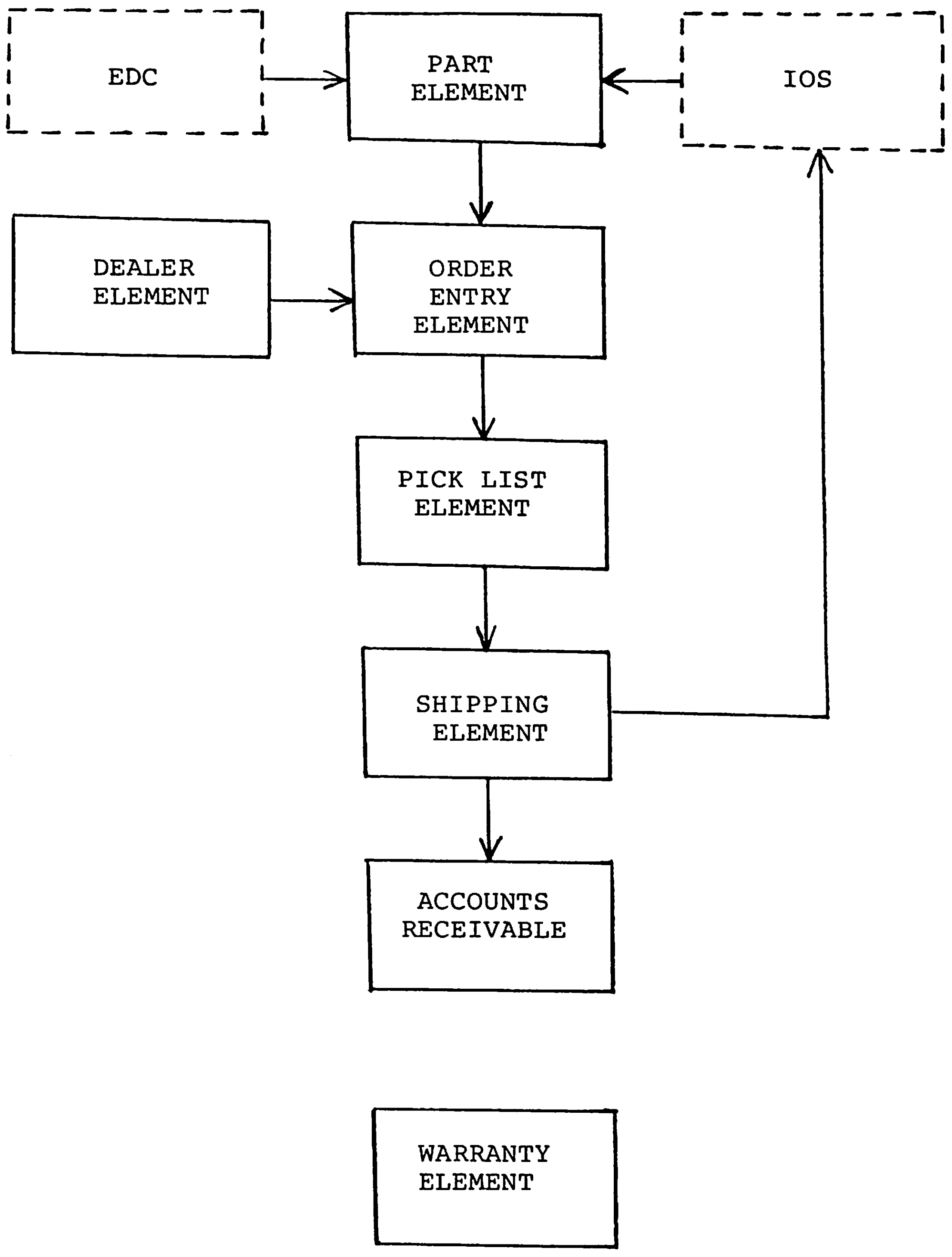


FIGURE 2. THE ORDER PROCESSING SYSTEM (OPS)

# SPSS/HP
# Statistical Package For The Social Science Hewlett-Packard Version

An Update

Marlys A. Nelson
Western Wisconsin Academic Computing Consortium
University of Wisconsin-River Falls
River Falls, Wisconsin 54022

Nicholas Elliott
Political Science Department
University of Wisconsin-River Falls
River Falls, Wisconsin 54022

SPSS/HP, an acronym meaning "Statistical Package for the Social Sciences"/Hewlett-Packard version, encompasses a variety of routines whose function it is to perform statistical analysis of large amounts of data. Also included are easy data entry, selection and modification facilities.

Development of SPSS/HP began in 1974 at DePaul University in Chicago, Illinois for the HP 2000 C'/F time-sharing system. This development was begun due to a growing need by their users for a meaningful statistical analysis system capable of being run on a small time-sharing system. The result was SPSS/HP, an authorized version of SPSS, the well-known package of statistical programs commonly used for research and instruction in the social sciences.

Two years later, the University of Wisconsin-River Falls became interested in the package and converted SPSS/HP to the HP 3000 computer system, expanding upon the HP2000 version to make efficient use of HP3000 system resources. Both sites are continuing to improve and enhance the package.

In designing such a system, the aim was to develop a package which would be familiar to experienced SPSS users but would also take full advantage of the time-sharing features and resources of the HP systems. A user who has gained experience with SPSS at another institution and/or involving another computer system can, with little effort, use the SPSS/HP package.

SPSS/HP is a fully interactive system. All a potential user is required to know is the proper log on. More sophisticated SPSS/HP users may take advantage of HP3000 features, such as the STREAM facility to operate SPSS/HP in a batch mode.

There are four (4) major areas in statistical analysis with which SPSS/HP is concerned. These areas are:

1. Data entry,
2. Data modification,
3. Data selection, and
4. Statistical analysis.

One of the more time consuming, if not tedious, tasks involved in statistical analysis is that of data entry. In an attempt to make this task less burdensome, several methods have been incorporated into SPSS/HP to accomodate several possible data forms and methods of data entry. In the most straightforward, simplest method, data may be entered directly into a SPSS/HP disc file from a terminal keyboard. Such an example is shown in Appendix A. Alternate data entry methods include reading from magnetic tape, punched cards, optical mark cards or existing disc files. Data may also be entered in several forms, such as the traditional 80 'column' card format, BASIC data files (BASD type) or the traditional keyboard entry where each data element is separated by a comma. All these data entry options were designed to give the researcher flexibility in setting up his analysis.

Once all the necessary data has been entered, the user may begin to work with the data. Several facilities are included in SPSS/HP to allow data editing to correct erroneously entered data values, the modification of existing data and the creation of new data through standard mathematical operations. Existing data may be modified by performing an arithmetic operation, such as adding 10 to all data values for one variable, or by editing values individually, such as adding two existing variables, or by using one of the several SPSS/HP functions. The functions include routines which will create various statistical distributions, such as NORMAL, CHI, T or F distributions. Also, new data items may be created through the use of a mathematical-type formula of the form 'NEWVAR=SQRT(4*VAR1)'.

Facilities have also been included to allow the user to select for analysis at a given time only a subset of the entire data file. This selection may be done by selecting only cases where one of the variables involved takes on a specified value. For example, suppose a variable called SEX is coded to indicate 1=MALE respondent and 2=FEMALE respondent. In one specific instance, let us assume we wish to analyze only the MALE responses. This could be accomplished by selecting for analysis only the cases where the variable SEX has the value 1. Subgroups of the total data file may be devised in such ways.

The last step is then the statistical analysis itself. SPSS/HP currently includes 15 different statistical routines. See Appendix A for a list and brief description of the function of each. Also included in Appendix A is a sample run of the newest statistic, ANOVAU, which performs an analysis of variance for designs which have an unequal number of subjects per cell.

In developing, converting and expanding a package of this nature, problems do occur which may -- depending upon the method undertaken to correct the problem -- affect the overall design of the package. Three such problems encountered during the work on SPSS/HP for the HP3000 will be described below. Hopefully, this will serve a dual purpose. First, to illustrate some of the considerations dealt with specifically when SPSS/HP was developed, and secondly, these same problems and solutions may apply to any type of work in the area of statistics or the conversion of large program packages.

One major concern an interactive program package must face is that of execution speed. A user does not want to sit in front of a terminal keyboard for endless periods of time waiting for calculations to be performed. Ideally, the user hopes to press a few keys and have the answer flash on his screen. He can then examine the results and decide upon his next course of action. Two programming concerns contribute towards execution speed -- the amount of time spent reading and manipulating the data file to obtain the necessary information and the size of each program segment.

The first factor, the amount of time necessary to read and manipulate the data file, is obviously best if minimized. This idea has been followed in SPSS/HP where, with minor exceptions, the data file is never read more than once for any given statistic. In a few instances, this concept was not followed in the attempt to allow the program to determine from the data several factors which otherwise the user would be required to supply. One such example is the routine which performs an analysis of variance for balanced designs, ANOVA. SPSS/HP requires two passes over the data file in this instance. The first pass identifies the specifications of the design, such as the number of levels within each factor in the design and the number of subjects per cell in the design, while the second pass reads the data to perform the actual statistical calculations. If only one pass had been made over the data, the ANOVA user would have been required to supply the routine with the specifications of the design. By compromising in this instance on the number of times the data file is read, the use of ANOVA was simplified.

The second factor playing a role in determining program execution time, the size of each program segment, proved to be a double-edged sword. Again, ideally, the smaller a program segment, the faster execution time obtained. The computer is able to spend less time swapping program segments in and out of main memory since smaller segments force less displacement of other program segments already memory resident. Striving for a small program segment size creates another problem. The actual number of program segments is then increased, one large segment split into two smaller segments for example. The actual number of program segments in any one program has a finite upper limit of 63, however. Faced with such a problem, it appears that the only solution lies in a larger program segment size with the hope that execution time does not degrade to a substantial degree. Upon closer examination, however, another solution becomes apparent. While it is true each program may contain at most 63 program segments, there seems to be an unlimited number of programs which may be linked together thru the concept of Process Handling. In process handling, program A, termed the father process, begins execution. When condition 1 occurs, it is directed to activate program B1 for execution. Program B1 becomes the son process and runs to completion at which time program A, the father, again begins execution at the point following the request to activate and execute the son. This linkage of programs can be extended further whereby the father process may activate different son processes and each son may become a father and activate another son. By using this technique, response time did not differ drastically and the various segments within the SPSS/HP package can remain small -- yet the total package remains highly expandable.

A major concern faced when dealing with any mathematical formula to be calculated by computer is the problem of precision. Currently, standard precision of six (6) digits on internal computations is employed. Output values may often be less accurate, however, such as four (4) digit accuracy. Extended precision, type LONG in BASIC, would provide accuracy up to fourteen (14) digits but was not implemented for several reasons. The initial data is entered in standard precision. The entry of data accurate to fourteen (14) digits is rather ridiculous. The need for precision arises in the calculations required for a given statistic.

In this case, a statistical routine would need to read the data in standard precision and convert it to double precision for use in calculations. In an attempt to do this on a limited scale in routines such as REGRESSION, it was found that internal BASIC system routines controlling the conversion and formatting of numeric output could not handle such tasks precisely. Beyond the 4th or 5th digit, accuracy was lost which meant the conversion to LONG served no purpose. An alternative method would be the storage of all data in LONG form initially, eliminating the need for conversion; but this would double the storage necessary for all data files, an unhappy thought for systems constrained by on-line disc storage. At present, it was felt that all calculations were precise enough to delay any efforts at further solutions.

The final problem encountered evolved slowly. One basic design goal set at DePaul University was to avoid the use of scratch disc files. Some programming technique, similar to using scratch files, was necessary, however, to handle situations in which data was deleted or added from an existing data file. The use of the original data file twice within a routine was developed as a solution to this problem. The original data file was assigned as two separate files within a given routine, for example, as files #1 and #2 in a BASIC program. The old information would be read from file #1, the addition or deletion or modification of the data would be made; and then the information would be printed on file #2. This was a nice technique for the HP2000, which can handle the file buffering to avoid reading on file #1 the information just printed on file #2. It was discovered, however, that the HP3000 was not designed to handle such file manipulation. Overlapping of the file information began to occur after a certain number of file reads and prints. Faced with the failure of this technique, the only feasible alternative was the use of scratch disc files, a task easier on the HP3000 than on the HP2000. Temporary disc files may be built and used without any user being aware of the existence of such files. The only restriction imposed by the use of scratch files is that sufficient disc space must exist so that a temporary file identical in size to the original data file may be created---a restriction which has not shown itself to be a hardship as of yet.

Several enhancements of SPSS/HP have been developed in the past few months. The first such enhancement increased the maximum number of variables which may be contained in an SPSS/HP data file. Previous versions have had an upper limit of 108 variables, but this limit has now been raised to allow 500 variables. As before, the number of cases is limited only by the amount of available disc space.

A second recent enhancement is the capability to enter variable labels. This feature allows each variable declared on a SPSS/HP data file to be given a 1 to 40 character description intended to explain the purpose of the variable. These variable labels may be added, edited, or deleted at any time. Variable labels will be printed by each statistical routine. In the future, it is planned to include value labels, which would allow up to 10 values per variable to be given a 1 to 20 character description. This description would also be printed by each statistical routine.

The last major enhancement to SPSS/HP has been the addition of a new statistical routine. This routine, ANOVAU, performs a fixed effects factorial analysis of variance for a design in which there exists an unequal number of subjects per cell. ANOVAU allows for either a two-way or three-way analysis of variance and no factor within the design may contain more than 40 levels. The routine provides the experimentor with four (4) models with which he may test his hypothesis. These models are: 1) The complete linear model; 2) the method of fitting constants; 3) hierarchical analysis; and 4) the unadjusted main effects analysis. A complete table of cell means and variances is also printed along with the analysis of variance summary table. ANOVAU will also handle a design which may contain up to five (5) covariates.

## APPENDIX A

Included in the appendix is a sample run showing data entry from a terminal keyboard into a SPSS/HP data file.

Also included is a table listing all the statistical routines currently available through SPSS/HP and a brief description of the function of each of these routines.

The output produced by the newest statistical routine, ANOVAU, is enclosed. This serves to show the form of the statistical output and to illustrate the routine itself.

```
:RUN SPSSHP



SPSSHP : VERSION 4.04          5/31/78
SPSSHP NEWS LAST CHANGED ON    5/31/78
WED   OCT  4, 1978      8:16 AM


TYPE NEWS FOR NEW FEATURES AND KNOWN PROBLEMS IN SPSSHP.
TYPE INSTRUCT FOR BASIC DIRECTIONS FOR USING SPSSHP.


NEXT ? CREATE FILE!DATA00,3,24,LABELED

FILE SUCCESSFULLY CREATED - USE 'FILE NAME!DATA00' TO WORK
WITH THIS NEW FILE.


NEXT ? FILE NAME!DATA00

NEXT ? VARIABLE LIST!READ,YEAR,ACTIV

NEXT ? N OF CASES!24

NEXT ? READ INPUT DATA

ENTER DATA ONE CASE AT A TIME.
TYPE 'PROMPT' OR 'NOPROMPT' TO TURN PROMPTS ON OR OFF RESPECTIVELY.
'STOP' WILL INTERRUPT DATA ENTRY.

CASE    1, READ TO ACTIV
?1,1,8
CASE    2, READ TO ACTIV
?1,2,5
CASE    3, READ TO ACTIV
?1,2,8
CASE    4, READ TO ACTIV
?1,3,2
CASE    5, READ TO ACTIV
?1,3,7
CASE    6, READ TO ACTIV
?1,3,7
CASE    7, READ TO ACTIV
?1,3,9
CASE    8, READ TO ACTIV
?1,4,5
CASE    9, READ TO ACTIV
?1,4,7
CASE   10, READ TO ACTIV
?1,4,7
    •
    •
    •
CASE   24, READ TO ACTIV
?2,4,2

DATA ENTRY COMPLETED.
```

| STATISTIC NAME | FUNCTION |
|---|---|
| CONDESCRIPTIVE | Computes ten descriptive statistics for interval level data, including the mean, variance and standard deviation. |
| FREQUENCIES | Computes frequency distributions for categorized data. Histograms may be requested. |
| BREAKDOWN | Computes descriptive statistics for sub-groups of cases. A one-way analysis of variance can also be performed. |
| CROSSTABS | Computes contingency tables from categorized data. Four non-parametric statistics may be requested. |
| T-TEST | Computes the t statistic employing one of three user defined parameters of difference of means. |
| PEARSON CORR | Computes Pearson product-moment correlations. |
| SCATTERGRAM | Performs bivariate regression and prints scatter diagrams. |
| REGRESSION | Performs multiple regression. A backward stepwise procedure is available. |
| RESIDUAL | Performs residual analysis. Durban-Watson statistics and a plot of auto-correlation functions of residuals and residuals against time may be plotted. |
| ANOVA | Performs a factorial analysis of variance for balanced designs. |
| ANOVAU | Performs a fixed effects factorial analysis of variance for unbalanced designs. |
| DISTRIBUTIONS | Computes frequency distribution for continuous data. Histogram option is available. |
| SPEARMAN CORR | Computes Spearman's rank-order correlations. |
| T-SQUARE | Computes Hotelling's T(2) statistic |
| U-TEST | Performs Mann-Whitnet tests. |

```
NEXT ? ANOVA;ACTIV BY READ,YEAR

NEXT ? OPTIONS;3

NEXT ? EXECUTE

STATISTICAL PACKAGE FOR THE SOCIAL SCIENCES - HP VERSION 4.04

FILE : DATA00

(CREATION DATE = WED, OCT  4, 1978,  8:18 AM )

--------------------------------------------------------------------------

                A N A L Y S I S   O F   V A R I A N C E

                      ACTIV      STUDENT ACTIVISM
                  BY  READ       STUDENT READING LEVEL
                      YEAR       YEAR IN SCHOOL

--------------------------------------------------------------------------

                         MEAN AND VARIANCE TABLE

VARIABLES        READ        YEAR          MEAN      VARIANCE

  FOR FACTOR : READ

    LEVEL          1.00         .00         6.500       4.056
    LEVEL          2.00         .00         3.357       2.863

  FOR FACTOR : YEAR

    LEVEL           .00        1.00         4.333       6.667
    LEVEL           .00        2.00         4.600       7.300
    LEVEL           .00        3.00         4.875       6.696
    LEVEL           .00        4.00         4.800       5.200

  FOR FACTOR : READ      X YEAR

    CELL           1.00        1.00         8.000        .000
    CELL           1.00        2.00         6.500       4.500
    CELL           1.00        3.00         6.250       8.917
    CELL           1.00        4.00         6.333       1.333
    CELL           2.00        1.00         3.600       4.300
    CELL           2.00        2.00         3.333       6.333
    CELL           2.00        3.00         3.500       1.667
    CELL           2.00        4.00         2.500        .500

GRAND MEAN   =      4.667
```

```
ANALYSIS OF VARIANCE : METHOD OF FITTING CONSTANTS

                         SUMMARY TABLE

----------------------------------------------------------------------
                                SUM OF   DEGREES  MEAN           PROB.
SOURCE OF VARIANCE              SQUARES  FREEDOM  SQUARES   F    OF F
----------------------------------------------------------------------
MAIN EFFECTS                     60.091     4
  READ                           58.966     1      58.966 13.62 .0000
  YEAR                            2.472     3        .824   .19 .9854

TWO-WAY INTERACTIONS              1.959     3
  READ      X YEAR                1.959     3        .653   .15 .9908

ERROR                            69.283    16       4.330
-------------------------------------------------
TOTAL                           131.333    23


NEXT ? FINISH


RUN ENDED
```

## DATA MANAGEMENT

Series "B"

PRESENTATION TITLE: Adopting a Transaction Processor - getting sophisticated becomes Easier

INDIVIDUAL(S) NAME(S): Robin C. Wheeler

ADDRESS: Domtar Construction Materials
2001 University Ave.
Montreal, Quebec

ABSTRACT:

Outline

A. Selection of Transaction Processor

1. Hardware Efficiency Objectives
2. Organizational Objectives
3. Standards
4. Transaction Processor Selected - Overview (Terminal Application Processing System)

B. Application Considerations

1. Objective of the Application
2. Cost/Benefit
3. Design Overview

C. Considerations of a Distributed Network

1. Linking three HP300's (Montreal, Toronto, Calgary)
2. Communications "Futures", X.25

D. Distributed Data Bases

# IMAGE DATA BASE DESIGN AND PERFORMANCE MEASUREMENT

Presented To:

HP GENERAL SYSTEMS USERS GROUP
7th International Meeting

October 30th - November 3rd 1978

Denver Hilton
Denver, Colorado

BY

Orland J. Larson
*IMAGE/3000 Product Manager*

General Systems Division
HEWLETT-PACKARD COMPANY

## OUTLINE

I. INTRODUCTION
   - A. OVERVIEW
   - B. CONCERNS OF THE DATA BASE DESIGNER
   - C. LOGICAL DATA BASE DESIGN
   - D. PHYSICAL DATA BASE DESIGN

II. DATA BASE DESIGN METHODOLOGY
   - A. OVERVIEW OF DESIGN AIDS
   - B. DATA BASE DESIGN AIDS
     1. DATA BASE ENVIRONMENT
     2. USER FUNCTION FLOW DIAGRAM
     3. USER FUNCTIONS AND REQUIREMENTS
     4. ITEM/FUNCTION MATRIX
     5. DATA BASE MODEL
     6. DATA BASE DIRECTORY
     7. DETAILED TRANSACTION ANALYSIS

III. IDEA – A PERFORMANCE MEASUREMENT TOOL
   - A. OVERVIEW OF IDEA
   - B. REQUIREMENTS
   - C. OPERATION
   - D. SAMPLE SCRIPT CREATION
   - E. SAMPLE OUTPUT OF IDEA RUN
   - F. GRAPHS GENERATED FROM IDEA RESULTS
   - G. DISADVANTAGES OF PERFORMANCE MEASUREMENT
   - H. ADVANTAGES OF PERFORMANCE MEASUREMENT

IV. SUMMARY

# DATA BASE DESIGN

DATA BASE ENVIRONMENT
MAJOR APPLICATIONS
DEFINE OBJECTIVES
THOROUGH ANALYSIS of APPLICATIONS
INTERVIEW USERS
UNDERSTAND IMAGE CAPABILITIES
FUNCTIONAL ANALYSIS
DATA GATHERING
OUTPUT REQUIREMENTS
SORTED CHAINS
INPUT REQUIREMENTS
AJAX EMPLOYMENT AGENCY
DATA ANALYSIS
BATCH VS ON-LINE UPDATING
PROCESSING FREQUENCY
PERFORMANCE REQUIREMENTS
ON-LINE REQUIREMENTS
APPLICATION HOST LANGUAGE
ITEM/FUNCTION MATRIX
IDENTIFY KEYS
PRIMARY PATHS
MASTER VS DETAILS
MANUAL MASTERS vs AUTOMATIC
TRANSACTION LOGGING
IDENTIFY NEEDS VS WANTS
IDENTIFY ACCESS METHODS
UTILIZE IDEA
DATA BASE FLOW-CHART
DATA BASE DIRECTORY
SECURITY REQUIREMENTS
MODES
RECOVERY
PASSWORDS
QUIREMENTS
QUERY
"WHERE IN THE @WXX**☆ DO I START?"
?
J KOSIS

HEWLETT PACKARD

## CONCERNS OF THE DATA BASE DESIGNER

- **MUST FULFILL TWO MAIN OBJECTIVES**
  - PROVIDE A STABLE DESIGN TO SATISFY NEEDS OF CURRENT AND FUTURE USERS
  - PROVIDE COST/PERFORMANCE TRADEOFF INFORMATION TO MANAGEMENT
- **CHANGEABILITY OF USER REQUIREMENTS**
  - OUTPUT FORMATS (RAPIDLY)
  - POLICIES (MODERATELY)
  - MANUAL PROCEDURES AND INPUT REQUIREMENTS (SLOWLY)
- **COMPLEXITY OF INTERACTIONS**
  - DATA PEOPLE MACHINES PROCEDURES
- **DATA INTEGRITY**
  - MAKING "ALL DATA" AVAILABLE TO "ALL APPROPRIATE USERS"
- **FLEXIBILITY**
  - CHANGING STRUCTURE WITH MINIMUM IMPACT ON EXISTING USERS
  - DATA INDEPENDENCE
- **LACK OF A DATA BASE DESIGN METHODOLOGY**
- **LACK OF PERFORMANCE MEASUREMENT TOOLS**
  - THROUGHPUT ESTIMATES
  - RESPONSE TIME ESTIMATES

## LOGICAL DATA BASE DESIGN

➧ **INVOLVES THE BUSINESS ENVIRONMENT**

- CURRENT AND FUTURE INFORMATION REQUIREMENTS
- OVERALL OPERATIONAL PLANS AND POLICIES
- ANALYSIS OF FUNCTIONS AND INTERACTIONS (INTERVIEWS)
  - TOP MANAGEMENT VIEW
  - FUNCTIONAL MANAGEMENT VIEW
  - OPERATIONAL PERSONNEL VIEW

➧ **IDENTIFICATION OF DATA ELEMENTS**

- DATA ELEMENTS USED BY EACH FUNCTION
  - DEFINITION
  - SIZE AND DATA TYPE
  - VALUES
  - SECURITY SPECIFICATIONS
- DATA ELEMENTS SHARED BETWEEN FUNCTIONS

➧ **IDENTIFICATION OF LOGICAL DESIGN COMPONENTS**

  - KEYS
  - ATTRIBUTES
  - RELATIONSHIPS

HEWLETT hp PACKARD

## PHYSICAL DATA BASE DESIGN

- **INVOLVES THE TECHNICAL ENVIRONMENT**

- **PROVIDES STRUCTURES AND ACCESS METHODS NECCESSARY TO IMPLEMENT THE LOGICAL DATA BASE DESIGN**

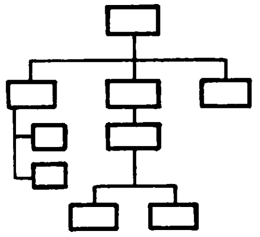
- **CONCERNED WITH:**
  - CAPABILITIES AND LIMITATIONS OF A SPECIFIC DBMS
  - COMPUTER HARDWARE UTILIZED
  - USER REQUIREMENTS OF EACH APPLICATION
  - PERFORMANCE REQUIREMENTS OF EACH APPLICATION
  - PERFORMANCE MEASUREMENT
  - BACKUP AND RECOVERY PROCEDURES
  - DATA BASE IMPLEMENTATION
  - DATA BASE MONITORING
  - SECURITY

- **DATA BASE DESIGN DECISIONS**

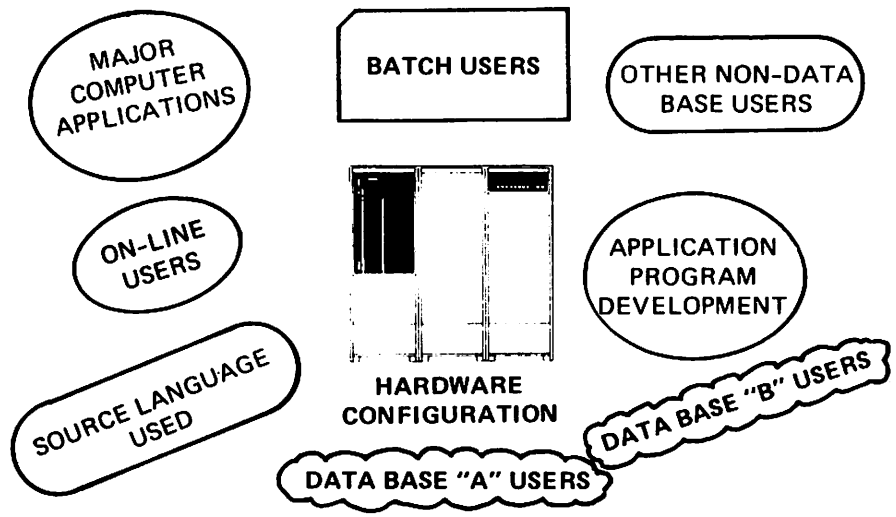
HEWLETT PACKARD

# DATA BASE DESIGN AIDS OVERVIEW

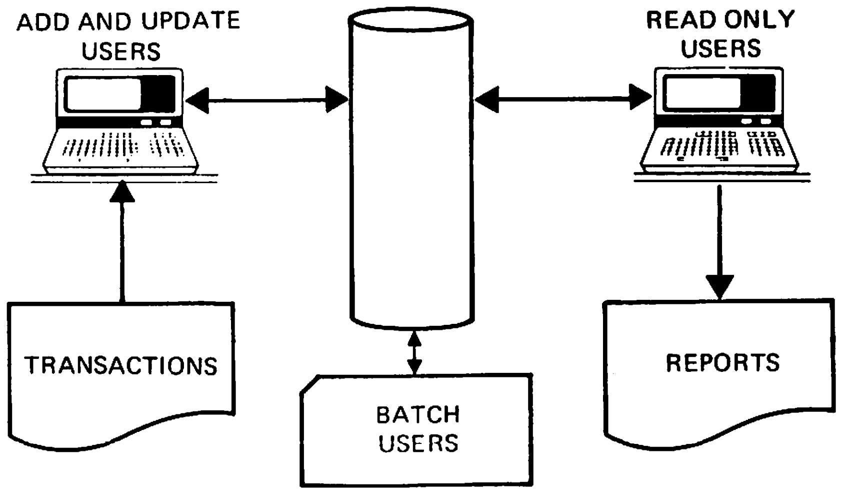
## PRELIMINARY INVESTIGATIONS STRUCTURE - FUNCTIONS - OBJECTIVES OF THE COMPANY

| ORGANIZATIONAL STRUCTURE | BUSINESS FUNCTIONS | OBJECTIVES |
|---|---|---|
|  | Services Provided<br>Products Manufactured | Reduce Costs<br>Increase Business<br>Make A Profit<br>Improve Services<br>New Products |

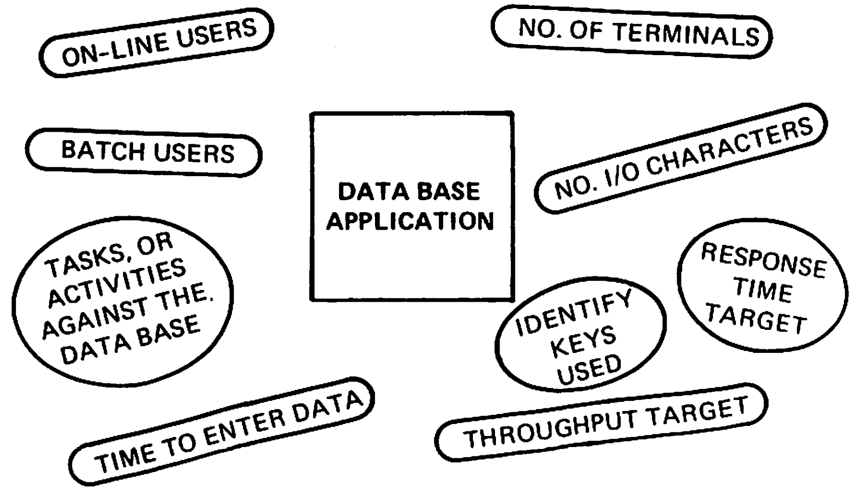
## DESIGN AID #1 COMPUTER/DATA BASE ENVIRONMENT



## DESIGN AID #2 FUNCTION/ACTIVITY/FLOW DIAGRAM



## DESIGN AID #3 USER FUNCTIONS AND REQUIREMENTS



B-02.07

HEWLETT PACKARD

# DATA BASE DESIGN AIDS OVERVIEW

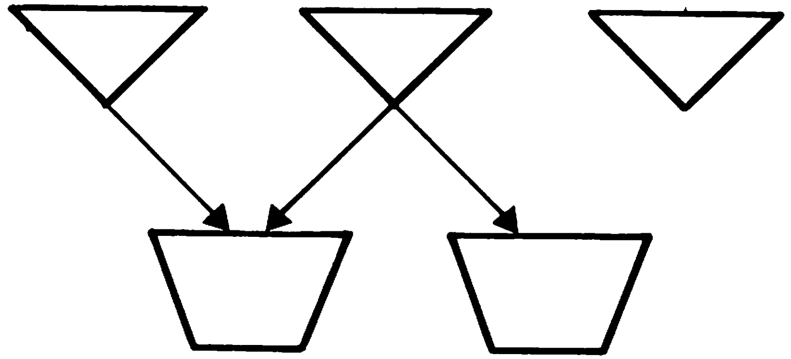
## DESIGN AID #4
## ITEM/FUNCTION MATRIX

FUNCTIONS

| ITEMS | TYPE/ SIZE | SEARCH ITEM | SORT ITEM | UPDATE FREQ. | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

(ADDITIONAL DATA USAGE STATISTICS MAY BE APPROPRIATE)

## DESIGN AID #5
## DATA BASE MODEL



## DESIGN AID #6
## DATA BASE DIRECTORY

| SETS | ITEMS | TYPE/ SIZE | SECURITY | KEY | SORT | P/C | CAP | DESC. |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |

## DESIGN AID #7
## DETAILED ACTIVITY ANALYSIS AGAINST THE DATA BASE

| USER | TRANS ID | NO. TERMS | TIME TO ENTER | NO. I/O CHAR | DB NAME | OPEN MODE | SET NAME | CALL | KEY | ARG |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | |

B-02.08

DESIGN AID #1

DATA BASE ENVIRONMENT

| | | | |
|---|---|---|---|
| | HARDWARE ENVIRONMENT | | |
| COMPANY NAME: ________ | COMPUTER ________ | MODEL ________ | |
| TYPE OF BUSINESS: ________ | MEMORY SIZE ________ | | |
| ADDRESS: ________ | DISCS ________ | CAPACITY ________ | |
| DATA BASE ADMINISTRATOR: ________ | TAPES ________ | B.P.I. ________ | |
| TELEPHONE NUMBER: ________ | PRINTERS ________ | L.P.M. ________ | |
| | NO. TERMINALS ________ | MAKE, MODEL & BAUD RATE ________ | |

| MAJOR ACTIVITIES OR APPLICATIONS | NO. BATCH USERS | NO. ONLINE USERS | NO. READ ONLY USERS | NO. ADD & UPDATE USERS | SUBSYS. OR LANG. USED | PRIORITY |
|---|---|---|---|---|---|---|
| | | | | | | |

DESIGN AID #1

DATA BASE ENVIRONMENT

COMPANY NAME: ACME WIDGETS
TYPE OF BUSINESS: WIDGET DISTRIBUTOR
ADDRESS: CUPERTINO, CALIF.
DATA BASE ADMINISTRATOR: O J. LARSON
TELEPHONE NUMBER: (408) 253-1234

HARDWARE ENVIRONMENT

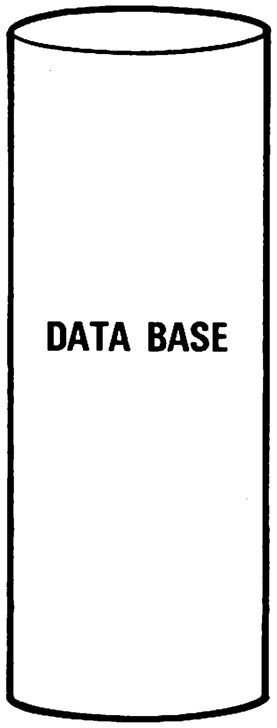
COMPUTER HP 3000 SERIES II MODEL 8
MEMORY SIZE 512 KB
DISCS (2) 7920 CAPACITY 50 MB EAC
TAPES (2) 7970 B.P.I. 1600
PRINTERS 30133A L.P.M. 600
NO. TERMINALS 46 MAKE, MODEL & BAUD RATE HP 2645 9600 Baud

| MAJOR ACTIVITIES OR APPLICATIONS | NO. BATCH USERS | NO. ONLINE USERS | NO. READ ONLY USERS | NO. ADD & UPDATE USERS | SUBSYS. OR LANG. USED | PRIORITY |
|---|---|---|---|---|---|---|
| ORDER PROCESSING | 1 | 13 | 2 | 11 | COBOL/ QUERY | |
| INVENTORY CONTROL | 1 | 5 | 3 | 2 | COBOL | |
| ACCOUNTS RECEIVABLE | 1 | 3 | 1 | 2 | COBOL | |
| ACCOUNTS PAYABLE | 1 | 3 | 2 | 1 | COBOL | |
| PAYROLL | 2 | 6 | 4 | 2 | COBOL | |
| PERSONNEL/ADMIN. | 2 | 6 | 4 | 2 | COBOL | |
| COBOL PROGRAM DEV. | 2 | 10 | | | COBOL | |

## DESIGN AID #2

## USER FUNCTION FLOW DIAGRAM

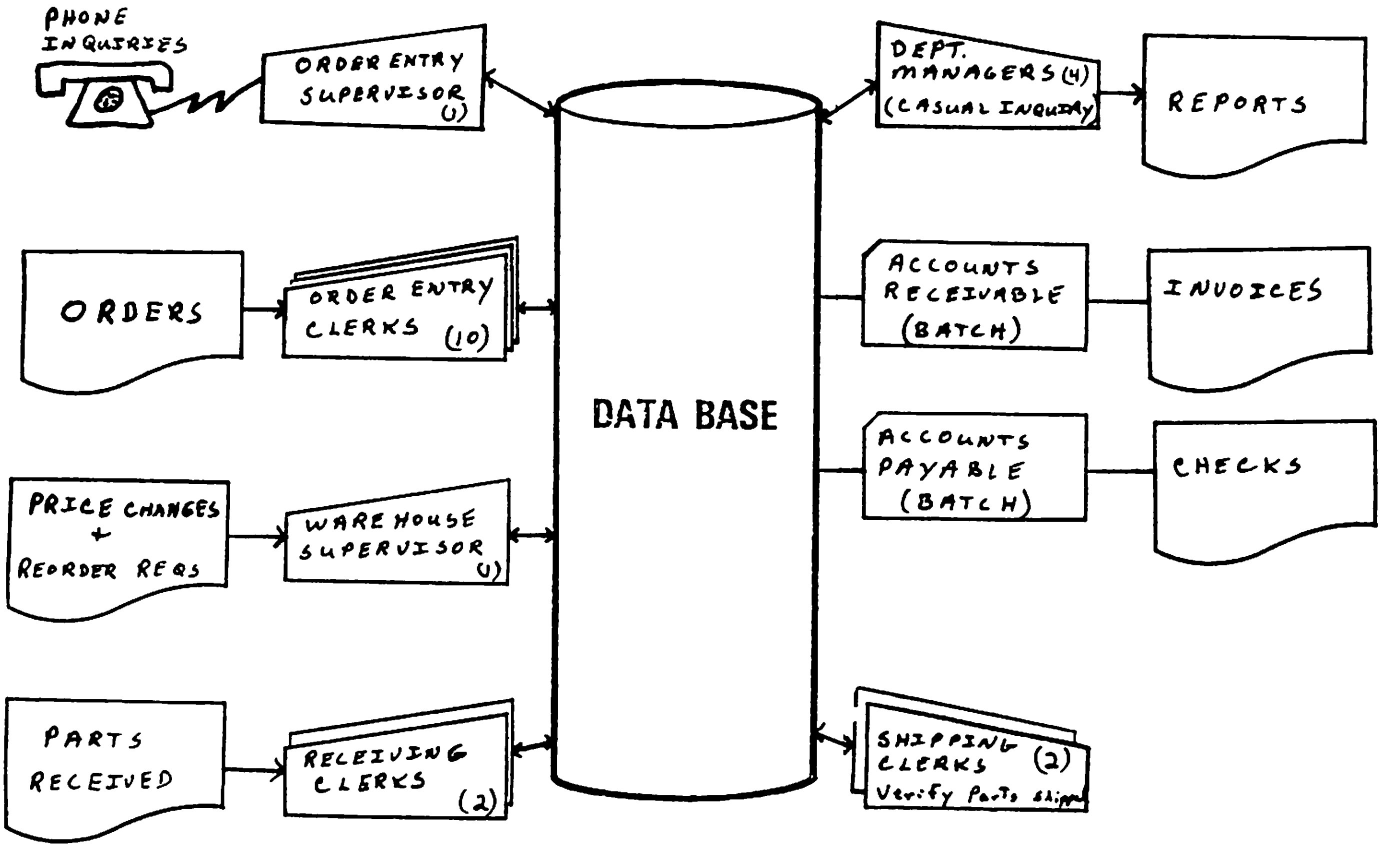
**INDICATE, WITH SYMBOLS, THE DATA FLOW AND ACTIVITY AGAINST THE DATA BASE.**



# DESIGN AID #2

## USER FUNCTION FLOW DIAGRAM

INDICATE, WITH SYMBOLS, THE DATA FLOW AND ACTIVITY AGAINST THE DATA BASE.


PHONE INQUIRIES
ORDER ENTRY SUPERVISOR (1)
ORDERS
ORDER ENTRY CLERKS (10)
PRICE CHANGES + REORDER REQS
WAREHOUSE SUPERVISOR (1)
PARTS RECEIVED
RECEIVING CLERKS (2)
DATA BASE
DEPT. MANAGERS (4) (CASUAL INQUIRY)
REPORTS
ACCOUNTS RECEIVABLE (BATCH)
INVOICES
ACCOUNTS PAYABLE (BATCH)
CHECKS
SHIPPING CLERKS (2) Verify Parts Shipped

DESIGN AID #3

USER FUNCTIONS AND REQUIREMENTS

DATA BASE APPLICATION: ______________________________

OBJECTIVES: ______________ ______________

______________ ______________

______________ ______________

| USER NAME OR BATCH PROGRAM | DATA BASE FUNCTION OR ACTIVITY | NO. ONLINE TERMINALS | NO. OF I/O CHAR. | KEY INFORMATION IDENTIFIER | ESTIMATED TIME TO ENTER DATA | HOURLY THRUPUT TARGET | RESPONSE TIME TARGET |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

DESIGN AID #3

USER FUNCTIONS AND REQUIREMENTS

DATA BASE APPLICATION: ORDER PROCESSING/INV. CONTROL/ACCTS PAY + REC

OBJECTIVES: ENTER ORDERS — KEEP INVENTORY UPDATED

HANDLE PHONE INQUIRIES O/L — BILLING (ACCTS RECEIVABLE)

KEEP PRICES & INVENTORY CURRENT — ACCTS PAYABLE

| USER NAME OR BATCH PROGRAM | DATA BASE FUNCTION OR ACTIVITY | NO. ONLINE TERMINALS | NO. OF I/O CHAR. | KEY INFORMATION IDENTIFIER | ESTIMATED TIME TO ENTER DATA | HOURLY THRUPUT TARGET | RESPONSE TIME TARGET |
|---|---|---|---|---|---|---|---|
| O.E. SUPER | CHK ORDERS | 1 | 100 | ORDER-NO or COMP-ID | 15-25 secs | LOW VOLUME | 2-5 secs |
| O.E. CLERKS | ADD/UPDATE ORDERS | 10 | 150 | ORDER-NO or COMP-ID | 35-50 secs | 60/HOUR EACH | 1-3 secs |
| WAREHOUSE SUPER | READ/UPDATE | 1 | 100 | ITEM-NO | 15-25 secs | LOW VOLUME | 2-5 secs |
| RECEIVING CLERKS | UPDATE INVENTORY | 2 | 100 | ITEM-NO | 25-40 secs | 60/HOUR EACH | 1-3 secs |
| DEPT. MGRS | CASUAL INQ/ REPORTS | 4 | 200 | COMP-ID, ORDER-NO. | 15-40 secs | LOW VOLUME | 2-5 secs |
| SHIPPING CLERKS | VERIFY PARTS SHIPPED | 2 | 100 | ITEM-NO | 15-30 secs | 60/HOUR EACH | 2-5 secs |
| ACCTS RECEIVABLE | READ AND PRINT BILLING INFORMATION (INVOICES) | — | — | COMP-ID | — | BATCH | — |
| ACCTS PAYABLE | PRINT CHECKS | — | — | COMP-ID | — | BATCH | — |

## DESIGN AID #4

## ITEM/FUNCTION MATRIX

FUNCTIONS

| ITEM NAME | TYPE/SIZE | SEARCH ITEM | SORT ITEM? | UPDATE FREQUENCY | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | |

DESIGN AID #4

ITEM/FUNCTION MATRIX

| ITEM NAME | TYPE/SIZE | SEARCH ITEM | SORT ITEM? | UPDATE FREQUENCY | OE SUPER | OE CLERK | WAREHOUSE SUPER | RECEIVING CLERK | DEPARTMENT MGRS | SHIPPING CLERK | ACCOUNTS REC/PA |
|---|---|---|---|---|---|---|---|---|---|---|---|
| COMP-ID | X8 | X | | | X | X | | | | | X |
| CO-NAME | X50 | | | | X | X | | | A | | X |
| VAL | Z10 | | | HIGH | X | | | | L | | X |
| AMT-LD | Z10 | | | HIGH | X | | | | L | | X |
| ORD-NO | X8 | X | | | | X | | | | X | X |
| ITM-NO | X8 | X | | | | X | X | X | | X | X |
| CUR-CNT | Z6 | | | HIGH | | | X | X | ↓ | | |
| UNT-CST | Z6 | | | HIGH | | | X | X | | | |
| TAX | Z6 | | | HIGH | | | X | X | | | |
| REC-CDE | X4 | | | | | | X | | | | |
| PAC-CDE | X4 | | | | | | X | | | | |
| SHP-CDE | X2 | | | | | | X | | | | |
| SLS-PRC | Z6 | | | | | | X | | | | |
| DOL-LD | Z10 | | | HIGH | | | X | | | | |
| ORD-DTE | X6 | | X | | | X | | | | | |
| REC-DTE | X6 | | | | | X | | | | | |
| ORD-AMT | Z10 | | | | | X | | | | | |
| SHP-DTE | X6 | | X | | | | | | | X | |
| NO-ITM | J | | | | | | | | ↓ | X | |

FUNCTIONS

# DESIGN AID #5
# DATA BASE MODEL

MASTER DATA SETS

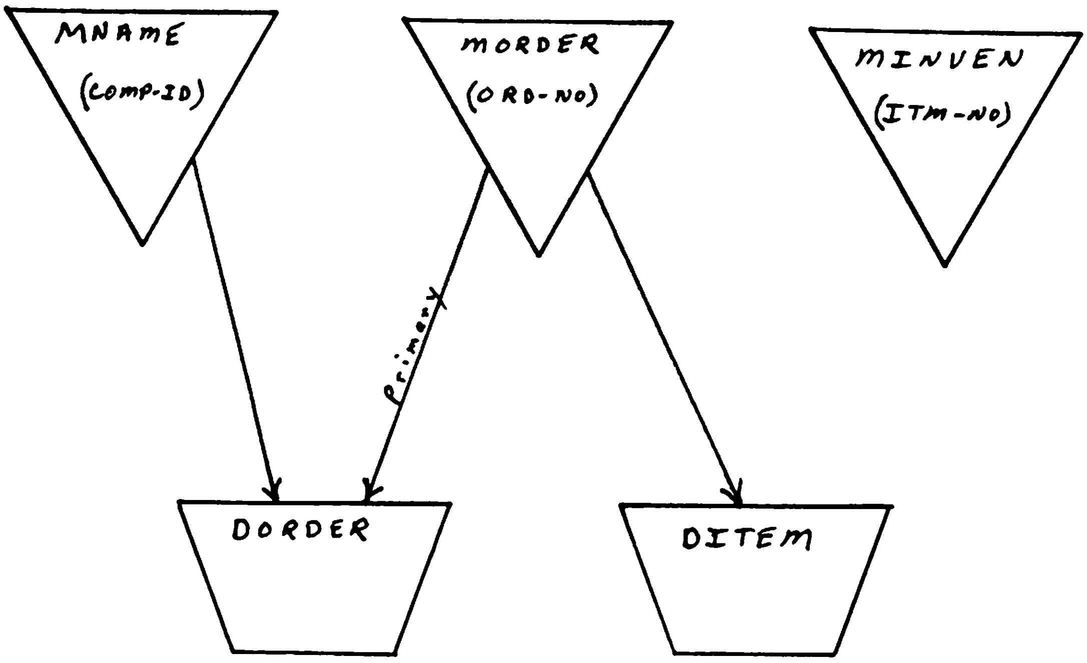
DETAIL DATA SETS

HEWLETT PACKARD

# DESIGN AID #5
# DATA BASE MODEL

MASTER DATA SETS

DETAIL DATA SETS



DESIGN AID #6

# Data Base Directory

PAGE ___ OF ___

DATA BASE NAME ____________________

| DATA SET NAME | | | ITEM | | TYPE / DESIG | READ/WRITE SECURITY | KEY | SORT | P/C | CAP | DESCRIPTION |
|---|---|---|---|---|---|---|---|---|---|---|---|
| # | MASTER | DETAIL | | NAME | | | | | | | |
| | | | | | | | | | | | |

DESIGN AID #6

# Data Base Directory

PAGE 1 OF 1

**DATA BASE NAME** "ORDENT"

| DATA SET NAME | | | ITEM | | TYPE | READ/WRITE | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| # | MASTER | DETAIL | # | NAME | DESIG | SECURITY | KEY | SORT | P/C | CAP | DESCRIPTION |
| 1 | MNAME | | | | M | | | | | 100 | Customer Name File |
| | | | 1 | COMP-ID | X8 | | X | | 1 | | Truncated Company Name |
| | | | 2 | CO-NAME | X50 | | | | | | Full Company Name |
| | | | 3 | VAL | Z10 | | | | | | Value Current Orders |
| | | | 4 | AMT-LD | Z10 | | | | | | Account Receivables |
| 2 | MORDER | | | | A | | | | | 100 | ORDERS OUTSTANDING |
| | | | 5 | ORD-NO | X8 | | X | | 2 | | Order Number |
| 3 | MINV EN | | | | M | | | | | 100 | INVENTORY FILE |
| | | | 6 | ITM-NO | X8 | | X | | 0 | | Item Number |
| | | | 7 | CUR-CNT | Z6 | | | | | | No. of Items on Hand |
| | | | 8 | UNT-CST | Z6 | | | | | | Unit Cost of Item |
| | | | 9 | TAX | Z6 | | | | | | Total Tax on Item |
| | | | 10 | REC-CDE | X4 | | | | | | Reception by Inventory |
| | | | 11 | PAC-CDE | X4 | | | | | | Packing Code |
| | | | 12 | SHP-CDE | X2 | | | | | | Shipping Code |
| | | | 13 | SLS-PRC | Z6 | | | | | | Item Sales Price |
| | | | 14 | DOL-LD | Z10 | | | | | | Item Dollar Load |
| 4 | | DORDER | | | | | | | 2 | 100 | CUSTOMER'S ORDERS CHAINS |
| | | | 1 | COMP-ID | X8 | | X | | | | Truncated Company Name |
| | | | 5 | ORD-NO | X8 | | X2 | | | | Order Number |
| | | | 15 | ORD-DTE | X6 | | | S2 | | | Date Order Placed |
| | | | 16 | REC-DTE | X6 | | | | | | Date Order Received |
| | | | 17 | ORD-AMT | Z10 | | | | | | Amount of Order |
| 5 | | DITEM | | | | | | | 1 | 100 | ITEMS CURRENTLY ON ORDER |
| | | | 5 | ORD-NO | X8 | | X | | | | Order Number |
| | | | 6 | ITM-NO | X8 | | | | | | Item NoNumber |
| | | | 18 | SHP-DTE | X6 | | | S | | | Item Shipping Date |
| | | | 19 | NO-ITM | J | | | | | | Number of Items on O[illegible] |

VI-16

## DESIGN AID #7
## DETAILED ACTIVITY AGAINST THE DATA BASE
## (IDEA SCRIPT FILES)
## WORKSHEET

| TRANSACTION INFORMATION | | | | | | | | | TRANSACTION STEPS | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| USER | SCRIPT FILE NAME | USER TRANS NO. | NO PROCS (TERMS) | PAUSE | | NO. ITERATIONS | NO. I/O CHARS | STACK SIZE | STEP NO. | DATA BASE NAME | OPEN MODE | DATA SET NAME | CALL TYPE | GET MODE | KEY NAME | INITIAL ARGUMENT VALUE | NO. OF GETS |
| | | | | MIN. | MAX. | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | |

B-02.21

DESIGN AID #7
DETAILED ACTIVITY AGAINST THE DATA BASE
(IDEA SCRIPT FILES)
WORKSHEET

| TRANSACTION INFORMATION | | | | | | | | | TRANSACTION STEPS | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| USER | SCRIPT FILE NAME | USER TRANS NO. | NO PROCS (TERMS) | PAUSE | | NO. ITERATIONS | NO. I/O CHARS | STACK SIZE | STEP NO. | DATA BASE NAME | OPEN MODE | DATA SET NAME | CALL TYPE | GET MODE | KEY NAME | INITIAL ARGUMENT VALUE | NO O GE |
| | | | | MIN. | MAX. | | | | | | | | | | | | |
| OE-CLERK | ORDSCRPT | 66 | 10 | 35 | 50 | 50 | 150 | 4 | 1 | ORDTST | 1 | MNAME | LOCK | — | — | — | — |
| | | | | | | | | | 2 | — | | MNAME | PUT | — | COMP-ID | COMP6666 | — |
| | | | | | | | | | 3 | — | | DORDER | PUT | | COMP-ID | COMP6666 | — |
| | | | | | | | | | 4 | | | DITEM | PUT | | ORD-NO | ORD6666 | — |
| | | | | | | | | | 5 | | | — | UNLK | — | — | — | — |
| REC-CLERK | | 77 | 2 | 25 | 40 | 50 | 100 | 4 | 1 | ORDTST | 1 | MINVEN | LOCK | — | — | | — |
| | | | | | | | | | 2 | — | — | MINVEN | GETU | 7 | ITM-NO | ITM777 | — |
| | | | | | | | | | 3 | | — | — | UNLK | | — | | — |
| SHP-CLERK | | [illegible] | [illegible] | 15 | 30 | 50 | 100 | 4 | 1 | ORDTST | 1 | DITEM | LOCK | — | — | | |
| | | | | | | | | | 2 | | | DITEM | GETU | 5 | ORD-NO | ORD888 | 5 |
| | | | | | | | | | 3 | | | — | UNLK | | | | |

HEWLETT PACKARD

# IMAGE DATA-BASE EVALUATIVE ANALYZER

A DATA BASE PERFORMANCE MEASUREMENT TOOL

B-02.23

HEWLETT PACKARD

## OVERVIEW OF IDEA

- **ESTIMATES DATA BASE:**
  - LOAD TIME
  - RESPONSE TIME
  - THROUGHPUT
- **PROVIDES INITIAL DESIGN FEEDBACK**
- **PROVIDES FUTURE CHANGE IMPACT**
- **INTERACTIVELY PROMPTS YOU TO CREATE SCRIPT FILES**
- **GENERATES OWN TEST DATA**
- **EXECUTES FROM A SINGLE TERMINAL OR MULTIPLE TERMINALS**
- **PRODUCES SUMMARY OR DETAILED REPORTS**
  - THROUGHPUT RATES
  - RESPONSE TIMES
- **RESIDES IN THE HEWLETT-PACKARD CONTRIBUTED LIBRARY**
- **RUNS ON THE HP 3000 SERIES I, II, AND III COMPUTERS**

## IDEA REQUIREMENTS

- DESIGN MUST BE COMPLETE
- DATA BASE MUST BE CREATED (EMPTY)
- UNDERSTANDING OF THE DATA BASE APPLICATION
- UNDERSTANDING OF IMAGE CALL PROCEDURES
- DETAILED ACTIVITY DETERMINED AGAINST THE DATA BASE (SCRIPT FILES)

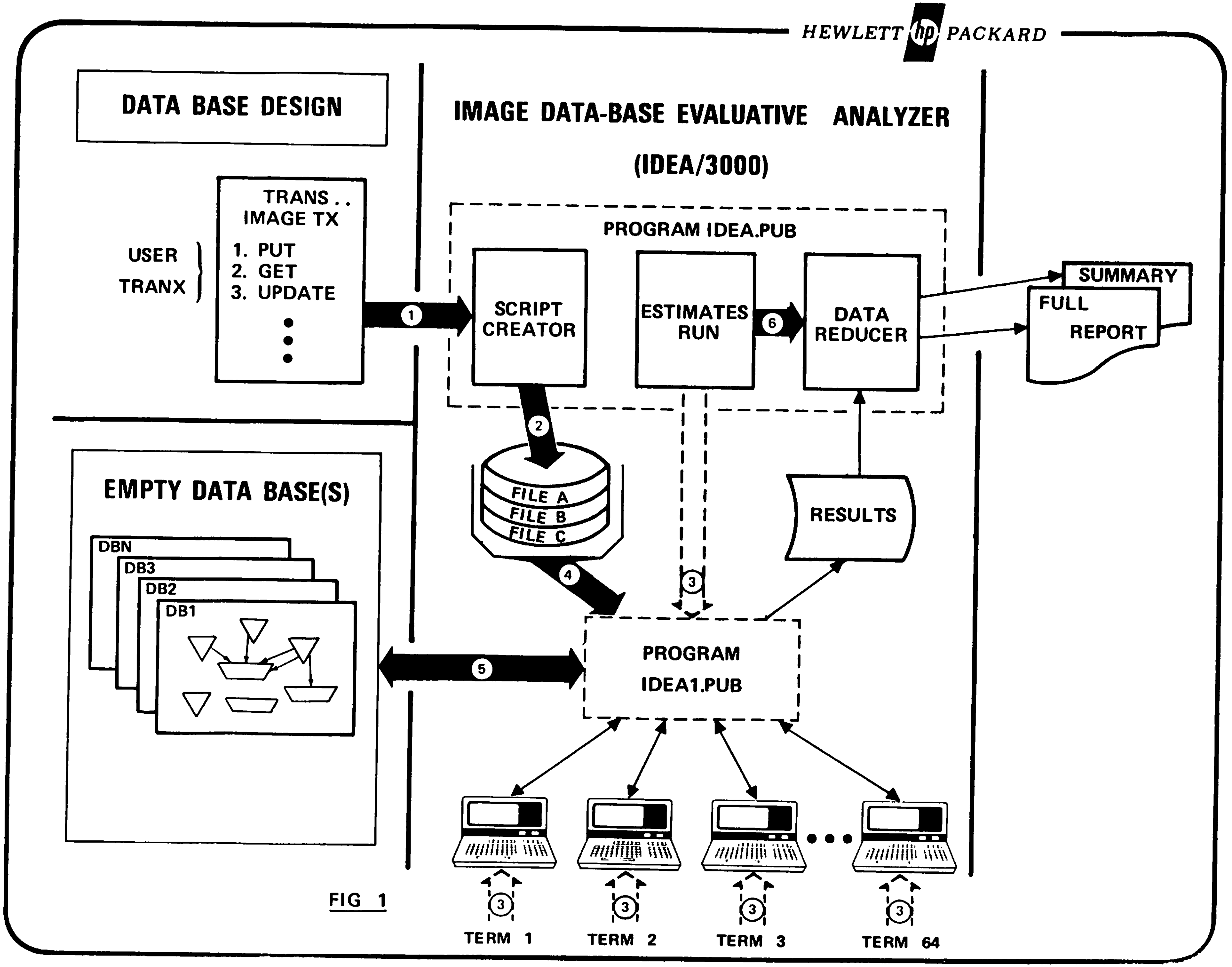

DATA BASE DESIGN
IMAGE DATA-BASE EVALUATIVE ANALYZER
(IDEA/3000)
USER TRANX
TRANS.. IMAGE TX
1. PUT
2. GET
3. UPDATE
PROGRAM IDEA.PUB
SCRIPT CREATOR
ESTIMATES RUN
DATA REDUCER
SUMMARY
FULL REPORT
1
2
6
FILE A
FILE B
FILE C
RESULTS
4
3
EMPTY DATA BASE(S)
DBN
DB3
DB2
DB1
5
PROGRAM IDEA1.PUB
TERM 1
TERM 2
TERM 3
TERM 64

FIG 1

SAMPLE SCRIPT CREATION

```
:RUN IDEA


 SELECT:(IDEA, Version 3, May 1978)
 1 = CREATE SCRIPT
 2 = LOAD RUN
 3 = TEST RUN
 4 = TIMING RUN
 5 = REPORT
 6 = ESTIMATE LOAD TIME
 7 = ERASE DATA BASE
 8 = LIST SCRIPTS
 EXIT
1
 RECORD NUMBER 1
 01   TRANSACTION NUMBER(0<N<97)? 66
 02   NUMBER OF PROCESSES? 10
 03   MINIMUM PAUSE? 32
 04   MAXIMUM PAUSE? 50
 05   NUMBER OF ITERATIONS? 25
 06   NUMBER OF I/O-CHARS? 150
 07   STACK SIZE(0 TO 16K)? 4
 WHICH LINE IS INCORRECT(NONE-CR)? C/R

 RECORD NUMBER 2
 01   TRANSACTION NUMBER  66
 STEP NUMBER 01
 02   DATA-BASE NAME(REF) =  ?
 02   DATA-BASE NAME(REF) = ORDTST
 03   OPEN-MODE? 1
 04   DATASET NAME= ?
 04   DATASET NAME= MNAME
 05   GET,GETD,GETD,PUT,LOCK OR UNLK? LOCK
 WHICH LINE IS INCORRECT(NONE-CR)? C/R

 RECORD NUMBER 3
 01   TRANSACTION NUMBER  66
 STEP NUMBER 02
 02   DATA-BASE NAME(REF) =  ?  ORDTST
 02   DATA-BASE NAME(REF) =      C/R
 04   DATASET NAME= ? MNAME
 04   DATASET NAME=          C/R
 05   GET,GETD,GETD,PUT,LOCK OR UNLK? PUT
 07   KEY NAME? COMP-ID
 08   INITIAL VALUE(AT MOST 20 CHARS)? COMP[illegible]
 WHICH LINE IS INCORRECT(NONE-CR)? C/R

 RECORD NUMBER 4
 01   TRANSACTION NUMBER  66
 STEP NUMBER 03
 02   DATA-BASE NAME(REF) =  ? [illegible]
 02   DATA-BASE NAME(REF) =      C/R
 04   DATASET NAME= ? MNAME
 04   DATASET NAME= ORDERS
 05   GET,GETD,GETD,PUT,LOCK OR UNLK? PUT
 07   KEY NAME? COMP-ID
 08   INITIAL VALUE(AT MOST 20 CHARS)? COMP6666
 WHICH LINE IS INCORRECT(NONE-CR)? C/R
```

SAMPLE SCRIPT FILE

```
661000350050000015                         00XACT                                        04********************0150
660000000000000000  ORDTST                 01LOCKMNAME                                   01********************0001
660000000000000000  ORDTST                 01PUT MNAME           COMP-ID                 01COMP6666            0001
660000000000000000  ORDTST                 01PUT DORDER          COMP-ID                 01COMP6666            0001
660000000000000000  ORDTST                 01PUT DITEM           ORD-NO                  01ORD6666             0001
770200250040000015                         00XACT                                        04********************0100
770000000000000000  ORDTST                 01LOCKMINVEN                                  01********************0001
770000000000000000  ORDTST                 01GETUMINVEN                                  07ITM777              0001
770000000000000000  ORDTST                 01UNLKMINVEN                                  01********************0001
880200150030000015                         00XACT                                        04********************0100
880000000000000000  ORDTST                 01LOCKDITEM                                   01********************0001
880000000000000000  ORDTST                 01GETUDITEM           ORD-NO                  05ORD888              0005
880000000000000000  ORDTST                 01UNLKDITEM                                   01********************0001
```

# SAMPLE OF IDEA OPTIONS

```
:RUN IDEA


SELECT:(IDEA, Version 3, May 1978)
1 = CREATE SCRIPT
2 = LOAD RUN
3 = TEST RUN
4 = TIMING RUN
5 = REPORT
6 = ESTIMATE LOAD TIME
7 = ERASE DATA BASE
8 = LIST SCRIPTS
EXIT
2

SCRIPT FILE NAME?ORDSCRPT
STARTING MASTER LOAD PROCESS
STARTING DETAIL LOAD PROCESS
LOADING COMPLETE

SELECT:(IDEA, Version 3, May 1978)
1 = CREATE SCRIPT
2 = LOAD RUN
3 = TEST RUN
4 = TIMING RUN
5 = REPORT
6 = ESTIMATE LOAD TIME
7 = ERASE DATA BASE
8 = LIST SCRIPTS
EXIT
3

SCRIPT FILE NAME? ORDSCRPT
TRANSACTION 66 CHECKED
TRANSACTION 77 CHECKED
TRANSACTION 88 CHECKED
TEST RUN O.K.  DO YOU WANT A TIMING RUN(Y/N)? N


SELECT:(IDEA, Version 3, May 1978)
1 = CREATE SCRIPT
2 = LOAD RUN
3 = TEST RUN
4 = TIMING RUN
5 = REPORT
6 = ESTIMATE LOAD TIME
7 = ERASE DATA BASE
8 = LIST SCRIPTS
EXIT
4

SCRIPT FILE NAME?ORDSCRPT
 14 PROCESSES REQUIRED
12345678901234
STARTING TIMING RUN
END OF TIMING RUN
SORTING IDEALOG
STARTING REPORT
```

SAMPLE OF IDEA OPTIONS (Cont.)

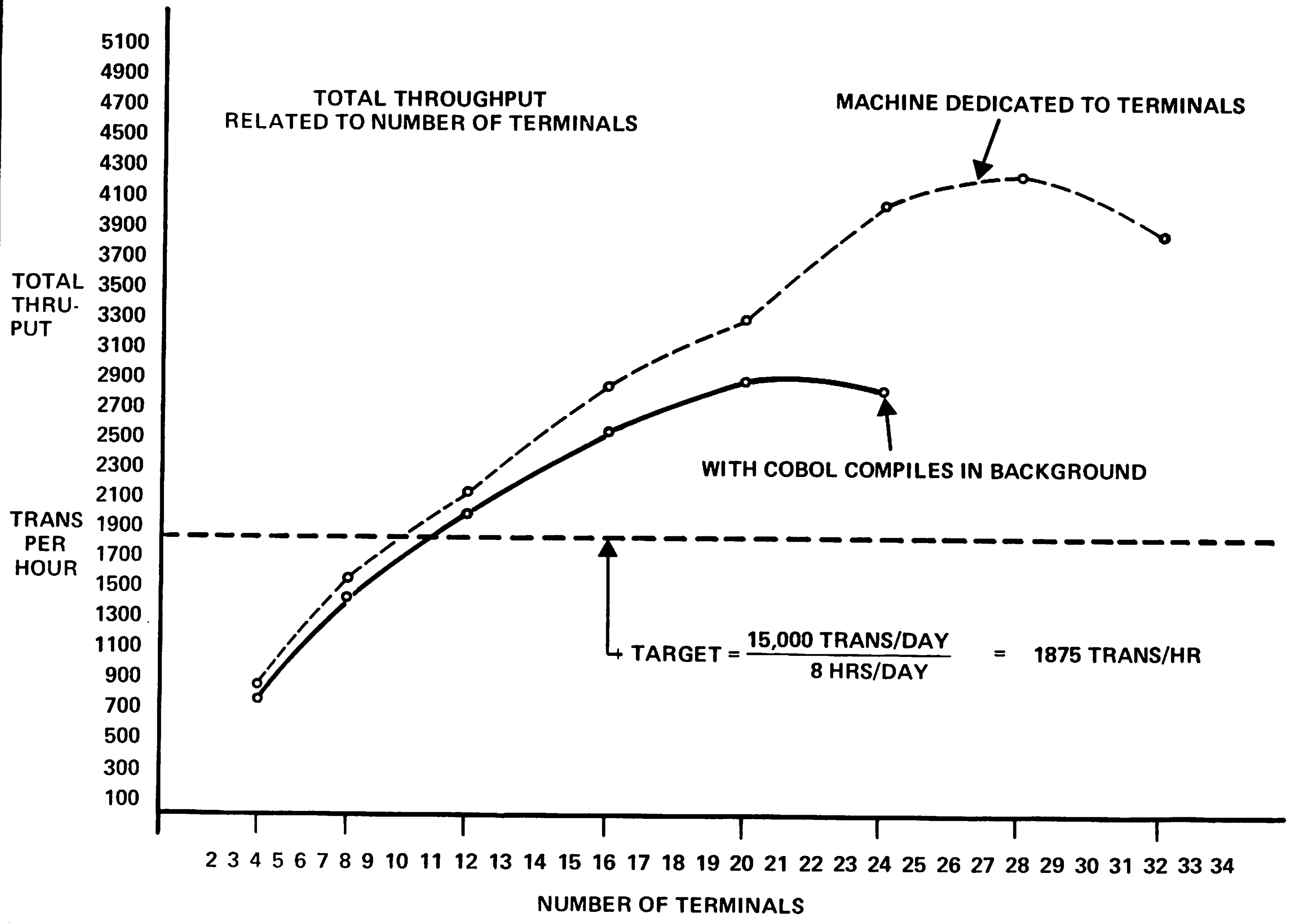
```
1

SELECT:(IDEA, Version 3, May 1978)
1 = CREATE SCRIPT
2 = LOAD RUN
3 = TEST RUN
4 = TIMING RUN
5 = REPORT
6 = ESTIMATE LOAD TIME
7 = ERASE DATA BASE
8 = LIST SCRIPTS
EXIT
6
WHICH DATA BASE? ORDTST

APPROX. LOAD/DBLOAD TIME =  0  HRS    21  MINS

NOTE:  This is an estimated time
based on 0.15 sec/PUT and 0.1 sec
per update per chain in a DETAIL.
Chain sorting times are excluded.


SELECT:(IDEA, Version 3, May 1978)
1 = CREATE SCRIPT
2 = LOAD RUN
3 = TEST RUN
4 = TIMING RUN
5 = REPORT
6 = ESTIMATE LOAD TIME
7 = ERASE DATA BASE
8 = LIST SCRIPTS
EXIT
7
WHICH DATA BASE? ORDTST
ERASED

SELECT:(IDEA, Version 3, May 1978)
1 = CREATE SCRIPT
2 = LOAD RUN
3 = TEST RUN
4 = TIMING RUN
5 = REPORT
6 = ESTIMATE LOAD TIME
7 = ERASE DATA BASE
8 = LIST SCRIPTS
EXIT
8
LISTING ON
3

SCRIPT FILE NAME? ORDSCRPT
TRANSACTION 66 CHECKED
```

## SAMPLE IDEA OUTPUT

```
MON. OCT 16, 1978,  8:32 AM                      SUMMARY REPORT                                      PAGE   1
IDEA V.0 SERII                                     (AVERAGES)

 XACT/      NO OF     RECORDING      THINK-TIME     TOTAL-DELAY    RESPONSE      CYCLE-TIME    TRANSACTIONS
PROCESS     TIMES     TIME    (A)                       (B)        TIME   (C)     (A+B+C)        PER HOUR

 66/ 1       24        .0 SECS       43.1 SECS      43.9 SECS      .9 SECS      44.9 SECS         80
 66/ 2       26        .0 SECS       41.5 SECS      41.8 SECS      .5 SECS      42.4 SECS         85
 66/ 3       24        .0 SECS       43.8 SECS      44.3 SECS     1.0 SECS      45.4 SECS         79
 66/ 4       25        .0 SECS       42.1 SECS      42.3 SECS      .6 SECS      43.0 SECS         84
 66/ 5       25        .0 SECS       42.9 SECS      43.3 SECS      .8 SECS      44.2 SECS         81
 66/ 6       25        .0 SECS       42.2 SECS      42.6 SECS      .9 SECS      43.5 SECS         83
 66/ 7       25        .0 SECS       42.2 SECS      42.4 SECS     1.7 SECS      44.2 SECS         81
 66/ 8       25        .0 SECS       41.7 SECS      42.4 SECS      .6 SECS      43.1 SECS         84
 66/ 9       25        .0 SECS       42.5 SECS      43.2 SECS      .9 SECS      44.3 SECS         81
 66/10       25        .0 SECS       42.3 SECS      42.7 SECS      .8 SECS      43.6 SECS         83
 77/ 1       36        .0 SECS       32.6 SECS      32.8 SECS      .9 SECS      33.8 SECS        107
 77/ 2       36        .0 SECS       32.8 SECS      33.0 SECS      .8 SECS      33.9 SECS        106
 88/ 1       47        .0 SECS       21.8 SECS      22.0 SECS      .9 SECS      23.0 SECS        157
 88/ 2       44        .0 SECS       23.3 SECS      23.7 SECS     1.0 SECS      24.8 SECS        145
                                                                 -----------
                    AVG RESPONSE-TIME FOR TERMINAL MIX             .9 SECS

                                                                                              ----------
                    TOTAL HOURLY THRU-PUT FOR THE MIX                                            1336
```

HEWLETT PACKARD

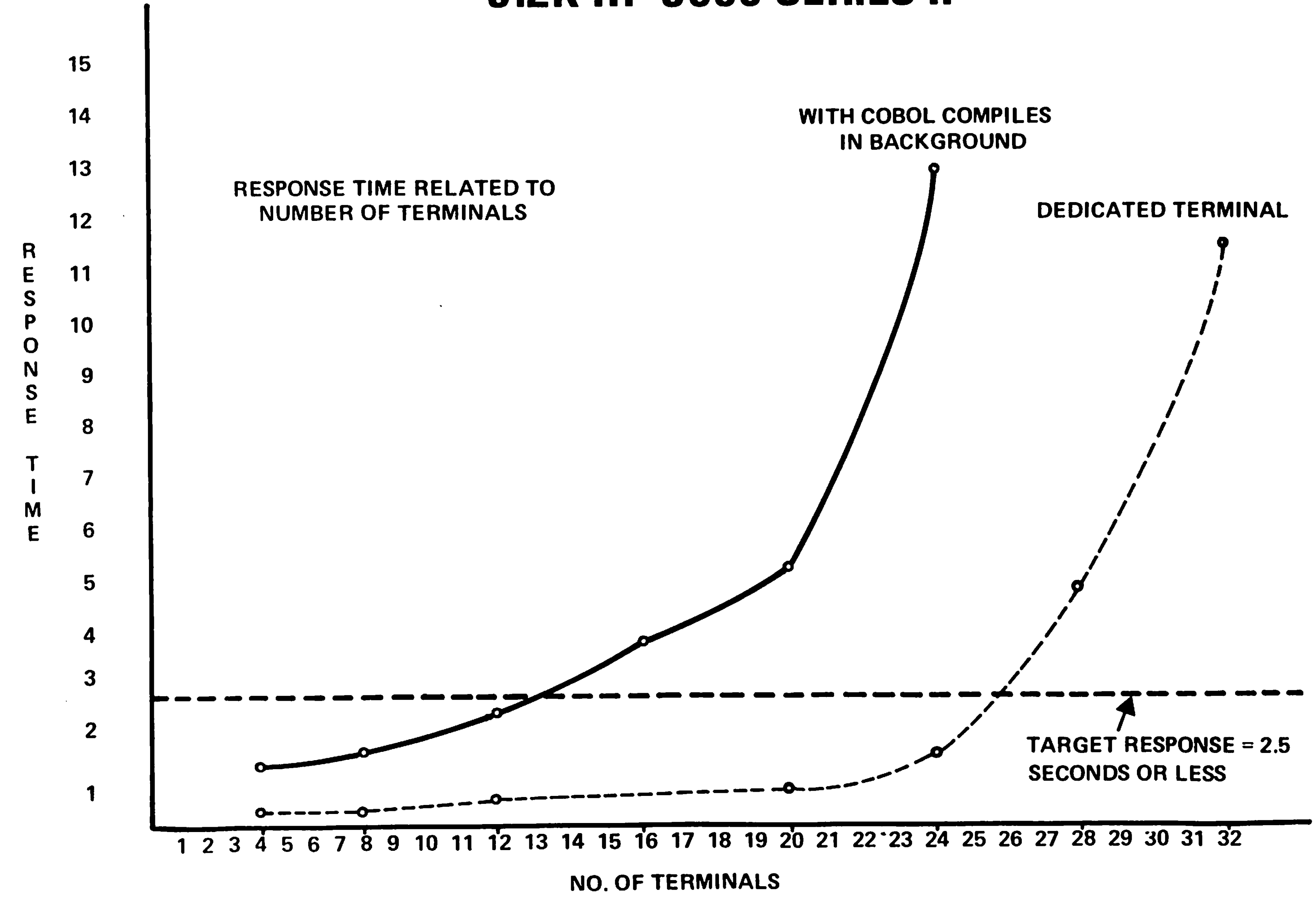
# 512K HP 3000 SERIES II



$$\text{TARGET} = \frac{\text{15,000 TRANS/DAY}}{\text{8 HRS/DAY}} = \text{1875 TRANS/HR}$$

HEWLETT PACKARD

# 512K HP 3000 SERIES II



## DISADVANTAGES OF PERFORMANCE MEASUREMENT

- **DOES NOT TAKE INTO CONSIDERATION THE APPLICATION OVERHEAD SUCH AS:**
  - APPLICATION PROGRAM SOURCE LANGUAGE
  - DATA EDITING ROUTINES
  - NUMBER CRUNCHING
- **DATA BASE IS USUALLY NOT FULLY LOADED**
- **DATA BASE ACTIVITY IS USUALLY "BEST GUESS" SITUATION**

# ADVANTAGES OF PERFORMANCE MEASUREMENT

- NOT A SIMULATION - PERFORMS THE ACTUAL DISK I/O
- FEASIBILITY OF DESIGN MAY BE DETERMINED WITHOUT WRITING APPLICATION PROGRAMS
- DEDICATED COMPUTER SYSTEM NOT REQUIRED
- AIDS IN DETERMINING HARDWARE CONFIGURATION REQUIREMENTS
- ESTIMATES "BEST CASE" THROUGHPUT AND RESPONSE TIME
- MAY BE USED TO MODEL CURRENT DATA BASE ACTIVITY AND THEN THE IMPACT OF FUTURE DESIGN CHANGES

HEWLETT PACKARD

## SUMMARY

- DATA BASE DESIGNER COULD BENEFIT FROM A COMMON DATA BASE DESIGN METHODOLOGY
- DATA BASE DESIGNERS NEED A TOOL TO MEASURE PERFORMANCE <u>BEFORE</u> APPLICATION IMPLEMENTATION
- HEWLETT-PACKARD HAS BOTH !

# IMAGE DATA BASE DESIGN AIDS

## INTRODUCTION

One of the concerns of the data base designer is the lack of a data base design methodology. The following data base design aids have been developed to provide some direction and a logical approach to assist the data base designer in this most important phase in the implementation of a data base.

The approach taken here is by no means the only way to design a data base. It is simply a logical approach starting by gathering the general overall information about the host computer system and the data base applications and then working down to the detail of the actual DBMS calls against the data base. The main benefit of using these design aids is that they force the designer to gather the pertinent information related to the data base application so intelligent design decisions can be made.

It is assumed that the users of these design aids have a good understanding of IMAGE.

## DATA BASE DESIGN AIDS

### Preliminary Investigations

Before beginning the data base design process, the data base designer should first become familiar with the organizational structure, business functions, and objectives of the company and the specific group for whom the data base is being designed. This is very important especially if this data base will be used by upper or functional management to assist in the day to day decision making process.

### Design Aid #1 - Computer/Data Base Environment

The purpose of this design aid is to get anoverall picture of the environment in which the data base will be residing. This aid provides information on the company, the computer hardware, and the major activities or applications that will be using the computer system and the data base.

### Design Aid #2 - User Function Flow Diagram

The purpose of this design aid is to identify the data base users and their respective functional activities relating to a specific data base. The forms used and the number of users invovled in each type of function are shown on a flow diagram. This design aid is useful because it provides an indication of the overall activities being applied against a data base.

The key point to remember here is that each arrow touching the side of the data base symbol represents a transaction against the data base. This transaction may consist of a single DBMS call or a series of DBMS calls. There will be an opportunity to identify these specific calls later, in design aid number 7, after the data base design has been established.

## Design Aid #3 - User Functions and Requirements

The purpose of this design aid is to describe the data base functions in more detail. The objectives of the data base should be listed to remind the designer of what the data base is to provide.

Each of the arrows (functions) touching the side of the data base symbol in design aid #2 should now be described further. This includes identifying each user or batch program and describing the data base function or activity, the number of online terminals, the number of I/O characters transmitted to and from the terminals, the key information identifier (the major search item that a data base user will utilize to identify or locate a data base record, e.g., order number, part number, etc.), the estimated time to enter the data, the hourly throughput target, and the response time target.

## Design Aid #4 - Item/Function Matrix

The primary purpose of this design aid is to identify all the items in the data base and then relate them to the functions described in the preceding design aids.

In addition, the item type and size are included because these are useful later in the data base directory. Also, indicating whether the item is a search item (key) is important because the search item is the basis for a manual or automatic master data set. Furthermore, indicating that an item is a sort item assumes a detail data set chain will be sorted by that item. The update frequency of an item can be useful to identify the high activity items which can affect the design of the data base by possibly locating those high activity items together in the same data set.

This is the last design aid before the actual data base design decisions are made. Additional data usage statistics may be appropriate before the final design decisions are made.

## Design Aid #5 - Data Base Model

The purpose of this design aid is to provide a visual representation of the data base by showing the data set relationships. This is where the design decisions are made! The data base designer must now "earn his keep" by assimilating all the information gathered in the previous design aids as well as drawing on prior data processing experience to come up with a design that satisfies the needs of the data base users.

If there are any words of wisdom to assist the designer in making these important design decisions, they are, "There is no perfect design!". The data base designer is usually well aware of the fact that the needs and objectives of a company or organization within a company are constantly changing.

Design Aid #6 - Data Base Directory

The purpose of this design aid is to provide a means of writing down the structure of the data base. Later, this structure can easily be translated into the data base SCHEMA using the TEXT EDITOR.

The information requested is self explanatory.

Design Aid #7 - Detailed Activity Against the Data Base

The purpose of this design aid is to indicate the detailed activity against the data base for the purpose of identifying the DBMS calls required to complete a transaction. This information alone may help the experienced designer to estimate the load on the system. This design aid is primarily used as a worksheet for a performance measurement tool called IDEA which is an acronym for IMAGE Database Evaluative Analyzer.

Two versions of IDEA (Series I and Series II) which were written by HP System Engineers are currently available in the Contributed Library (the Series II version will work with the Series III). A new enhanced version of IDEA has been made available to our field system engineers who specialize in performance measurement consulting.

# NEW FEATURES AND LIMITATIONS OF IDEA VERSION #3

1. Each transaction can reference more than 1 data base.

2. Keys must be of type U or X with a maximum length of 254 bytes.

3. Within any transaction, the product of NUMBER OF ITERATIONS and NUMBERS of PROCESSES must be less than 32768.

4. IDEA permits simulation of up to 60 terminals. The actual limit may be less due to the number of data bases and data sets involved and to the system configuration.

5. The maximum record size is 512 words.

6. Modifying the script file must be done under the EDITOR. Remember to KEEP the modified file UNNUMBERED.

7. TIMING RUN processes do not "give up" at the first functional failure. They perform "retries" designed to force success so that they may continue with minimal impact on performance. If, for example, a directed GET fails, the data set is "rewound" and a serial GET performed. This fails only if the data set is empty, in which case the TIMING RUN is terminated.

   NOTE: These "retries" are logged and appear on the REPORT.

8. For each directed GET, a random number between 1 and 100 is used as the address of an entry to be a read. The probability that a "retry" will be required depends, in this case, on the capacity and fullness of the data set being accessed.

9. The processing which handle a transaction are created with a user specified stack size.

10. IDEA can be run remotely.

11. Multiple data bases may be accessed within a single transaction.

12. Key lengths can exceed 20 characters.

13. Mode 2,4, and 5 "GETS" (with or without update or delete) can be multiple GETS.

14. No longer necessary to always perform a data base load.

15. The loading is about 7 times as fast.

16. Data bases can be erased by IDEA.

17. The only "files" the user has to "build" are the data bases.

18. A test run can be made to check the script.

19. A "firsttime" pass through the script is also performed by each process prior to proceeding to the timing portion of the timing run. Puts and deletes are bypassed during this pass. IDEA monitors this on the screen by displaying one of the digits 1,2,3,4,5,6,7,8,9,0 cyclically, and in that order, each time a newly activated process completes this first-time pass. It only creates and activates another one after the preceeding one has performed successfully.

20. IDEA handles all files and data bases so that none of them are left lying around. If the program should abort for any reason, this will generally not be true.

21. The timing processes all terminate when any one of them terminates. They do this without logging any more timing records.

22. An impatient user can also force early termination by entering Control-Y at any time after the timing run has begun. The response to the Control-Y may be quite slow due to the resource cleanup which transpires.

23. The logging file (IDEALOG) is sorted only once into the sort file (IDEASORT)/ Both files have 108 byte records with a blocking factor of 7.

24. The timing processes all close the input script file to release the system resources tied up by leaving them open.

25. The timing processes append share the IDEALOG file in multi-access mode. This minimizes the number of resources needed to support the logging function.

26. IDEA and the timing processes communicate via a job control word (JCW). Local rins are used to control access to the user's terminal, and the JCW when writing II.

27. A local rin is also used to queue up each timing process until they all have been successfully activated.

28. The timing process does not give up at the first functional failure; it performs recovery style retries suitable to the function. Directed gets, for example, are implemented by the generation of a random number between 1 and 100 which is used as the address of the record to be read. If this fails, for any reason, the data set is "rewound" and a serial get is performed. This will succeed unless the data set is empty, in which case the timing run is terminated.

Helpful Hints

1. It is best to run IDEA stand-alone. This permits you to obtain timing data not impacted by other processes. It also maximizes the probability that the system resources required for a given TIMING RUN will be avaiable.

2. If you wish to test a given script with a varying number of processes, start with the maximum and modify the script for lower values on subsequent runs. In this manner, the starting script can be used to LOAD the data base once so that all subsequent runs can be performed without reloading.

3. Scripts with PUTs and/or DELETEs are likely to encounter problems during TIMING RUNs which may lead to early termination. For PUT scripts, problems include full data sets, duplicate masters, absence of a required chain head.

4. For DELETE scripts, problems include empty data sets or attempting to delete a master with related detail entries.

5. Processes are launched for each transaction in the same order as the transactions are defined in the script. By entering the transactions with the slowest cycle time (including THINK time) first, all processes will get into play as early as possible. This will make the resulting statistics most meaningful and with a minimum number of iterations.

# IMAGE's COMING OF AGE: Breaking free from restrictions to Data-Base transformations.

F. ALFREDO REGO

Chairman, Software Department
Instituto de Informática y Ciencias de Computación
Universidad Francisco Marroquín
6a. Avenida 0-28, zona 10
Guatemala, GUATEMALA.

## ABSTRACT

A computerized data base should reflect an organization's way of behaving. As real-world circumstancies change, forcing the organization to adopt new ways and abandon old ones, the data base should also adapt itself.

Hewlett-Packard provides tools, such as DBUNLOAD and DBLOAD, which allow a limited set of transformations to IMAGE/3000 data bases. But these tools do not lend themselves to the easy implementation of the radical transformations that are sometimes necessary. The restrictions of these tools, we feel, are analogous to the do's and don'ts imposed on children by loving parents.

Taking into consideration that children (just like computer users) eventually come of age and will do their own thing despite formidable restrictions, we have developed a software system called "DATABASE.UTILITY" to help IMAGE/3000 users out of their data-base transformation predicaments.

"DATABASE.UTILITY" is an MPE 'group.account' that contains a set of software modules designed specifically to allow a large selection of transformations to IMAGE/3000 data bases without having to mess around with magnetic tapes or schema recompilations. And, in good parental fashion, this system also keeps a watchful eye for any possible difficulties that might potentially upset the health, consistency and integrity of the "adolescent" data base.

## KEYWORDS AND PHRASES

Concurrent data-base operation and evolution, data-base adaptability, data-base consistency, data-base conversion, data-base design, data-base integrity, data-base redesign, data-base restructuring, data-base management systems, DBMS's, data-base transformation, Hewlett-Packard's IMAGE/3000 Data-Base Management System, root file transformation, schema changes.

## MOTIVATION FOR THE DEVELOPMENT OF "DATABASE.UTILITY"

How can I be ABSOLUTELY sure that my data-base design is perfect? How can I GUARANTEE that I will NEVER have to change it to meet unexpected shifts in my organization's way of doing things?

If I can not answer these questions to my satisfaction, then what type of tuning (and fine-tuning) tools do I _need_ to facilitate the constant and inevitable evolution of my data base?

What type of questions worry me about the tools I _have_ currently available to me? And what type of questions linger in my mind as I dream of better and more effective ways to do what I _have_ to do anyway?

- Why do I have to COMPLETELY STOP the operation of my live data base, even when I only want to make very slight changes like password reassignments? Could I maintain concurrent data-base access while I do certain non-radical transformations or while I radically transform data sets that are not being currently accessed? ("DATABASE.UTILITY" ANSWERS: yes.)
- Why do I have to spend (a sometimes very long) time to DBUNLOAD my WHOLE data base to magnetic tape before I transform my schema (assuming, of course, that I do not want to lose the live data I presently have!)? Could I skip the whole DBUNLOAD trip? ("DATABASE.UTILITY" ANSWERS: yes.)
- Why do I have to spend (a sometimes even longer) time to DBLOAD my previous data base, even though I merely want to optimize the storage locations of a primary path's entries? Could I simply reshuffle these entries without having to think and worry about the consequences of having to reshuffle the whole data base as well? ("DATABASE.UTILITY" ANSWERS: yes.)
- Why do I have to PURGE my entire data base, when all I want is to change the name of a data item? Could I simply make changes such as this without having to kill (and then re-issue life to) my data base? ("DATABASE.UTILITY" ANSWERS: yes.)

- Why do I have to EDIT and recompile my schema, when I simply want to change the read/write capabilities of a user class? Could I dinamically do this while the data base continues to earn its living? ("DATABASE.UTILITY" ANSWERS: yes.)
- Why do I have to CREATE, from the newly produced root file, a brand-new data base, if the old one was just fine except for the capacity of a data set? Could I change the capacity of a data set without having to go through this process once more? ("DATABASE.UTILITY" ANSWERS: yes.)
- Why am I at the mercy of subtle schema changes that CAN cause very unpleasant surprises, even after my previous data base has apparently been successfully DBLOADed to my new data base? Could I have some 'editor' which would make sure I do not clobber my schema? Could I know, before I ruin anything, that my data-base transformation request is illegal? Could I have a dialogue with the system to "discuss" the possible consequences of subtle changes in transformation requests? ("DATABASE. UTILITY" ANSWERS: yes.)
- Why do I have to write special application programs whenever I need to transform my data base in ways that are not supported by IMAGE/3000's transformation utilities? Could I have a flexible, non-procedural system that would even assemble data entries from bits and pieces taken from other data entries from the same data base, or from other data bases, or even from good old MPE files? Could I do data-type conversions (from integer to byte, from integer to double-integer, from byte to real, from integer to logical, from floating-point to byte with decimal-point suppression and decimal-place right-justification, etc...) if the source data type does not match the destination data type? ("DATABASE.UTILITY" ANSWERS: yes.)

## DESCRIPTION OF "DATABASE.UTILITY"

"DATABASE.UTILITY" is an MPE 'group.account' with privileged capabilities assigned to it by the computer installation's system manager.

All our design trade-offs have one main objective: to preserve data-base consistency and integrity. We strongly feel the same way about preserving other user's domains and, of course, about preserving the operating system itself! Therefore, all privileged instructions in "DATABASE.UTILITY" are executed in bracketed fashion (that is to say, the programs execute in user, non-privileged mode 99% of the time; whenever it is imperative that privileged instructions be executed, a dynamic call to the

GETPRIVMODE system intrinsic is made immediately BEFORE the privileged instruction; then, a dynamic call to the GETUSERMODE system intrinsic is made immediately AFTER the privileged instruction.)

A good 90% of all module execution times is spent in making reasonably sure that the requested transformations are legal and will not produce unpleasant results. Complete log-on subsystems, analogous to MPE's, check to see that only authorized users access the programs. An IMAGE/3000 data base (of course!) is kept for all programs, users and transformations as applied to the various data bases in an installation.

At the least sigh of trouble, the target data base or data set is purged and the old one can be salvaged.

When necessary, the root file is appropriately "updated"; MPE files are created or purged as needed; data sets are re-organized to include or exclude structural information; data sets and data items are re-numbered if any intermediate elements have been eliminated, etc.

The Data Base Administrator (DBA) can easily obtain an up-to-date picture of the transformed data base by means of QUERY's "FORM" command and our own "PASSES" program. "FORM" lists data sets, data items and paths as defined within the data base's structure. "PASSES" lists passwords and user read/write classes.

## DESCRIPTION OF "DATABASE.UTILITY" FUNCTIONS

### 1) NON-TRANSFORMATIONAL

ASSEMBLE: Assembles data entries ("records") for master or detail data sets from one or more data sets or MPE files. The source data sets may be mixed from several data bases and may be either details or masters. The source MPE files may be accessed sequentially or directly by key.

The program asks all relevant questions, such as data-item types (integer, byte, logical, etc.), beginning byte or word in the source entry/record, etc. If it detects inconsistencies (for instance, if the source data-item type is byte and the target data-item type is double-integer), it explains them and requests instructions to perform one of several possible actions, depending on the particular circumstances: ignore and re-try? do data conversion? ...

2) GLOBAL DATA-BASE TRANSFORMATIONS

COPY: Copies a data base from a source 'group.account' to a destination 'group.account'. The data-base creator MUST RELEASE the data base beforehand.

RELEASE: Releases the root file and all MPE privileged files assigned to a data base, so it may be copied from another 'group.account' (or accessed through QUERY or other application programs when running from other 'group.account')

SECURE: Reverses the effect of "RELEASE", securing the data base from access through other "group.account'.

RENAME: Assigns a new name to a data base. Changes MPE file names as well as internally-kept IMAGE names.

PASSES: Reports, lists, modifies, assigns, re-assigns, takes away, etc., maintenance passwords and read/write passwords and class numbers.

PURGE: Purges the root file and all MPE files assigned to a data base. Loops around asking for other data bases to be purged, instead of ending after having purged a data base as "DBUTIL, PURGE" currently does.

AUDIT: Produces a report of the usage of "DATABASE.UTILITY" programs by user, program, data base, etc.

3) TRANSFORMATIONS OF DETAIL DATA SETS

NEWDTAIL: Adds new detail data sets to the data base (with the appropriate new data items, if needed.)

CAPDTAIL: Modifies the capacity of a detail data set, preserving all current chains and making sure, in the case of a decrease in capacity, that the target capacity is at lease equal to the lowest permissible capacity for the given set's status.

KILLDET: Deletes a detail data set, making sure that it is safe to do so. It optionally dumps the entire set to an MPE file in the format specified by the user (or dumps only those data entries within the data set that meet the boolean specifications given by the user.)

## 4) TRANSFORMATIONS OF MASTER DATA SETS

NEWMASTR: Adds new automatic or manual master data sets to the data base (with the appropriate new data items, if needed.)

CAPMASTR: Modifies the capacity of a master data set, preserving hashing properties for calculated access and chain-head structural information. Reduces synonym occurrences by suggesting program-calculated prime-number capacities in the neighborhood of the user-specified capacity.

KILLMAST: Deletes a master data set, making sure that it is safe to do so and that no chains will be left hanging without chain heads. It optionally dumps the entire set to an MPE file in the format specified by the user (or dumps only those data entries within the data set that meet the boolean specifications given by the user.)

## 5) TRANSFORMATIONS OF DATA ITEMS

NEWITEM: Adds new items to existing data sets.

KILLITEM: Deletes a data item from an existing data set. If all data-set references to a given data item have been deleted, the item is also deleted from the root's item table.

RDEFITEM: Re-defines the type of a data item (from integer to byte, for instance), and does all the appropriate data conversions if necessary. If the new data type has a different word-length count, all data sets that reference the given item are re-organized to reflect the new structure.

## 6) TRANSFORMATIONS OF ELEMENT REFERENCES

NEWNAME: Assigns a new name to a data item or a data set. Checks non-duplicity and legality of new name.

## 7) TRANSFORMATIONS OF ACCESS PATHS

NEWPATH: Defines a new path connecting an existing master data set to an existing detail data set by means of an existing data item (i.e., it upgrades non-key data items to the status of key data items or SEARCH ITEMS.)

CLOSPATH: Deletes a path between a master data set and a detail data set. (i.e., it downgrades key data-items or SEARCH ITEMS to the status of non-key data items.)

SORT: Upgrades non-sort data items to the status of sort data items for a given path.

UNSORT: Downgrades sort data items for a given path to the status of non-sort data items.

PRIMARY: The path most frequently accessed in chained mode should be specified by the Data Base Administrator as the primary path for a detail data set. Should this state of affairs change, the DBA can specify that another path become the primary path for the detail data set by means of this program.

PAVEPATH: Reshuffles the entries of a detail data set so that the entries of each chain within the primary path will be in contiguous storage locations (for efficiency's sake in chained retrieval.)

## -- CONCLUSIONS --

### A) PARTICULAR:

"DATABASE.UTILITY" worries and keeps track of all IMAGE/3000 internal housekeeping, while the data base evolves.

The user is, then, free to concentrate his/her energies on the ONLY housekeeping task that really matters to him/her: THE DATA BASE'S ACCURATE REFLECTION OF THE ORGANIZATION'S WAY OF DOING BUSINESS.

The user can, now, specify WHAT he wants to have in his data base, knowing that tomorrow he can easily re-specify his requirements without having to fear lack of compatibility. HOW this is accomplished is the responsibility of "DATABASE.UTILITY".

### B) GENERAL:

Our research and development team has concentrated its efforts on helping users of Data Base Management Systems realize the tremendous potential of this emerging computer technology.

In this report we mention only some of the projects that keep us busy and happy in the Land of Eternal Spring, Guatemala. Currently, we have software systems in various stages of development. The wide range of status is illustrated by the fact that some have been successfully installed at customer sites after alpha, beta and gamma tests and some others have just been dreamed up (such as our data dictionary project.)

To widen the scope of our activities, we would appreciate hearing from those researchers, developers, users and friends of Data-Base Management technology who share our enthusiasm. We will carefully examine and consider all suggestions and criticisms as well as any proposals for cooperation.

## ACKNOWLEDGMENTS

We wish to express our appreciation to all our associates and friends who, directly or indirectly, made our software projects possible. In particular, we are indebted to: Jon Bale, Gabriel Buzzetti, Enrique Castillo, Alex Dengo, Pablo Gutierrez, Lissa Hanckel, Max Holzheu, Jerry Johnson, Einar Klanderud, Orly Larson, José Mirón, Félix Montes, Manuel Ponciano, Clara María Ramírez, Patricia Springmühl de Rego, Simón Sibony, Sergio Tenenbaum, John R. Trudeau, Fred White, René Woc.

# Faster with FAST KSAM

by
Stephen M. Butler
Director of Data Processing
Paradise Valley Hospital
National City, CA

## BACKGROUND

Being a hospital, we maintained a manual index to patient numbers. The mechanical device was overloaded (280,000 in a 250,000 capacity Diebold), and the repair bills were worse.

During the analysis of the HP-3000 a prototype computerized Medical Index using IMAGE was written; but ALPHA or NUMERIC sequence searches could not be used--who wants a sorted chain of 280,000+ entries. A phonetic search was implemented but the chain had cases of 2000+ entries. A generic key capability for the phonetic-birthdate search mechanism was needed.

The prototype index gave us the impetus to acquire the 3000. As the conversion from a Honeywell 115 neared completion, KSAM became available. It had the capabilities needed:

1. Multiple keyed ISAM.

2. Generic keys.

We claim to be the GAMMA test site for KSAM--if such exists. It seems there were updates every other week--at least we saw our SE! Finally there was a version of KSAM clean enough to attempt a load of the 280,000+ records. It took 5 days for the disk drives to survive the shake out test; but one of the other systems started acting funny. Soon we were in the GAMMA test phase again. After several attempts to fix the bug, the SE took our test program and disappeared. (He claimed he wouldn't come back until there was a version of KSAM that would work on his machine!)

The next week he laid the update tape on our desk; we had become past masters of doing KSAM updates. The SE's parting comment was, "That 5 day load should be faster." "Did we get FAST KSAM?" "Can't say; but don't tell anybody else."

The reload took 2-1/2 days. Not the 1000% improvement expected, and there were more bugs. So, we wrangled a day at the lab to find out why a particular bug had so many facets and why it was taking upwards of a month to fix a problem we felt was critical. A lot of information was passed in both directions, and the lab thanked us for being a BETA test site. Wished somebody had told us sooner!

## Faster with FAST KSAM

The new version of KSAM was quoted to be the pre-release version that would follow the MIT following 1814. That was Feb. 9, 1978. It passed all our tests and is now on the 1814 M.I.I.

Following tips from the lab, the load took 20 hours. Not bad for 280,000 records. In retrospect we are happy to have picked KSAM.

### KEYFILE

An understanding of the KEYFILE will help in knowing why the following tips work. A close reading of the new Appendix B in the KSAM manual will be useful.

First, the KEYFILE record size is one sector (128 words; 256 bytes).

The CONTROL block is described in FIGURE 1. The number of keys per record is the main item of interest.

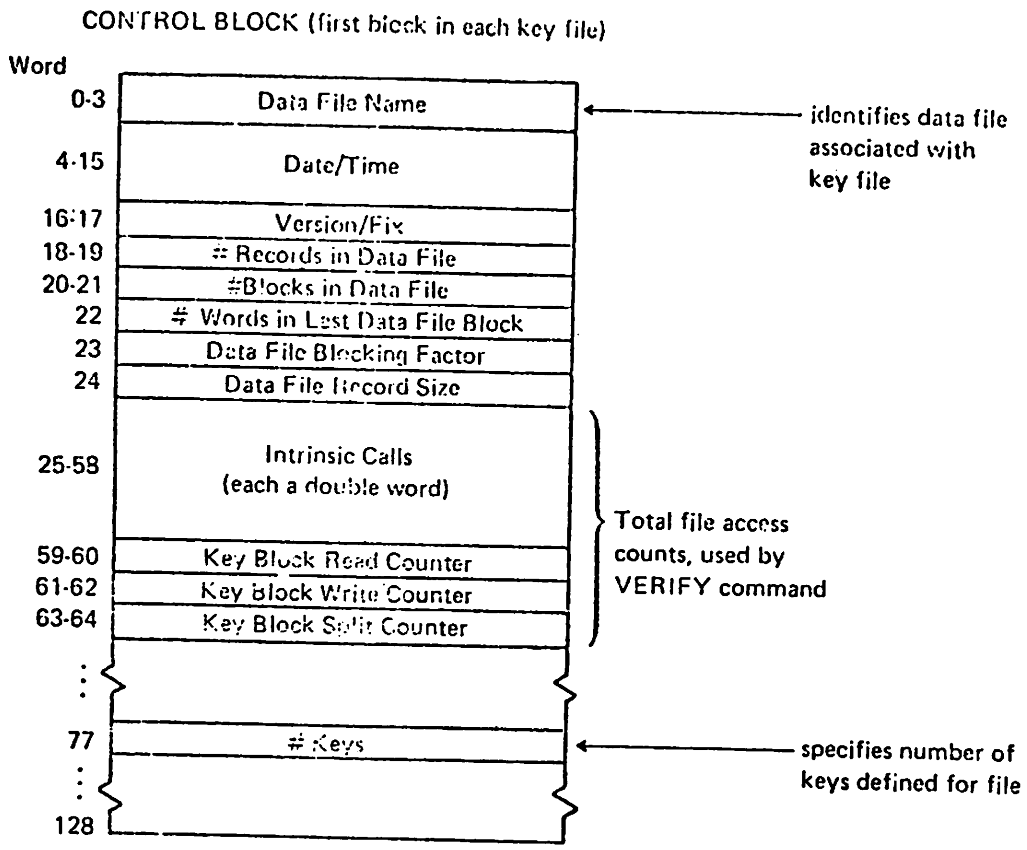

CONTROL BLOCK (first block in each key file)
Word
0-3
Data File Name
identifies data file associated with key file
4-15
Date/Time
16-17
Version/Fix
18-19
# Records in Data File
20-21
#Blocks in Data File
22
# Words in Last Data File Block
23
Data File Blocking Factor
24
Data File Record Size
25-58
Intrinsic Calls (each a double word)
Total file access counts, used by VERIFY command
59-60
Key Block Read Counter
61-62
Key Block Write Counter
63-64
Key Block Split Counter
77
# Keys
specifies number of keys defined for file
128

FIGURE 1. CONTROL BLOCK layout. Note word 77.

The KEY DESCRIPTOR block has one entry of 8 words for each defined key. The detailed layout is in FIGURE 2. The most useful items right now are the pointers to the ROOT KEY ENTRY block for each of the defined keys.

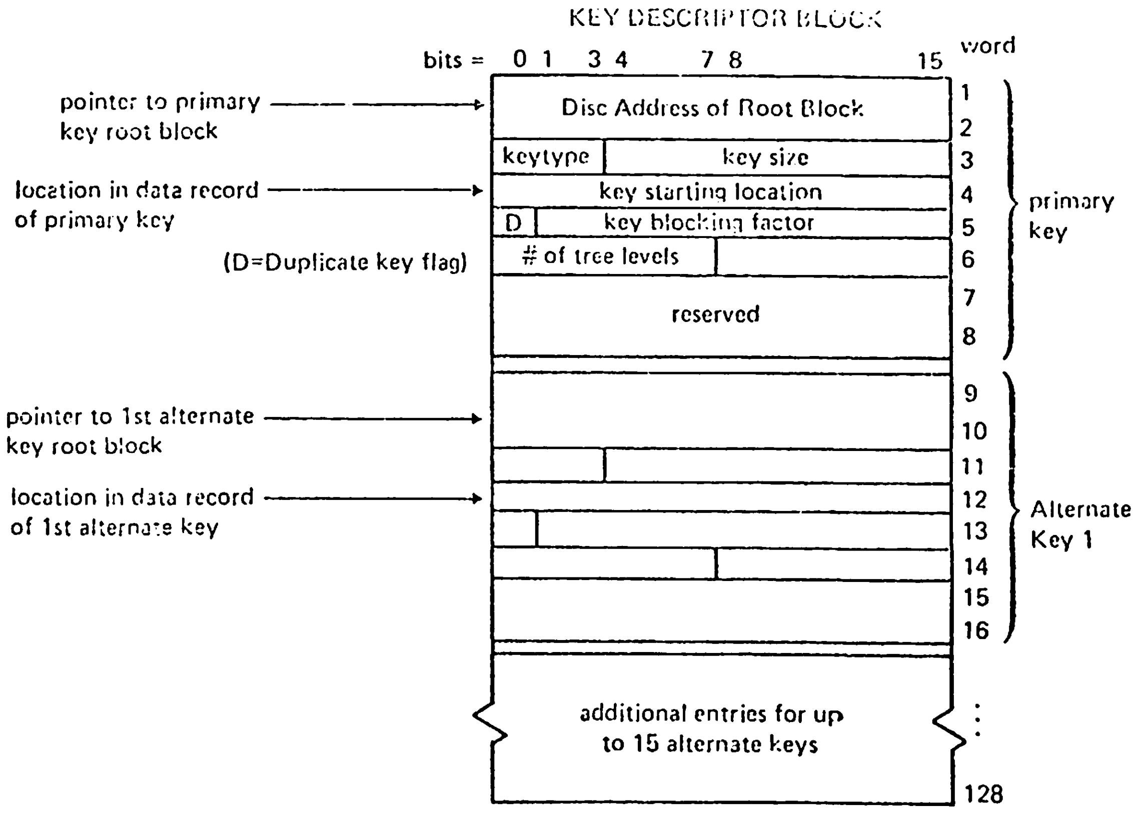

KEY DESCRIPTOR BLOCK
bits = 0 1 3 4 7 8 15
word
pointer to primary key root block
location in data record of primary key
(D=Duplicate key flag)
Disc Address of Root Block
keytype
key size
key starting location
D
key blocking factor
# of tree levels
reserved
primary key
pointer to 1st alternate key root block
location in data record of 1st alternate key
Alternate Key 1
additional entries for up to 15 alternate keys
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
128

FIGURE 2. KEY DESCRIPTOR BLOCK. Each entry consists of 8 words. The RESERVED area is a pointer to the free list chain for this key.

The KEY ENTRY blocks contain the key values and pointers used to make KSAM do its thing. A quick look at FIGURE 3 will show that a key entry is composed of:

1. Double-word relative record number of the KEY ENTRY block that sequentially comes before this entry.

2. KEY value as it is in the Data Record. A slack byte is at the end if the key length is odd. This slack byte IS NOT initialized.

3. Double-word relative record number of the data record in the data file.

4. Double-word relative record number of the KEY ENTRY block that sequentially comes after this entry.

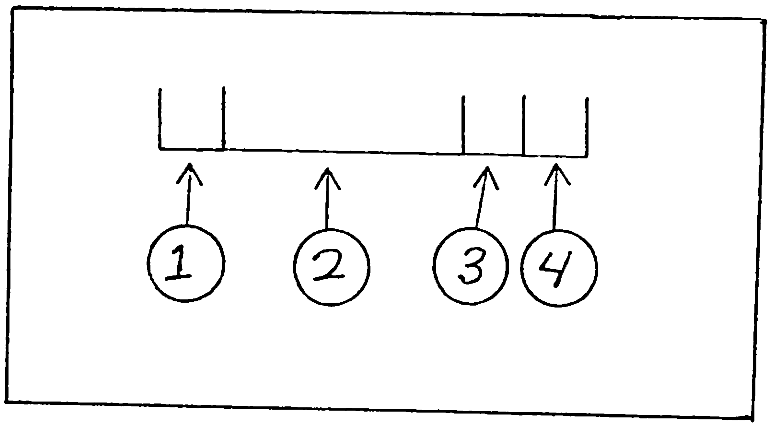


FIGURE 3. KEY ENTRY. See text for description of numbered items.

With two or more keys item 4 becomes item 1 for the next key in that KEY ENTRY block, see Figure 4. Thus there is 1 more KEY ENTRY pointer than the number of active keys in that block. Since each KEY ENTRY has a data pointer also, the number of double word pointers can be written as

$$2N + 1$$

where N is the number of keys per KEY ENTRY block.

Each KEY ENTRY block starts out with a double-word integer whose value is the relative record number of that block. The next word has a count of the number of active keys in this KEY ENTRY block. Subsequent records within the same KEY ENTRY block do not have this information. The key value within a KEY ENTRY block can be split across the physical blocks of the KEYFILE. Using FCOPY to dump the KEYFILE with ';NOKSAM; OCTAL;CHAR' options will allow a person to inspect the actual layout for a particular file. Thus a person could simulate conditions that would normally be hidden deep inside a large file. Once you know the general layout you will quickly pick up the specific pieces of information needed to navigate through the KEYFILE.

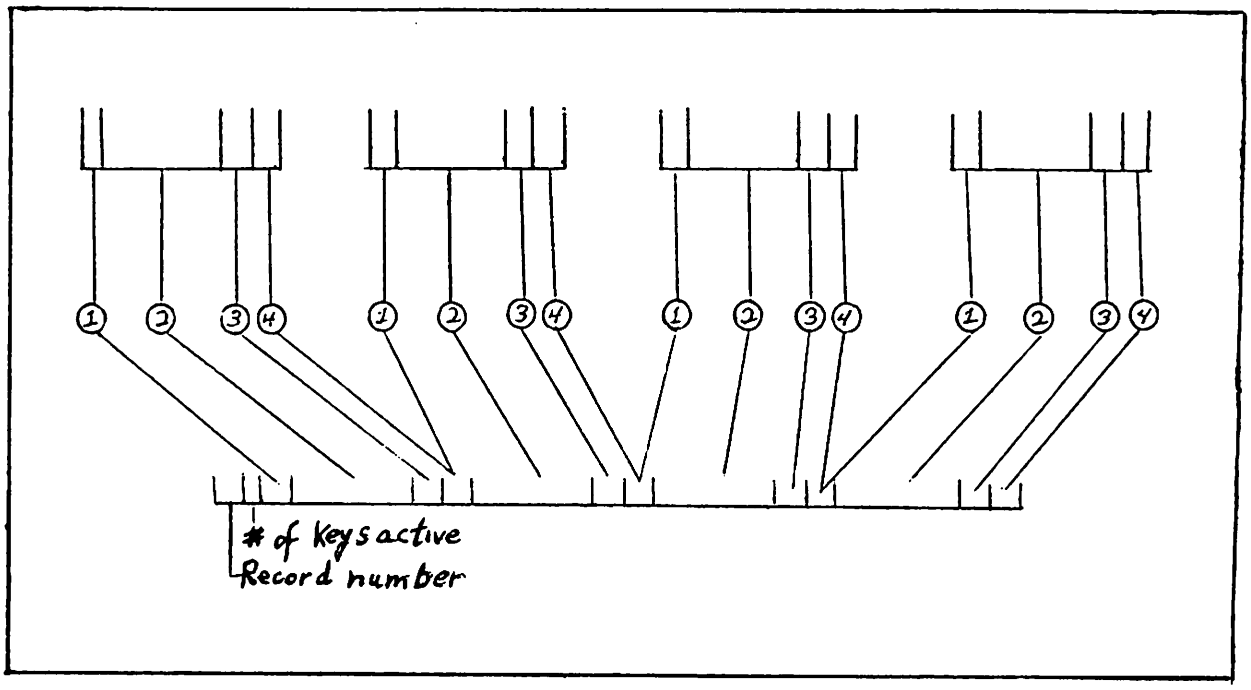


FIGURE 4. KEY ENTRY BLOCK. Words 1 & 2 contain the relative record number for this block. Word 3 contains the number of active keys. Notice how items 4 and 1 are shared by adjacent KEY ENTRYs.

We have spent upwards of two days at a time sifting through a KEYFILE in this manner in order to pin point KSAM bugs. On one occasion a bug seemed to occur only on our 280,000 record KSAM file. My boss bet a milksnake that I couldn't simulate it with less than 100 records. I did it with 8--but I first knew exactly what was happening.

FIGURE 5 can be followed to calculate the blocking factor (BF) for each key. A specified BF is used as a minimum and is adjusted upwards to make full use of any remaining area in the last sector. The default BF is chosen so that the KEY ENTRY block will span 8 sectors--1024 words (2048 bytes). If the KEY ENTRY block spans more than 16 sectors (2048 words or 4096 bytes), the BF is reduced so a maximum of 16 sectors is used.

With multiple keys the largest KEY ENTRY block size is used to calculate the BF for all keys. Thus all KEY ENTRY blocks occupy the same number of sectors. This along with the 16 sector maximum are by-products of requirements for using the KSAM extra data segment.

| | |
|---|---|
| KS = | key size in bytes |
| ES = | key entry size in words |
| BF = | blocking factor (number of key entries per block) |
| BS = | key block size |
| FL = | data file limit in records |
| NB = | number of sectors per key block |
| FS = | key file size in sectors |
| ⌈⌉ = round up | ⌊⌋ = round down |

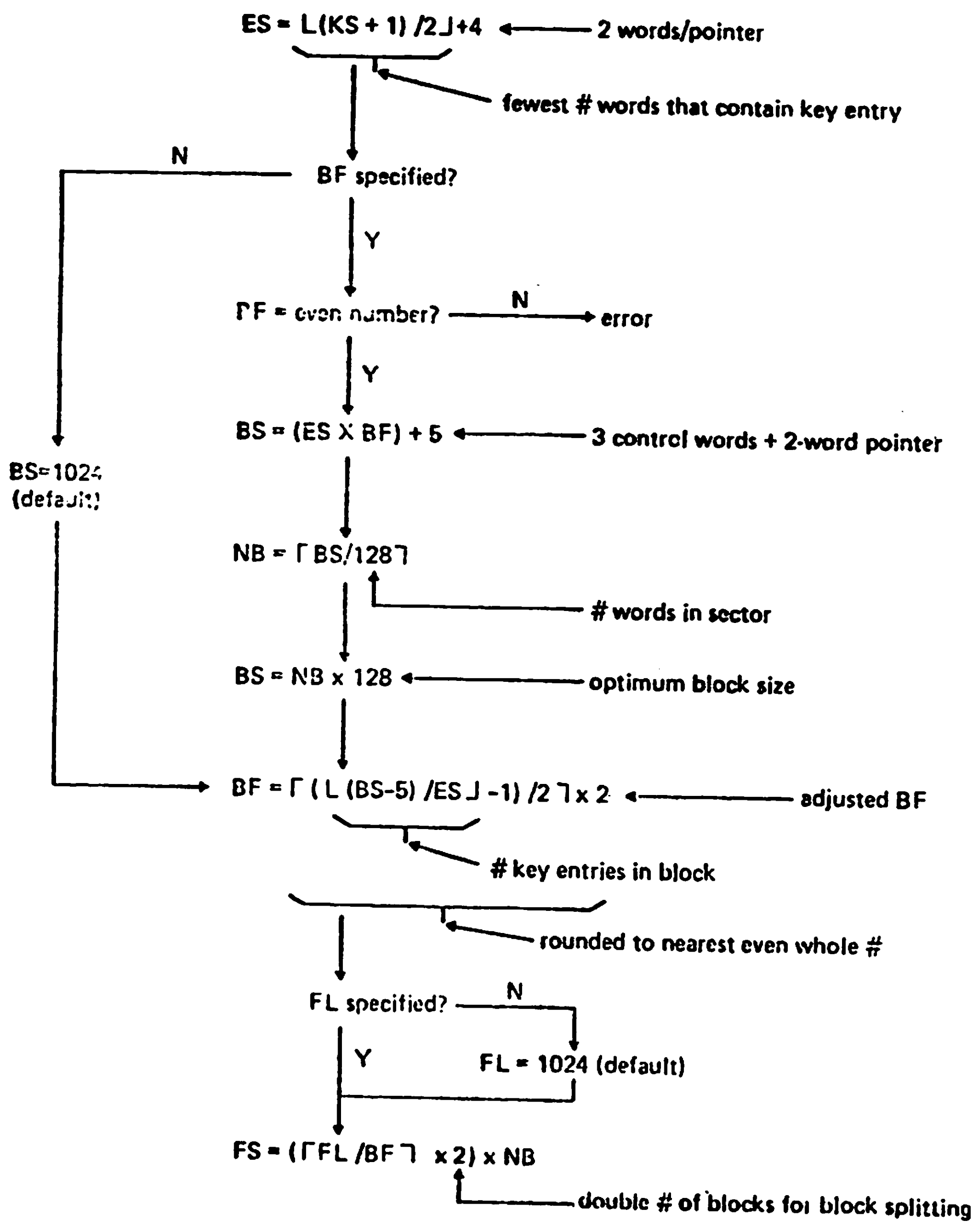


FIGURE 5. Calculating blocking factor (BF) and file size (FS) for one key.

Assume a file with 2 keys defined as:

KEY = B,1,53,12
KEY = B,54,13,20

**For Key 1:**

KS=53
FL=1024 (default)
BF=12

Calculation of FS:

$ES=\lfloor (53+1)/2 \rfloor +4 = 27+4 = 31$

$BS=(31\times 12)+5 = 377$

$NB=\lceil 377/128 \rceil = \lceil 2.9 \rceil =$ 3 sectors

$BS= 3\times 128 = 384$

$^{*}BF=\lceil (\lfloor (384-5)/31 \rfloor -1)/2 \rceil \times 2$
$= \lceil (\lfloor 12.2 \rfloor -1)/2 \rceil \times 2$
$= \lceil (12-1)/2 \rceil \times 2$
$= \lceil 5.5 \rceil \times 2 = 6\times 2 = 12$

$FS=(\lceil 1024/12 \rceil \times 2) \times 3$
$=(\lceil 85.3 \rceil \times 2)\times 3$
= 516 sectors

**For Key 2:**

KS=13
FL=1024 (default)
BF=20

$ES=\lfloor (13+1)/2 \rfloor + 4 = 7+4 = 11$

$BS= (11\times 20)+5 = 225$

$NB=\lceil 225/128 \rceil = \lceil 1.75 \rceil =$ 2 sectors

$BS= 2\times 128 = 256$

$^{*}BF=\lceil (\lfloor (256-5)/11 \rfloor -1)/2 \rceil \times 2$
$= \lceil (\lfloor 22.8 \rfloor -1)/2 \rceil \times 2$
$= \lceil (22-1)/2 \rceil \times 2$
$= \lceil 10.5 \rceil \times 2 = 11\times 2 = 22$

$FS=(\lceil 1024/22 \rceil \times 2) \times 2$
$=(\lceil 46.5 \rceil \times 2) \times 2$
= 188 sectors

Since key 1 has the largest block size (384 words in 3 sectors), its blocking factor is unchanged. The blocking factor for key 2 is adjusted so it has the same block size. The following values are used:

ES=11 ← entry size calculated for key 2
BS=384 ← block size of key 1 (now used for key 2, also)
FL=1024 ← default file size in words
NB=3 ← number of sectors needed for each block of 384 words

Calculate the new blocking factor for key 2:

$^{*}BF=\lceil (\lfloor 384-5/11 \rfloor -1)/2 \rceil \times 2$
$= \lceil (\lfloor 34.4 \rfloor -1)/2 \rceil \times 2$
$= \lceil 16.5 \rceil \times 2 = 17\times 2 = 34$

$FS=(\lceil 1024/34 \rceil \times 2) \times 3$
$=(\lceil 30.1 \rceil \times 2) \times 3 =$ 186 sectors

Summing the two file sizes and adding two sectors for control and key descriptor information, the total file size in sectors is:

516 + 186 + 2 = 704 sectors

*The algorithm to calculate BF can be expressed more simply if the result can be checked for an even number:

$BF=\lfloor BS-5/ES \rfloor$ If BF is an odd number, set BF=BF-1

FIGURE 6. Calculating file size (FS) for multiple keys.

FIGURE 6 shows how the size of the KEYFILE is calculated. Since each block can be a minimun of half full, twice as many KEY ENTRY blocks are assigned as would be needed if each block were full.

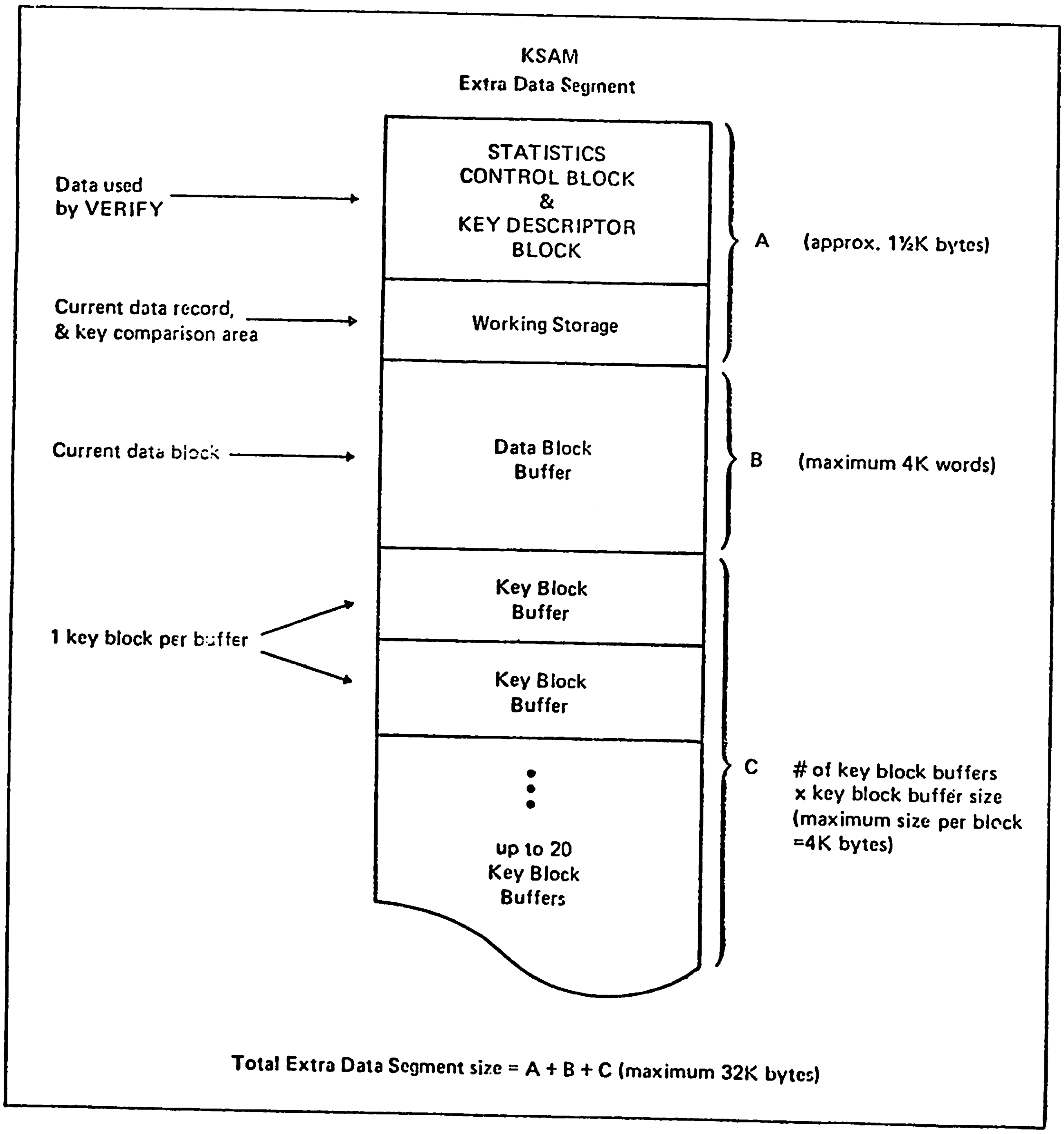

KSAM
Extra Data Segment
Data used by VERIFY
STATISTICS CONTROL BLOCK & KEY DESCRIPTOR BLOCK
Current data record, & key comparison area
Working Storage
A (approx. 1½K bytes)
Current data block
Data Block Buffer
B (maximum 4K words)
1 key block per buffer
Key Block Buffer
Key Block Buffer
up to 20 Key Block Buffers
C # of key block buffers x key block buffer size (maximum size per block =4K bytes)
Total Extra Data Segment size = A + B + C (maximum 32K bytes)

FIGURE 7. KSAM XDS.

## KEY BUFFERS

The new features of FAST KSAM can now be put to use. An extra data segment (XDS) is used to handle all I/O to the KSAM file. The size of this XDS is limited to 16K words. Approxiametly 1-1/2K are used for overhead and control information. Only one buffer is used for the DATAFILE; it has a maximum of 4K words and is the size of one block from the DATAFILE. We use a program that calculates the best BF that will fit in 8 or fewer sectors so the data buffer will be 1K or less. The rest of the XDS can be used for KEY ENTRY buffers.

To find how many buffers could be used (all calculations in words):

1. Subtract the 1-1/2K of overhead.

2. Subtract the size of the data buffer. Lets assume 1K.

3. Divide what's left by the size of a KEY ENTRY block--default is 1K.

4. Round the answer down to the next integer.

   So, (16K - 1-1/2K - 1K) / 1K = 13 buffers.

To get them:

```
:FILE ksamfile;DEV=,,13
```

If this would cause the XDS to be larger than 16K words, KSAM will automatically decrease the number of buffers.

Since KSAM does have a fairly good algorithm for choosing the default number of key buffers (see FIGURE 8), once the file has stablized you may wish to restrict the use of the FILE equation to loading or otherwise making large numbers of changes to the file. If the file is empty, KSAM will default to the minimal number of buffers for the type of open specified. For this reason you should specify the number of buffers you will actually need as KSAM will not allocate more buffers as the file is filled.

Each process that opens the KSAM file gets its own XDS. The number of buffers in these XDSs are dependent upon the type of open specified and the number of keys in the file at the time of opening. Therefore, these XDSs could have differing numbers of key buffers.

| Access Type | Buffers Assigned |
|---|---|
| Read Only Access | 1 buffer per level in *primary* key structure |
| Write Only Access | 3 buffers per primary key + 3 buffers per alternate key + 3 buffers |
| Other Access (Read/Write or Update) | 1 buffer per level in primary key structure + 1 buffer per level in alternate key structure + 3 buffers |
| | (up to a maximum of 20 buffers) |

FIGURE 8. Default key buffers allocated at FOPEN.

## DUPLICATES

The other ISAM packages that I am familiar with do not allow for duplicate keys. At first glance, one would think it is a blessing that KSAM does; but to paraphrase the LAB: If there are more than 10 duplicates for a particular key, then don't have this key or make it unique.

Whenever a key is added to the file it is added after any duplicates that exist for that value. KSAM must search the KEYFILE to find that last entry. A START causes a search for the first entry.

Two of the most common ways of making duplicates unique are:

1. Put a time stamp (HR:MM:SS) after each key. For calls less than 1 sec. apart, this would still leave them duplicates.

2. Put a copy of the primary key after the other keys.

In COBOL the primary key must be unique. In the Medical Index case it was 7 bytes long so we were not any worse off than using the time stamp. Another method will be proposed in the ENHANCEMENT section.

## TIMINGS

A stand-alone environment is not readily available on our system. The following timings show both CPU seconds and WALL TIME to load 10,000 records into an empty KSAM file.

| | Default Buffers | 13 Buffers |
|---|---|---|
| KSAM expected duplicate keys and duplicate keys were loaded. | 1111/5333 | 533/1010 |
| KSAM expected duplicate keys; but all keys loaded were unique. | 870/4627 | 326/484 |
| KSAM expected unique keys and unique keys were loaded. | 1183/5992 | 389/567 |

FIGURE 9. This shows the CPU seconds/WALL seconds to load 10,000 records into an empty KSAM file. Three keys were used--7 bytes, 20 bytes, and 43 bytes. The BASIC procedures were used to load the file.

## ONLINE BENEFITS

By correctly specifying the number of key buffers and utilizing unique keys there will be a marked improvement in throughput. But the other benefits even outweigh this.

An example please: two users will access a KSAM file that has 4 records in it. We will assume 1 defined key and a KEY ENTRY blocking factor of 4. Therefore, the ROOT KEY ENTRY block is full. Any new records added to the file will cause a key-block-split. We proceed:

1. Both users open the file for shared access.

2. User A LOCKs the file and reads the first two logical records.

3. User A UNLOCKs the file and User B LOCKs it.

4. User B writes a record whose logical value places it #3.

5. User B UNLOCKs the file and posts the updated buffers.

6. The file now has 1 KEY ENTRY in the ROOT block. This points to two other blocks. The first block contains the keys User A just read. The second block contains the two keys User A expects to see. In actual practice he should get the record that User B just posted.

7. User A LOCKs the file again and calls for the next READ (sequential of course).

8. The next KEY ENTRY that User A would previously have used would have been #3 in the ROOT. At least that is all that KSAM knows. But the ROOT now has only one entry. Since the 3rd entry no longer exists we are at the end of the file, so return an EOF condition.

User A was lucky. If there had been many KEY ENTRY blocks and User A had been down several levels, the following possibilities could have happened (we have seen results to indicate they have happened to us):

1. The current block would no longer be included in the key structure; but the process is not aware that is has been placed in a free buffer list, so the process uses it.

2. The same for a previous level; ie, the ROOT or one of the intermediate levels was moved away from where we expected it after the last access.

3. The current or a higher level was reshuffled. All blocks are active; but not necessarily in the same tree structure as before.

We turned this in as a bug--and promptly got laughed at. This is one of those dubious features we all enjoy. KSAM will not keep track of any reorganization that may occur while the file is unlocked. The buffers are refreshed by the physical blocks that were last used in the XDS. KSAM will not check to be sure that these contain the logical values last used. So, you must reposition the pointer yourself. You can do that by using the START procedure with a relop of strictly greater than the key that was returned in the last read even though a number of changes may have occurred to the point of deleting the record last read. This is a multi-user online environment, right! Again:

Faster with FAST KSAM

1. LOCK the file.

2. Do a START using last key read and greater than relation.

3. Now do that sequential READ.

4. If you were going to do a READBYKEY or a REWRITE in random/dynamic mode, then items 2 & 3 are not needed.

5. UNLOCK when done.

That process can't be done with duplicate keys. If the last READ was in the middle of a duplicate key, the START would bypass the duplicates not read. Unique keys are a MUST in order to do the above.

In the case of updates to the file, one more item is needed--RECORD LOCKING. Set aside one byte in the record to be a locked/unlocked flag. When a record is read prior to updating it:

1. Check that field--if it is locked, then report it as being locked or work out a mechanism to hang until it frees up. We don't like hanging up, since a process might abort and leave a record flagged as locked. It hasn't happened in 6 months at our site, but? If unlocked, then continue.

2. Set the flag for lock.

3. REWRITE the record.

Now you can UNLOCK the file and KNOW that when you're ready to update that record it WILL BE the SAME. P.S. Be sure to reset that flag!

## PROPOSED_ENHANCEMENTS

If we find time between this writing and the International meeting, we may have a set of COBOL copylibs to simulate this. We want KSAM to do a lot of the dirty work for us, so:

1. It should automatically call the LOCK for us if we failed to. Of course, only we know when to UNLOCK, so this is only a onesided benefit. It would still be useful.

2. The first call following a LOCK (whether directly or by #1) should cause a call to the START to reposition the pointer, unless this is a READBYKEY or START or REWRITE in random/dynamic mode.

3. Force all keys to be unique by:
   A. Assigning a double-word integer as the primary key.
   B. Appending this integer (4 bytes) to the end of every key. (Of course, if one key ends at the place the primary key takes off, then just increase the length of that key.)

4. Allow record locking (if necessary, set aside one byte) ie, a read with lock option.

If everything else in KSAM worked the same, we could then define a next step that would reuse space left by deleted records.

1. The record with a primary key of zero should hold two pointers:
   A. The next integer to be used for the primary key (ie, EOF pointer).
   B. The primary key value of the most recent record deleted.

2. The alternate keys for primary key of zero and all deleted records should be set to HIGH-VALUES except for the last 4 bytes which would be a copy of the primary key.

3. An EOF pointer would be returned upon reading a record with HIGH-VALUES in all bytes except the last 4 of any key.

4. Every deleted record would have the primary key value of the previously deleted record. (A push down stack or free list chain.)

5. Whenever a REWRITE occurs it would be keyed off of the primary key.

6. A WRITE would first use up the free list chain before incrementing the primary key.

You will notice the above has been specified in such a manner that a user of the current KSAM could write a set of procedures to make KSAM function as suggested. Now for the bombshell. KSAM already appends a double-word integer to <u>ALL</u> keys. It is more properly called the data record pointer.

If the lab would do the above, they could do it at a more simple level (ie, they wouldn't need to use the primary key).

1. They could use the EOF pointer as now.

2. They would need to set up a free list chain for the DATAFILE (they already have one for the KEYFILE).

3. They would have to keep track of the key and record number of the last logical record read. Then automatically reposition according to the first set of proposed enhancements.

4. They already append the record pointer to the keys so no physical change would be necessary--as would be if one of us users was to try.

5. In short, the lab has all the information necessary to do the job except for the free chain list in the DATAFILE.

Whether or not the lab does this enhancement, we already do something similar by using the primary key to append to all others; but we intend to write the procedures necessary to make KSAM look like our proposal. Anybody interested?

There are some enhancements that only the lab can do:

1. A central XDS similar to the recent IMAGE update. Only one XDS per file no matter how many users are using that same file. A small XDS may be needed for each process to keep track of the last logical key value and other local data.

2. Implement a LOCKing similar to IMAGEs.

3. OK, let's go for it! USE THE IMAGE CALLS TO HANDLE KSAM. This means:
   A. A schema processor to look for data sets type K or KSAM.
   B. We could have FAST sorted chains.
   C. In fact we could now have sorted Master files (just a KEYFILE to point to the Master set entries).
   D. DBFIND would also function as START for KSAM files.
   E. DBGET would take over as:
      - the current calculated DBGET would work for READBYKEY.
      - a new mode for sequential reads as opposed to serial.
   F. The DBLOCK would give the same type of locking scheme for KSAM as is now being enhanced for IMAGE.

These ideas will be implemented in our shop as far as possible. We plan to write a set of routines that will handle both the KSAM and IMAGE procedures. If the lab beats us, we shall be very happy to conceed the race!

# INFORMATION MANAGEMENT: AN INVESTMENT FOR THE FUTURE

DAVID C. DUMMER

D.C. DUMMER & ASSOCIATES LIMITED

We are all witnesses to the current information explosion that affects every aspect of our lives. Some of us may wonder if this explosion can be contained and controlled.

Computer technology has nurtured the evolution of devices that perform data storage and manipulation functions at reasonable cost. Government, industry and commerce have rapidly made use of such devices in an effort to improve information systems for decision-making processes. The better the quality and timeliness of information, the more powerful and competitive the user can become.

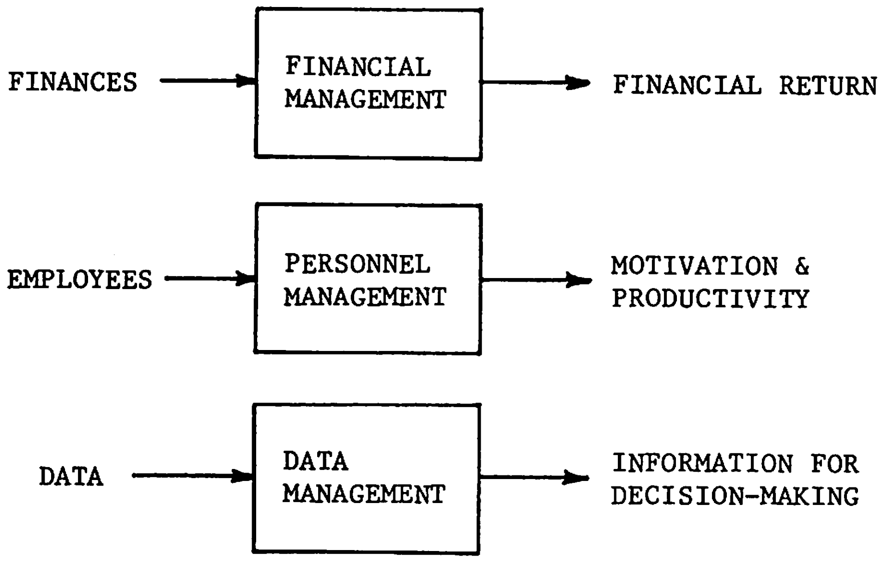
Unfortunately, the technologies that support the effective utilization of information systems have trailed the dramatic advances in computer hardware and software. There is still not a general awareness that data is an organizational resource that requires management control, administration and involvement. The first illustration draws an analogy to other resource management areas in that good data management will directly benefit information systems needed for decision-making. Data management is, however, a far less developed area than either financial or personnel management. Very few organizations that have attempted to establish a corporate data base resource have been completely successful. The history of integrated management information systems contains many failures, and indeed, the downfall of several organizations.

Like many technological advances, those related to information handling are full of promise, yet also hide many dangers. Failures can generally be attributed to two major causes: the incomplete and incorrect application of the technologies and resulting information handling facilities; and the lack of necessary changes to organizational structure to complement the integrated information structure.

It is relatively easy for senior executives to decide upon a data resource management strategy, but it is a far different matter to understand all of the different components necessary for strategy success. These problems are compounded by the fact that there are very few professionals in this area with adequate levels of experience.

The second illustration highlights the resistance that corporate management typically meets in the introduction of data base technology. Re-

RESOURCE MANAGEMENT



## INTRODUCING DATA BASE TECHNOLOGY TO AN ORGANIZATION

| RESISTANCE TO: | CORPORATE OPPORTUNITIES: |
|---|---|
| - CHANGE IN METHODS AND PROCEDURES | - IMPROVED METHODS AND PROCEDURES |
| - LOSS OF DATA OWNERSHIP | - CORPORATE DATA MANAGEMENT |
| - LOSS OF DATA CONTROL | - CORPORATE CONTROL STANDARDS |
| - CHANGE IN THE POWER STRUCTURE | - PROFESSIONAL DATA MANAGEMENT |
| - CHANGE IN THE STATUS OF DP USERS | - IMPROVED DATA UTILIZATION |
| - CHANGE IN STAFF REQUIREMENTS | - REDUCED COST AND ABILITY TO CONTROL |

sistance can occur in both data processing and user departments as the need for new responsibilities and procedures becomes evident. Few executives are equipped with all of the correct rebuttals to the criticisms that result from the resistance. So intense can the resistance become that often the data base approach is introduced in a compromised manner and one that is far from beneficial to the organization.

Corporate opportunities that can be realized by the data base approach are immeasurable and there is an ever increasing responsibility on the data processing profession to ensure that the approach is fully understood and supported. The second illustration also enumerates the respective benefits that can accrue to the organization, but, in order to achieve these benefits, the organization must be willing to invest the necessary developmental resources into the data base approach.

Once the data base approach has been adopted as a means to achieve data resource management, it is important to realize that a data sharing concept has been introduced within the organization. That is, the common data in the corporate data base is to be shared by all those in the organization that have a right and need to access the data. The major technical facility that supports data sharing is a data base management system (dbms). A dbms is often presented as the solution to the problem of data sharing, but, in reality, it is just one of the facilities needed to achieve data sharing in a resource management environment. Crucial to the success of data sharing is data administration, sometimes referred to as data base administration. Data administration encompasses the other facilities, procedures and tools needed to manage the data base. The third illustration shows the major aspects of data administration. The degree to which an organization addresses and implements facilities in these areas depends on the complexity, integration and value of the data together with limitations imposed by the budget available for data management.

First and foremost of the required administration facilities is a data dictionary and directory (DD&D). The DD&D is essentialy an information system about the data and data processing systems used in the organization. To the person or persons responsible for data administration, it represents a tool to document and control the data base facility. If the data base driven information systems are an integral part of the operations of the organization, then the data base will normally have to be flexible and changeable, to reflect and support business dynamics. The DD&D should be designed and organized to provide for this type of environment.

For some organizations, the DD&D will evolve into the hub, or nervecenter, of their data processing facilities. It will control, monitor and service the corporate data base together with the associated information systems.

DATA SHARING REQUIREMENTS

$$\begin{tikzcd}
\text{DATA SHARING} \arrow[r] \arrow[d] & \text{DATA BASE MANAGEMENT SYSTEM} \\
\text{DATA ADMINISTRATION} \arrow[r] & \text{- DATA DICTIONARY \& DIRECTORY}
\end{tikzcd}$$

- BACKUP & RECOVERY
- SECURITY CONSIDERATIONS
- MAINTENANCE CONSIDERATIONS
- EFFICIENCY CONSIDERATIONS
- AUDITING CONSIDERATIONS
- USAGE STATISTICS

There are other data administration facilities which complement the DD&D, many of which are still in the evolutionary stage. These facilities are aimed at such considerations as ensuring that the corporate data base is always organized in the most efficient manner, and is normally available, and is recoverable in the event of system failures.

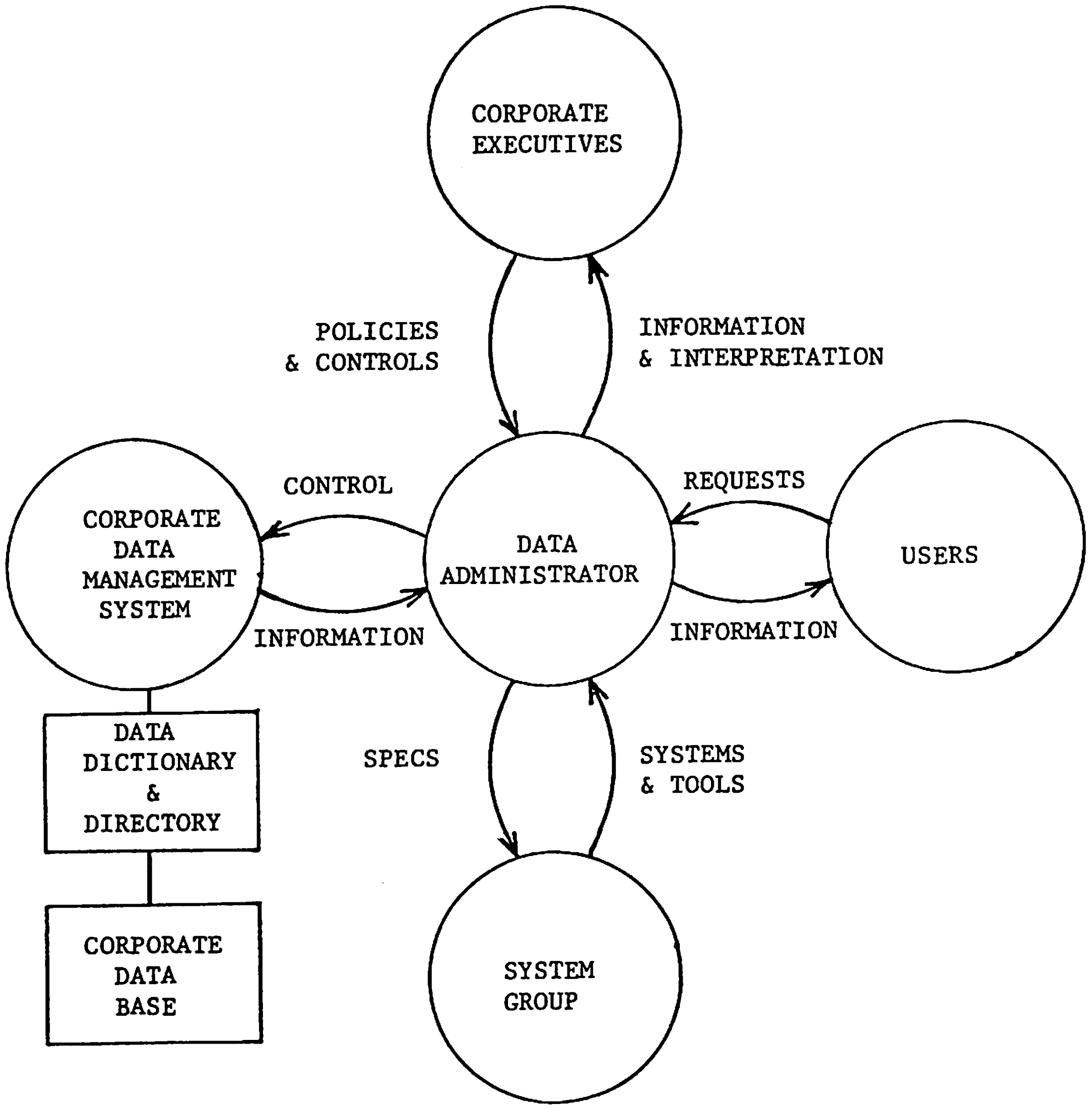
Another important aspect of information systems concerns auditability and performance measurement. These are typically topics that are considered only at system implementation, but a data administration function can ensure that system audit becomes a design parameter during analysis and development.

The fourth illustration shows the responsibilities of the person (Data Administrator) or persons performing the data administration function. Like other resource managers, the Data Administrator (DA) must be placed in an organizational structure such that he or she can coordinate and be held responsible for all of the technical and administrative components needed to effect data sharing. The DA is responsible to the corporate executives to inform them of requirements and situations which demand their participation and decision-making; and then to enact and administer the resulting policies and data control measures. The DA is responsible to data processing users in the response to queries and requests and in the provision of facilities that make data more accessible to those with the right to access it. The DA interfaces with the systems group in order to obtain the hardware and software which is required to provide and support the data base and processing environment for which he or she is responsible. The final interface within the organization is to the corporate data management system which embodies all of the facilities needed to provide control and administration of the corporate data base.

With a perception of the data administration requirements, the specification of a suitable DD&D can be detailed. The fifth illustration lists the major components of a DD&D. Again, the complexity and contents of the DD&D should match the requirements and budget of the organization. The components cover the documentation of how data is perceived in the organization (documents, forms, used by department, etc.); how data is structured in the data base designs; and how data is used in the data processing systems. Supporting these directories are dictionary and directory entries which document data items and their attributes, data validation rules, data security measures and data item synonyms and associates.

The sixth illustration depicts the role of the DD&D in an organization. It is the method by which the DA documents, controls and administers the corporate data base and information systems. It is a source of information and a design capture device for data base and information system designers. It is a source of information and a change capture

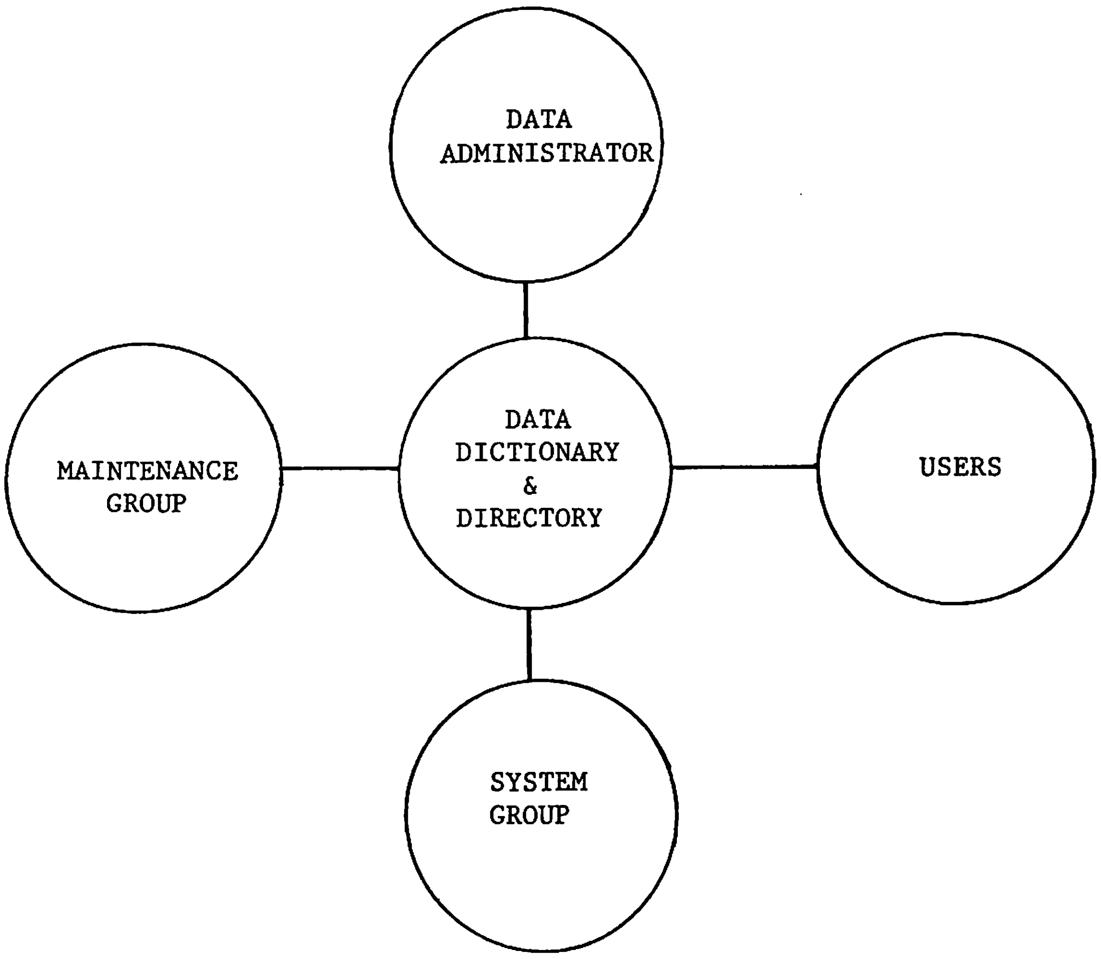
ROLE OF THE DATA ADMINISTRATOR



## DATA DICTIONARY & DIRECTORY CONTENTS

- DICTIONARY & DIRECTORY OF THE NATURAL DATA ITEMS AND DATA ITEM GROUPINGS IN AN ORGANIZATION
- DICTIONARY & DIRECTORY OF THE DATA STRUCTURES DESIGNED FOR THE CORPORATE DATA BASE(S)
- DICTIONARY OF DATA ITEM ATTRIBUTES
- DICTIONARY OF DATA ITEM VALIDATION RULES
- DIRECTORY OF DATA ITEM SECURITY MEASURES
- DIRECTORY OF DATA ITEM SYNONYMS AND ASSOCIATES
- DIRECTORY & CROSS-REFERENCE OF DATA ITEMS TO PROGRAMS AND TO DOCUMENTS

ROLE OF THE DATA DICTIONARY & DIRECTORY



device for the maintenance group, as existing data bases and information systems are modified and enhanced. It is a source of information to users who can discover what data is available without the need to contact the DA or associated staff. This is particularly true in the case of an online DD&D which can support information queries from the users.

One data administration topic that is currently receiving a lot of attention concerns data security. The seventh illustration contains a list of items needing protection consideration and a list of events that can constitute a threat to the data and system security. The first list identifies that security measures applied to data files and data bases are of little value if security measures are not applied to: the computer memory and storage devices during data processing; the hard copy data listings and reports distributed in the organization; the data transmission lines where remote terminals, workstations or satellite computers are involved; and the security measures themselves. In a similar manner to security provided by lock and key, data base and processing security is only a deterrent and each extra level of security will typically involve exponential increases in the cost and time of the security measure implementation and practice. These extra measures of security protection should be applied only where sensitivity of the information involved warrants it.

The second list covers the major threats to the security and availability of data. These are important considerations in a data sharing environment since the data has been organized in a logically or physically central location and it is collectively more vulnerable to security violation. One of the most important tasks for the DA in an organization is to achieve the correct balance between data accessibility and privacy.

The challenge that faces most organizations at this time is the effective creation of a data sharing environment. It requires an investment in terms of people, funds and organizational change; but the future benefits of a well managed data resource to aid and make possible decision-making are immeasurable.

The presentation of the paper will enlarge upon these topics and suggest various methods of evaluating and implementing data administration facilities and procedures.

## DATA SECURITY

| | |
|---|---|
| WHAT TO PROTECT? | - DATA ITEMS AND DATA FILES |
| | - COMPUTER MEMORY AND DATA STORAGE DEVICES |
| | - DATA LISTINGS AND INFORMATION REPORTS |
| | - DATA TRANSMISSION |
| | - SECURITY SYSTEM(S) AND PROCEDURE(S) |
| PROTECT FROM WHAT? | - HUMAN AND SYSTEM ERRORS |
| | - ENVIRONMENTAL ACCIDENTS OR CATASTROPHES |
| | - MISCHIEVIOUSNESS AND FRAUDULENCE |
| | - INDUSTRIAL AND POLITICAL ESPIONAGE |

# SIGMA - GENERALIZED INFORMATION SYSTEM

by Marcos A. X. de Carvalho
Systems Analyst
Promon Engenharia S.A.
Brazil

## TABLE OF CONTENTS

1. INTRODUCTION

2. SYSTEM DESCRIPTION

2.1 Data Structure

2.1.1 Structure Overview
2.1.2 Data Base Structure
2.1.3 Linkage among Data Bases

2.2 Data Base Handling

2.2.1 Data Base Updating
2.2.1 Information Retrieval

2.3 Reports

2.3.1 Definition
2.3.2 Selection
2.3.3 Edition

2.4 Additional Resources

2.4.1 Data Editing
2.4.2 Handling of Dates and Monetary Values

3. SYSTEM PERFORMANCE

## 1. *INTRODUCTION*

SIGMA (Generalized Information System) is a system for information handling, which provides the user with the following capabilities:

- . Batch and interative data entry
- . Selection of information through key-words and pre-defined coditions
- . Report generation defined by the user
- . Storage of data in more than one data base so that hyerarchical structures can be processed.

SIGMA was developed by the Systems Division of Promon Engenharia S.A. to assist Project Management in:

- . Project Control
- . Resources Control
- . Development of Interim-Systems
- . Budgeting
- . Information Control

## 2. *SYSTEM DESCRIPTION*

SIGMA is a system composed of a set of programs operating on a predefined data base structure under IMAGE. The information stored in the data bases is defined according to the user's application and it can be selected and edited in a report form by means of an oriented language.

In this section we will show:

- . Data Base Structure
- . Reports Available
- . Additional Resources

### 2.1 Data Base Structure

#### 2.1.1 Structure Overview

In the structure handled by SIGMA, the information can be grouped in more than one Data Base.

The data bases can be interconnected to form a hyerarchical structure.

To illustrate SIGMA structure, consider the following example:

. We are controlling a set of information about equipments specified in a project. This control begins with the specification of equipment by the engineering department up to its purchase by the purchasing department. Control is executed through four documents:

. Equipment List

. Requisition

. Inquiry

. Purchase Order

To each document there is associated a set of information as follows:

<u>Equipment List</u>

|  |  |
|---|---|
| .. Code | (Tag Number) |
| .. Description | (Technical Characteristics) |
| .. Cost | |
| .. Scheduled delivery date | |
| .. Importation flag | |
| $\boxed{\text{.. Requisition code}}$ | → *LINK INFORMATION* |

<u>Requisition</u>

.. Code

.. Scheduled issue date

.. Issue date

.. Date received by purchasing

.. Note

$\boxed{\begin{array}{l}\text{.. Purchase order code}\\ \text{.. Inquiry code}\end{array}}$ → *LINK INFORMATION*

<u>Inquiry</u>

.. Code

.. Scheduled issue date

.. Issue date

.. Supplier 1
2
3

.. Note

<u>Purchase Order</u>

.. Code

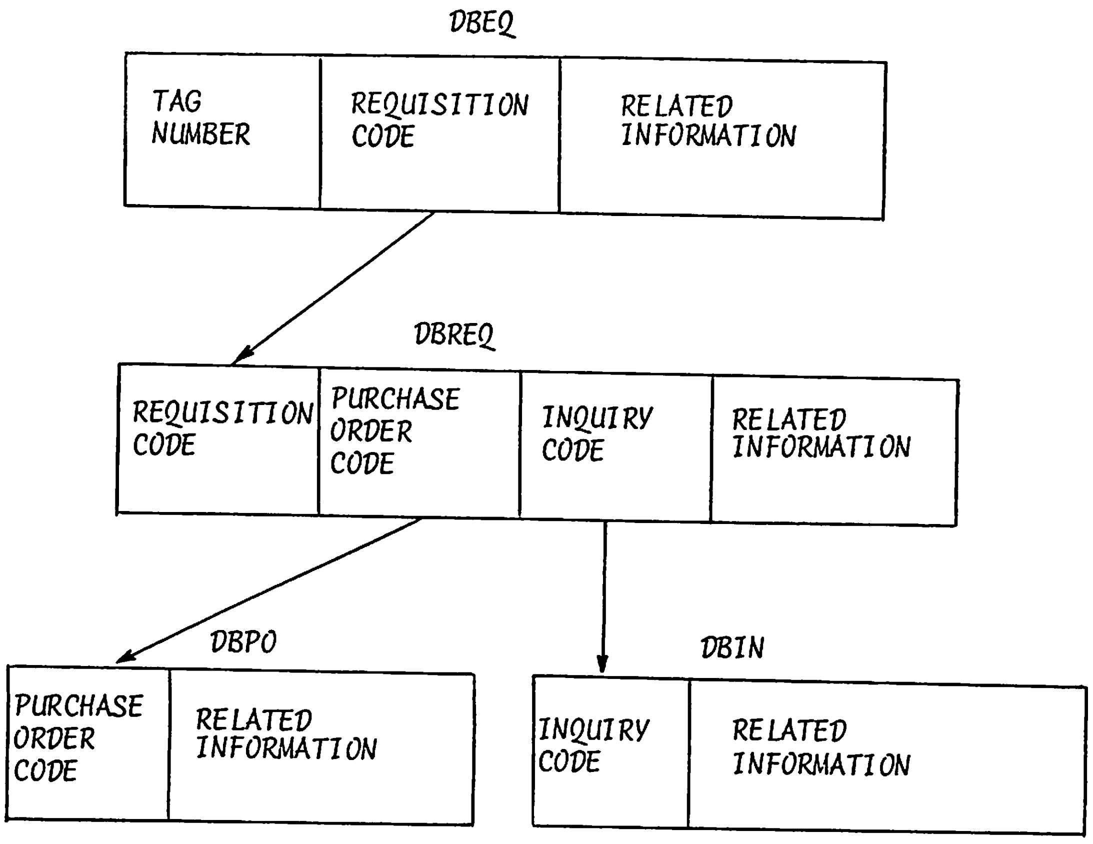
.. Supplier

.. Issue date

.. Order value

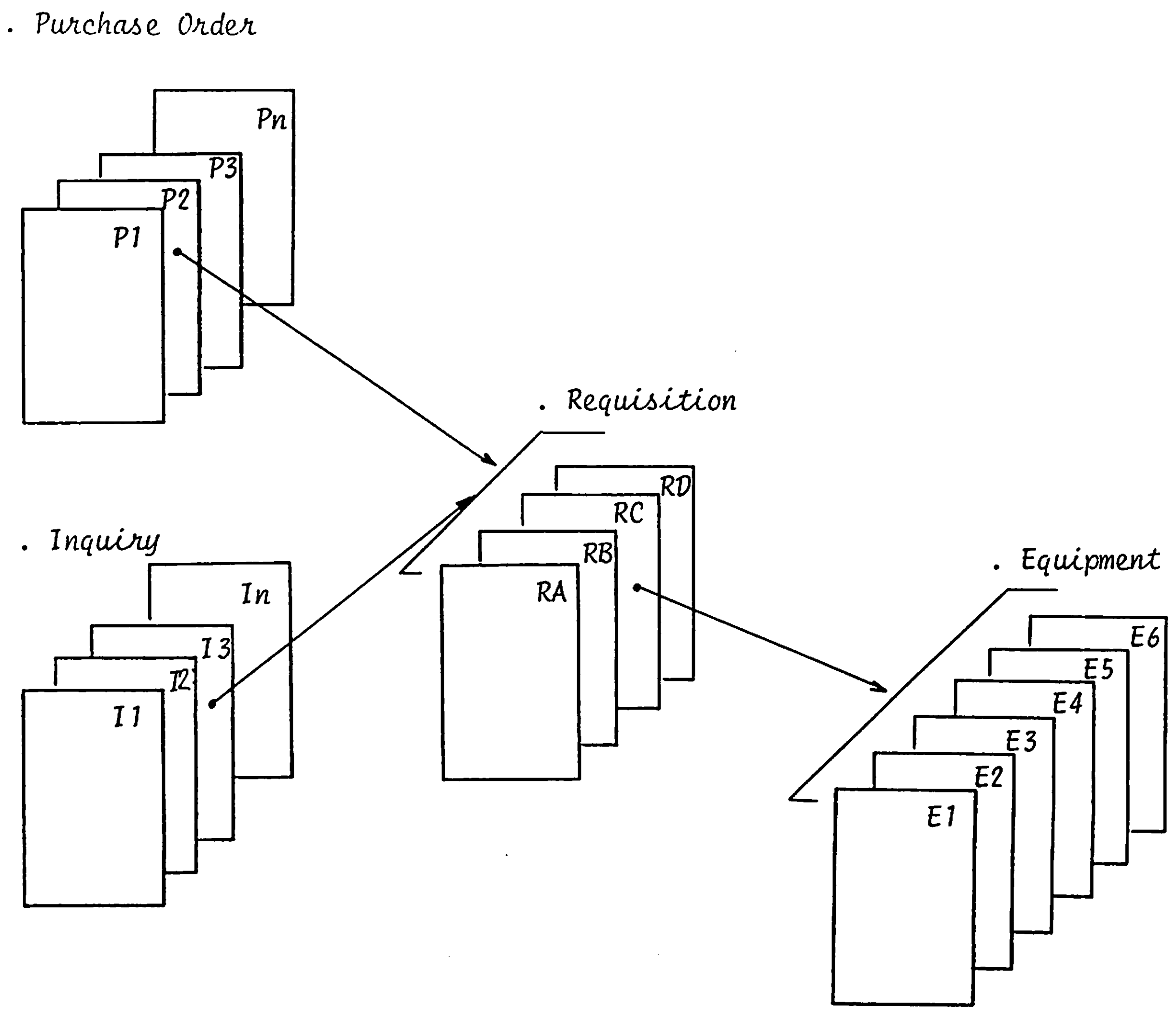
.. Note

To each document there corresponds an IMAGE DATA BASE, which are linked each other as shown in the picture below.



- DBEQ - Equipment data base
- DBREQ - Requisition data base
- DBPO - Purchase order data base
- DBIN - Inquiry data base

This data stored have the following hyerarchy:



## 2.1.2 Data Base Structure

SIGMA data bases have their own structure and they are dimensioned according to user's requirement.

There is no limitation to the number of IMAGE data bases used in the applications of SIGMA, this being indicated by the nature and complexity of the information under control.

In the definition of the SIGMA data structure, we shall use the following concepts:

. ITEM (CODE)

. DATUM(MNEMONIC)

. INFORMATION UNIT

. KEY WORD

. <u>ITEM</u>

Is a set of information identified by a CODE. The SIGMA data base is constituted by the set of all ITEMS.

. <u>DATUM</u>

Is an information identified by a mnemonic and associated to an ITEM. The ITEM is constituted by a set of DATA.

. <u>INFORMATION UNIT</u>

Is the triple:

.. CODE (of the ITEM)

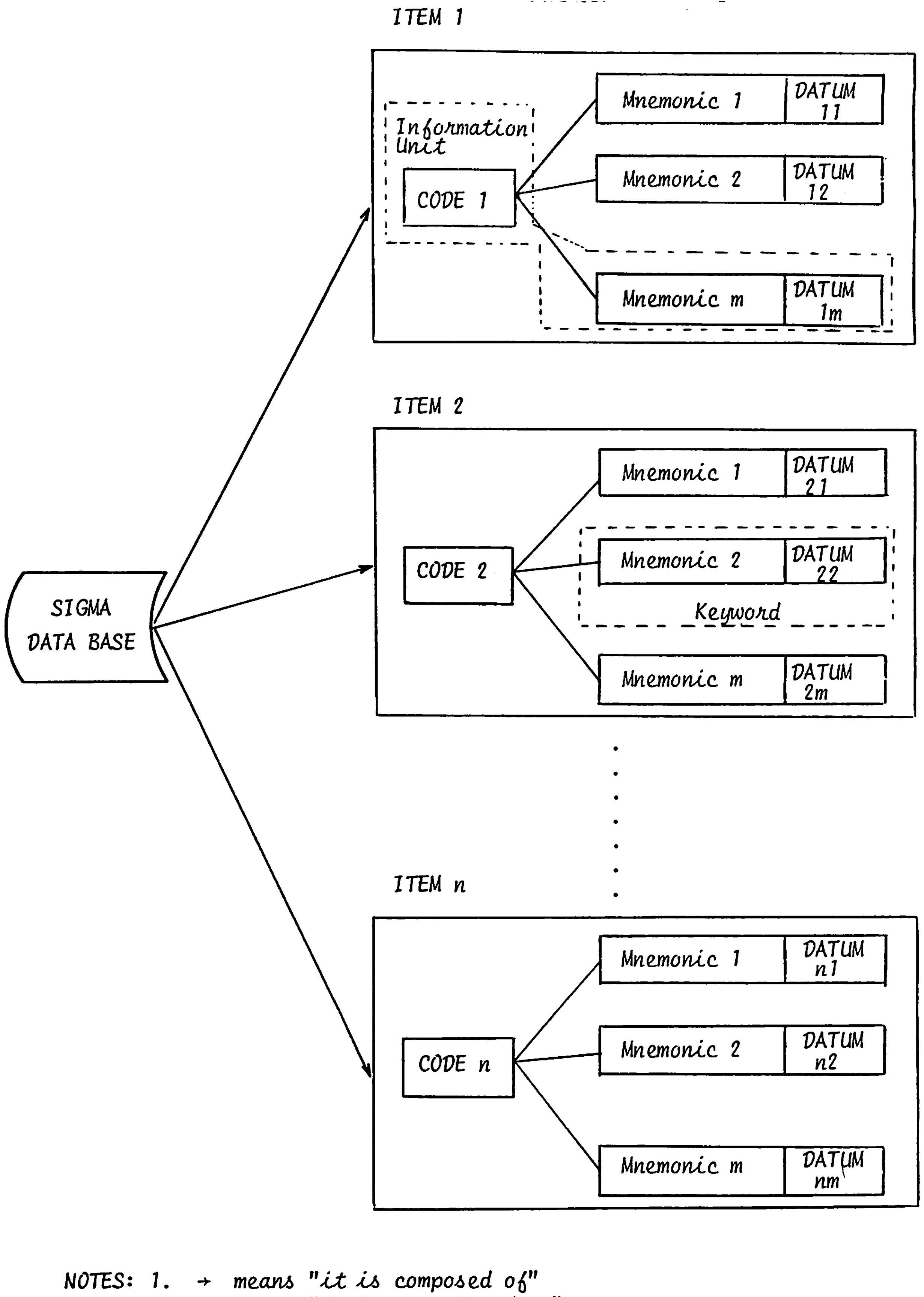
.. MNEMONIC (of the DATUM)

.. DATUM

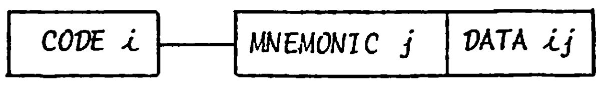
. <u>KEY WORD</u>

Is the double MNEMONIC + DATUM especially chosen by the retrieval of ITEMS.

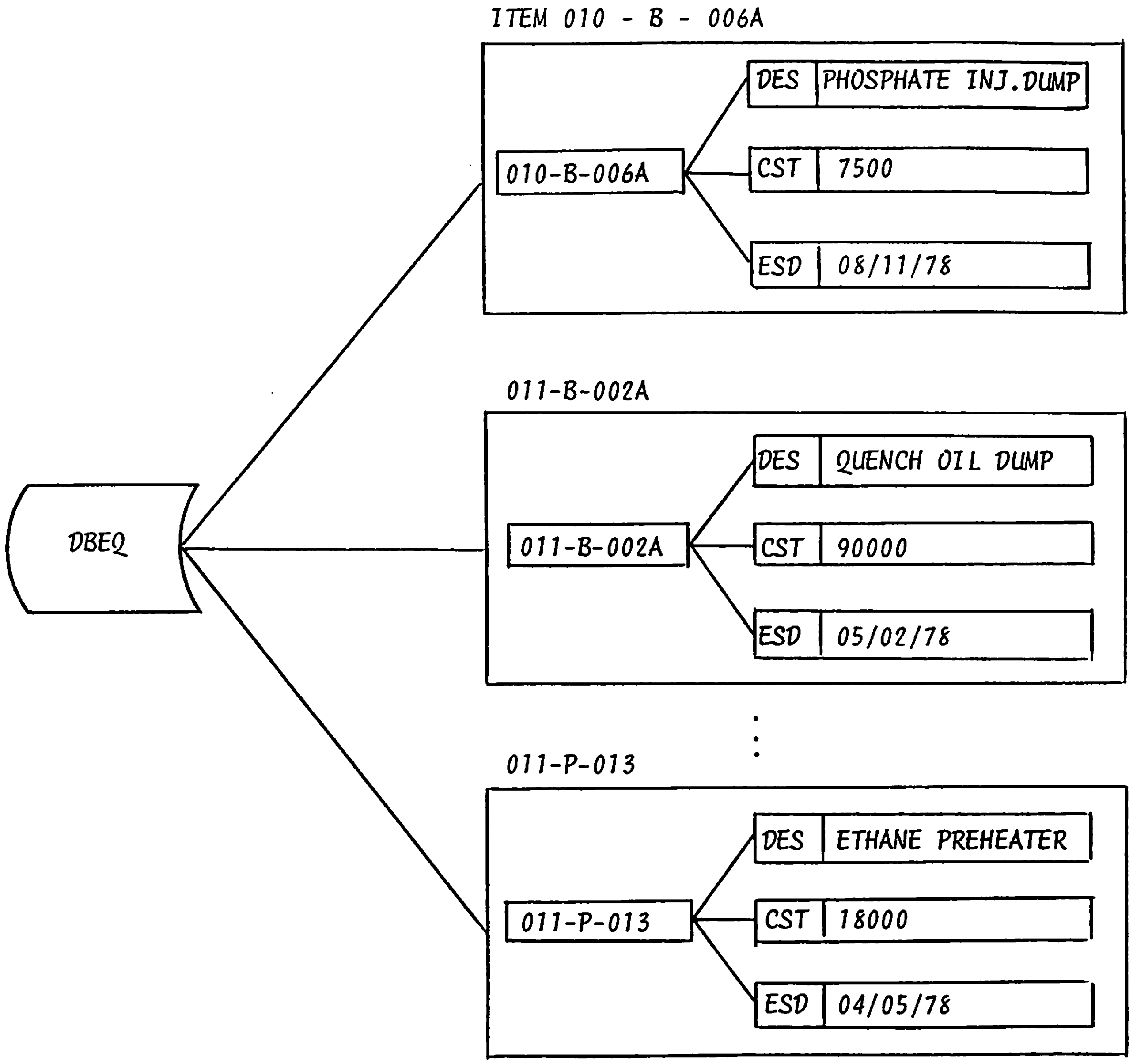
In the next figure we have a graphic representation of these concepts.



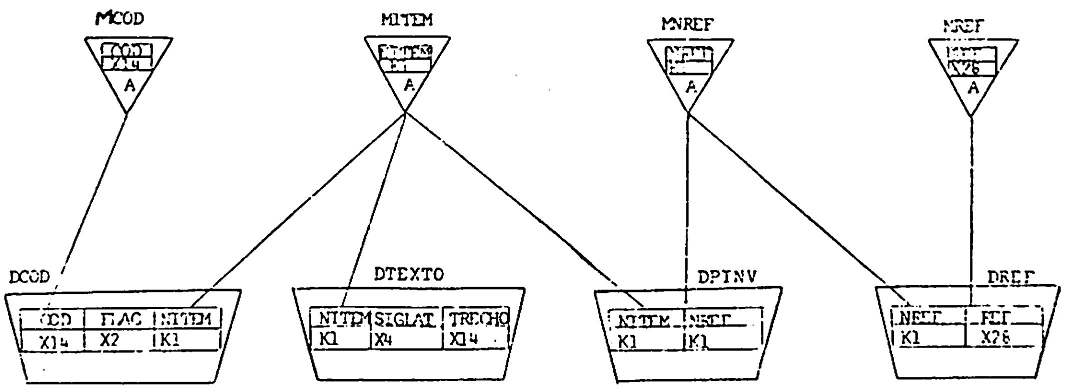
NOTES: 1. → means "it is composed of"
2. — means "it is associated to"
3. n = number of ITEMS; m = number of mnemonics
4. The UNIT INFORMATION is composed of



In the next figure we have an example of the data base DBEQ, which contains the EQUIPMENT LIST.



SIGMA uses IMAGE's data bases with a structure as is shown in figure below:



The detail data sets have the following characteristics:

. Code data set (DCOD)

In this detail data set are stored the CODES of the ITEMS. The elements of the data set are:

.. the code (COD)

.. the atualization flag (FLAG)

.. the internal number of the item generated by the system (NITEM)

. Text data set (DTEXTO)

In this detail data set are stored the information unities. The elements of the data set are:

.. the number of the item (NITEM)

.. the mnemonic of the DATUM

.. the DATUM

. Inverted search data set (DPINV)

In this detail data set are stored the association: number of item (NITEM) - number of keyword (NREF).

. Keyword data set (DREF)

In this detail data set are stored the keywords. The elements of data set are:

.. Keyword (REF)

.. Number associated to keyword (NREF)

This structure is handled by 10 programs, seven in SPL and three in FORTRAN. The programs are funcionally distributed as follows:

$$\begin{tikzcd}[row sep=tiny]
\text{. SIGMA01} \arrow[dr] & \\
 & \text{DATA ENTRY} \\
\text{. SIGMA02} \arrow[ur] & \\
\text{. SIGMA03} \arrow[dr] & \\
 & \text{REPORT SELECTION} \\
\text{. SIGMA04} \arrow[ur] & \\
\text{. SIGMA05} & \text{REPORT EDITING} \\
\text{. SIGMA06} & \text{REPORT DEFINITION} \\
\text{. SIGMA07} \arrow[ddr] & \\
\text{. SIGMA08} \arrow[dr] & \\
 & \text{AUXILIARY PROGRAMS} \\
\text{. SIGMA09} \arrow[ur] & \\
\text{. SIGMA10} \arrow[uur] &
\end{tikzcd}$$

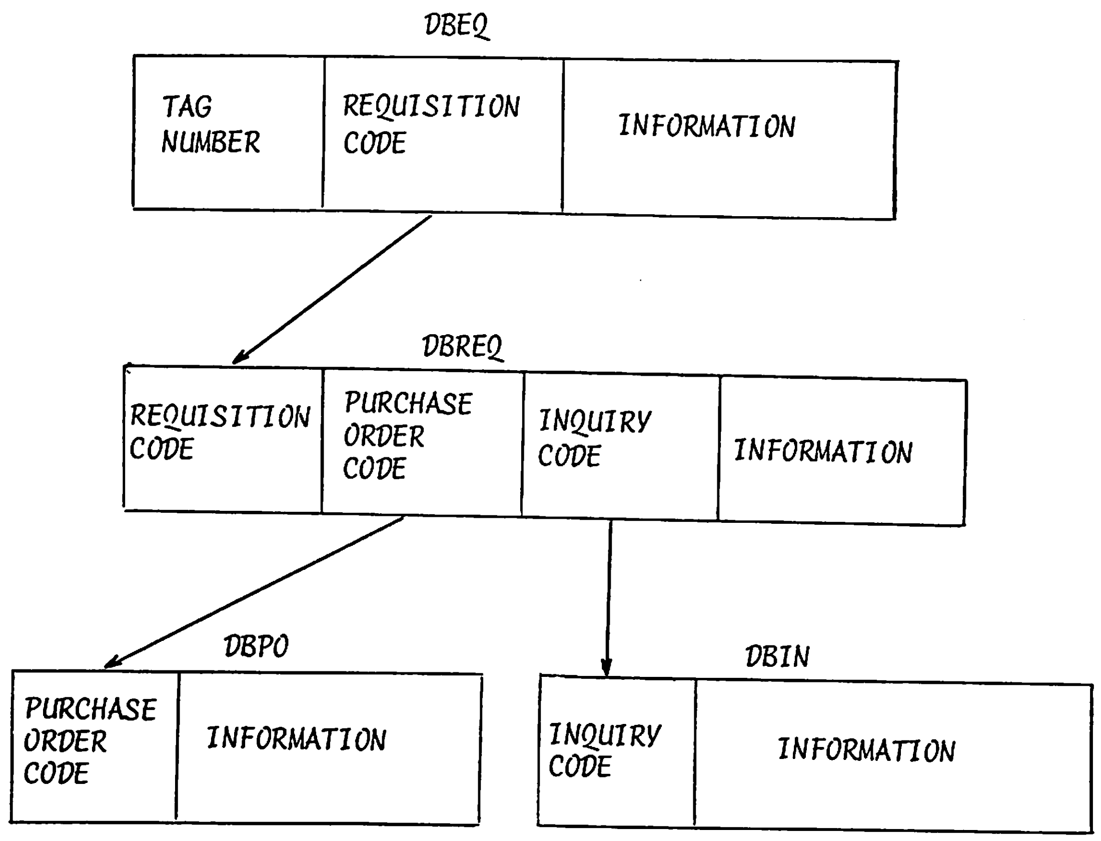
## 2.1.3
## Data Base Linkage

SIGMA data bases are independent among themselves regarding creation and updating.

In the process of information retrieval however, it is possible to agree with various data bases, since there are linkage elements among the data bases.

A linkage element between two data bases is a data element common to two data bases. In one of them it is necessarily a code.

The figure below shows the linkages among data bases DBEQ, DBREQ, DBIN, DBPO.



## 2.2
## Data Base Handling

### 2.2.1
### Data Base Updating

Data base updating is performed through four commands:

. ADD

. REPLACE

. DELETE

. EQUIVALENCE

. *ADD*

```
A  code1 [/code2],mnemonic,"data"
```

*A 011 - B - 011A,CST,"4000"*

*A 011 - B - 011A/011 - B - 011A,ESD,"01/06/78"*

. *REPLACE*

```
R code1 [/code2],mnemonic,"data"
```

*R 011 - F - 012,CST,"700000"*

*R 011 - F - 012/011 - F - 013,ESD,"06/03/78"*

. *DELETE*

```
D code1 [[/code2], [mnemonic]]
```

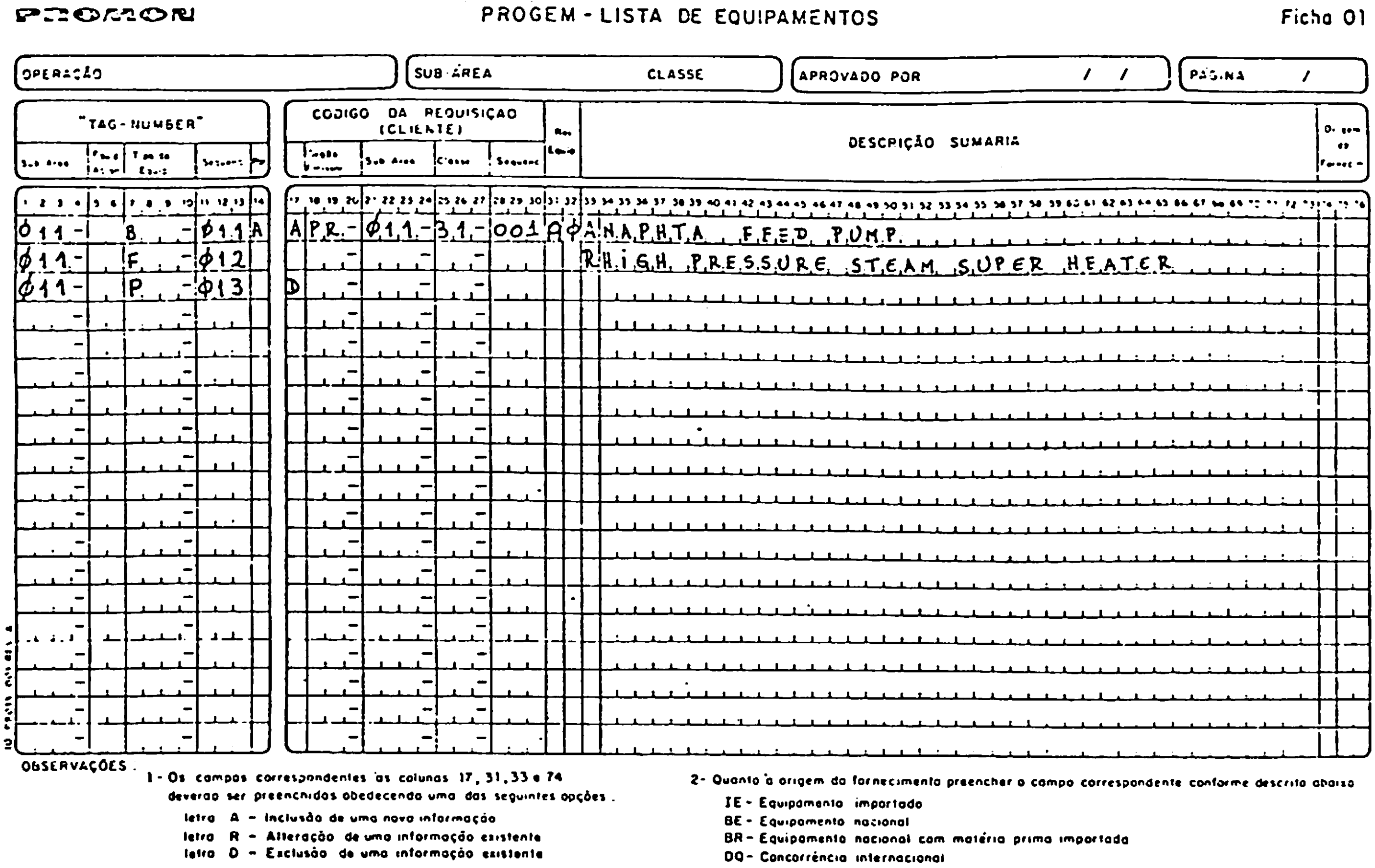
*D 011 - B - 002A*

*D 011 - B - 002A/011 - B - 002B*

*D 011 - B - 002A/CST*

*D 011 - B - 002A/011 - B - 002B,CST*

For batch processes, it is possible to update the data base through records, with pre-defined lay-outs. The next figure shows this alternative.



PROMON PROGEM - LISTA DE EQUIPAMENTOS Ficha 01

OPERAÇÃO | SUB ÁREA CLASSE | APROVADO POR / / | PÁGINA /

| "TAG-NUMBER" Sub Área | Fase | Tipo de Equip. | Sequência | | Codigo da Requisição (Cliente) | Sub Área | Classe | Sequência | Rev. | | | Descrição Sumaria | Origem do Fornec. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 2 3 4 | 5 6 | 7 8 9 10 | 11 12 13 | 14 | 17 18 19 20 | 21 22 23 24 | 25 26 27 | 28 29 30 | 31 | 32 | 33 | 34 … 73 | 74 … 76 |
| Ø11- | | B - | Ø11 | A | A PR- | Ø11- | 31- | 001 | A | Ø | A | NAPHTA FEED PUMP | |
| Ø11- | | F - | Ø12 | | | | | | | | R | HIGH PRESSURE STEAM SUPER HEATER | |
| Ø11- | | P - | Ø13 | | D | | | | | | | | |

OBSERVAÇÕES:

1 - Os campos correspondentes às colunas 17, 31, 33 e 74 deverão ser preenchidos obedecendo uma das seguintes opções.
letra A - Inclusão de uma nova informação
letra R - Alteração de uma informação existente
letra D - Exclusão de uma informação existente

2 - Quanto à origem do fornecimento preencher o campo correspondente conforme descrito abaixo
IE - Equipamento importado
BE - Equipamento nacional
BR - Equipamento nacional com matéria prima importada
DQ - Concorrência internacional

## 2.2.2
## Information Retrieval

We have three types of information retrieval:

- . Through the CODE
- . Through the KEY WORDS
- . From the ITEMS which were updated

<u>Retrieval through CODE</u>:

Retrieval through the CODE is executed by the command:

$$\boxed{L \quad code_1 \left[ [/code_2], [mnemonic] \right]}$$

```
=>L 011-  B  -002A,DES
* 011-  B  -002A
    DES QUENCH OIL PUMP
=>L 011-  B  -002A
* 011-  B  -002A
    CST 90000
    DES QUENCH OIL PUMP
    ESD 02/05/78
    REQ PR-011-31-001
    TIP IMP
=>L 011-  B  -002A/011-  B  -002B,DES
* 011-  B  -002A
    DES QUENCH OIL PUMP
* 011-  B  -002B
    DES QUENCH OIL PUMP
```

In this type of retrieval we have:

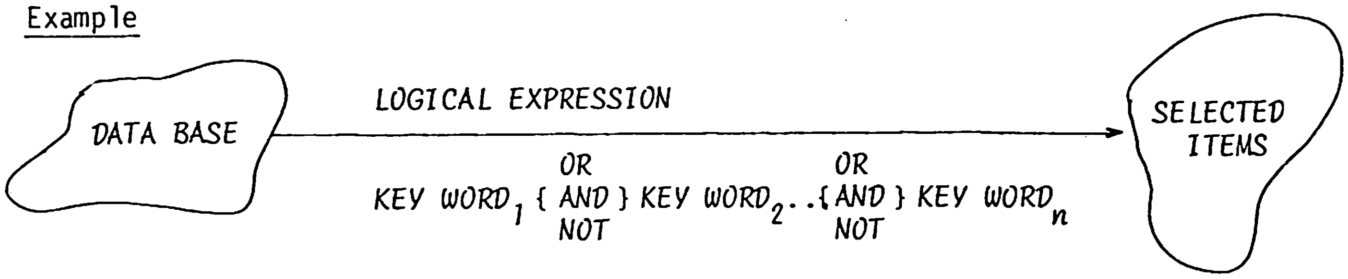
.. all information must be in the same DATA BASE

.. the retrieved information can only be edited in a standard form

Retrieval through KEYWORDS

The KEY WORDS can be combined through Boolean operators (OR,AND,NOT) making logical expressions which will be used in the selection of a sub set of ITEMS.

Example



The retrieved ITEMS can be edited by a user pre-defined report.

Retrieval through ITEMS

The third type of retrieval is that where only the ITEMS which were updated in a given period are retrieved.

We will see in more detail these two last types of retrieval in the next section.

## 2.3
## Reports

The printing of reports requires three phases or steps:

. Definition

. Selection

. Edition

### 2.3.1
### Report Definition

The reports are defined through an oriented language:

The following elements are supplied in the definition of a report:

. Report title

. Selection specification

. Heading specification

. Column specification

. Sort specification

In the next figure we show the definition of three reports.

. REPORT 1

The characteristics are:

. All information is in the same data base

. ALL ITEMS are selected

```
      BEGIN "REP1";
        FLAG:=" ";
        << SELECTION SPECIFICATION >>
          A:="$$$:ALL EQUIPMENTS";
        << HEADLINE  SPECIFICATION >>
          CABEC:= " EQUIPMENT LIST REPORT";
          CABEC:= " PROJECT CODE: PS01";
          CABEC:= " REPORT  CODE: REP1";
        << COLUMN    SPECIFICATION >>
          COLUNA:=COD," EQUIPMENT          CODE",14;
          COLUNA:=DES,"EQUIPMENT  DESCRIPTION",35;
          COLUNA:=ESD,"SCHEDULEDDELIVERY",9;
          COLUNA:=TIP,"IMP",3;
          COLUNA:=CST,"    COST",10,(M);
        << SORT      SPECIFICATION >>
          $CONTROLE:=COD,(1,4),A;
          $CONTROLE:=COD,(7,3),A;
          $CONTROLE:=COD,(5,2),A;
          $CONTROLE:=COD,(11,4),A;
       END.
```

. REPORT 2

In this report we have:

. The information is in three data bases

. Only the imported equipments (A:=TIP:IMP) are selected.

```
BEGIN "REP2";
  FLAG:=" ";
  << SELECTION SPECIFICATION >>
    A:="TIP:IMP";
  << HEADLINE  SPECIFICATION >>
    CABEC:= " REQUISITION  REPORT - ONLY THE EQUIPMENTS TO BE IMPORTED";
    CABEC:= " PROJECT CODE: PS01";
    CABEC:= " REPORT  CODE: REP2";
  << COLUMN    SPECIFICATION >>
    COLUNA:=REQ," REQUISITION      CODE",13;
    COLUNA:=COD," EQUIPMENT         CODE",14;
    COLUNA:=DES,"EQUIPMENT  DESCRIPTION",32;
    COLUNA:=RSI,"SCHEDULED  ISSUE",9;
    COLUNA:=RID," ISSUE   DATE",8;
    COLUNA:=RRP,"RECIVED PURCH.",8;
  << SORT      SPECIFICATION >>
    $CONTROLE:=REQ,(1,13),I,A;
    $CONTROLE:=COD,(1,4),A;
    $CONTROLE:=COD,(7,3),A;
    $CONTROLE:=COD,(5,2),A;
    $CONTROLE:=COD,(11,4),A;
 END.
```

. REPORT 3

In this report we have:

. The information in the three data bases

. Only the items that were updated in the last period (FLAG:="*") are selected.

NOTE: This period is defined by the user of the SYSTEM.

```
BEGIN "REP3";
  FLAG:="*";
  << SELECTION SPECIFICATION >>
    A:="$$$:ALL EQUIPMENTS";
  << HEADLINE  SPECIFICATION >>
    CABEC:= " PURCHASE ORDER REPORT - "
             T 26 "ONLY THE ITEMS THAT WAS UPDATED AFTER 10/05/78";
    CABEC:= " PROJECT CODE: P801";
    CABEC:= " REPORT  CODE: REP3";
  << COLUMN    SPECIFICATION >>
    COLUNA:=POD,"PURCHASF ORDER     CODE",8;
    COLUNA:=REQ," REQUISITION       CODE",13;
    COLUNA:=COD,"  EQUIPMFNT         CODE",14;
    COLUNA:=DES,"EQUIPMENT  DESCRIPTION",32;
    COLUNA:=TIP,"IMP",3;
    COLUNA:=CST,"COST ($)",9,(M);
    COLUNA:=SUP,"SUPPLIER",14;
  << SORT      SPECIFICATION >>
    $CONTROLE:=POD,(1,6),I,T,A;
    $CONTROLE:=REQ,(1,13),I,T,A;
    $CONTROLE:=COD,(1,4),A;
    $CONTROLE:=COD,(7,3),A;
    $CONTROLE:=COD,(5,2),A;
    $CONTROLE:=COD,(11,4),A;
    TOTAL:=CST;
  END.
```

2.3.2
Selection

SIGMA allows the following types of selections in the report edition:

. Through KEY WORDS (Logical expression)

. Through CONDITIONS

. Only the ITEMS that we updated in one period (FLAG:="*").

2.3.3
Edition

Edition is done in the final phase of the report issuing so that it is possible to use the same selection to issue various types of reports.

SIGMA possesses the following capabilities for edition:

. classification of columns

. inhibition of columns

. totalization of columns

. generation of columns as linear combination of previous columns.

In the next figures we show three typical reports.

```
                                                                              PAG      1
  21/SET/1978      16:49 HORAS
 EQUIPMENT LIST REPORT
 PROJECT CODE: PS01
 REPORT  CODE: RFP1

.....................................................................................
  EQUIPMENT    EQUIPMENT  DESCRIPTION                        SCHEDULED IMP     COST
    CODE                                                     DELIVERY
.....................................................................................

010-  B  -006A PHOSPHATE INJECTION PUMP                      11/08/78  IMP      7,500

010-  B  -006B PHOSPHATE INJECTION PUMP                      12/01/78  IMP      7,500

011-  B  -002A QUENCH OIL PUMP                               02/05/78  IMP     90,000

011-  B  -002B QUENCH OIL PUMP                               02/05/78  IMP     90,000

011-  B  -002C QUENCH OIL PUMP                               02/05/78  IMP     90,000

011-  B  -002D QUENCH OIL PUMP                               02/05/78  IMP     90,000

011-  B  -004B MIDDLE OIL DRAW-OFF PUMP                      03/10/78  IMP      2,000

011-  B  -005A FUEL OIL PUMP                                 02/05/78  IMP      2,000

011-  B  -011A NAPHTA  FEED PUMP                             01/06/78  IMP      4,000

011-  B  -011B NAPHTA  FEED PUMP                             01/06/78  IMP      4,000

011-  F  -012  HIGH PRESSURE STEAM SUPER HEATER              06/03/79          800,000

011-  F  -013  HIGH PRESSURE STEAM SUPER HEATER              06/03/79          800,000

011-  P  -013  ETHANE PREHEAETER                             05/04/78           18,000

011-  P  -015A QUENCH OIL COOLER                             03/09/78  IMP     54,000

011-  P  -015B QUENCH OIL COOLER                             03/09/78  IMP     54,000

011-  P  -015C QUENCH OIL COOLER                             03/09/78  IMP     54,000

011-  P  -015D QUENCH OIL COOLER                             03/09/78  IMP     54,000

011-  P  -015E QUENCH OIL COOLER                             03/09/78  IMP     54,000

011-  P  -015F QUENCH OIL COOLER                             03/09/78  IMP     54,000

011-  P  -016A FUEL OIL COOLER                               07/07/78  IMP     35,000

011-  P  -016B FUEL OIL COOLER                               07/07/78  IMP     35,000

011-  P  -016C FUEL OIL COOLER                               07/07/78  IMP     35,000
.................................                            ....................
```

CONTINUE...

```
  21/SET/1978      16:56 HORAS
 REQUISITION  REPORT - ONLY THE EQUIPMENTS TO BE IMPORTED
 PROJECT CODE: P901
 REPORT  CODE: REP2

 ••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
 REQUISITION     EQUIPMENT     EQUIPMENT  DESCRIPTION                SCHEDULED    ISSUE    RECEIVED
    CODE           CODE                                                 ISSUE      DATE     PURCH.
 ••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••

 PR-010-31-001 010-  B  -006A PHOSPHATE INJECTION PUMP          01/03/77  01/02/77 01/04/77

               010-  B  -006B PHOSPHATE INJECTION PUMP          01/03/77  01/02/77 01/04/77

 PR-011-31-001 011-  B  -002A QUENCH OIL PUMP                   01/02/77  01/05/77 01/05/77

               011-  B  -002B QUENCH OIL PUMP                   01/02/77  01/05/77 01/05/77

               011-  B  -002C QUENCH OIL PUMP                   01/02/77  01/05/77 01/05/77

               011-  B  -002D QUENCH OIL PUMP                   01/02/77  01/05/77 01/05/77

               011-  B  -004B MIDDLE OIL DRAW-OFF PUMP          01/02/77  01/05/77 01/05/77

               011-  B  -005A FUEL OIL PUMP                     01/02/77  01/05/77 01/05/77

               011-  B  -011A NAPHTA  FEED PUMP                 01/02/77  01/05/77 01/05/77

               011-  B  -011B NAPHTA  FEED PUMP                 01/02/77  01/05/77 01/05/77

 PR-011-42-001 011-  F  -012  HIGH PRESSURE STEAM SUPER HEATER  01/10/77  01/11/77 01/12/77

               011-  F  -013  HIGH PRESSURE STEAM SUPER HEATER  01/10/77  01/11/77 01/12/77

 PR-011-43-001 011-  P  -013  ETHANE PREHEAETER                 01/12/77  01/06/77 01/10/77

               011-  P  -016A FUEL OIL COOLER                   06/07/77  06/10/77 06/12/77

               011-  P  -016B FUEL OIL COOLER                   06/07/77  06/10/77 06/12/77

               011-  P  -016C FUEL OIL COOLER                   06/07/77  06/10/77 06/12/77

               011-  P  -016D FUEL OIL COOLER                   06/07/77  06/10/77 06/12/77

 PR-011-43-002 011-  P  -015A QUENCH OIL COOLER                 06/07/77  06/10/77 06/12/77

               011-  P  -015B QUENCH OIL COOLER                 06/07/77  06/10/77 06/12/77

               011-  P  -015C QUENCH OIL COOLER                 06/07/77  06/10/77 06/12/77

               011-  P  -015D QUENCH OIL COOLER                 06/07/77  06/10/77 06/12/77

               011-  P  -015E QUENCH OIL COOLER                 06/07/77  06/10/77 06/12/77

               011-  P  -015F QUENCH OIL COOLER                 06/07/77  06/10/77 06/12/77
 ••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••

 ••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
```

```
  21/SET/1978      17: 2 HORAS
 PURCHASE ORDER REPORT - ONLY THE ITEMS THAT WAS UPDATED AFTER 10/05/78
 PROJECT CODE: P301
 REPORT  CODE: REP3

-----------------------------------------------------------------------------------------------------
PURCHASE  REQUISITION     EQUIPMENT     EQUIPMENT  DESCRIPTION              IMP COST ($)  SUPPLIER
 ORDER         CODE           CODE
  CODE
-----------------------------------------------------------------------------------------------------

31-001    PR-010-31-001 010-  B  -006A PHOSPHATE INJECTION PUMP           IMP      7,500 WORTHINGTON

                        010-  B  -006B PHOSPHATE INJECTION PUMP           IMP      7,500 WORTHINGTON
=====================================================================================================
         *                                                                        15,000
=====================================================================================================

          PR-011-31-001 011-  B  -002A QUENCH OIL PUMP                    IMP     90,000 WORTHINGTON

                        011-  B  -002B QUENCH OIL PUMP                    IMP     90,000 WORTHINGTON

                        011-  B  -002C QUENCH OIL PUMP                    IMP     90,000 WORTHINGTON

                        011-  B  -002D QUENCH OIL PUMP                    IMP     90,000 WORTHINGTON

                        011-  B  -004B MIDDLE OIL DRAW-OFF PUMP           IMP      2,000 WORTHINGTON

                        011-  B  -005A FUEL OIL PUMP                      IMP      2,000 WORTHINGTON

                        011-  B  -011A NAPHTA  FEED PUMP                  IMP      4,000 WORTHINGTON

                        011-  B  -011B NAPHTA  FEED PUMP                  IMP      4,000 WORTHINGTON
=====================================================================================================
                                                                                 372,000
=====================================================================================================
=====================================================================================================
*                                                                                387,000
=====================================================================================================

42-001    PR-011-42-001 011-  F  -012  HIGH PRESSURE STEAM SUPER HEATER       800,000 CONFAB

                        011-  F  -013  HIGH PRESSURE STEAM SUPER HEATER       800,000 CONFAB
=====================================================================================================
                                                                               1,600,000
=====================================================================================================
=====================================================================================================
*                                                                              1,600,000
=====================================================================================================
-----------------------------------------------------------------------------------------------------

                                                                                            CONTINUA-
```

## 2.4
Additional Resources

### 2.4.1
Data editing

SIGMA does the following editing checks:

. CODE of ITEM

. Date

. Monetary Values

### 2.4.2
Handling of Dates and Monetary Values

. Dates

.. $(\mathrm{Date}_1 - \mathrm{Date}_2)$ = running days

.. $(\mathrm{Date}_1 + \text{Running Days}) = \mathrm{Date}_2$

. Monetary

.. $\mathrm{COST} = \mathrm{COST}_0 \{\pm\} K_1 \mathrm{COST}_1 \{\pm\} \ldots \{\pm\} Kn.\mathrm{COSTn}$

$\{K_1, K_2, \ldots K_n\} \varepsilon R$

## 3. SYSTEM PERFORMANCE

SIGMA performance has been kept high, considering its generality. In the figure below we show some results of SIGMA application.

. DATA STORAGE

| DATA BASE | # ITEMS | # DATA PER ITEM | # KEY WORDS PER ITEM | DISC SPACE (# SECTORS) |
|---|---|---|---|---|
| 1 | 2700 | 23 | 4 | 5012 |
| 2 | 8000 | 19 | 4 | 11142 |
| 3 | 3500 | 4 | 0 | 2426 |

. DATA BASE UPDATING

| | |
|---|---|
| .. 1000 information (no key-word) | = $ 25.00 |
| .. 1000 key-words | = $ 37.00 |

. REPORT WITH 1000 LINES

| | |
|---|---|
| .. All information are in the same data base | = $ 15.00 |
| .. The information are in two data bases | = $ 60.00 |

NOTE: Reference values used in costs calculations:

. Elapsed time = $ 12.00 per hour

. CPU time = $ 160.00 per hour

. 1000 Disc Access = $ 1.50

. 1000 Printed Lines = $ 2.75

. 1000 Card Read = $ 2.00

. LIMITATIONS

.. Maximum Number of UNIT INFORMATION per IMAGE data base = 64000
.. Maximum Number of DATA per ITEM = 100
.. Maximum Number of KEYWORDS per ITEM = 24
.. Maximum Number of Characters per CODE = 14
.. Maximum Number of Characters per DATUM = 56
.. Maximum Number of Characters per KEYWORD = 25
.. Maximun Number of IMAGE data base in a report = 9
.. Maximum Number of Columns on a report = 14
.. Maximum Number of headlines = 3

oOo

# A DATA DICTIONARY/DIRECTORY DRIVEN CLINICAL DATA MANAGEMENT SYSTEM

ERIC S. HERBEL
HOECHST-ROUSSEL PHARMACEUTICALS, INC.

## ABSTRACT

A clinical research data management system was developed which is driven by a hierarchical data base form of a data dictionary using variable length record KSAM files. The actual structure of the data dictionary (the hierarchical data base approach, and variable length lists implemented in KSAM variable length record files) is emphasized. The system includes data entry, editing, on-line corrections, a variety of IBM/370 resident interfaces, and a generalized IMAGE data base schema write/loader. All of the system modules are driven by the data dictionary. Details of the IMAGE interface are also emphasized.

## OUTLINE

- The Data Dictionary/Directory concept in general
  - and the HP implementation in detail
- Overview of the clinical data management system
  - Data Entry - key to disk emulation
  - Data Preparation - editing, corrections, listing, etc.
  - Data Dictionary - definition, modification, reporting
  - Data Management - Image schema writer/loader, interfaces to IBM/370 DBMSs.
- IMAGE/3000 schema writer/loader in depth.

## DESCRIPTION OF DATA DICTIONARY/DIRECTORY

Commercially available Data Dictionary/Directory systems are typically characterized by:

- identification of each element or field (name, aliases, description)
- type specifications for each element (numeric/character, etc.), field size and other physical descriptions

- reporting specifications for the field (output format, and default label)
- data editing criteria - a range or table of acceptable values
- administrative information such as who can access various elements, and information on which applications use specific elements
- and maintenance responsibility for the element.

A data directory is an extension of the data dictionary concept intended primarily for machine processing of the meta data(data which defines data). On the other hand, a data dictionary is intended to aid in human understanding of the data. Applications of the directory concept have included automatic generation of data definitions (schema) for input to various DBMS.

## IMPLEMENTATION OF A DATA DICTIONARY/DIRECTORY ON THE HP3000

Most data dictionary systems currently in use appear primarily concerned with aiding the DATA BASE ADMINISTRATOR and other HUMAN data dictionary users. We were certainly concerned with this aspect of the use of the data dictionary, but placed more emphasis on the DIRECTORY side of the concept. The net result of this emphasis shift is that the majority of application programs in the system are DRIVEN by the central Data Dictionary/ Directory.

The Data Dictionary/Directory implemented on the HP contains most of the traditional elements mentioned above, but it does not presently contain security information or pointers to application programs. However some significant additions were made.

First at the element or field level a number of modifications were made.

- KEY fields are noted; this will cause an IMAGE automatic master to be created for it
- On-line as well as background edit specificatiOns were added facilitating editing during data entry or later in background
- A link to a decode table for the field was included,e.g. the decode file table entry name(used as a prefix for its KSAM key).
- Should the field be verified during data entry?
- Should the field be automatically duplicated from a previous record in the data entry form(set of defined records)?

The second and most important set of additions center of the masterfile's structure. This structuring information includes physical RECORD definitions, logical UNIT definition, and when these UNITs occur, or the ENCOUNTER definition.

RECORD content, i.e. which elements make up each 80 byte masterfile record ( this is the data entry medium); also where each field starts on the record, and whether it's repeated as a contiguous block or array.

Logical UNIT structure, i.e. which 80 byte records make up a logical UNIT of information (later used as the entity equivalent to an IMAGE data set). These logical units are generally entered together during data entry ( called a FORM at data entry ). Various key fields appearing on every record aid in the logical connection of records in a UNIT.

These structuring entities are sufficient in most cases to process the data, i.e. define the data entry form, define specifications for the batch editor, and define the schema for an IMAGE data base. However, we've gone one step further in the structure progression - adding the element of time, i.e. when a group of logical UNITs are to occur. This is called an ENCOUNTER definition, corresponding to a clinical visit encounter between a subject and physician. During each visit, various UNITs of logically related data are to be collected, so an encounter consists of a list of logical UNITs. To facilitate automatic assignment of UNITs to a given encounter, an ENCOUNTER CLASSIFICATION EQUATION is included - which defines membership criteria, e.g. VISIT NUMBER=5 or Days into study is between 4 and 7. This structure actually captures the final dimension necessary to fully define a clinical study to application programs which will categorize, inventory and even analyze the body of data.

## ACTUAL DATA DICTIONARY/DIRECTORY FILE STRUCTURE

To facilitate most efficient use of the entire system - a true network structure was chosen. As a result, individuals defining a new data dictionary/directory can "point" to any entity already defined, at either the UNIT, RECORD or FIELD level. This automatic GLOBAL capability has greatly enhanced the use of the system. This has also encouraged use of STANDARD UNITs or RECORDs across studies.

The data dictionary/directory system is implemented in a set of variable length record binary KSAM files(except the field level file which is fixed length). Each KSAM file in the set corresponds to a level in the structural hierarchy.

The linkage between the various files in the hierarchy is maintained by a variable length array of pointers with associated information pointing to the next lower level file. The pointers are in the form of KSAM keys into the appropriate subordinate file. Selected information belonging to a subordinate file is often stored along with the pointer to the record(i.e., the Field start column is stored at the RECORD file level). This convention was adopted to facilitate use of global information, i.e. information used by many studies or records, and information thought to change most frequently was stored at a higher level in the hierarchy. As an example, the length, edit specs, label, etc. would probably not change for the field blood pressure between studies. The starting position on the 80 byte record probably would change from study to study, so the starting column was stored at the RECORD level in the

```
--------------------------------------------------------------------
|                                                                  |
|                       --------------------                       |
|                       | STUDY Subfile    |                       |
|                       --------------------                       |
|                      *         *          *                      |
|                    *           *            *                    |
|                  *             *              *                  |
|                *      --------------------      *                |
|              *        | ENCOUNTER Subfile|        *              |
|            *          --------------------          *            |
|          *          *                                 *          |
|        *         *                                      *        |
|      *        *                                           *      |
-------------------                               ------------------
| UNIT Subfile    |                               | FORM Subfile    |
-------------------                               ------------------
|           *                                             *        |
|               *                                     *            |
|                   *                             *                |
|                        *                   *                     |
|                        ---------------------                     |
|                        | RECORD Subfile    |                     |
|                        ---------------------                     |
|                                 |                                 |
|                                 v                                 |
|                        ---------------------                     |
|                        | FIELD Subfile     |                     |
|                        ---------------------                     |
|                                                                  |
|                                                                  |
--------------------------------------------------------------------
```

Figure 1. Illustration of the data dictionary hierarchy.

hierarchy, so many RECORDs can "point" to the same common blood pressure definition.

The STUDY level file records contain descriptive fixed length information about a clinical study(masterfile), as well as three variable length pointer stacks. These three stacks contain character type keys to the attached UNITs, the ENCOUNTERs and the data entry FORMs. The three stacks share contiguous space in the record - with the base of the FORM pointer stack just after the UNIT stack and so on. The Data Dictionary/Directory is generally accessed from FORTRAN, so, the fixed length elements of the record can be accessed after a record is read(using FREADBYKEY) simply by an EQUIVALENCE of the items to the LOGICAL type buffer. The stacks are accessed by actually moving a LOGICAL word at a time into a LOGICAL typed TRANSFER buffer which has been equivalenced to a character variable.

The ENCOUNTER file records contain a fixed length portion which includes the primary key, the ENCOUNTER CLASSIFICATION EQUATION and the top of stack counter, followed by a variable portion containing the member UNIT stack. Each entry in the member UNIT stack contains a PL/I like structure (heterogeneous typed structure) containing the UNIT key (character type) and the number of that type UNIT required in the ENCOUNTER(integer type). Each entry in the stack is accessed by moving a word at a time from the appropriate address in the record buffer to a logical type transfer buffer. The character variable followed by an integer variable are equivalenced to the transfer buffer. Note that the encounter classification equation is later parsed then interpreted by various application programs.

The UNIT file records are very similar to the previous level records, i.e. a fixed length portion for UNIT description followed by a variable portion for the attached RECORD stack (also containing the number of each RECORD type required in UNIT). A new value is added to the fixed length portion of the record - the global counter - to aid in controlling use of the global facility. This is present also at the RECORD and FIELD level. The counter is set to 1 if the UNIT record is local, i.e. not pointed to by any other study, but it's incremented every time it's pointed to. This allows detection of its global status(to provide for a warning on the effects of modifying it), as well as providing for local deletes. If an entity is global, and it's to be "deleted", the global counter is decremented, and the pointer to it is removed(the actual record is not removed). If the counter is 1 and it's being deleted - then actually FREMOVE it.

The data entry FORM file is almost identical.

The RECORD file record is basically the same, except that each entry in its member FIELD stack contains a larger heterogeneous typed structure. The structure includes the FIELD's key(character), its starting column (integer), the member of contiguous repeats of the field on the record(integer) and whether it's required (for editing purposes).

The FIELD file record is the simplest - fixed length binary - containing all the basic information such as type, length, description(label), etc.

## REASONS FOR USING KSAM/3000

The Data Dictionary/Directory could have been implemented using a set of direct access files, with an index into the top most level (study level) providing a table of contents. Pointers to subordinate entities, e.g. pointers to all attached RECORDs in any UNIT, could have been the actual record address in the subordinate file. This approach would have been very efficient in tree traversal time(data dictionary hierarchy traversal). Problems associated with this approach are:

- MPE presently doesn't support direct access variable length records (variable length records being very important for efficient implementation of the overall structure)
- broken pointers, i.e. bad record addresses are very difficult to fix - probably requiring a dual pointer structure (pointer to son from father and vice versa)
- a garbage collection mechanism was needed - would have been necessary to explicitly set this up
- the global use of UNITs, RECORDs or FIELDs would have become extremely difficult to implement - one would need a separate list of availabe entities with their record addresses.

IMAGE/3000 itself could have been used to implement the data dictionary (something occassionally done in commercial systems). The straight hierarchical structure of any single LOCAL data dictionary would have been implemented satisfactorily as illustrated in Figure 3.

The global entity concept would have been very difficult(impossible) to implement using this structure, though. Also, information such as Field start column couldn't have easily or efficiently been stored at the RECORD level (no variable length records in IMAGE, or compound items as chain heads).

The Data Dictionary/Directory was implemented in KSAM/3000 because:

- variable length keyed-access records are supported, so variable length pointer lists with associated information is possible.
- using keys as pointers rather than actual record addresses greatly simplifies implementation of global entities and security/accuracy of pointers.
- KSAM takes care of garbage collection automatically.

Traversing the tree structure using keys into subordinate files is not as fast as direct access using a record address - but it's close (KSAM takes care of that). More importantly though, this small disadvantage in speed is considerably offset by all other capabilities gained (as stated above).

## OVERVIEW OF THE SYSTEM DRIVEN BY THE DATA DICTIONARY/DIRECTORY

The clinical data management system implemented on the HP3000 includes:

- a key-to-disk emulating DATA ENTRY system
- a set of DATA PREPARATION modules, including a batch editor, on-line error corrections, a masterfile inventory procedure and various listing/reporting utilities
- the DATA DICTIONARY/DIRECTORY subsystem itself - containing an interactive formatted screen definition/modification procedure, along with many reporting and auxiliary use

utilities

- and the DATA MANAGEMENT subsystem which contains segments for masterfile tracking(primary residence(IBM vs HP), statistics, etc.), masterfile movement, and an IMAGE schema writer/loader. (Interfaces to IBM resident DBMSs also exist).

All elements of the system are generalized and function for any study based on information gained primarily from the Data Dictionary/ Directory(some modules are implemented using other similarly constructed central files, such as the masterfile tracking file).

## DATA ENTRY SUBSYSTEM

This is a key to disk emulating "keypunch" operation which supports an input/verify data entry cycle. Data is keyed on a micro-processor controlled data entry terminal(the standard HP2645A) utilizing a microprogram downloaded from the HP3000. The data is keyed in 80 column card images using card masks or FORMS which are constructed from information contained in the Data Dictionary/Directory. The microprogram allows for automatic tabbing from field to field , automatic as well as manual duplication of fields (from the previous record in the FORM) and automatic insertion of RECORD identification information. The microprogram communicates with the HP3000 in a background mode with buffering and accepts keying in foreground mode. This results in instantaneous response at the keyboard independent of load on the HP3000 - and also reduces the processing drain data entry would normally place on the HP. During on-line verification , the record being keyed is matched against the corresponding record previously keyed during input. Any discrepancies are resolved then.

Completed data entry batches are automatically routed to the appropriate masterfile based on information contained in the Masterfile Tracking file (this includes automatic submission of IBM bound jobs for those masterfiles resident on the IBM).

## DATA PREPARATION SUBSYSTEM

The Data Preparation subsystem supports background editing and inventory of masterfiles, interactive error corrections and various listing utilities. All modules of the subsystem gain information from the Data Dictionary/Directory and the Masterfile Tracking Control file.

The batch EDITOR works on a complete masterfile or a selected subset, sorts the file by record id and other key fields, then produces a formatted report. The report is constructed using field names as column headers and edit specifications (acceptable data range) for flagging of data errors - both from the data dictionary. The EDITOR knows where to pick up each field on a record based on the field start column and length information, also in the Data

Dictionary. A similar version of the EDITOR exists on the IBM/370, written in PL/I, also running off of the Data Dictionary. The IBM resident version is run by jobs automatically submitted from the HP if the Masterfile Tracking file says the file's primary residence is the IBM.

The interactive ERROR CORRECTION procedure is a formatted screen oriented program which directly updates an HP resident masterfile. Masterfiles are maintained as KSAM files on the HP, so a particular record to be corrected can be located very rapidly using FREADBYKEY or FFINDBYKEY. Once desired modifications are made to the record, it's FUPDATEd on the KSAM file. A log of corrections is maintained for backing them out, or reapplying them in the case of system fails.

The batch INVENTORY procedure takes a masterfile, sorts the file by patient number, then performs an inventory of data records comparing the data present to the expected data structure. This expected data structure - which records comprise a unit, and when/how many units are to occur - is constructed based on the Data Dictionary ENCOUNTER and UNIT structuring records. The INVENTORY procedure runs in either a total inventory listing mode or in an exception mode - listing only patients with missing records.

This subsystem also includes maintenance and reporting facilities for a Decode file - a KSAM/3000 based data decoding table. An ISAM version of this decode file is maintained on the IBM/370 through the HP resident maintenance system - any updates made to the HP file cause a job to be automatically submitted to the IBM for identical updates to be made there.

## DATA DICTIONARY/DIRECTORY SUBSYSTEM

The Data Dictionary/Directory subsystem includes an interactive formatted screen oriented DD/Dir definition and modification program, automatic coding manual generation, a data entry FORM definition interface, various reporting utilities and a translate/export module to translate the Data Dictionary into its IBM/370 version and export a copy of it automatically.

The definition/modification program uses a formatted screen (not in block mode - but under program control) which is set up with an indented hierarchy corresponding to that of the data dictionary. The formatted screen is traversed in a manner corresponding to the tree structure of the data dictionary. When changes are to be made, the user simply defines the path down the tree to the item to be changed, then enters the information to be modified.

Reporting utilities exist which produce a comprehensive list of any Data Dictionary's content, study encounter structure, as well as a coding manual to aid data coding (prior to data entry).

The translate/export module traverses any data dictionary, produces a sequential version of it and ships this to the IBM/370 where it's used by many IBM resident DBMS interfaces (as well as the batch editor).

## DATA MANAGEMENT SUBSYSTEM

The Data Management subsystem includes the Masterfile Tracking control file maintenance/report utilities, masterfile movement utilities and an IMAGE schema writer/loader. The entire system is controlled by a top level program in the Data Management system - called the Utility controller - which runs as a very high level language interpreter.

The Utility controller program of the subsystem accepts English like commands, parses them, and takes action on them by calling one of many programs in the system (using the MAIL intrinsics to pass information to the called programs).

The Masterfile Tracking control file contains information on the primary residence of all masterfiles, predicted number of records in the masterfile, and a variety of other accounting/security types of information. In addition, information on the existence of an IMAGE data base, or any other IBM resident DBMS, and the date that it was last loaded, is contained in this file. This information is very useful, not only to tell how current a data base is, but also, whether it already exists, and should be purged before being created(recreated).

Masterfile movement utilities are also available as a part of the Utility controller. The command MOVE HPD<masterfile> TO OS<dsname> is sufficient to prompt the system to automatically submit a job to move the masterfile named to an IBM/370 resident data set(building it if necessary). The reverse command is possible, and in fact, movement between any of 4 logical origins/destinations is possible. The jobs to move the masterfiles in any direction are submitted automatically to MRJE/3000.

The system includes interfaces to many IBM/370 resident DBMS packages, and to the IMAGE/3000 system. The IBM resident DBMS interfaces all use the Data Dictionary to create the equivalent of an IMAGE schema, then later in the same job, actually load the masterfile desired into the DBMS. The IMAGE/3000 interface submits a background job to run 2 programs (with standalone sorts between) which write a Schema, then actually load the data into the created database. The details of this set of programs are covered below.

## IMAGE/3000 SCHEMA WRITER/LOADER

The IMAGE Schema Writer is an interface program which reads the data dictionary, and primarily constructs appropriate schema definition statements from the data FIELD types. The procedure will create a schema containing all UNITs of a study or only selected UNITs depending the request made to the front end program (run by Utility controller).

Total output from the program includes the completed schema ready for processing by DBSCHEMA.PUB.SYS, a detail load map, and a summary or data set load map. The schema is produced by building two intermediate files - one for the SET part and one for the ITEMS part of the schema. As the data dictionary is traversed - accumulating all FIELDs belonging to each UNIT, the schema ITEM part for each field is written (knowing what type the FIELD is, its size, etc.). At the same time, a posting to the SET part is made for the FIELD. Each Data Dictionary UNIT is considered to be equivalent to an IMAGE Data Set, so whenever a new UNIT is started in the traversal operation, the appropriate SET definition is written, followed by its FIELD or item entries. The buffer position for the Loader procedure is also calculated for each field in each data set. The detail load map file generated by the schema writer program consists of a record for each Field in the data dictionary containing the field's input buffer address, the IMAGE data type, the field's parent Record Id, starting column, length, and decoding information. This information is sufficient for later posting of the data to the data base. The summary load map produced contains an entry for each data set being created, with the number of fields in the data set, the total buffer size (used for acquiring the proper size extra data segment in the Loader procedure), and the name and sequence number of the data set.

After the Schema writer produces the Schema, Detail load map and summary or data set load map, the DBSCHEMA.PUB.SYS program is invoked to process the new schema(FILE equation being issued by the schema writer). After the root file is created, the DBUTIL.PUB.SYS program is executed, to create the new data base. Finally the detail load map is sorted by Record id, then start column and the masterfile is sorted by patient id, then record date, then record id. The Loader procedure is now ready to run.

The Loader program reads the sorted Detail Load map, and constructs a tree structure containing all information to convert/decode from the masterfile records into a logical buffer to be passed to the IMAGE data base. The procedure also reads the Data Set load map, and creates an extra data segment of proper size to act as data accumulation areas for each of the Data Sets. The program then begins reading the masterfile and converting/moving the data to the appropriate extra data segment location. When a logical breakpoint in the data occurs(change of patient, or change of date), all extra data segments which have been posted, are accessed, posting their contents to the IMAGE data base. Various safety checks are built in to prevent overwriting of the extra data segment buffers, i.e., if a record member of a buffer has already been posted, and it's about to be posted again, the buffer is cleared first (this should not happen normally due to the UNIT structure of the Data Dictionary).

During the loading process, blank or missing data is noted, by posting a bit in a MISSING field(a field which is automatically generated for each data set) which corresponds to the sequence number of the field in the data set. This MISSING bit is to be used by various application programs accessing the data base to ensure proper statistical treatment of missing data. Each UNIT occurrence is also categorized into one of the ENCOUNTERs of the Data Dictionary, with the categorization being also posted to an ENCOUNTER field in each data set(this aids in data retrieval later). Data which is to be decoded for reporting purposes, is decoded during the load operation - the Decode table field name and field value (or encode) are concatenated, then used as a KSAM key into the decode file - retrieving the text to be inserted into the data base.

HEWLETT PACKARD

# HP VIEW/3000

A SOURCE DATA ENTRY SYSTEM

Presented to:

HP GENERAL SYSTEMS USERS GROUP
7TH INTERNATIONAL MEETING

October 30 - November 3, 1978

Denver Hilton
Denver, Colorado

By:

*Jutta Kernke*
Product Manager
General Systems Division
Hewlett-Packard Company

# CONTENTS

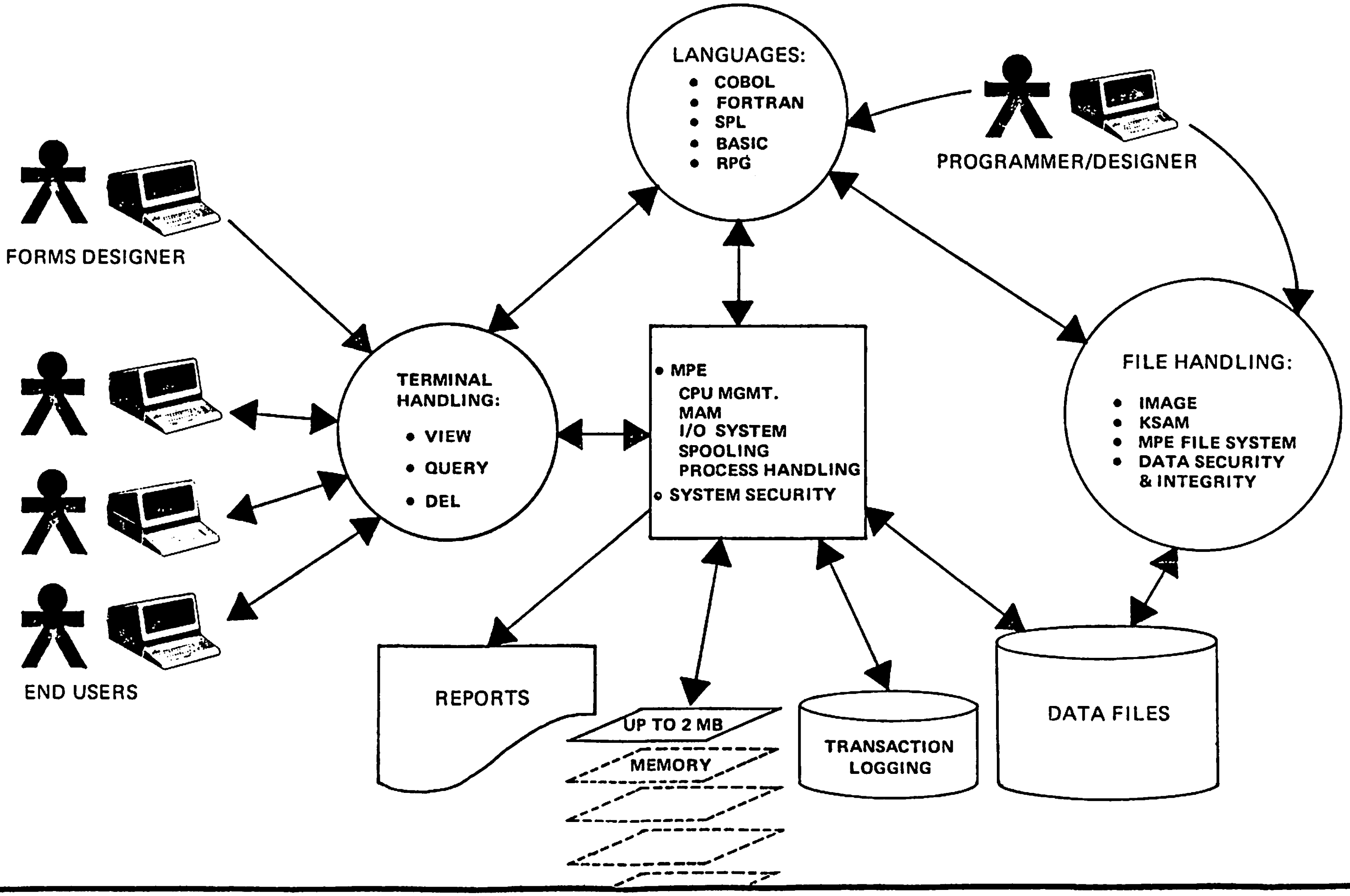
## VIEW/3000

| | PAGE |
|---|---|
| AVAILABLE RESOURCES ON THE HP3000 | 2 |
| VIEW PRODUCT DESCRIPTION | 3 |
| COMPARE VIEW VS DEL | 4 |
| STAND-ALONE SOURCE DATA ENTRY | 5 |
| FRONT-END TO TRANSACTION PROCESSING | 6 |
| MAIN COMPONENTS | 7 |
| FORMS DESIGN | 8 |
| COMPREHENSIVE EDITING | 13 |
| SOURCE DATA ENTRY | 17 |
| PROGRAM INTERFACE | 21 |
| REFORMATTING CAPABILITY | 25 |
| SUMMARY | 32 |

# VIEW/3000

HEWLETT hp PACKARD

## AVAILABLE RESOURCES FOR A TRANSACTION PROCESSING SYSTEM ON THE HP3000



## PRODUCT DESCRIPTION

. . . . . . . . . . . . . . . . . .

# VIEW/3000

IS A SOFTWARE PACKAGE THAT PROVIDES TWO MAJOR CAPABILITIES:

- A SELF-CONTAINED DATA ENTRY SYSTEM ... can be implemented with no programming effort, and a FORMS SPECIFICATION UTILITY for drawing forms, defining simple and advanced editing.
- A FRONT-END TO TRANSACTION PROCESSING APPLICATIONS ... using the FORMSPEC utility and an extensive library of high-level procedures to facilitate terminal-oriented and user-written applications.

HEWLETT PACKARD

## HOW DOES VIEW COMPARE TO DEL???

- DEL AND VIEW ARE "NOT" COMPATIBLE
- VIEW IS A CAPABILITIES ENHANCEMENT
- VIEW IS NOT A PERFORMANCE ENHANCEMENT
- VIEW SHOULD BE FOCUSED ON AS A "NEW" DATA ENTRY PRODUCT "NOT" AS AN ENHANCEMENT OF DEL.

## VIEW/3000

<u>"STAND-ALONE SOURCE DATA ENTRY PACKAGE"</u>

- NO PROGRAMMING EFFORT
- FILL-IN-THE-BLANKS DESIGN
- FREE FORM DRAWING
- START ENTERING DATA IMMEDIATELY
- REFORMATTING CAPABILITY
- PRE-SET FUNCTION KEYS

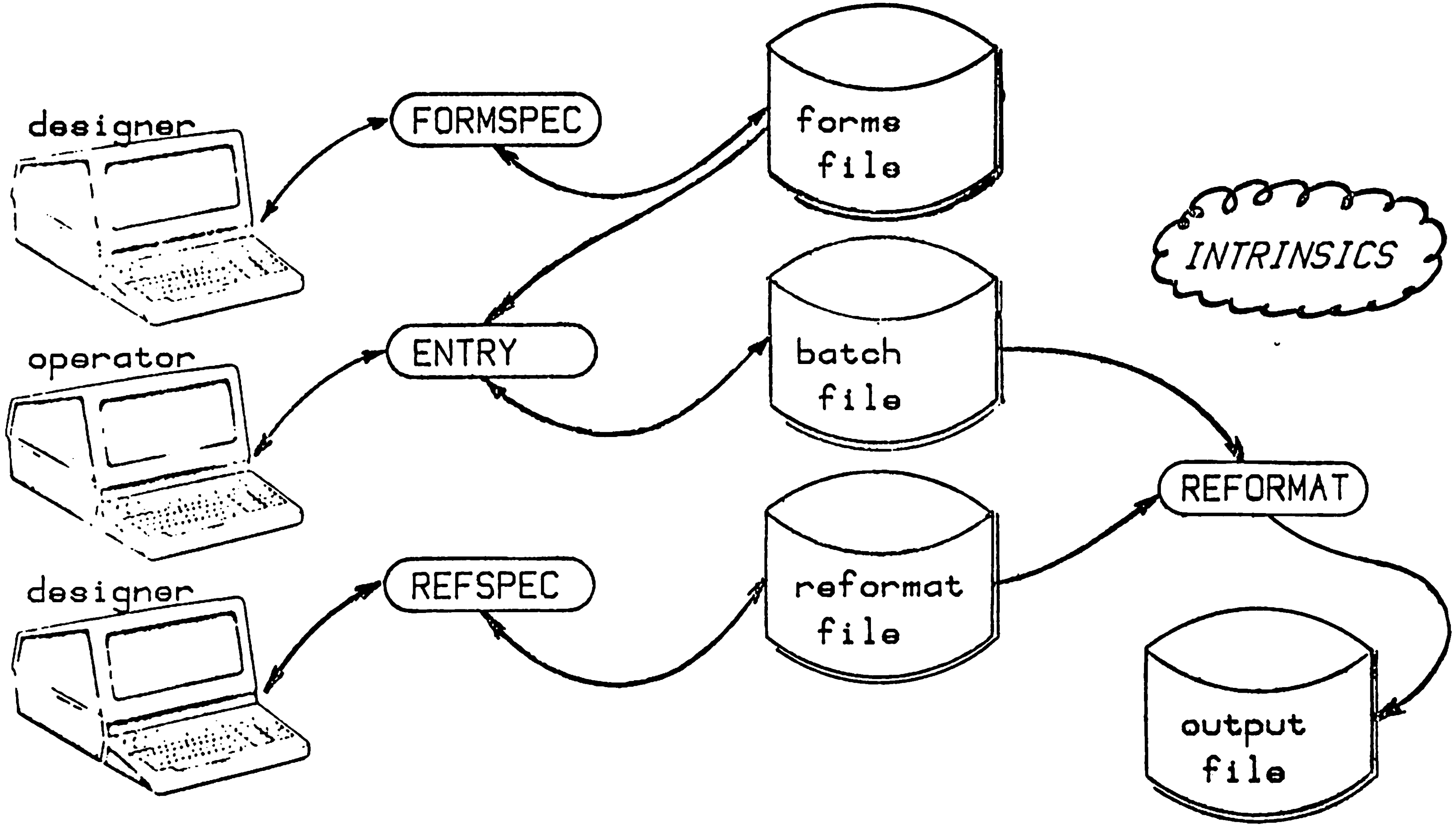
## VIEW/3000

"FRONT-END TO TRANSACTION PROCESSING"

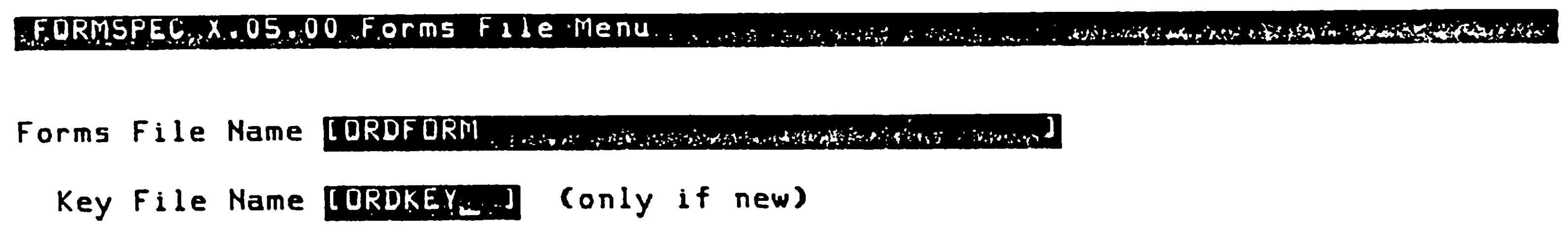
- COBOL, RPG, BASIC, FORTRAN AND SPL INTERFACE
- SIMPLE DESIGN, TEST, IMPLEMENTATION, AND MAINTENANCE
- PROGRAMMER PRODUCTIVITY INCREASES

HEWLETT hp PACKARD

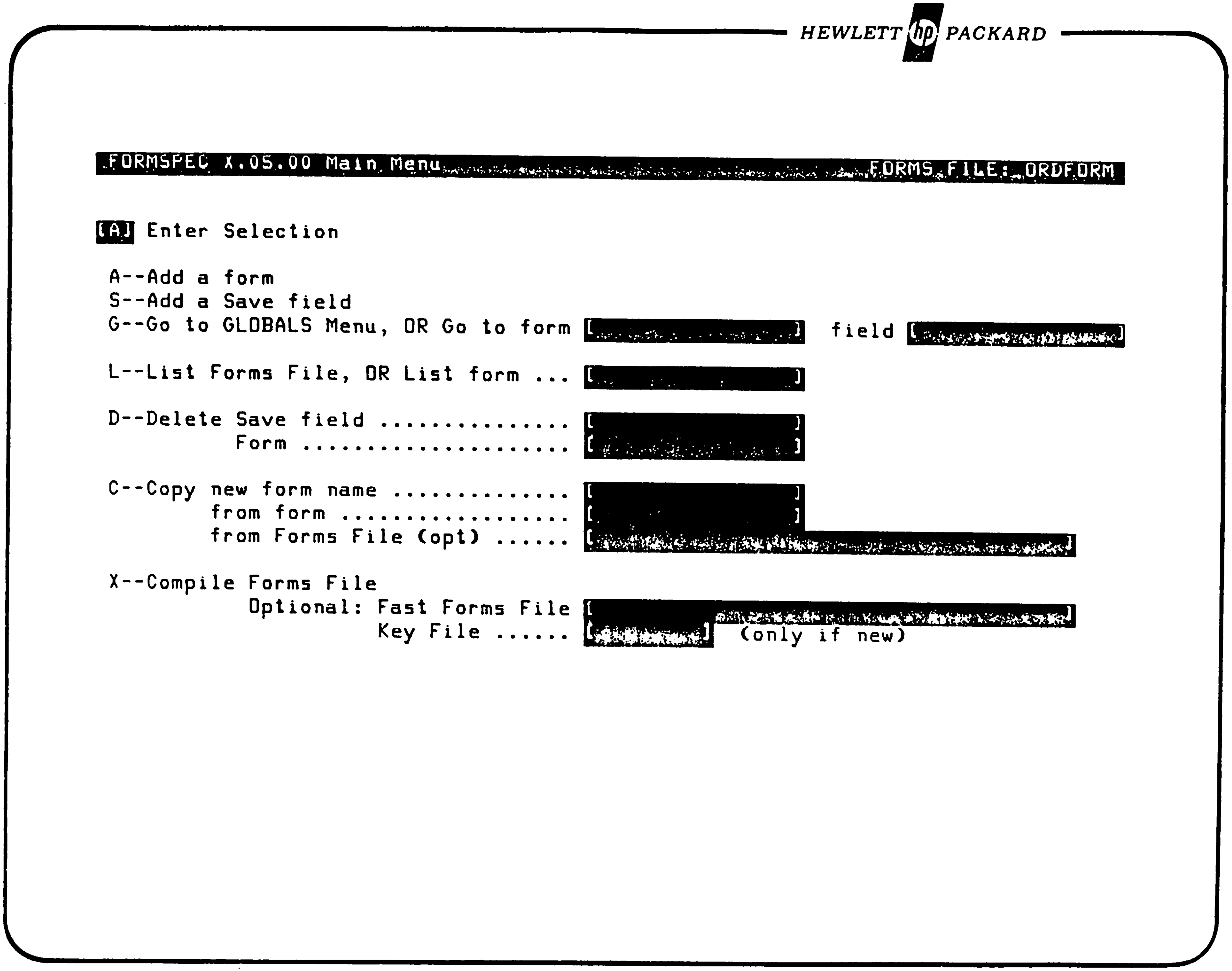
VIEW/3000 - main components



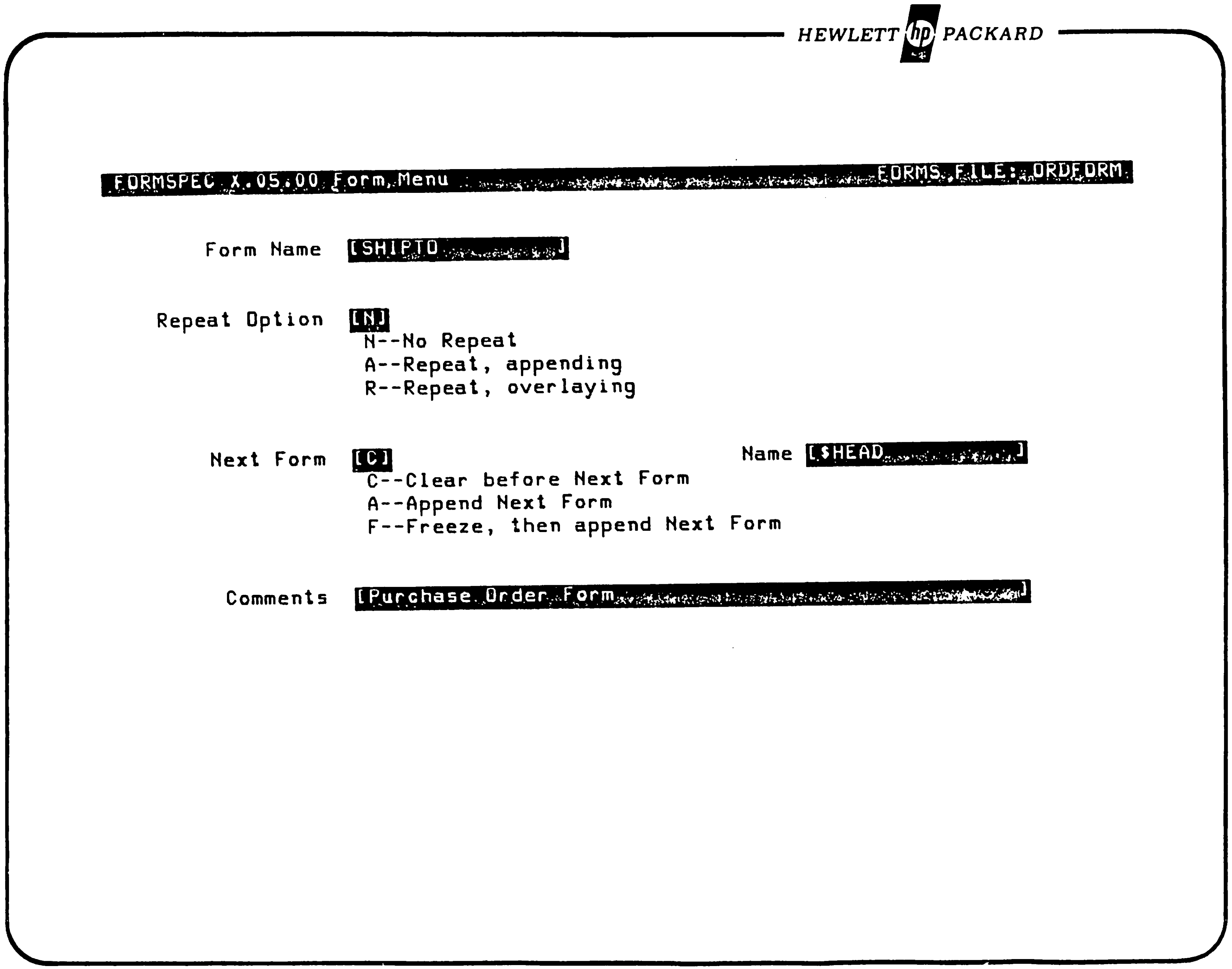
HEWLETT PACKARD



```
FORMSPEC X.05.00 Forms File Menu

Forms File Name [ORDFORM                    ]

  Key File Name [ORDKEY  ]  (only if new)
```



```
 FORMSPEC X.05.00 Main Menu                                  FORMS FILE: ORDFORM

[A] Enter Selection

 A--Add a form
 S--Add a Save field
 G--Go to GLOBALS Menu, OR Go to form [                ]  field [                ]

 L--List Forms File, OR List form ... [                ]

 D--Delete Save field ............... [                ]
           Form ..................... [                ]

 C--Copy new form name .............. [                ]
         from form .................. [                ]
         from Forms File (opt) ...... [                                    ]

 X--Compile Forms File
            Optional: Fast Forms File [                                    ]
                      Key File ...... [        ]  (only if new)
```

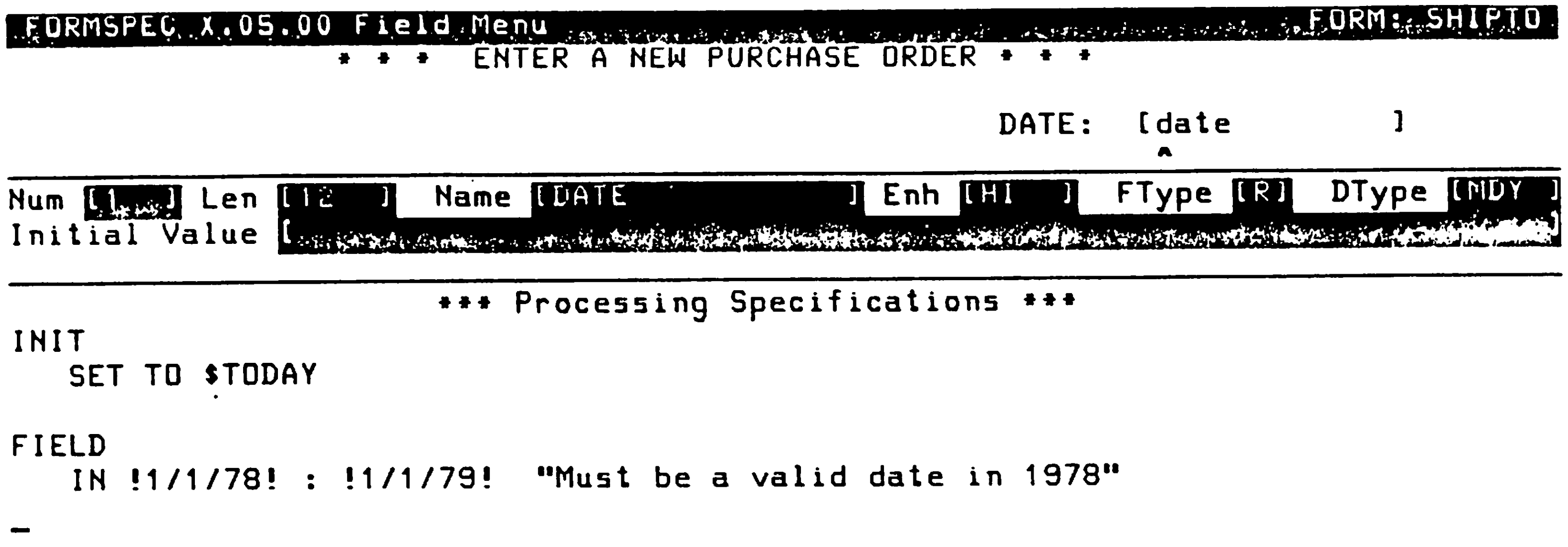


```
 FORMSPEC X.05.00 Form Menu                                   FORMS FILE: ORDFORM

          Form Name  [SHIPTO          ]

      Repeat Option  [N]
                      N--No Repeat
                      A--Repeat, appending
                      R--Repeat, overlaying

          Next Form  [C]                       Name [$HEAD           ]
                      C--Clear before Next Form
                      A--Append Next Form
                      F--Freeze, then append Next Form

           Comments  [Purchase Order Form                                    ]
```

HEWLETT PACKARD

```
             * * *  ENTER A NEW PURCHASE ORDER * * *

                                           DATE:  [date        ]

          SHIP TO:  [name                    ]
                    [addr1                   ]
                    [addr2                   ]  ZIP [zip  ]



    ORDER #          QTY.           PART #                  PRICE
  [ordernum ]       [qty ]       [partnum     ]          [price     ]
```

HEWLETT PACKARD



```
FORMSPEC X.05.00 Field Menu                                                      FORM: SHIPTO
                      * * *  ENTER A NEW PURCHASE ORDER * * *

                                                          DATE:  [date          ]
                                                                  ^
Num [1  ] Len [12  ]  Name [DATE              ] Enh [HI  ]  FType [R]  DType [MDY ]
Initial Value [                                                                     ]

                              *** Processing Specifications ***
INIT
   SET TO $TODAY

FIELD
   IN !1/1/78! : !1/1/79!  "Must be a valid date in 1978"

_
```

# Comprehensive Data Editing & Formatting

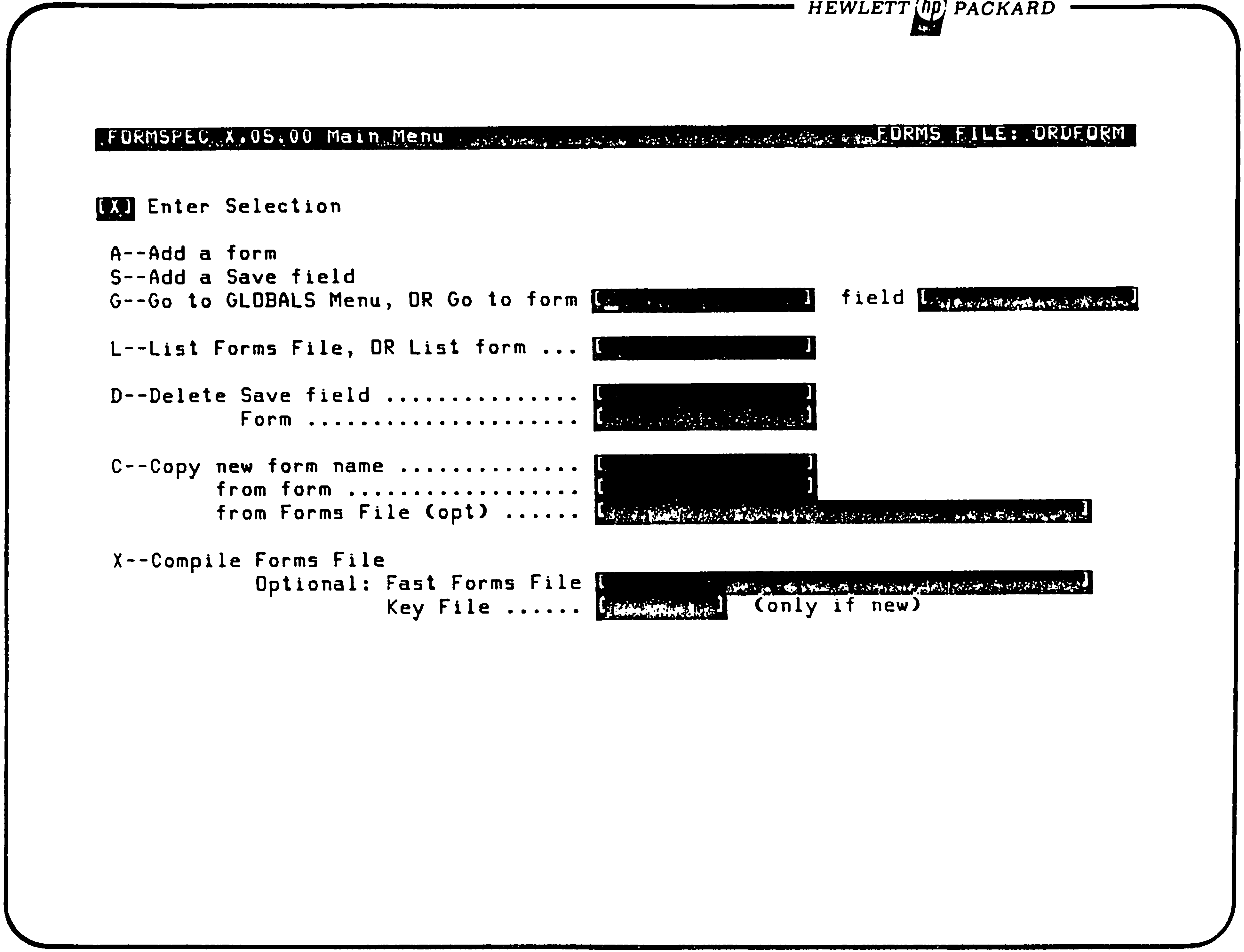
High-level language for editing/formatting specification:

- Length check
- Range check
- Table check
- Comparison for < = >
- Justify
- Fill
- Arithmetic expressions
- Pattern match
- Field moves
- Logical/conditional (if-then-else)
- Modulus 10/11 check digits
- Custom error messages

## PHASES

| | |
|---|---|
| INIT | PERFORMS STATEMENTS DURING THE INITIALIZATION PHASE. FOR EXAMPLE: SET TO 100.00 |
| FIELD | PERFORMS THE STATEMENTS DURING THE FIELD PHASE UNTIL DATA IS ERROR FREE. FOR EXAMPLE: IN 20:200; IF IN 100:200 THEN JUSTIFY RIGHT |
| FINISH | PERFORMS STATEMENTS DURING THE WRAP-UP. FOR EXAMPLE: SET TO SFIELD1; CHANGE NFROM TO $END |

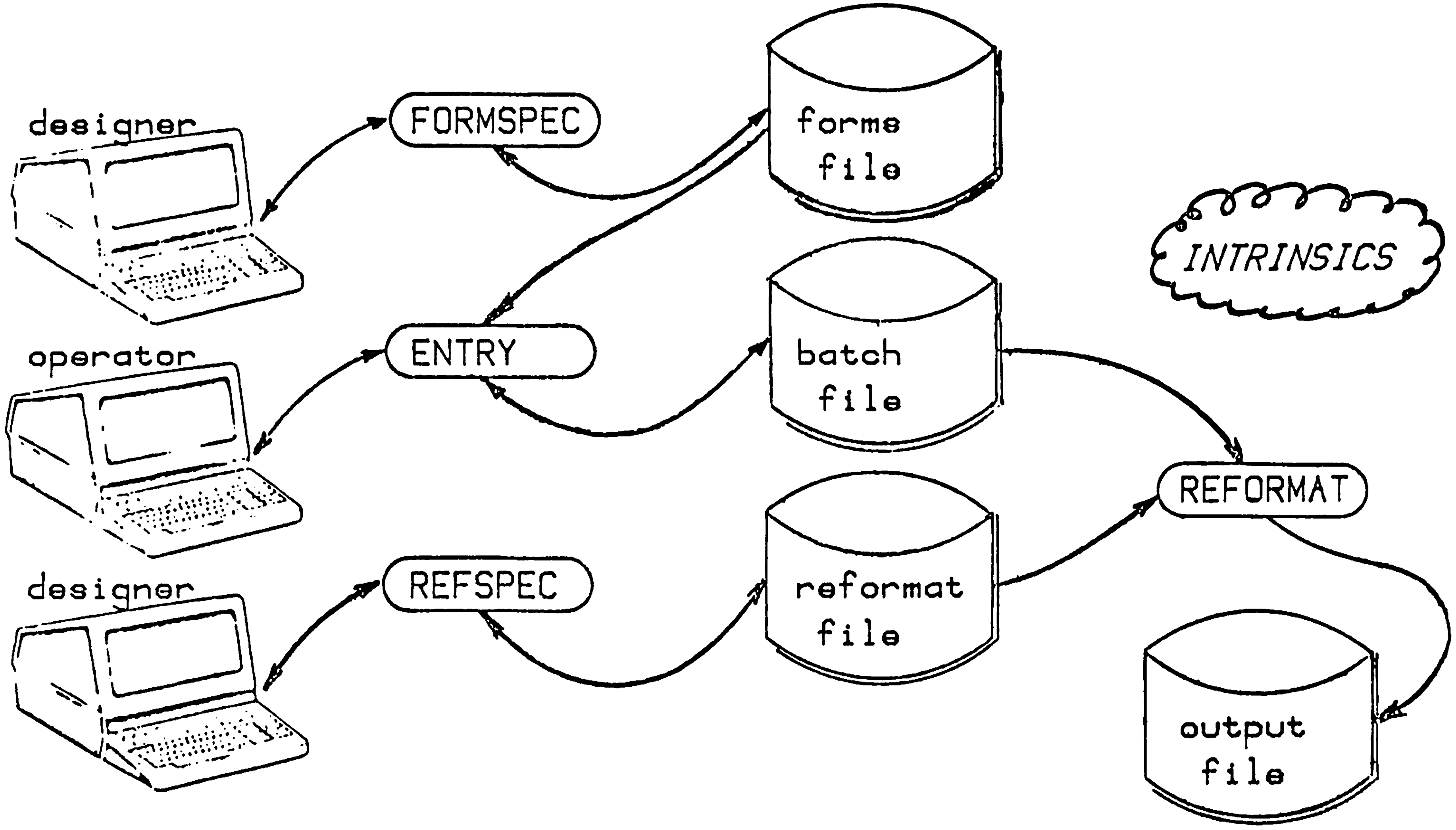
HEWLETT PACKARD



```
 FORMSPEC X.05.00 Main Menu                                    FORMS FILE: ORDFORM

[X] Enter Selection

 A--Add a form
 S--Add a Save field
 G--Go to GLOBALS Menu, OR Go to form [                ]  field [                ]

 L--List Forms File, OR List form ... [                ]

 D--Delete Save field ............... [                ]
           Form ..................... [                ]

 C--Copy new form name .............. [                ]
         from form .................. [                ]
         from Forms File (opt) ...... [                                     ]

 X--Compile Forms File
          Optional: Fast Forms File [                                     ]
                    Key File ...... [         ]  (only if new)
```

HEWLETT PACKARD

## VIEW/3000 - main components



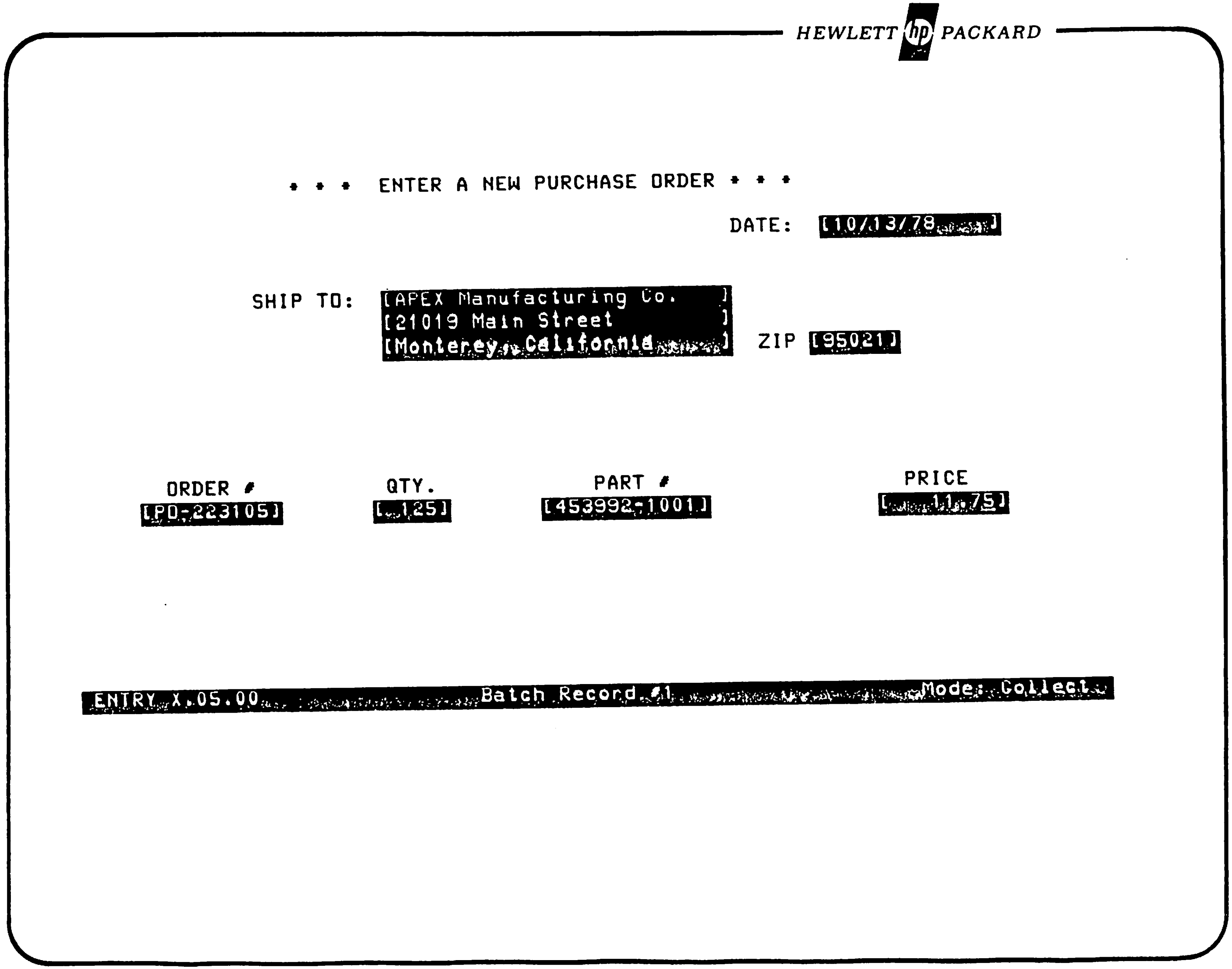
## Friendly, Ready-To-Run Data Entry Program

- Provides standard data entry features via special function keys
  - "Browse" through already entered data
  - Modify and/or delete already entered data
  - Resume to prior point of data collection
- Allows immediate execution of data entry formats and editing without use of compilers
- Stores data records in MPE sequential files

```
:RUN ENTRY.PUB.SYS

HP32209X.05.00 ENTRY (C) HEWLETT-PACKARD CO. 1978

 Enter Forms File name and press RETURN: ORDFORM
 Enter Batch File name and press RETURN: BATCHFL

_
```

HEWLETT PACKARD

```
          * * *  ENTER A NEW PURCHASE ORDER * * *
                                              DATE:  [date        ]

        SHIP TO:  [name                         ]
                  [addr1                        ]
                  [addr2                        ]  ZIP [zip   ]



     ORDER #         QTY.          PART #                    PRICE
   [ordernum ]      [qty ]       [partnum     ]            [price     ]
```



```
                                                   HEWLETT hp PACKARD

            * * *  ENTER A NEW PURCHASE ORDER * * *
                                        DATE:  [10/13/78      ]

        SHIP TO:  [APEX Manufacturing Co.     ]
                  [21019 Main Street          ]
                  [Monterey, California       ]  ZIP [95021]


    ORDER #          QTY.           PART #                 PRICE
  [PO-223105]      [ 125]      [453992-1001]            [   11.75]


 ENTRY X.05.00                  Batch Record #1                  Mode: Collect
```

HEWLETT PACKARD

## USER-WRITTEN PROGRAMS

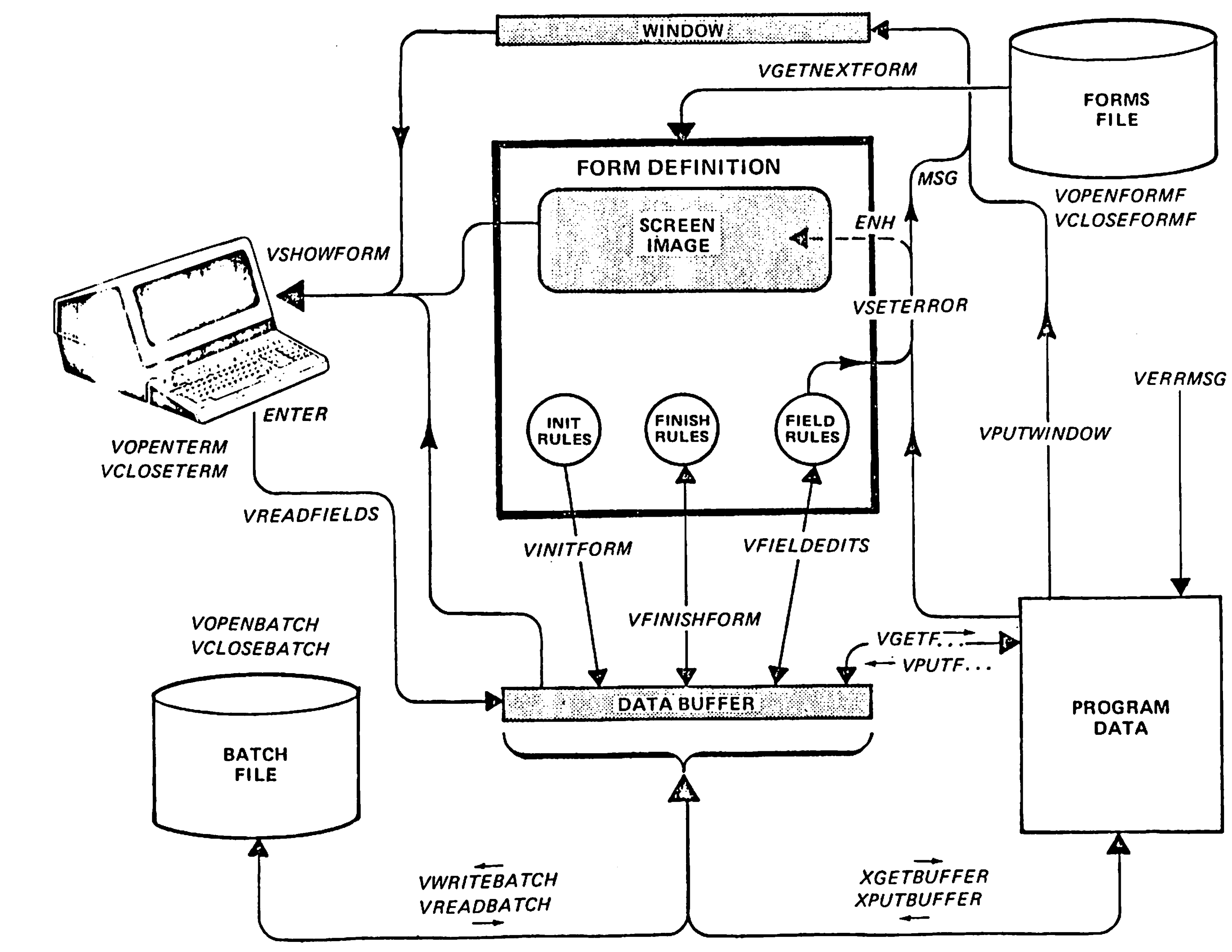
1 INTERFACES WITH 5 LANGUAGES: RPG, COBOL, FORTRAN, BASIC, SPL.

2 MANAGES INTERFACE BETWEEN PROGRAM AND TERMINAL, FORMS FILE, AND ENTERED DATA.

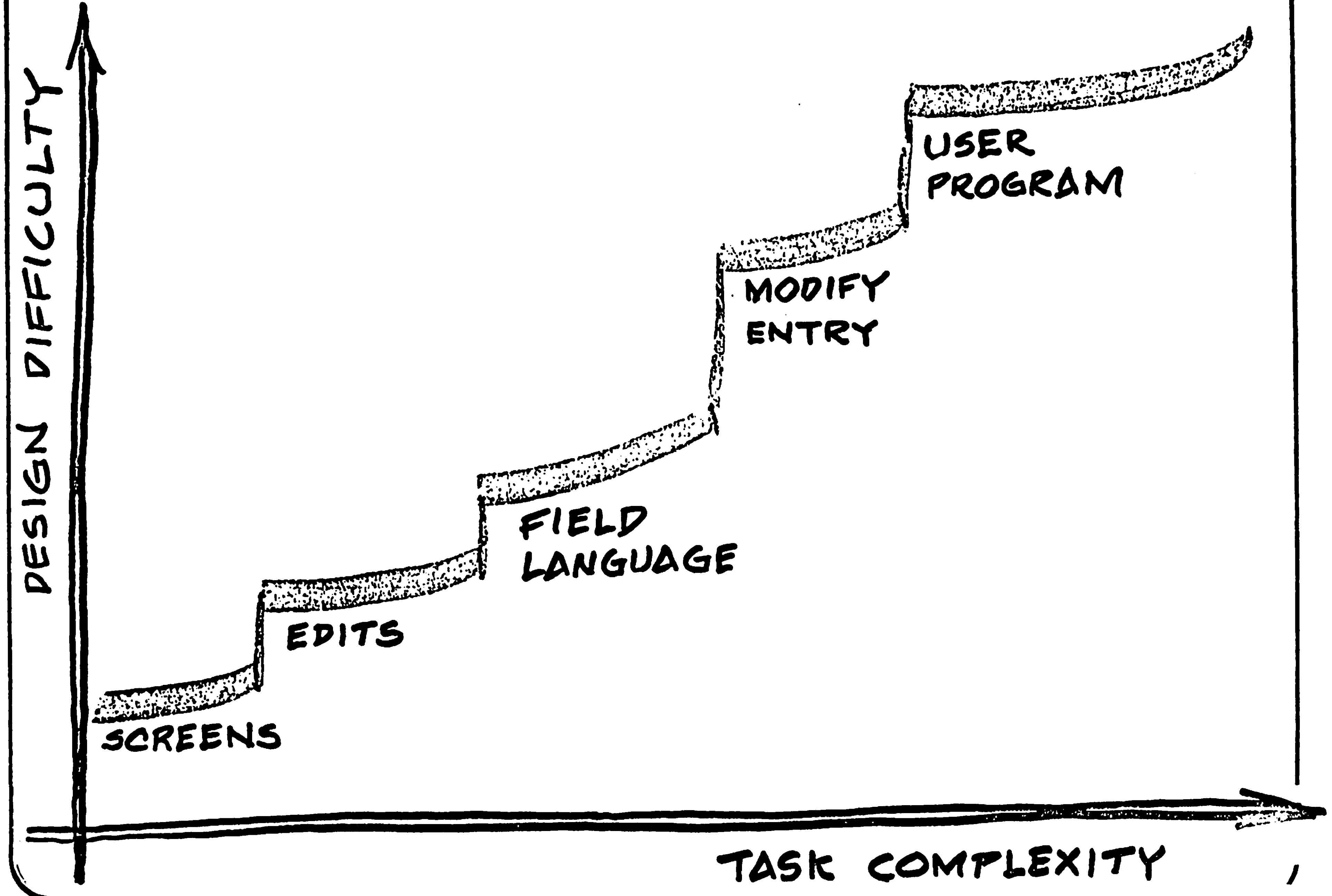
3 PHYSICAL ORDER OR LOCATION OF THE DATA FIELDS ON THE SCREEN IS IRRELEVANT TO THE APPLICATION PROGRAM.

HEWLETT PACKARD

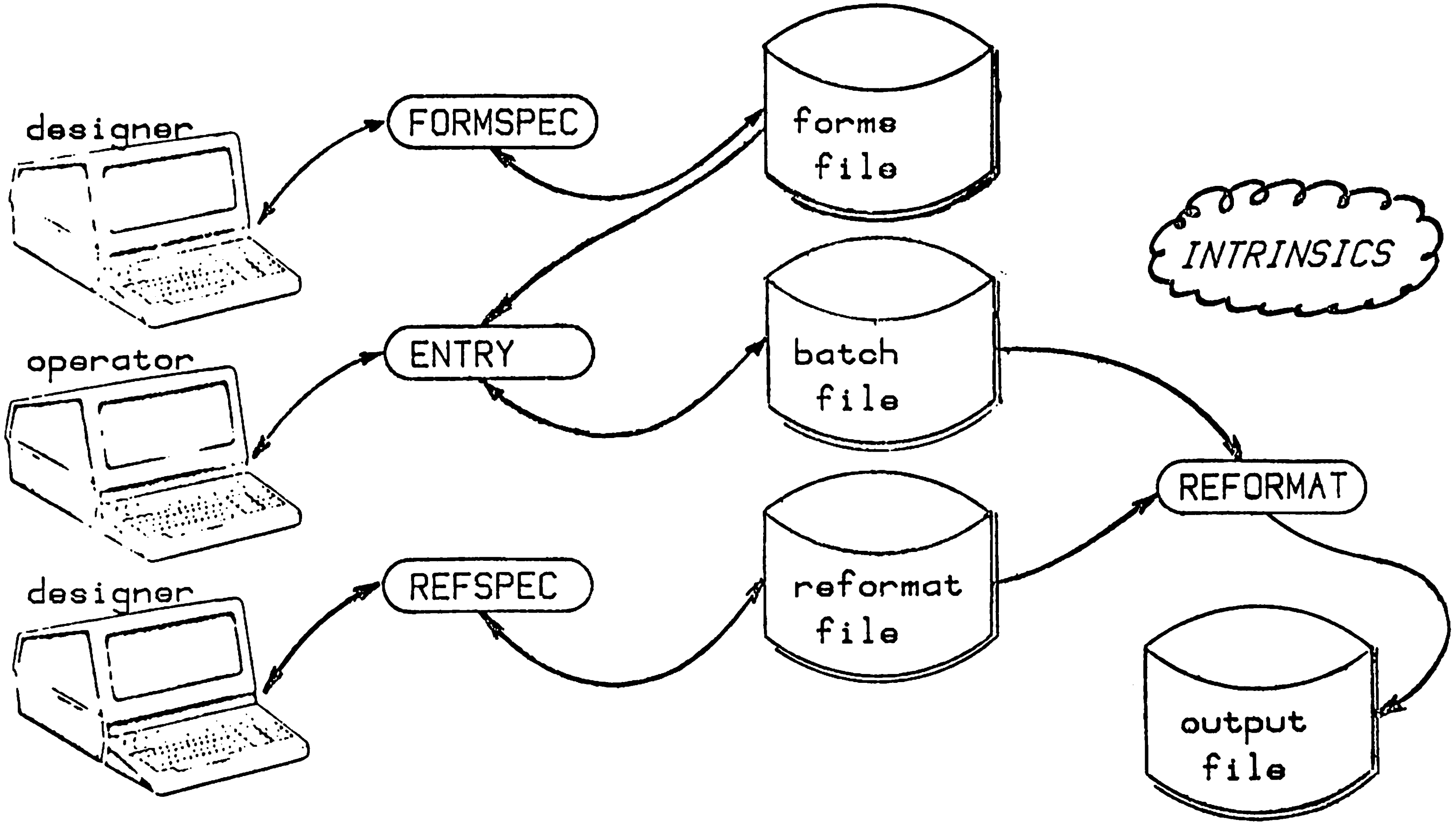
VIEW/3000 INTRINSICS



HEWLETT PACKARD



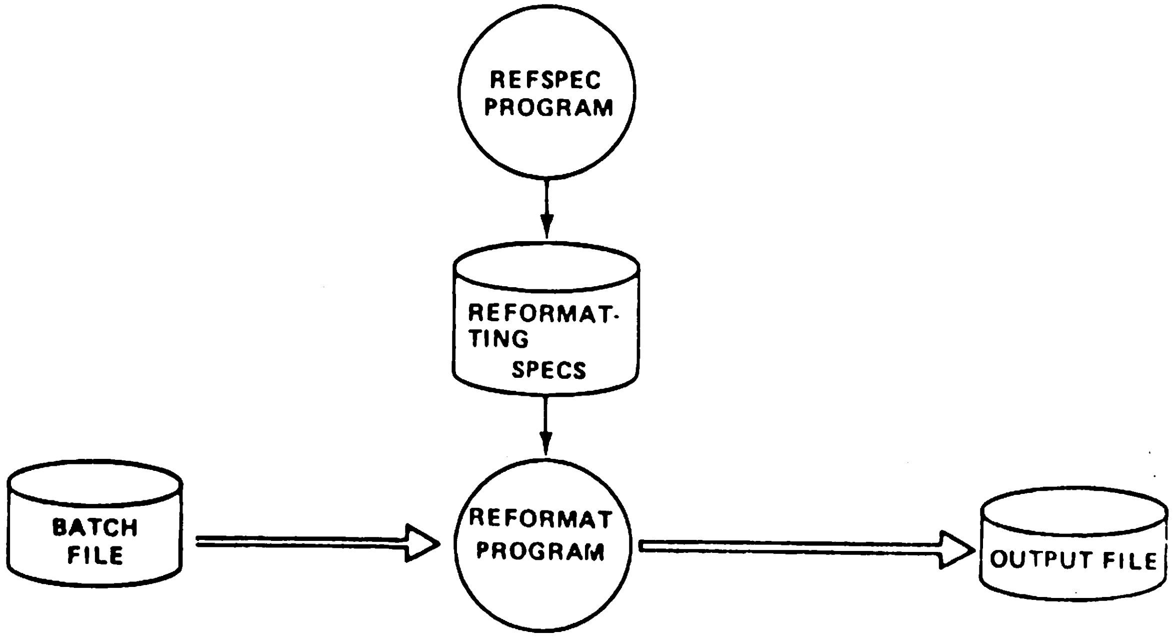
VIEW/3000 - main components

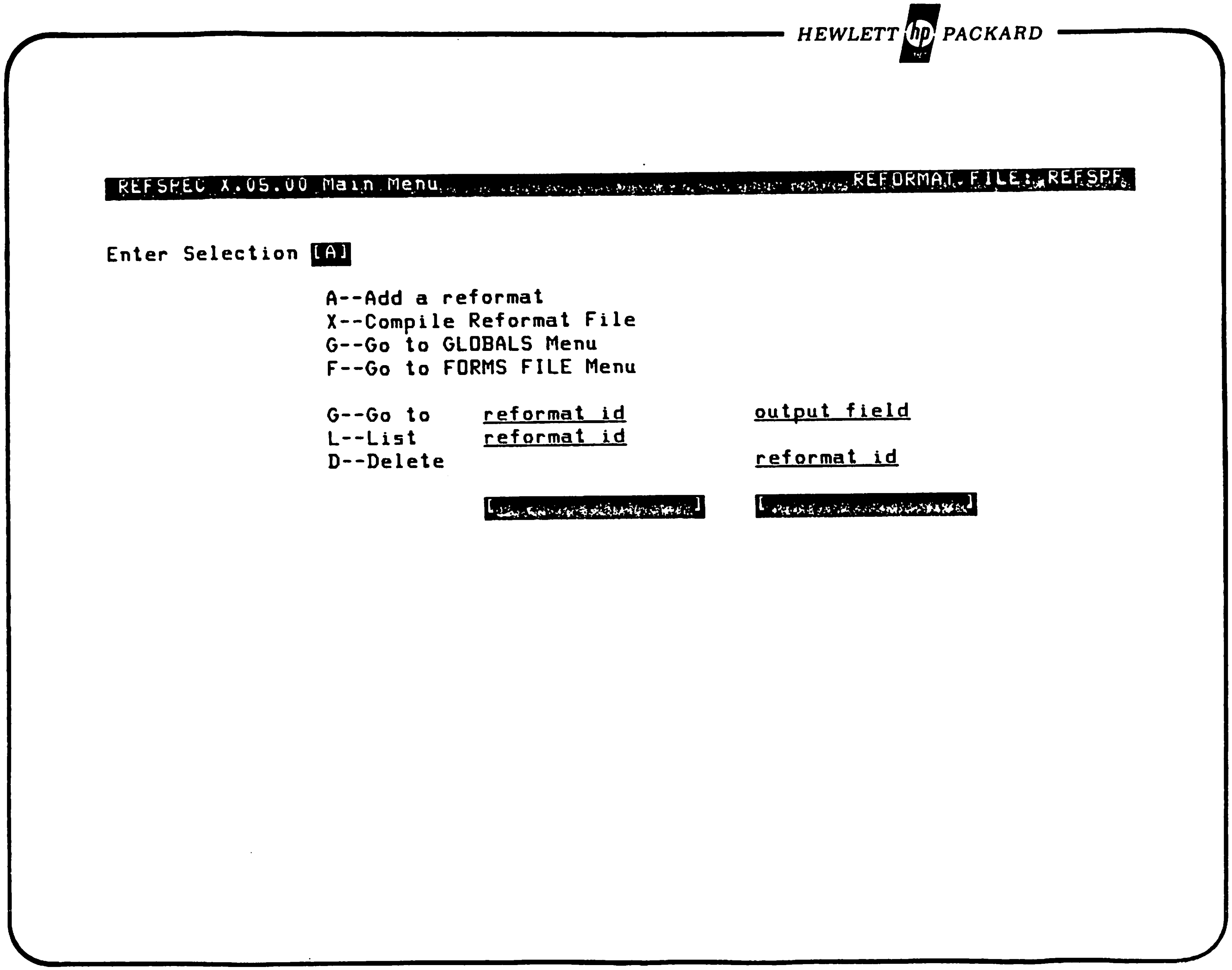


## Flexible Reformatting Utility

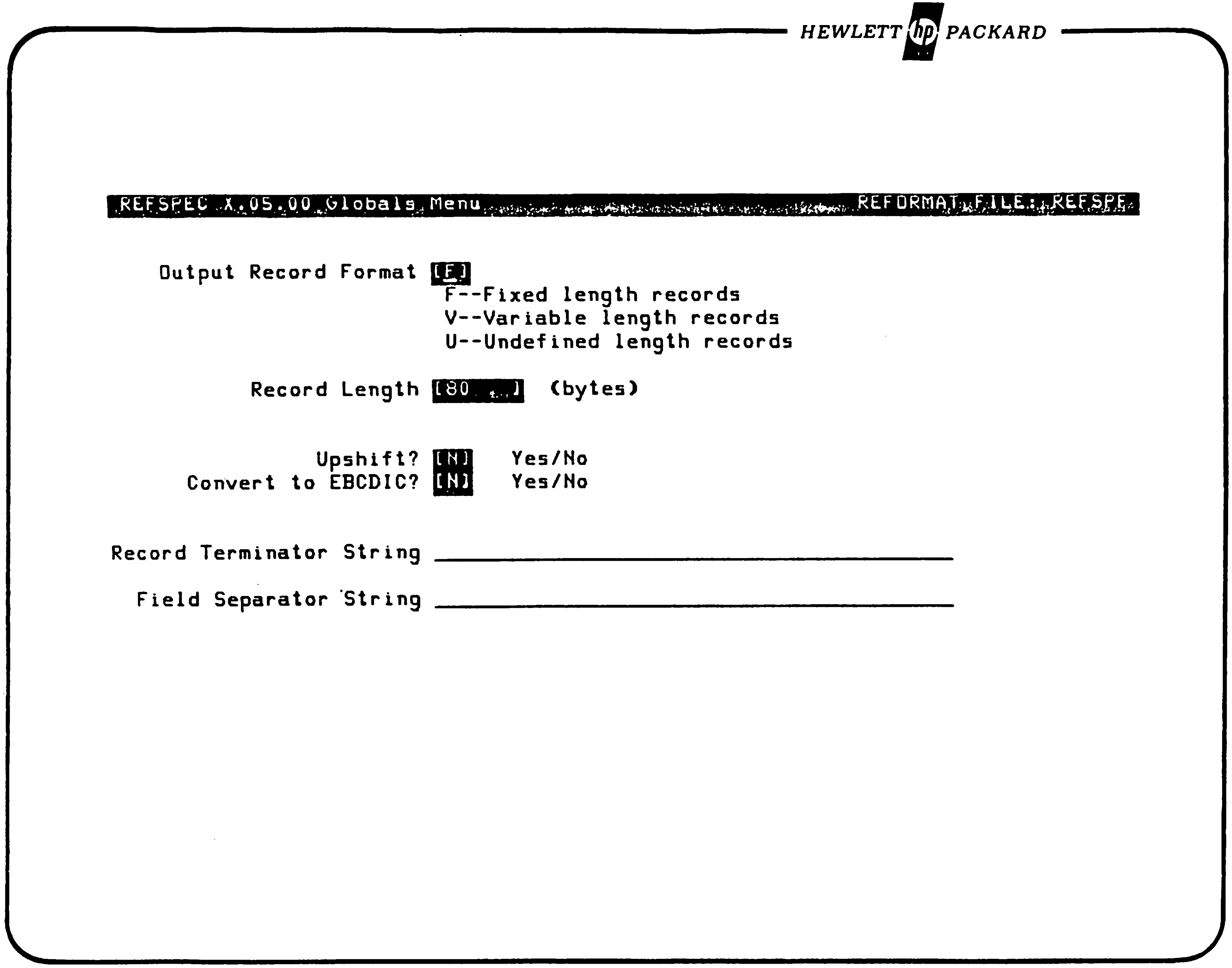
A separate batch program to:

- Meet data format requirements of existing programs
- Perform batch-type formatting
  - Combine data from several forms into a single record
  - Split data from a single form into two or more records
  - Rearrange data within a record
  - Adjust data within a field (e.g., "justify" or "fill")

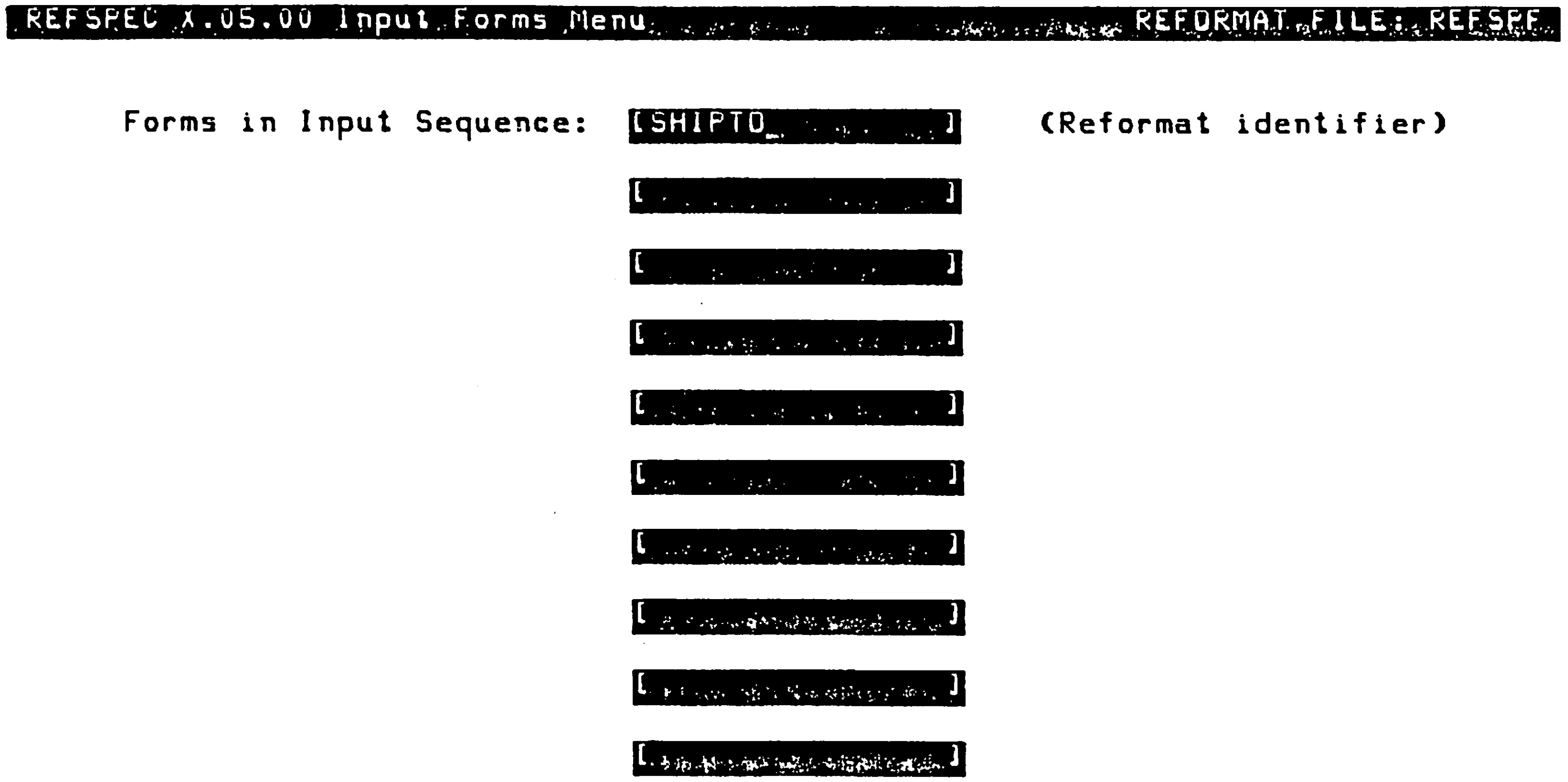



HEWLETT PACKARD

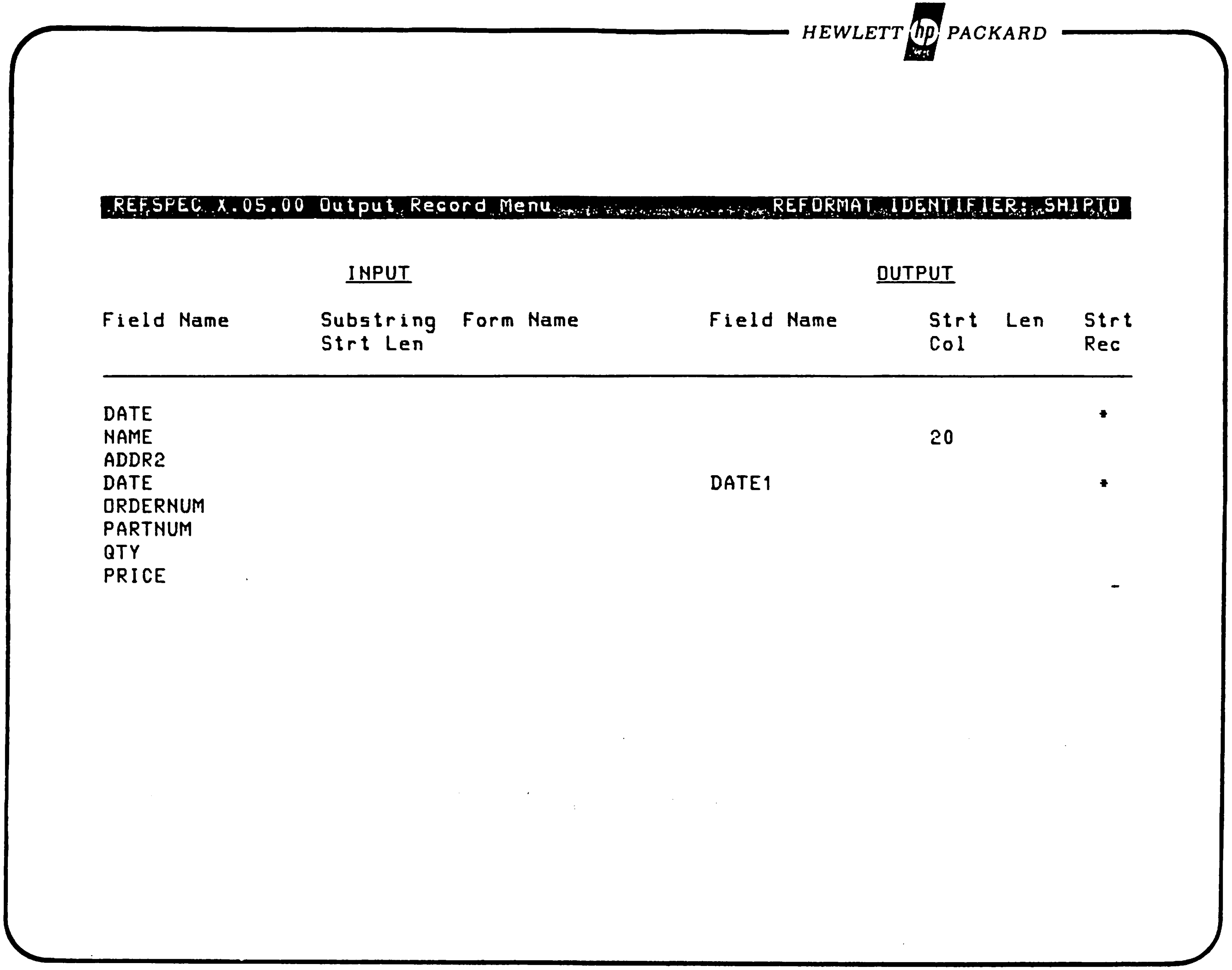
```
REFSPEC X.05.00 Main Menu                                    REFORMAT FILE: REFSPF

Enter Selection [A]

                    A--Add a reformat
                    X--Compile Reformat File
                    G--Go to GLOBALS Menu
                    F--Go to FORMS FILE Menu

                    G--Go to    reformat id          output field
                    L--List     reformat id
                    D--Delete                        reformat id

                                [                ]   [                ]
```



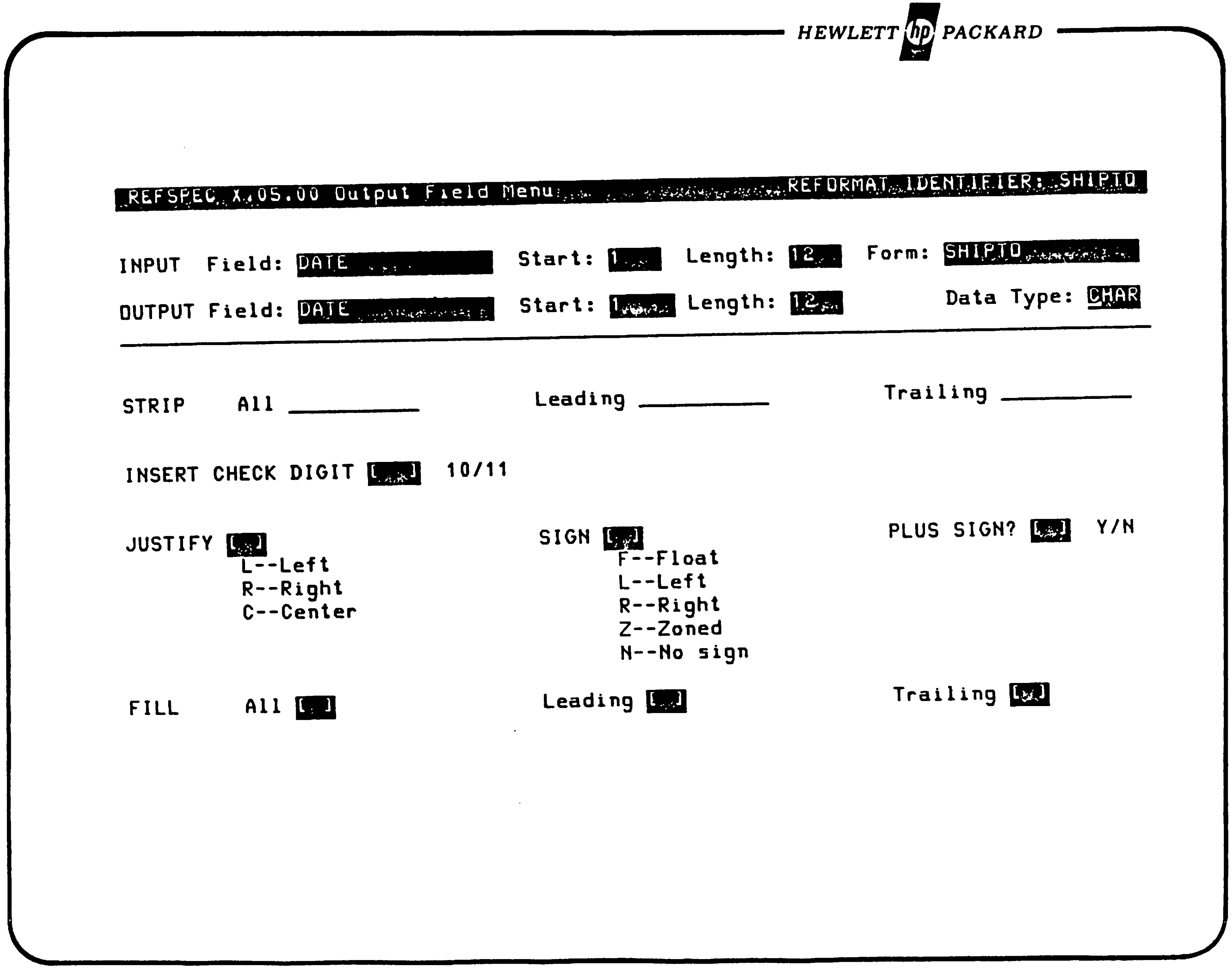
```
REFSPEC X.05.00 Globals Menu                                  REFORMAT FILE: REFSPE

        Output Record Format [F]
                              F--Fixed length records
                              V--Variable length records
                              U--Undefined length records

               Record Length [80   ]  (bytes)


                   Upshift? [N]    Yes/No
         Convert to EBCDIC? [N]    Yes/No


   Record Terminator String ________________________________________

     Field Separator String ________________________________________
```



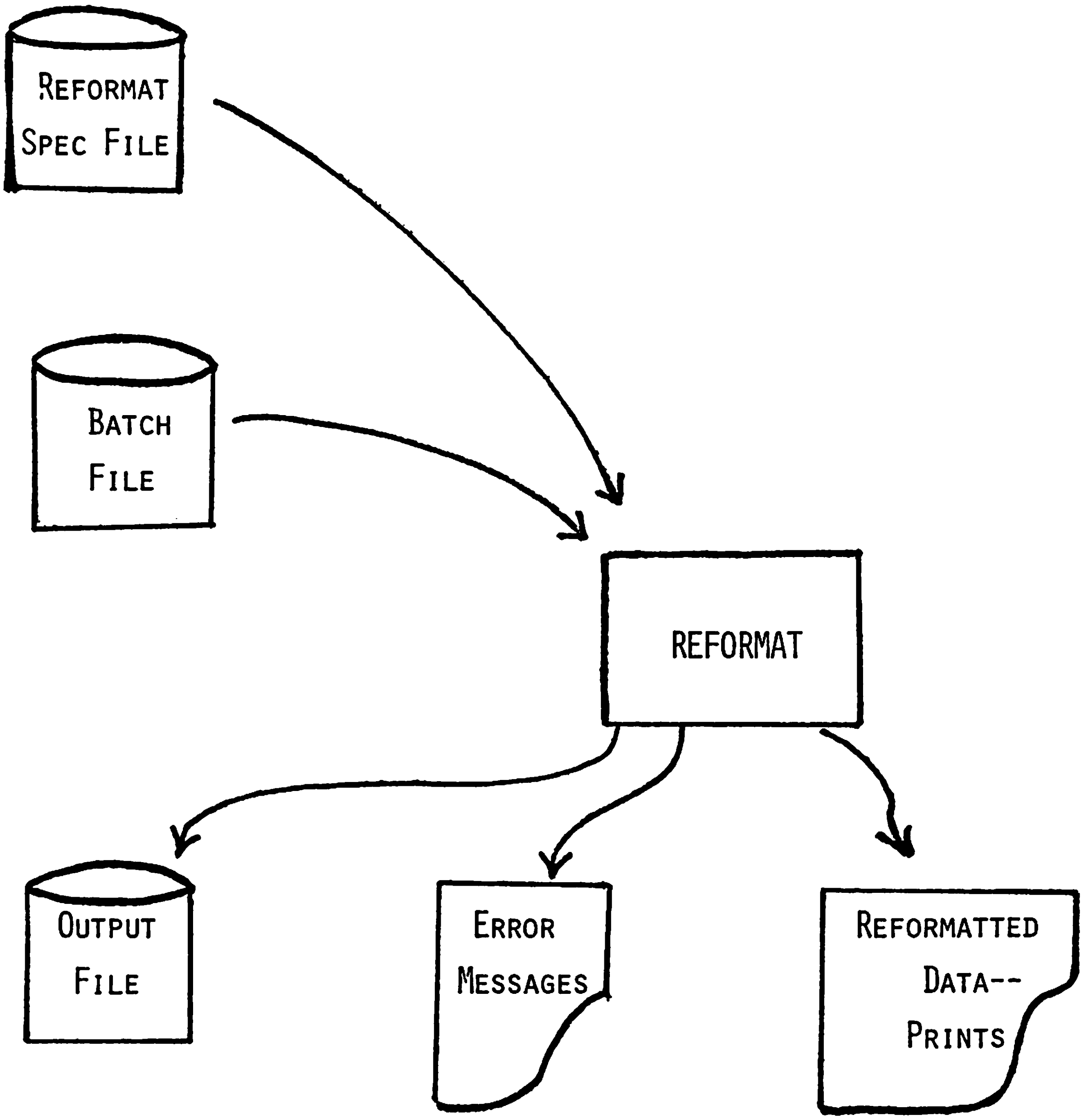
```
 REFSPEC X.05.00 Input Forms Menu                              REFORMAT FILE: REFSPF

        Forms in Input Sequence:  [SHIPTO_          ]   (Reformat identifier)
                                  [                 ]
                                  [                 ]
                                  [                 ]
                                  [                 ]
                                  [                 ]
                                  [                 ]
                                  [                 ]
                                  [                 ]
                                  [                 ]
```



```
 REFSPEC X.05.00 Output Record Menu                      REFORMAT IDENTIFIER: SHIPTO

                    INPUT                                          OUTPUT

Field Name         Substring  Form Name              Field Name          Strt   Len    Strt
                   Strt Len                                              Col           Rec
--------------------------------------------------------------------------------------------

DATE                                                                                    *
NAME                                                                     20
ADDR2
DATE                                                 DATE1                              *
ORDERNUM
PARTNUM
QTY
PRICE
                                                                                         -
```



```
REFSPEC X.05.00 Output Field Menu                    REFORMAT IDENTIFIER: SHIPTO

INPUT  Field: DATE          Start: 1    Length: 12    Form: SHIPTO
OUTPUT Field: DATE          Start: 1    Length: 12         Data Type: CHAR
________________________________________________________________________________

STRIP    All ________         Leading ________          Trailing ________

INSERT CHECK DIGIT [ ]   10/11

JUSTIFY [ ]                   SIGN [ ]                  PLUS SIGN? [ ]   Y/N
         L--Left                   F--Float
         R--Right                  L--Left
         C--Center                 R--Right
                                   Z--Zoned
                                   N--No sign

FILL     All [ ]              Leading [ ]               Trailing [ ]
```


REFORMAT SPEC FILE
BATCH FILE
REFORMAT
OUTPUT FILE
ERROR MESSAGES
REFORMATTED DATA-- PRINTS

## Summary

■ **What is it?**

- Stand-alone source data entry
- Front-end to transaction processing systems

■ **Major features!**

- Simple forms design
- Comprehensive editing
- Data entry without programming
- Reformatting to meet existing requirements
- Adaptable language interface (BASIC, COBOL, FORTRAN, RPG, SPL)

# FACTORY DATA COLLECTION

by

LINDA SIENER
HEWLETT-PACKARD COMPANY
DATA SYSTEMS DIVISION

One of the most challenging aspects of manufacturing control is the collection of timely and accurate factory data. The quality of this data largely determines the effectiveness of today's modern control systems. Late or inaccurate data can only produce low quality information. Timely shop information is, therefore, essential to the control of manufacturing processes. This same information is also important to the accountability of direct labor records in the financial system.

Typically, the shop floor environment is the most difficult area from which to extract information. The factory workers are primarily concerned with manufacturing items and accurately reporting their work is incidental to their job. What is needed is a factory data collection system that can help make the data collected from the workers more accurate and timely and, at the same time, easily blend into the current operating methodology.

Advances in electronic technology have now made it possible to cost-effectively introduce factory data collection terminals directly into the manufacturing area. No longer does data have to be written down, keypunched, checked for accuracy and corrected days after it was written down. By using the factory data collection terminals, the data can be collected and validated at its source, i.e., the factory worker. But what about the major programming effort necessary to monitor all these factory data collection terminals as well as validate the data as it is entered? And once that is done, how about the task of maintaining these programs? As everyone involved in designing or implementing computer systems knows, the cost of software is steadily increasing and the trend in industry is toward application packages. Because of this, Hewlett-Packard offers Datacap/1000, on the HP 1000, which assists the programmer in quickly and easily implementing data collection with these factory data collection terminals. Also, Datacap makes modification to the data collection system very simple. Datacap helps meet the challenge of collecting timely and accurate data in the factory.

COGELOG

ASK / 3000

THE NECESSARY COMPLEMENT TO IMAGE/QUERY

- ASK/3000 may access and report any information from an IMAGE/3000 data base

- Like QUERY, it enables you to:
  - -read,add,modify,delete information
  - -select entries
  - -generate reports

  but almost every limitation you have met using QUERY has disappeared !

- With ASK/3000 you can:

  -FIND and REPORT items from <u>different data-sets</u>

  -Access <u>sub-items</u> in any command

  -Call <u>user procedures</u> for solving a problem that cannot be solved with standard parameters

  -Use new <u>report statements</u> as IF, TRAILER or existing statements with greater capacities as HEADER DETAIL...

  -Use registers associated to Detail, Group or Total statements to perform <u>any calculation</u>

  -Call ASK/3000 from a <u>user program</u>

  -Etc....

COGELOG

QUESTIONS AND ANSWERS ON....ASK/3000.

1. Is ASK COMPATIBLE WITH QUERY ?

Yes, all reports written with QUERY may be run with ASK without any modification.

2. IS IT POSSIBLE TO SELECT RECORDS FROM A DETAIL DATA SET ACCORDING TO THE VALUE OF AN ITEM LOCATED IN A MASTER ?

Yes. The FIND command accepts multidata set criterions and this is true for any type of data set.

3. IS ASK HANDLING SUB-ITEMS ?

Yes. ASK allows you to write and modify sub-items. In all commands a sub-item may replace an item.

4. MAY I SELECT RECORDS ON THE VALUE OF THE FIRST CHARACTER OF AN ITEM ?

Yes. You can mask any non significant character by a "\" and select records on the value of the other;
example : FIND NUMBER = "\2\3\\"

5. CAN ASK SORT ON THE FIRST CHARACTER OF AN ITEM ?

Yes. Sorts can be performed on any substring inside an item (or sub-item). Overlapped keys are allowed.

6. MAY I REPORT ITEMS FROM DIFFERENT DATA SETS ?

Yes. You only have to specify to which data set belongs the desired item.

7. IF A PROBLEM CANNOT BE SOLVED BY ASK/3000 , AM I OBLIGED TO RE-WRITE THE WHOLE REPORT IN STANDARD PROGRAMMING ?

No. You can write a user procedure and call it in any HEADER,TRAILER, DETAIL, GROUP, or TOTAL statement.

8. MAY I EXECUTE CONDITIONALLY A REPORT STATEMENT ?

Yes. A new statement "IF" enables you to manage flags. Any report statement can be conditionally executed according to one of these flags.

9. <u>CAN ASK BE CALLED FROM A USER PROGRAM</u> ?

   Yes, You can use ASK's functions in your own program by creating an ASK process, and sending to it the name of an XEQ-file to be executed.

10. <u>MAY I SAVE A VALUE COMPUTED IN A REPORT</u> ?

    Yes, You can use the UPDATE function in a register statement to store a register content in an item. For any other storage, you have to write a user procedure.

11. <u>IF SUCCESSIVE REPORTS NEED THE SAME SORTS ON THE SAME DATA SET, CAN ASK SORT THIS SET ONLY ONCE</u> ?

    Yes. ASK keeps sort parameters in the select file, and the select-file can be saved as permanent.
    ASK knows when it has to perform the sort or not.

12. <u>WHAT ELSE CAN I DO WITH ASK</u> ?

    Many things, such as :

    . ROUND results of divisions in packed decimal format.
    . Print dates in EUROPEAN format.
    . Prevent DETAIL, GROUP, TOTAL lines to be spread on 2 pages.
    . Manage ASK output files.
    . Generate TRAILER lines at bottom of pages.
    . Execute registers functions only in GROUP or TOTAL lines.
    . Send messages to the operator.
    . Etc...

    Moreover, you can use ASK as a frame for any batch program, by writing a pseudo-report containing sorts, and register statements. You only have to write the code of your application as user procedures.

    You will find a summary of ASK capabilities and a detailed example in the next pages.

    This example is a typical supplier accounting report. You will find a schema of the data-base, the text of the report and the listing generated by ASK.

13.HOW CAN I BUY ASK ?

First, you can get more information about ASK.If you request it, we will send you free of charge, the reference manual and/or a demo tape (valid two months) to test it on your own site.

14.WHEN CAN I GET ASK ?

ASK is immediately orderable from :

COGELOG
1 res des Quinconces
91190 GIF SUR YVETTE
FRANCE

15.HOW IS ASK MAINTAINED ?

If you happen to find a bug in ASK, send us a detailed "bug-report". When the bug is corrected, every ASK installation will receive a new tape to load on its system. Moreover, a "bug-status-report" will be sent periodically with temporary turn-arounds for yet uncorrected bugs.

16.WHAT WILL HAPPEN IF H.P. MODIFIES ITS SOFTWARE ?

ASK is an SPL-written application program which directly calls IMAGE intrinsics (like any of your data-base application programs). More than 700 sites use the IMAGE subsystem all over the world. This is why HP CANNOT modify IMAGE external specifications. Therefore, ASK will run as long as IMAGE does.

*** DATA BASE SCHEMA EXAMPLE ***

```
$CONTROL NOLIST,ERRORS=2,TABLE
$TITLE "DATA BASE DEFINITION"
<<
             ***  DATA BASE SCHEMA  ***
>>
BEGIN DATA BASE SB;

PASSWORDS:

4   COMPTA;
5   FOURNI;

ITEMS:

  << A C C O U N T I N G   E N T R I E S   >>

      DAY   ,       I1  (/4) ; << ENTRY DAY >>
      MONTH,        I1  (/4) ; << ENTRY MONTH >>
      YEAR ,        I1  (/4) ; << ENTRY YEAR >>
      NENTRY,       I1  (/4) ; << ENTRY NUMBER >>
      VALUE,        P12 (/4) ; << ENTRY VALUE >>
      COMMENT,      U16 (/4) ; << COMMENT >>

  << S U P P L I E R S >>

      NSUPP ,        U6    ; << SUPPLIER NUMBER >>
      SNAME ,        U34   ; << SUPPLIER NAME >>
      STREET,        U34   ; << STREET NAME AND NUMBER >>
      ZIPCODE ,      U34   ; << ZIPCODE AND CITY NAME >>

$PAGE
SETS:

<<

** MSUPP :  S U P P L I E R   M A S T E R
-------------------------------------------

>>
NAME: MSUPP,AUTOMATIC (/5);

ENTRY:
      NSUPP(1);    << SUPPLIER NUMBER >>

CAPACITY: 1011;

<<

** DECRM : A C C O U N T I N G   E N T R I E S   O F   T H E   M O N T H
-------------------------------------------------------------------------

>>
NAME: DECRM,DETAIL (4,5/4,5) ;

ENTRY:
      DAY,
      MONTH,                      << ENTRY MONTH >>
      YEAR,                       << ENTRY YEAR >>
      NSUPP,                      << SUPPLIER NUMBER >>
      NENTRY,                     << ENTRY NUMBER >>
      COMMENT,                    << COMMENT >>
      VALUE,                      << ENTRY VALUE >>

CAPACITY: 2500;
<<

** DSUPP  :  S U P P L I E R    D E T A I L
--------------------------------------------

>>
NAME: DSUPP,DETAIL  (4,5/4,5);  << SUPPLIER DETAIL >>

ENTRY:
      NSUPP(MSUPP),
      SNAME,
      STREET,
      ZIPCODE,

CAPACITY: 550;
END.
```

*** REPORT SOURCE EXAMPLE ***

```
REPORT
*****************************************************
*             SUPPLIERS ACCOUNTING REPORT           *
*****************************************************
* OUTPUT CONTROL WORDS
* --------------------
LINES=56
EUROPE
NOSPLITGROUP
NOSPLITTOTAL
DYNAMICTRAILER
OUT=LP,"MOUNT SUPPLIER FORM ON LP  PLS."
*
* MEANING OF REGISTERS:
* ---------------------
*        R1 : SUB TOTAL FOR EACH SUPPLIER
*        R2 : GENERAL TOTAL
*
* PERMANENT HEADER
*'FULLDATE' IS AN USER PROCEDURE THAT PRINTS THE MONTH IN FULL LETTERS
*
H1,PAGENO,69,SPACE A1
H1,"PAGE-NUMBER",65
H2,"SUPPLIERS  ENTRIES    :                        PAGE",65
H2,FULLDATE(65),X
H3,48("-"),65
*
* CONDITIONAL HEADER : THESE STATEMENTS ARE THE SAME AS
*        THE GROUP STATEMENTS. THEY REPEAT THE GROUP LINES IF A
*        PAGE EJECT OCCURS BETWEEN A GROUP AND A TOTAL STATEMENT
*
H4,68("-"),69,SPACE B1,IF F4
H5,"I SUPP # :            NAME    : ",30,IF F4
H5,NSUPP,18,IF F4
H5,DSUPP.SNAME,66,IF F4
H5,"I",69,IF F4
H6,"I                     ADDRESS : ",30,IF F4
H6,"I",69,IF F4
H6,DSUPP.STREET,66,IF F4
H7,"I",2,IF F4
H7,"I",69,IF F4
H7,DSUPP.ZIPCODE,66,IF F4
H9,68("-"),69,IF F4
H10,"I   DATE   I       COMMENT       I",35,IF F4
H10,"    DEBIT        I     CREDIT    I",69,IF F4
H11,68("-"),69,IF F4
*
* GROUP: BEGINNING OF FRAME
*
G28,68("-"),69,SPACE B1
G27,"I SUPP # :           NAME    : ",30
G27,NSUPP,18
G27,DSUPP.SNAME,66
G27,"I",69
G26,"I                    ADDRESS : ",30
G26,"I",69
```

```
G26,DSUPP.STREET,66
G25,"I",2
G25,"I",69
G25,DSUPP.ZIPCODE,66
G23,68("-"),69
G22,"I   DATE   I        COMMENT        I",35
G22,"     DEBIT       I      CREDIT     I",69
G21,68("-"),69
G21,SET F4
*
S1,DAY
S2,NSUPP
S3, DSUPP.ZIPCODE
*
* MEANING OF THE FLAGS:
*       F1  :  SET = POSITIVE VALUE, CLEAR = NEGATIVE VALUE
*       F2  :  SET = POSITIVE SUB-TOTAL, CLEAR = NEGATIVE SUB-TOTAL
*       F3  :  SET = POS. GENERAL TOTAL, CLEAR = NEG. GENERAL TOTAL
*       F4  :  SET = BETWEEN GROUP/TOTAL, CLEAR = NOT BETWEEN
*
IF,VALUE > "0" THEN SET F1 ELSE CLEAR F1
IF,R1 > "0" THEN SET F2 ELSE CLEAR F2
IF,R2 > "0" THEN SET F3 ELSE CLEAR F3
D1,"I   /  /78 I",13
D1,"I                  I                     I",69
D1,DAY ,5
D1,MONTH,8
D1,COMMENT,31
D1,VALUE,67,E1,IF NOT F1
D1,VALUE,50,E1,IF F1
R1,ADD,VALUE
R2,ADD,R1,T2
*
* SUB-TOTAL  ..END OF FRAME
*
T20,68("-"),69
T21,R1,67,E1,IF NOT F2
T21,R1,50,E1,IF F2
T21,"I          BALANCE................:",35
T21,"I",69
T22,68("-"),69
T22,CLEAR F4
T2,R1
*
* TRAILER: THIS STATEMENT CLOSES A FRAME IF A PAGE EJECT OCCURS
*          BETWEEN A GROUP AND A TOTAL
*
Z1,68("-"),69,IF F4
*
* GENERAL TOTAL
*
TF1,"TOTAL BALANCE .............: ",36,SPACE B2
TF1,R2,67,E1,IF NOT F3
TF1,R2,50,E1,IF F3
E1,"ZZZ,ZZZ,ZZ9.99"
END.
```

*** USER PROCEDURE EXAMPLE ***

```
!JOB JOBUSER,USER.GROUP
!SPL,USLUSER
$CONTROL SUBPROGRAM,SEGMENT=USER,LIST,ADR
BEGIN
<<
  THIS PROCEDURE PRINTS THE CURRENT DATE IN FULL LETTERS
  THE PRINT POSITION IS PASSED THROUGH THE OPTIONAL PARAMETER '
>>
PROCEDURE FULLDATE(DBNAME,ISORT,LINEBUF,REGISTERS,STORAGE,FLAGS,PARAM);
INTEGER ARRAY DBNAME,ISORT,LINEBUF,REGISTERS,STORAGE;
LOGICAL ARRAY FLAGS;
INTEGER PARAM;

   BEGIN
   INTEGER N,D,INDEX,DAY,YEAR;
   BYTE ARRAY OUTPUT(*) = LINEBUF;
   INTEGER ARRAY MONTH'TABLE(0:11);
   INTEGER ARRAY M1(*) = MONTH'TABLE,  M2(*) = MONTH'TABLE(1),
                 M3(*) = MONTH'TABLE(2), M4(*) = MONTH'TABLE(3),
                 M5(*) = MONTH'TABLE(4), M6(*) = MONTH'TABLE(5),
                 M7(*) = MONTH'TABLE(6), M8(*) = MONTH'TABLE(7),
                 M9(*) = MONTH'TABLE(8),  M10(*) = MONTH'TABLE(9),
                 M11(*) = MONTH'TABLE(10), M12(*) = MONTH'TABLE(11);
   BYTE ARRAY STRING(0:107) = PB := "  JANUARY"," FEBRUARY","
           "    APRIL","      MAY","     JUNE","     JULY","
           "SEPTEMBER","  OCTOBER"," NOVEMBER"," DECEMBER";
   INTRINSIC ASCII,CALENDAR;

   M1:=M3:=M5:=M7:=M8:=M10:=M12:=31;        <<INIT. MONT'TABLE>>
   M4:=M6:=M9:=M11:=30;
   D := CALENDAR;
   DAY := D.(7:9);
   YEAR := D.(0:7);
   IF YEAR/4*4 = YEAR THEN M2 := 29 ELSE M2 := 28;
   INDEX := 0;
   DO  BEGIN
       N := MONTH'TABLE(INDEX);
       IF DAY-N < 0 THEN GOTO MONTH'FOUND;
       DAY := DAY - N;
       INDEX := INDEX + 1;
       END  UNTIL INDEX = 12;
       RETURN;
MONTH'FOUND:
   MOVE OUTPUT(PARAM-15) := STRING(INDEX*9),(9),2;
   MOVE * := "  19";
   ASCII(YEAR,10,OUTPUT(PARAM-2));
   END;
END.
!SEGMENTER
SL SL
PURGESL SEGMENT,USER
USL USLUSER
ADDSL USER,PMAP
EXIT
!EOJ
```

*** REPORT EXAMPLE ***

```
                                                      PAGE-NUMBER    1

                    SUPPLIERS ENTRIES      :          OCTOBER  1978
                    ------------------------------------------------

----------------------------------------------------------------
I  SUPP #: 190399  NAME     :  PUB SHARK                       I
I                  ADDRESS :  37 RUE DE LA REPUBLIQUE          I
I                             54140 JARVILLE MALGRANGE         I
----------------------------------------------------------------
I   DATE   I        COMMENT       I     DEBIT      I    CREDIT   I
----------------------------------------------------------------
I 21/10/78 I  RECRUTEMENT         I                I      228.00 I
I 22/10/78 I  RECRUTEMENT         I        228.00  I             I
----------------------------------------------------------------
I        BALANCE................:                           0.00 I
----------------------------------------------------------------

----------------------------------------------------------------
I  SUPP #: 200099  NAME     :  TRANSWORLD                      I
I                  ADDRESS :  17 RUE CHILDEBERT                I
I                             69002 LYON                       I
----------------------------------------------------------------
I   DATE   I        COMMENT       I     DEBIT      I    CREDIT   I
----------------------------------------------------------------
I 19/10/78 I  78/TH/08/129/Z      I        334.40  I             I
I 22/10/78 I  78/TH/08/129/Z      I                I    2,234.40 I
I 22/10/78 I  78/TH/08/129/Z      I      1,900.00  I             I
----------------------------------------------------------------
I        BALANCE................:                           0.00 I
----------------------------------------------------------------

----------------------------------------------------------------
I  SUPP #: 2008    NAME     :  JOJO INC                        I
I                  ADDRESS :  1 RUE DE LA GRANDE TRUANDERIE    I
I                             75004 PARIS                      I
----------------------------------------------------------------
I   DATE   I        COMMENT       I     DEBIT      I    CREDIT   I
----------------------------------------------------------------
I 22/10/78 I  POT DE VIN          I      4,380.05  I             I
I 22/10/78 I  11024               I                I   29,396.41 I
I 25/10/78 I  11024               I     25,016.36  I             I
----------------------------------------------------------------
I        BALANCE................:                           0.00 I
----------------------------------------------------------------

----------------------------------------------------------------
I  SUPP #: 1908    NAME     :  VENUS AND SONS                  I
I                  ADDRESS :  16 RUE FERRUS                    I
I                             75014 PARIS                      I
----------------------------------------------------------------
I   DATE   I        COMMENT       I     DEBIT      I    CREDIT   I
----------------------------------------------------------------
I 12/10/78 I  TEST1               I         13.00  I             I
I 13/10/78 I  TEST2               I          3.00  I             I
----------------------------------------------------------------
```

```
                                                      PAGE-NUMBER   2

                        SUPPLIERS ENTRIES     :            OCTOBER    78
                        ----------------------------------------------

-----------------------------------------------------------------------
I  SUPP #: 1908     NAME     :   VENUS AND SONS                        I
I                   ADDRESS :    16 RUE FERRUS                         I
I                                75014 PARIS                           I
-----------------------------------------------------------------------
I   DATE    I          COMMENT          I      DEBIT       I    CREDIT    I
-----------------------------------------------------------------------
I 13/10/78 I  TEST3                    I            2.00 I                I
I 14/10/78 I  TEST4                    I            1.00 I                I
I 17/10/78 I  TEST5                    I          105.00 I                I
I 18/10/78 I  460178                   I        3,785.76 I                I
I 19/10/78 I  460178                   I                 I       3,785.76 I
-----------------------------------------------------------------------
I         BALANCE...............:               124.00                   I
-----------------------------------------------------------------------

-----------------------------------------------------------------------
I  SUPP #: 180199   NAME     :   BILLY THE KID BANK                    I
I                   ADDRESS :    37 RUE DE VAUVENARGUES                I
I                                75018 PARIS                           I
-----------------------------------------------------------------------
I   DATE    I          COMMENT          I      DEBIT       I    CREDIT    I
-----------------------------------------------------------------------
I 22/10/78 I  1496/78                  I       14,736.84 I                I
I 22/10/78 I  1496/78                  I        2,593.68 I                I
I 22/10/78 I  1496/78                  I                 I      17,330.52 I
-----------------------------------------------------------------------
I         BALANCE...............:                                   0.00 I
-----------------------------------------------------------------------

-----------------------------------------------------------------------
I  SUPP #: 1909     NAME     :   OUT OF SPACE                          I
I                   ADDRESS :    23 AVENUE DU CHATEAU                  I
I                                78000 VERSAILLES                      I
-----------------------------------------------------------------------
I   DATE    I          COMMENT          I      DEBIT       I    CREDIT    I
-----------------------------------------------------------------------
I 22/10/78 I  37021                    I            6.58 I                I
I 22/10/78 I  37021                    I                 I          44.00 I
I 22/10/78 I  37021                    I           37.42 I                I
-----------------------------------------------------------------------
I         BALANCE...............:                                   0.00 I
-----------------------------------------------------------------------


      TOTAL BALANCE .............:               124.00
```

## MACHINE UTILIZATION

Series "C"

# MPE OBJECT CODE FORMATS
# AN INTRODUCTION TO USL AND PROGRAM FILES

Matthew J. Balander
The B & B Computer Company

ABSTRACT: The format and management of USL and PROGRAM files under the MPE III operating system, running on Hewlett Packard 3000/II and 3000/III computer systems, are presented in this paper. USL files are used to store relocatable binary modules, and PROGRAM files to store fully prepared programs. The presentation is aimed at programmers implementing compilers on HP3000 systems. The reader is assumed to be familiar with the architecture of these systems, and to understand basic concepts of relocatable code, link editing, and so on. The scope of the presentation is intended to provide the reader an adequate background with which to successfully pursue a compiler-writing project.

CONTENTS

1. Introduction
   1.1 Overview
   1.2 Conventions

2. PROGRAM files
   2.1 Contents of PROGRAM files
   2.2 The fixed area
   2.3 The global DB part
   2.4 The segments list
   2.5 The externals list
   2.6 The entry points list

3. USL files
   3.1 Contents of USL files
   3.2 File addressing
   3.3 Record zero
   3.4 The directory
      3.4.1 Directory lists
      3.4.2 Directory entries
      3.4.3 Header information blocks
   3.5 The information area
      3.5.1 Code modules
      3.5.2 Information headers

## 1. Introduction

### 1.1. Overview

All operating systems adopt conventions concerning the formats of object code files. These files must be in correct formats to be processed by system segmenters, linkage editors, and loaders. The MPE operating system defines four types, or formats, of object code files, as follows: user subprogram libraries (USL's), relocatable libraries (RL's), segmented libraries (SL's), and program files (PROGRAM's). Of these, RL's and SL's have rather specialized uses, and their formats are of little interest to the compiler writer. The formats of USL and PROGRAM files, on the other hand, are of great interest. If a compiler is to produce absolute code, it will generate PROGRAM files. If it is to produce relocatable code, it will generate USL files.

This paper presents an overview of the formats of USL and PROGRAM file formats used by MPE III, on HP3000/II and HP3000/III computer systems. It is neither exhaustive nor scrupulously detailed. Readers with some experience in object code formats will not find it difficult to fill in details not included here by examining USL and PROGRAM files.

A word of caution to the reader is appropriate at this point. Hewlett Packard is unco-operative, and seems quite indifferent to the needs of its users to understand MPE conventions. Because of this, all the information in this paper had to be deduced from examination of USL and PROGRAM files. In consequence, although the author believes the information included here to be fully accurate as of the date of this writing, the reader should keep in mind that it may nonetheless include some errors. For the most part, though, it may be used with confidence. The author has written a compiler which generates USL files based on section three of this paper, and a PROGRAM file decompiler based on section two. Both are operating satisfactorily.

### 1.2. Conventions

A number of conventions are adopted to enable concise explanations and illustrations. These conventions are applied consistently, but occassional, well-marked deviations do occur. The conventions are as follows:

- o SPL/3000 notation is used in all cases where illustrative code is provided.

- o "P" represents a variable of type integer pointer. In all tables, illustrations, and examples it is assumed to point to the first word of the entity under discussion.

- o In the case of single bit fields, "1" is the "on" state, and "0" is the "off" state. Similarly, the on state is the true state, and the off state is the false state.

- o Names in PROGRAM and USL files, such as procedure or segment names, are a variable number of words in length. The first byte of a name is always in the P.(8:8) field, the length of the name in bytes in P.(4:4), and various context-dependent information in P.(0:4). The name continues in as many consecutive bytes as needed, beginning with P.(8:8). A name field is always an integral number of words in length. The last byte is thus wasted if the name is an even number of bytes long. In illustrations, the P.(0:8) field will be diagrammed explicitly, but the remainder of the name will be shown simply as a large undivided area. The reader should keep in mind that this represents a variable length field. In the text of the paper, this entire group of fields is referred to as a "name field," and the various parts are not explicitly mentioned unless necessary for the discussion.

- o In illustrations, when a word is divided, unless indicated otherwise, it is divided into bytes, or into nibbles. Because of this, no explicit bit field indications are given in illustrations for byte or nibble aligned fields.

## 2. PROGRAM files

The loader in MPE processes PROGRAM files. In a program file, all relocatable references have been resolved, global storage laid out and initialized, and segmentation completed. The only link editing remaining to be done is to establish linkages with external program units to be found in SL's. This last linking step involves only the completion of segment transfer tables. Particular STT entries have already been assigned to particular externals, so no relocation per se need be performed at load time.

PROGRAM files have the following file characteristics:

- o Their record length is 128 words.
- o They are identified by a file code of 1029.
- o The file length is between 4 and 32727 records, inclusive
- o The records are fixed length, binary, without carriage control.
- o The file consists of only a single extent.

All these restrictions are imposed by the MPE segmenter and loader. Other file configurations are not acceptable.

### 2.1. Contents of PROGRAM files

A program file is logically divided into five parts, each of which must begin on a record boundary. The parts are as follows:

1. A "fixed area" containing pointers to the rest of the file, and other miscellaneous information
2. An image of the initial global DB area for the program, already fully initialized
3. The code segments comprising the program
4. A list of externals referenced by the program
5. A list of entry points to the program.

These five parts will occur in a PROGRAM file in the order listed. If a program uses no global DB storage, the global DB area part of the file will be omitted. Since all five parts of program files begin on record boundaries, record addresses are used throughout the file for indicating the location of needed information. All record addresses used in program files are single word integers.

## 2.2. The fixed area

The fixed area occupies the first one or two records of the file. The length of the area, one or two records, depends on whether or not all the information to be included in the area fits in a single record. It will always fit in two. Table 2.2A lists the contents of the fixed area in order.

Table 2.2A -- Fixed Area Contents

| Field | Contents |
|---|---|
| P.(0:1) | program contains fatal error |
| P.(1:1) | program contains non-fatal error |
| P.(2:1) | zero the DB area prior to starting execution of the program |
| P.(3:1) | program contains at least one privileged segment |
| P.(4:2) | (use undetermined, only zero observed) |
| P.(6:1) | program has NS capability |
| P.(7:1) | program has BA capability |
| P.(8:1) | program has IA capability |
| P.(9:1) | program has PM capability |
| P.(10:1) | program has CR capability |
| P.(11:1) | program has RT capability |

Table 2.2A -- Fixed Area Contents (cont.)

| Field | Contents |
|---|---|
| P.(12:1) | program has MR capability |
| P.(13:1) | (use undetermined, only zero observed) |
| P.(14:1) | program has DS capability |
| P.(15:1) | program has PH capability |
| P(1) | the number of segments in the program |
| P(2) | the number of words in the global DB area of the program's run-time stack |
| P(3) | beginning record number of the image of the global DB area of the stack (should be ignored if P(2)=0) |
| P(4) | beginning record number of the segments list |
| P(5) | the initial stack size (";STACK=" of prep command) |
| P(6) | the initial DL size (";DL=" of prep command, zero if ";DL=" not specified) |
| P(7) | the maxdata specification (";MAXDATA=" of prep command, -1 if ";MAXDATA=" not specified) |
| P(8) | beginning record number of entry points list |
| P(9) | logical segment number of the entry points to the program |
| P(10) | PB relative address of primary entry point |
| P(11) | execution time DB relative address of a table used by TRACE/3000 (-1 if table not used) |
| P(12) | execution time DB relative address of the FORTRAN logical units table, the FLUT (-1 if FLUT not used) |

Table 2.2A -- Fixed Area Contents (cont.)

| Field | Contents |
|---|---|
| P(13) | beginning record number of the externals list |
| P(14) | STT number of primary entry point |
| P(15) | execution time DB relative address of TRAPCOM', a common block used for interfacing to user trap routines |
| P(16) to P(27) | these locations have been observed only with the value zero--they are probably reserved, and should always be zero |

Following P(27) are two variable length subareas of the fixed area. The first begins in P(28), and is P(1) bytes in length. That is, there is one byte for each segment in the program. This subarea is always an integral number of words in length, so a byte is sometimes wasted on the end. It is believed that the loader uses this subarea for mapping logical segments to actual segments. After a program has been prepared, but before it has ever been loaded, this subarea will contain all zeros.

The second subarea begins in the word immediately following the last word of the first subarea. The second subarea includes a one-word segment descriptor for each segment in the program. These segment descriptors are in the same order as the segments themselves in the file. The format of a segment descriptor is given in table 2.2B.

Table 2.2B -- Segment Descriptor Format

| Field | Contents |
|---|---|
| P.(0:1) | segment is privileged |
| P.(1:1) | (use undetermined, only zero observed) |
| P.(2:14) | the length of the segment, in words |

## 2.3. The global DB part

The global DB part is simply an image of all DB and secondary DB which is to be allocated when the program begins execution. As many records as necessary are used to hold the needed number of words. Unused words at the end of the last record, if any, are ignored. If the number of words of DB is given in the fixed area as zero, then this area may be omitted from the file altogether, and the record pointer to this area in the fixed area may be set to one (one is the only value for this pointer yet observed in this context).

This DB area image is fully initialized. It includes TRACE/3000 tables, the FLUT table, common blocks, DB and secondary DB arrays and simple variables, and anything else which must be placed in the DB area. There is no opportunity to add to the DB global area once the program begins execution, since it will be delimited by DB and the initial Q register setting. Allowance must be made for all global run-time storage at the time the PROGRAM file is generated.

## 2.4. The segments list

The code segments which comprise the program are placed, one after another, in the segments list. Each segment begins on a record boundary, and occupies an integral number of records. The actual length of each segment, in words, is given in the segment descriptor words in the fixed area of the PROGRAM file. Unused words at the end of the last record of a segment, if any, are ignored.

The code segments of a program are often referred to by their logical segment numbers. A logical segment number simply gives the position of the segment in the segments list. The first segment in the segments list is logical segment number zero, the next is logical segment number one, and so on. Actual segment numbers, which will be used in the XCST for the program, are assigned by the loader when the program is loaded. Segment transfer tables will contain the actual segment numbers used when the program was last loaded.

There is a one-to-one correspondence between the entries in the segment descriptor words in the second subarea of the fixed area, and the segments in the segments list. The first descriptor applies to the first segment, the second to the second, and so on. The record number of any particular segment in the file must be deduced from the segment descriptor words. If, for example, it were desired to find the second segment, the first segment descriptor word would be used to calculate the number of records occupied by the first segment. This number would be added to the record number of the beginning of the segments list given in the fixed area. The resulting record number is the first record of the desired segment.

## 2.5. The externals list

The externals list includes entries for all externals referenced by the program. For each external it gives the segment and STT numbers of the program segments to be patched with the actual segment and STT numbers of the external. In addition, a parameter information block is included in each entry, which indicates the calling sequence which the program uses to call the external. This list is organized as a simple linear list. The list is terminated by an "entry" with zero in its first word.

Illustration 2.5 shows the format of entries on the externals list. Each entry begins with a name field. The P.(0:4) field of the name field is unused, and should be set to zero. Following the name field is a word with three fields. P.(0:4) of this word should be set to zero. It is unused. The use of P.(4:4) is not fully known, but it appears to be used by the loader to indicate how the external reference was satisfied. When generating the program file, it should be set to zero. P.(8:8) gives the number of references to this external by the program. (This number should never exceed the maximum number of code segments allowed for a single program, since a given external should occur only once in any given segment transfer table.)

Following the word giving the number of references, there is a one-word reference descriptor for each reference. This field is thus variable in length. P.(0:8) of each descriptor gives the segment transfer table entry number for this external in a referencing program segment, and P.(8:8) gives the logical segment number of the referencing program segment. This entire area is shown as a single undivided area in illustration 2.5, but the reader should keep in mind that it is a variable length field.

After the reference descriptors is a parameter information block. This parameter information block is in the same format used in USL files.

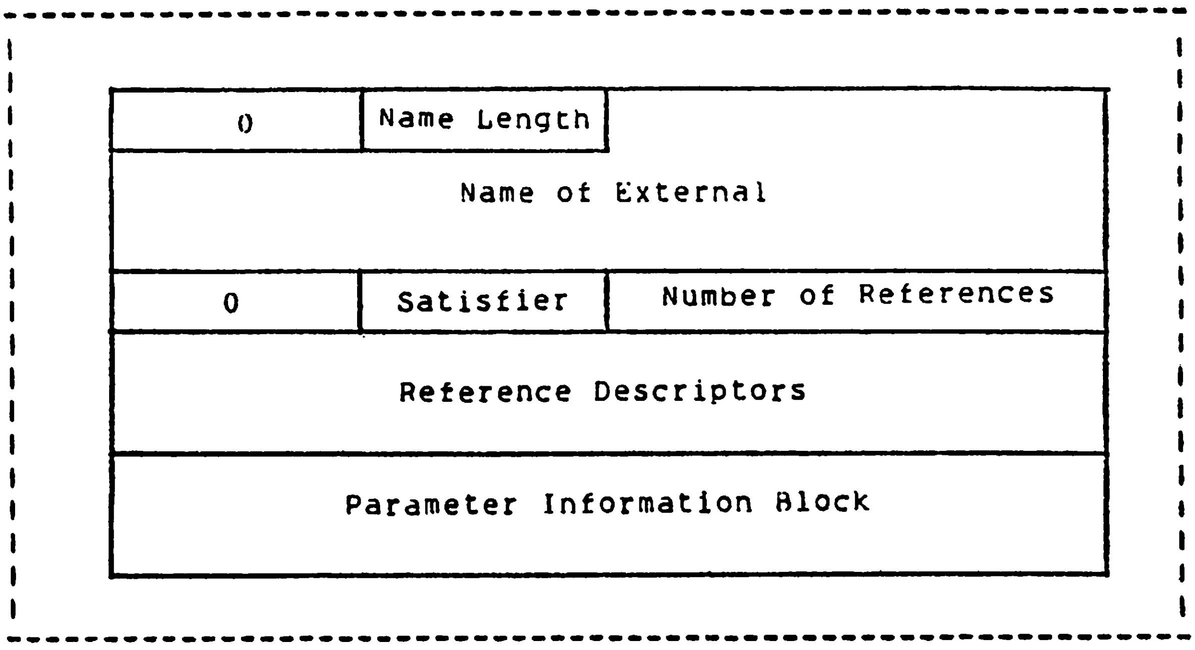


Illustration 2.5 -- Format of Externals List Members

## 2.6. The entry points list

The entry points list gives all entries to the program. At least the name of the outer block is included in this list. All entry points to the program must of course be in the same segment as the primary entry point. The logical segment number of this segment is indicated in the fixed area in the beginning of the program file. This list is a simple linear list. It is terminated by a list "member" with zero in its first word.

Members of the entry points list all consist of a name field followed by exactly two words. The P.(0:4) field of the name field is unused and should be set to zero. The first of the two words following the name gives the PB relative address of the entry point. The second gives the segment transfer table number of the entry point.

The format of entry points list members is shown in illustration 2.6.

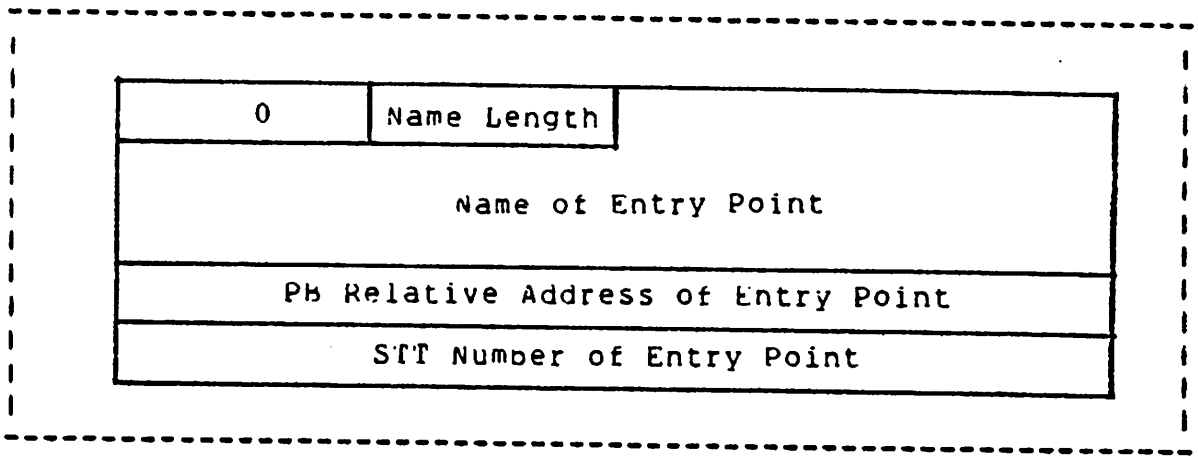


Illustration 2.6 -- Format of Entry Points List Members

## 3. USL files

USL files are the principal form of input to the MPE segmenter. From a USL file with the appropriate contents the segmenter can generate SL, RL, and PROGRAM files. "Relocatable binary modules" are stored in USL's. Segmentation, global and secondary DB address assignments, external procedure references, PB relative references, and the like have not yet been resolved in these files. Although USL files constitute the most complex form of compiler output on HP3000 computer systems, they are also the most flexible, giving the user the most options.

USL files have the following file characteristics:

- o Their record length is 128 words.
- o They are identified by a file code of 1024.
- o The file length must be from 4 to 32727 records, inclusive.
- o Records must be fixed-length binary, without carriage control.

These restrictions are imposed by the MPE segmenter. Unlike program files, USL files may be composed of any number of extents.

## 3.1. Contents of USL files

A USL file is logically divided into three major parts, each of which must begin on a record boundary. The parts are as follows:

1. "record zero" containing pointers to the rest of the file, list heads, and other miscellaneous information

2. the "directory" containing one entry for every RBM, segment, and entry point in the file

3. the "information block" containing all information headers and code modules.

These three parts always occur in the order specified, and are of a fixed length in any given file. This length may vary from file to file, but within any one file, the boundaries are clearly defined by information in record zero. Record zero is always simply the first record in the USL file.

The directory always begins with the second record of the USL file, at file address 00001 000 (see 3.2 for a discussion of file addressing). The information block always begins at a file address specified in record zero. In both the directory and the information block, information is placed in successive contiguous locations. Record boundaries are not recognized. If any entries or headers are deleted from either of these two areas, the space they formerly occupied is not recovered. It is referred to as "garbage," and is never used again. Never-used directory space is referred to as "available" directory, and never-used information block space as "available" information.

An intrinsic named ADJUSTUSLF, documented in the MPE segmenter reference manual, may be used to expand the directory or the information block as desired. The directory, however, may not exceed 32K words, and the file may not exceed a total of 32K-1 records.

A USL file may be initialized to the empty state via the intrinsic INITUSL. This intrinsic is documented in the MPE segmenter reference manual.

In total, then, a USL file consists of the following five parts, in this order: record zero, in-use directory, available directory, in-use information, and available information. These areas are delimited by pointers and length values kept in record zero.

## 3.2. File addressing

It is appropriate at this point to discuss USL file addressing. Locations within a USL file are identified by specifying the word within the file which is of interest. The first word in the file is word number zero. Seven bits are required to specify the offset of a word from the beginning of a record. Up to 15 bits may be required to specify record within file. Full file addresses are thus normally stored in double words, in which strictly speaking only the low order 22 bits are significant.

File addressing within the directory is somewhat different. Single word addresses are used. P.(9:7) is still a word offset into a particular record, but P.(1:8) is now the record number.

Unused high-order bits in both double and single word file addresses should be set to zero. This will provide compatibility with the MPE segmenter.

The high order bit of a file address often has special significance. It may be used to indicate that the address is a thread link instead of a normal link. It may be used to indicate the end of a list. It has various uses, which should be carefully considered when interpreting any file address.

The address zero has special significance. It is the null address. No pointer ever points to record zero.

File addresses are often relative to some location in the file. The starting address of the information block is normally used as the base address. The value of a relative address (even if it is zero) is added to the base address to obtain an actual file address. The high-order bit of an address should not be involved in this calculation, since it has special interpretations. It should be set to zero in both the base and the offset before adding. It is essential always to consider whether an address is absolute or relative. This depends on the context in which the address occurs.

In symbolic form, file addresses are represented by two octal numbers. The first is five digits in length, and specifies the record number. The second is three digits in length, and specifies the offset to the desired word in the record. These two numbers are separated by a blank. If the high-order bit of an address is on, then '(1)' is prepended to the five-digit record number. (If in the given context the high-order bit has no significance, the '(1)' may be omitted.)

## 3.3. Record zero

Record zero occupies the first record of a USL file. It contains pointers and counters essential to interpreting the remainder of the file. Table 3.3 lists the contents of record zero in order. A name is given to each field which is used as a convenience in referring to the field.

Table 3.3 -- USL File Record Zero Contents

| Word(s) | Name | Contents |
|---|---|---|
| 0 | | the constant '1' -- apparently identifies the file as a valid USL file |
| 1 | NDE | number of entries in the directory |
| 2 | DL | the directory length, in words (number of words which have already been allocated; includes garbage) |
| 3 | DG | number of words of DL which are 'garbage' |
| 4 | NDGE | number of directory garbage entries |
| 5 | BDL | list head for the block data list |
| 6 | IPL | list head for the interrupt procedure list |
| 7 | SL | list head for the segments list |
| 8,9 | FL | length of the entire USL file, in words |
| 10 | SAAD | start address of available directory space (should be equal to 00001 000 + DL) |
| 11 | ADL | available directory length, in words |

Table 3.3 -- USL File Record Zero Contents (cont.)

| word(s) | Name | Contents |
|---|---|---|
| 12,13 | SAIB | start address of the information block (should be equal to SAAD + ADL) |
| 14,15 | IBL | length of the information block, in words (number of words which have already been allocated; includes garbage) |
| 16,17 | SAAIB | starting address of the available information block space |
| 18,19 | AIBL | length in words of the available information block |
| 20,21 | IBG | number of words of IBL which are 'garbage' |
| 22 | NIBGE | number of garbage information block entries |
| 23-32 | | apparently reserved; should always be set to zero |
| 33-127 | | hash list heads |

The use of many of these fields will be explained in subsequent sections of this paper. The use of others is indicated in illustration 3.3. This illustration labels various locations and areas in a USL file with the names assigned to the fields of record zero which refer to those portions of the USL file.

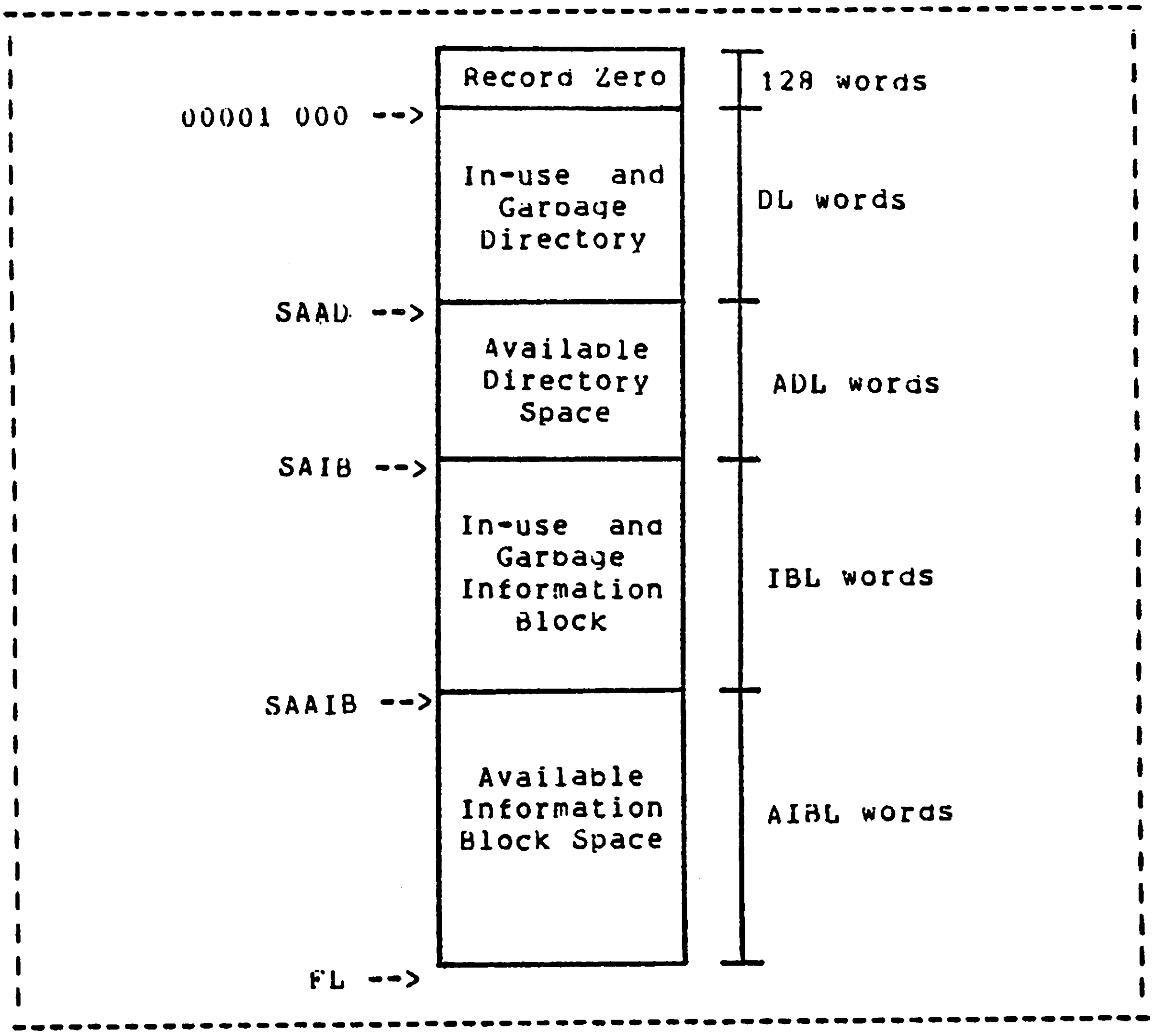

Record Zero
128 words
00001 000 -->
In-use and Garbage Directory
DL words
SAAD -->
Available Directory Space
ADL words
SAIB -->
In-use and Garbage Information Block
IBL words
SAAIB -->
Available Information Block Space
AIBL words
FL -->

Illustration 3.3 -- Use of Record Zero Fields

## 3.4. The directory

The USL file directory contains all information needed to process and manipulate information contained in the information block. A great variety of items are found in the directory, but they are easily classified into distinct groups. The elementary directory data item is the 'entry.' The data structures formed from entries are familiar sorts of trees and linked lists. Several implementation aspects of the directory, which do not conveniently fall into any other section, are listed below:

1. The directory begins at file location 00001 000.

2. Each directory entry consists of some number of contiguous words. Entries are not, however, necessarily contiguous within the directory. That is, there may be some unused space between entries.

3. File record boundaries are not recognized within the directory. Entries may span record boundaries freely.

4. All pointers to entries in the directory are absolute pointers. If the pointer is contained within record zero, or within the directory itself, it is a single word, and not a double word, pointer. (See section 3.2.)

5. All pointers to the information block which are contained in the directory are double word pointers. All such pointers are relative to SAIB. (See sections 3.2 and 3.3.)

The entries contained in the directory are related on lists. There are only eight types of entries, and only four types of lists, all of which are described in detail in the following subsections.

## 3.4.1. Directory lists

The four types of lists in the directory are as follows: the interrupt procedure list, the segment list, the block data list, and the hash lists. Each of these four is discussed in a separate subsection below. With each subsection is provided an illustration of the list discussed.

For every list in the directory, there is a certain type of entry, or there are certain types of entries, which are included on that list. With the exception of the hash lists, only entry types appropriate to the list may be placed on the list. All entry types are appropriate to the hash lists. Entries may be included only on lists for which they are appropriate.

Every entry is on exactly two directory lists. It is on one, and only one, of the block data list, the interrupt procedure list, and the segment list. It is also on one, and only one, of the hash lists.

## 3.4.1.1. The interrupt procedures list

The interrupt procedure list ('IPL') is a linear linked list. Its list head is IPL in record zero (see 3.3). All interrupt procedure directory entries are on this list. Entries are linked together by their brother pointers, with a link of zero terminating the list. As of this writing, no further information is available about the IPL directory list.

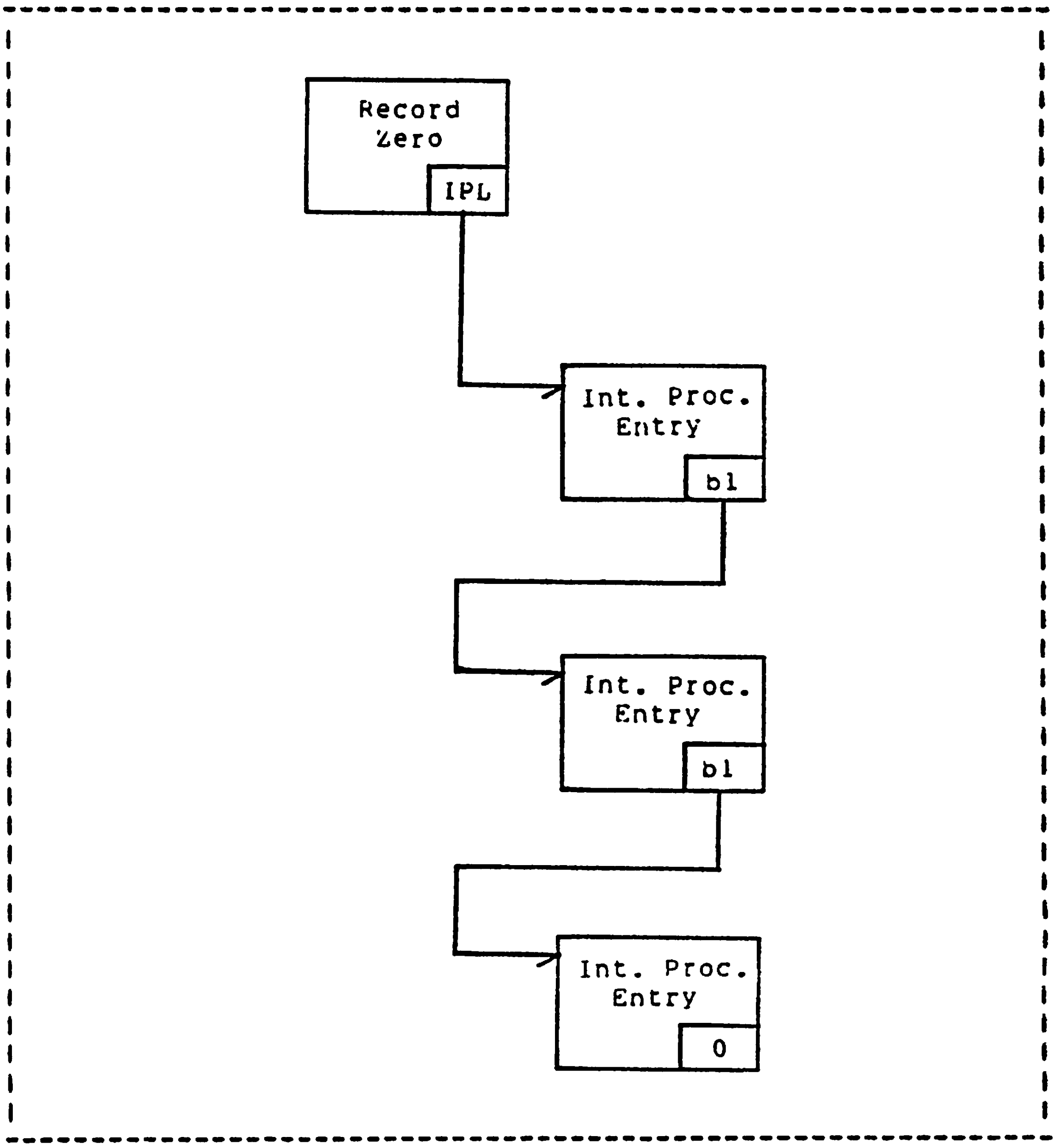


Illustration 3.4.1.1 -- Interrupt Procedures List

## 3.4.1.2. The block data list

The block data list ('BDL') is a linear linked list. Its list head is BDL in record zero (see 3.2). Block data subprograms generate block data RBM's. All block data directory entries are on the BDL directory list. They are linked together by their brother pointers, with a link of zero terminating the list.

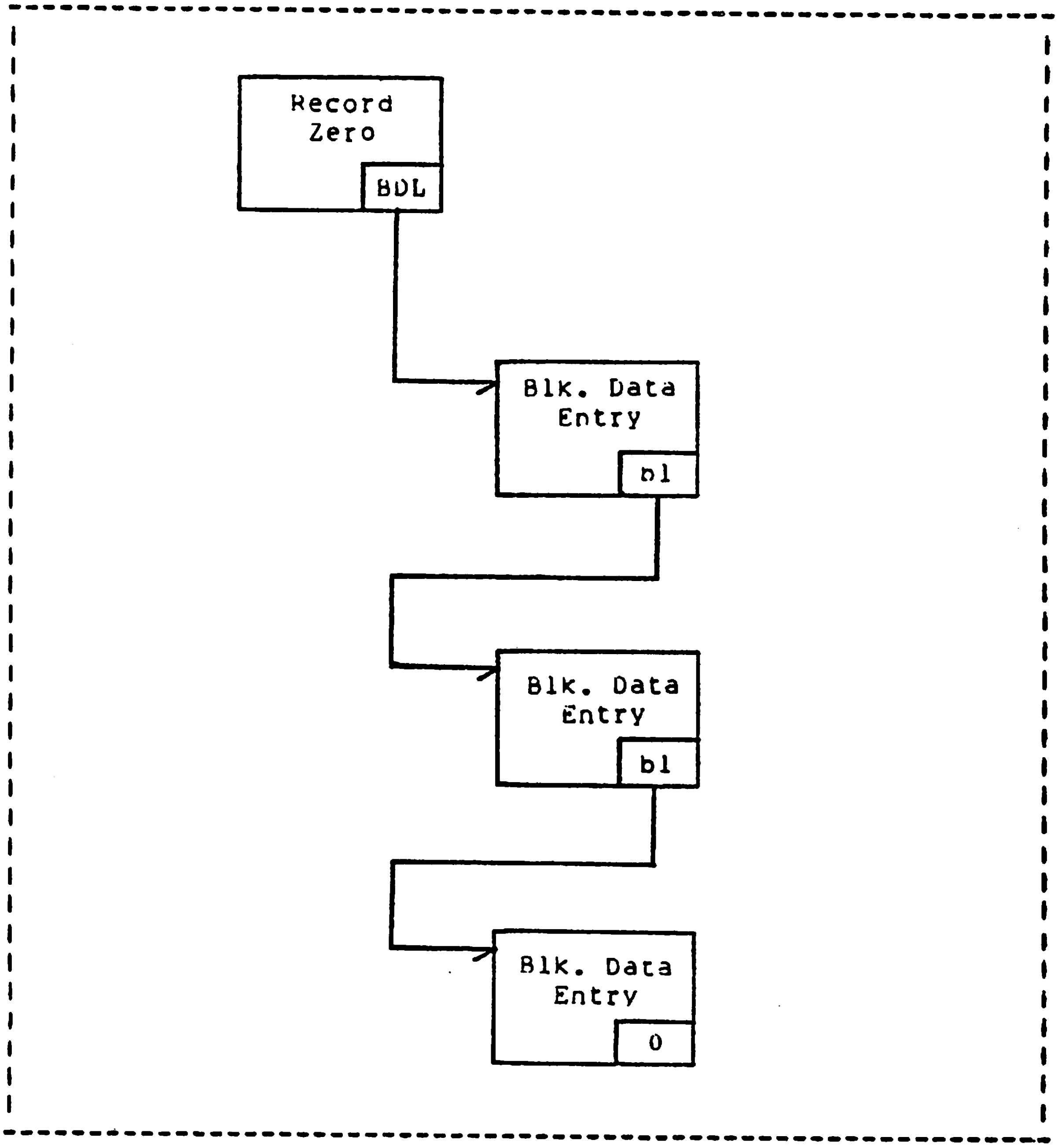


Illustration 3.4.1.2 -- Block Data List

### 3.4.1.3. The segment list

The segment list is really a tree. The pointer to the root of the tree is SL in record zero (see 3.2). The following three types of entries are found in the tree: segment entries, RBM entries (which may be either primary outer block or primary procedure type entries), and secondary entry point entries (which may be secondary outer block, secondary procedure with parameters, or secondary procedure without parameters type entries).

The segment entries do actually form a linear linked list, with SL in record zero as the list head. They are linked by their brother pointers, with a link equal to zero terminating the list.

A segment entry may have zero or more sons. The immediate sons of a segment entry must be RBM entries. The son pointer of a segment entry points to the segment entry's first son. All the sons of a given segment are linked in a linear list by their brother pointers. This family list is terminated by a link with its high-order bit turned on, and which points back to the parent segment entry. For example, if a segment entry is located at 00033 027, then the link terminating the family list would be (1)00033 027. If this segment entry had no sons, then the segment entry's son pointer would be (1)00033 027.

Each RBM entry may also have zero or more sons. The sons of an RBM entry are secondary entry point entries. The son pointer of an RBM entry is analogous to the son pointer of a segment entry. It points to the first son, and other sons are linked into the family by their brother pointers. Again, the family list is terminated by a link with its high-order bit turned on, and which points back to the parent RBM entry. If an RBM entry has no sons, then its son pointer points to itself, just as in the case of segment entries with no sons.

The links terminating family lists in the segment list are referred to as 'thread' links, since they refer back to the root of the subtree in which a node is located. The son relationship is defined only from segment to RBM entries, and from RBM to secondary entry point entries. The brother relationship is defined from segment to segment, from RBM to RBM, and from secondary entry point to secondary entry point entries. The father relationship is defined from RBM to segment, and from secondary entry point to RBM entries. The tree is thus a left-son/right-sibling, threaded tree data structure.

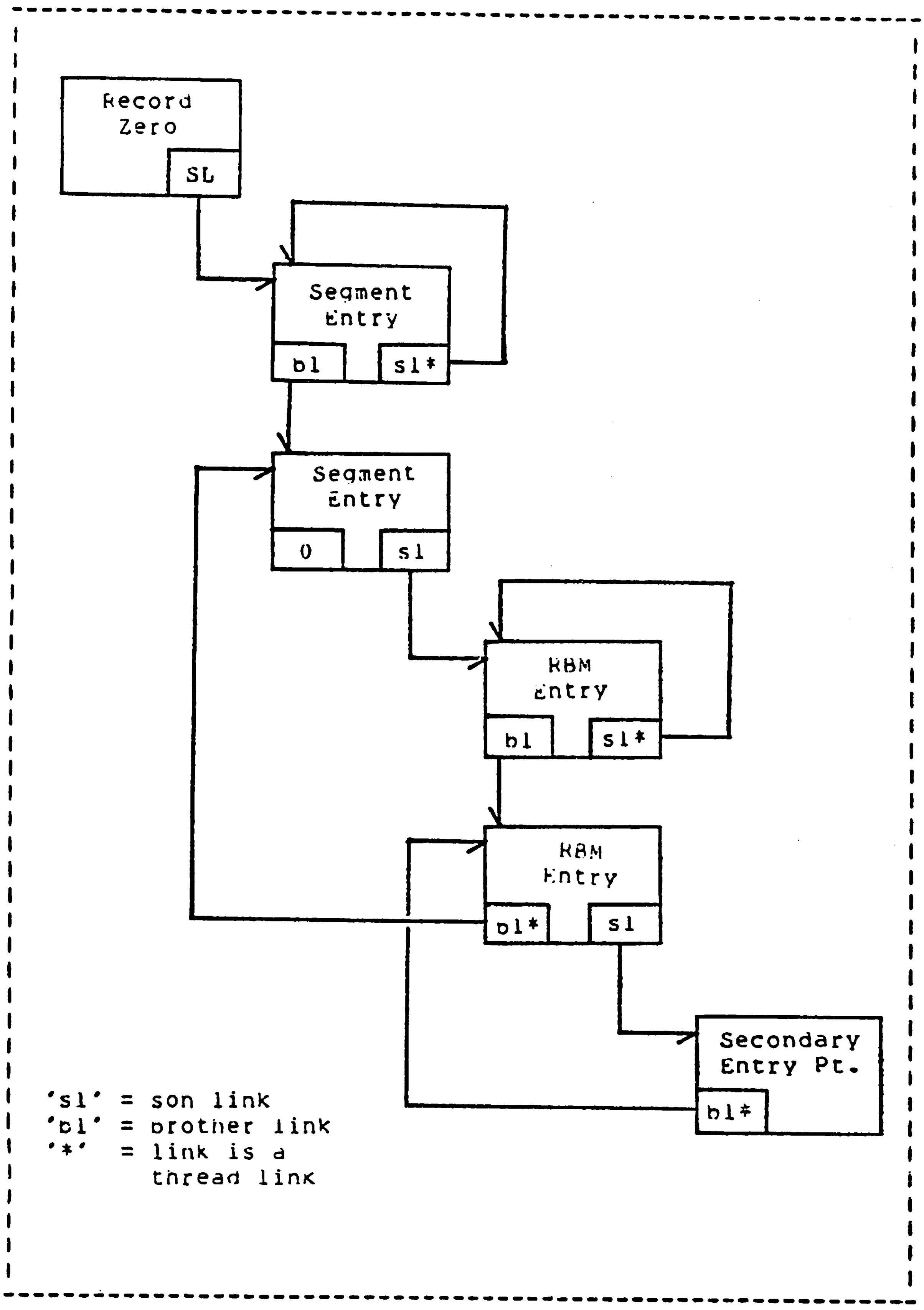


Illustration 3.4.1.3 -- Segment List

### 3.4.1.4. The hash lists

To facilitate quick access to directory entries by name, every directory entry is also placed on a hash list. USL files use 95 hash lists, the list heads of which are in record zero (see 3.2). Each list head is a single word absolute pointer into the directory. Directory entries which hash to the same list are linked to each other by their hash links (see 3.4.2). Each of the 95 hash lists is thus a linear linked list. A zero link terminates a hash list. Hashing a name produces an integer between 0 and 94, inclusive, which is used as an index into the 95 hash list heads to access the hash list on which the entry referred to by the name is located.

An entry should always be added to a hash list nearest to the hash list head. That is, the hash list head should be made to point to the entry, and the entry's hash link to point to the entry formerly pointed to by the hash list head.

The MPE segmenter refers to the 'index' of an RBM, or directory entry. This is a reference to how recently the entry was added to its hash list. The most-recently added is indexed one, the next-most-recently is indexed two, and so on. "Least-recent" on any given list refers to the entry on the list with a hash link equal to zero. When the MPE segmenter refers to the index of an entry, only the entries of a given name are considered. The entire list is not relevant. The index zero has a special meaning. It refers to the most-recent active entry having the given name on the hash list (active/inactive entries are discussed in the MPE segmenter reference manual).

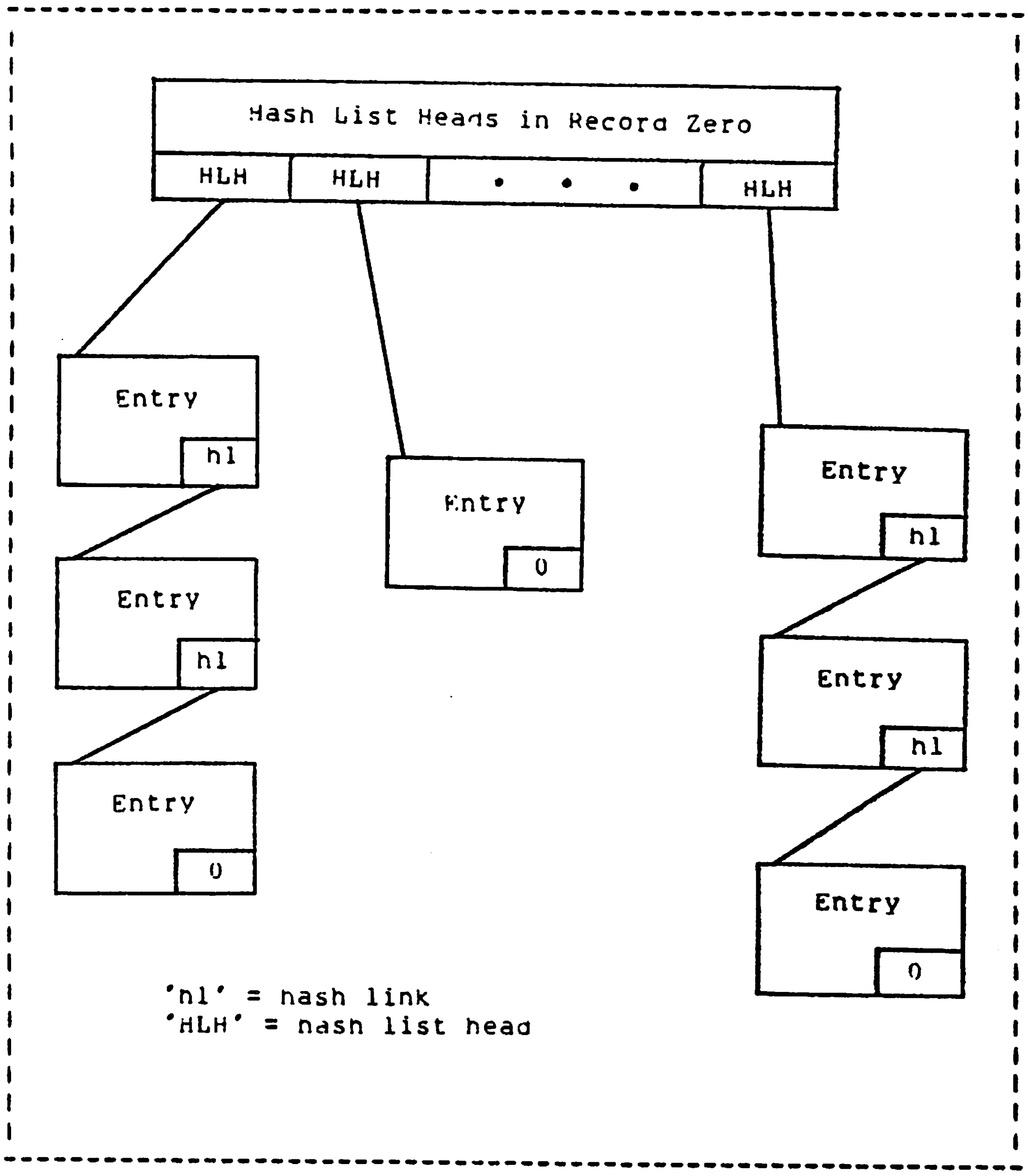


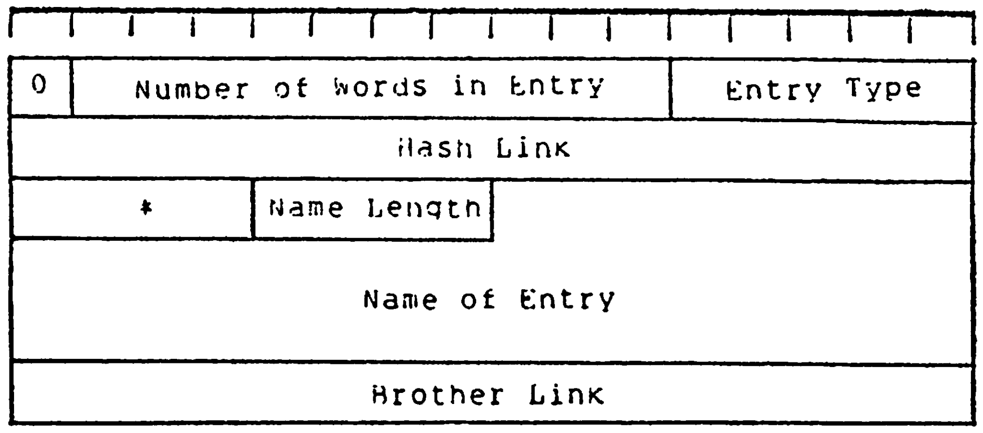
Illustration 3.4.1.4 -- Hash Lists

## 3.4.2. Directory entries

In this section, all eight entry types found in the directory are presented. All entries have some fields in common, which together form a standard directory entry prefix. Included in this prefix is a name field, giving the name associated with the entry. 'P1' is used to denote an integer pointer which points to the word immediately following the last word used by the name field. Illustration 3.4.2 shows the layout of the prefix. The contents of the prefix are described in table 3.4.2A.

Table 3.4.2A -- Directory Entry Prefix Contents

| Field | Contents |
|---|---|
| P(0).(1:10) | number of words in this entry |
| P(0).(11:5) | type of this entry; types have following designations:<br>1 segment entry<br>2 primary outer block entry<br>3 secondary outer block entry<br>4 primary procedure entry<br>5 secondary procedure entry, without parameters<br>6 interrupt procedure entries<br>7 block data entry<br>8 secondary procedure entry, with parameters<br>Other entry types are undefined |
| P(1) | the entry's hash link (see 3.4.1.4) |
| P(2) | the entry's name field |
| P1(0) | the entry's brother link |



* See Table 3.4.2B

Illustration 3.4.2 -- Standard Prefix

The P(2).(0:4) field (that is, the first nibble of the name field) has important uses. The interpretation of this field depends on entry type. These interpretations are given in table 3.4.2B.

Table 3.4.2B -- Interpretation of P(2).(0:4)

| Entry | Field | Interpretation |
|---|---|---|
| 1 | 0:1 | shows whether entry is active or not ('1'=inactive) |
| | 1:3 | reserved; should be set to zero |
| 2 | 0:1 | shows whether entry is active or not ('1'=inactive) |
| | 1:1 | shows whether entry is callable ('1'=uncallable) |
| | 2:1 | shows whether program unit must execute in privileged mode |
| | 3:1 | reserved; should be set to zero |

Table 3.4.2B - Interpretation of P(2).(0:4) (cont.)

| Entry | Field | Interpretation |
|---|---|---|
| 3 | 0:1 | shows whether entry is active or not ('1'=inactive) |
|  | 1:1 | shows whether entry is callable ('1'=uncallable) |
|  | 2:2 | reserved; should be set to zero |
| 4 | 0:1 | shows whether entry is active or not ('1'=inactive) |
|  | 1:1 | shows whether entry is callable ('1'=uncallable) |
|  | 2:1 | shows whether program unit must execute in privileged mode |
|  | 3:1 | shows whether entry is hidden |
| 5 | 0:1 | shows whether entry is active or not ('1'=inactive) |
|  | 1:1 | shows whether entry is callable ('1'=uncallable) |
|  | 2:1 | reserved; should be set to zero |
|  | 3:1 | shows whether entry is hidden |
| 6 | 0:1 | shows whether entry is active or not ('1'=inactive) |
|  | 1:2 | apparently an interrupt procedure type number |
|  | 3:1 | reserved; should be set to zero |

Table 3.4.2B - Interpretation of P(2).(0:4) (cont.)

| Entry | Field | Interpretation |
|---|---|---|
| 7 | 0:1 | shows whether entry is active or not ('1'=inactive) |
| | 1:1 | set if fatal error in block data RBM |
| | 2:1 | set if non-fatal error in block data RBM |
| | 3:1 | reserved; should be set to zero |
| 8 | 0:1 | shows whether entry is active or not ('1'=inactive) |
| | 1:1 | shows whether entry is callable ('1'=uncallable) |
| | 2:1 | reserved; should be set to zero |
| | 3:1 | shows whether entry is hidden |

In the subsections of this section all entry types are explained. A diagram of most types is presented to illustrate the format of the entry. Mention will often be made of "parameter information blocks," "header information blocks," and "header information sets" ('PIB', 'HIB', and 'HIS', respectively). HIB and HIS are described in 3.4.3, and so are not further discussed here. A PIB provides the calling sequence of a program unit. It will always in illustrations be drawn as a single large area, but the reader should keep in mind that it is really comprised of one or more words, and is thus a variable length field. 'P2' is used to denote an integer pointer pointing to the word immediately following the PIB.

## 3.4.2.1. Segment entries

For segment type directory entries, only a single word is appended to the standard prefix. That word contains the son link of the entry. (The significance of a son link is discussed in 3.4.1.)

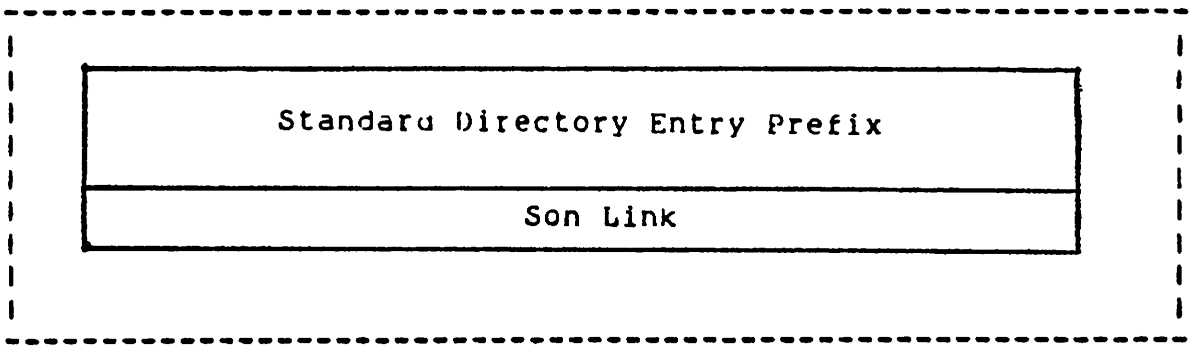


Illustration 3.4.2.1 -- Segment Entry

## 3.4.2.2. Primary outer block entries

A number of fields follow the standard prefix in primary outer block type directory entries. They are described in table 3.4.2.2. In this table, 'P' is assumed to be an integer pointer to the word of the entry immediately following the standard prefix. Each field is given a name to simplify reference to the field in this paper.

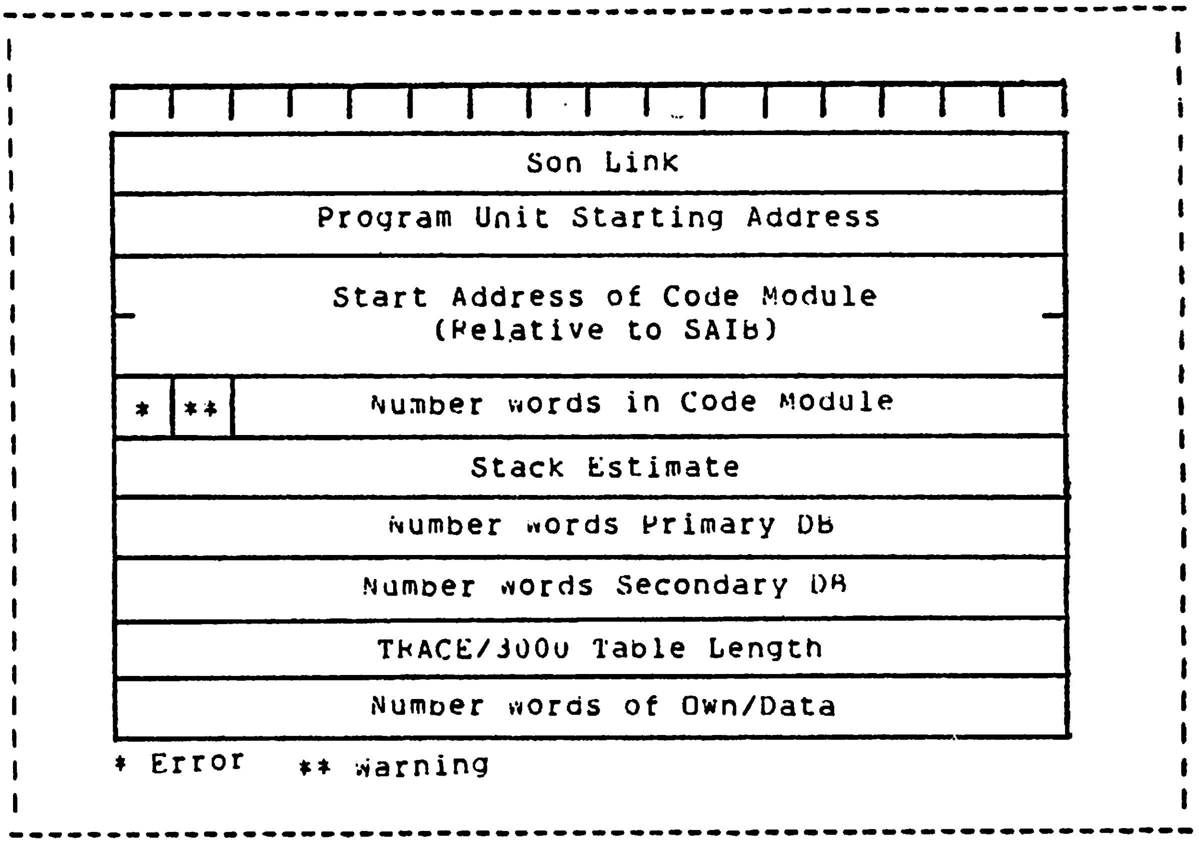
Table 3.4.2.2 -- Primary Outer Block Fields

| Field | Name | Contents |
|---|---|---|
| P(0) | SONL | son link of the entry |
| P(1) | PUSA | program unit starting address (address within the code module of the entry point) |
| P(2),P(3) | SAC | starting information block file address of the code module (this address is relative to SAIB; see section 3.3) |
| P(4).(0:1) | ERROR | set if program unit contains a fatal error |
| P(4).(1:1) | WARN | set if program unit contains a non-fatal error |

Table 3.4.2.2 -- Primary Outer Block Fields (cont.)

| Field | Name | Contents |
|---|---|---|
| P(4).(2:14) | CODELEN | number of words in the object code module |
| P(5) | STACKEST | an estimate of the number of words of stack needed by the program unit |
| P(6) | PDB | the number of words of primary DB allocated by this program unit |
| P(7) | SDB | the number of words of secondary DB allocated by this program unit |
| P(8) | TRCLEN | number of words in a table used by TRACE/3000 |
| P(9) | DATALEN | number of words in secondary DB reserved by DATA (FORTRAN) or OWN (SPL) declarations |
| p(10) to end of entry | | header information block for the program unit |

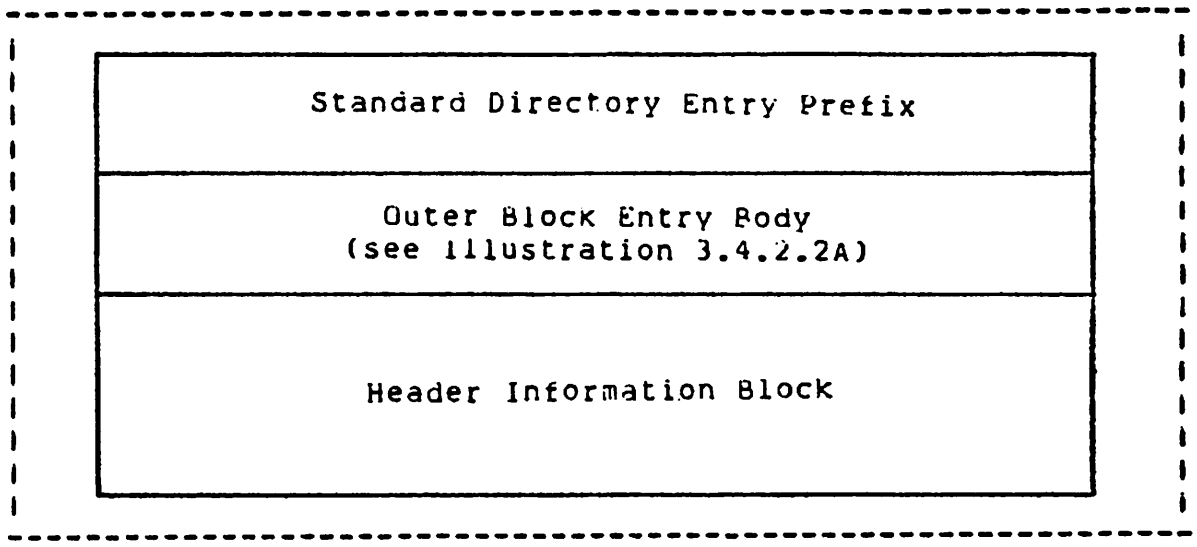
SDB and DATALEN are not the same thing. DATALEN refers to the number of words in a 'secondary DB array. Associated with each program unit is an area of secondary DB space. This area includes such things as the FORTRAN logical units table, format strings (those referenced in a read statement and containing 'H' specifications must be globally located, to retain their values), own/data variables, and the like. Included in DATALEN is only that amount of storage to be allocated for own/data variables--this portion of the secondary DB storage allocated by a program unit is referred to as the secondary DB array. The following are not included in DATALEN or SDB: the FORTRAN logical units table, TRACE/3000 tables, and common arrays. SDB does include the number of words used by globally located format strings.



Son Link
Program Unit Starting Address
Start Address of Code Module (Relative to SAIB)
*
**
Number words in Code Module
Stack Estimate
Number words Primary DB
Number words Secondary DB
TRACE/3000 Table Length
Number words of Own/Data
* Error
** Warning

Illustration 3.4.2.2A - O.B. and Proc. Entries Body

The first ten words of this entry are shown in illustration 3.4.2.2A. The format of primary block entries as a whole is shown in illustration 3.4.2.2B.



Standard Directory Entry Prefix
Outer Block Entry Body (see Illustration 3.4.2.2A)
Header Information Block

Illustration 3.4.2.2B -- Primary Outer Block Entry

## 3.4.2.3. Secondary outer block entries

Only a single word is appended to the standard prefix for secondary outer block type directory entries. This word indicates the location of the entry point in the code module associated with the parent directory entry. The word is given as a word-offset from the beginning of the code module, and so is analogous to PUSA, described in 3.4.2.2.

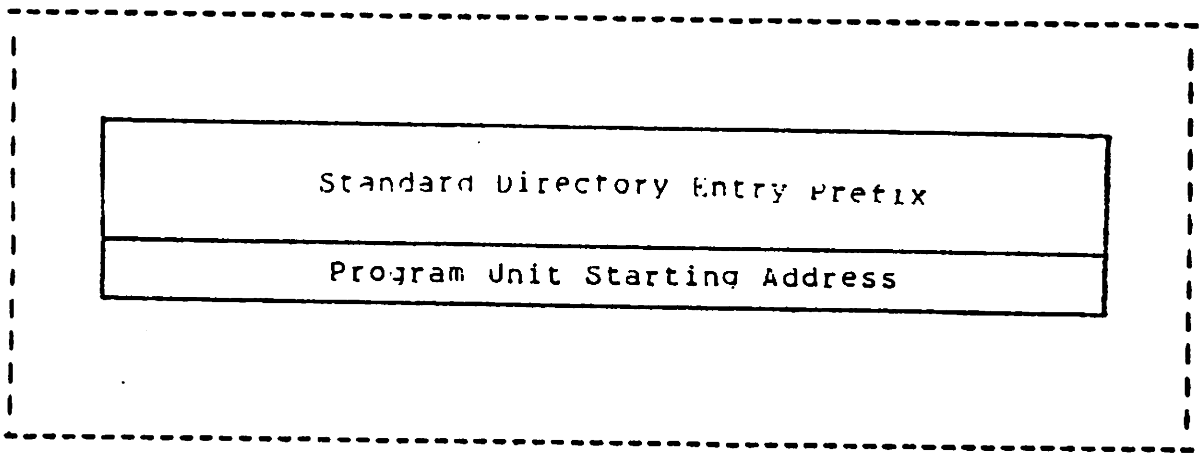

Standard Directory Entry Prefix
Program Unit Starting Address

Illustration 3.4.2.3 -- Secondary Outer Block Entry

## 3.4.2.4. Primary procedure entries

Primary procedure type directory entries append to the standard prefix exactly the same information as is appended by primary outer block entries, with one exception. Between DATALEN and the header information block, a parameter information block is inserted.

## 3.4.2.5. Secondary procedure without parameters entries

The secondary procedure without parameters directory entry type appends to the standard prefix only a single word. This word contains the address of the entry point to the code module associated with the parent entry. The address is a word-offset from the beginning of the code module, and so is analogous to PUSA, described in 3.4.2.2.

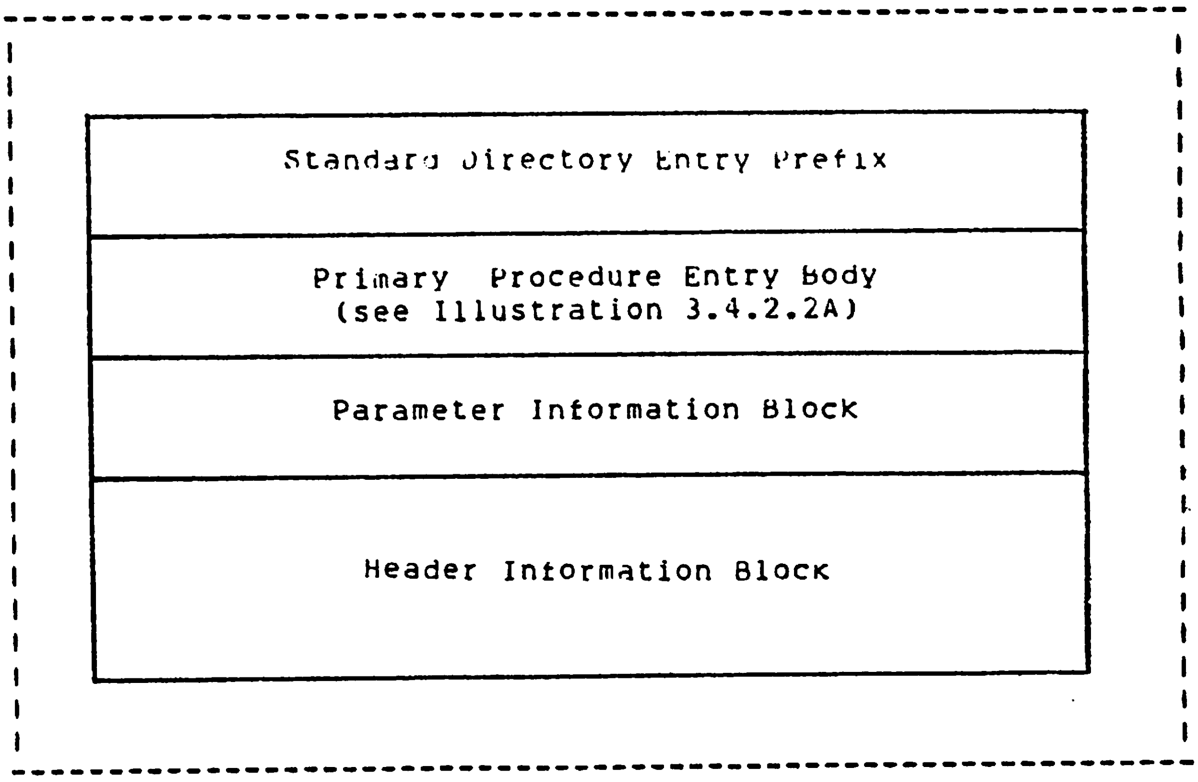

Standard Directory Entry Prefix
Primary Procedure Entry Body
(see Illustration 3.4.2.2A)
Parameter Information Block
Header Information Block

Illustration 3.4.2.4 -- Primary Procedure Entry

```
---------------------------------------------------------------
|                                                             |
|    ------------------------------------------------------   |
|    |                                                    |   |
|    |          Standard Directory Entry Prefix           |   |
|    |                                                    |   |
|    |----------------------------------------------------|   |
|    |           Program Unit Starting Address            |   |
|    ------------------------------------------------------   |
|                                                             |
---------------------------------------------------------------
```

Illustration 3.4.2.5 -- Sec. Procedure, No Parms., Entry

### 3.4.2.6. Interrupt procedure entries

After the standard prefix, interrupt procedure type directory entries append five words. These five words are followed by a header information block. The proper interpretation of the five words has not yet been determined.

### 3.4.2.7. Block data entries

A block data type directory entry appends after the standard prefix a number of subentries. Each subentry contains information for one block of common. (As far as the segmenter is concerned, every block of common is named. The name "COM'" is used to refer to blank common.) There is no explicit indication of the number of subentries present. This must be deduced from the subentries themselves, and from the number of words in the entry as a whole.

The first word of a subentry gives the number of words in the common block. Following this is a name field giving the name of the common block. Beginning in the word immediately following the name field, there is a header information block for the subentry. Thus, in a manner of speaking, the common block name and length are prepended to the relevant header information block.

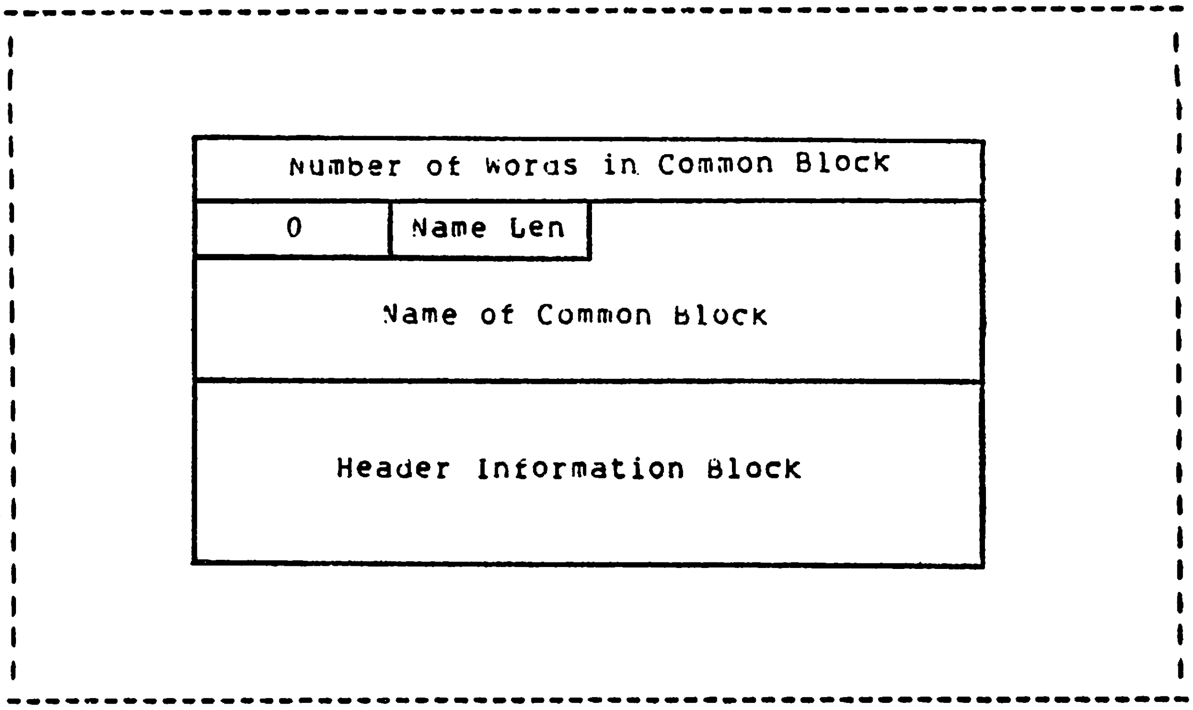


Illustration 3.2.4.7A -- Block Data Subentry Format

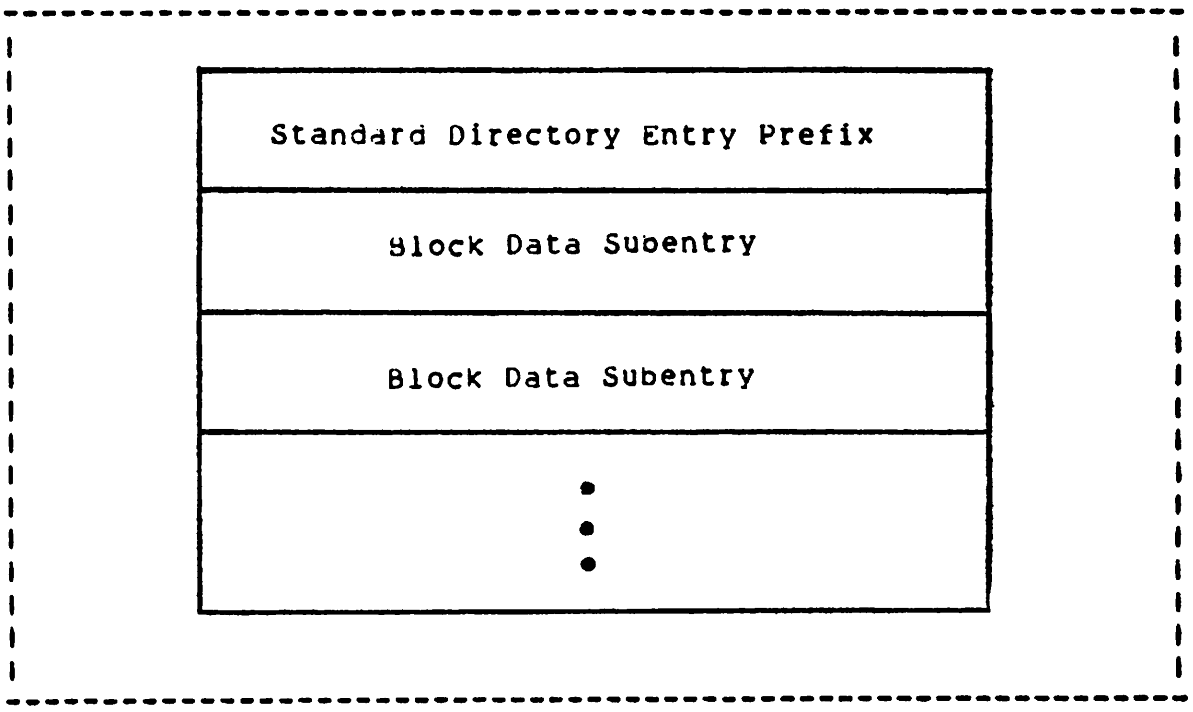


Illustration 3.2.4.7B -- Block Data Entry Format

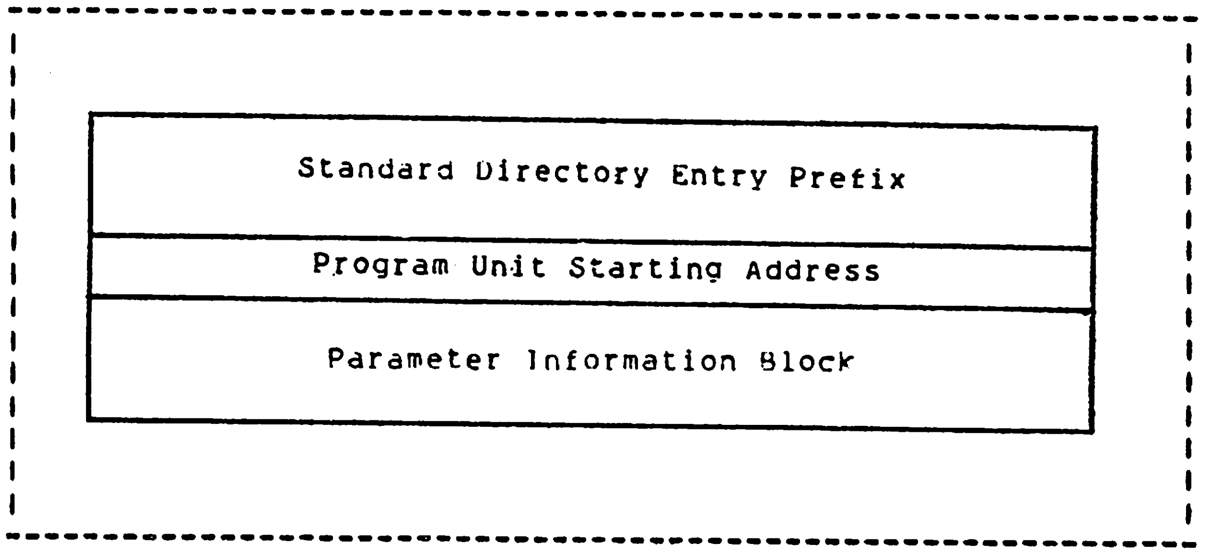

Standard Directory Entry Prefix
Program Unit Starting Address
Parameter Information Block

Illustration 3.4.2.8 -- Sec. Procedure, with Parms. Entry

### 3.4.2.8. Secondary procedure with parameters entries

One word, and then a parameter information block, are appended to the standard prefix in secondary procedure with parameters type directory entries. The word placed between the prefix and the PIB contains the address of the entry point to the code module associated with the parent entry. This address is given as a word-offset from the beginning of the code module.

### 3.4.3. Header information blocks

In a USL file in MPE, a 'header' is an entry in the information block of the file which provides information necessary for relocating a program unit, and for binding it with other program units. The various types of headers which are possible are discussed in section 3.5. In this section, the header information blocks found in directory entries are discussed. HIB's are used to provide information about the number, types, and lengths of headers associated with a directory entry.

A header information block is divided into header information sets. Each HIB is a more or less distinct entity. Any number of header information sets (including zero) may be included in any header information block. There is no explicit indication of the number of header information sets are in a header information block. This must be deduced from the HIB itself, and from the number of words in the directory entry as a whole. A HIS begins with a word which specifies in its (1:15) field the number of 'header descriptor words' that are present in the HIS. The (0:1) field of this word is set to zero unless this HIS is the last HIS of the HIB. In this case, (0:1) is set to one. This word is followed by a double word file address which is relative to SAIB (see 3.3). This address points to the first word of the of the first of the actual headers corresponding to the HIS. (All headers corresponding to a given HIS must be contiguous in the information block. See section 3.5.)

In the third and following words of a HIS are header descriptor words. There is one descriptor word for each header associated with the HIS, and the descriptors are in the same order as the headers themselves in the information block. A header descriptor has only three fields. The first, (0:1), is unused, and should be set to zero. The second, (1:10), gives the length of the associated header in words. (The length of headers is thus limited to a maximum of 1023 words.) The third, (11:5), gives the number of the type of the header. Header type numbers are presented in section 3.5.

## 3.5. The information area

The information block in a USL file contains all header entries. All addresses in the directory which refer to the information block are relative to SAIB (see 3.3). Because of this, the entire information block may be moved up and down in the file, changing only a few fields of record zero. Record boundaries are not recognized in the information block, but it should nonetheless begin on a record boundary.

### 3.5.1. Code modules

A code module is a special sort of header. It has no associated header descriptor word, it may be longer than the normal maximum of 1023 words, and it is never explicitly included in any HIS (see 3.4.3). It may, however, be placed anywhere within the headers associated with a HIS. The starting address of the code module, SAC (see 3.4.2) must be used to detect the presence of the code while sequentially processing the headers. If the code module is not needed when detected, it may simply be skipped. It is, as are all headers of a single HIS, contiguous with both the preceding (if any) and following (if any) headers.

The code module contains for the most part finished code, ready to be placed into a program. There can be many exceptions to this, however, depending on other headers associated with directory entries associated with the code module. There are various linear lists and relocatable addresses in the code module itself which are used by these other headers. The relevant lists and addresses will be discussed below together with the appropriate headers.

## 3.5.2. Information headers

There are twelve types of information headers. They are numbered as follows:

```
     0  null (a garbage header)
     1  PCAL, LLBL, or program unit PB address
     2  PB address
     3  own/data variable (for address correction)
     4  secondary DB initializations
     5  a table for TRACE/3000
     6  variables declared GLOBAL
     7  variables declared EXTERNAL
     8  primary DB declarations and initializations
     9  common (accomplishes only address correction)
    10  FORTRAN logical units
    11  globally located formats
```

These numbers are used in header descriptor words. (Header descriptor words were introduced in section 3.4.3.) Every header begins with a header descriptor word which describes it. The format of these descriptors is as follows:

```
    (0:1)   reserved, should be set to zero
    (1:10)  the length of the header in words
    (11:5)  the number of the header type
```

Although all headers begin with a descriptor word, each is thereafter highly individual. Each type is described in a separate subsection below.

All of the headers associated with a given HIS must be contiguous within the information block. The directory gives only the file address of the first word of the first header of any HIS. If the headers are not contiguous, it will not be possible to locate them in the file.

## 3.5.2.0. Null headers

A null header is a garbage entry. It simply takes up as much space as indicated in the header descriptor word. It has no significance to the program unit with which it is associated.

## 3.5.2.1. PCAL headers

PCAL headers provide all information needed to link the program unit to external program units. It actually has three functions, as follows: to make PCAL patches, to make LLBL patches, and to make procedure PB relative address patches. It is structured as indicated in table 3.5.2.1.

Table 3.5.2.1 -- PCAL Headers

| Field | Contents |
|---|---|
| P(0) | header descriptor word |
| P(1) | word offset into code module to the first word of a linked list of references to the program unit described in the header; each word in the list in the code module has the following format:<br>.(0:1) 0=patch in a PCAL instruction, 1=patch in an LLBL instruction<br>.(1:1) 0=patch as indicated by .(0:1) 1=patch in PB relative address of the program unit<br>.(2:14) link to next list item (this is a self-relative backwards pointer; the list terminates with a zero pointer) |
| P(2) | a name field, giving the name of the external program unit (P(2).(0:4) is unused, and should be set to zero) |
| P1 | following the name field is a parameter information block |

### 3.5.2.2. PB address headers

This header provides a means of patching words in the program unit which contain PB relative addresses. After the header descriptor word, the header is simply a series of pointers, each of which is a word-offset into the code module. (The number of these pointers must be deduced from the length of the header as a whole.) In each word in the code module thus pointed to, the compiler must place a PB relative address. This address will be corrected by the MPE segmenter at prepare time by adding to it the PB relative address of the first word of the program unit.

### 3.5.2.3. Own/data headers

At compile time, the run time address of an own or data variable is not known. It is assigned at prepare time. The MPE segmenter solves this problem by requiring the compiler to place in the code module a pointer to the variable. This pointer will of course then be part of the code at run time. The compiler initializes this pointer to the offset into the program unit's secondary DB array assigned by the compiler to the variable. At prepare time, the segmenter will add to this value the DB offset of the program unit's secondary DB array, thereby providing the code at run time with the correct pointer value.

After the header descriptor word, the entire header consists of pointers each of which is a word-offset into the code module. (The number of pointers must be deduced from the length of the header.) Each points to a location which is to be patched at prepare time. The high order bit of the pointer determines whether a byte or a word pointer is being initialized. If .(0:1)=1, then the contents of the code module word specified by the word offset in .(1:15), and the correction added at prepare time, are byte offsets. If .(0:1)=0, they are word offsets. (It is believed that the high order bit of the code module word pointed to is also interpreted in this way. That is, if either high order bit is on, either in the header pointer or in the code module word, then the address is to be a byte address.)

## 3.5.2.4. Secondary DB initial values headers

This header may be used to place initial values into the program unit's secondary DB array. The word which follows the header descriptor word gives the offset into the secondary DB array at which the first of the given initial values is to be placed.

The third word of the header has two fields. The .(0:1) field determines whether a byte initialization or a word initialization is to be performed. If .(0:1)=1, then the second word of the header is a byte offset, and the fourth word of the header is a byte count giving the length of the initial values in the header. In this case, the initial values begin in the fifth word of the header and continue for as many bytes as the fourth word indicates. If .(0:1)=0, then the second word of the header is a word offset. In this case, the initial values begin in the fourth word, and continue to the end of the header.

The .(1:15) field of the third word gives a replication factor. The initial values specified in the header will be placed in successive locations in the secondary DB array as many times as indicated by this field. Thus, if the initial values are "xxyxx" and the replication factor is 2, then "xxyxxxxyxx" will be placed into the secondary DB array, beginning at the location specified in the second word of the header.

## 3.5.2.5. TRACE/3000 header

This header provides information for use at run time by TRACE/3000. After the header descriptor word, there is a word pointing to a linked list in the code module. After this, beginning in the third word of the header, and continuing to the end of the header, is data which is believed to be initial values of some sort. No further information is available as of this writing about this header type.

## 3.5.2.6. Global variable headers

It is possible to declare a variable GLOBAL in one program unit, EXTERNAL in another, separately compiled program unit, and have the MPE segmenter resolve all references to the variable. (SPL/3000 is the only Hewlett Packard language allowing explicit declaration of GLOBAL or EXTERNAL attributes.) Following the header descriptor word is a data descriptor word, which gives the type and structure of the variable. The fields of this data descriptor are as follows:

| | |
|---|---|
| .(0:4) | the mode of the variable (0=null, 1= value, 2=reference) |
| .(4:6) | the variable's structure (0=simple variable, 1=pointer, 2=array) |
| .(10:6) | the type of the variable (0=null, 1= logical, 2=integer, 3=byte, 4=real, 5=double, 6=long, 7=complex, 8=label (passed SPL fashion), 9=character (as in FORTRAN/3000), 10=label (passed in in FORTRAN/3000 fashion), 11=any) |

In the left byte of the third word of the header is the run time DB relative address of the variable. (Global storage address assignments for primary DB are normally made by a compiler while compiling an outer block, and are not in any way relocated by the segmenter.) The .(8:4) field of the third word is reserved and should be set to zero. .(12:4) contains the length of the name of the variable, in bytes. The name itself begins in the left byte of the fourth word, and continues for as many bytes as necessary. The name is always an integral number of words in length, and so a byte is sometimes wasted.

## 3.5.2.7. External variable headers

A variable declared EXTERNAL is to be matched at prepare time with a variable declared GLOBAL in some other program unit. The first word of the header is of course a header descriptor word. The second word is a data descriptor word, which has the format described in section 3.5.2.6. Following the second word is a name field. The .(0:1) bit of the first word of the name field is a 'trace' bit. If it is on, it indicates that the variable may be traced by TRACE/3000 at run time. .(1:3) is reserved, and should be set to zero.

If the trace bit is on, then in the word immediately following the name field is an offset into the TRACE/3000 symbol table. If the trace bit is off, this offset is not present.

Following the name field, and the TRACE/3000 symbol table offset, if present, is a series of pointers, each of which is an offset into the code module. Each points to the first of a list of instructions to be patched with the address of the appropriate GLOBAL variable. Each of the instructions to be patched must be a memory reference instruction, since GLOBAL variables will always reside in the primary DB area. The address fields of the instructions to be patched (the right byte in memory reference instructions) serves as the link field for the list. The links are self-relative backward pointers. Each list is terminated by a zero pointer.

There is no explicit indication of the number of pointers in the header. This must be deduced from the length of the header.

### 3.5.2.8. Primary DB headers

For the purposes of the MPE segmenter, primary DB words are classified into word pointers, byte pointers, and data. After the descriptor word in this header there is a series of words, each of which is divided into eight two-bit fields. All these fields, in order of occurance, correspond to primary DB locations. The first it for DB+0, the second for DB+1, and so on. The values of the fields are interpreted as follows:

| | |
|---|---|
| 0 | the initial value is not an address |
| 1 | the initial value is not an address |
| 2 | the initial value is a word address which points to the secondary DB area |
| 3 | the initial value is a byte address which points to the secondary DB area |

Initial values in the header that are addresses are relative to the beginning of the program unit's secondary DB area. The entry, after the array of two-bit-field words, contains initial values. There must be PDB (see section 3.4.2.2) two-bit fields, and PDB initial values.

There may be a slack word between the two-bit-field array and the initial values. Because of this, the initial values should always be accessed from the end of the header. That is, if P is an integer pointer to the last word of the header, then P(-(PDB-1)) accesses the first initial value.

Normally, only an outer block program unit would make use of this header type. Non-outer block program units should not be allocating primary DB storage, and the value of PDB for them should be zero.

## 3.5.2.9. Common variable headers

The MPE segmenter allocates secondary DB storage for all common blocks. In order for a program unit to access a variable in common, it must use this header. For each common variable referenced in one of these headers, the MPE segmenter will allocate a pointer in the primary DB area, and properly initialize it to point to the common variable. Specified instructions will be patched with the address of this pointer.

Following the header descriptor word is an integer which gives the length in words of the common block to which the header applies. Beginning in the third word is a name field, giving the name of the common block to which the header applies (blank common is named "COM'"). The .(0:4) field of the name field is reserved and should be set to zero.

Beginning in the word immediately following the name field is a series of variable descriptors. There is no explicit indication of the number of variable descriptors in the header. This must be deduced from the header's length and contents. Table 3.5.2.9 gives the format of variable descriptors.

It must be noted that if the trace bit (P(0).(1:1)) is not on, then the displacement into the TRACE/3000 array (P(2)) is not included. It is simply omitted, and the list heads move up to fill in its place.

Table 3.5.2.9 -- Variable Descriptor Formats

| Field | Contents |
|---|---|
| P(0).(0:1) | 0=DB pointer is to be of type word, 1=DB pointer is to be of type byte |
| P(0).(1:1) | 'trace bit'; 0=variable will not be traced by TRACE/3000 at run time, 1=variable may be traced |
| P(0).(2:14) | the number of lists of instructions which are to be corrected (there are this many list heads later in the varible descriptor) |
| P(1) | the displacement within the common block of the variable |
| P(2) | displacement within a TRACE/3000 array of information about the variable (NOTE: this field is present only if the trace bit (P(0).(1:1)) is set; otherwise it is completely omitted) |
| P(3) to P(2+P.(2:14)) -or- P(2) to P(1+P.(2:14)) | the list heads of the lists of instructions to be patched; each list head is an offset into the code module to the first word of a list (the lists are formed the same as the code module lists used by EXTERNAL variable headers, described in section 3.5.2.7) |

## 3.5.2.10. FORTRAN logical units table headers

This header indicates which FORTRAN logical units are referenced by the program unit. The MPE segmenter will construct the FORTRAN logical units table from the information contained in FLUT headers. After the header descriptor word there are exactly seven words. These words contain a bit map, in which the first bit corresponds to logical unit number zero, the second to logical unit 1, the third to LU 2, and so on. If a bit is on, the corresponding logical unit will be included in the FLUT table at run time. The bits are numbered from left to right. The 'left-most' word is the one which occurs nearest to the header descriptor word. Legal LU's range from 1 to 99, inclusive.

## 3.5.2.11. Format headers

Formats which include an 'H' specification, and are referenced in a READ statement, must be globally located to retain between calls to the program unit values read into the 'H' specification. This header allows that. It contains a format string which is to be placed in the secondary DB area.

After the header descriptor word is a word which gives a word offset into the code module. The code module word thus indicated is the first of a list of words to be initialized at prepare time with the DB relative address of the format string. Within the list in the code module, the .(2:14) field of each member of the list is a self-relative, backward pointer to the next list element. A link of zero terminates the list. If the .(0:1) field of such a code module word is set on, then the DB relative pointer placed into that word is to be a byte address; if .(0:1)=0, then the DB relative pointer placed into the code module word is to be a word address. The pointer placed into the word will point at run time to the beginning of the format string.

The third word of the header gives the length, in bytes, of the format string. The fourth and following words, as many as necessary, contain the format string itself.

:EOF:

# TITLE: HP 3000/OPTIMIZING ON-LINE PROGRAMS

AUTHOR: ROBERT M. GREEN
ROBELLE CONSULTING LTD.
#130-5421 10TH AVENUE
DELTA, B.C. V4M 3T9
CANADA (604)943-8021

I have identified five general principles which help in optimizing the performance of on-line programs:

* Make each disc access count.

* Maximize the value of each terminal input.

* Minimize the run-time program size.

* Avoid constant demands for execution.

* Optimize for the common events.

## FIRST PRINCIPLE: MAKE EACH DISC ACCESS COUNT

Disc accesses are the most critical resource on the HP 3000. The system is capable of performing about 30 disc transfers per second, and they must be shared by system processes (spooling, console operator, etc.), memory management and user programs. (This rate can be increased to 58 per second under the best circumstances, and can degrade to 24 per second when randomly accessing a large file.) Another interesting fact is that a 4096-word transfer takes about the same overhead as a 128-word transfer. Therefore, it is better to read 4096 words in one transfer than to read 128 words 32 times. Another point to remember is that IMAGE database transactions require a lot of immediate disc accesses (from a DBUPDATE, which does one disc write, to a multi-key DBPUT that may require ten or more disc reads/writes).

Some of the operations that consume extra disc accesses on the HP 3000 are:

Increasing the number of keys in a detail dataset, thus causing IMAGE to access an extra master dataset on each DBPUT. Also, making a field a key value means that a DBDELETE/DBPUT is required to change it (which is 10 times slower than a DBUPDATE).

Increasing the program stack size by 8,000 bytes, thus causing the MPE memory manager to perform extra swapping disc accesses to find room in memory for the larger stack.

Improperly segmenting an active program, causing many absence traps to the memory manager to bring the code segments into main memory.

Defining a database or KSAM file with overly large blocksize, thus forcing each user terminal to access a large extra data segment that must be swapped in and out of main memory. (Note: the trade-offs will change when [if?] IMAGE is revised to use shared buffers.)

NOBUF Disc Accesses

When designing your next on-line application, see if there is some way that a random disc file can be used instead of an IMAGE dataset or a KSAM file; Then open that file with NOBUF and access it via directed reads and writes to specific blocks. Normally, when you open a file, the program is assigned an extra data segment to hold the buffer space for the file. Transfers between the file and the program are always done through this extra data segment. When the program requires a record, MPE first checks to see if the record is already in the extra data segment buffers; if so, it is merely transferred from the extra data segment to the user stack. If the block containing the desired record is not in the buffers, MPE issues a read against the disc to bring the block into main memory.

Although this sounds very clever and efficient, it has one major flaw: the extra data segment itself can be swapped. This means that in order to do any file access on a busy system, it may be necessary to read the extra data segment into memory before accessing the data in the disc file. On a heavily loaded system, this could cause a large number of unnecessary disc transfers. NOBUF access does away with all this by providing a direct interface between the user program and the disc files. Blocks are transferred to and from the user stack and the disc without any intervening buffer area. NOBUF is the fastest way to use random disc storage from a user program.

The user program must provide its own buffer space in the stack and call for transfers of data via the block number within the file. When multi-record access is used, it is possible to transfer multiple blocks at a time. The user is responsible for determining which block contains the record that he desires and where within the block the record is located. Simple subroutines can be written to handle this transformation.

A typical use for this kind of file is as a data entry transaction file. As the operator enters the data, it is buffered in the stack until a block is full; then the entire block is written to the disc in one operation. For even better throughput and response time, you might try writing the blocks to the disc with the NO-WAIT option; when this is used, MPE overlaps the write operation to the disc with your next print and read from the terminal. Without NO-WAIT, your program would be suspended until the disc write could be completed by MPE.

(Warning: Be certain that you know when the end-of-file is updated; otherwise, you might find that you have an empty transaction file when the system crashes. I suggest that you move the end-of-file to the limit of the file at the start of the day by writing a null entry in the last record position and then closing the file.) When the transaction file is full (or the day ends), a batch program is used to put the transactions into the final IMAGE dataset or KSAM file. This job can be done in low priority or after hours.

## SECOND PRINCIPLE: MAXIMIZE THE VALUE OF EACH TERMINAL READ

Each time a program reads from the terminal, it is suspended and may be swapped out of memory. When the operator hits the carriage return key, the input operation is terminated, and the process must be dispatched again. In order to dispatch a process, MPE must ensure that the data stack and at least one code segment are resident in main memory. If the process is going to access the disc, it may be necessary to make an extra data segment resident also. Unless the computer has enough main memory so that no user segments are ever swapped out, it is desirable to have the process get as much work done as possible before it suspends for the next terminal input (and is swapped out again).

The simplest way to program data entry applications is by prompting for and accepting only one field of data at a time. This is also the least efficient way to do it. The user data stack must be made resident every time the user hits 'return'. (Therefore, the less often the user hits 'return', the larger your stack can afford to be.) Since it is inefficient, fast response time cannot be guaranteed, and the resulting delays are very irritating to operators. They can never work up any input speed, because they never know when the computer is ready for the next input line. If response time and throughput are the only considerations, it is always preferable to keep the operator typing as long as possible before hitting the 'return' key. Multiple transactions should be allowed per line, with suitable separators, and multiple lines should be allowed without a 'return'.

## THIRD PRINCIPLE: MINIMIZE THE RUN-TIME PROGRAM SIZE

The HP 3000 is an ideal machine for optimizing because of the many hardware features available at run-time to minimize the effective size of the program. Even quite large application systems (6000 lines of code) can be organized to consume only a small amount of main memory at any one time. Each executing process on the HP 3000 consists of a single data segment called the "stack", and one or more extra data segments for system storage, such as file buffers. Although a process is always executing some code in a code segment, the code does not properly belong to the process, since one copy is shared by all processes in the system. If a program is to be executed by several terminals, most optimizing should be directed to the data areas (which are duplicated for each user).

Large programs which are not logically segmented make it harder for the memory manager to do its job, and thus cause many disc accesses to be consumed in swapping. In an extreme case, the system can almost be brought to a complete standstill by a very large program executing on many terminals at the same time. The articles listed in Appendix A provide strategies and examples for code segmentation. To simplify a complex problem, follow these guidelines: 1) put initialization, termination and error handling in separate code segments; 2) minimize the number of calls across segment boundaries at run-time; 3) remain in a segment as long as possible; 4) keep segments small (2K-8K words), but don't use too many segments (MPE has a limited overall code segment capacity).

Many more terminals can be supported on a given system if data stack sizes are kept modest (ex: less than 6000 bytes on a 192K-byte machine), and if the code is properly segmented. The simplest way to keep the stack small is to make all data variables local (DYNAMIC in COBOL) and to use global storage only for buffers and control values that must be accessed by all subroutines. The reason that this is so effective is that dynamic local storage is allocated on the top of the stack when the subroutine is entered and is released automatically when the subroutine is left. This means that if the main program calls 3 large subroutines in succession, they all reuse the same space in the stack. The stack need only be large enough for the deepest nesting situation.

Since the amount of dynamic stack space that will be required by the program is not known at the start of execution, the 3000 provides methods (both automatic and programmatic) to expand the dynamic area. Whenever a stack overflow occurs, MPE automatically allocates more space (up to a MAXDATA limit). Unfortunately, there is no automatic mechanism for reducing the stack size when that additional space is no longer needed. The user application program can include a check in the mainline and shrink the stack back down to the desired size after returning from an oversize subroutine. (See Appendix B for an example.)

The other major way to reduce the size of a data stack is to ensure that constant data items (such as error messages, screen displays) are stored in the code segment instead of the data segment. Since they are never to be modified, there is no logical reason that they must be in the data stack. By moving them to the code segment, one copy of them can be shared by all users running the program. In SPL, this is done by including =PB in a local array declaration or MOVE'ing a literal string into a buffer. In COBOL, constants can be moved to the code segment by DISPLAY'ing literal strings in place of declared data items. In FORTRAN, both FORMAT statements and DISPLAY'ed literals are stored in the code.

## FOURTH PRINCIPLE: AVOID CONSTANT DEMANDS FOR EXECUTION

The HP 3000 is a multiprogramming, virtual memory machine that depends for its effectivenss on a suitable mix of processes to execute. Although the sizes of the segments to be swapped have an effect on performance, this is dependent upon the frequency with which memory residency is demanded. Given the same overall configuration and application program sizes, the system supports many more terminals if each one only executes for 5 seconds every 30 seconds than if each one must execute for 60 seconds at a time. Each additional terminal that is demanding continuous execution (in high priority) makes it harder for the operating system to provide proper response time to all other terminals.

Here are some examples of the kind of operation that can destroy response time if performed in high priority:

EDIT/3000, a GATHER ALL of a 3000-line source file.

QUERY, serial read of 100,000 records

SORT, sorting 50,000 records.

COBOL, compiling on 4 terminals at once.

All of these operations should be done in low priority in batch STREAM jobs. These jobs can even be created dynamically by on-line programs. In this way, the on-line user still requests the high-overhead operation, but the system fulfills the request when it has the time.

## FIFTH PRINCIPLE: OPTIMIZE FOR THE COMMON EVENTS

In any application where there is a large variation between the minimum and maximum load that a transaction can cause, the program should be optimized around the most common size of transaction. In any application with a large number of on-line functions, it is likely that a small number of functions are used most of the time. In this case, all optimization efforts should be aimed at the commonly used functions and other functions left as is. This is especially feasible on the HP 3000 because of code segmentation and dynamic stacks.

If N is the average number of records in a transaction (i.e, the number of lines on a customer order, maximum is 500), then allow room in your stack for N records. If you only allowed for one record, then there would be unneeded disc thrashing. Alternatively, if you provide room for the maximum number, then the data stack is much larger than actually needed most of the time. Having a larger data stack may cause the system to overload, eliminating the benefits of keeping the records in your stack. It is recommended that room in the stack be allowed for slightly more than the average number, and that a NOBUF disc file be used to "page" this area on very large transactions.

## OPTIMIZING CASE STUDY #1: QEDIT

QEDIT is a high-speed, low-overhead source program editor developed by Robelle Consulting Ltd. The primary objective of QEDIT is to provide the fastest possible editing with the minimum possible system load. Other objectives include conservation of disc space, similarity to EDIT/3000 in command syntax, ability to recover the workfile following a system crash or program abort, and increased programmer productivity.

### QEDIT and the First Principle: Disc Accesses

In order to reduce disc accesses, QEDIT had to eliminate the overheads of the TEXT, KEEP and GATHER ALL commands of EDIT/3000. These three operations have the most drastic impact upon the response time of the other users. QEDIT attacks the problem of KEEPs by providing an interface library that fools the HP compilers into thinking that a QEDIT workfile is really a "card image" file. As a result, it is never necessary to KEEP a workfile before compiling it. Since KEEPs are rarely used, most TEXTs are eliminated. TEXT is only needed when you want to make a backup or duplicate copy of an existing file. It was anticipated that most users would maintain their source files exclusively in workfile format, so the TEXT'ing of workfiles was optimized (by using NOBUF, multi-record techniques) to be at least 4 times faster than a normal TEXT of a card image file. The GATHER ALL operation is slow because it makes a copy of the entire workfile in another file. QEDIT renumbers up to 12 times faster by doing without the file copy.

Disc accesses during interactive editing (add, delete, change, etc.) were minimized by packing as many contiguous lines as possible into each disc block. The resulting workfile is seldom over 50% of the size of a normal KEEP file or 25% of the size of an EDIT/3000 K-file (workfile). Most QEDIT users maintain all of their source programs in workfile form, since this saves disc space, simplifies operations (there need only be one copy of each version of a source program), and provides optimum on-line performance.

QEDIT always accesses its workfile in NOBUF mode and buffers all new lines in the stack until a block is full before writing to the disc. Wherever possible in the coding of QEDIT, unnecessary disc transfers have been eliminated. For example, the workfile maintains only forward direction linkage pointers, which reduces the amount of disc I/O substantially. Results of a logging test show that reducing the size of the workfile and eliminating the need for TEXT/KEEP reduces disc accesses and CPU time by 70-90%.

## QEDIT and the Second Principle: Terminal Accesses

QEDIT allows multiple commands per line, plus multiple data lines per data line input (i.e, you can enter 7 lines of text without hitting 'return'). All interaction with the terminal is done directly through the READX and PRINT instrinsics.

## QEDIT and the Third Principle: Program Size

QEDIT is a completely new program, written in highly structured and procedurized SPL. The resulting program file consists of 7 code segments of 1780 words (decimal) each. Only two code segments are required for most editing commands, while the most common function (adding new lines) requires only one code segment most of the time.

QEDIT uses a minimum data stack and no extra data segments. All error messages are contained in the code, isolated in a separate code segment that need not be resident if you make no errors.

## QEDIT and the Fourth Principle: Constant Demands

Most QEDIT commands are so fast that they are over before a serious strain has been placed on the host machine. For example, a 2000-line source program can be searched for a string in four seconds. For those operations which still are too much load, QEDIT provides the ability to switch priority subqueues dynamically. In fact, the system manager can dictate a maximum priority for certain operations such as compiles or TEXT and KEEP commands.

## QEDIT and the Fifth Principle: Common Events

The entire design of QEDIT is based on the observation that program editing is not completely random. When a programmer changes line 250, he is more likely to require access to lines 245 through 265 next than he is to lines 670 through 710. This observation dictated the design of the indexing scheme for the QEDIT workfile.

There are many examples of optimizing for the most common events in QEDIT: the blocksize will hold about a screenful of data lines, built-in compiler, fast renumbering command (600 lines per second) in place of a GATHER command, faster TEXT'ing of workfiles than KEEP files (4 to 7 times faster).

## Results of Applying the Principles to QEDIT

In less than 7 seconds, QEDIT can text 1000 lines, renumber them and search for a string. Commands are 80% to 1200% faster than EDIT/3000, program size is cut in half, and disc I/O and CPU time are reduced by up to 90%.

In order to measure performance, an editor-callable "procedure" was written that calculates the elapsed time (using TIMER intrinsic) and the processor time (PROCTIME intrinsic) between events. QEDIT measured faster than EDIT/3000 by these percentages:

| | |
|---|---|
| Renumber | 1204% faster |
| List to printer | 115% faster |
| Find string | 613% faster |
| Change | 645% faster |
| Keep | 82% faster |
| Text from keepfile | 44% faster |
| Text from workfile | 733% faster |

The more efficient the programming of an operation, the less system resources it consumes. MPE provides a "logging" facility to record the resource usage of programs for later analysis. Both QEDIT and EDIT/3000 were used to perform a typical program maintenance change (edit, compile, correct errors). According to the logging statistics, QEDIT reduced overhead by these percentages:

| | |
|---|---|
| Physical disc transfers | 93% reduction |
| Disc space required | 87% reduction |
| cpu time | 72% reduction |
| Program size | 63% reduction |
| Total data space | 53% reduction |
| Data stack size | 43% reduction |

Programming of QEDIT began in March 1977 and user-site testing in September 1977. At the present time (September 1978), there are 20 QEDIT user installations. QEDIT shows what can be accomplished by applying all of these optimizing principles in the design of one system. In any given application system, it may not be possible to take advantage of all five principles; but to whatever extent they can be applied, the resulting system will provide better service than it would have.

For more information on QEDIT, contact me directly:

Robert M. Green
Robelle Consulting Ltd.
#130-5421 10th Ave.
Delta, B. C. Canada
V4M 3T9
Phone: (604) 943-8021

## OPTIMIZING CASE STUDY #2: APPLYING PAYMENTS

In this accounts receivable system, 24,000 invoices per month are posted to 10,000 customer accounts. The number of unpaid items per customer varies from one or two (a lot of accounts) to 500 (a few major accounts). The A/R are maintained on an open-item basis. That is, the invoices appear on the customer's statement each month until they are matched up with a payment and considered reconciled. About 200-300 cheques are posted to the database each day. The problem is to allow the A/R clerks to "apply" the payments to the proper invoices in the cheapest possible manner. Certain other constraints exist: the machine is a Series I, only dumb terminals are to be used, and the system is already supporting about 17 terminals and seems fully loaded.

The computer system cannot tolerate the overhead to scan down the chain of records for each account (DBGETs) and print them on the screen. There is too heavy a load already. In addition, the software would have to skip over (i.e., get and ignore) a large number of paid invoices to find the unpaid ones.

### A/R and the First Principle: Disc Accesses

A/R uses a database to index entries by account and sort them by date, but allows no on-line updates to the database. They are too slow and too hard to control (recover/balance). Updates are only allowed by sequential batch programs.

Each clerk is provided with a transaction disc file for her "ledger", containing copies of her active accounts (14 entries/block). She also has a printout that shows each account and gives its location in the file. The disc file and associated lineprinter report are prepared in batch. The user accesses this file in on-line mode and converts the entries into database transactions.

The transaction file is accessed in NOBUF mode and contains only unpaid invoices. All on-line activity is done into this file, then, at night, those entries which have been marked in the file for application are retrieved from the database and updated.

### A/R and the Second Principle: Terminal Accesses

The user input syntax allows (but does not require) many individual instructions to be entered in each input line. This example applies a payment to seven invoices and writes a small adjustment against one invoice:

```
        /1ABDEFGH,A(1010-1010,7.50,C)
```

A major design problem was how to refer to the items that are on the customer's account. The invoice number is too long for efficient data entry (and subject to errors). A sequence number could have been assigned to each entry on an account. However, invoices are not paid in sequence; eventually, the sequence numbers would be as large as the invoice numbers. A quick calculation showed that the time required to assign new sequence numbers was prohibitive (because of DB inefficiencies). The scheme settled upon assigned relative position numbers to each unpaid item on a dynamic basis, but these numbers are not actually stored in the database. In order to shorten input, an alphanumeric code was used (A,B,C...Z,A1,...). In retrospect, a pure numeric sequence number might have been better because of the input speed of numeric keypads.

A/R and the Third Principle: Program Size

A/R is written entirely in SPL/3000. Stack sizes are modest (2K-3K decimal or less), and only one disc block is kept in the stack. SPL procedures were created to simulate a mini-file system for the transaction files. The procedures do all deblocking and disc input/output. This simplified coding of the three major programs.

A/R and the Fourth Principle: Constant Demands

There are a few special transactions that can take up to a minute, but they are very rare and can be ignored. Most transactions are very short, and all data is available in memory, or is one disc read away.

A/R and the Fifth Principle: Common Events

This principle was applied heavily to A/R. The most common event is to apply a payment that came in yesterday to an old invoice(s). Also, most accounts have less than 10 outstanding items. Therefore, this system anticipates the next day's requests by creating the batch-file/printout of all the accounts with an unapplied payment. For those accounts that require attention, but have no payment, the clerk loads them into her batch file on-line (rare). The blocksize was picked so that most accounts could fit in one block.

The transaction to apply a payment is:

```
>10117A67/1AGH
           ^Invoice lines
         ^Payment number
       ^Disc location of the account (from printout)
  ^Account number
^Prompt Character
```

Since starting production, we have discovered that usually the account # and location # entered is just the one that sequentially follows the last one. Therefore, the system will someday be changed to allow entry of * for next account.

When converting from the manual system to a pure on-line computer system, the ability to write notes on the customer's account card was lost. After a few months, we found ourselves under heavy pressure to create new types of transactions in the system to handle the many special cases that arose (paid twice, overpaid, short-paid, took credit note twice, etc.). The original design only allowed for four types of transactions: invoice, payment, adjustment and journal entry (a large adjustment with a unique number assigned for control purposes). Rather than clutter up the design, we added the ability to write multi-line comments for any journal entry. With these comments, the A/R clerk can now communicate directly with the customer's accounts payable clerk to explain the problem in English. Since the comments are kept in a separate dataset, indexed by the unique journal entry number, there is no additional overhead on ordinary transactions.

Basis for future expansion: since most accounts pay in a simple pattern, the computer will (in batch) pre-apply the payments when creating the transaction file. Then the operator need only take action if the computer has selected incorrectly.

Results of Applying the Principles to A/R

The application maintains 10,000 accounts with 24,000 invoices per month, using two ADM-3 terminals on a CX-3000 with 15-19 other terminals doing less optimized things. A/R staff has been reduced from seven people to two. At the same time, the three terminals used for program development were switched to QEDIT. Response time has actually improved on old applications. At the same time, 2 terminals have been added to the system.

For more information on this example of applying optimizing principles, contact the user site directly:

Gary Nordman, Manager of Systems Development
Malkin & Pinton Industrial Supplies
325 East Fifth Avenue
Vancouver, B.C. Canada
V5T 1H6

## APPENDIX A: REFERENCES ON HP 3000 OPTIMIZING

[1] Transaction Processing on the HP 3000 Series III.
Some HP field System Engineers have this
internal HP document which describes the internal
workings of these software products:
IMAGE
KSAM
FILE SYSTEM
COBOL
FORTRAN

[2] COMMUNICATOR No. 14.
Page 87, Block/Page mode problems.

[3] COMMUNICATOR No. 12.
Segmentation in COBOL

[4] COMMUNICATOR No. 5.
Segmentation for Maximum Efficiency
of System-Type Programs.

[5] JOURNAL-3000 Vol 1, No. 6.
KSAM vs. IMAGE
HP 3000 with Front-End Processor
FORTRAN Optimization

[6] JOURNAL-3000 Vol. 1, No. 5.
QEDIT, Quick Program Editing,
Small Appetite for Computer Time.

[7] JOURNAL-3000 VOL. 1, No. 4.
Using Extra Data Segments.
Common Programming Errors with IMAGE/3000.

[8] CONTRIBUTED LIBRARY, Vol I/II.
IDEA Program
IDEAII Program
RESP Program
IDLE Program
PROGSTAT PROGRAM

[9] CONTRIBUTED LIBRARY, Vol III.
SOO Program
DBREBILD Program

[10] CONTRIBUTED LIBRARY, "Vol IV".
DBSTAT Program
DBCHANGE Program

[11] SCRUG MEETING LIBRARY, March 1978.
FASTER - An essay on writing programs for greater efficiency.
OVERLORD (See also SOO.)
DBSTAT - Internal efficiency of master datasets.
SHOWVM - Shows virtual memroy.
STACKOPT - Stack optimizing routines.

[12] SCRUG MEETING NOTES, March 1978.
Extra Data Segments and Process Handling
Operator Utilities

[13] INTERNATIONAL USERS MEETING, 1977.
KSAM (see extra data segment size, load times)
IMAGE for the advanced User
Optimizing FORTRAN IV/3000
RPG/3000 Programming Optimization
Data Entry Techniques
Segmentation
MPE II Measurement and Optimization
MPE C Measurement and Optimization

[14] INTERNATIONAL USERS MEETING, February 1975.
Software Optimization Through Segmentation

[15] INTERNATIONAL USERS MEETING, May 1974.
Program Performance

[16] CCRUG MEETING MINUTES, May 9, 1978.
IDEA Program
DBDRIVER Program

[17] PERFORMANCE GUIDELINES/SERIES III (HP 5953-0533).
Note the extra load of synchronous terminals(p.9) and the dramatic increase in the number of terminals supported when a simple file is used instead of IMAGE/DEL/COBOL.

[18] SPL/3000 FOR COMMERCIAL APPLICATIONS, EFFICIENCY WITH EASE OF MAINTENANCE.
Report available from Robelle Consulting Ltd.

## APPENDIX B: SHRINKING THE STACK SIZE

The following SPL code can be added to any program that calls a lot of procedures (or subprograms in COBOL) in order to dynamically optimize the size of the data stack.

### CHECKSTACK LIBRARY SUBROUTINES

1.Checks for excessive dynamic stack space after subroutine calls and adjusts the stack size; consists of three routines that are intended to be called from the mainline of an application program that uses many subprograms with varying data requirements.

2.Contents: CHECKSTACK1, CHECKSTACK2, CHECKSTACK3.

3.Parameters: WORKSPACE, 20 bytes of global data in the calling program. The proper COBOL definition is:

```
          01 CHECK-STACK-SPACE .
            05  PRINT-RESULTS-FLAG  PIC  S9(3) COMP VALUE N.
      *         N=0(NO PRINTOUT),1(ON TERMINAL),
      *           2(ON CONSOLE),3(ON BOTH).
            05  FILLER              PIC X(18).
```

HOW TO USE CHECKSTAK:

1. Add the WORKSPACE to the data division of your program and set the desired PRINT'FLAG value(see step 4).

2. At the start of program execution:

```
CALL "CHECKSTACK1" USING CHECK-STACK-SPACE.
```

This call should occur once at the start of the mainline. The purpose is to record the size of the dynamic stack area before any subprograms are called. This size is determined by STACK=XXXX in the :PREP or :RUN commands.

3. After returning from each subprogram call:

```
CALL "CHECKSTACK2" USING CHECK-STACK-SPACE.
```

This call compares the current dynamic stack area with the initial size and if it is over 512 words larger (1024 bytes), reduces it back to the initial.

4. At the end of program execution:

```
CALL "CHECKSTACK3" USING CHECK-STACK-SPACE.
```

This call prints statistics on stack usage on either $STDLIST or the console or both. Format is:

```
      GLOB99   STK99   #OK99   AVG99    #ADJ99    SIZ99

      ^ Global stack size in decimal words
              ^ Initial dynamic stack size
                     ^ Number of "OK" subprogram calls
Average stack size per "OK" call^
     Number of times stack was adjusted ^
            Average stack size per adjusted call ^
```

Start with the default value for STACK= (about 800) and a large value for MAXDATA (20000). If all of the subprogram calls are adjusted (i.e., OK=0), increase the STACK= value. Try to find a value where most of the subprogram calls execute without having to shrink the stack afterwards, but not so large that there are no large subprograms left ot adjust.

INSTALLATION OF CHECKSTAK:

1. Type the following SPL source code into the system using QEDIT or EDIT/3000 and create a source file.

2. Compile the source file and make corrections until there are no errors:

   ```
   :SPL SOURCE
   ```

3. When you have a successful compile, save the USL file, using this command:

   ```
   :SAVE $OLDPASS,USLSPL
   ```

4. Either COPY the segment called "LIBSEG1" into the USL file of your application program (using the :SEGMENTER commands AUXUSL and COPY) or add it to an SL (:SEGMENTER or :SYSDUMP).

```
$CONTROL LIST,SUBPROGRAM,MAIN=LIBSEG1,ERRORS=9
BEGIN

PROCEDURE  CHECKSTACK1 ( BUF ) ;
    INTEGER ARRAY   BUF;
BEGIN
<< DEFINE STRUCTURE/USE OF BUF >>
DOUBLE ARRAY DBUF (*) = BUF;
DEFINE
  PRINT'FLAG    = BUF#       ,INITIAL'SPACE   = BUF(1)#
 ,SHRINK'COUNT = BUF(2)#   ,OK'COUNT        = BUF(3)#
 ,OK'SPACE   = DBUF(2)#     ,SHRINK'SPACE    = DBUF(3)#
 ;
INTEGER Z,Q;

IF NOT ( 0<=PRINT'FLAG<=3 ) THEN
  PRINT'FLAG := 1; <<DEF>>

PUSH (Z,Q);  Z:=TOS; Q:=TOS;

INITIAL'SPACE := Z - Q;
BUF(2) := 0;
MOVE BUF(3) := BUF(2),(7);

END; <<CHECKSTACK1 >>


PROCEDURE CHECKSTACK2 ( BUF ) ;
    INTEGER ARRAY   BUF;
BEGIN
<< DEFINE STRUCTURE/USE OF BUF >>
DOUBLE ARRAY DBUF (*) = BUF;
DEFINE
  PRINT'FLAG    = BUF#       ,INITIAL'SPACE   = BUF(1)#
 ,SHRINK'COUNT = BUF(2)#   ,OK'COUNT        = BUF(3)#
 ,OK'SPACE   = DBUF(2)#     ,SHRINK'SPACE    = DBUF(3)#
 ;
INTEGER Z, Q, STACKSIZE;
INTRINSIC ZSIZE;

PUSH ( Z,Q ); Z:=TOS; Q:=TOS;
STACKSIZE := Z - Q;
IF STACKSIZE > (INITIAL'SPACE + 512) THEN BEGIN
  ZSIZE ( Q + INITIAL'SPACE );
  SHRINK'COUNT := SHRINK'COUNT + 1;
  SHRINK'SPACE := SHRINK'SPACE + DOUBLE(STACKSIZE);
  END
ELSE BEGIN
  OK'COUNT := OK'COUNT + 1;
  OK'SPACE := OK'SPACE + DOUBLE(STACKSIZE);
  END;

END; <<CHECKSTACK2>>
```

```
PROCEDURE CHECKSTACK3 ( BUF) ;
    INTEGER ARRAY   BUF;
BEGIN
<< DEFINE STRUCTURE/USE OF BUF >>
DOUBLE ARRAY DBUF (*) = BUF;
DEFINE
  PRINT'FLAG    = BUF#      ,INITIAL'SPACE   = BUF(1)#
 ,SHRINK'COUNT = BUF(2)#   ,OK'COUNT        = BUF(3)#
 ,OK'SPACE   = DBUF(2)#     ,SHRINK'SPACE    = DBUF(3)#
 ;
INTEGER ARRAY P(0:38);
BYTE ARRAY P'(*)=P;
INTEGER TERMINAL;
INTEGER GLOBAL'SPACE;
INTRINSIC PRINT,PRINTOP, ASCII,DASCII,WHO,DATELINE;

IF PRINT'FLAG = 0 THEN RETURN;

IF PRINT'FLAG=2 OR PRINT'FLAG=3 THEN BEGIN
  << PRINT IDENTIFYING MESSAGE ON THE CONSOLE >>
  P:="  "; MOVE P(1):=P,(38);
  MOVE P :="CHECK-STACK:";
  WHO(,,,P'(12),P'(21),P'(30),,TERMINAL);
  MOVE P'(39) := "ON";
  ASCII(TERMINAL,10,P'(42));
  P'(20):=P'(29):=".";
  PRINTOP(P,-46,0);
  END;

P:="  "; MOVE P(1):=P,(38);
MOVE P:="GLOB";
PUSH (Q);
GLOBAL'SPACE := TOS;
ASCII(GLOBAL'SPACE,10,P'(4));
MOVE P'(10):="STK";
ASCII(INITIAL'SPACE,10,P'(13));
MOVE P'(19):="#OK";
ASCII(OK'COUNT,10,P'(22));
MOVE P'(28):="AVG";
DASCII(OK'SPACE/DOUBLE(OK'COUNT),10,P'(31));
MOVE P'(37):="#ADJ";
ASCII(SHRINK'COUNT,10,P'(41));
MOVE P'(47):="SIZ";
DASCII(SHRINK'SPACE/DOUBLE(SHRINK'COUNT),10,P'(50));

IF PRINT'FLAG=2 OR PRINT'FLAG=3 THEN
  PRINTOP(P,-56,0);
IF PRINT'FLAG=1 OR PRINT'FLAG=3 THEN
  PRINT(P,-56,0);

END; << CHECKSTACK3 >>
END <<LIBRARY >> .
```

PETE FRATUS
FUTURA SYSTEMS, INC.

# ON-LINE TAPE LIBRARY

## I. OVERVIEW

TAPELIB provides interactive access to information on an site's tape library. It was written to be simple, easy-to-use and low in resource usage. Normally, the program would be run as a job to provide the operator with access through the REPLY command. It can also be run in a session by System or Account Managers. Information on the single sequential file can be read and updated. The Console Operator and System Manager have access to all commands; Account Managers use a subset of the commands. Mass changes can be made with EDITOR and listings in different sequences can be made with SORT.

Commands allow the user to find tapes by number, account, account and group, or tape label. The operator can find scratches to be used and can update the library to reflect the use. The user can scratch tapes by number or find it first by account then scratch it. The Operator and System Manager can scratch any tape. Account Managers can only scratch tapes belonging to their accounts.

TAPELIB was designed to be used on a system with 300 to 400 small and large reels which are used to backup the system and store files off-line. The features of TAPELIB reflect this design but allow tailoring to any type of system.

Three of four digit tape numbers can be used. The first character of the tape number can designate the reel size by using different ranges of numbers for each size or different letters for each size. For example, we use 100 and 200 reel numbers for seven inch reels. Eleven inch reels are numbered 400 and 500. This system will handle a large variety of numbering schemes such as:

a. Large reels are 0001 thru 1999, small reels are 2000 thru 3999.
b. Large reels are three digits with first digit even, small reels are three digits with the first digit odd.
c. Large reels only, three digits, many different ranges ( use 4 digit tape numbers with a lead-

ing zero ).

Accounts and groups are eight characters each. Tape labels can be up to sixteen characters. Each tape has an associated creation date and sequential reel number, i.e. 1 of 3, 2 of 3, 3 of 3.

When a tape is scratched, a scratch code and scratch date are added to the entry. The data file is ASCII with no integer or packed fields, so editing can be done easily.

## II. THE DATA FILE

The file is fixed, ASCII and 64 bytes. The fields are:

| FIELD | POS. | LEN. | CONTENTS |
|---|---|---|---|
| TAPENO | 0 | 4 | TAPE NUMBER, LEFT JUSTIFIED |
| ACCT | 6 | 8 | ACCOUNT NAME |
| GRP | 15 | 8 | GROUP NAME |
| LBL | 24 | 16 | TAPE LABEL |
| CREATEDATE | 41 | 6 | TAPE CREATION DATE |
| REELNO | 49 | 2 | REEL NUMBER OF TAPE IN SET ( 12 = REEL 1 OF 2 ) |
| SORN | 55 | 1 | SCRATCH TAPE ( S OR BLANK ) |
| SCRATCHDATE | 57 | 6 | SCRATCH DATE ( or BLANK ) |

The data file is created with EDITOR. SET LENGTH=64.

## III. OPERATING PROCEDURES

SESSION MODE:

The program is initiated with :RUN TAPELIB.
Commands available to the user are

| | |
|---|---|
| ALPHA(AL) | - LISTS TAPES BY ACCOUNT AND GROUP |
| EXIT(EX) | - TERMINATES THE PROGRAM |
| HELP(HE) | - LISTS COMMANDS AND DESCRIPTIONS |
| LABEL EQUALS(LE) | - LISTS ALL TAPES WITH GIVEN LABEL |
| NUMBER(NU) | - LISTS A TAPE BY ITS NUMBER |
| SCRATCH(SC) | - SCRATCHES A TAPE |

JOB MODE:

The CP entry point must be used to provide a console request, so :RUN TAPELIB,OP.
Commands available to the Console Operator or System

Manager are

```
 LARGE SCRATCH(LS) - LISTS 4 LARGE SCRATCH TAPES
 SMALL SCRATCH(SS) - LISTS 4 SMALL SCRATCH TAPES
 USE(US)           - SAVES SCRATCH WITH NEW LABEL
```

## IV. CHANGES WITH EDITOR

When a large number or changes are necessary the ASCII file can be TEXTED with EDITOR, even while TAPELIB is running in JOB mode. A sample EDITOR session would look like this:

```
:EDITOR                    (Note: lines shortened to fit page.)

/TEXT TAPELIB.DATA,UNN
/FIND "123";MODIFY
    24 123  SYS    PUB   FULL-DUMP     780601  12 S 780620
MODIFY   24
123  SYS    PUB    FULL-DUMP     780601  12 S 780620
     RFUTURA TYPE   DAILY STORE   780624
123  FUTURA TYPE   DAILY STORE   780624

/FIND "448";M
   221 448  HABAND MAIL  EAST-LIST     780522
MODIFY  221
448  HABAND  MAIL  EAST-LIST     780522
                                        RS 780624
448  HABAND  MAIL  EAST-LIST     780522        S 780624

/FQ1
/F "400"
   180    400  NESTER   PUB           780413
/GQ 180/LAST TO 1000
/CQ FIRST/179 TO 2000
/GQ 2000/LAST TO 100
/GQ 1000/LAST TO 400
/LIST 400.
   400    400  NESTER   PUB           780413
/LIST 123
   123    123  FUTURA   TYPE     DAILY STORE      780624
/K TAPELIB.DATA,UNN

/EXIT

 END OF SUBSYSTEM
:
```

The above EDITOR was used to find a few tapes by number and change them. Then the tapes were renumbered with the GATHER command. Since our tapes are numbered 100-188 and 400-490, each group was gathered high and gathered again low to make the line number and tape number match. A USE file containing these commands helps the operator text, renumber, and list a few lines to verify the match. From that point, modifying a tape by number will get the needed line for that tape: see LIST 123 above.

## V. USING SORT TO GET LISTINGS

It has been helpful to have alphabetic and numeric listing of the tape library file each day. We use two sequences:

```
1.  SORN     - major      2.  TAPENO
    ACCOUNT
    GROUP
    LABEL
    DATE
    REEL     - minor
```

The first sequence places all scratches at the end of the list then groups the rest of the list by account, group, etc. The second provides a numeric list by tape number for verifying the library. Output from the sort goes directly to the printer. The SORT parameters are:

```
1.   !FILE TAPELIBA;DEV=LP
     !RUN SORT.PUB.SYS
     >INPUT TAPELIB.DATA
     >OUTPUT *TAPELIBA
     >KEY 55,1,BYTE;6,45,BYTE
     >END

2.   !FILE TAPELIBN;DEV=LP
     !RUN SORT.PUB.SYS
     >INPUT TAPELIB.DATA
     >OUTPUT *TAPELIBN
     >KEY 1,4,BYTE
     >END
```

## VI. INSTALLING TAPELIB

To install TAPELIB use this procedure:

Number your tape library with three or four digit numbers. Using EDITOR, key in one entry for each tape; use the format in Section II.

Use GETFILE to retrieve TAPELIB.PUB and TAPELIB.JCB from the contributed library.
Do any of the tailoring listed in Section VII.
Run the program or stream the job.

## VII. TAILORING TAPELIB

The current but changable limitations of TAPELIB are:

| | |
|---|---|
| name of the tape library file | "TAPELIB.DATA.LIB3" |
| length of the tape number | = 3 |
| number of scratch tape numbers in sorted array | = 100 |
| starting number for small reels | "1" and "2" |
| starting number for large reels | "4" and "5". |

Each of these parameters can be changed by altering the DEFINES and EQUATES in the source file. The imposed limits are:

| | |
|---|---|
| name of tape library file | limited only by MPE |
| length of tape number | 3 or 4 |
| number of scratches | unlimited |
| starting number for small or large reels | up to 5 different digits |

To make changes, find the block of EQUATES and DEFINES in the source.
Replace the existing LIBNAME with your library file name.

Replace the existing contents of LARGENO and SMALLNO with the starting digits of your library.

If your library uses 4 digit tape numbers, equate TAPENO-LEN to 4, TAPENODATELEN to 10, and SCRSIZE to 10 times the equated value of NOSCR.

If 100 scratches is too large or too small, equate NOSCR to the desired number.

Series I users need to set the conditional compile switch X1=ON to replace CLOCK and CALENDAR with CHRONOS.

# SYSTEM PERFORMANCE MEASUREMENT AND OPTIMIZATION

JIM SQUIRES, H-P Systems Engineer, Fullerton, Ca.
ED SPLINTER, H-P Systems Engineer, Los Angeles, Ca.

Obtaining optimum performance from a computer system is often critical to the success of an installation. This is particularly true today when data processing managers are being ask to produce more with little or no increase in their budget.

When most users comment about a system's performance they are really stating how well the system meets their expectations. This means that what is felt to be a performance problem might well turn out to be an expectation problem. If users fail to consider the strengths and limitations of the system while designing application programs great disappointment can result.

Fortunately performance measurement tools formerly used only at the factory have matured and are now being distributed to the field for SE use. With these tools now available at a point closer to the customer, performance problems are being addressed more quickly and in many cases with impressive results.

On the following pages is a representative report based on one of the machines where performance was judged by the users to be unsatisfactory. The report is presented to the customer during a meeting that usually lasts in the neighborhood of two hours. At that time attention is focused on the areas where improvements in performance can be realized as quickly as possible. At all times it must be remembered that the object is to optimize the combination of the computer system and its users not just the system.

## INTRODUCTION

The purpose of this report is to present an analysis of the performance of your HP3000. This information should help in answering questions such as:

1. Is system response time being restricted by CPU, memory or disc contention?
2. Are any programs unexpectedly dominating the mix?
3. Can additional applications and/or users be added to the system without adversely affecting response?
4. What might be done to improve system performance?

The data presented here was obtained from a trace of system activity collected with the event monitoring facility. A record of each occurence of selected events is written to tape for subsequent analysis. For this report the primary events monitored are associated with memory management activity, process dispatching and IO device activity.

Since data is collected with a monitor using software traps, the monitoring activity necessarily has an effect on the performance of the system. Experience indicates, however, that the results are skewed only slightly and in most cases is undetectable. In any case the information obtained gives one a far greater insight into the system's activity than is obtainable in any other way currently available.

Raw data is usually collected for a period of time much longer than that chosen for detailed analysis. On a heavily loaded system about 500,000 events are recorded on the tape each hour activity is monitored. Since detailed analysis is quite time consuming the tape is scanned for a general picture of the activity recorded. A 15-30 minute 'window' is then chosen for detailed analysis.

The operating system of any computer is designed to manage the system's resources, principally, the processor, main memory and disc resolving conflicts arising from competition amoung the community of users. When demand for any resource approaches the capacity, the management task becomes difficult causing system efficiency to decline.

Performance of a system has to be discussed in the context of the system workload. The programs which make up this workload can be characterized by the type and amount of system resources required for their execution.

In the following sections this report moves from the general to the specific in its investigation of the utilization of the three principal resources mentioned above. First, utilization from an overall point of view is discussed. Then a summary of information by program for all jobs and sessions is presented. Next is a detailed report for each program that was found to be a significant resource user. The section on conclusions and recommendations provides a summary of the significant bottlenecks in the system and suggests ways to improve system performance.

## DATA COLLECTION

Period Monitored: Mon, Jun 26, 1978 1:32 - 2:23pm

Total number of events recorded: 461,301

Window chosen for analysis: 1:53 - 2:13pm (1200 secs)

```
--------------------------
| Unless otherwise noted |
|  ALL TIMES IN SECONDS  |
|  ALL LENGTHS IN BYTES  |
--------------------------
```

| OVERALL RESOURCE UTILIZATION | SECTION II |
|---|---|

## OVERALL CENTRAL PROCESSOR UTILIZATION

A fairly good first approximation of how well a system is performing is given by noting the amount of time the CPU spends in each of four states:

1. CPU busy - the time during which some process was executing;
2. Waiting for swaps - the time during which the memory manager (MAM) is waiting for a disc I/O to complete and no other process has sufficient memory resources to run;
3. Waiting for disc I/O - the time during which a process other than MAM is waiting for a disc I/O to complete and no other process is requesting the CPU;
4. Idle - the time during which no process is requesting the CPU and no process is waiting for I/O to terminate.

| CPU STATE | Percent of Window | Total Mins |
|---|---|---|
| Busy | 74.57 | 14.9 |
| Waiting for swaps | 15.31 | 3.1 |
| Waiting for disc | 7.87 | 1.6 |
| Idle wait | 2.25 | 0.5 |

FIGURE 1-1. CPU usage during the window. The average CPU busy interval was 14 ms and the average idle time was 5 ms. These two figures were distorted by the program IDLE.

The CPU busy time can be broken down as follows:

| | |
|---|---|
| Memory manager | 10.36% |
| Other MPE processes | 2.00 |
| The program IDLE | ≈30.52 |
| Other user processes | 31.69 |

Note that the CPU was being used by the memory manager or was being held for swapping 25.67% of the time. This indicates that the memory manager is having considerable difficulty meeting the requests for main memory.

## OVERALL MEMORY UTILIZATION

As your system is presently configured the resident portion of MPE uses 84,464 bytes of memory leaving 439,824 bytes of 'linked' memory for swapping. The size of resident memory is, to some extent, controlled by responses to configuration questions while doing a SYSDUMP.

In analyzing the utilization of memory it is useful to note whether the allocation was for code or data and if for code whether it came from a program file or a segmented library (SL).

| Segment Type | Percent Memory | Average Alloc | Average Presence | Num of Swaps | Overlays per Swap |
|---|---|---|---|---|---|
| DATA | 30.75 | 3679 | 8.13 | 4912 | 0.741 |
| SL | 44.23 | 5272 | 12.47 | 3331 | 1.412 |
| PROGRAM | 8.46 | 4393 | 18.02 | 550 | 0.391 |
| Totals | 83.44 | 4327 | 10.41 | 8793 | 0.973 |

FIGURE 2-1. Memory allocation information. This data applies only to linked memory and does not include any segments allocated prior to the start of monitoring.

The principal concern here is how much swapping went on and how many segments currently in memory had to be overlayed to make room for the new one. The number of swaps shown here indicates a higher than desirable rate of swapping. There were over 7 swaps per second. It is possible to average about 30 disc I/Os per second on the HP3000. This means that about 25% of the maximum possible disc activity was used for swapping.

## OVERALL DISC ACTIVITY

For best performance, disc I/O requests should be evenly distributed over the available drives to reduce arm contention and seek times.

| Drive | Request Count | Percent of Total | Transfer Length | % Busy | Seconds Between Requests |
|---|---|---|---|---|---|
| 1 R | 11171 | 40.1 | 2924 | 25.65 | 0.107 |
| 1 W | 6613 | 23.7 | 3100 | 15.52 | 0.181 |
| 2 R | 1228 | 4.4 | 2816 | 2.80 | 0.974 |
| 2 W | 871 | 3.1 | 1887 | 1.54 | 1.378 |
| 3 R | 1596 | 5.7 | 2261 | 3.57 | 0.752 |
| 3 W | 808 | 2.9 | 1169 | 1.55 | 1.485 |
| 4 R | 1146 | 4.1 | 2244 | 2.19 | 1.045 |
| 4 W | 901 | 3.2 | 2003 | 1.65 | 1.330 |
| 12 R | 1094 | 3.9 | 3044 | 2.62 | 1.093 |
| 12 W | 607 | 2.2 | 1673 | 0.99 | 1.973 |
| 13 R | 1164 | 4.2 | 1632 | 2.15 | 1.027 |
| 13 W | 661 | 2.4 | 1388 | 0.93 | 1.809 |
|  | 27860 |  |  |  |  |

FIGURE 2-2. Global disc activity. The top line for each device applies to reads, bottom line to writes. During the window 74.35 million bytes of data were transferred between disc and memory. Swapping traffic accounted for 57.04 million bytes or 76.7% of the total. Each second an average of 23.22 disc I/O requests were made.

## OPERATING SYSTEM DISC REQUESTS

Certain user activities cause the system to access disc storage. It is useful to identify these activities and tabulate their respective I/O loads. The memory manager is almost always the system process which uses disc most heavily. All MAM requests are associated with swapping. The LOADER accesses program files and SL files to resolve external references when the :RUN command is issued. Other system processes which access disc memory include DEVREC (to verify signon information) and LOG (for system logging, if enabled).

| DRIVE | SYSTEM PROCESSES MAM | LOADER | OTHER | USER PROCS | TOTAL |
|---|---|---|---|---|---|
| 1 R | 7862 | 853 | 72 | 2384 | 11171 |
| 1 W | 5466 | 43 | 2 | 1102 | 6613 |
| 2 R | 65 | 1 | 123 | 1039 | 1228 |
| 2 W | | 2 | 43 | 826 | 871 |
| 3 R | 480 | 95 | 12 | 1009 | 1596 |
| 3 W | | 2 | 5 | 801 | 808 |
| 4 R | 168 | 74 | 31 | 873 | 1146 |
| 4 W | | 65 | 6 | 830 | 901 |
| 12 R | 111 | 26 | | 957 | 1094 |
| 12 W | | 24 | | 583 | 607 |
| 13 R | 108 | | 15 | 1041 | 1164 |
| 13 W | | | 29 | 632 | 661 |
| | 14260 | 1185 | 338 | 12077 | 27860 |
| | 51.18% | 4.75% | 1.21% | 43.35% | |

FIGURE 2-3. Operating system disc access requests. The top line for each device applies to reads and the bottom line is for writes.

## SYSTEM WORKLOAD BY JOB/SESSION | SECTION III

During the monitoring window of a system it is possible for each user to run one or more programs. Figure 3-2 shows which programs were run by each user. Also shown are those programs used by the operating system. Included for each user's program is the CPU, memory and disc resources used. Figure 3-1 contains information to help identify users.

The system programs listed are run on behalf of users without their knowledge or intervention. Normally, all of these programs make small demands on system resources. Since most of these programs are run at a much higher priority than user programs, their impact is quickly felt when they become heavily used. The fact that MAM used over 10% of the CPU is an immediate indication that the operating system is have trouble meeting the demand for memory. There is not enough real memory to efficiently handle all the requests.

Note the impact that the COBOL compile (#J5) had on the system. The combined compile and prep used 21% of the CPU and 28% of the memory during the 6 minutes that the operation took. When response times are a problem, COBOL compiles should be kept to an absolute minimum.

The data clearly shows the impact that the A4000 processes have. With only a couple of exceptions, those processes with more than 190 swaps are associated with your application program. The exceptions are significant since they include COBOL and the EDITOR. The EDITOR process during Session 41 ran 18 minutes using over 7% of the available memory and over 4% of the CPU during this time.

| JOBNUM | JIN | INTRODUCED | JOB NAME |
|---|---|---|---|
| #S29 | 20 | MON 1:33P | MANAGER.SYS |
| #S34 | 52 | MON 1:35P | A40,RON.FMS |
| #S35 | 40 | MON 1:35P | OSSUSER.FMS |
| #S36 | 23 | MON 1:37P | A40,OSSUSER.FMS |
| #S37 | 27 | MON 1:37P | A40,OSSUSER.FMS |
| #S38 | 26 | MON 1:37P | A40,OSSUSET.FMS |
| #S39 | 28 | MON 1:38P | A40,OSSUSER.FMS |
| #S40 | 25 | MON 1:40P | A40,OSSUSER.FMS |
| #S41 | 53 | MON 1:42P | GEORGE.FMS |
| #S48 | 32 | MON 1:47P | A40,OSSUSER.FMS |
| #S50 | 31 | MON 1:49P | A40,OSSUSER.FMS |
| #S52 | 30 | MON 1:50P | A40,OSSUSER.FMS |
| #S53 | 34 | MON 1:50P | A40,OSSUSER.FMS |
| #S55 | 22 | MON 1:52P | A40,OSSUSER.FMS |
| #S56 | 51 | MON 1:56P | ANNIE.FMS |
| #S57 | 23 | MON 1:57P | A40,OSSUSER.FMS |
| #S58 | 21 | MON 1:57P | OP.FMS |
| #S60 | 27 | MON 1:59P | A40,OSSUSER.FMS |
| #S61 | 46 | MON 1:59P | A40,OSSUSER.FMS |
| #S64 | 35 | MON 2:02P | A40,OSSUSER.FMS |
| #S65 | 40 | MON 2:03P | OSSUSER.FMS |
| #S67 | 50 | MON 2:04P | RONK.FMS |
| #S68 | 41 | MON 2:04P | A40,OSSUSER.FMS |
| #S71 | 55 | MON 2:12P | TOM.FMS |
| | | | |
| #J 1 | 10 | MON 1:34P | IDLE,DAN.FMS |
| #J 5 | 10 | MON 1:50P | WCOMPILE,ANNIE.FMS |
| #J 6 | 10 | MON 1:56P | MANAGER.SYS |

FIGURE 3-1. Session and job identification.

Figure 3-2 on the next two pages contains summary information about each program that was running anytime during the 1200 second window. The number of seconds that the program was observed is shown in column two. CPU usage is shown as a percentage of the seconds observed. Memory used is a percentage of memory available during the time observed. Column five indicates the average size of all segments (code and data) allocated for the program. The disc IO count in column six applies only to IOs associated with files opened by the program. The swap count includes all memory manager IOs caused by this program including the initial allocation of each segment. Overlays indicate how many segments already in memory had to give up memory when the average segment for this program was made present.

| J/S# | PROG | SECS SEEN | %CPU USED | %MEM | AVG SEG | DISC I/Os | SWAPS | OVER LAYS |
|---|---|---|---|---|---|---|---|---|
| SESSIONS | | | | | | | | |
| 29 | COMMANDS | 1197 | .06 | .37 | 3607 | 11 | 81 | .19 |
| 34 | SEGDVR | 182 | .33 | 1.21 | 2257 | 2 | 105 | .13 |
| | SEGPROC | 185 | 3.78 | 2.78 | 3698 | 657 | 78 | .69 |
| | COMMANDS | 1117 | .47 | .91 | 3080 | 300 | 281 | .30 |
| 35 | A4000 | 585 | 3.02 | 12.67 | 4769 | 260 | 535 | 1.39 |
| | COMMANDS | 617 | .05 | .15 | 3128 | 16 | 24 | .00 |
| 36 | A4004 | 241 | 1.26 | 3.00 | 4570 | 98 | 114 | .32 |
| | COMMANDS | 259 | .11 | .55 | 3015 | 16 | 32 | .00 |
| 37 | COMMANDS | 338 | .26 | .41 | 2957 | 18 | 56 | .34 |
| | A4000 | 338 | 3.16 | 12.09 | 4830 | 224 | 456 | 1.40 |
| 38 | A4004 | 1195 | 1.39 | 3.45 | 6177 | 528 | 278 | 1.66 |
| 39 | A4002 | 1197 | 2.62 | 9.68 | 5304 | 1124 | 559 | 1.35 |
| 40 | A4003 | 1200 | .53 | 1.94 | 5154 | 113 | 179 | 1.55 |
| 41 | EDITOR | 1119 | 4.14 | 7.35 | 3722 | 1601 | 538 | .40 |
| | COMMANDS | 1143 | .04 | .06 | 3241 | 14 | 39 | .33 |
| | EDITOR | 54 | 2.59 | 9.38 | 4214 | 68 | 49 | .63 |
| 48 | A4000 | 1188 | .66 | 4.67 | 5898 | 134 | 352 | 1.63 |
| 50 | A4001 | 1196 | .98 | 4.49 | 5403 | 405 | 308 | 1.24 |
| 52 | A4000 | 1165 | .93 | 4.37 | 5402 | 299 | 346 | 1.62 |
| 53 | COMMANDS | 1200 | .11 | .34 | 3227 | 70 | 79 | .22 |
| | A4004 | 1197 | .01 | .03 | 2841 | 0 | 13 | .15 |
| 54 | COMMANDS | 205 | .30 | .69 | 2691 | 24 | 60 | .25 |
| 55 | COMMANDS | 4 | 1.86 | 4.29 | 1130 | 3 | 4 | .00 |
| | A4003 | 626 | .90 | 3.38 | 4404 | 247 | 196 | 1.14 |
| 56 | FCOPY | 106 | .56 | 2.96 | 3688 | 7 | 48 | .90 |
| | FCOPY | 301 | .65 | 4.56 | 3890 | 15 | 122 | .52 |
| | COMMANDS | 852 | .35 | .54 | 2673 | 98 | 127 | .40 |
| 57 | COMMANDS | 29 | 1.22 | 4.11 | 2695 | 24 | 25 | .24 |
| | A4004 | 892 | .32 | .41 | 3437 | 138 | 61 | 1.02 |
| 58 | COPYOP | 195 | 1.33 | 3.67 | 3602 | 93 | 72 | .47 |
| | COMMANDS | 262 | .52 | 1.17 | 3150 | 42 | 59 | .85 |
| 59 | COMMANDS | 15 | 3.01 | 1.93 | 2445 | 28 | 16 | .56 |
| 60 | A4000 | 737 | 2.96 | 5.81 | 5219 | 1165 | 357 | 1.96 |
| | COMMANDS | 108 | .35 | 1.00 | 2634 | 24 | 27 | .04 |
| 61 | A4004 | 30 | 3.15 | 11.09 | 3207 | 12 | 43 | .46 |
| | COMMANDS | 75 | .78 | 2.08 | 2542 | 36 | 32 | .53 |
| | A4004 | 64 | 4.29 | 8.00 | 4270 | 139 | 58 | 1.00 |
| 62 | COMMANDS | 16 | 2.76 | 4.97 | 2595 | 28 | 26 | .65 |
| 63 | COMMANDS | 13 | 3.29 | 5.23 | 2402 | 28 | 30 | .27 |
| 64 | A4003 | 611 | 1.27 | 4.76 | 4896 | 298 | 211 | 1.21 |
| | COMMANDS | 26 | 1.50 | 3.47 | 2727 | 24 | 16 | .19 |
| 65 | COMMANDS | 37 | 1.16 | 5.31 | 2485 | 22 | 27 | .22 |
| | A4000 | 522 | 2.09 | 5.51 | 4041 | 272 | 323 | .80 |
| 66 | COMMANDS | 26 | 2.02 | 1.68 | 2634 | 28 | 31 | 1.00 |

FIGURE 3-2A. Summary information of each program seen during the window. COMMANDS refers to the command interpreter.

```
----------------------------------------------------------------------------
|                | SECS | %CPU |       | AVG  | DISC  |       | OVER |
| J/S#  PROG     | SEEN | USED | %MEM  | SEG  | I/Os  | SWAPS | LAYS |
|--------------------------------------------------------------------------|
|  67 EDITOR     |  219 | 10.13|  4.70| 4067 |   381 |    81 |  .32 |
|     COMMANDS   |  302 |   .37|   .81| 3146 |    46 |    65 |  .31 |
|     EDITOR     |  215 | 10.63|  6.51| 3948 |   437 |   101 |  .26 |
|  68 COMMANDS   |   28 |  1.14|  2.93| 2789 |    24 |    16 |  .13 |
|     A4003      |  316 |  2.77|  8.87| 4925 |   278 |   196 | 1.31 |
|  69 QUERY      |  275 |  2.13|  5.34| 5164 |   179 |   148 |  .97 |
|     COMMANDS   |  399 |   .33|  1.42| 3013 |    70 |   110 |  .18 |
|  70 COMMANDS   |  136 |   .54|  2.03| 2980 |    31 |    56 |  .50 |
|     FCOPY      |   69 |  6.16|  6.59| 3438 |    65 |    43 |  .93 |
|  71 COMMANDS   |   20 |  1.67|  4.19| 2813 |    26 |    25 |  .68 |
|     EDITOR     |   41 |  4.09|  4.90| 4158 |    67 |    48 |  .83 |
|                                                                          |
| JOBS                                                                     |
|                                                                          |
|   1 IDLE       | 1143 | 30.52|  1.38| 2421 |     9 |    26 |  .00 |
|   5 COBOL      |  299 | 16.47| 19.93| 8650 |   493 |   224 | 2.98 |
|     COMMANDS   |  358 |   .26|   .44| 3060 |    48 |    40 | 1.17 |
|     SEGPROC    |   69 |  4.44|  7.60| 3436 |   202 |    57 |  .91 |
|   6 COMMANDS   |   33 |  4.48|  6.07| 3297 |    58 |    42 |  .33 |
|   8 FORTRAN    |   86 | 10.81|  9.16| 3838 |   440 |    89 |  .58 |
|     COMMANDS   |  179 |   .60|  3.12| 3053 |    43 |    81 |  .44 |
|     FORTRAN    |  197 |  4.30|  9.79| 4513 |   426 |    68 | 1.28 |
|                                                                          |
| SYSTEM                                                                   |
|                                                                          |
|     PROGEN     |  287 |   .27|   .47| 4210 |    16 |    20 | 1.05 |
|     MAM        | 1200 | 10.36|   .00|    0 | 14260 |     0 |  .00 |
|     IOSYS      | 1199 |   .06|   .20| 2014 |     0 |    76 |  .01 |
|     IOMSG      | 1165 |   .04|   .27| 1880 |     0 |   133 |  .02 |
|     LOG        | 1198 |   .02|   .25| 2415 |    63 |    91 |  .07 |
|     UCOP       | 1200 |   .15|   .52| 1945 |     4 |   174 |  .10 |
|     DEVREC     | 1198 |   .11|   .47| 2424 |    68 |   121 |  .05 |
|     PRIMSG     | 1197 |   .02|   .05| 1196 |     0 |    29 |  .00 |
|     LOAD       | 1132 |  1.52|   .29| 4870 |  1185 |    45 |  .73 |
|     SPOOLER    |  443 |   .17|   .27| 3117 |    14 |    14 | 1.36 |
|     SPOOLER    | 1061 |   .72|  1.13| 4424 |   173 |   101 |  .52 |
----------------------------------------------------------------------------
```

FIGURE 3-2B. Summary information of each program seen during the window. COMMANDS refers to the command interpreter for the job or session.

Programs normally spend very little time actually using the CPU. Most of the time is spent waiting for some event to terminate. Three events usually account for the majority of the wait time.

(1) User requested I/O,
(2) Absent code or data segment,
(3) Human think time at a terminal.

Of course the third item usually doesn't apply to batch programs. In this case waiting for a higher priority process to give up the CPU is the third significant event.

Interactive programs may also be held up waiting for terminal output. This is caused by writing large amounts of information (i.e. large forms) to the screen in block mode. Adding more terminal buffers to the system configuration will sometimes help. Occasionally programs are seen with significant wait due to database or file locking.

Once the events dominating the wait time have been identified it may be possible to improve performance of individual programs and thus the system as a whole. When the CPU is the limiting resource, the solution is usually an additional computer or another more powerful.

When absent segments are responsible for most of the wait, performance can be improved by adding more real memory to the system. This condition can be caused by segments which are excessively large ( over 10,000 bytes). In this case reducing segment sizes may improve performance to a satisfactory level. The cost of modifying programs to accompolish this must be balanced against the cost of the required additional memory. Frequently adding memory is the most cost-effective solution.

User disc I/O wait time can normally be reduced only through reduction of I/O requests by the program. In a few instances moving files between drives to balance arm contention may help.

Locking waits can often be reduced by carefully rethinking where and when lock requests are issued. Applications locking multiple files (or databases) can make reduction of locking contention very difficult.

| PROCESS/ SESSION | SECONDS OBSERVED | USING CPU | ABSENT SEGMENT | USER DISC IO | FILE LOCK | TERM READ |
|---|---|---|---|---|---|---|
| | | % | % | % | % | % |
| A4000/35 | 585 | 3.0 | 21.1 | 1.6 | 2.3 | 66.8 |
| 37 | 338 | 3.2 | 30.3 | 2.4 | 14.8 | 45.7 |
| 48 | 1188 | 0.7 | 6.4 | 0.6 | 17.4 | 74.4 |
| 52 | 1165 | 0.9 | 6.2 | 1.0 | 17.4 | 70.7 |
| 60 | 737 | 2.9 | 13.9 | 5.7 | 10.5 | 53.0 |
| 65 | 522 | 2.0 | 19.2 | 1.5 | 15.2 | 58.6 |
| A4001/50 | 1196 | 1.0 | 3.2 | 1.6 | 13.4 | 66.3 |
| A4002/39 | 1197 | 2.7 | 9.6 | 3.5 | 22.3 | 49.8 |
| A4003/40 | 1200 | 0.5 | 3.5 | 0.4 | 0.7 | 78.2 |
| 55 | 626 | 0.9 | 6.5 | 1.6 | 13.8 | 74.7 |
| 64 | 611 | 1.3 | 8.1 | 1.6 | 9.6 | 78.5 |
| 68 | 316 | 2.7 | 13.9 | 2.9 | 15.3 | 52.4 |
| A4004/36 | 241 | 1.3 | 9.8 | 1.5 | 0.0 | 83.5 |
| 38 | 1195 | 1.4 | 3.3 | 2.3 | 10.7 | 72.5 |
| 53 | 1197 | 0.0 | 0.1 | 0.0 | 0.0 | 99.0 |
| 57 | 892 | 0.3 | 1.0 | 0.4 | 0.0 | 97.8 |
| 61 | 30 | 3.2 | 27.9 | 1.4 | 0.0 | 27.2 |
| 61 | 64 | 4.4 | 33.2 | 5.9 | 0.0 | 42.8 |

FIGURE 4-1. Program wall-time distribution. This chart shows what was happening to programs during the time each was being observed. For instance, during the 585 seconds that program A4000 (#S35) was visible within the window it spent 3% of that time executing, 21.1% of the time waiting for a segment to be made present before execution could continue, 1.6% of the time waiting for file I/Os to complete, 2.3% of the time for a data-base lock and 66.8% of the time waiting for response from a terminal read.

The following processes spent a significant amount of time waiting for 'blocked I/O'. This is caused by writing to a terminal when termbufs are unavailable or by nobuf I/O. The largest factor in this case is probably due to the nobuf I/O calls issued by IMAGE.

| | |
|---|---|
| A4000/60 | 13.64% |
| A4001/50 | 11.70 |
| A4002/39 | 16.57 |
| A4003/40 | 14.41 |
| A4003/68 | 10.06 |

```
------------------------------------------------------------------------
| Program: A4000                       #S35    OSSUSER.FMS             |
|                  Observed:  13:36:40 - 14:02:45      585.4 secs     |
|----------------------------------------------------------------------|
|                 |  OCCURRENCES    |SECS BETWEEN|  AVERAGE  |  CPU    |
| WAITING FOR:    | PRCNT    COUNT  | OCCURENCES |    WAIT   |BETWEEN  |
|-----------------|-----------------|------------|-----------|---------|
| ABSENT SEG      | 45.15      521  |    1.123   |     .237  | 0.034   |
| DISC I/O        | 22.70      262  |    2.233   |     .035  | 0.067   |
| TERM READ       | 16.90      195  |    2.996   |    2.005  | 0.087   |
| HIGHER PRI      | 12.31      142  |    4.120   |     .021  | 0.124   |
| MPE RESOURCE    |  1.47       17  |   34.240   |     .390  | 0.994   |
| TERM WRITE      |  1.21       14  |   41.522   |    1.480  | 1.204   |
| DATA BASE LOCK  |   .26        3  |  180.617   |    4.562  | 5.281   |
|----------------------------------------------------------------------|
| MEMORY AND SWAPPING LOAD:                                            |
|                                                                      |
|                   MEMORY USED       TIME IN MEMORY            OVER   |
|                PRCNT   AVG SIZE    AVERAGE   TOTAL     SWAPS  LAYS   |
|                ---------------     --------------      -----------   |
| STACK97        1.008      7816     15.099    332.1        22  1.772  |
| DSEG100         .002      7872       .701       .7         1  2.000  |
| DSEG101         .002      7640       .668       .6         1  5.000  |
| DSEG104         .006      7552      2.149      2.1         1  3.000  |
| DSEG94          .004      5440      1.935      1.9         1  2.000  |
| DSEG74          .120      5026      6.621     59.5         9   .333  |
| DSEG99          .185      2216      7.958    214.8        27   .000  |
| DSEG84          .177      1464     10.385    311.5        30   .000  |
|                                                                      |
| Avgs for 28                                                          |
| data segs      2.015      2353      6.655                238   .256  |
|                                                                      |
| Avgs for 44                                                          |
| SL segs       10.643      6714     13.472                296  2.311  |
|                                                                      |
| Avgs for 1                                                           |
| prog seg        .011      3904      7.357                  1   .000  |
------------------------------------------------------------------------
```

FIGURE 4-2. Process detail. This illustrates the level of detailed information available for each process active within the window. Events which caused the process to wait are listed at the top in order of number of occurrences. Note that this process used only 67 ms of CPU time between each disc I/O which occurred on the average every 2.2 seconds.

While the data here indicates a terminal response time of 991 ms (time between reads minus wait for read) actual response time was around 15 secs. The difference is caused by multiple reads being issued for a formatted screen.

## CONCLUSIONS AND RECOMMENDATIONS | SECTION V

The data presented in this report indicates a couple of reasons for your reduced response time. The memory manager is having trouble meeting demand for main memory. This is shown by the fact that MAM used over 10% of the CPU during the window. A swap rate of over 7 per second is another indication.

The memory contention problem can be reduced to some extent by reducing the size of the five large SL segments used by you application program. You should be able to identify these using the individual program data sheets in Section IV.

The second problem is database locking contention. Data in Section IV shows that most executions of your application spent about 15% of the time waiting to lock databases. This problem was particularly severe for A4002 (#S39) which spent over 22% of the time waiting for database access. No suggestions can be made here since a solution, if available, would require intimate knowledge of the application.

The information presented in Section III shows the impact of program development activity. Note the CPU and memory usage during executions of EDITOR, FCOPY and COBOL. When response times become unbearable, it may be necessary to curtail on-line program development.

During the 50 minutes that data was being collected 54 log-ons occurred. Since the log-on process places a heavy load on the system, even though for a short time, an attempt should be made to reduce this activity. At least 29 sessions lasted less than one minute.

An effort should be made to keep at least 20,000 free sectors on the system disc. At least 12-15% of the total disc space should be free at all times. Theory and experience both show that when the average utilization of any resource approaches the capacity, a large performance degradation results.

Data in Section IV seems to indicate that when the memory and locking bottlenecks are removed, it is unlikely that another "bottleneck" will be uncovered. Almost certainly the CPU will not restrict you for the foreseeable future. Disc activity level is low enough to suggest no problem will occur here either.

| PERFORMANCE OPTIMIZATION TOOLS | APPENDIX A |
|---|---|

There are several programs more or less available that in some way provide information pertaining to performance. Many are contributed and thus may or may not execute properly with the lastest release of MPE. The programs marked as contributed are in general circulation but have not been formally placed in the contributed library.

| | |
|---|---|
| 1. MONITOR | the combination of a software monitor built into MPE and a data reduction program. This is the most complete performance tool available. Output from the reduction program requires careful interpretation. Dedicated tape controller and drive required. Available only to SEs. |
| 2. SAMPLER* | used to identify those sections of code which are executed most frequently. Requires installation of an additional clock board on the system to be sampled. Dedicated tape controller and drive required. Available only to SEs. |
| 3. TRACER* | measures segment boundary crossings to determine which segments are referenced and how often. Can not be used with COBOL programs. Dedicated tape controller and drive required. Available only to SEs. |
| 4. TUNER2 | shows current and maximum use of several system tables. Used to determine whether configuration parameters have been correctly chosen. Contributed. |
| 5. OVERLORD | useful in determining who is executing what program and the corresponding stack size. SOO (Son of Overlord) provides the same functions. In the contributed library. |
| 6. SHOWVM | indicates, at a gross level, how real and virtual memory is being used. Helpful in determining how much memory a particular program uses. In contributed library. |
| 7. SHOWQ | a system command which displays information about process scheduling subqueues. If the rightmost of the three columns displayed grows longer than the center column, then the system has insufficient real memory for the current load. |

* General availability currently scheduled for early 1979.

| | |
|---|---|
| 8. FREE2 | displays the amount of free space on all system disc packs and indicates how fractured the space is. Badly fractured disc space can cause a considerable performance degradation. HP supported system utility. |
| 9. SYSINFO | prints system configuration information without doing a SYSDUMP. Contributed. |
| 10. PROGINFO | prints stack size, segment size and external reference information about program files. An extended version of PROGSTAT. Contributed. |
| 11. KSAMUTIL | displays blocking factors and intrinsic call counts associated with KSAM files. HP supported utility delivered with KSAM. |
| 12. LOG FILES | contain information associated with log-ons, file closes and I/O transfer counts by process. Some contributed programs available to reduce information in these MPE generated files. |
| 13. LISTFXX | used to carefully track use of disc space. Indicates location of first extent of each file and date of last access in addition to other information. When used in conjunction with FREE2 the amount of recoverable lost disc space can be calculated. In the contributed library. |
| 14. DBDRIVER | used to quickly obtain information about an IMAGE database such as the size of the DBCB and individual transaction times for a single user. Distributed with IMAGE but is unsupported. |
| 15. IDEA | helps determine performance characteristics of an IMAGE based application. Can measure the effect of multiple users. Timings apply only to the IMAGE response times, not application processing times. Contributed. |
| 16. DBSTAT | determines length of synonym chains in IMAGE master data-sets. Long synonym chains can cause dreadful performance degradation. Contributed. |
| 17. LISTDIR2 | used to determine location of the first extent of disc files. HP supported system utility. |
| 18. LISTF | The '-1' option displays the file label which contains the address of each extent of the file. Output can be written to a disc file for subsequent programatic processing. HP supported MPE command. |

## MPE USE OF MAIN MEMORY | APPENDIX B

The amount of memory reserved exclusively for MPE is referred to as fixed memory as opposed to linked memory for which all processes compete. The size of fixed memory can have a substantial effect on performance of machines with less than 512kb of memory.

| FIXED MEMORY GENERAL LAYOUT | SIZE BYTES | | ITEM |
|---|---|---|---|
| Driver linkage table | 256 | | |
| CST block table | 132 | * | 30 |
| Bank table | 192 | | |
| Job process count table | 88 | | |
| System global area | 768 | | |
| Data segment table | 3072 | * | 4 |
| Code segment table | 1536 | * | 2 |
| Code segment table extension | 4000 | * | 3 |
| Process control block table | 4096 | * | 5 |
| Interrupt control stack | 1088 | * | 10 |
| Terminal buffers | 4112 | * | 7 |
| I/O queue | 2832 | * | 6 |
| Interrupt linkage/Device info table | 6520 | | |
| System buffers | 4664 | * | 8 |
| Working set table | 4464 | * | 30 |
| Memory management table | 3840 | * | 9 |
| Virtual bit map | 512 | | |
| Virtual disc space locator | 768 | | |
| Logical-physical device table | 528 | | |
| Timer request list | 456 | * | 12 |
| Job cutoff table | 424 | | |
| System internal resource table | 440 | | |
| Breakpoint table | 520 | * | 13 |
| Memory management code | 10200 | | |
| Miscellaneous system routines | 3272 | | |
| System clock & timer req list code | 3304 | | |
| Resident IO code | 12064 | | |
| Internal interrupt handler code | 1880 | | |
| Dispatcher code | 1376 | | |
| Disc driver code | 2424 | | |
| Tape label input buffer | 552 | | |
| Miscellaneous areas | 5000 | | |

FIGURE B-1. Items in fixed memory of the 2Mb system at the Fullerton HP Technical Center listed in approximately the sequence they actually appear in memory. Lines with asterisks indicate segments whose size is directly affected by responses given at SYSDUMP time. The item numbers are references to Appendix C of the System Manager Manual.

## SEMI-RESIDENT SYSTEM CODE

In addition to the 60-90kb of fixed memory several MPE segments are referenced so frequently that they tend to spend much of the time in memory. On the average 3000, MPE is probably holding at least 150kb of memory at all times. Since this is not affected by the total amount of memory on the system adding more memory to systems in the 256-512kb size can substantially improve performance.

The following system code segments tend to spend at least 75% of the time in memory:

| | | |
|---|---|---|
| FILESYS1 | 8952 | FREAD, FWRITE |
| FILESYS1A | 5400 | FILESYS support routines |
| FILESYS2 | 5624 | FPOINT, FCONTROL, FUPDATE |
| FILESYS5 | 4208 | FILESYS support routines |
| FILESYS6 | 3472 | FILESYS support routines |

Other code segments which spend over 60 percent of the time present:

| | | |
|---|---|---|
| ALLOCUTIL | 5848 | Device allocation utilites |
| PINT | 3048 | CALENDAR, CLOCK, Process support |
| DATASEG | 5040 | Data segment handling routines |
| CHECKER | 1536 | GETPRIVMODE, Intrinsic errors |
| UTILITY1 | 3208 | ASCII, BINARY, WHO, READ, PRINT |
| IOTERM0 | 5128 | Terminal driver |
| | 23808 | Tota: )ytes |

Some MPE data segments which nd to stay in memory are listed below with a typical size in .ords.

| | | |
|---|---|---|
| UCOP request queue | 208 | |
| LDEV table | 2928 | |
| Disc directory seg | 2056 | |
| Job master table | 1024 | |
| Volume table | 200 | |
| FMAVT | 528 | |
| Process/job xref | 264 | |
| | 7208 | Total bytes |

| Program | Total Words | Num Segs | Global DB-QI | STACK QI-ZI | MAX DATA | SL Segs | SL Words | Tot Segs |
|---|---|---|---|---|---|---|---|---|
| APL | 138520 | 49 | 6883 | 1024 | 31000 | 21 | 48012 | 70 |
| BASIC | 41564 | 24 | 496 | 800 | 31000 | 24 | 53344 | 48 |
| BASICOMP | 33184 | 18 | 1146 | 800 | 30000 | 15 | 37756 | 33 |
| COBOL | 84892 | 35 | 3953 | 2000 | 32000 | 18 | 41212 | 53 |
| EDITOR | 33380 | 15 | 995 | 4600 | 8000 | 17 | 39680 | 32 |
| FCOPY | 16080 | 5 | 4893 | 800 | 31000 | 19 | 50304 | 24 |
| FORMAINT | 7088 | 2 | 1722 | 800 | 20000 | 10 | 23199 | 12 |
| FORTRAN | 45188 | 21 | 1701 | 2500 | 32767 | 16 | 36124 | 37 |
| FREE2 | 696 | 1 | 4172 | 800 | DFALT | 10 | 23048 | 11 |
| LISTDIR2 | 8936 | 4 | 795 | 800 | 8192 | 16 | 42460 | 20 |
| LISTEQ2 | 1012 | 1 | 2487 | 800 | 15000 | 5 | 12800 | 6 |
| MERGE | 2856 | 1 | 2 | 800 | 15000 | 15 | | 16 |
| MPMON | 2956 | 1 | 5524 | 800 | DFALT | 15 | | 16 |
| MRJE | 20488 | 7 | 3221 | 800 | DFALT | 19 | 42540 | 26 |
| MRJEMON | 1860 | 2 | 394 | 1100 | 5200 | 23 | 52208 | 25 |
| MRJEOUT | 1816 | 2 | 2306 | 1024 | DFALT | 12 | 29352 | 14 |
| QUERY | 42820 | 20 | 4174 | 1300 | 11000 | 28 | 62684 | 48 |
| RESTORE | 1948 | 1 | 2413 | 800 | DFALT | 8 | 20404 | 9 |
| RJE | 5400 | 8 | 927 | 1000 | DFALT | 24 | | 32 |
| RPG | 55396 | 22 | 3146 | 800 | 32000 | 15 | 37604 | 37 |
| SEGDVR | 1028 | 1 | 369 | 800 | DFALT | 5 | 10488 | 6 |
| SEGPROC | 12676 | 10 | 905 | 1000 | 24000 | 19 | 41828 | 29 |
| SORT | 2976 | 1 | 1 | 800 | 15000 | 14 | 36026 | 15 |
| SPL | 40628 | 30 | 3290 | 2500 | 32767 | 16 | 36124 | 46 |
| SPOOK | 7404 | 3 | 1791 | 800 | 30000 | 17 | | 20 |
| SYSDUMP | 16092 | 5 | 3425 | 1000 | 16000 | 22 | | 27 |
| | | | | | | | | |
| | 626884 | 299 | | | | | | |

FIGURE B-1. Listing of HP software showing the size of the program in words and the number of segments in the program file. The size of the initial stack is shown along with the maximum size it may grow to. SL segs refers to the number of SL segments that are directly referenced by the program file. These segments may in turn reference other segments. The sum of the length of all directly referenced SL segments is included under the heading SL words. Total segments is simply the sum of the program segments and the SL segments.

| MPE USE OF SYSTEM DISC | APPENDIX C |
|---|---|

## SYSTEM DISC LAYOUT

| USE OF DISC SPACE | APPROX SECTORS | |
|---|---|---|
| Disc label | 1 | |
| Defective tracks table | 4 | |
| Cold load information | 22 | |
| Free space table | 32 | |
| System directory | 384-6000 | (configurable) |
| Virtual memory swapping area | 1024-32767 | (configurable) |
| System files and tables | 7500 | |
| User file area | 149,000 - 187,000 | (7920) |

System software uses about 15,000 sectors of the user area.

On non-system discs (other than private volumes) the only overhead is for the disc label, the defective tacks table and the free space table. All other space is available for user files. Master volumes of private volume sets will have a file directory using 1000 - 4000 sectors.

| Drive type | Tracks | Sectors | Bytes |
|---|---|---|---|
| 7906 | 1,600 | 76,800 | 19,660,800 |
| 7920 | 4,075 | 195,600 | 50,073,600 |
| 7925 | 7,335 | 469,440 | 120,176,640 |

## PUB.SYS DISC SPACE

| FILE | SECTORS | FILE | SECTORS | FILE | SECTORS |
|---|---|---|---|---|---|
| APL | 1201 | FORTRAN | 384 | QUERY | 383 |
| BASIC | 349 | FREE2 | 43 | RESTORE | 40 |
| BASICOMP | 281 | INITIAL | 400 | RJE | 61 |
| CICAT | 2785 | LISTDIR2 | 84 | RPG | 472 |
| COBOL | 718 | LISTEQ2 | 32 | SEGDVR | 16 |
| COMMAND | 501 | MERGE | 29 | SEGPROC | 118 |
| DPAN2 | 297 | MPMON | 73 | SL | 5001 |
| EDITOR | 284 | MRJE | 195 | SORT | 30 |
| FCOPY | 174 | MRJEMON | 24 | SPL | 362 |
| FORMAINT | 76 | MRJEOUT | 40 | SPOOK | 79 |
| | | | | SYSDUMP | 162 |

## DISC I/O CONSIDERATIONS

All HP7900 family drives have a 937.5 kb/sec data transfer rate and all have identical seek times of 5 ms track-track, 25 ms average random and 45 ms typical full stroke. The 7906 and 7920 each have 48 sectors/track with a 8.3 ms average rotational delay. The 7925 has 64 sectors/track with an average rotational delay of 11.1 ms.

On the average, the HP3000 is capable of completing about 30 disc transfers per second. If a program has exclusive use of the system and is reading sequential sectors without buffering, as many as 58 transfers per second may be achieved. A program randomly writing sector size blocks to a large (115,000 sector) file might see a transfer rate on the order of 20-24 transfers per second. Swapping activity must be considered since it will use some of the 30 transfers available each second.

When a user reads a record, the data will be returned in about 5 milliseconds if the logical record is already in a buffer in memory.

## TERMINAL I/O CONSIDERATIONS

Every character passing to or from a terminal connected to the 3000 through the ATC (asynchronous terminal controller) causes a CPU interrupt. This is true whether the terminal is strapped for character, line or block mode. This can cause performance problems when the aggregate character rate approaches 2000 per second. While this indicates only 8 terminals can be simultaneously transferring a constant 240 characters per second, this is in fact very difficult to achieve for more than one or two seconds at a time.

```
--------------------------------------------------------------------------
|                                                  |                     |
|   PERFORMANCE IMPROVEMENT CHECKLIST              |   APPENDIX E        |
|                                                  |                     |
--------------------------------------------------------------------------
```

1. Reduce unnecessary logons.

2. Allocate often used program files.

3. Keep segment sizes under 5,000 words.
   Are any process stacks larger than necessary?

4. Check for file or database locking conflicts.

5. Will primary paths help database access?
   Are their long sort chains?
   Are there long synonym chains?
   Are any master data-sets more than 80% filled?
   Are all IMAGE DBCBs as small as practical?
   Use "*" in Image item lists whenever possible.

6. On-line program development team should use the textfile-
   masterfile' technique to reduce Text and Keep overhead.

7. Maintain sufficient free disc space.
   Is there at least 15,000 free sectors on the system disc?
   Is at least 10% of the total disc space free?

8. Sorts invoked from inside programs may run slower because
   less stack space is available for workspace. The stack
   may be left much larger than necessary for the rest of the
   program execution time.

9. Data files with high access rates should be evenly
   distributed over available drives. Program files with high
   swaps rates should reside on fast drives.

10. Keep system tables reasonably configured.

13. Are too many batch jobs executing concurrently?

14. Do any processes have more files open than necessary?

15. Are KSAM file blocking factors optimum?

16. Are any programs making excessive use of DEL edits?

# USING EXTRA DATA SEGMENTS: SAFE AND EFFICIENT

Rick Ehrhart
Hughes Aircraft Company, El Segundo
M/S 335/510
P.O. Box 92426
Los Angeles, CA 90009
(213) 648-0755

## Abstract

Handling extra data segments on the HP 3000 has given rise to numerous lectures, papers and classes. The methodology has always been to use the data segment "DS" intrinsics with the extra data segment "DS" capability. The purpose of this paper is to demonstrate a new methodology, one using the privileged "PM" capability.

## Introduction

There are two types of data segments on the HP 3000: the stack and the extra data segment. There is one and only one stack for each process to utilize the data in a predefined structure. The extra data segment, on the other hand, can be utilized by one or more processes in a user defined structure. It may be used as a table or an area for process-to-process communication.

For most applications, the "DS" instrinsics are quite adequate, but when the programmer is dealing with system level processes, the overhead of the "DS" intrinsics is unnecessary. This paper will show how to use three very powerful and privileged assembly level instructions that work on data segments: MTDS, MFDS, MDS.

## Techniques

To acquire an extra data segment, the programmer still must use the GETDSEG intrinsic. In non-privileged mode, GETDSEG returns a logical index; but in privileged mode, GETDSEG returns the Data Segment Table Index or "DSTX". This value is the unique index for the extra data segment that the process acquires. All data segments, whether a stack or an extra data segment, have a unique DSTX. The DSTX must be saved for all further operations.

To move data into the extra data segment from the stack, the programmer uses the "DS" intrinsic DMOVOUT. DMOVOUT checks the boundaries of both the stack and the extra data segment and then moves the data out from the stack to the extra data segment. However, in privileged mode, the programmer doesn't have to use DMOVOUT, he/she may use the assembly instruction MTDS, Move To Data Segment. This one instruction moves words from a DB-relative location to the extra data segment starting at a specified offset for a count. Usage is straightforward and simple. Example One has a procedure that shows how to use the MTDS assembly instruction. Basicly, the following items are pushed onto the top of the stack: target DSTX, the one returned from GETDSEG, the target offset, the source DB-relative address, where the data is in the stack, and a positive count of the number of words to be moved. The one drawback is the fact that the programmer has to make sure that the DSTX, the offset, the DB-relative address, and the count, are all valid.

To move data into the stack, the programmer used to call DMOVIN; but now the programmer may use the second privileged assembly instruction MFDS, Move From Data Segment. This instruction expects the following values on the top of the stack: the target DB-relative address, the source DSTX, the source offset in the DSTX, and a positive count. To see how to use the MTDS instruction, please see Example Two.

The third privileged assembly instruction is MDS, Move using Data Segments. It moves data from one data segment to another. This instruction requires the following values on the stack: the target DSTX, the target offset, the source DSTX, the source offset, and a positive or negative count. Please see Example Three for usage.

To release the extra data segment, the programmer uses the FREEDSEG intrinsic. The DSTX returned from GETDSEG is used as the index. The process has to be in privileged mode to call FREEDSEG.

## Conclusion

It is highly recommended that the programmer read the HP 3000 SERIES 2 MACHINE INSTRUCTION SET REFERENCE MANUAL, Part. No. 30000-90022 before attempting to use these privileged assembly instructions, since with these instructions, the programmer has a chance to destroy the system's integrity in one swift blow. For example, if the DSTX is invalid, the system will come to a halt with system failure 16; or if the DSTX is wrong, the instruction could overlay another user's stack or even a system table.

The above techniques do cut down on overhead when working with extra data segments. This is very useful for critical systems like a communications system, on-line monitor, or where processes have to communicate very quickly. These techniques have been used at Hughes Aircraft Company quite successfully, and the benefits are well worth the dangers.

```
<<**>>  EXAMPLE  ONE  <<**>>
<<------------------------------------------------------------------->>
<<>> PROCEDURE MOVE'TO'XDS(TARGET'DSTX,TARGET'OFFSET,SOURCE,COUNT);
     VALUE TARGET'DSTX,TARGET'OFFSET,SOURCE,COUNT;
     INTEGER TARGET'DSTX,TARGET'OFFSET,SOURCE,COUNT;
     OPTION PRIVILEGED;
BEGIN
 TOS := TARGET'DSTX;
 TOS := TARGET'OFFSET;
 TOS := SOURCE;
 TOS := COUNT;
 ASSEMBLE(MTDS 4);
 END;  << MOVE'TO'XDS >>


<<**>>  EXAMPLE  TWO  <<**>>
<<------------------------------------------------------------------->>
<<>> PROCEDURE MOVE'FROM'XDS(TARGET,SOURCE'DSTX,SOURCE'OFFSET,COUNT);
     VALUE TARGET,SOURCE'DSTX,SOURCE'OFFSET,COUNT;
     INTEGER TARGET,SOURCE'DSTX,SOURCE'OFFSET,COUNT;
     OPTION PRIVILEGED;
BEGIN
 TOS := TARGET;
 TOS := SOURCE'DSTX;
 TOS := SOURCE'OFFSET;
 TOS := COUNT;
 ASSEMBLE(MFDS 4);
 END;  << MOVE'FROM'XDS >>


<<**>>  EXAMPLE  THREE  <<**>>
<<------------------------------------------------------------------->>
<<>> PROCEDURE MOVE'XDS(TARGET'DSTX,TARGET'OFFSET,SOURCE'DSTX,
                        SOURCE'OFFSET,COUNT);
     VALUE TARGET'DSTX,TARGET'OFFSET,SOURCE'DSTX,SOURCE'OFFSET,COUNT;
     INTEGER TARGET'DSTX,TARGET'OFFSET,SOURCE'DSTX,SOURCE'OFFSET,COUNT;
     OPTION PRIVILEGED;
BEGIN
 TOS := TARGET'DSTX;
 TOS := TARGET'OFFSET;
 TOS := SOURCE'DSTX;
 TOS := SOURCE'OFFSET;
 TOS := COUNT;
 ASSEMBLE(MDS 5);
 END;  << MOVE'XDS >>
```

```
<<**>>  EXAMPLE  ONE  <<**>>
<<-------------------------------------------------------------------->>
<<>> PROCEDURE MOVE'TO'XDS(TARGET'DSTX,TARGET'OFFSET,SOURCE,COUNT);
     VALUE TARGET'DSTX,TARGET'OFFSET,SOURCE,COUNT;
     INTEGER TARGET'DSTX,TARGET'OFFSET,SOURCE,COUNT;
     OPTION PRIVILEGED;
BEGIN
 TOS := TARGET'DSTX;
 TOS := TARGET'OFFSET;
 TOS := SOURCE;
 TOS := COUNT;
 ASSEMBLE(MTDS 4);
 END;  << MOVE'TO'XDS >>


<<**>>  EXAMPLE  TWO  <<**>>
<<-------------------------------------------------------------------->>
<<>> PROCEDURE MOVE'FROM'XDS(TARGET,SOURCE'DSTX,SOURCE'OFFSET,COUNT);
     VALUE TARGET,SOURCE'DSTX,SOURCE'OFFSET,COUNT;
     INTEGER TARGET,SOURCE'DSTX,SOURCE'OFFSET,COUNT;
     OPTION PRIVILEGED;
BEGIN
 TOS := TARGET;
 TOS := SOURCE'DSTX;
 TOS := SOURCE'OFFSET;
 TOS := COUNT;
 ASSEMBLE(MFDS 4);
 END;  << MOVE'FROM'XDS >>


<<**>>  EXAMPLE  THREE  <<**>>
<<-------------------------------------------------------------------->>
<<>> PROCEDURE MOVE'XDS(TARGET'DSTX,TARGET'OFFSET,SOURCE'DSTX,
                        SOURCE'OFFSET,COUNT);
     VALUE TARGET'DSTX,TARGET'OFFSET,SOURCE'DSTX,SOURCE'OFFSET,COUNT;
     INTEGER TARGET'DSTX,TARGET'OFFSET,SOURCE'DSTX,SOURCE'OFFSET,COUNT;
     OPTION PRIVILEGED;
BEGIN
 TOS := TARGET'DSTX;
 TOS := TARGET'OFFSET;
 TOS := SOURCE'DSTX;
 TOS := SOURCE'OFFSET;
 TOS := COUNT;
 ASSEMBLE(MDS 5);
 END;  << MOVE'XDS >>
```

INSTALLATION MANAGEMENT

Series "D"

# I N S T A L L A T I O N   D E S I G N
# A N D
# O P E R A T I O N   C O N S I D E R A T I O N S

Longs Drug Stores: an example

by: Bill Gates

## LONGS DRUG STORES, INC.

- 114 STORES LOCATED IN CALIFORNIA, HAWAII, ALASKA AND ARIZONA
- $550 MILLION SALES IN FISCAL '78
- 5200 EMPLOYEES
- DECENTRALIZED OPERATION - NO CENTRAL WAREHOUSES

## HARDWARE CONFIGURATION

| | | |
|---|---|---|
| ONE | HP3000 SERIES II MODEL 9 WITH: | |
| | | 512 BYTES MAIN MEMORY |
| | ONE | HP7905 SYSTEM DISC |
| | ONE | HP7905 SPOOLER DISC |
| | THREE | 47M BYTE ISS DISCS |
| | TWO | 223M BYTE TELEFILE (AMPEX) DISCS |
| | THREE | 1600 BPI TAPE DRIVES |
| | ONE | 1250 LPM LINE PRINTER |
| | TWO | 200 LPM LINE PRINTERS |
| | ONE | HP2635 SYSTEM CONSOLE |
| | TWO | TI 743 HARD COPY TERMINALS |
| | ONE | DIABLO HARD COPY TERMINAL |
| | 24 | HP2640 (44) CRT TERMINALS |
| | TWO | SELECTOR CHANNELS |
| TWO | HP2100 DOS-TCS SYSTEMS WITH: | |
| | ONE | 4M BYTE DISC (EACH) |
| | 14 | HP2640 CRT-TERMINALS (EACH) |
| | ONE | 1600 BPI TAPE DRIVE |
| TWELVE | DATAPOINT 1500 DISKETTE TERMINALS | |

## APPLICATIONS

ACCOUNTS PAYABLE - 7000 INVOICES/DAY

PAYROLL - 5200 EMPLOYEES PAID WEEKLY

ACCOUNTS RECEIVABLE - 4000 TRANSACTIONS/DAY

GENERAL LEDGER

FINANCIAL REPORTING

CASH RECEIPTS

INTER-STORE TRANSFERS

INVENTORY

PHARMACY DRUG INFORMATION

ASSETS/DEPRECIATION

WORK PROCESSING

COM (MICROFICHE)

## OPERATING SCHEDULE/MIX

3 SHIFTS - 5 DAYS/WEEK

DAY SHIFT - ALL TERMINAL WORK
          - TYPICAL 12 - 20 SESSIONS, ONE-TWO JOBS

NIGHT SHIFT - HEAVY BATCH 3 - 4 JOBS

GRAVE SHIFT - LIGHT - MODERATE BATCH - "CLEAN-UP"

## SOFTWARE IN USE

- 95% COBOL
- 5% SPL AND BASIC
- MOST DATA STRUCTURED WITH <u>IMAGE</u> DATA-BASE SYSTEM
- HEAVY USE OF <u>QUERY</u> LANGUAGE
    - BY PROGRAMMERS FOR TEST/DEBUG
    - BY USERS FOR REPORTING AND LIMITED UPDATING

## ACCOUNTING STRUCTURE

- TWO MAIN ACCOUNTS:
- PRODUCTN (PRODUCTION)
- PROGDEV (PROGRAM DEVELOPMENT)

OTHERS: SYS
SUPPORT
ACCOUNT 1
PLAYLAND

## PROGDEV - PROGRAM DEVELOPMENT

### GROUPS

- BY APPLICATION SYSTEM

  ACP, PAY, ETC.

ACPS - CONTAINS WORKING SOURCE AND OBJECT FILES

ACP - CONTAINS ACCOUNTS PAYABLE TEST DATA

### USERS

PROGRAMMERS (NO HOME GROUP)

### SECURITY

ACCOUNT PASSWORD

## PRODUCTN (PRODUCTION)

GROUPS

PUB - ALL OBJECT PROGRAMS
- PRODUCTN STREAM FILE
- DATA BASE SCHEMA FILES
- (EXECUTE ACCESS TO ANY)
- (OTHER ACCESS TO AL, AM)

SOURCE ALL PRODUCTION SOURCE
- (READ ACCESS TO ANY)
- (OTHER ACCESS TO AL, AM)

XEQ ALL QUERY XEQ FILES

APPLICATION GROUPS
PAY, ACP, ETC. ALL DATA FILES
AND QUERY PROC.FILES

USERS

MGR - USED INFREQUENTLY

AL - RUNS MOST JOBS

STANDARD USERS (OUTSIDE EDP DEPT)

EXCEPTIONAL USERS (PROGRAMMERS)

NOTE: ALL PROGRAMS RUN FROM APPLICATION GROUPS, ACCESSING ONLY DATA WITHIN THE GROUP (EXCEPT FOR CERTAIN DATA BASES ALLOWING READ ACCESS TO AC.

## SECURITY

A. PHYSICAL SECURITY

B. DATA SECURITY

C. DISASTER CONTINGENCY PLANNING

D. AUDIT FUNCTION

## BACKUP

- DONE BY STORE RATHER THAN SYSDUMP

  REASONS:
    - STORE MAY BE SELECTIVE
    - STORE MAY BE RUN DURING OTHER PROCESSING

- MOST BACKUP DONE BY APPLICATION
    - USES SET OF GENERATION BACKUP TAPES
      101ACP, 102ACP..........120ACP ← VOLUME ID'S
    - ONLY FILES NECESSARY FOR RECOVERY (IN CASE OF CRASH) ARE STORED.
    - FREQUENCY DEPENDS UPON CHARACTERISTICS OF APPLICATION GROUP

- "SYSTEM" BACKUP DONE EACH EVENING AT 6:00 P.M.
    - FUTURE DATE SYSDUMP
    - STORE OF @.PUB.SYS
    - STORE OF @.PUB.PRODUCTN, @ XEQ.PRODUCTN, @SOURCE. PRODUCTN.
    - STORE OF MISC. INTERACTIVE GROUPS

## CONSOLE OPERATIONS PROGRAM

- READS JOB STREAMS TO TEMP FILE
- ALLOWS CONSOLE OPERATOR TO ENTER PROGRAM "PARAMETER CARDS"
- STREAMS FROM TEMP FILE

## SYSTEM MAINTENANCE

- "GOLD" BOOK KEPT
  - SYSTEM PROBLEM LOG
  - MAINTENANCE LOG
  - COPY OF SERVICE CONTRACT (S)
  - CURRENT CONFIGURATION
- "SYSDATA" JOB
  - RUN EACH MONDAY MORNING
  - CONTAINS:
    1) FREE 2 LISTING
    2) REPORT @.@ (+ RESET ACCT.)
    3) LISTF @.PROGDEV, 2
    4) LISTF @.PRODUCTN, 2
    5) MEMLOGAN. LISTING
    6) DUMMY SYSDUMP
    7) DATABASE UTILITY LISTING
  - REVIEWED AT DEPARTMENT MEETING
- COLD LOAD DONE EACH FRIDAY EVENING AFTER SYSDUMP
- RELOAD DONE AFTER P.M.
- WEEKLY COMPUTER SCHEDULE PREPARED

## MISC. RECOMMENDATIONS

- USE "EXCESSIVE" BACKUP AT BEGINNING
- WORK WITH CUSTOMER ENGINEER - LEARN YOUR HARDWARE!
- LEARN SYSTEM UTILITIES
- MAKE FREQUENT CONTACTS WITH OTHER SITES
- SET UP COMMUNICATION METHOD TO USERS
  (IN CASE OF SYSTEM CRASH OR DOWN TIME)
- "MANAGE" SYSTEM
- USE SECURITY FROM BEGINNING

## DATA PROCESSING SECURITY

I. PHYSICAL SECURITY

II. DATA SECURITY

## DISASTER CONTINGENCY PLANNING

I. PROTECTION AGAINST DISASTER

II. EDP OPERATION AFTER DISASTER

## CURRENT PHYSICAL SECURITY AT LONGS

- RESTRICTED ACCESS TO COMPUTER ROOM

## CURRENT DATA SECURITY AT LONGS

I. STANDARD APPLICATION DESIGN CONTROLS
   - A. DIVISION OF RESPONSIBILITY
   - B. EXTERNAL INPUT AND OUTPUT BALANCING (BY USERS)
   - C. USER APPROVAL OF PROGRAM CHANGES

II. ACCESS TO DATA RESTRICTED
   - A. PASSWORDS
   - B. USER CAPABILITIES
   - C. EDP DEPT. RESTRICTIONS

III. DATA ACCESS "AUDITABLE"
   - A. EDP AUDITOR
   - B. JOBS MUST "TIE TOGETHER"
     1. JOB REQUEST SHEET
     2. $STDLIST
     3. CONSOLE LOG
     4. SYSTEM LOG

## DISASTER CONTINGENCY PLAN

I. PROTECTION AGAINST DISASTER

A. HALON FIRE PREVENTION SYSTEM

B. OFF SITE BACK-UP OF FILES

II. EDP OPERATION AFTER DISASTER

A. BACK-UP SITES FOR COMPUTER

B. BACK-UP LOCATION FOR USERS

## HOW SECURITY WAS "INSTALLED"

### PHYSICAL

- WAS WIDE OPEN, GRADUALLY CLOSED IT OFF
- PHYSICAL ALTERATIONS
- INCREASED OPERATIONS PERSONNEL
- SET UP FORMAL PROGRAM TESTING PROCEDURES

### DATA

STEPS IN CHRONOLOGICAL ORDER:

- SET UP "OPEN" PASSWORD SYSTEM
  (EVERYONE KNEW PASSWORDS)
- STARTED SYSTEM LOGGING
- JOB CONTROL TIGHTENED
- SET UP MECHANISM FOR PASSWORD MAINTENANCE
  (BUT KEPT PASSWORDS "OPEN)
- SET UP FORMAL PROGRAMMER "SIGN-OUT" OF PRODUCTION DATA FOR TESTING - INCLUDED AUDIT
- CLOSED OFF PRODUCTION PASSWORDS TO PROGRAMMERS AND USERS (EXCEPTING THEIR OWN).

- DEVELOPED CONSOLE OPERATOR PROGRAM WHICH "INSERTS CORRECT PASSWORDS INTO JOB STREAMS - CLOSED OFF PASSWORDS TO OPERATIONS (EXCEPT FOR CONSOLE OPERATOR PASSWORD)

- OBTAINED ENOUGH DISC SPACE TO GIVE PROGRAMMERS SEPARATE "TEST" DATA BASES.
  IN PROGRAM DEVELOPMENT ACCOUNT - DEVELOPED UTILITIES TO HELP.

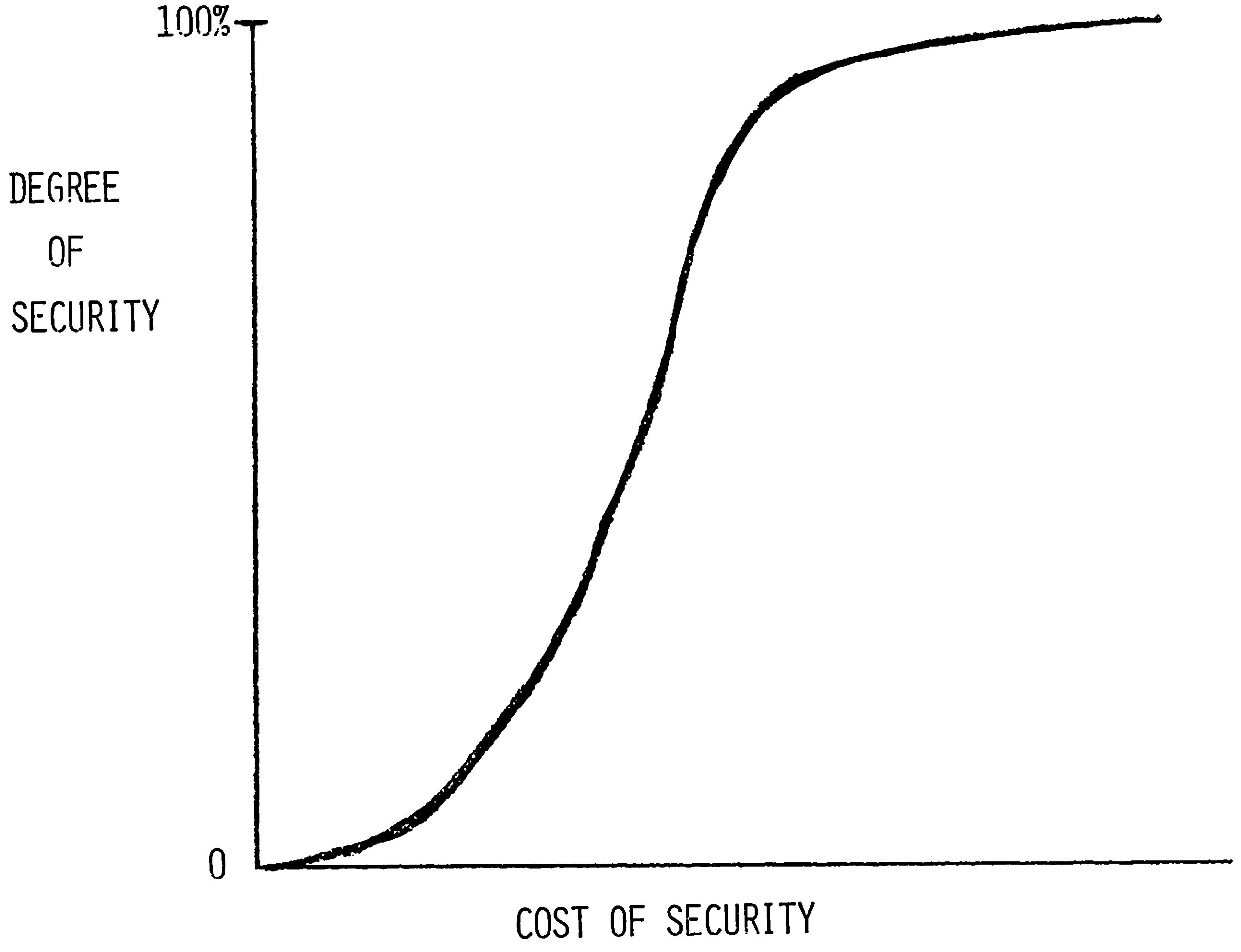
## <u>REMAINING "HOLES" IN SECURITY</u>

- TEST FILES MAY HOLD CONFIDENTIAL DATA

- LOG RECORDS DO NOT INDICATE IF A FILE HAS BEEN MODIFIED.

- LACK OF LOG INFORMATION FOR STORE/RESTORE UTILITY

COSTS OF COMPUTER SECURITY



## SUGGESTIONS FOR IMPLEMENTING SECURITY

1) USE GRADUAL PHASES

2) INVOLVE EDP AND USER PERSONNEL

   - EXPLAIN "TRADE-OFFS"

   - CHALLENGE PERSONNEL TO DEVELOP GOOD COMPROMISES BETWEEN SECURITY REQUIREMENTS AND EFFICIENT OPERATIONS.

   - EXPLAIN THAT LARGE "LOOP-HOLES" WILL EXIST DURING IMPLEMENTATION.

3) EXPECT VARYING DEGREES OF PERSONNEL RESISTANCE, RIDICULE, AND HOSTILITY. THIS SHOULD DECREASE OVER TIME.

## PERSONNEL OBJECTIONS TO SECURITY PLAN

OBJECTION: "THIS WHOLE SECURITY SET-UP IS A SHAM BECAUSE OF (ANY LOOPHOLE)! IT IS NOT PERFECT, SO IT IS WORTHLESS."

ANSWER: MOST EDP INSTALLATIONS ARE WIDE OPEN AS FAR AS SECURITY. EXPERIENCED COMPUTER CRIMINALS ARE LOGICAL PERSONS AND WOULD PREY ON THESE SHOPS RATHER THAN ONE WITH EVEN A MODEST ATTEMPT AT SECURITY. INEXPERIENCED COMPUTER CRIMINALS CAN BE INTIMIDATED BY LESS-THAN PERFECT SECURITY PRECAUTIONS.

---

OBJECTION: "THESE SECURITY PROVISIONS WILL MAKE MY WORK LESS CONVENIENT AND INEFFICIENT."

ANSWER: TRUE - HOWEVER, OUR COMPANY HAS CHOSEN TO ACCEPT THE COSTS INVOLVED WITH MAKING OUR INSTALLATION SECURE. IT'S UP TO US TO MINIMIZE THOSE COSTS.

# AN EXTENDED OPERATING ENVIRONMENT FOR THE SUPPORT OF APPLICATION PROGRAMS

RICHARD A. BERGQUIST AND STEVEN M. COOPER
AMERICAN MANAGEMENT SYSTEMS, INC.

The Brownboard Order and Rollstock Distribution System (BOARDS) supports order processing, invoicing, inventory control, and planning for the Shipping Container and Containerboard Marketing Division (SCD) of the Weyerhaeuser Company. The function of BOARDS is described more fully in the paper, "Decision Support System for the Management of Containerboard Logistics" by P. DiGiammarino and R. Schwartz.[1]

American Management Systems, a management consulting and system development firm, began work on BOARDS with Weyerhaeuser in September, 1976. Several subsystems are in production use; the system will be fully operational in early 1979.

BOARDS is written in COBOL, SPL, and FORTRAN, in order to utilize the advantages of each language. COBOL was chosen for the majority of the system because of its widespread use and report generating abilities. SPL was chosen for its efficiency and ability to interface with all aspects of the Operating System. FORTRAN was chosen for number processing routines such as those that employ linear programming techniques.

In order to provide an enhanced environment for the programmer without the need of becoming familiar with all aspects of the system, sets of common routines were developed. These routines also insure consistency and compatiability across the system and allow for easy maintenance of these technical functions.

Sets of common routines have been provided that extend the services provided by MPE, KSAM, IMAGE, and DEL. These sets of routines are summarized in Figure I and are described in more detail in the remainder of this paper.

## BATCH JOB SUBMISSION COMMON ROUTINES

MPE provides a convenient method of introducing batch jobs through the use of the STREAM command. However, one problem associated with STREAMing jobs is security. To be STREAMed, the User ID must be provided along with all appropriate passwords. The problem here is that either the user must

FIGURE I

| COMMON ROUTINES | FUNCTION |
|---|---|
| BATCH JOB SUBMISSION | Handles User ID's and passwords.<br>Parameter substitution.<br>Provides interactive front-end for batch job submission. |
| FORMS | Replacement for DEL, enhancements include:<br>- Protected variable data.<br>- Dynamic screens with multiple forms and repetitions.<br>- Mixed line and page mode transfer. |
| VERSION NUMBERS | Solves concurrent update problem without locking data base for entire transaction.<br>Identifies transactions that failed during data base modification either because of program failure or system failture.<br>Extends data base lock across process boundries. |
| DATA BASE ACCESS | Performs IMAGE calls.<br>Performs 'before' logging to protect against program aborts.<br>Performs 'after' logging to protect against file system errors.<br>Prevents 'Deadly Embraces'. |

be prompted for the password, the password must be hardcoded into a program, or passwords must be kept within the jobstreams on disc. Prompting the user is unacceptable for human engineering reasons, hardcoding does not allow for changing of User Id's or passwords, and leaving passwords on disc leaves them accessible to anyone who can STREAM the file. Finding none of these alternatives acceptable, a common routine to stream batch programs was designed. The common routine is passed the name of a template file, job parameters, and a parameter string.

The common routine creates a jobstream from a JOB statement which it creates and the statements contained in the template file. The JOB statement is based on the user running the program and includes all needed passwords. The User ID and passwords are kept in a table in a separate, hidden SL procedure. This allows passwords to be changed by simply modifying an SL segment, while allowing only legitimate procedures to access the passwords. As a further security measure, the User ID which is used does not have interactive capability -- i.e., it is restricted to batch access.

The remainder of the jobstream comes from the template file. The streaming procedure substitutes the run parameters which where passed to it into the jobstream wherever an ampersand occurs as the first character. This allows an interactive program to prompt the user for parameters and to then substitute the parameters into the batch system.

## HP2645A COMMON ROUTINES

The BOARDS terminal network consists of HP2645A terminals with 12K memory connected at 2400 baud via multiplexers (one time-division mux and one statistical mux), as well as various other ASCII devices that dial into standard BELL-103 type modems. Early in our design phase, we determined the needs of programs that would use the HP2645A's in Forms Mode, and evaluated the Hewlett Packard product, DEL/3000[2], in light of these requirements. This analysis resulted in two observations: 1) We needed certain features not provided by DEL/3000; and 2) DEL/3000 had features we did not need or want to pay for in terms of processing time. We therefore developed a set of COBOL-callable, SPL common routines, that are used as a total replacement for DEL/3000. Some of the differences between DEL/3000 and the BOARDS routines are described below.

A DEL form is made up of protected fields, whose contents are fully defined when the form is created, and unprotected fields, whose contents are alterable by the user and are read back to the computer whenever the form is read. In addition to these field types, we have defined variable-data protected fields. These fields may be filled by the program at run-time, but are protected on the screen so that they are unalterable and are not transmitted when the form is read. Three examples where these fields proved useful are:

- The user specifies that (s)he would like to enter an order for customer code 'ABC'. The name and address for this customer are then retrieved from the data base and displayed on the screen in variable-data, protected fields, for visual-verification by and information for the terminal user.

- Whenever an unprotected field is found to be in error, it is set blinking and a two-character error code is written to a variable-data, protected field at the beginning of the line that contains the field in error.

- The second line of every screen in the system is called the Message Line. It is an 80-byte variable-data, protected field in which the application program can inform the user as to the status of the processing and inform the user as to what is next expected of him/her.

DEL/3000 allows only one form to be on the screen at one time. Forms may be chained together, but this means that after one form has been displayed and processed, the screen will be cleared and the next form will be displayed. The BOARDS form routines allow multiple forms and multiple repetitions of forms to be displayed on the screen (and in terminal memory) at one time. This feature has given us the capability of having dynamically sized screens whose length is determined at run-time. It has also given us the ability to produce composite screens where each piece is selected at run-time from the form file. Similarly, the same form may be used as part of more than one screen.

The BOARDS common routines allow for the programatic control of the memory-lock feature of the HP2645A. All of the form routines function correctly whether or not memory-lock is set.

All screens in BOARDS have a similar first line containing, among other things, a one-byte unprotected field, called the Control Field. Various values may be entered by the user into this field in order to alter the normal flow of a program. Several of these values are processed by the common routines and the application program is not aware that this processing occurred. For instance, 'E' means exit, 'R' means redisplay the data in the unprotected fields. In several of these cases, once the Control Field is read, there is no need to read the rest of the screen. So when the ENTER key is depressed, the computer addresses the cursor to the Control Field and triggers a field read. If a value has been entered, it is processed and the rest of the screen is not read. If a value is not found, the terminal is programatically strapped-for-page, and a page read is triggered. This allows us to minimize terminal I/O's which can become important since some of our screens contain several thousand bytes of unprotected data.

Another Control Field value that is automatically handled by the common routines is 'P'. Whenever this value is entered, a hardcopy printer listing of the screen is produced. If the terminal is equipped with an HP2631/240 character printer, the terminal is instructed to copy the contents of terminal memory to the printer. For terminals without attached printers, the contents of the screen (unprotected, protected, and variable-data, protected fields) are written line-by-line into a printer spool file.

Many program functions must operate in inquiry mode on both HP2645A and other, non-screen mode, terminals. We therefore developed common routines to allow the <u>same</u> application program to operate on a variety of different terminal types for display only functions. When a program is run from a non-HP2645A terminal, the escape-sequences are stripped out and the unprotected, protected, and variable-data fields are combined on a line-by-line basis and printed, in character-mode, to the terminal.

Finally, I/O error recovery is attempted in the common routines. This is especially useful in dealing with the multiplexers, since saturation conditions can arise that would result in lost data. When an input error is detected, the read is re-initiated repetitively until it is either successful or the maximum retry value is reached. If none of the retries succeed, the screen is blanked and the form is re-written to the terminal under the assumption that either an output error occurred while writing the screen or the terminal user inadvertently damaged or erased the screen.

## <u>VERSION NUMBER ROUTINES</u>

The version number routines serve three purposes. First, they protect against concurrent update of a data base without locking the entire data base throughout the transaction. Second, they allow detection of incomplete transactions caused by system or program failures. Third, they extend a data base lock across process boundries.

Within a data base, a logical data path is formed by data which are logically grouped together but which may physically cross data set boundries. An example of this is a purchase order contained in a header record and line items which reside in a detail data set. The logical path in this case would consist of the header record and all of the line items.

If two users were to update the same path concurrently, some changes might be lost. Figure II shows how two users concurrently changing a path might 'lose' a change.

FIGURE II

| Time | User A | User B |
|---|---|---|
| Ø | User A gets purchase order XYZ in order to update it | |
| 1 | | User B also gets purchase order XYZ in order to update it. |
| 2 | User A adds two bolts to the purchase order and updates the data base. | |
| 3 | | User B adds three nails to the purchase order and updates the data base. However, User B is unaware that the purchase order has changed since it was originally obtained. |
| 4 | At this point the purchase order does not reflect the changes that User A applied. That change was lost when User B applied his change using the data base record retrieved before User A's change was made. | |

To protect against concurrently updating the same path, one must lock the data base when one begins the transaction and unlock it when the transaction is completed. If there are many users trying to update records in the same data base, this method is unacceptable; one user might not finish a transaction in a timely manner and the data base will be tied up for an extended period of time.

The version number routines use a data item (version number) for each logical data path. When a transaction begins, the version number is saved. When the user has completed all of his or her modifications and is ready to update the data base, the data base is locked, the version number is re-read and compared with the original version number. If the version number has not changed, then the path has not been modified and the program may continue with the user's modifications. At the end of the transaction, the version number is incremented. If the version number that was re-read has changed, the user must begin the transaction again because the records within the path have changed since the transaction began. The use of version numbers allows the data base to be locked only when modifications are in progress while still protecting against concurrent updates.

To detect if a transaction was only partially completed, an entry is made in a table (called the Integrity Table) whenever a modification begins and removed when the transaction completes. The Version Number routines add the table entry when the version number is re-read and found to be unchanged. The entry is deleted at the end of the transaction when the version number is incremented. By examining the Integrity Table while the system is quiesced, transactions which were only partially completed can be identified.

Under IMAGE, a data base is locked by a process. If the process that has a data base locked is aborted, the data base is unlocked. To extend a path lock across process boundries (used for backing out of incomplete transactions as described in the next section) the Version Number routines examine the Integrity Table whenever a version number is read. If an entry is found in the table, then the path is currently locked or a transaction was only partially completed. In either case, the path is inaccessible and modifications are not allowed.

## DATA BASE ACCESS COMMON ROUTINES

From a programmer's point of view, the data base routines are functional equivalents to their IMAGE counterparts. While performing as their IMAGE counterparts, the access routines are also performing 'before' and 'after' logging of all transactions as well as protecting against deadlocks when the data bases are locked.

Protection against deadlocks was accomplished by requiring that all data bases be locked at the same time (i.e., one call). The Data Base Access routine then locks the data bases in lexicographical order.

Two types of logging take place; 'before' and 'after'. 'Before' logging consists of saving a copy of all records associated with the transaction before any modifications are done. The 'before' log is essentially a snapshot of the data base before the transaction took place. 'After' logging consists of saving a copy of all records associated with the transaction after modifications are done. The 'after' log is essentially a snapshot of the data base after the transaction takes place.

## 'BEFORE LOGGING'

The 'before' log is used to restore the data base to its original state should the transaction process complete abnormally. 'Before' logging is done to Extra Data Segments to improve performance.

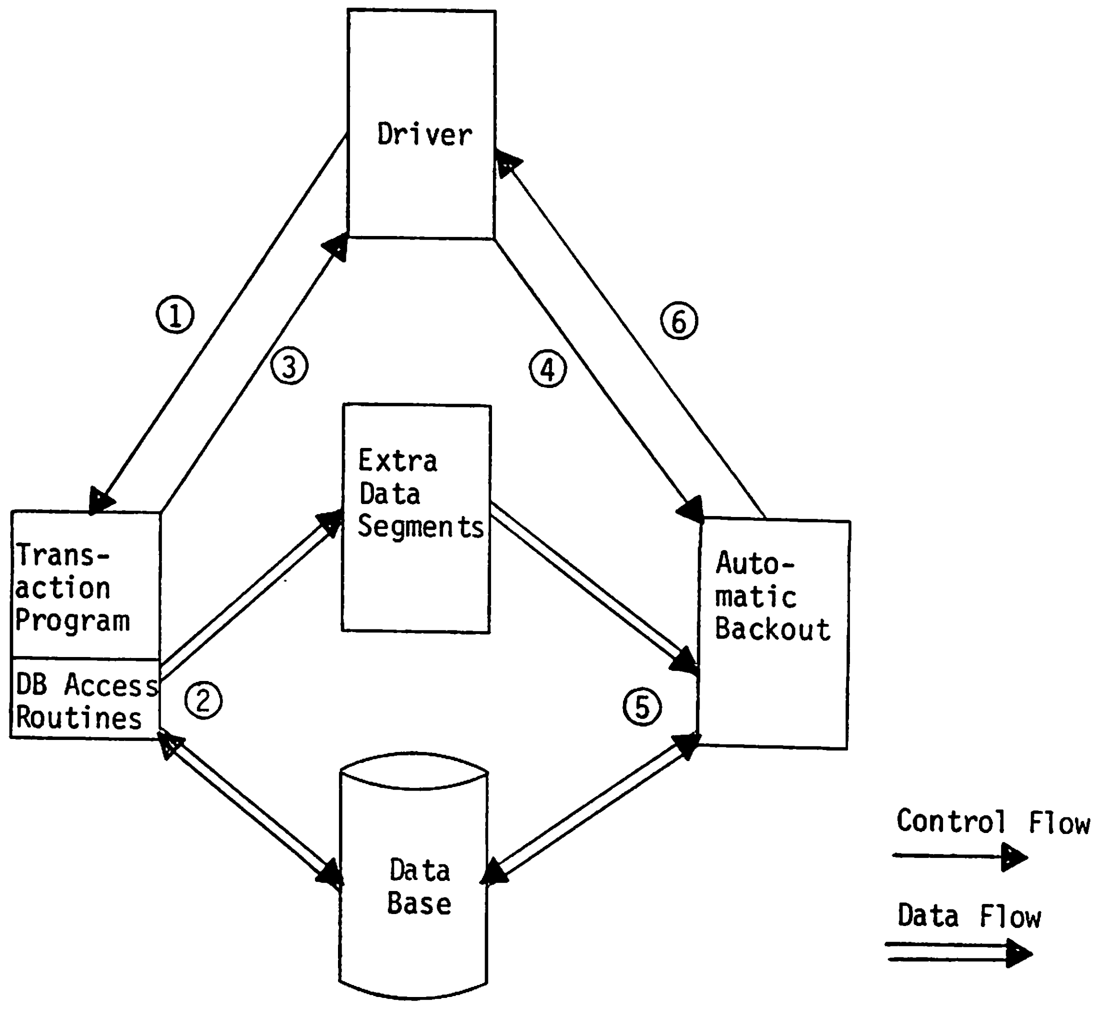
Since IMAGE requires a call to the GET procedure before a recond can be updated or deleted, all GETs are logged as they are performed. If the retrieved record is later updated or deleted, we record that information in the Extra Data segment. Likewise, all PUTs to the data base are also recorded in the EDS. Because the order of modifications is important, each individual data base modification is assigned a sequence number which is saved along with its buffer. If the transaction completes successfully, the 'before' log is purged. If the program does not complete successfully, the log remains which allows the data base to be restored to its state before the transaction began.

## THE DRIVER PROGRAM AND TRANSACTION BACKOUT

The operating system, MPE, provides for multiple processes to communicate via the Job Control Word (JCW) facility. The JCW is set to an error value by MPE whenever a program terminates in an error state. A program may also set the JCW to a particular value indicating an unsuccessful transaction. By examining the JCW, a father process can tell if a son process completed successfully, was aborted by MPE, or terminated due to an error condition. This facility is used by a program known as the Driver to oversee the operation of all programs within BOARDS.

All programs within BOARDS are son processes of the Driver. It is the Driver's function to prompt the user for the function (s)he wishes to perform and then initiate the correct program to handle the user's function. The Driver then sleeps, waiting for the program to complete. If the program does not complete successfully, the Driver initiates a program known as Automatic Backout whose function is to restore the data base to its original state. Figure III shows the control and data flow in the case that a program aborts.

FIGURE III



1. The driver initiates a transaction program on Request from user.

2. The program through the Data Base Access routines retrieves and updates the data base. Logging takes place to extra data segments.

3. The program aborts. MPE or the program sets a JCW to an error value.

4. The Automatic Backout program is initiated to restore the data base to its initial state.

5. Automatic Backout restores the data base.

6. Control is returned to the driver and it is ready to process next users request.

Automatic Backout retrieves the Extra Data segments which contain the before record and applies the opposite operation as the aborted program applied. Figure IV summarizes the required operations. All transactions are done in the reverse order that they were done by the aborted program. This is necessary to insure that updates are done in the correct order and that IMAGE master/detail constraints are satisfied.

## 'AFTER LOGGING'

'After' logging is used to protect against data base transactions beging lost because of some error which makes the data base unusable. In such a case, an old version of the data base must be restored. If transactions were not logged, then all transactions that were entered since the time of the backup must be re-entered by the user. By logging the transactions as they occur to a tape, a program can be run to perform the data bases transactions which are recorded on the tape. In this manner, users need not re-enter data in order to recover from the loss of a data base.

The Data Base Access routines call upon the Malkin and Pinton transaction logging[3] system to perform the 'after' logging.

## CONCLUSION

Through the use of common routines, the features of MPE and other HP-supplied system software have been expanded and utilized in COBOL application programs, without requiring that the programmers become familiar with MPE or SPL. These common routines serve as an Extended Operating Environment for the application programs within BOARDS.

FIGURE IV

| DATA BASE OPERATION PERFORMED | FIXUP OPERATION | COMMENTS |
|---|---|---|
| PUT | DELETE | |
| UPDATE | UPDATE | with 'before' record |
| DELETE | PUT | |

## REFERENCES

1) DiGiammarino, P. and Schwartz, R., Decision Support System for the Management of Containerboard Logistics, 1978.

2) Data Entry Library Reference Manual, Hewlett Packard, Santa Clara California, 1977.

3) Malkin and Pinton Industrial Supplies, Transaction Logging System for the HP3000 Computer System, Vancouver, British Colombia, Canada, 1977.

# BEACON/GUARDIAN: INSTALLATION MANAGEMENT SOLUTIONS IN SEARCH OF ELUSIVE PROBLEMS

WAYNE E. HOLT
WHITMAN COLLEGE

## I SYNOPSIS

Solutions in search of problems? It sounds odd but really isn't. These two software modules are solutions to a very specific type of installation management problem, and frankly would not be useful to most shops.

Guardian is a security module designed to increase the security provisions on both data files and program files in an environment where multiple users work in the same group and account, yet each have varying, overlapping, and often contradictory responsibilities. It is especially designed for the type of site where Users are totally responsible for data entry and report generation, both on-line and batch, rather than a site where the central DP shop handles everything.

BEACON (Batch Entry Access CONtrol) is a companion module to Guardian. If the site allows Users to stream jobs at will, someone or something has to play policeman and handle the traffic according to pre-defined rules (or in-flight instructions) in order to prevent the system from becoming clogged and dying. BEACON handles all job schedules, both regular cyclic and demand-mode jobs; it controls the number of jobs allowed to run simultaneously based upon system load and time of day; it modifies quantum, tpri, cpri, and dpri based upon time of day; it provides the DP Center with a concise monitor of all computer activity in a convenient format on any designated terminal; and it provides an easy to use method of rescheduling or expediting jobs.

These software modules are designed to operate in a User-oriented data processing center. The application packages that are used are specifically designed to enable a non-sophisticated User to very quickly learn to operate the terminals and become productive. Smart terminals (2645A's) with softkeys provide the tools needed to allow this type of user to request jobs with little or no understanding of the events set in motion by the request. Without BEACON/Guardian, such a shop would be difficult to manage successfully.

## II DATA PROCESSING PHILOSOPHY

Whitman College is typical of many of the current wave of "emerging" computer users. It had enjoyed a very modest degree of involvement with data processing for many years, primarily for business and financial uses. Such involvement called for few resources to be expended since the needs were correspondingly few. The times changed and it found itself unprepared to compete with those colleges that were using modern information technology to pursue a diminishing supply of student applicants.

Because of its size, it would be quite difficult to create and staff a large Data Processing Center, complete with both Systems and Operations personnel. The only viable alternative was to establish a Center whose philosophy was oriented toward total control by the User of the actual "data processing" function. The systems design and programming functions were retained by the Center, with the User required to know little more than :HELLO and :BYE in order to be fully effective.

A smart terminal, the HP2645A, is the hardware link that allows the User-oriented software to function smoothly. Application programs using either DEL or VIEW enable sophisticated screen techniques to facilitate data entry by non-DP personnel. These programs download the terminal softkeys with run instructions in such a manner that the User needs only to press a single key to initiate a job.

The actual technique is relatively simple (refer to Figure 1). After first loading the softkeys by means of a cassette tape or computer program, the User presses the key that best describes the type of work needed to be performed. This causes a program to be initiated that presents the User with a menu (Figure 2). The User selects the job function to be performed by keying an "X" in the appropriate box. Note that the menu lists items in plain English, using phrases that the User understands. The menu program then downloads a softkey with the necessary :RUN or :STREAM command, along with generating any required :FILE statements. The User then presses that key when ready to begin work.

The drawbacks to implementation of this type of philosophy is that it causes problems in two major areas: security and system performance. For these reasons, Guardian and BEACON were designed. They are the software tools that make this kind of environment manageable.

## III THE GUARDIAN SYSTEM

The MPE file management system provides levels of security that perform quite well in most situations, particularly where a central organization is responsible for production job set-up and related processing. It still functions in a user-controlled environment, but not with the same degree of effectiveness.

Consider, for example, a Registrars Office. This type of office usually has a mixture of regular and part-time staff, each with varying degrees of responsibility. Even among the regular staff, certain functions such as grade adjustments can only be performed by specific individuals within the office. When this office is automated in a manner described earlier, the traditional approach of adding lockwords can cause more security breaches than it prevents. Requiring a User to learn multiple passwords usually tends to promote a casual attitude toward such things.

Some Users, when faced with such an array of mumbo-jumbo, have actually been known to post the passwords on the Terminal itself!! A User should only be required to know his/her own unique Log-on, and it should NOT be shared with anyone else or known by anyone else except, of course, the System Manager.

The approach taken by Guardian is very simple. The Security File contains an entry for every valid User and the corresponding programs that he/she is allowed to run. All production software calls Guardian to validate each request to run a program. Guardian also checks the authorized run time and User terminal number as well, treating any violations as potential security breaches.

Further, all important data files are lockworded. Users do NOT know these lockwords; rather, they are known only by Guardian, which supplies them when the files are opened. This implies that Users have no access to their data except through approved software that interfaces with Guardian. Unfortunately, this also provides an obvious path for security breaches, since only a single password is needed to run any particular program to which a given User has access. On the other hand, all approved production software creates standard audit trails that can be analyzed for irregularities. Thus, the "most obvious path" is also the most dangerous for any potential security violator.

This software technique, oriented toward the specific User, is coupled with normal MPE methods to provide a high level of security for all situations. There are two program modules in the Guardian system. Refer to Figure 3 for the interrelationships with the BEACON system.

Guardian. This module is a called subroutine, executed by any program in production mode. it is responsible for

- o Authorizing a specific User to have access to a given program,
- o Verifying the time of day of use of the program,
- o Verifying the location of the terminal being used, and
- o Opening all required files for the calling program.

It returns status flags and file buffers to the calling program upon successful termination.

Sentinel. This module is a stand alone program that is used to build entries in the Security File. It is used to create and modify the records that provide Guardian with the information it uses to verify log-on requests by Users. Using prompt-answer techniques, it presents all known programs and jobs to the Manager for yes-no responses in relationship to a User's capabilities. It uses the Activity File in the BEACON system as a source for this information.

## IV THE BEACON SYSTEM

When Users have the capability of performing almost any task (data entry, report generation, file maintenance, etc.) whenever they choose to do so, the computer system is not going to perform well. The Computer Center could set the system limits down to ensure good response time in a FIFO (First In First Out) situation, but overall throughput would suffer at the hands of such arbitrary measures. Constant operator intervention would be required to manage such a situation, thus defeating the whole purpose.

BEACON is designed to remedy this situation. Fundamentally, its primary purpose is to take a batch job request from a User and run it at the appropriate time in an efficient manner. In order to do this, BEACON actually performs a wide variety of tasks. These tasks are split between various program modules. Please refer to Figure 3.

Gopher. This module is a called subroutine, executed by any program that requests a job to be launched. The calling program passes the name of the job to be launched and any associated run time parameters to Gopher, which in turn builds a Job Request Record in the Activity File. It builds the record based on the contents of a Priority Record permanently resident in the Activity File, which includes

- o Execution priority on a scale of 1 (high) to 9 (low),
- o Literal Description of the Job,
- o Normal start time (HH:MM), and
- o Average run duration in minutes.

The Job Request Record is built as a "one time" record and will eventually be deleted when launched by BEACON. A Wizard number is assigned to each Job Request Record and is returned to the calling program by Gopher.

Wizard. This module is a stand-alone program that should be run by the operator or account manager. It is designed to perform a wide variety of tasks associated with running the BEACON module.

It is the tool that is used to modify the contents of the Job Request Records in the Activity File when responding to spot requests by the Users. Any key field can be altered -- date, time, or priority of launch. Also, hot jobs can be placed in a demand-mode launch priority regardless of the System load if the priority is altered to "0".

One of its most important functions is to build a Job Request Record for "permanent" or "cyclic" jobs. These records are identical to the "one-time" Job Request Records except that the run interval prevents them from being deleted by BEACON after launching.

Wizard also maintains the Clock Records on the Activity File. These 24 records control various parameters that affect system performance:

- o Default execution priority for batch jobs,
- o Maximum execution priority for both batch and sessions,
- o Quantum, tpri, cpri, dpri,
- o BEACON delay interval,
- o Video monitor LDEV number assignment, and
- o BEACON status flag.

Wizard can modify any of these items; BEACON will implement them automatically. This allows an unmanned system to "tune" itself as the workload shifts during the 24-hour day.

Snoopy. This module is designed to be run by the User as well as the operator. It produces a formatted display (on either the terminal or the line printer) of all jobs waiting to be launched and their expected launch times. Snoopy takes into account the average run times as well as the System Job limits from the various Clock records in order to create as accurate a forecast as possible.

<u>BEACON</u>. The BEACON module itself is actually quite simple and very efficient in terms of system resource utilization. It is designed to be constantly "up" as a streamed job although it is normally in a "sleep" mode. It could be run in an interactive session if there were an unused port, but this is not likely. It can be started and stopped by Wizard, which also controls the frequency of waking up.

Upon waking up, BEACON checks the System clock and reads the appropriate Clock Record in the Activity File. It then adjusts any system parameters that may have been changed since the last time that it was awake.

Next, it reads the Job Request Records on the Activity File in a sequential mode and determines if a job is ready to be launched by comparing the time fields. If there is a priority "0" (demand-mode) job in the queue it will be launched immediately. BEACON will continue searching the Job Request Records until they are all examined or a "launch ready" status is determined for one of them. If no job is ready to launch, BEACON will skip on to the next step. Otherwise, it checks the system load and determines whether to launch or not.

If a job is launched, BEACON will then log the System Job number and the exact launch time in the Job Request Record and do a KSAM delete. Note that this only "flags" the record for deletion and renders it invisible to BEACON. It will also regenerate any "permanent" Job Request Record with the new launch date/time.

The next step is optional, depending upon a flag set in the current Clock Record. If a valid LDEV has been specified, BEACON will output a current system status display, showing important system performance data. This display could easily be connected to a large video monitor for informational display purposes.

BEACON then goes to sleep for the period of time specified in the current Clock Record.

Reporter. This module is used to generate an Activity Summary report that analyzes BEACON's performance. It is run prior to the daily FCOPY "cleanup" of the deleted records on the Activity File. It finds the Job Request Records of all launched jobs and compares expected launch time with actual launch time and calculates various statistics. Such a report is valuable in setting the values in the Clock Records in the Activity File. It also matches the Wizard number with the final Job number, as an aid in tracking down "lost" jobs.

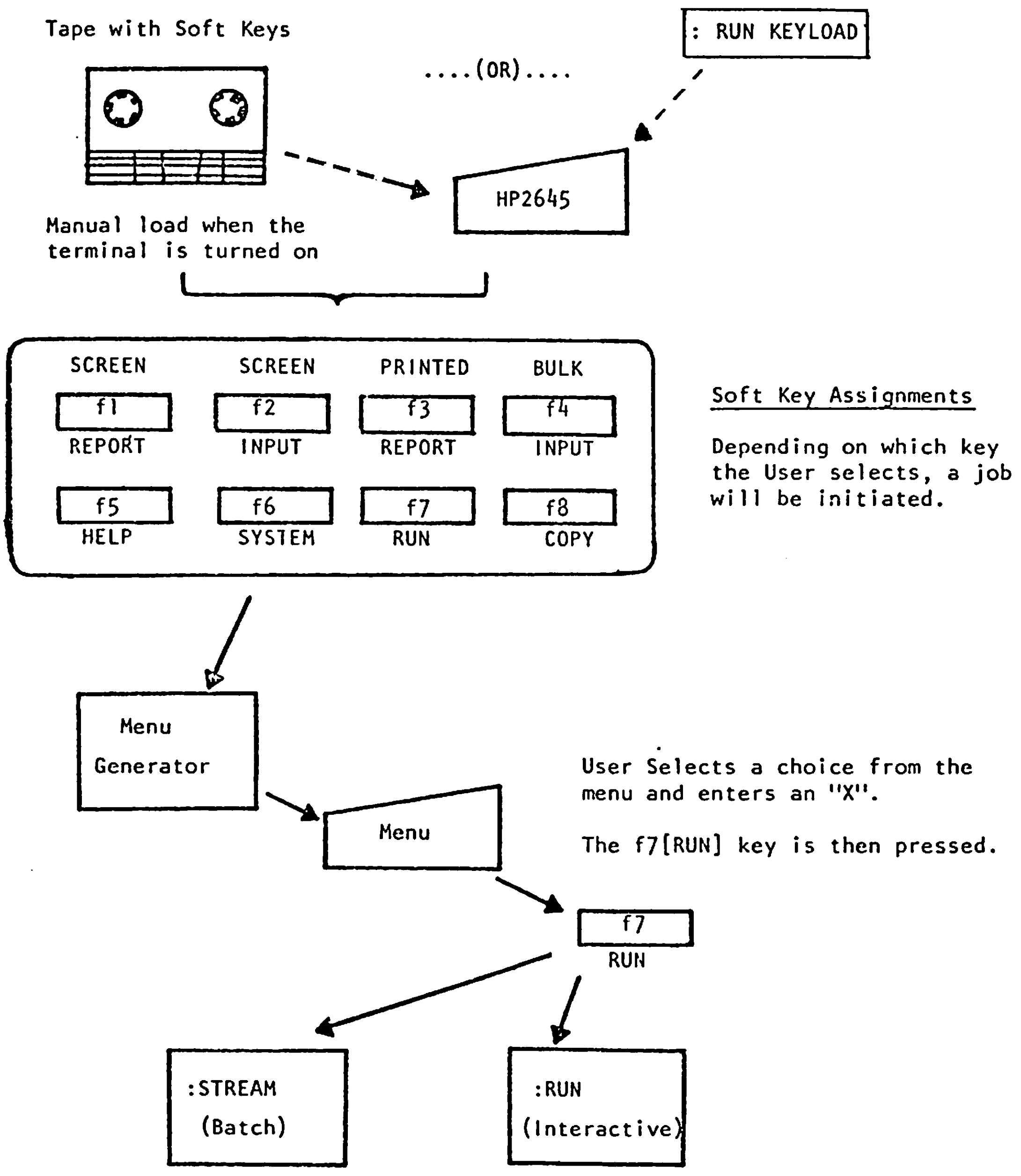
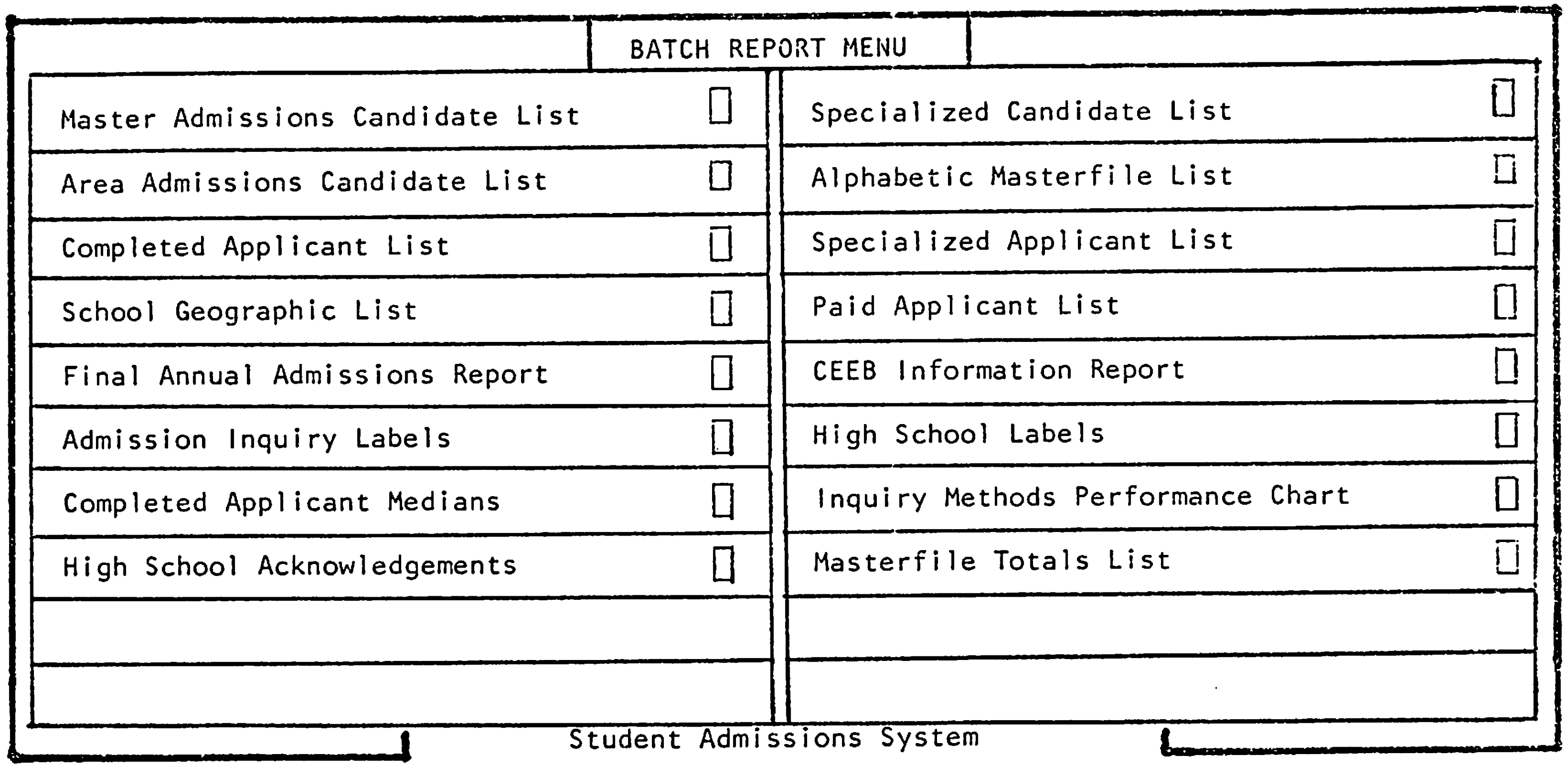

Tape with Soft Keys
....(OR)....
: RUN KEYLOAD
HP2645
Manual load when the terminal is turned on
SCREEN f1 REPORT
SCREEN f2 INPUT
PRINTED f3 REPORT
BULK f4 INPUT
f5 HELP
f6 SYSTEM
f7 RUN
f8 COPY
Soft Key Assignments
Depending on which key the User selects, a job will be initiated.
Menu Generator
Menu
User Selects a choice from the menu and enters an "X".
The f7[RUN] key is then pressed.
f7 RUN
:STREAM (Batch)
:RUN (Interactive)

Figure 1



BATCH REPORT MENU

| | |
|---|---|
| Master Admissions Candidate List ☐ | Specialized Candidate List ☐ |
| Area Admissions Candidate List ☐ | Alphabetic Masterfile List ☐ |
| Completed Applicant List ☐ | Specialized Applicant List ☐ |
| School Geographic List ☐ | Paid Applicant List ☐ |
| Final Annual Admissions Report ☐ | CEEB Information Report ☐ |
| Admission Inquiry Labels ☐ | High School Labels ☐ |
| Completed Applicant Medians ☐ | Inquiry Methods Performance Chart ☐ |
| High School Acknowledgements ☐ | Masterfile Totals List ☐ |
| | |
| | |

Student Admissions System

Figure 2

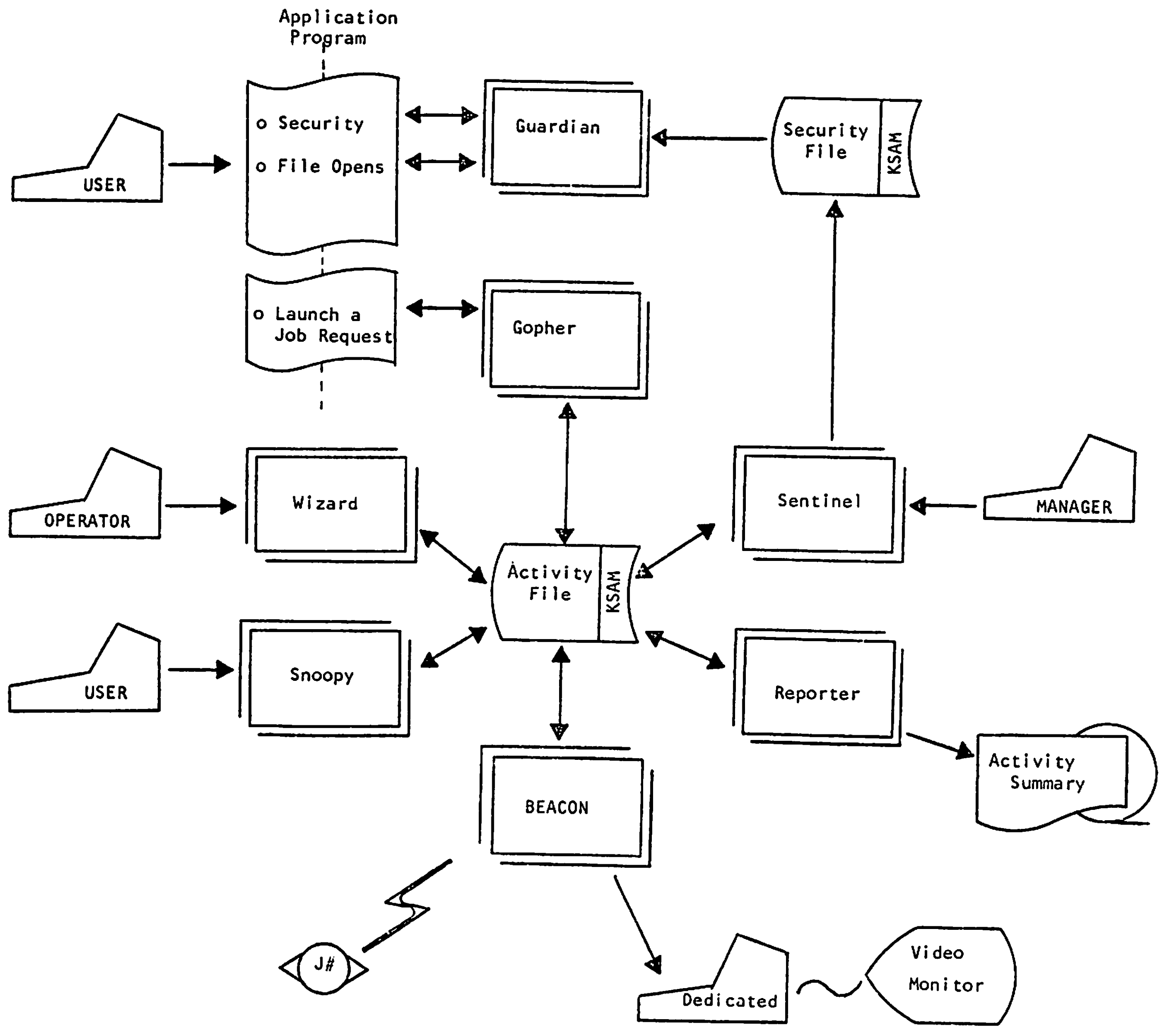

Application Program
o Security
o File Opens
o Launch a Job Request
USER
Guardian
Security File
KSAM
Gopher
OPERATOR
Wizard
Activity File
KSAM
Sentinel
MANAGER
USER
Snoopy
Reporter
Activity Summary
BEACON
J#
Dedicated
Video Monitor

Figure 3

LANGUAGES

Series "E"

# THE CHANGING WORLD OF COBOL

GREG GLOSS

HP GENERAL SYSTEMS

This talk will cover some of the differences between COBOL/3000 and ANSI COBOL-74 along with how HP users will be able to convert. There will also be a brief discussion of the directions the ANSI COBOL Committee is taking with the next version of the COBOL Standard.

## NEW FEATURES IN COBOL-74

1. Indexed I/O
2. Relative I/O
3. Enhanced SORT/MERGE Facility
4. File Status
5. STRING/UNSTRING
6. Multiple COPYLIB Files

## CONVERSION CONSIDERATIONS

1. Conversion Guide/Program
2. FIPS Pub 43
3. New Reserved Words
4. EXAMINE vs. INSPECT
5. Microcode Support
6. COPYLIB Editor
7. REMARKS/NOTE Paragraphs deleted

## COBOL-80 DIRECTIONS (NOT FINAL)

I. NEW FEATURES

   A. CODASYL Data Base Facility
   B. Reference Modification
   C. 48 Levels of Subscripting
   D. USAGE BIT

II. FEATURES TO BE PHASED OUT OR DELETED

A. 77 & 66 Level Data items
B. LABEL RECORDS Clause/VALUE OF Clause
C. ADD/SUBTRACT CORRESPONDING
D. ALTER Statement
E. PICTURE Character A
F. Most IDENTIFICATION DIVISION Paragraphs
G. File related clauses moved
H. INSPECT TALLYING. . .REPLACING

## THE CHANGING WORLD OF COBOL

- NEW FEATURES IN COBOL-74
- CONVERSION CONSIDERATIONS
- COBOL-80

## NEW FEATURES

- INDEXED I/O
- RELATIVE I/O
- ENHANCED SORT/MERGE FACILITY
- FILE STATUS
- STRING/UNSTRING
- MULTIPLE COYPLIB FILES

## SORT STATEMENT

```
SORT FILE-NAME-1 ON { ASCENDING  } KEY DATA-NAME-1  [DATA-NAME-2] . . .
                    { DESCENDING }

                 [ ON { ASCENDING  } KEY DATA-NAME-3  [DATA-NAME-4] . . . ] . . .
                 [    { DESCENDING }                                      ]

     [COLLATING SEQUENCE IS ALPHABET-NAME]

     { INPUT PROCEDURE IS SECTION-NAME-1 [ { THROUGH }  SECTION-NAME-2 ] }
     {                                   [ { THRU    }                 ] }
     { USING FILE-NAME-2  [,FILE-NAME-3] . . .                           }

     { OUTPUT PROCEDURE IS SECTION-NAME-3 [ { THROUGH }  SECTION-NAME-4 ] }
     {                                    [ { THRU    }                 ] }
     { GIVING FILE-NAME-4                                                 }
```

## MERGE STATEMENT

```
                      ┌ASCENDING ┐                    ┌            ┐
MERGE FILE-NAME-1 ON  ┤          ├ KEY DATA-NAME-1   │,DATA-NAME-2│ . . .
                      └DESCENDING┘                    └            ┘
                   ┌                                                     ┐
                   │  ┌ASCENDING ┐                    ┌            ┐    │
                   │ON┤          ├ KEY DATA-NAME-3   │,DATA-NAME-4│ . . .│ . . .
                   │  └DESCENDING┘                    └            ┘    │
                   └                                                     ┘
   ┌                                        ┐
   │COLLATING SEQUENCE IS ALPHABET-NAME     │
   └                                        ┘

    USING FILE-NAME-2, FILE-NAME-3  [,FILE-NAME-4] . . .

   ┌                                          ┌┌THROUGH┐                   ┐┐
   │OUTPUT PROCEDURE IS SECTION-NAME-1        ││       │ SECTION-NAME-2    ││
   │                                          └└THRU   ┘                   ┘│
   ┤                                                                        ├
   │GIVING FILE-NAME-5                                                      │
   └                                                                        ┘
```

# STRING STATEMENT

$$\text{STRING} \left\{\begin{matrix}\text{IDENTIFIER-1}\\ \text{LITERAL-1}\end{matrix}\right\} \left[\begin{matrix}\text{,IDENTIFIER-2}\\ \text{,LITERAL-2}\end{matrix}\right] \ldots \ \underline{\text{DELIMITED}} \text{ BY} \left\{\begin{matrix}\text{IDENTIFIER-3}\\ \text{LITERAL-3}\\ \text{SIZE}\end{matrix}\right\}$$

$$\left[ , \left\{\begin{matrix}\text{IDENTIFIER-4}\\ \text{LITERAL-4}\end{matrix}\right\} \left[\begin{matrix}\text{,IDENTIFIER-5}\\ \text{,LITERAL-5}\end{matrix}\right] \ldots \ \underline{\text{DELIMITED}} \text{ BY} \left\{\begin{matrix}\text{IDENTIFIER-6}\\ \text{LITERAL-6}\\ \text{SIZE}\end{matrix}\right\} \right] \ldots$$

$$\underline{\text{INTO}} \text{ IDENTIFIER-7} \left[\text{WITH POINTER IDENTIFIER-8}\right]$$

$$\left[\text{; ON } \underline{\text{OVERFLOW}} \text{ IMPERATIVE-STATEMENT}\right]$$

# UNSTRING STATEMENT

```
UNSTRING identifier-1
    [ DELIMITED BY [ALL] {identifier-2} [,OR [ALL] {identifier-3}] . . . ]
                         {literal-1   }             {literal-2   }

    INTO identifier-4  [,DELIMITER IN identifier-5]  [,COUNT IN identifier-6]

        [,identifier-7  [,DELIMITER IN identifier-8]  [,COUNT IN identifier-9]] . . .

      [WITH POINTER identifier-10]  [TALLYING IN identifier-11]

     [;ON OVERFLOW imperative-statement]
```

## CONVERSION CONSIDERATIONS

- CONVERSION GUIDE/PROGRAM
- FIPS PUB 43
- NEW RESERVED WORDS
- EXAMINE/INSPECT
- MICROCODE SUPPORT
- COPYLIB EDITOR
- REMARKS/NOTE PARAGRAPH

FIPS PUB 43

FEDERAL INFORMATION
PROCESSING STANDARDS PUBLICATION

1975 DECEMBER 1

U.S. DEPARTMENT OF COMMERCE / National Bureau of Standards

# AIDS FOR COBOL PROGRAM CONVERSION (FIPS PUB 21 to FIPS PUB 21-1)

CATEGORY: SOFTWARE
SUBCATEGORY: PROGRAMMING LANGUAGE

E-04.10

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402 · Price $1.10
Stock Number 003-003-01483-4 Catalog Number C 13.52:43

The following Reserved Words have been added to ANSI COBOL in the 1974 standard:

ALSO
BOTTOM
CANCEL
CD
CHARACTER
CODE-SET
COLLATING
COMMUNICATION
COUNT
DATE
DAY
DEBUG-CONTENTS
DEBUG-ITEM
DEBUG-LINE
DEBUG-NAME
DEBUG-SUB-1
DEBUG-SUB-2
DEBUG-SUB-3
DEBUGGING
DELETE
DELIMITED
DELIMITER
DESTINATION
DISABLE
DUPLICATES
DYNAMIC
EGI
EMI
ENABLE
END-OF-PAGE
EOP
ESI
EXCEPTION
EXTEND
INITIAL
INSPECT
LENGTH
LINAGE
LINAGE-COUNTER
MERGE
MESSAGE
NATIVE
ORGANIZATION
OVERFLOW
POINTER
PRINTING
PROCEDURES
QUEUE
RECEIVE
REFERENCES
RELATIVE
REMOVAL
REWRITE
SEGMENT
SEND
SEPARATE
SEQUENCE
SORT-MERGE
STANDARD-1
START
STRING
SUB-QUEUE-1
SUB-OUEUE-2
SUB-QUEUE-3
SUPPRESS
SYMBOLIC
TABLE
TERMINAL
TEXT
TIME
TRAILING
UNSTRING

The following words are tentatively planned for HP extensions to COBOL at a future date. This list is subject to change.

| | |
|---|---|
| C01 | INTRINSIC |
| C02 | LABELS |
| C03 | NOLIST |
| C04 | SEQ |
| C05 | SW0 |
| C06 | SW1 |
| C07 | SW2 |
| C08 | SW3 |
| C09 | SW4 |
| C10 | SW5 |
| C11 | SW6 |
| C12 | SW7 |
| CC | SW8 |
| CONDITIONALLY | SW9 |
| EBCDIC | UN-EXCLUSIVE |
| EXCLUSIVE | VOL |
| EXDATE | WHEN-COMPILED |
| FREE | |

# EXAMINE STATEMENT

$$\underline{\text{EXAMINE}}\ \text{IDENTIFIER}$$

$$\left\{\begin{array}{l} \underline{\text{TALLYING}} \left\{\begin{array}{l} \underline{\text{UNTIL}}\ \underline{\text{FIRST}} \\ \underline{\text{ALL}} \\ \underline{\text{LEADING}} \end{array}\right\} \text{LITERAL-1} \left[\underline{\text{REPLACING}}\ \underline{\text{BY}}\ \text{LITERAL-2}\right] \\ \\ \underline{\text{REPLACING}} \left\{\begin{array}{l} \underline{\text{ALL}} \\ \underline{\text{LEADING}} \\ \left[\underline{\text{UNTIL}}\right]\ \underline{\text{FIRST}} \end{array}\right\} \text{LITERAL-3}\ \ \underline{\text{BY}}\ \ \text{LITERAL-4} \end{array}\right\}$$

INSPECT STATEMENT

Format 1

```
INSPECT identifier-1 TALLYING

{ , identifier-2 FOR { , { {ALL    } {identifier-3} } [ {BEFORE} INITIAL {identifier-4} ] } ... } ...
                         { {LEADING} {literal-1   } }   {AFTER }         {literal-2   }
                         { CHARACTERS               }
```

Format 2

```
INSPECT identifier-1 REPLACING

{ CHARACTERS BY {identifier-6} [ {BEFORE} INITIAL {identifier-7} ]                                        }
{               {literal-4   }   {AFTER }         {literal-5   }                                          }
{ { , {ALL    } { , {identifier-5} BY {identifier-6} [ {BEFORE} INITIAL {identifier-7} ] } ... } ...      }
{ {   {LEADING} {   {literal-3   }    {literal-4   }   {AFTER }         {literal-5   }   }     }          }
{ {   {FIRST  } {                                                                        }     }          }
```

Format 3

```
INSPECT identifier-1 TALLYING

{ , identifier-2 FOR { , { {ALL    } {identifier-3} } [ {BEFORE} INITIAL {identifier-4} ] } ... } ...
                         { {LEADING} {literal-1   } }   {AFTER }         {literal-2   }
                         { CHARACTERS               }

   REPLACING

{ CHARACTERS BY {identifier-6} [ {BEFORE} INITIAL {identifier-7} ]                                        }
{               {literal-4   }   {AFTER }         {literal-5   }                                          }
{ { , {ALL    } { , {identifier-5} BY {identifier-6} [ {BEFORE} INITIAL {identifier-7} ] } ... } ...      }
{ {   {LEADING} {   {literal-3   }    {literal-4   }   {AFTER }         {literal-5   }   }     }          }
{ {   {FIRST  } {                                                                        }     }          }
```

## NEW FEATURES FOR COBOL-80

- CODASYL DATA BASE FACILITY
- REFERENCE MODIFICATION
- 48-LEVEL SUBSCRIPTING
- USAGE BIT

## FEATURES TO BE PHASED OUT IN COBOL-80

- 77 AND 66 LEVEL DATA ITEMS
- LABEL RECORDS CLAUSE/VALUE OF CLAUSE
- ADD/SUBTRACT CORRESPONDING
- ALTER STATEMENT
- PICTURE CHARACTER "A"
- MOST IDENTIFICATION DIVISION PARAGRAPHS
- FILE RELATED CLAUSES MOVED BETWEEN ENVIRONMENT AND DATA DIVISIONS
- INSPECT TALLYING. . .REPLACING

# TIPS ON CONVERTING IBM FORTRAN PROGRAMS
# TO THE
# HP 3000

BY

GARY ANDERSON AND DEEPAK SINHA

McMASTER UNIVERSITY

## A. INTRODUCTION

This paper will be useful to anyone wishing to embark on the task of converting IBM FORTRAN software to the HP 3000 Series II or Series III computers. It should also be helpful to anyone who wishes to consider the general question of program portability to the HP 3000.

The material presented here is based primarily on the experience of the authors resulting from the successful conversion of the BMDP and SPSS statistical systems to HP 3000 Series II.

The problems encountered during these projects arose largely from the following four sources:

1) Incompatibilities between IBM/FORTRAN and HP 3000/FORTRAN.

2) Architectural features of HP 3000 Series II computer which impose certain restrictions on programs it can run.

3) Difference between the EBCDIC and ASCII character sets.

4) Difficulties with some library functions on the HP 3000.

The following sections attempt to cover the above problem areas and our proposed solutions in detail.

## B. FORTRAN incompatibilities between IBM and the HP 3000

### B.1) Type declaration order:

The order in which type declarations can be made is more restrictive on the HP 3000 than on IBM. All type declarations must be made before any DATA statements on the HP 3000. Further, the data declarations cannot be interspersed with type specifications. For example, consider the following FORTRAN code for IBM and its equivalent on the HP 3000:

```
IBM                              HP 3000

SUBROUTINE EX1                   SUBROUTINE EX1
 .                                .
REAL B(3)/1.,2.,3./,A,D          REAL B(3), A,D
INTEGER*2 I, J/9/,K              INTEGER*2 I,J,K
DATA K/5/,A/4.5/                 INTEGER*4 INTI, INT2
INTEGER*4 INTI, INT2             DATA B/1.,2.,3./
 .                               DATA J/9/, K/5/, A/4.5/
 .                                .
                                  .
END                              END
```

### B.2) REAL and INTEGER specifications:

IBM FORTRAN allows REAL*8 and REAL*4 type specifications. On the HP 3000 REAL*8 must be replaced by DOUBLE PRECISION and REAL*4 simply by REAL wherever they occur.

The use of INTEGER*4 and INTEGER*2 specification on IBM is compatible with the HP 3000 and needs no change.

It must be pointed out that the default integer size on IBM is 32 bits, i.e. any variable specified as INTEGER*4, INTEGER, or any integer constant (e.g. 1,2,99, etc.) has a length of 32 bits. On the HP 3000, the default integer length is 16 bits. This incompatibility can be easily removed, however, by including the $INTEGER*4 command ahead of the source-code before compilation. Consider the following equivalent examples on the two machines.

```
  IBM                      HP 3000

PROGRAM EX2              $INTEGER*4
REAL*8 A,B               PROGRAM EX2
REAL*4 C,D               DOUBLE PRECISION A,B
INTEGER I,X              REAL C,D
INTEGER *4 J,K           INTEGER I,X
INTEGER *2 L,M,N         INTEGER*4 J,K
 .                       INTEGER*2 L,M,N
 .                        .
J=K+10                    .
 .                       J=K+10
 .                        .
END                       .
                         END
SUBROUTINE XYZ           SUBROUTINE XYZ
INTEGER I,J,A            INTEGER I,J,A
 .                        .
 .                        .
END                      END
                         SUBROUTINE ABC
SUBROUTINE ABC           INTEGER X,Y
INTEGER X,Y               .
 .                        .
 .                       END
END
```

The lengths of the REAL and DOUBLE PRECISION variables on both the machines are 32 bits and 64 bits respectively. (Remember, though, that the length of DOUBLE PRECISION variables on the HP 3000 CX machine is 48 bits).

B.3) Mixed specifications:

IBM FORTRAN allows double precision and real variables to be declared in the same statement by appending *2 or *4 to the variable name. The IBM convention of appending *(number) to a variable or function name is totally unacceptable on HP 3000. Consider the following equivalent examples:

```
 IBM                          HP 3000

REAL FUNCTION EX3*8(N)    FUNCTION EX3(N)
REAL *8 A,B*4,C           DOUBLE PRECISION A,C,EX3
INTEGER I,J*2,K*4         REAL B
 .                        INTEGER*4 I,K
 .                        INTEGER*2 J
END                        .
INTEGER FUNCTION INT*2(I)  .
 .                        END
 .                        FUNCTION INT(I)
END                       INTEGER*2 INT
                           .
                           .
                          END
```

B.4) Logical variables:

There are two kinds of logical variables in IBM FORTRAN; One byte logical variables, which are typed using the LOGICAL*1 declaration, and 4 byte logical variables, which are typed using the LOGICAL declaration. All logical variables in HP 3000 FORTRAN have a length of 2 bytes.

LOGICAL*1 variables or arrays in IBM programs are often used to store character strings only, and logical tests are not performed on them. In this case, these variables or arrays can simply be typed as CHARACTER*1 on HP 3000 and they become exactly equivalent.

When LOGICAL*1 and LOGICAL variables or arrays are being used in the logical context (i.e. logical tests are being performed on them and they are set to TRUE or FALSE, etc.) then there are clearly two options:-

a) Declare them as LOGICAL on HP 3000, but with caution. What if those variables or arrays are equivalenced with some other arrays? Or, what if they are a part of some other big array, the starting address being passed through a subroutine call? Clearly, the IBM calculations for the space required by the array will need readjustment in HP 3000 programs. Also, since the lengths of IBM LOGICAL, REAL and INTEGER words are the same, an array declared as REAL or INTEGER in one subroutine can be declared as LOGICAL and interpreted logically in another subroutine and vice-versa. This operation is obviously invalid on the HP 3000 because of the length difference of a logical variable.

b) Declare a LOGICAL*1 variable as CHARACTER*1 and a LOGICAL variable as an INTEGER or REAL. All the logical tests, initializations and assignment statements will then have to be changed. Consider the following equivalent examples for this case:

```
IBM                HP 3000

PROGRAM EX4        PROGRAM EX4
LOGICAL*1 A,B      CHARACTER A,B,CTRUE, CFALSE
LOGICAL X,Y        INTEGER X,Y,ITRUE, IFALSE
IF (A) G=G+1       DATA CTRUE/%1C/, CFALSE/%0C/
IF (X) Y=FALSE     DATA TRUE/1/, IFALSE/0/
B = FALSE          IF (A.EQ. CTRUE) G = G + 1
 .                 IF (X. EQ. ITRUE) Y = IFALSE
 .                 B = CFALSE
END                  .
                     .
                   END
```

Another fact to note is that the bit representation for the constant TRUE is different on the two machines:

```
              IBM                                  HP 3000

Bit  0   1     30   31                  Bit  0   1       14   15
    [0 | 0|...|0 |  1 ] (=1) TRUE           [1 | 1|...|1 |  1 ](=-1)
    [0 | 0|...|0 |  0 ] (=0) FALSE          [0 | 0|...|0 |  0 ] (=0)
```

In view of the above facts it is not possible to give a pat solution for every problem arising out of the use of LOGICALs but it is recommended that every situation involving these variables should be examined carefully.

B.5) Hexadecimal constants:

IBM FORTRAN allows initialization of variables using hexadecimal constants (e.g. Z18005024 etc.). They are not allowed in HP 3000 FORTRAN and must, therefore, be replaced by octal or some other equivalent constant.

B.6) Branching control in subroutine calls:

The character "&" in an IBM subroutine call statement is used to control branching on return from the subroutine. It must be replaced by "$" on HP 3000. Consider the examples:

```
        IBM                       HP 3000

     PROGRAM EX5               PROGRAM EX5
     DATA A/Z12345678/         DATA A/%0221505317R/
     CALL SUB (A,B, &10)       CALL SUB (A,B, $10)
       .                         .
       .                         .
  10 I = I + 1              10 I = I + 1
       .                         .
       .                         .
     END                       END
```

B.7) Type incompatibilities:

IBM FORTRAN allows incompatibility between the types of the actual arguments (those provided in a CALL statement) and the dummy arguments (those within a subroutine). For example, a REAL argument can be passed to a subroutine whose corresponding dummy argument is INTEGER. The HP 3000 allows such incompatibilities only if a $CONTROL CHECK=2 statement has been included ahead of the subroutine before compilation. The following equivalent examples further clarify this point.

```
        IBM                       HP 3000

     PROGRAM EX6               PROGRAM EX6
     REAL A                    REAL A
     CALL SUB (A)              CALL SUB (A)
       .                         .
       .                         .
     END                       END
     SUBROUTINE SUB (I)        $CONTROL CHECK = 2
     INTEGER*4 I               SUBROUTINE SUB (I)
                               INTEGER*4 I
       .                         .
       .                         .
     END                       END
```

B.8) Array bounds:

A dynamic array bound must be a dummy argument of the subroutine statement in HP 3000 FORTRAN. Two equivalent examples are given below and some comments are made:

```
IBM                                 HP 3000

SUBROUTINE EX7 (A,B)                SUBROUTINE EX7 (A,B)
REAL A(N), B(M), C(L)               REAL A(1), B(1), C(1)
COMMON/ABC/M,N                      COMMON/ABC/M,N
READ (5,10) A                       READ (5,10) (A(I), I=1,N)
WRITE (6,11) B                      WRITE (6,11) (B(I), I=1,M)
 .                                   .
 .                                   .
ENTRY DUMB (C,L)                    ENTRY DUMB (C,L)
 .                                   .
 .                                   .
END                                 END
SUBROUTINE XYZ (P,I)                SUBROUTINE XYZ (P,I)
REAL P(I)                           REAL P(I)
 .                                   .
 .                                   .
END                                 END
```

It should be noted in the above examples that subroutine XYZ needed no modification because I, the dynamic bound of array P, is a dummy argument of subroutine XYZ; but M, N and L were not dummy arguments of subroutine EX7 and hence could not be used for dimensioning A, B, and C. Note that L is a dummy argument of entry DUMB but that is not sufficient. The implicit length of the arrays A, B, and C was changed to 1 but this required the modification of all the READs, WRITEs or any other statements using the implicit length of the arrays.

B.9) Basic external functions:

All the basic external functions in the HP 3000 FORTRAN, particularly the double precision functions, require explicit typing in a program. For example, DSQRT must be declared as DOUBLE PRECISION in the HP program, but this is not necessary in an IBM program.

B.10) Scientific subroutine library functions:

Certain IBM scientific subroutine library routines like GAMA, DGAMA, LGAMA and DLGAMA are available on HP 3000 but under the names GAMMA, DGAMMA, LGAMMA and DLGAMMA respectively. (Note the two M's in the spellings).

B.11) Length of common blocks:

IBM FORTRAN allows the length of a labelled (or named) common block to vary from one subroutine to another. HP 3000 FORTRAN permits this phenomenon only for a single blank (or unnamed) common block. All the labelled common blocks must be of the same length in every subprogram or else a segmenter error results. The following equivalent examples contain one proposed solution.

```
IBM                                HP 3000

PROGRAM EX9                        PROGRAM EX9
COMMON/A/A,B,C(10)                 COMMON/A/A,B,C(10), PAD (90)
 .                                  .
 .                                  .
END                                END
SUBROUTINE ONE (I,J,K)             SUBROUTINE ONE (I,J,K)
COMMON/A/P                         COMMON/A/P, PAD (101)
 .                                  .
 .                                  .
END                                END
SUBROUTINE TWO (X)                 SUBROUTINE TWO (X)
COMMON/A/A,Y,Z(100)                COMMON/A/A,Y,Z(100)
 .                                  .
 .                                  .
END                                END
```

In order to determine the maximum length of a common block, the source should be compiled with the $CONTROL MAP, CROSSREF ALL option. This enables one to know which surbroutines use a particular common block and what the lengths are of the common blocks in those subroutines.

## C. RESTRICTIONS DUE TO HP 3000 SYSTEM ARCHITECTURE

### C.1) Variables in COMMON or DATA:

The total number of variables declared in different COMMON blocks or DATA statements must not exceed 255 on HP 3000. The reason for this is that the compiler places the address of each such variable or array in the primary DB area of the stack and the DB relative addressing is not allowed to exceed 255 words by the system.

The problems imposed by this restriction have proven to be extremely difficult, especially in the conversion of a big system like SPSS.

One solution is:

a) Eliminate a common block by including all or some of its variables in the argument list of the affected subroutines.

b) Eliminate a DATA declaration by initializing its variables using assigment statements in the code. This process, however, cannot be carried out indiscriminately. There is one very important consideration to keep in mind when comtemplating this change. A DATA variable is like a global variable, in other words it retains its value throughout the program execution. For example, suppose that a subroutine, when entered once, set the value of some variable A; now, when it was entered again, if A was not a global variable, its value would be indefinite. By being removed from a DATA declaration, the status of a variable is changed from global to local. Hence, before removing a variable from a DATA statement one needs to understand the program logic to ascertain whether or not this particular variable needs to be global. Only those DATA variables, not required to be global, may be initialized by assignment statements.

It is the experience of the authors that this process can be tedious, time consuming and has the potential for introducing an unending string of "bugs" in a program.

A new capability likely to be available in the future version of the FORTRAN compiler may remove this restriction, at the expense of execution time, by providing the $MORECOM compiler option.

C.2) Subroutine arguments:

The maximum number of arguments in a single subroutine on HP 3000 cannot exceed 54. The reason for this limitation is that whenever a subroutine or function is called, the address or values of the actual arguments and a four word stack marker are placed on the stack and Q-minus addresses are assigned to them. Q-minus addressing cannot exceed 63 words, hence the limit.

C.3) Local variables:

On an IBM machine, the local variables within a subprogram retain their values between calls to this subprogram, unless this subprogram happens to be overlaid with another subprogram. In particular, programs with only one overlay always retain the values for local variables throughout a run of the program. This is never true on HP 3000 because the system stack is dynamically increased when a subroutine is called and decreased when the subroutine is exited. Consequently, all the local variables are lost with the updating of Q register.

The only solution to this problem is to understand the program logic so as to determine which variables need to be removed from the list of local variables and be made global. Once identified, these variables must be placed in a common block.

C.4) Addressing:

IBM and many other machines use a byte oriented addressing scheme. This means that given an address, the system can identify the proper byte in memory which may or may not be at a word boundary. HP 3000 system uses related but different addressing schemes for words and bytes. The right most bit of a byte address indicates whether it is the left or the right byte of the word whose address is given by the remaining bits. The implication is that given an address, the system also has to know whether it is a word or byte address.

The HP 3000 FORTRAN compiler generates word addresses for REAL, INTEGER, DOUBLE PRECISION or LOGICAL variables but byte addresses for CHARACTER type variables or strings. For this reason, it is not possible in HP 3000 FORTRAN, as opposed to IBM FORTRAN, to pass a character string or variable in a subroutine CALL statement when the corresponding dummy argument is not of a CHARACTER type. The converse also holds true.

One can use equivalencing of variables to solve this problem but a more innovative solution, and one that has been used in the SPSS and BMDP-77 conversion projects, is to use an SPL program to modify the address in question and pass this modified address to the called subroutine. The following examples should clarify this concept.

```
IBM                          HP 3000

PROGRAM EX10                 PROGRAM EX10
REAL A                       REAL A
CALL SUB1 (A)                CALL SUBIM(A)
  .                            .
  .                            .
END                          END
SUBROUTINE SUB1 (B)          SUBROUTINE SUB1(B)
LOGICAL*1 B(1)               CHARACTER B(1)
  .                            .
  .                            .
END                          END
                             $CONTROL SUBPROGRAM
                             BEGIN
                             PROCEDURE SUB1(B);
                             BYTE ARRAY B;
                             OPTION EXTERNAL;
                             PROCEDURE SUBIM(B);
                             REAL B;
                             BEGIN TOS:= @ B & LSL(1);
                             SUB1(*); END;
                             END.
```

Conversion of a word address to a byte address is straight forward but while converting a byte address to a word address one must note that if the byte address is not at a word boundary then there is no way it can be converted to a word address. However, such a situation rarely arises.

C.5) Code segmentation:

As opposed to specifying overlays on an IBM system, one needs to define code segments on HP 3000. All the code must belong to some code segment. There is an allowed maximum size and an allowed maximum number of code segments which are fixed at the time of system configuration.

The implication is that there is a limit to how large a program that runs on HP 3000 can be and how large a subprogram can be. If a subroutine, or any other subprogram cannot fit within one code segment then it must be split up into two or more subprograms. The task of splitting subroutines is a difficult one and generally introduces "bugs" in the program. We have found that a code segment size of 8K bytes will generally handle the largest of FORTRAN subroutines or programs.

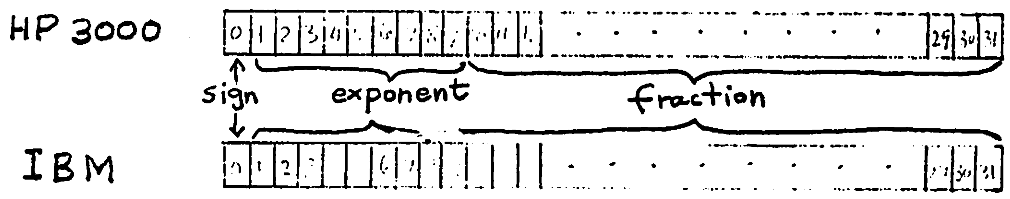
C.6) Data stack limitation:

The entire data stack used by a HP 3000 program must not exceed 32,767 words. This restricts the size of scratch areas and other arrays in a program. In order to use the stack judiciously, one must have as few global variables and arrays as possible so that there is more space available for local variables. There is a permanent allocation of stack for the global variables but a dynamic allocation for the local variables. This is one reason why a lot of DATA declarations in SPSS had to be eliminated.

It may be noted that in order to use the maximum stack size, the program should be prepped or run with STACK parameter set to 819 and MAXDATA parameter set to 32767.

C.7) Real word configuration:

The exponent and fraction parts of a real word on the two machines are as shown here:



In addition, on the HP 3000, a 1 is always implied to left of the binary point. Real 0 is the only exception to this rule.

It should be obvious that the range of magnitude for the real numbers is more on HP but the precision is smaller by one bit compared to IBM.

C.8) Distinction between BLANK and ZERO:

The IBM formatter upon detecting blanks in a field being read using Fw.d specification, turns the left most bit of the real word ON (the remaining bits are 0). This real word, while being distinct from +0, has an arithmetic value equal to 0. The HP 3000 formatter returns a +0 upon detecting blanks under the same conditions. Hence there is no way of distinguishing between blanks and zeroes while reading a field under Fw.d specification.

It may be noted that even if one, somehow, manages to turn the left most bit ON, the value of the real word will not be -0, it will be -.863617 E-77 because of the biased exponent and an implied 1 to the left of the binary point.

D. <u>INCOMPATIBILITIES DUE TO EBCDIC & ASCII CHARACTER SETS</u>

D.1) Alphabetic and numeric tests:

Very often, in order to determine whether a character is alphabetic or numeric, its numerical value is tested. As one would expect, the numerical values for the EBCDIC character set used by IBM are different from those for the ASCII character set used by the HP 3000. The following table compares some of the values assuming that the character in question occupies the right most byte of the word.

| | EBCDIC | ASCII |
|---|---|---|
| A - Z | 193 - 233 | 65 - 90 |
| 0 - 9 | 240 - 249 | 48 - 57 |
| BLANK | 64 | 32 |

The following equivalent examples include one instance of such a test.

<u>IBM</u>

```
    PROGRAM EX11
    LOGICAL *1 A(4)
    INTEGER *4 I
    EQUIVALENCE (I,A)
    DATA I/4Hbbb9/
    IF (I.GE.240 AND
    I.LE.249) GO TO 10
     .
     .
10  INUM = I - 240


    END
```

<u>HP 3000</u>

```
    PROGRAM EX11
    CHARACTER A(4)
    INTEGER *4 I
    EQUIVALENCE (I,A)
    DATA I/4Hbbb9/
    IF (I.GE.48 AND
    I.LE.57) GO TO 10
     .
     .
10  INUM = I - 48
     .
     .
    END
```

D.2) Alphabetic sorting:

Imagine a real or double precision array which needs to be sorted in alphabetic order. One method is to compare the numerical values of the words and take the appropriate action. The left most bit of every EBCDIC alphabet or numeral is 1, which in the position of a sign bit makes a number negative. The implication is that the value of A is greater than the value of B etc. In ASCII character set the left most bit of alphabets and numerals is 0 which implies that the numerical value of A is less than the numerical value of B etc. Hence, the logic of all the tests, performed on EBCDIC character strings to sort them in an ascending alphabetical order, will have to be reversed to sort ASCII character strings in the same order.

## E. PROBLEMS WITH HP 3000 SCIENTIFIC SUBROUTINE LIBRARY FUNCTIONS

### E.1) DFLOAT:

This double precision function, on HP 3000, always returns a zero regardless of the input. One has to supply his or her own DFLOAT function in the program or replace DFLOAT(I) by DBLE(FLOAT(I)) type of conversion.

### E.2) LGAMMA:

This function on HP 3000 aborts when the input is smaller than 7.0. One way to get around this problem is to use DLGAMMA which seems to work correctly.

### E.3) Initialization of DOUBLE PRECISION words:

As the HP 3000 FORTRAN manual suggests, it is not possible to initialize a double precision word using %"ABCDEFGH"D form. Only the first four characters of the string are stored. To get around this problem, one has to equivalence the double precision word with real words and initialize the real words.

## F. CONCLUSION

Although there are many potential areas of incompatibilities between IBM FORTRAN code and that on the HP 3000, the potential gains from converted software can make the effort well worthwhile. There is a great wealth of well written and thoroughly debugged software on IBM systems which will run on the HP 3000 once converted. We feel that our effort in converting the SPSS and BMDP-77 systems, for example, has been an extremely successful one. In fact, we are actively pursuing other FORTRAN software packages to convert to the HP 3000 as we have found that programs compiled from FORTRAN seem to run surprisingly efficiently on the HP 3000 computer system.

## G. ACKNOWLEDGMENTS

The authors would like to acknowledge the contributions of the members of the software group in the Computation Services Unit at McMaster and their assistance in finding solutions to the conversion problems described in this document. Their general support role in making these two rather massive conversion projects successful is also greatfully acknowledged. Dr. Khursheed Ahmed's assistance in the initial assessment of the projects and in the conversion design as well as his ongoing support have been most helpful. Mr. Kim Clark has proved to be a rich source of useful suggestions. Maria Wong has provided valuable input all through the projects with user interfacing problems and with testing and improving the final products.

Lastly, the authors would like to acknowledge the support of Hewlett Packard, SPSS Inc. and the personnel of the Health Sciences Computing Facility at UCLA.

## H. REFERENCES

1) Anderson, G. D., "Converting IBM 360 and 370 FORTRAN to the HP 3000 Series II" Journal on the HP 3000 Users Group, Vol 1, #3, Sept./Oct. 1977.

2) Anderson, G. D., "BMDP Program Conversion to the Hewlett Packard System 3000 Computer" Internal document available from Department of C.E.B., McMaster University, Hamilton, Ontario, L8S 4J9.

3) HP 3000 Series II Computer System, FORTRAN Reference Manual.

4) IBM System/360 and System/370 FORTRAN IV Language Reference Manual.

# WRITING SPL ROUTINES WHICH ARE CALLABLE FROM BASIC

BY

WARREN KUEHNER
SYSTEMS ENGINEERING SUPERVISOR

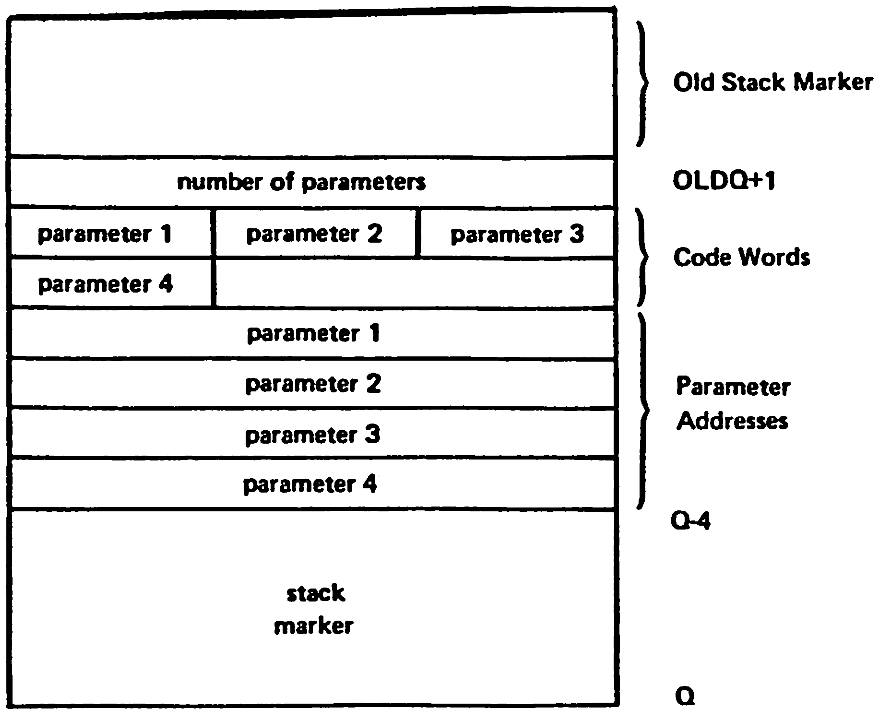
HEWLETT-PACKARD
NEELY SALES REGION
ENGLEWOOD, CO.

This paper discusses programming techniques involved in the creation of SPL routines to be called from BASIC. It specifically covers both trivial and more complex programming examples, and consideration, for putting these routines into operation with BASIC programs. This paper assumes a good knowledge of both SPL and BASIC.

## HOW BASIC CALLS A SUBROUTINE:

The key to understanding the possibilities for BASIC callable SPL routines is to understand what additional information (other than that which is normally passed) BASIC provides with a call. In addition to the passed parameters and the stack marker, BASIC places two other pieces of information on the stack ahead of the passed parameters. One is the number of parameters passed. The other(s) are a code word for each passed parameter indicating what data type it is and whether it is a simple variable or an array. (Refer to Appendix F of the BASIC/3000 manual for information on the codes and the passed parameters). This information can be used by the SPL routine to verify the correctness of passed parameters. Another use is to allow the passage of a variable number of parameters to the SPL routine.

When BASIC calls an external routine, the stack is as illustrated on the following page.



BASIC first calls a routine (which generates a stack marker) which places an integer indicating the number of parameters, the parameter codes, and the addresses of the passed parameters (in that order) on the top of the stack, and then calls the desired external routine.

## THE TRIVIAL PROGRAMMING CASE:

Given the foregoing background information, it should be obvious that in the trivial case, SPL routines can be written in the normal way and called from BASIC. For "quick and dirties" this could be adequate. It should be noted that BASIC does no passed parameter checking of any kind and this approach is potentially risky.

THE MORE COMPLEX CASE:

It has been pointed out that the additional information on the stack can be useful for at least two reasons; parameter checking and variable parameter passing. The following program example illustrates how to get at this information:

```
INTEGER DELTAQ = Q - Ø; (( THIS VARIABLE, WHICH IS LOCATED AT
                           Q (WHICH CONTAINS THE DISTANCE BACK
                           TO THE LAST LOCATION OF Q) ALLOWS
                           US TO FIND OUR WAY AROUND))

INTEGER POINTER NUMPARM;(( POINTERS TO THE VARIABLES CON-
                CODES,     TAINING THE NUMBER OF PARAMETERS,
                PLIST;     THE PARAMETER CODES, AND THE BASE
                           OF THE PARAMETER ADDRESSES))

@NUMPARMS: = (@DELTAQ + 1) - DELTAQ; (( NUMPARMS MUST POINT
                                        TO THE ADDRESS OF "Q"
                                        PLUS 1 LESS THE VALUE
                                        OF DELTAQ))

@CODES: = (@DELTAQ + 2) - DELTAQ:    (( SEE ABOVE))

@PLIST: = @CODES + (NUMPARMS + 2)/3; (( THIS CALCULATION WILL
                                        CAUSE PLIST TO POINT
                                        TO THE ADDRESS THE
                                        FIRST PASSED PARAMETER))
```

By indexing through PLIST and equating its contents to other pointers, one can locate any of the passed parameters. For example, to locate the third passed parameter which is a string (byte array) the following is necessary:

```
     BYTE POINTER STRING;
     @ STRING: = PLIST (2)
```

Obviously, the problem of checking parameters can be solved, being able to get at the number of parameters and their types.

The variable parameter passing problem can also be solved with this information.

The idea of variable parameter passing can be very useful. Good examples are the Basic callable Image routines. DBGET can, for example, be passed a variable number of strings to receive the buffer of data from the data base. This gets around the limitation on string length as well as allows the placement of logical pieces of the data in the data base in strings dimensioned to contain them.

It should also be noted that information about the length of a string variable (and also about array dimensions) is available to the SPL programmer, and that this too can be quite useful.

For example, if STRING is a string variable (byte array), then STRING (-1) contains its length as dimensioned in the BASIC program. (Refer again to Appendix F of the BASIC/3000 manual for more information.)

NOW I WROTE IT---HOW CAN I MAKE IT RUN?

If one is running his BASIC programs from the interpreter, the external routine called must be located in an SL file, either in his group, in his account's PUB group, or in the system SL. The interpreter automatically searches all three. To place the routine in the SL, one must use the SEGMENTER, and the procedure is as follows:

```
: SPL MYPROG, MYUSL
$ CONTROL SEGMENT = MYSEG
  :
  END.
: SEGMENTER
- USL MYUSL
- SL SL ((BUILD AN SL IF YOU DON'T HAVE ONE))
- ADDSL MYSEG
- EXIT
: ----NOW RUN!
```

To use the routine from a compiled program, the steps are the same as for a compiled program in any language, that is, the routine may be either:

- Compiled into the same USL file as the BASIC program and then that USL PREPED.
- Put into an RL file (with the SEGMENTER) against which the USL can be PREPED

```
: PREP MYUSL, MYPROG; RL = MYRL
```

- Called from an SL. Remember if the SL is in your group to use the appropriate LIB = parameter, that is:

  ```
  : RUN MYPROG; LIB = G.
  ```

The purpose of this paper was to present the tools to write BASIC callable SPL routines. I would enjoy your comments, criticisms, or to hear what you've been able to do with these ideas.

# DEC/3000

# AN EXAMPLE OF SPECIAL-PURPOSE LANGUAGE DESIGN AND IMPLEMENTATION

Matthew J. Balander
The B & B Computer Company

ABSTRACT: Special-purpose programming languages are often an excellent method of reducing costs and improving productivity. This paper discusses some of the factors influencing the decision to develop and employ a special-purpose language, and presents an example of such a language. The language presented, DEC/3000, provides powerful data-entry capabilities for programmers producing applications in host languages. In tests, the use of this special-purpose language has indeed reduced costs and improved programmer productivity.

## 1. Why another language?

Despite the (quite proper) effort to develop good general purpose programming languages in the discipline of computer science, there are nonetheless still problems for which special-purpose languages should be emphasized regardless of considerations of universal applicability or portability. Factors which motivate the effort to design and implement such languages, to adopt them for use in applications, and to educate programmers in their use include cost effectiveness, programmer productivity, program reliability, and maintainability of the final product.

Languages should not of course be implemented to meet every whim of each programmer. Problem areas for which languages should be provided must be very carefully defined, if the language to be provided is to fulfill its goals of lowered software production costs and improved programmer productivity. An appropriate problem area may be identified as follows:

- o It must be possible to clearly define the problem area. It is, for example, an inadequate isolation of the problem area to state, "A language is needed to help with data entry tasks." The problem area must be carefully, clearly, and fully specified. The construction of a "language" is under consideration. It is critical to understand fully exactly what the proposed language is supposed to be able to "talk" about.

o The programming of solutions in the problem area, using available languages, is tedious, complex, and error-prone. Existing languages, in other words, are not well suited to the expression of solutions to tasks in the problem area. The degree of difficulty with which solutions are provided in existing languages is a measure of the cost of solving problems in those languages, and, in general, an indication of the future costs of reliability and maintainability of the software. (The cost of the proposed language may thus be measured relative to the cost of existing languages by noting the reduction in complexity of solutions provided in the special-purpose language.)

o An area is a candidate for a special-purpose language if programs are frequently produced to meet needs in the area, or such programs are heavily used. The effort involved in designing and implementing a special-purpose language cannot be justified if the language produced will rarely be used.

The decision to produce a special-purpose language is thus largely governed by economic factors. If a special language will reduce costs sufficiently, it will provide a cost-effective means of providing solutions to data processing problems. Languages achieve cost reductions in the software development cycle in several ways. Some of the more straight-forward are as follows:

o By increasing the power of the language being used by programmers (defining "power" loosely as the amount of work done for the programmer by a line of code) the size of programs, measured in lines of code, is reduced. Given a fixed cost per line of finished code regardless of language, there is a direct savings in program development cost in using a powerful special-purpose language.

o A well-designed language is easy to use. Its idioms and methods of expression are "natural" for expressing solutions to the category of problems for which it was designed. This reduces the complexity of solutions to problems. Complexity of a program varies inversely with reliability of the program. The less complex a program is, the more reliable it is. Greater reliability means lower costs for software development. As a result of reduced complexity, testing is simplified, and debugging is both easier and less time consuming.

- The increased power and reduced complexity of a special-purpose language serve to reduce software maintenance costs. Adjustments and modifications of programs will require less programmer time, affect fewer lines of code, and may with greater confidence be undertaken by programmers who did not originally write the software being modified. Program modification in addition enjoys the same benefits from increased power and reduced complexity that are experienced during original software development.

From all these remarks it is plain that the use of special-purpose programming languages can be very beneficial. It is also plain that the benefits received depend heavily on the language design and implementation.

## 2. How is a special-purpose language designed?

Once a problem area has been defined for which a special-purpose language is being considered, the language must be designed. The design of a language is an art, not a technology. It will always remain so, since language is an intensely human activity, and is fundamentally alien to mechanical processing. It is far beyond the scope of this paper to discuss language design in detail, but some central considerations should be mentioned.

The language designer must know the problem area. This involves not just a passing acquaintance, but an intimate familiarity. Such knowledge usually is a result only of much work in the field. The language designer must be familiar with the problems of the problem area, and with the sorts of solutions with which they are best met. He must be aware of the limitations of available languages when applied to problems in the field, and have some experience with trying to "make do" with these languages.

The language designed must be powerful. If it is not sufficiently powerful, many of the benefits of implementing the language will be lost. Language constructs and idioms must be provided which allow the programmer to express as concisely and unambiguously as possible his intentions. Brief constructs are preferred to wordy ones, and one line of code to two or more. On the other hand, the APL-ish, mystical, "does-it-all" one-liner type of language is most certainly not desirable. Readability should always be maintained.

When providing power, it is necessary to be cautious that flexibility is not unduly limited. Regardless of the power of the language, it will not be used by programmers if they are unable to express solutions to all relevant problems with it. In zeal for making the programmer's life easier, by "doing all the work for him," care must be taken not to make it difficult or impossible for him to solve atypical or unanticipated problems using the language.

The language designed must be "natural." That is, it must express programs in much the same way that the programmer thinks, or should think, about them. A judicious compromise between machine efficiency and programmer comfort must be found. It is here that the art of the language designer will be most needed.

3. How is the language to be implemented?

While language design is the most critical phase of the production of a language, implementation is by no means less important. A good, reliable, easy-to-use implementation of the language must be provided to programmers. It must take into account their needs and work. An implementation which is difficult to use, or which is unpredictable or unreliable in its operation, will not be used. The documentation, not only of the language, but also of the implementation of the language, must be comprehensive, clear and concise. It is an integral part of the implementation as a whole.

There are three very viable options for implementation of a language: a pre-processor, an interpreter, and a compiler. The simplest of these is to implement a pre-processor for some existing language. The special-purpose language then takes the form of statements embedded in a program text ultimately intended for some existing processor. This text is first fed through the pre-processor, which converts the special-purpose language statements into source statements of the language in which the special-purpose statements are embedded, and outputs a program text which may be submitted to the existing processor. Such a concept is familiar and has been used successfully in such systems as RATFOR, a pre-processor for FORTRAN programs providing while, do-until, and similar constructs.

If all existing languages are so unsuited to the problem area that it is not wise to pre-process special-purpose language statements into one of them, then the implementation must provide all the functions of a source-language processor. One way in which this is often accomplished is to implement an interpreter for the language. Interpreters have many familiar advantages, and are not as difficult to implement as most other options. In addition, the technology of interpreters is well documented in the literature, and good advice may be found there to assist in writing the interpreter itself.

Although interpreters have many well known advantages, they also have many well know disadvantages. They are slow, often bulky, and usually have no convenient mechanism for accessing the full capabilities of the operating system and allied subsystems in which they are used. It the disadvantages of an interpreter weigh heavily in a particular situation against its use, the next alternative is to implement a full compiler. Implementing a compiler is undoubtedly the most difficult and time-consuming of the options, but possesses most of the advantages of the other options, as well as some unique to the use of a compiler. A well written compiler provides access to all system capabilities (assuming the language design has been done well enough to allow the language itself to express access to these capabilities). It provides much more efficiency than an interpreter, and is more congenial to a variety of environments, both batch and interactive.

The type of implementation to be used depends on the nature of the problem area itself, on the time and resources available for implementation, and the ways in which it is desired to use the finished processor. Pre-processors are the easiest to write, port, and maintain, compilers the most difficult, and interpreters are somewhere in between. In some cases hybrid implementations are most appropriate. Here the craftsmanship of the programmer implementing the language is of great importance. If the available programmers are not skilled in compiler writing, or have no experience in compiler writing, it may be unwise to ask them to implement a production compiler. On the other hand, the concepts and engineering of pre-processors are not of great difficulty, and most programmers with some experience in text processing can implement pre-processors in a reasonable amount of time. Interpreters, as compilers, require some special skills, and should not be undertaken by a staff without some prior experience in the area.

## 4. Can you give me an example?

The author has written a number of data entry programs, each of which was a portion of a larger data processing system. In each case, interactive terminals with forms and block mode capabilities were used. These programs were implemented in FORTRAN, COBOL, and SPL, as appropriate to the data processing system for which the data entry program was being written. The specifications for the programs indicated that the forms/block mode capabilities of the terminals were to be used whenever possible to assist terminal operators in entering data correctly.

In preparing to write these programs, it was necessary to learn a great deal about the terminals to be used (HP264X terminals were used for these particular projects). Not only was it incumbent on the programmer to be fully conversant with the application, but also to be an expert on the terminals themselves. Programming terminals such as HP264X terminals is in fact a form of low-level machine programming, and as such is quite prone to errors. No special software tools were available to assist in managing the terminals; all functions had to be explicitly provided by the applications programs.

In one case, eighty lines of SPL code were needed for the "DEFINEs" used to code the declaration of a single form in a fashion which was at least partially readable. In one FORTRAN program approximately two hundred lines of code were devoted exclusively to terminal management. In general, it was found that approximately one third of each of these programs were devoted to terminal management, with approximately one hundred lines per program devoted to defining and managing forms for display on the terminals.

Not only were terminal and form management lengthy, they were also difficult to code. The escape sequences used with HP264X terminals are meaningless in and of themselves. They are simply details which must be coded precisely in order for the terminal to behave as desired. Programmers who are not the original author of these programs will find it difficult to modify the forms involved when data needs change; the original author himself will find changes difficult.

A search was of course made for alternative methods of coping with the problem of managing these types of terminals during data entry. The only software available at the time was Hewlett Packard's DEL/3000. It was evaluated, and rejected for several reasons. First, it did not relieve the programmer of the necessity of being an expert on the terminals. To declare a form in DEL/3000 the programmer must actually produce the form from the keyboard of an HP264X terminal. He must still be familiar with all appropriate escape sequences. Second, DEL/3000 had to be used from an HP264X terminal. Not all terminals used for program development at the installation where this work was being accomplished were HP terminals. Also, because of this restriction, work on the forms and terminal management had to be done on-line instead of in (less expensive) batch. Third, DEL/3000 held out no hope of ever supporting any but Hewlett Packard terminals. Other forms/block mode capability terminals are available, and it is not wise to write one's software so that one is locked into a single vendor. Other objections, such as the "un-aesthetical" nature of DEL/3000, were raised, but could not be related to clearly definable, measurable problems.

The rejection of DEL/3000 was the rejection of the only available software tool for forms/block mode data entry with sufficient power to be considered. The remainder of this paper discusses the alternatives which we considered, and describes the solution we reached. We turned to consider what sort of tools we should implement ourselves. (In the meanwhile, the writing of data entry programs was undertaken using the various available languages, since it was not feasible to delay their production till more powerful tools were available.) What ever we came up with had to meet the following goals:

- o The tool produced must perform virtually all terminal and form management during data entry.
- o The tool must relieve the applications programmer of the necessity of understanding the programming of the terminal being used. The programmer should never need to know what escape sequences are used to perform terminal functions.
- o It must not be necessary to have one of the data entry terminals available for the programmer's use during coding of the application.
- o The tool must provide for adequate documentation of forms. It is not considered sufficient for forms to be simply entered at the keyboard; there must be a permanent, off-line record of each form clearly showing all protected/unprotected fields, display enhancements, and so on.
- o The tool must allow for future use of non-HP terminals with a minimal conversion effort. It is acceptable to need to rework the tool itself, but minimal changes should be necessary in applications programs.
- o The tool must dovetail with existing applications, and existing application languages, so that a conversion process may be undertaken, and so that programmers and analysts are not restricted in their choice of language.

Clearly a new programming language was called for. A carefully designed language, together with a comprehensive run-time support library, is capable of meeting all these goals. Implementation method was decided immediately. A pre-processor is not feasible because of the last goal. The new language must dovetail with a number of existing languages (at the least with COBOL, SPL, and FORTRAN), but pre-processors are oriented to a single application language. The difficulty of implementing several pre-processors, and the maintenance nightmares arising from having three or more programs that do the "same" thing, caused the firm rejection of a pre-processor. An interpreter was rejected for the same reasons that the pre-processor was rejected. (It is possible to construct an interpreter which is invoked by a program when needed, but such constructs are difficult on the HP3000 under MPE. Also, the use of such a system is burdensome on the programmer, and is not conceptually straight-forward.) we settled on implementing a compiler.

Since it is desirable to use the data entry tool with a variety of applications coded in a variety of languages, a "host language" concept was adopted. That is, program units coded in the new, special-purpose language would be used together with program units coded in some "host language." It was agreed that only the following restrictions should be placed on the choice of host language: the host must be able to call external program units, and to pass simple integer and integer array parameters by reference to these external program units. In this way, the special-purpose language program units would be accessable from at least COBOL, BASIC, SPL, and FORTRAN. Since no other potential hosts were in use at the time, this was considered to be sufficient.

The name "DEC" was adopted for the compiler and associated language. It is an acronym for "Data Entry Compiler." Since all DEC programs were to be used together with a host program, DEC could be devoted to solely the data entry/terminal management tasks which motivated it in the first place. Access to data bases and other files, complex data checking, and complex program logic could be left to the host. The following scope for the DEC language was defined:

> DEC is a special-purpose programming language in which data entry forms and activities utilizing forms/block mode terminals may be expressed. This is to include the definition and documentation of forms, the specification of elementary data editing and checking, the specification of type conversions, and the specification of the correspondence of data record fields with form fields.

This definition guided language design. The language designed, which is described in separate documentation, is primarily declarative in nature; it includes no explicit verbs. Actions are implied by the declarations (for example, type conversion actions are implied by the declaration of data types), but no actions are explicitly coded by the programmer. A sample DEC program is included at the end of this paper. It is heavily commented to explain the features of the language.

Terminal management tasks are shared between the compiler and the run-time support library, "DECLIB". All terminal manipulation may be performed via library procedures, and

DEC-generated procedures. There is a single entry point to DECLIB for all functions, regardless of terminal type. The DECLIB procedures are internally structured in such a way that additional types of terminals may be easily accomodated, while requiring normally only a single line in each application program to be changed to take advantage of these additional types.

Since DEC is a compiler, it inputs a source language file, and outputs RBM's in a USL file, and thus may be used during coding and program entry from any sort of terminal. It may be used either interactively or in batch.

The DEC language has been designed so that it documents forms very well, as may be seen by examining the sample at the end of this paper. All features of the display are evident from even a rapid examination of a source program. In addition, the programmer has great flexibility in placing comments in a DEC program. This encourages good documentation.

## 5. what have the results been?

The six goals for the DEC language have substantially been met. A six hundred line FORTRAN program has been reduced to two hundred lines of FORTRAN and about one hundred lines of DEC. Again defining "power" loosely, since one line of DEC replaced in this application four of FORTRAN, DEC may be said to be roughly four times more powerful than FORTRAN. This implies that DEC programs cost about one fourth as much to write as equivalent FORTRAN programs, and this has indeed been our experience. All programs converted to use DEC, regardless of language, were substantially reduced in size.

As an additional benefit, unintended but very welcome, all converted data entry programs now work on all our HP264X terminals. This is despite the fact that individual programs were originally implemented for a particular model, such as the HP2645A, or the HP2640B. In addition, there is now complete freedom to use page or line mode. The programs were formerly hard-coded to use one mode or the other.

Programmers have learned DEC in a very short time. A single afternoon is sufficient to read the manual and begin applying DEC to actual problems. Since DEC is straignt-forward, concise, and "natural," programmers find it easy to use.

Changes to a form, and to its associated data entry program, can now be made in a single afternoon, and in many cases in much less that an afternoon. Formerly one or two days or more could be consumed in such changes, depending on the extent of the change. Using DEC, a form can be entirely reorganized, and be back in service the same day.

## 6. Can you summarize all this?

In general, there are circumstances in which special-purpose programming languages can be used to good advantage. The design, implementation, and adoption of such a language can often be justified on a cost basis. While language design is a difficult art, analysts and many programmers with appropriate experience should be able to construct a language which would provide the desired cost benefits. Several options are available for implementing the language. The cost effectiveness of these options depends in part on the experience of the programming staff, and in part on the expected cost savings from use of the language.

An example has been presented for which available software products were unable to fill a well-defined need. Goals were established which upon examination indicated that a special-purpose programming language would be a viable method of obtaining a satisfactory software product to fill the identified need. An appropriate language was designed and implemented using a compiler and host language arrangement. The result was that the goals which had been established were substantially met, and in fact the cost of producing software using the new language is significantly lower than the cost of similar software produced without this tool.

```
* <--- LINES BEGINNING WITH "*" ARE COMMENTS
<< COMMENTS MAY ALSO BE ENCLOSED IN ANGULAR BRACKETS >>

$CONTROL USLINIT,LIST,SOURCE,MAP,CODE
* Note that compiler control is similar to HP compilers.  This
* specifies that a listing is to be produced, that it should
* include the source images, a symbol map, and code.  Code output
* is in an assembly-like format for readability.

********************************************************************
* A DEC program consists of a declaration of a form, followed     *
* by one or more "data entry," or "DE," sections which declare    *
* data entry information associated with the most recently        *
* declared form.                                                  *
********************************************************************

-FORM  FORMPROC  << name of procedure will be FORMPROC >>
  << You could call this procedure in COBOL with 'CALL FORMPROC >>
  << USING . . . or in FORTRAN with CALL FORMPROC(. . .)       >>

  0,0,"THIS IS A SAMPLE FORM"
  * This statment says, beginning in row 0, column 0, place the
  * following data: "THIS IS A SAMPLE FORM"

  2,0,"'  '"
  * This statement says, beginning in row 2, column 0, place a
  * two character unprotected field.  The initial contents of
  * this field are to be "  ".

  3,0,"'XX'"
  * Same as last statement, except row 3, and initial contents "XX".

  4,0,"!B'XX'"
  * Same as last, except row 4, and the "XX" is to be in inverse
  * video (display enhancement B).

  5,4,"\C!DTHIS WILL BE IN ALTERNATE CHAR SET C, ENHANCEMENT D"
  * At row 5, column 4, place the specified data using alternate
  * character set C, display enhancement D.

  LABEL:  12,12,"DATE:  !B'  '/'  '/'  '"
  * The terminal operator will see "DATE:    /  /  ", the last eight
  * characters of which will be in inverse video (display enhancement
  * B).  Note that there are only six unprotected characters.  Even
  * though these six unprotected characters are in three separate
  * unprotected fields, the fact that the field has a label ("LABEL")
  * indicates to DEC that all six characters should be considered
  * a single data field.  The label also allows reference to this
  * field in a later data entry section.

-MROF  << end of declaration of this form >>
```

```
-DE IN=INPUTPROC, OK=OKPROC
  << Declare data entry information associated with FORMPROC >>
  << Can later in host program CALL INPUTPROC(. . .), etc.   >>

  << We can reference fields defined in the form section FORMPROC >>
  << only by referring to labels declared there.  "LABEL" is the  >>
  << only such label in this example.                             >>

  0,4,P,LABEL; VERIFY NUMERIC-NO-BLANKS; SAVE
  << Beginning in byte zero of a data record, there is a 4-byte   >>
  << long, packed decimal field.  Its data source is the form     >>
  << field labeled LABEL.  Verify that the data input by the user >>
  << on the form is  all numeric, with no blanks.  After data has >>
  << been successfully input, and you go back to erase all the    >>
  << unprotected fields in the form, do not erase this one, SAVE  >>
  << it.                                                          >>

  * In addition, data editing (including justifying, blank and zero
  * filling, and the like) can all be specified.  Constants may also
  * be specified as a data source, instead of a field declared in a
  * form section.  Totaling of fields and auto incrementing or
  * decrementing of field contents may all be specified.

-ED   << ends the data entry section >>

********************************************************************
* The host language program would be similar to the following:     *
*                                                                  *
* OPENDETERM(CBUF,. . .);  << OPEN THE TERMINAL FILE (DECLIB) >>   *
* FORMPROC(CBUF);    << DISPLAY THE FORM >>                        *
* INPUTPROC(CBUF,. . .);  << INPUT DATA FROM THE FORM >>           *
* << USER MAY HERE DO ANY DATA MANIPULATION DESIRED.  IF NOT       *
*    SATISFIED WITH DATA ENTERED, RE-EXECUTE THE INPUTPROC         *
*    STATEMENT AFTER GIVING USER A DIAGNOSTIC.  IF ALL IS OK, THEN *
*    CONTINUE. >>                                                  *
* OKPROC(CBUF,. . .);  << ERASE SCREEN EXCEPT "SAVE" FIELDS >>     *
* << REPEAT THE INPUTPROC/OKPROC SEQUENCE UNTIL ALL DATA IS IN >>  *
* CLOSEDETERM(CBUF,. . .); << WHEN ALL DONE CLOSE FILE (DECLIB) >> *
********************************************************************

-END  << ENDS DEC PROGRAM >>
```

# DATA COMMUNICATIONS

Series "F"

PRESENTATION TITLE: Cutting Communications Costs with Statistical Multiplexors

INDIVIDUAL(S) NAME(S): Roger L. Evans

ADDRESS: Micom Systems, Inc.
9551 Irondale Avenue
Chatsworth, CA 91311

ABSTRACT:

Most HP users use Teletype or teletype-compatible terminals. Such terminals cannot be "multidropped" to allow several to share one telephone line. Neither do they incorporate retransmission-on-error, so high-speed operation (above 1200 bps) is often subject to unacceptable error rates caused by normal telephone network noise. But high speed is desirable to minimize operator wait time during interaction with the system.

In the past, minicomputer users wishing to support several CRT's and a printer at a location remote from the computer have had to use several phone lines (each equipped with modems) or use time-division multiplexors (TDM). The weaknesses of the TDM are that it is inefficient and provides no retransmission-on-error.

The microcomputer has permitted implementation of a new generation of concentrator or "statistical multiplexor" which provides end-to-end error control and dynamic bandwidth assignment on the shared telephone line. The MICOM Micro800 Data Concentrator is the first statistical multiplexor to be designed specifically to meet the needs of the minicomputer user. It costs no more than a TDM. As a result it can be used in a cost-effective manner on the relatively short-distance telephone lines which are characteristic of the minicomputer system user rather than the large corporate network.

## SYSTEM DEVELOPMENT

Series "G"

# Programming For Survival

by

Gerald T. Wade
Product Specialist

Hewlett-Packard
Neeley Sales Region
Englewood, Colorado

## ABSTRACT

This article deals with programming techniques to generate maintainable programs which will survive the loss of their author. It is geared primarily for the individual programmer rather than the leader of a programming team but it should be useful to both. The techniques described apply both to the creation of an entire program or only to the creation of one subsystem, as is the case in a team effort. One of the desires of the author is to promote "Egoless Programming" with overall program quality rather than individual programmer mystique as the end result.

## FORWARD:

The purpose of this article is to discuss programming techniques which will aid in the maintainability, hence the ultimate survival, of programs. I take as a basic premise that few if any programs are ever written such that they never require modification. This modification may be necessary due to changes in program requirements, operating systems, host computer, etc. or to ´Bugs´ found in the program. Whatever the reason, very few programs escape the need for maintence. As a corollary I purpose that in many cases the modification of a program will be made by someone other than the original programmer. In general programs tend to be written to satisfy the needs of a particular site and thus tend to remain at that site even after their original programmer has left in search of a better position. The conclusion that must be drawn is that in general a program will have to be modified and that such modification may be done by other than the original programmer.

If I may digress a moment into historical speculation: In the early days of computer programming the prime constraint on the programmer was his hardware. Hardware tended to be bulky and extremely expensive. As a result an installation would have to operate on the least amount it possibly could. One of the main hardware restrictions was the amount of real memory available in the computer. Virtual memory systems came along later to ease the situation but they brought with them penalties in program speed and complexity. As a result much emphasis was placed on coding a program such that it required the least amount of memory possible. Programming techniques developed with this goal in mind. For example; Variable locations might be reused for several purposes as the program progressed. This saved using a separate location for each use but made it difficult to determine the exact contents of that location as the program ran. Sections of code were often overwritten with data arrays after they had been executed. This technique saved much memory but could be very confusing if a modification tried to use the overwritten code. A similiar technique was to modify a section of code in order to configure it to perform various functions as the program progressed. This was done primarily in assembly or machine language but it too could cause much difficulty for someone not aware of what was being done.

All of the difficulties mentioned were dismissed by the programmers by explaining that they were smart enough not to do anything like that. While this might seem to be a valid argument I shall draw on my speculative history to discredit it. Hindsite shows us two basic occurences.

First, as I have mentioned, these programs were often required to be maintained by other programmers after the original author left. This means that someone else would have to learn what tricks had been used in order to avoid errors. If the programs were well documented and told just what had been done then this would be no serious problem. This was generally not the case. Program documentation was skimpy and often even misleading. I attribute this fact to the programmers ego and his sense of survival. Remembering that the prime method for determining a good from a bad program, and by association a good from a bad programmer, was how little memory was used, the programmer was naturally reluctant to reveal all the tricks he used to achieve this objective. By keeping these techniques to himself he would be better able to stay one step ahead of his competitors and boost his own ego or ´mystique´. Extensive documentation was often omitted as a time consuming task with little reward for the programmer. Thus the program documentation was not sufficient to allow another programmer to modify or maintain the program. Second, even if a programmer was modifying his own program at a later date he might have forgotten all the tricks he had originally used and thus fall into the same traps as the outside programmer.

The result of these types of problems often was that programs became ineptly patched, slower to execute, and unreliable. As an end result they would have to be thrown out and a new program written in their place, even when the changes to the original program were all small ones. We have set the stage for much duplication of programming effort and a bad reputation for the computer industry due to the unreliability of its programs.

What of the present and future? While technology has been advancing at a blinding rate and removing most if not all of the original constraints on the programmers, they have failed to update their techniques to keep pace. Memory cost and bulk has been so reduced that most installations can now afford to buy essentially all they need. With the advent of high speed secondary storage devices, (drum, disc, and soon magnetic bubbles), virtual memory systems become much more viable. What we have then is a totally new environment for the programmer. In the past he had to conserve memory by whatever means possible but today he can sacrifice some program size for clarity and maintainibility.

The rest of this article will deal with methods to write programs such that they will (1) accomplish their objectives reliably (2) be as simple as possible to maintain and modify, even by a programmer other than the original author (3) possibly survive and remain in use long after the original author has gone. It is my contention that the ability to write simple to understand, reliable programs that remain in use is a far better goal than to be thought a 'magic man' for the ability to write programs that no one else can understand.

The first sections of the article deal with general techniques that might be applied to any computer system. The remainder of the article deals with special techniques that may be used in order to write programs for the HP-3000 system.

## PROGRAM STRUCTURE

The term STRUCTURED PROGRAMMING has become a popular buzz word in todays society. As such its definition has become so twisted as to make it almost a useless term. Let me define what I mean when I say that a program should be properly structured.

A program may be thought of as existing at several different levels. The outermost level is the grossest look at the program and answers the question. Just what is this program going to do ? or Why was this program written ? The next level breaks the program into major functions or ´blocks´. For example, there might be an initialization block, a function selection block (if the program has multiple functions), a block to perform each function, a block to deal with anticipated error handling, and a block to handle any finishing housekeeping, such as closing files, printing summaries etc.

Each block within the program may then be further divided into smaller sub-blocks. A sub block might contain the code to read the input file, or to perform a sort on the data etc. The key thing about a sub-block is that it must be small enough to be fully comprehended by someone reading the program. This means, in practical terms that it should absoultely be no longer than one page of source code, preferable about one half page in length. One and only one operation should be preformed in the sub-block. Thus it would be improper to create a sub-block that reads and sorts the input file if indeed those are two separate operations. The beginning of each sub-block should have a comment that describes the operation to be performed and the person reading the program should be able to verify that the code in the sub-block does indeed perform just that operation.

This approach to structuring a program is also known as the "Top Down" programming method or the method of "sucessive redefinition." Many papers and books have been written about these methods and I will not elaborate further here except to say that I normally only follow the methods in gross structure but not in actual implementation. Some of them involve large amounts of formal structuring that I see as time consuming and appropriate only for the largest of programming tasks. For most programs it is only necessary to keep the concepts in mind during the programming, not to generate large amounts of paperwork.

## THE PROGRAM DEFINITION STATEMENT

At the start of each program there should be a comment that states the exact purpose of the program. It should answer the questions "Why was this program written ? What does it do "? The answers to these questions might seem all to obvious at first but their answers must be fully understood in order to direct any further development of the program. I was surprised to find the large number of programs that were written and actually being used without stopping to ask just what it was that they were trying to accomplish. An example might be a program that was written to "analyze the system log files". While this might sound like an admirable objective, it leaves many questions unanswered. Will this program allow me to print a report showing the number of power failures logged on a given day of the week ? Can it tell me how many lines each of the system users have printed on the line printer ? Depending on the answers to this type of question, the program might be a small task or a major programming effort. No program should be attempted with so skimpy a definition. A better statement of program function might be: "This program was written to read the system log files and to produce a summary file. The summary file will be one record for every job or session run and contain the total number or records transferred to each I/O device by that job." With this definition it is easier to tell just what the program will or will not do.

The program should adhere rigorously to its stated objectives. This is a means of avoiding the program that starts out to do a simple task and ends up growing in to a monsterous ´klugde´ that attempts more than its modest original framework can support. The temptation is great to take a program that does ´almost exactly´ what you want and add to it until it can perform both the old and the new function. The problem with this is that the ground rules for the program are being changed. For example, a program that was designed to read the system log files and print out a summary of powerfails might be a candidate to be built into one that can also produce a summary of I/O errors by device. Then it might be modified to create job summaries and print histograms of system usage (CPU or CONNECT TIME). This sounds like a viable thing to do since the original program already has the code necessary to read the log files and extract some of the information. It seems that a lot of time might be saved by avoiding the duplication of that code. I do hereby put it to you that you should avoid this pitfall like the plague! "Why"? you might ask. The problem arises in the fact that at the time the original program was written certain decisions had to be made as to the best way to handle the task. These decisions were made based on the original design objectives of the program. By allowing the design objectives of the program to change after the fact, you may have locked yourself into some no longer valid decisions. For instance, when the original decision as to how to handle reading the logfiles (whether to read record at a time utilizing the file system or to invest the time necessary to write a special internal deblocker) was decided, the size of the task at hand could not justify the time spend in writing an internal deblocker vs the time lost in reading the few powerfail records. Thus the slower file system was used to read the records. In the later version of the program it would make a great deal of sense to do internal deblocking of the log records since the volume of records read was now very great. At this point, the program is already locked into the record at a time scheme of reading the log files and could not be easily converted into a new scheme. Thus, by revising the design objectives of a program you might have ended up with a program that has a much poorer performance than desired, or you will end up spending far more time to convert the old program to a more efficient scheme than if you had just started from scratch.

A further effect of allowing the design objectives to change after the program is written is that the program will end up looking 'LUMPY'. By lumpy, I mean that you will be able to see that the original program shell is here, and a 'LUMP' has been added here for this function, another one here, one there and so forth. In the end the program may be so lumpy and consist of so many different internal techniques, variable names, that it will be almost impossible to find and fix any bugs that it has. Indeed if allowed to continue, the program modifications will eventually create a program that is non functional and non maintainable, in short, totally useless.

If there is any question as to how much the initial program definition statement should encompass, I suggest the following guideline. Make the original program definition cover as large an area as the program will ever be allowed to perform. You may add qualifiers to the effect that it is envisioned that the program will perform these functions at a later time and that they are not completed at the present time. You may then plan the program such that these non-implemented functions will be possible, even to the point of including dummy program blocks to mark the places they will be added. In this way the initial decisions on the methods used by the program will be valid even if the program is expanded to its maximum.

The program definition statement serves two functions: First, it informs the person reading the program source as to just what to expect this program to do (and conversly what to expect it not to do). Second, it serves as a guide as to what modifications fall into the original concept of the program, and thus are vaild changes to it, and what functions are outside the original concept and should best be handled in another manner. The program definition statement should always reflect the current status of the program but should be changed only after a great deal of thought and soul searching.

## MAJOR PROGRAM BLOCKS

The structure of a program should be broken down into major blocks, each block responsible for a certain function. A function might be best described as that section of the program which performs a logically related set of tasks. An example might be for a program that is designed to read the system log files and then either print a summary of powerfailures or write a summary of the I/O written for each job or session run on the system. A possible division into functions might include the following functional blocks:

Initialization and selection of the desired function.

Extraction of the powerfail records from the log files and the accumulation of summaries.

Printing the powerfail summary.

Extracting the I/O records from the log files and the accumulation of summaries.

Printing the I/O summary.

Handling any internal errors or file errors encountered.

Closing the files and finishing up.

Notice that not all functions need be performed for each execution of the program.

At the beginning of each block should be comments that describe exactly what function that block is to perform. This is similiar to the program definition statement at the beginning of the program. The same rules apply to the block definition statement as applied to the program definition. That means that the block definition statement should rigorously define the function of that block. No deviations to the block′s function should be allowed without verifing that the statement will still be valid. Also at this point any interconnections, common files, etc. between this block and any other block should be stated.

This structuring by functional blocks will make the program easier to modify and maintain. Thus if the program is having trouble in the selection of a function, only the block containing that function need be examined. If you desire to change the format of the powerfail summary report, you again need to modify only one block. A necessary feature of the program blocks then is that they must be relatively independent. This will allow you to make a change in one block without affecting the function of the other blocks.

In certain cases it is necessary that two or more blocks will have to have a certain amount of interconnection. In the above example, for instance, the block that reads the logfile records and accumulates summaries must pass those summaries to the block that prints them. Also, if any block encounters an error that is being handled by the errors block then certain information about that error must be passed. In this case the functional blocks can not be 100% independent. In order to maintain the desired degree of maintainability to the program all that need be done is to ridgidly define the interface between the blocks. This might take the form of describing the parameters passed between subroutines or the layout of the program common areas. It becomes important to also include a description of just what values each block can modify in these communications areas. Careful attention to detail in defining the use of these interconnection areas will save untold problems later in the program life. I strongly suggest that this documentation take the form of comments within that actual source code rather than as a separate document whenever possible. This will insure that it is always carried with the program and not misplaced or lost. Once the usage of the interconnection areas has been defined and the program written, it should be strongly discouraged that any programmer be allowed to redefine them without carefully examining the entire program for consequences. Any such redefinitions should also be noted alongside, not in place of, the original definitions. In this way any problems of interconnection areas being wrong can easily be traced to the offending block.

## PROGRAM SUB-BLOCKS

Within a program block the program should be broken up into sub-blocks. Each sub-block should be a single entity that performs only one operation. The size of the sub-block should be such that it can be easily comprehended without a great deal of effort. The ideal situation is where a sub-block can be quickly scanned and verified that the operation is actually performed correctly. From practical experience this relates to from one half to one page of source code.

The sub-block should begin with a comment that states the operation to be performed. Again, this sub-block definition statement is important in that is serves as the guiding rule for the sub-block. The function of the block should be reflected by simply reading the sub-block definition statements.

## THE RULES OF BLOCKS

The block or sub-block is a special animal. In order to be useful several rules must be rigidly followed. Failure to do so may result in a program that only has the illusion of being properly structured. The rules are:

1). The only place for entry into the block's code is at its beginning.

2). The only place for exit from a block's code is at its ending. (The only exception should be an error exit ESCAPE)

3). Only certain easily recoginzable programming constructs will be allowed within a block. This list includes but is not limited to:
   a). straight line code (statement follows statement)
   b). if _ then _ else
   c). looping (DO UNTIL, DO WHILE, WHILE DO, REPEAT n TIMES, DO FOREVER, etc.).
   d). case (execute exactly one of the following)
   e). escape (escape a loop prematurely or error exit)

4). Certain programming constructs are to avoided whenever possible. These include but are not limited to:

   a). Ill defined or ambiguous branches (FORTRAN'S ASSIGNED GO TO, certain cases of COBOL'S PERFORM, FORTRAN'S COMPUTED GO TO while not as bad as an ASSIGNED GO TO should still be avoided)

   b). Backward branches. The major program flow should be strictly from top to bottom. Any backward branching should always be a part of one of the constructs in rule 3 and should be easily recognizable as such. (For example, FORTRAN will not support the DO FOREVER statement but it can be simulated by a single backward branch (GO TO). In this case comment statements should be used to clearly mark the code as a DO FOREVER.

These rules are not all encompassing nor are they inviolable but they do represent a collection of guidelines that I have found lead to programs that will survive the test of time and maintenance. Not all rules can be implemented in all languages but the concepts should apply to any language where the user has control of the program's flow. In cases where the syntax of the language used does not support a construct, the construct can usually be simulated using other features of the language. Always make sure that it is obvious which construct is being used. I recomment sticking to those constructs listed since that comprise the most commonly understood constructs. One of the objectives of structuring your programs is to make them readable by others.

The reasons for avoiding such constructs as FORTRAN´S ASSIGNED GO TO is that they make it very difficult to interpret the program flow. The destination of an ASSIGNED GO TO is not known until actual program execution time. This makes it very difficult for someone reading your code to determine how your program functions without actually simulating its execution thru exhaustive cases in order to find all possible values of the assigned variable at this point in the program. FORTRAN´S COMPUTED GO TO has some of the same problems in that it is a multidirectional branch but at least the possible targets of the branch are listed in the statement. This statement may be used to construct a CASE statement but it should be clearly marked that this is so. Also note that all CASE statements will eventually return to the same point in order to properly terminate the statement. COBOL´S PERFORM verb can cause much the same confusion as FORTRAN´S ASSIGNED GO TO when it is used to perform different subsets of the same set of paragraphs. This makes it very difficult to determine if execution of the paragraphs will have the desired result. This also violates the rule of always entering a block at the top and exiting at the bottom or it confuses the definition of the blocks.

## PROGRAM DESIGN VERIFICATION

The program definition statement should state just what the program is to accomplish. By reading the block definition statements it should be possible to see just what blocks should be involved in any subset of the programs operation. It should be possible, then, to verify if the blocks within the program are capable or satisfying the program definition statement. You should also be able to single out any block that is not necessary to the programs operations.

Now that the program definition can be seen to be satisfied by the block definitions it is necessary to determine if the blocks actually do what their definitions say they do. This is accomplished by reviewing the sub-block definitions. You should be able to follow the blocks structure thru each of the sub-blocks, remembering that all entry into the block is at its top and all exit from its bottom, thus all branching must be between blocks. If the rules of blocks have been followed it should be a simple matter to follow the program thru each sub-block, accepting the sub-block definition as describing properly the action of that block.

Once each block is verified to perform properly, given that its sub-blocks perform properly, all that is necessary is to verify that each sub-block will satisfy its definition statement. This is the first time that we must actually read the source code in the task of determining a programs correctness. If the code within the sub-blocks follow the rules of blocks and is of sufficiently small size then it should be a simple matter to verify that each one properly performs its function.

At this point the program will have a very good chance of performing the program definition as stated at the beginning of the program. If there are any problems then it should be easier to isolate them by placing debugging statements at first the interfaces between the blocks, then at the interfaces between the sub-blocks and finally within the sub-blocks if necessary. In this manner, you can eliminate the majority of code, simply by eliminating the functions and operations that are not in error. If the program has been properly structured then isolating a problem can be the easiest rather than the hardest part of debugging.

By the same token, program modification has also become much easier. In order to change a programs operation follow the following steps:

1). Examine the desired change to determine if it fits within the program definition statement. If not then seriously consider not making the change but rather writing a new program as any such change will have major ramifications.

2). Determine which block performs the function that needs to be changed. Also consider at this time if the change is in reality a new function, in which case it should have its own block.

3). Locate the sub-block within the block that is performing the operation to be modified or determine where in the blocks program flow a new sub-block should be located.

4). Make sure that the changes will not alter the block definition statement. If so then the entire program structure will have to be examined. If not, then you will only be concerned with this block.

5). If the changes are within a sub-block, make sure that the sub-block definition is still valid after the changes.

6). If you have altered the program flow within the block then make sure that the sub-blocks and program flow will still satisfy the block definition.

7). At this point, if the definition statements are still valid, the program flow is still good, and the rules governing the interconnecting areas have not been violated, the modification should be properly installed. Now go back and exercise the program thoroughly in and around the affected functions to verify proper operation and to verify that the modification functions properly.

## COMMENTS AND DOCUMENTATION

A few words need to be said about commenting and documenting a program. In the past this task was considered as separate from writing the program. Indeed, in many cases, the person writing the documentation was not the one that wrote the program. There are times when this approach is not bad, especially when the documentation in question takes the form of an extensive users manual. It would be a misapplication of talent to have a good programmer tied down for six months after each program, writing a book on how to turn on the system etc. The thing to bear in mind in the case of documentation is that there may be various levels of it. Some levels of documentation are best handled by full time documentors as in the case above, but there is a level of documentation that is the responsibility of the programmer and should always be done by them. This documentation consists of the comments within the program source, and a basic functional description of the program. The main use of this documentation will be by those persons that will have to maintain and modify the program. This means that the task of documenting should not have to be an exhaustive effort as the persons using it will at least be experienced programmers. All that is necessary is to explain the basic flow of the program and the major decisions about the program's design.

In many cases, documentation by the programmer has been put aside until the end of the program developement cycle. This was often justified by citing tight schedules, uncertainty in the programs final state etc. Documenting after the fact leads to skimpy or inaccurate documentation at best and possibly even to no documentation if the schedules are in reality tight.

It is imperitive that the programs internal documentation be written as the program is written. This will insure that it truly reflects the actual state of the program. In attempting to go back after a program is in use and document it, the reasons a certain technique was used is often forgotten. The reasons for choosing one technique over another is one of the most important things to know if you are contemplating a modification to that technique. It would take little actual time for the programmer to add a few comments at the top of a block or sub-block explaining how the block functions and either why the technique was chosen or a brief notice as to under what circumstances it would not be appropriate. This accomplishes a dual function: It informs the next programmer, or indeed yourself if you come back to this program at a later date, as to when to consider modifying the technique. It also forces you to consider such questions as applicability at the time you are writing the program rather after a technique is already locked into the program. It is far better to discover that a technique will not work in all cases before it is installed rather that having to remove and replace it after a program is finished.

The time to write the internal program documentation, then, is while the program is being written. The place to put the it is as comments in the program source code. This makes sure that whoever is responsible for maintaining the program will always have accurate documentation at hand. At first it might seem that by documenting as the program is written will take too much time. Upon closer examination and by actual trials it can be determined that just the opposite is true. The exercise of documenting as you go will force you into thinking out just what you are doing and consequently keep you from following too many incorrect paths. The end result will be that in the time that you could have written but not documented a program, you can have written and documented a program that has a much better chance of being correct. At the very least the program will be much more maintainable.

What constitutes good commenting within a program? Contrary to the beliefs of a lot of programmers, the more comments does not imply the better the documentation. It is the quality of the comments not the quantity that determines the quality of documentation. In fact, good documentation can be ruined by adding a lot of unnecessary comments and making it difficult to weed out the essential information. I offer to you the following model. It has proven to be useful to me in the programs I write.

1). Variable and Program Names:

Variables, subroutines, and programs should be named in such a way as to make their use as obvious as possible. Thus it might be easier to relate to CUSTOMER(INDEX) in a program rather than to C(I). I realize that certain languages place restrictions on names. These restrictions are being lifted as the languages grow, for instance HP-3000 fortran now allows names to be fifteen characters long rather than the old five character maximum. In any case try to avoid the naming of items within a program by arbitrary or cryptic names. The best way I know to make a program almost unmaintainable is to go thru it and replace all the variable names with a sequence of meaningless names. The names used become an important part of your documentation. If they are chosen properly then you should have little commenting to do within your sub-blocks. By the

[illegible]

serves as more or less a roadmap to find the block responsible for the function you are interested in. At the very least you should include the basic program flow thru the blocks, a brief description of each blocks function and how to locate it in the source code (subroutine name, etc.).

8). Program Modification History:

A short running history of the modifications to the program should be included next. It's this history that will tell you what changes have been made for each revision of the program. Items to be included are:

- revision code after the changes
- date the changes were completed
- a brief description of the changes
- the name of the person doing the modifications if other than the original author.

9). Author and Modifier Names:

A short description of the author(s) and of those persons responsible for making modifications to the program should be listed. This might include the address and/or phone number of the persons or any other information necessary to identify them.

10). Data Definition Sections:

The usage of any data arrays or block interconnection areas should be set off and commented in such a way as to make their intended use clear. This might be as simple as identifying which array is used to buffer data into and out of the program or as complex as to describe in detail the format of an internal communications array. Comment blocking should be used in order to separate the data definition areas for the various operations into blocks, much as the program code areas are blocked. It is easier to understand the data structures in a program if you can concentrate on only the desired subset of them. For example, if the format of the output records needs to be modified then it should be a simple matter to locate the definitions dealing with it and to be assured that all definitions in that area are strictly for that purpose. This will allow changes to be made without affecting other areas of the program and allow no longer needed variables to be discarded.

11). Block Definition Statements:

Each block within a program should begin with a comment describing the operation to be performed within that block. The format of the statement and of its data definitions should be similiar to those for the main program.

12). Sub-block Definition Statements:

Each sub-block should begin with a very short statement of what that sub-block is to perform. For example "This section sorts the input array into ascending order by customer name". Very little additional commenting should need to be done within the sub-block except maybe for a simple clarification comment here and there. If the variable names are properly chosen and the proper code constructs are being used then the sub-block can be almost self documenting.

If a program is internally documented well then most program maintenance and modification can be accomplished with little more than a source listing. If an additional document is desired it can often be created by excerpting the comments directly from the program, block and sub-block headers.

A word to the modifiers: Once the program has been written following all these guidelines, it is your responsibility to insure that by modifying it you don´t invalidate the original program concepts or documentation. A good rule of thumb should be to try to make any modifications such that, in the future, you couldn´t tell your code from the original unless it is so marked. This means that you should not force your own programming peculiarities into the program unless they match those already in use. Use the same variable naming scheme as the original author, even if you can think of a better one. This prevents confusion later with having to follow several conventions within the same program. Also try to make your code look like the original. This might mean doing the same indenting and following the same commenting scheme. You should also be responsible for updating any internal documentation that is affected by your changes. This is a good exercise in that it will force you to examine all the areas that your changes might affect. Also don´t forget to add a line to the program history records to indicate the changes you have made.

and then resume execution. Again, this is a relatively fast operation (maybe 40 microseconds). If the required segment is not in memory then MPE will suspend the program and make a request to the memory manager in order to have it made present. This process involves reading the segment from disc into memory and might involve swapping some other segment out of memory to make room for it. Disc transfer time on the HP-3000 is fast but is still orders of magnitude slower than direct memory access. Swapping a segment from the disc into memory might take approximately 500 to 2000 microseconds. If this operation is required infrequently then it is not a great burden but if it must occur a great number of times then it can cause a local thrashing condition for this program.

How do you control local thrashing? The main method is to reduce the number of intersegment subroutine calls to the minimum. Lets take the program example given above. In the case of the function involving subroutine A, there is one intersegment call at the beginning of the function and then one at the end in order to return to the main program. This fits nicely in our guidelines. In the second function, however, subroutine B calls subroutine C a great many times. If these two subroutines are in different segments then we might end up in the local thrashing situation. It makes sense, then, to put subroutine B and C into the same segment. That segment will be a little larger than if they were in different segments but the time necessary to call back and forth has been greatly reduced.

Lets summarize what we've learned so far. It is good to reduce the size of our code segments by making a greater number of smaller segments. this reduces the total real memory requirements for the system. If we place two routines that call each other a lot into separate segments then we can cause a local thrashing that wastes time and resources. This means that we have some tradeoffs to evaluate in the segmentation of programs.

Here are some time tested guidelines for the segmentation of program code:

1). Try to make the program segments as small as possible. (4000 words is a good size to shoot for).

2). Minimize intersegment calls between routines even at the expense of larger segments. (Don't worry about numbers as small as 10-20 calls per function but concentrate on those that might be called hundreds of times).

3). Once you enter a segment, try to remain in that segment for as long as possible. This might mean making a copy of a small subroutine that is used by several segments and adding it (under a slightly different name) to each segment that requires heavy use of it.

4). Once you leave a segment, try to remain out of it for as long as possible.

5). Place large and seldom used sections of code into their own routines and their own segments. Error handling routines that contain a large number of error messages are prime candidates (Ascii strings occupy a great deal of code space as compared to machine code). Initialization and light user interaction sections that require messages to be written and read are also good candidates. It is possible that these large sections of necessary code might only be called once, or in the case of error routines, not at all in the program execution. This means thet they will not often need to be present in real memory.

How is the best way to segment our sample program ? First of all, we have determined that subroutine A can be a separate segment. Subroutines B and C should be in the same segment since they call each other a lot. Subroutine D is the error handling segment so it can be in its own segment. The main program is rarely executed and could be in its own segment. The final segmentation would then be:

| | |
|---|---|
| SEGMENT 1: | MAIN |
| SEGMENT 2: | SUB A |
| SEGMENT 3: | SUB B and SUB C |
| SEGMENT 4: | SUB D |

## Data segments

MPE currently has the restriction that a programs modifiable area must be in a data segment called a ´stack´. It does not currently allow the data stack to be split into separate data segments. This means that the only control we have over data stacks is in their size. A certain amount of data stack will be required for the program to execute. The areas we can control have to do with data arrays and storage and with dynamic storage.

The storage necessary for data arrays declared in the main program or in the golbal or common areas of a program is always allocated in the data stack. Anything we can do to reduce the size of this storage area will make for a

[illegible]

In summary, the main point I am trying to get across is that the days are gone when programming could be considered a strictly solitary project. A program may be written by only one person, but if it is to survive, it must be written in such a way as to allow others to maintain and modify it. This means that a programmer must give some thought to the basic structure of the program and to organizing its internals in such a way as to make its operation obvious to other programmers.

I have outlined several techniques that might help to achieve this objective. There are probably other techniques that would also be useful. You might, for instance, make it a practice to have another programmer try to ´read´ your programs in order to determine if they are properly commented and written. This would accomplish yet another benefit in that it will be a method to share programming techniques and expertise. By having an experienced programmer read the novice´s program, he could help the novice to develope the proper standards and techniques. The experienced programmer might also learn a few new techniques from the novice if he keeps his eyes open. By having the novice read the expert´s programs, he could learn what techniques should be used in a given situation. The expert will be able to sharpen his documentation techniques by having the novice identify any areas of the program are difficult to follow.

A major point of this whole procedure is to combat the feeling that a program is the exclusive property of one programmer. It should be thought of as belonging to the entire group or to the company. In this way the programmers will not code cryptic programs in order to promote their own mystique. They will instead be writing prorams that will be understandable by others and as such can survive long after they are gone. A side benefit of writing programs that survive should be the writing of more reliable code and raising the standard of data processing in general.

# SOFTWARE QUALITY CONTROL

C. EDWARD McVANEY
J. D. EDWARDS & COMPANY

## The Predicament

Computers have brought space age technology to the ordinary businessman. Their potential seems boundless; the electronic equipment (hardware) is usually superlative; but the computer programs (software) can bring a plague to your organization.

The hardware side of the computer industry seems to be far more advanced than its software counterpart. The gripes and groans about computer failures are not all without substance, and ninety-nine times out of a hundred it seems it is the software that is at fault.

You have to have software! The computer won't work without it! But what do you do? How do you protect yourself? How do you assure a high quality result?

Knowing what to look for, and how to judge quality software, need not be a great mystery. You don't have to be a computer technician to make meaningful value judgments. Your common sense will serve you well if you're willing to do some research.

Before we go much further, we should make a clear distinction between "systems" and "applications" software. Applications software is the term computer specialists use to identify major business systems on a computer. For example:

. Payroll Systems
. Inventory Systems
. Billing Systems, and
. Accounting Systems

Applications software is sometimes referred to as a software system, a software package or just a system. In addition to the individual computer programs (usually ranging from 10 to 100 programs), it would include all the supporting documentation and related professional services including training and on-going maintenance.

Systems software, on the other hand, is a term used to describe the family of computer programs that are used by the computer to do its own work. For example:

. Operating System
. Compilers
. Utility Programs, and
. Communication Monitors

[illegible]

2. Systems Documentation - Documentation is to a computer system what a shop manual is to a mechanic - it is not very important until you need it. Systems documentation should consist of:

. Samples of Each Report
. Illustrations of each Video Display
. Copies of Forms and Documents
. A Glossary of Terms and Codes
. Clerical Procedures
. Computer Operating Instructions
. Program Descriptions
. Data Base Descriptions, and
. Other Appropriate Material

If these items are not readily available and professionally packaged so that they are easy to understand, you've noted a serious weakness in the system. You will be vulnerable to turnover of personnel and empire building if you don't have good documentation.

3. Clerical Procedures - Computer specialists have been known to forget about the people who have to use the system. Check the clerical operating procedures including the video terminal operating instructions to see if they are simple and effective. There's no need to explain the mundane, but at the very least, the unusual aspects of the system should be explained.

Of course, checking for voids goes beyond the particulars we've noted here - video displays, operating reports, input journals, forms, data bank, accounting controls and so on.

## (b) Fitness to Your Needs

In their eagerness to sell computer hardware, salesmen have been known to recommend a software system that didn't quite fit. Major misfits are not uncommon.

If you are told that your organization can be changed or adapted to fit the software, make sure you know what will have to be changed. A good software package should provide definable improvements to the flow of work within your business with a minimum of organizational or procedural changes. It had better, because modification costs dearly, not only in terms of time and money but, more importantly, because vital management data often will not be available while the modification is being done. The "data gap" can be filled, of course, by keeping a manual or semi-automated system running parallel with the computer, but we can't imagine any business owner wanting to do that for very long.

(c) Maintenance Costs

Good software packages, like good automobiles, don't require much maintenance. If a computer programmer has to devote a significant part of his time to maintaining a software system, you can bet it is poorly designed or poorly programmed. Conversely, if little or no maintenance is required, you should be encouraged.

The maintenance cost of an acquired software package can often be determined by consulting with other users whose names the vendor has given you as references.

(d) Feature Comparisons

Another effective technique for judging software quality is feature comparisons between comparable systems. This is basic consumerism. These features include:

. Processing Capacity
. Hardware Requirements
. Management Accounting Techniques
. Data Base Concepts
. Timeliness of Reports
. Video Screen Features
. Audit Controls, and
. Others, as appropriate

Many of the little "extras" in a software system are well worth having.

(e) Operating Costs

If a software package does not reduce operating costs (clerical salaries, etc.) or provide significant intangible benefits such as improved management information or better customer service, then something is wrong.

Innumberable software systems have been installed with the anticipation of reduced operating cost and significant intangible benefits only to have just the opposite come true.

Therefore, you should check the "track record" or references of a software system to see if it has performed as promised. While you're checking references, ask about programmer maintenance costs.

(f) Real Response

Video terminal computer systems should provide for almost instantaneous entry of data, updating of records and redisplay of updated records. Many software packages fall far short of this capability. It is not uncommon to find video terminal packages that do nothing more than conventional "keypunching" into the video terminals and then process the data in the off hours. If this is the case, computer records might not be updated until the next morning even though video terminals were used for data entry.

(g) Technical Support

Behind every good computer system there should be good people to provide the technical support that, from time to time, you may need to rely on.

Do these people know and understand your business? Are they competent enough to get things done? What is their track record? Where will they be a year from now when you need them? Is this a one man show or a professional team of computer specialists? You should answer these questions and assure yourself of proper support.

Technical support personnel are all important to the success of your system.

(h) Conventional Technology

More often than you might expect, computer systems are developed using non-standard technology, i.e. offbeat programming languages, homemade systems software, and mix 'n' match equipment. Beware of these approaches because they invariably create inflexibility, overdependence on support personnel, and early obsolesence.

(i) Data Controls

Precise control features are essential to the success of any software package. Check to see that edit controls are installed to check the accuracy, reasonableness and interrelationship of all input transactions. Also check to see that input balancing controls are in effect. The system should have provision for a complete audit trail of all sensitive transactions. This audit trail should balance to master file control totals and reference back to the originating entries.

(j) Predefined Standards

A very effective way to evaluate the quality of a software system is to compare it to independent standards of excellence.

J. D. Edwards & Company has developed such standards, establishing criteria for, among other items:

. Systems Documentation
. Input Controls
. Accounting Controls
. Data Base Design
. Computer Processing
. Report Design
. Programming Structure
. Programming Efficiency, and
. Numerous other items.

## Some Classic Examples

We have said that you don't have to be a computer expert to judge the quality of a software package. Let's illustrate how you may judge software quality with simple examples of a computer report and an input document.

### (a) Report Quality

Illustration Number 1 represents a construction loan status report for Mr. Charles Louis who is building a new home. This report is a classic example of computerized goobledygook. Note the following deficiencies which could be readily detected by a non-technician.

1. The name and address were typed rather than printed by the computer.

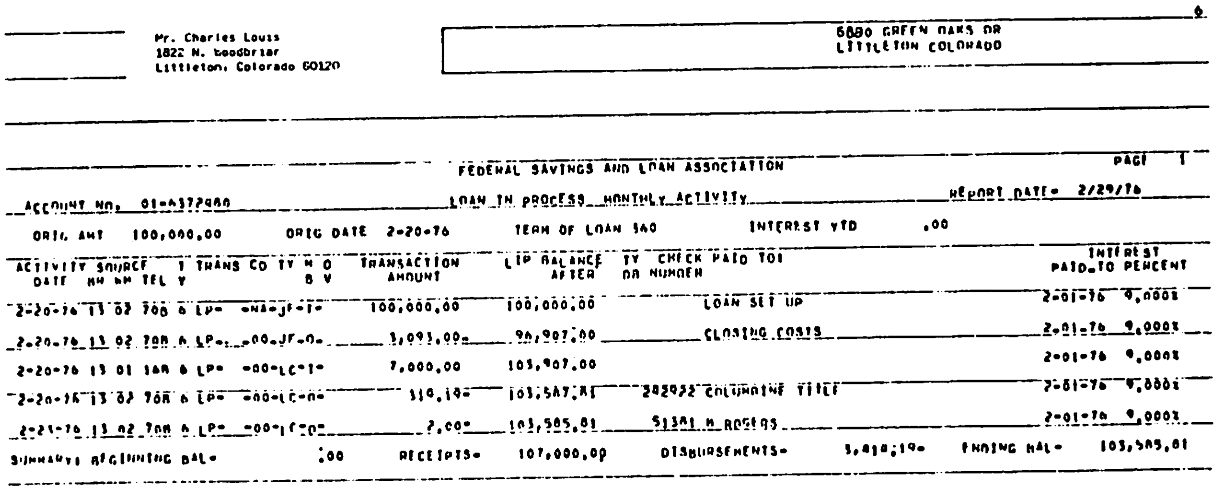
2. The following columnar headings are unintelligible:

| | |
|---|---|
| . BR | . TY |
| . WM | . MB |
| . TEL | . OV |
| . TY | . TY DB |
| . CD | |

3. Note that there are three columns labeled TY.

4. Even if you understood what the codes mean, how would you understand the data? What, for example, does "I" mean in the MB column?

Illustration No. 1



Mr. Charles Louis
1822 N. Woodbriar
Littleton, Colorado 60120

```
                                                              6080 GREEN OAKS DR
                                                              LITTLETON COLORADO

                          FEDERAL SAVINGS AND LOAN ASSOCIATION                              PAGE    1
 ACCOUNT NO.  01-6372980            LOAN IN PROCESS  MONTHLY ACTIVITY            REPORT DATE-  2/29/76
  ORIG. AMT   100,000.00      ORIG DATE  2-20-76      TERM OF LOAN 360      INTEREST YTD      .00
 ACTIVITY SOURCE    T TRANS CO TY M O   TRANSACTION    LIP BALANCE  TY  CHECK PAID TO:                INTEREST
  DATE   BR WM TEL  Y            B V      AMOUNT          AFTER     DB NUMBER                      PAID-TO PERCENT
 2-20-76 13 02 700 6 LP-  -MA-JF-I-   100,000.00     100,000.00             LOAN SET UP             2-01-76  9.000%
 2-20-76 13 02 708 6 LP-  -00-JF-0-     3,093.00-     96,907.00             CLOSING COSTS           2-01-76  9.000%
 2-20-76 13 01 168 6 LP-  -00-LC-I-     7,000.00     103,907.00                                     2-01-76  9.000%
 2-20-76 13 02 708 6 LP-  -00-LC-0-       319.19-    103,587.81    282922 COLUMBINE TITLE           2-01-76  9.000%
 2-21-76 13 02 708 6 LP-  -00-LC-0-         2.00-    103,585.81     51381 M ROGERS                  2-01-76  9.000%
 SUMMARY: BEGINNING BAL-        .00     RECEIPTS-    107,000.00    DISBURSEMENTS-    3,414.19-   ENDING BAL-   103,585.81
```

Mr. Charles Louis was somewhat perplexed. He felt insecure because he didn't understand what the report said. So he took the report to his CPA and asked him to explain it - he was equally baffled.

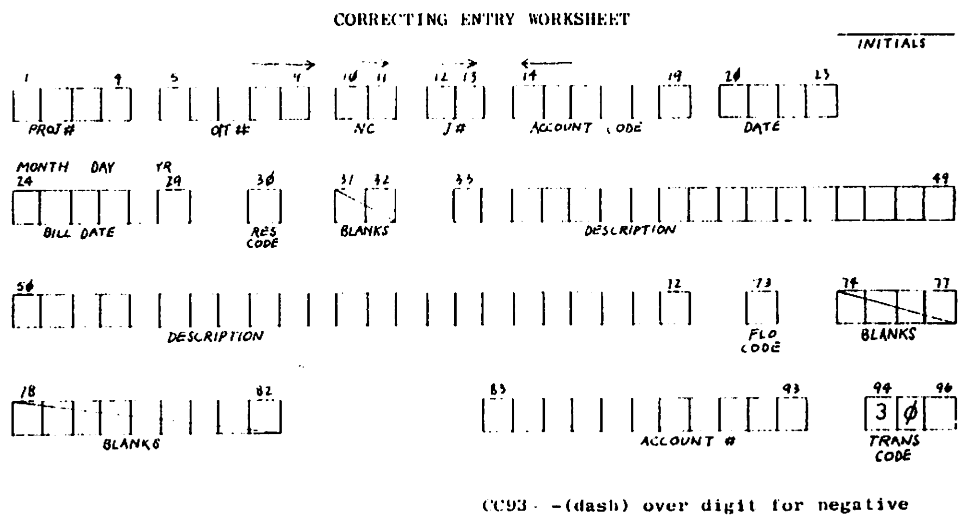
You should not tolerate this kind of poor report design in any software package. In almost all cases, computer reports can be made clear, concise and easy to interpret.

(b) Input Forms Quality

Illustration Number 2 represents a form used to correct a worksheet in a real estate management system. (This form has also reached the status of classic gobbledygook). Note the deficiencies which can be readily detected by a non-technician:

1. What do the arrows mean?
2. What are:
   . OIT#
   . NC
   . J#
   . RES Code
   . Fld Code?
3. What do you enter in the four digit date field?

Illustration No. 2



CORRECTING ENTRY WORKSHEET

INITIALS

1 4 PROJ# 5 9 OIT # 10 11 NC 12 13 J# 14 ACCOUNT CODE 19 20 DATE 23

MONTH DAY YR 24 29 BILL DATE 30 RES CODE 31 32 BLANKS 33 DESCRIPTION 49

50 DESCRIPTION 72 73 FLD CODE 74 77 BLANKS

78 82 BLANKS 83 ACCOUNT # 93 94 3 Ø 96 TRANS CODE

CC93 - -(dash) over digit for negative

The point is that neither computer input nor output forms need be as complex as many computer people make them. Look at Illustration Number 3, for example, the "Account Master Revisions" form reproduced on the next page. All column headings are in plain English - not computerese. Codes are plainly defined at the bottom of the sheet. There is plenty of room to write, and there are no mysterious arrows or slash marks.

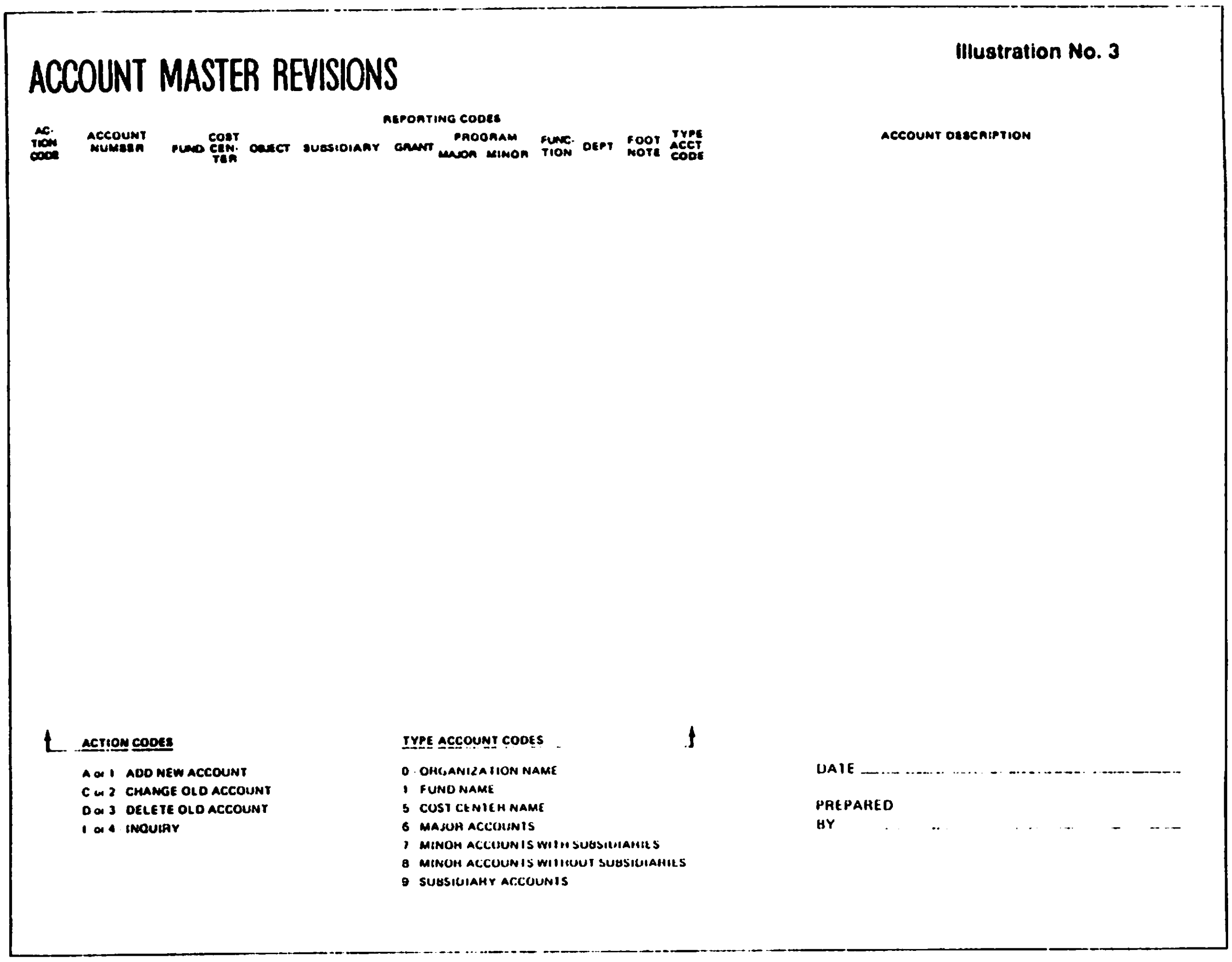


**Illustration No. 3**

# ACCOUNT MASTER REVISIONS

| ACTION CODE | ACCOUNT NUMBER | FUND | COST CENTER | OBJECT | SUBSIDIARY | GRANT | PROGRAM MAJOR | PROGRAM MINOR | FUNCTION | DEPT | FOOT NOTE | TYPE ACCT CODE | ACCOUNT DESCRIPTION |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

(REPORTING CODES: FUND through TYPE ACCT CODE)

**ACTION CODES**

| | |
|---|---|
| A or 1 | ADD NEW ACCOUNT |
| C or 2 | CHANGE OLD ACCOUNT |
| D or 3 | DELETE OLD ACCOUNT |
| I or 4 | INQUIRY |

**TYPE ACCOUNT CODES**

| | |
|---|---|
| 0 | ORGANIZATION NAME |
| 1 | FUND NAME |
| 5 | COST CENTER NAME |
| 6 | MAJOR ACCOUNTS |
| 7 | MINOR ACCOUNTS WITH SUBSIDIARIES |
| 8 | MINOR ACCOUNTS WITHOUT SUBSIDIARIES |
| 9 | SUBSIDIARY ACCOUNTS |

DATE ____________________

PREPARED
BY ____________________

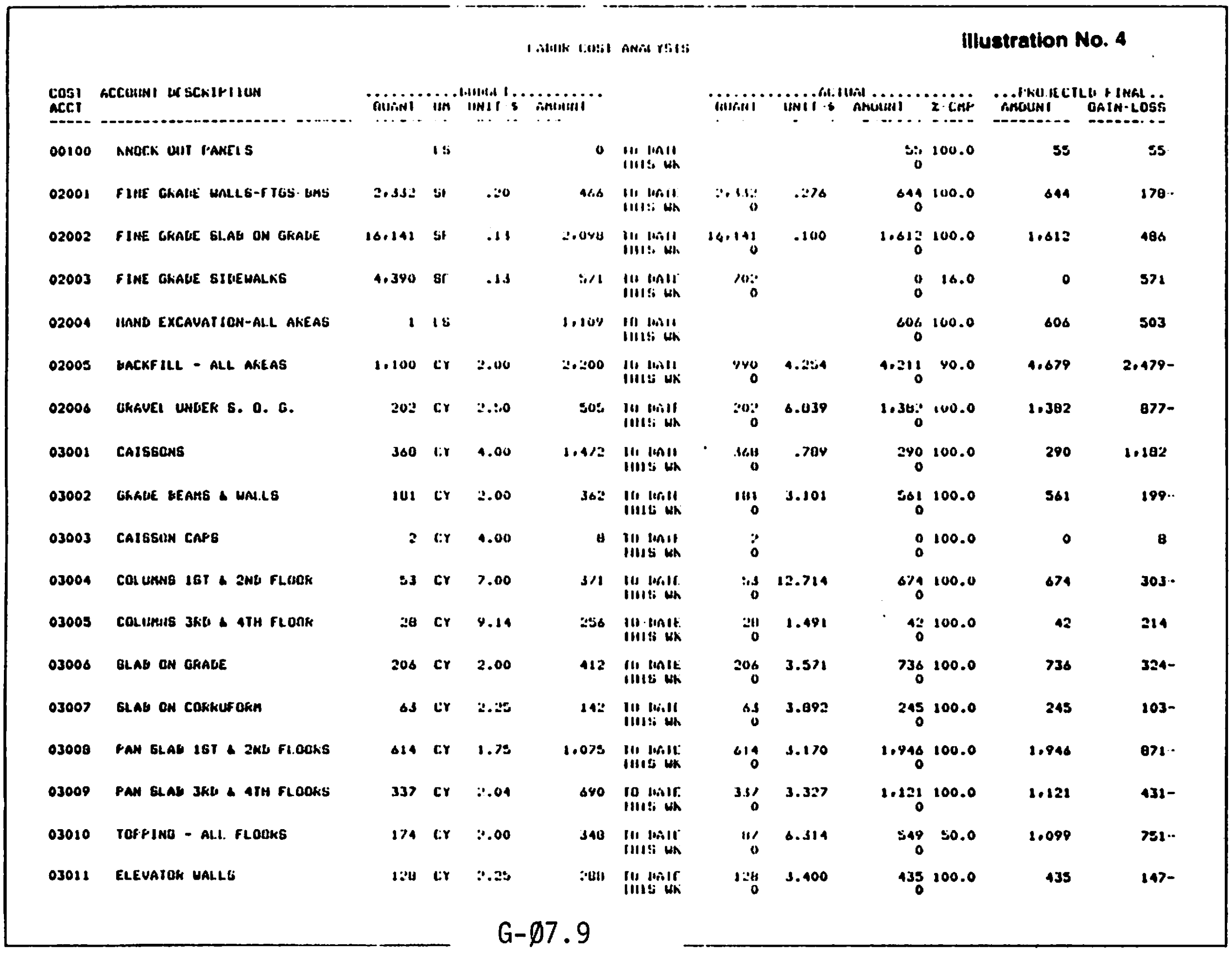


**Illustration No. 4**

```
                                          LABOR COST ANALYSIS

COST  ACCOUNT DESCRIPTION           ............BUDGET............            ..............ACTUAL.............  ...PROJECTED FINAL..
ACCT                                 QUANT  UM  UNIT $   AMOUNT              QUANT    UNIT $   AMOUNT  % CMP    AMOUNT    GAIN-LOSS
----- ----------------------------                                                                             ---------  ---------

00100  KNOCK OUT PANELS                     LS               0  TO DATE                               55 100.0        55        55
                                                                THIS WK                                0
02001  FINE GRADE WALLS-FTGS-BMS     2,332  SF     .20     466  TO DATE    2,332    .276             644 100.0       644       178-
                                                                THIS WK        0                       0
02002  FINE GRADE SLAB ON GRADE     16,141  SF     .13   2,098  TO DATE   16,141    .100           1,612 100.0     1,612       486
                                                                THIS WK        0                       0
02003  FINE GRADE SIDEWALKS          4,390  SF     .13     571  TO DATE      702                       0  16.0         0       571
                                                                THIS WK        0                       0
02004  HAND EXCAVATION-ALL AREAS         1  LS           1,109  TO DATE                              606 100.0       606       503
                                                                THIS WK                                0
02005  BACKFILL - ALL AREAS          1,100  CY    2.00   2,200  TO DATE      990   4.254           4,211  90.0     4,679     2,479-
                                                                THIS WK        0                       0
02006  GRAVEL UNDER S. O. G.           202  CY    2.50     505  TO DATE      202   6.839           1,382 100.0     1,382       877-
                                                                THIS WK        0                       0
03001  CAISSONS                        360  CY    4.00   1,472  TO DATE      368    .789             290 100.0       290     1,182
                                                                THIS WK        0                       0
03002  GRADE BEAMS & WALLS             181  CY    2.00     362  TO DATE      181   3.101             561 100.0       561       199-
                                                                THIS WK        0                       0
03003  CAISSON CAPS                      2  CY    4.00       8  TO DATE        2                       0 100.0         0         8
                                                                THIS WK        0                       0
03004  COLUMNS 1ST & 2ND FLOOR          53  CY    7.00     371  TO DATE       53  12.714             674 100.0       674       303-
                                                                THIS WK        0                       0
03005  COLUMNS 3RD & 4TH FLOOR          28  CY    9.14     256  TO DATE       28   1.491              42 100.0        42       214
                                                                THIS WK        0                       0
03006  SLAB ON GRADE                   206  CY    2.00     412  TO DATE      206   3.571             736 100.0       736       324-
                                                                THIS WK        0                       0
03007  SLAB ON CORRUFORM                63  CY    2.25     142  TO DATE       63   3.892             245 100.0       245       103-
                                                                THIS WK        0                       0
03008  PAN SLAB 1ST & 2ND FLOORS       614  CY    1.75   1,075  TO DATE      614   3.170           1,946 100.0     1,946       871-
                                                                THIS WK        0                       0
03009  PAN SLAB 3RD & 4TH FLOORS       337  CY    2.04     690  TO DATE      337   3.327           1,121 100.0     1,121       431-
                                                                THIS WK        0                       0
03010  TOPPING - ALL FLOORS            174  CY    2.00     348  TO DATE       87   6.314             549  50.0     1,099       751-
                                                                THIS WK        0                       0
03011  ELEVATOR WALLS                  128  CY    2.25     288  TO DATE      128   3.400             435 100.0       435       147-
                                                                THIS WK        0                       0
```

The "Labor Cost Analysis," Illustration Number 4, shows that computer output needn't be gobbledygook either. This weekly report on a construction project can be easily read by a non-expert. It keeps jargon and confusing abbreviations to a minimum.

## The Make versus Buy Decision

In theory, buying a software package should offer the advantages of having:

. Lower start-up costs, and
. Shorter start-up intervals.

In practice, things don't often work out that way. Many experienced data processors are skeptical of these supposed advantages - particularly when modification or changes to the software packages are required. Indeed, the more accomplished computer user often steers clear of purchased software packages all together.

On the other hand, custom designed and programmed software supposedly offers the advantages of being:

. Better suited to your operation needs,
. More flexible,
. Easier to maintain, and
. Less expensive to operate.

But custom designed and programmed software systems are not without peril. If you don't have access to a highly qualified team of systems analysts and computer programmers (either as in-house employees or outside contractors), you shouldn't try to develop a custom system.

A well designed custom system is, in the long run, usually more effective than a purchased system. If your needs are unique in any way, and you can afford to do it right the first time, you should seriously consider custom designed and programmed software packages. After all, you've probably custom fit every other aspect of your business, so why not do the same with you computer software.

## What It Should Cost

If software were cheap, the big, experienced users of computers would have known it long ago. Experienced computer users may invest nearly as much (sometimes even more) in software as hardware.

It would be nice if one or two thousand dollars of software were all you needed, but unless your business fits an available ready-made package very closely, you should plan on spending anywhere from 40% to 100% of the computer purchase price on software, depending on your needs.

Once More From The Beginning

When it's all said and done, what you really want is:

- . Better Management Information
- . Reduced Operating Costs
- . More Flexible Business Systems, and a
- . Decent Return on Your Investment.

Today's computers have the potential to do all of that in a most pleasing way - but be careful!

Software - particularly applications software - can be the fly in the ointment.

- . Check out each component of the system
- . Use common sense and be critical
- . Ask for professional help in technical areas
- . Follow a sound evaluation methodology
- . Look for voids, fitness for your needs, maintenance headaches, effective features and operating economics.
- . Be very careful about technical support personnel, and
- . Use predefined standards or evaluation criteria if you have access to them.

Finally, don't be tempted by low-cost or no-cost software purely because of price - or rule out a custom-design purely because of high start-up cost. But you're not likely to do either if you've been thorough up to the point of cost considerations. Remember that old saying ..........

The bitterness of poor quality remains long after the sweetness of low price is forgotten.

DENNIS DINAN
GLENN ENTIS

MORGAN GUARANTY TRUST COMPANY OF NEW YORK

# AN APPROACH TO ON-LINE APPLICATIONS MANAGEMENT

PRESENTED AT THE
HEWLETT-PACKARD GENERAL SYSTEMS USERS GROUP
7th INTERNATIONAL MEETING

OCTOBER 30 - NOVEMBER 3, 1978    DENVER, COLORADO

## TERMINAL AND APPLICATIONS MANAGEMENT SYSTEM

TAMP (for Terminal and Applications Management System) is an integrated system of processes and subprograms which are used to define and manage an on-line application system. Each application to be run under TAMP includes one TAMPFILE, which is a multi-key, variable-length KSAM file designed to provide a compact, flexible, and easy-to-use interface to the applications environment. This TAMPFILE, which will be discussed below, defines all processing to be done within the monitor, and includes TAMP terminal configuration, user/function security matrix, and all data files to be managed by the system.

The first step in implementing an application under TAMP is the creation of the TAMP KSAM file. Here the user is asked questions about the size of his application, such as the maximum number of functions which he expects to include in the system. If the user miscalculates the dimensions of his application, he also has the option of later expanding his TAMPFILE, without affecting its data. The user also enters the Security Manager's name and password, and in so doing, restricts all further modification of the file to the bearer of that password.

The next stage in preparing a TAMPFILE for use is the definition of the application. This data is entered through a terminal interactive program and is grouped in logical records which represent the resources managed by TAMP. These resources are user functions, system utilities, data files, terminals, and users. A brief description of each of these records, their relationships to each other, and their use in on-line processing may be found below.

Although the following outline of TAMP record types greatly simplifies their structure, it does provide the essential content and purpose of each type of record.

FUNCTION - There is one function record for each callable application entry point. Each function may be one of several types (program subprogram, SL subprogram, independent process, MPE stream) and is invoked accordingly. If the function expects data file numbers or data from the terminal as parameters, that is also included here.

UTILITY - Utility records are almost identical to function records. However, those functions which are designated as utilities are initiated automatically by the TAMP software and may not receive parameters or be called by users.

FILE - This record contains a filename qualified by group and account. The file which this record represents may be of any type, including MPE, KSAM, IMAGE, and SMOG (described elsewhere in this paper). Once defined on a TAMP record, a file may be opened by the TAMP software and passed as a parameter to one or more functions.

TERMINAL - This record contains the logical device number of a terminal to be included in on-line TAMP. For each active terminal TAMP record, a process will be created by the on-line software which opens and allocates the corresponding logical device.

USER - The user's record includes his name, encoded password, processing values, and maps indicating which functions he may display on his menu and which functions he may actually call (these two types of function access are completely independent). Also included in this record are the assignments by which a user can expect each of four terminal soft-keys to call for him a selected function.

The program which defines a TAMPFILE also serves as the TAMPFILE maintenance program, and is where the Security Manager really does his job. As each user (and user password) is added to the system, the Security Manager also enters which functions that user may see on his menu and/or actually call. The Security Manager may also assign for each user two application-defined processing values, which are handy for such things as data-base user class codes, maximum dollar amount which may be entered by the user, or whatever else fits the application.

Other features of the TAMPFILE maintenance program include terminal configuration, assignment of data-files to functions (the filenumbers or file-tables of these files will be passed as parameters to their assigned functions during on-line processing), and definition of system utilities, such as start-up, batch processing or logging routines.

The on-line portion of TAMP is the "outer block" within which all application programs will run. The TAMP software will call application defined start-up routines, open and allocate terminals, display sign-on screens, provide all first-level menus, and call requested application functions. This is accomplished through the use of a set of related processes which communicate with each other through a commonly held extra data segment.

On-line TAMP is actually a simple, two-level process tree consisting of a father process (FATHER'TAMP), and his various kinds of sons. FATHER'TAMP is the process which is created upon running TAMP. His first duties include initiating start-of-day application processing, which may also include restart and recovery procedures, and creating the extra data segment which will be used as the common communication area between processes. FATHER'TAMP next reads all terminal records from the TAMPFILE, and for each creates a terminal sign-on process. FATHER'TAMP also creates a process whose sole responsibility is communication with the system console, which here becomes the TAMP console as well. At this point, FATHER'TAMP suspends himself, and is reactivated only by sons with specific requests. All further processing depends on input from the user and console terminals.

The role of the common communication area in the extra data segment, as well as the orderly regulation of processes through use of Resource Identification Numbers (RINS) is a vital component of the on-line TAMP software. However, this processing is invisible to the user and beyond the scope of this paper. It should just be considered here that processes do not haphazardly suspend, terminate, and activate one another.

The task of the terminal sign-on processes is simple. Each displays a user sign-on screen, and then validates, by user name and password, attempted entries into the system. Upon a successful sign-on by a user, the terminal sign-on process for that user's terminal activates FATHER'TAMP with the appropriate message and then terminates.

Upon notice that a user has signed-on successfully, FATHER'TAMP creates a new process for that user. This process displays a welcome screen for the user, and, depending on the user's response, may display a menu of those functions which the Security Manager has permitted this user to see. The user may now request entry into a function in one of several ways, including selection by menu position, function name, or the user's own soft-key assignments.

The user's process calls the selected function in the manner specified in the function's record (e.g., if the selected function's type indicates that the function is a subprogram in one of the Segmented Libraries, the user's process will find, load, and jump to this subprogram). Also, if the function's record indicates that it expects data files and/or data from the terminal, the user process will open the appropriate files (if not already open) and pass the necessary data as parameters to the function.

When a user signs-off, control of his terminal is passed back to a terminal sign-on process. At the end of the day, FATHER'TAMP again takes control, signs remaining users off of the system, initiates application-defined end-of-day processing, and terminates.

## SMOG

SMOG is a HP2645A terminal interface and screen processing subsystem. It provides a convenient access to local and remote terminal devices using a wide range of the HP2645A terminal capabilities while keeping necessary terminal knowledge to a minimum.

SMOG makes very few demands on the application programmer. Developing screen images requires no knowledge of escape sequence characters. There are no stringent programming requirements necessary for accessing terminals in an interactive mode. All housekeeping, work areas, buffers and data transfers are handled transparently within the subsystem.

Screens exist in the system as logical records in a KSAM file. These records consist of the screen image in displayable form, control data and reporting data. Each record has an associated screen name which is used as the "key" when adding a screen record to a file or recalling an existing screen within a file. Also, there is a control record in the file. This contains global file information and sizing requirements necessary for allocating buffer space for screen records.

Working with screens, it is entirely up to the user how they will be designed. Then a screen file maintenance program assists the user in defining and entering the screen. The user must supply the row and column numbers of the start of each field, their lengths, their field type: protected or unprotected, any display enhancements and edit checking, plus any data which is to be used to initialize the fields. The maintenance program then interprets and assembles this data into a displayable character string with all appropriate escape sequences.

Counters and indexes are kept internally to accumulate read and write counts, field descriptions and status information about a screen.

Once entered, a screen may be easily modified, deleted, displayed, duplicated or renamed. Several reports are available including one containing a reproduction of the screen image. This report also contains a description of each field along with entry numbers which are used to address fields in subsequent screen maintenance or to programmatically access these fields for changing their enhancements, protection, or modifying their data content.

Screens in one file may be copied in entirety or selectively to new screen files. They may also be merged in the same manner with screens in other existing screen files. Provisions are included for both keeping and deleting another version of the same screen found during a merge.

Programmatically, SMOG has many useful features. One call to an initialization program sets up everything necessary to process. It is even smart enough to know how its process was started so that a terminal can be initialized properly. If a user father process was responsible for its creation, it expects that he also assigns a terminal to use, otherwise, it defaults to using the logon terminal. Also at this time, the caller's screen file is opened, the break key is disabled, buffer areas are dynamically allocated and a global work area is set up to maintain file numbers and status information which is necessary for on-line processing. None of this is ever seen by the programmer.

The programmer can now interact with the terminal using the screens from the screen file. Intrinsics are available to locate screens, to modify protected and unprotected data, display enhancements, and protection, to output a screen image, to read a screen's unprotected data, and to dynamically create and read from individual fields on a screen. The outputting of a screen can optionally include the field at which the cursor is to be positioned; all reads from the terminal can be armed with a function key interrupt feature which returns an indicator to the program, or can optionally be timed out. The programmer has the responsibility of calling an intrinsic to reset the terminal before exiting from a program.

SMOG is very versatile. It can alternate the use of both block and character mode of I/O. It is callable from COBOL, FORTRAN, and SPL. In each of these languages, it allows a simple approach to screen processing.

## KEDS

Keyed Extra Data Segments (KEDS) is a method of organizing ordered sets of tables within extra data segments. A set consists of one or more tables. A table is comprised of one or more elements. Access to these tables is permitted by table element within table within set.

This access method allows an application to keep all necessary tables "virtually" memory resident while permitting them to be shared by several processes within the same job or session. It allows for elements to be deposited into a table by a relative element number and retrieved by the same element number or by searching for a table element with a specific key value.

KEDS is relatively simple to use. Initially, table space must be allocated. This is done by an intrinsic call with which the program must supply a set name, the size of the data segments to be used, the number of tables in the set and a list containing each table description. Each table in a set must be assigned a name, an element size, and a maximum number of elements.

Using this information, KEDS calculates the total amount of memory needed and creates as many extra data segments necessary to contain all of the user's tables. A parent segment is also created through which the tables will be accessed. Therefore, by accessing the parent segment, one can determine if an element exists in a table, the segment in which it exists, its location within the segment, plus its length. Once created, the programmer must load table elements in the order in which they are to be used.

Elements can be deposited and retrieved very efficiently by KEDS, using relative numbers, but, this is not always the easiest or best way of accessing tables. If a programmer so desires, KEDS will search a table for an element with a specific key value. Keys are not restricted to any size or location within the element. Tables can exist which have several keys and the choice of which will be used in the search is made by the calling program. Tables can be ordered in ascending or random key sequence.

Data transfers are very flexible. One can specify that all, part or none of an element is to be transferred if it is found during a search. This is a powerful tool for validation tables or tables whose elements are suited for multiple purposes.

The complete capabilities available to the programmer are: create tables, append element, update element, read element, read element by key (random or sequential), and delete tables. Combined, they provide a simplified approach to memory resident table management.

G-Ø9.12

# Morgan Guaranty Trust Company of New York

G-Ø9.13

- Automated Banking System
- Euro-currency Automation
- Global Exposure System
- N.Y. Profitability System

## ENVIRONMENT

- HP 3000 Series II/III
- On-line & Batch
- Multi-user

G-Ø9.14

## REQUIREMENTS

- Self-contained system
- Computer management
- User & function oriented security
- Terminal management
- Full screen support

# TERMINAL APPLICATION MANAGEMENT PROGRAM

G-Ø9.16

G-Ø9.17

# SCREEN MANAGEMENT OFF·LINE GENERATION

KEYED

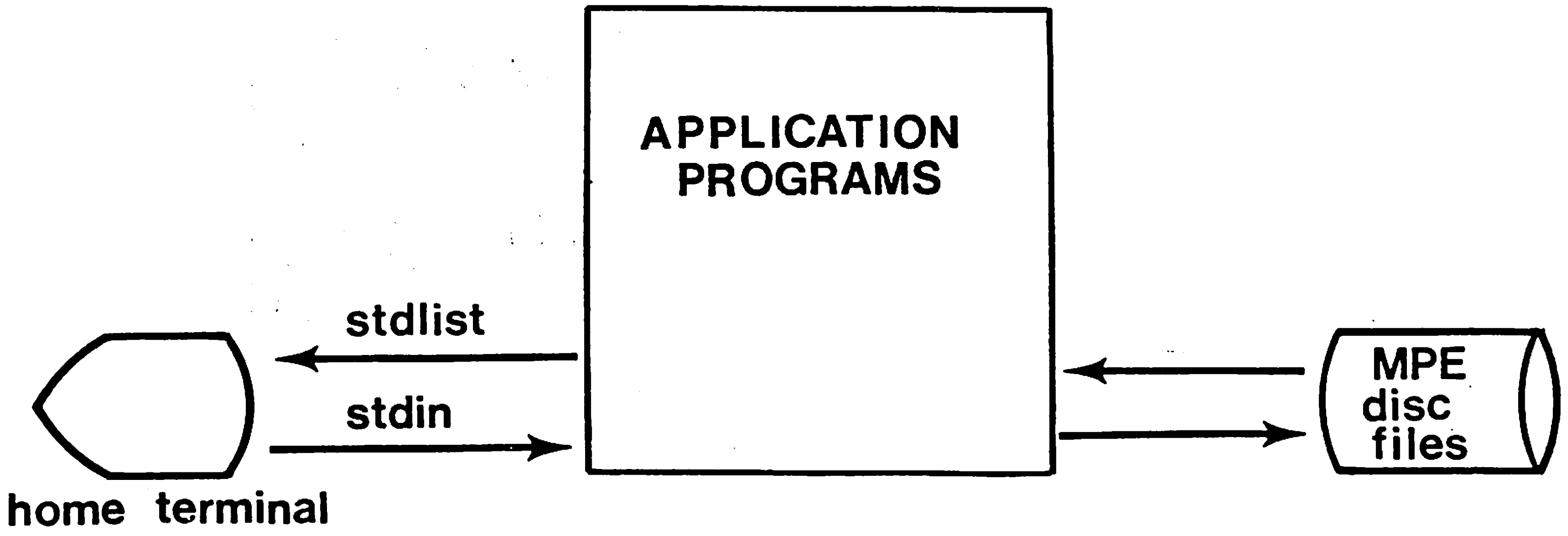
EXTRA

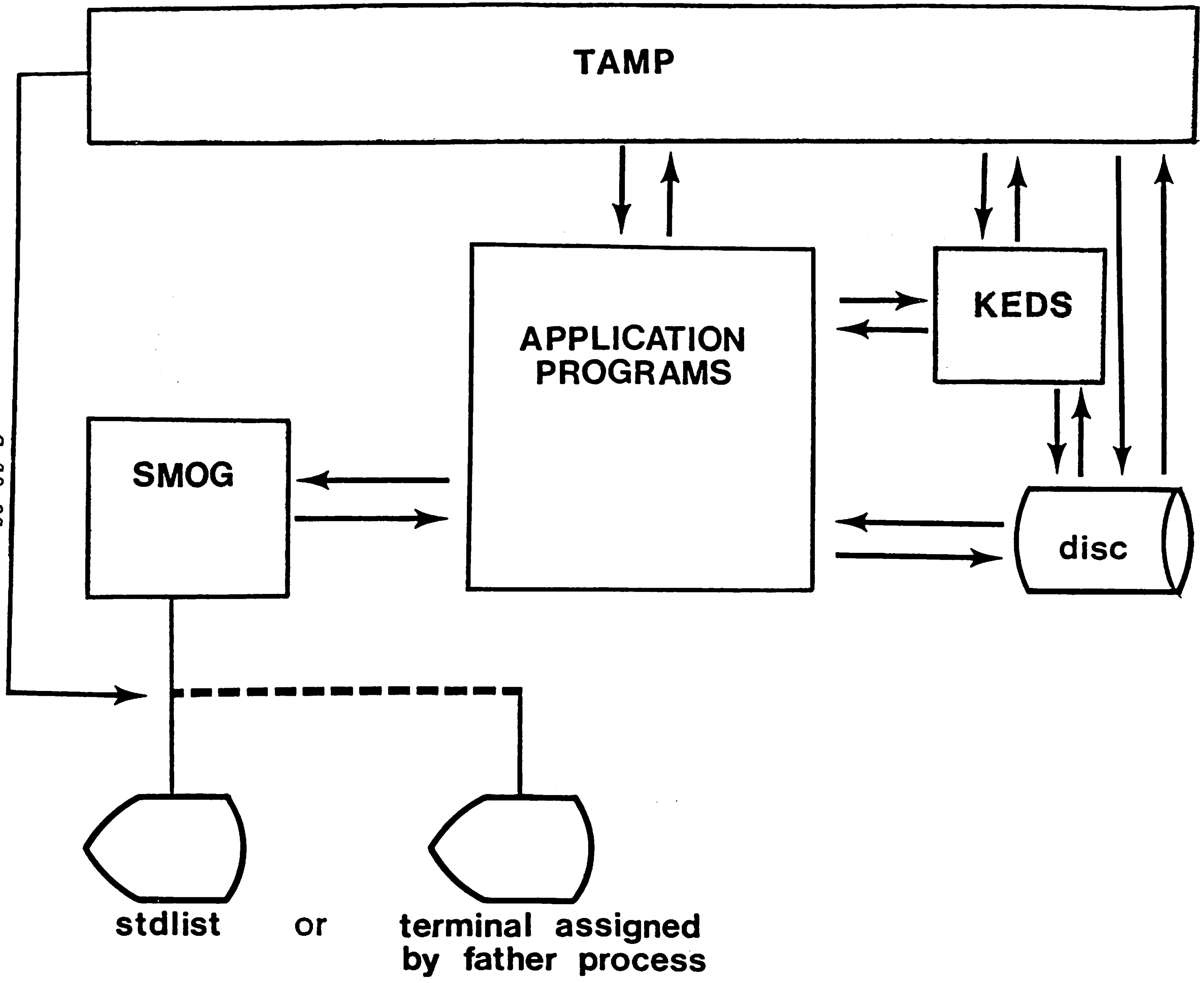
DATA

SEGMENTS

G-Ø9.19

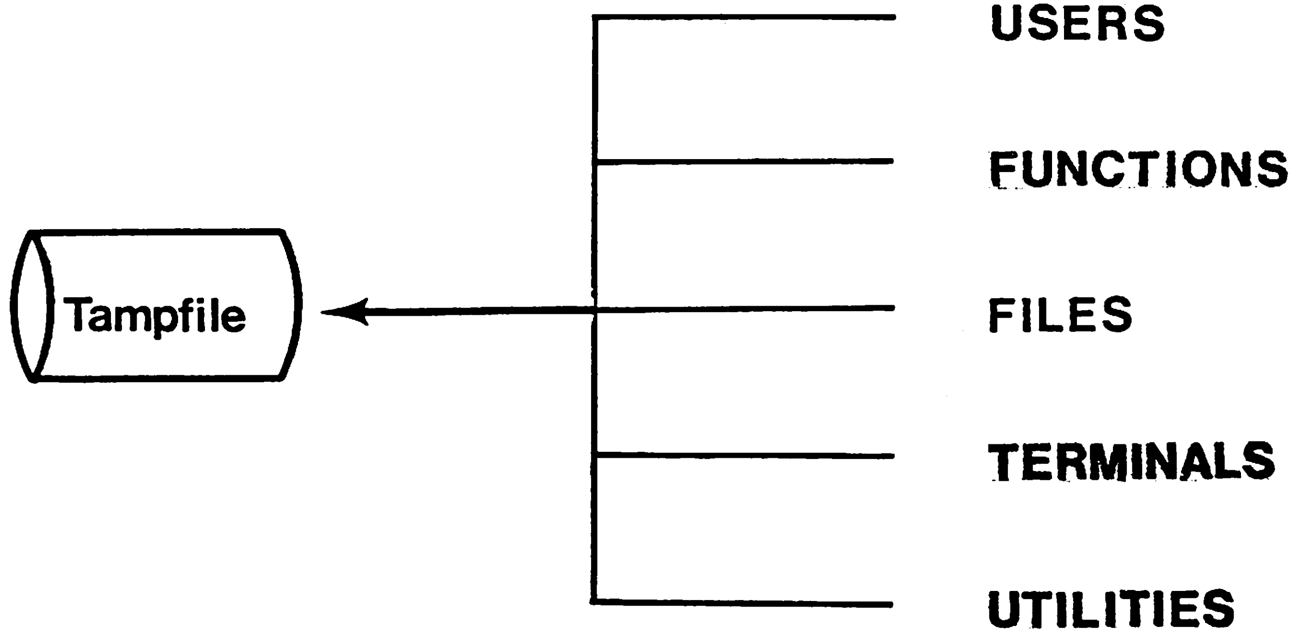
## Applications environment




TAMP
APPLICATION PROGRAMS
KEDS
SMOG
disc
stdlist
or
terminal assigned by father process

G-09.20

# TAMPFILE



G-Ø9.21

## USER

- name
- password
- function (view)
- function (call)
- processing values
- softkeys

## FUNCTION

- name
- type
- entry point
- segmented library
- run priority
- description
- data files

## FILE

- name
- type
- multi-access

G-Ø9.24

## TERMINAL

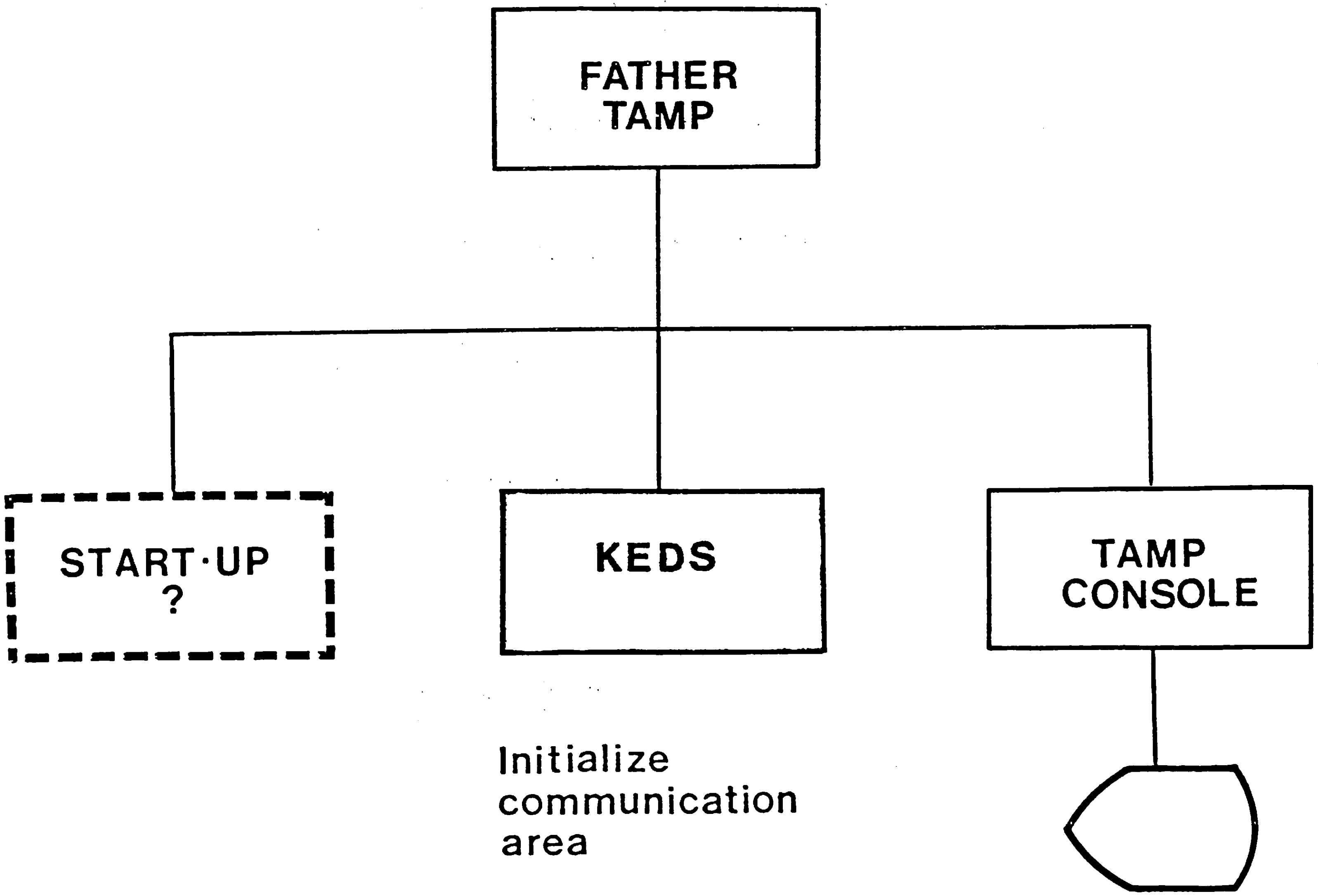
- name
- logical device no.
- hardware data

## UTILITY

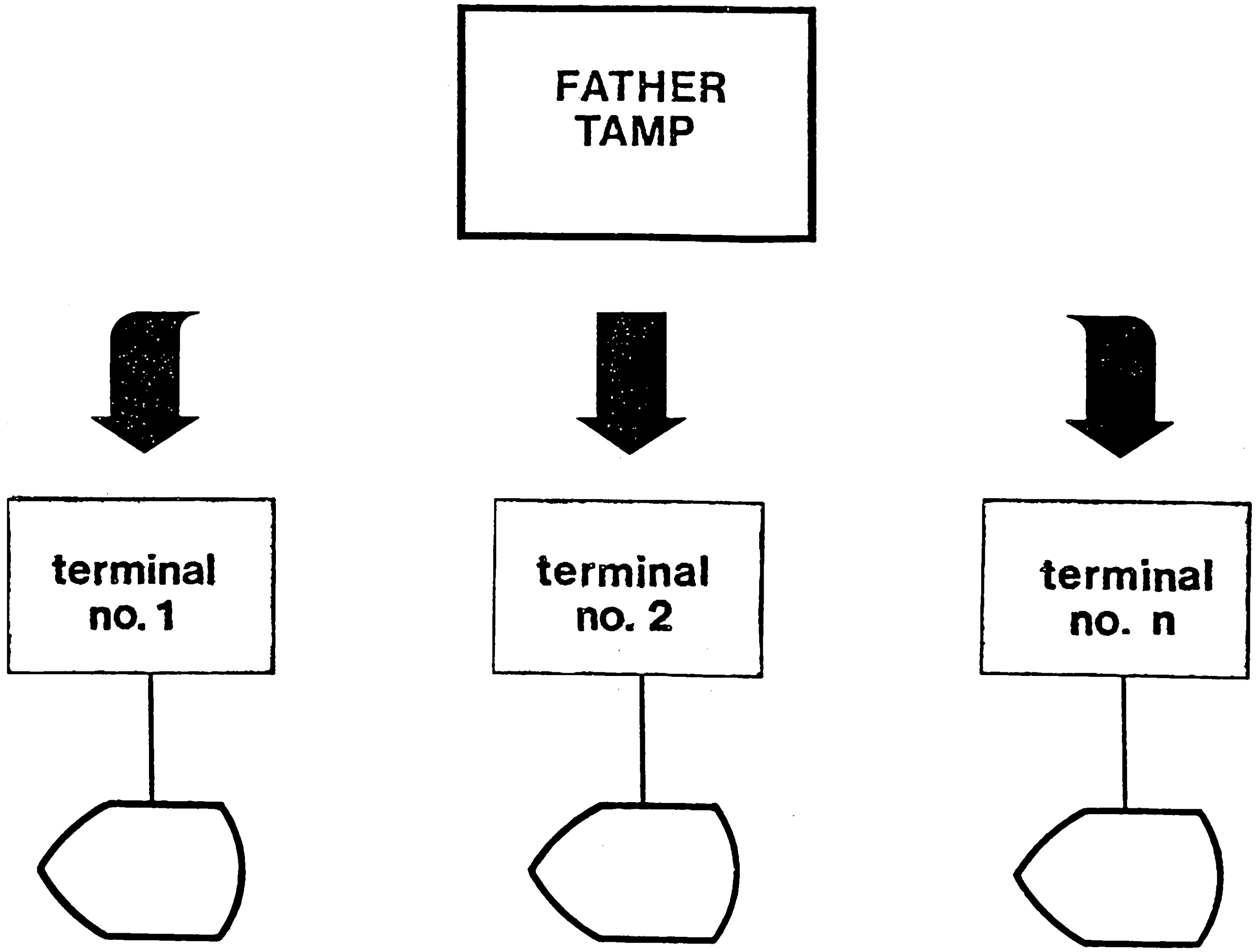
- name
- type
- entry point
- segmented library
- run priority

G-Ø9.26

# :RUN TAMP



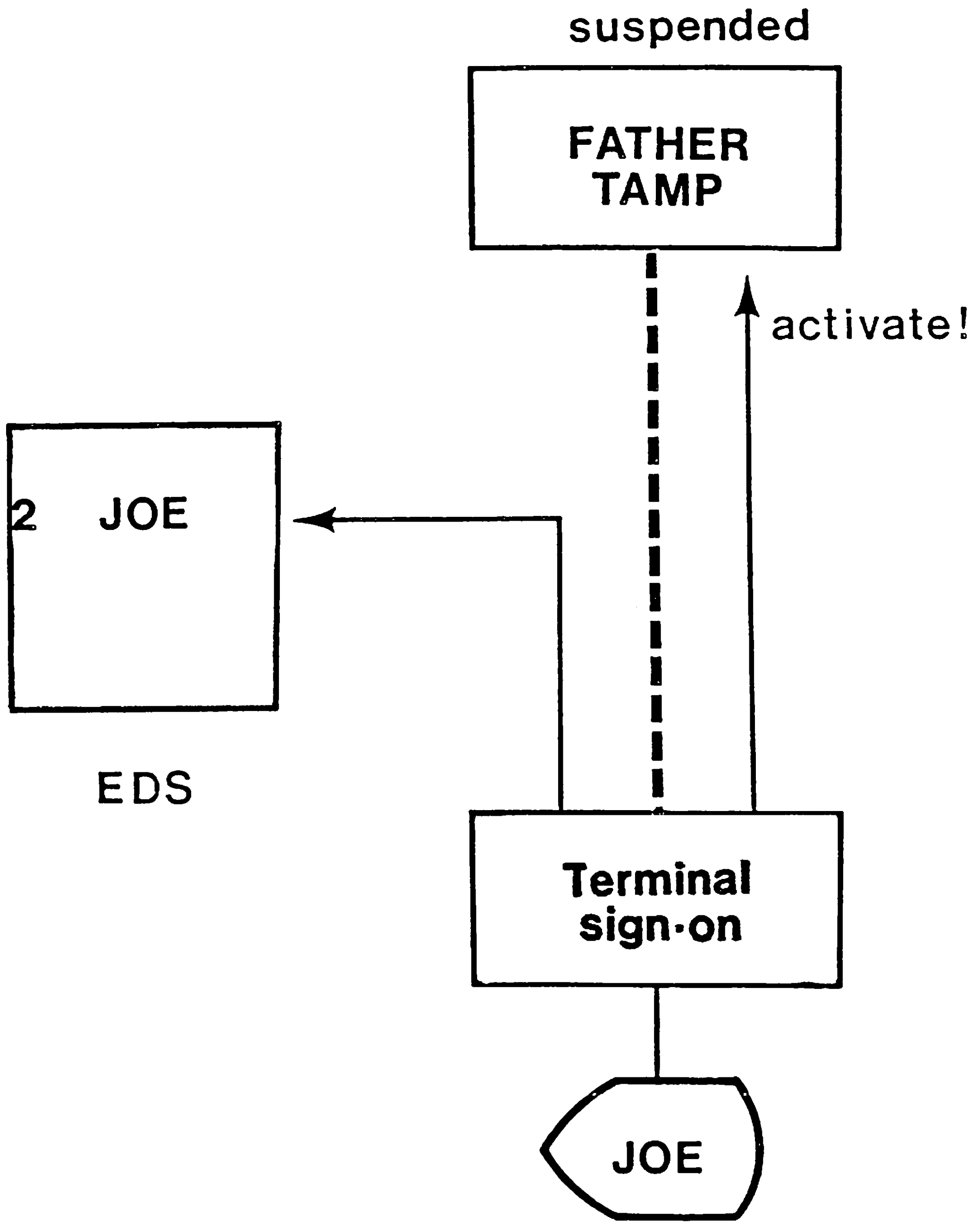
G-Ø9.27

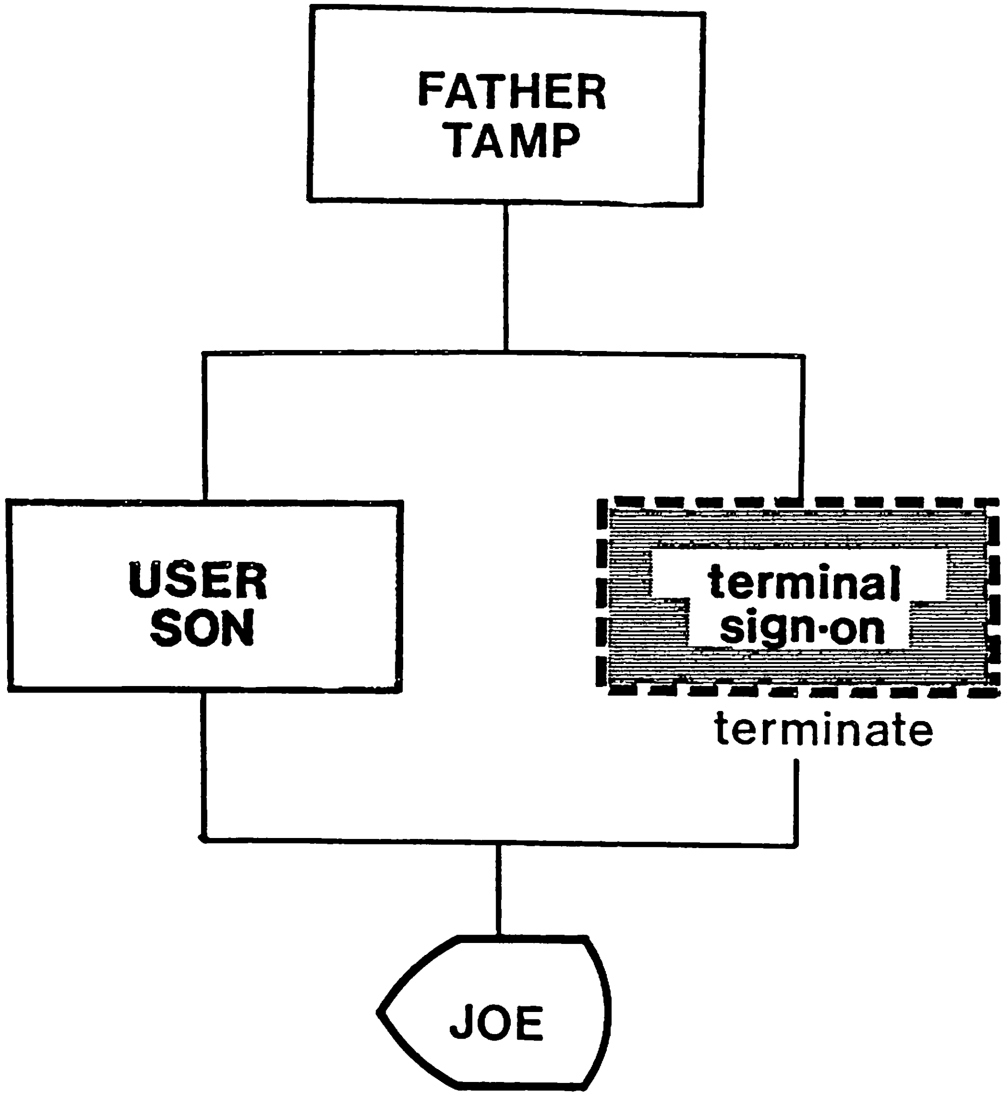


## Terminal sign-on processes

G-Ø9.28

G-Ø9.29


suspended
FATHER TAMP
activate!
2 JOE
EDS
Terminal sign-on
JOE



G-Ø9.3Ø

# USER MENU

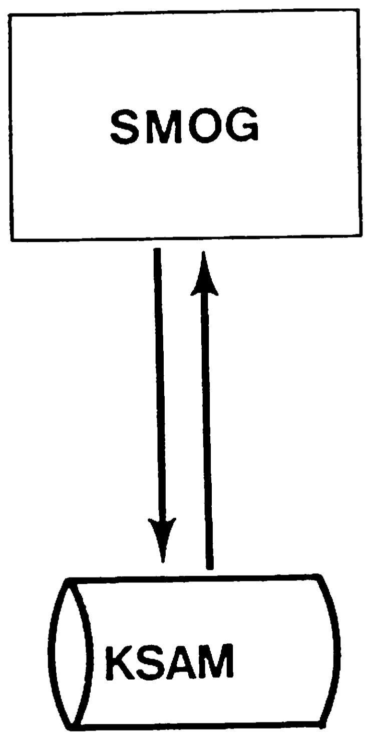
```
                                                99/99/99
                                                USERNAME


       1    ADDCUST     Add a customer

 sk1   2    FUNCNAME    This is a function description

       3    CHNGCUST    Change a customer

 sk2   4    REPTCUST    Report on a customer

       5    DELCUST     Delete a customer
```

G-Ø9.31

# SMOG

- Block mode transfers
- Modify protected & unprotected fields
- Modify & reset display enhancements
- Cursor positioning
- Cancel notification
- Field level I/O

G-Ø9.33



Build files

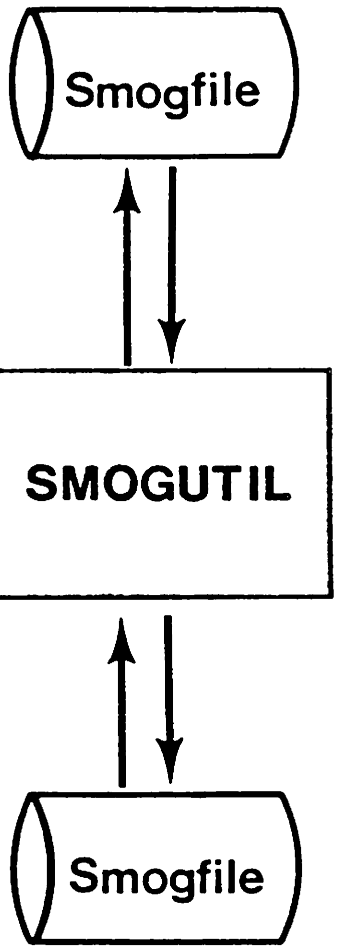
Add screens

Display ”

Modify ”

Delete ”

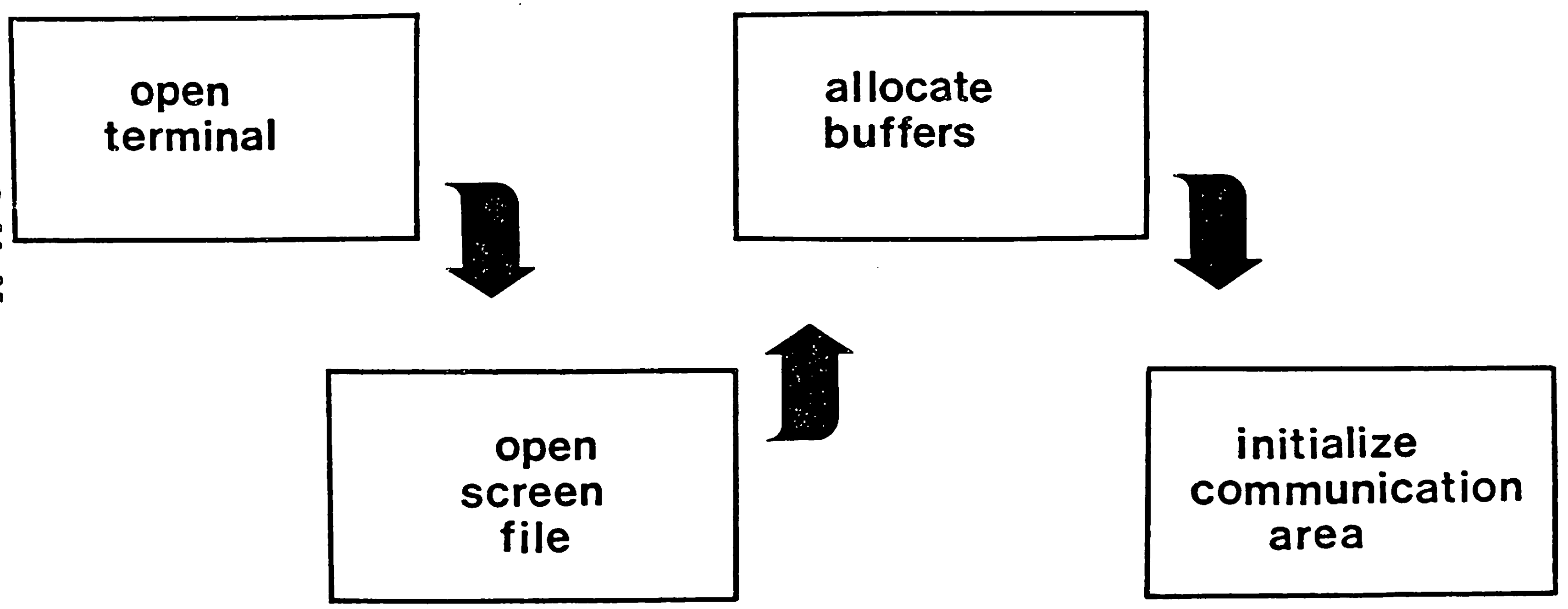
Report ”



- Copies screens
- Duplicates screens
- Renames screens
- Merges files

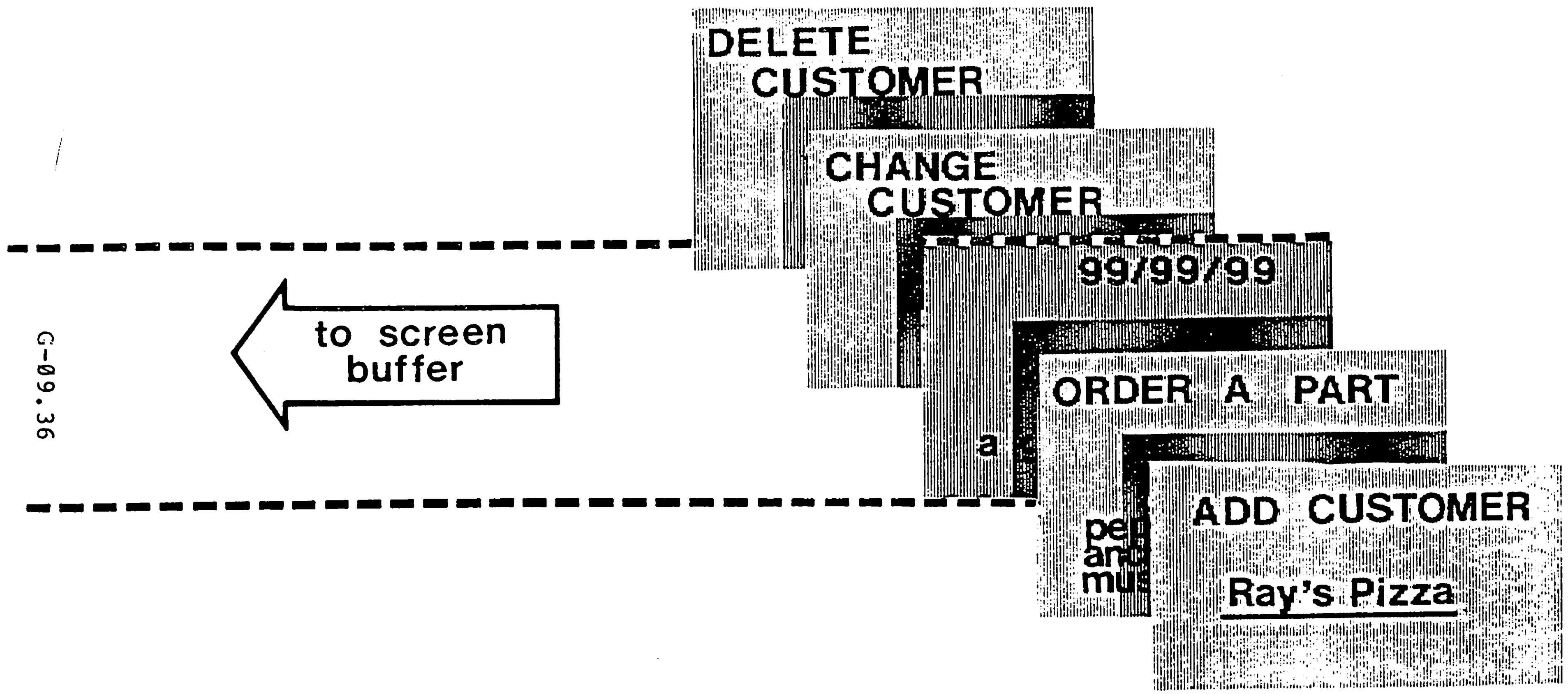
G-Ø9.34

# INITERM



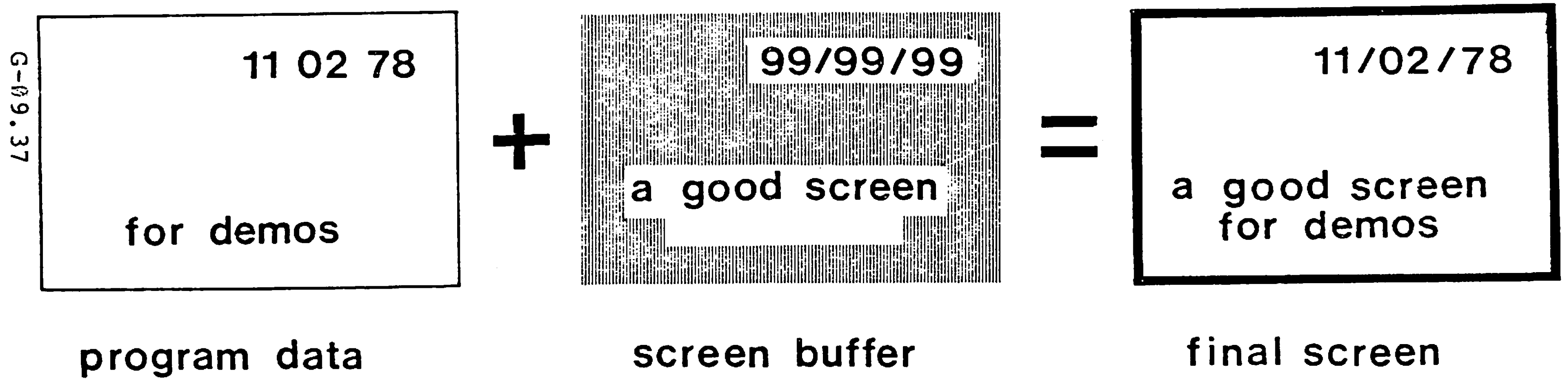
G-Ø9.35

# GETSCREEN

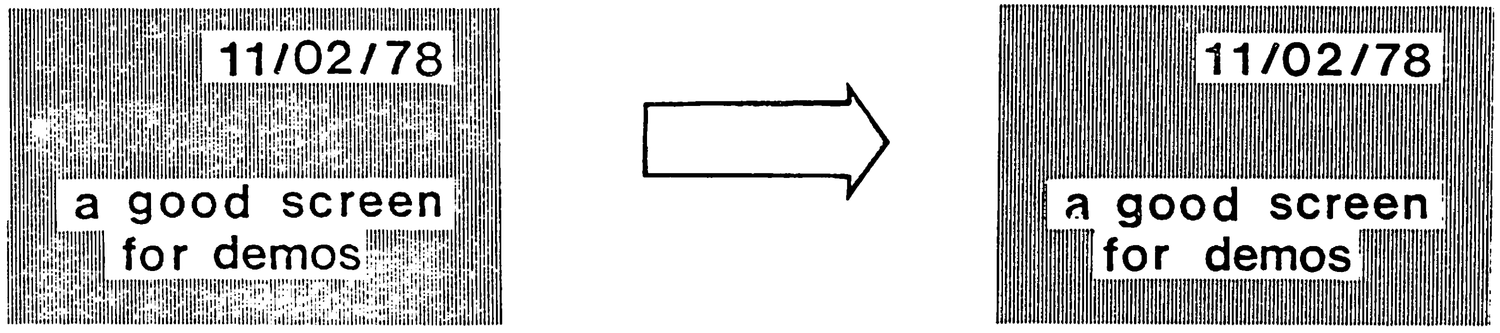


G-09.36

# MOVEDATA & ENHANCE



# PUTERM



SCREEN BUFFER

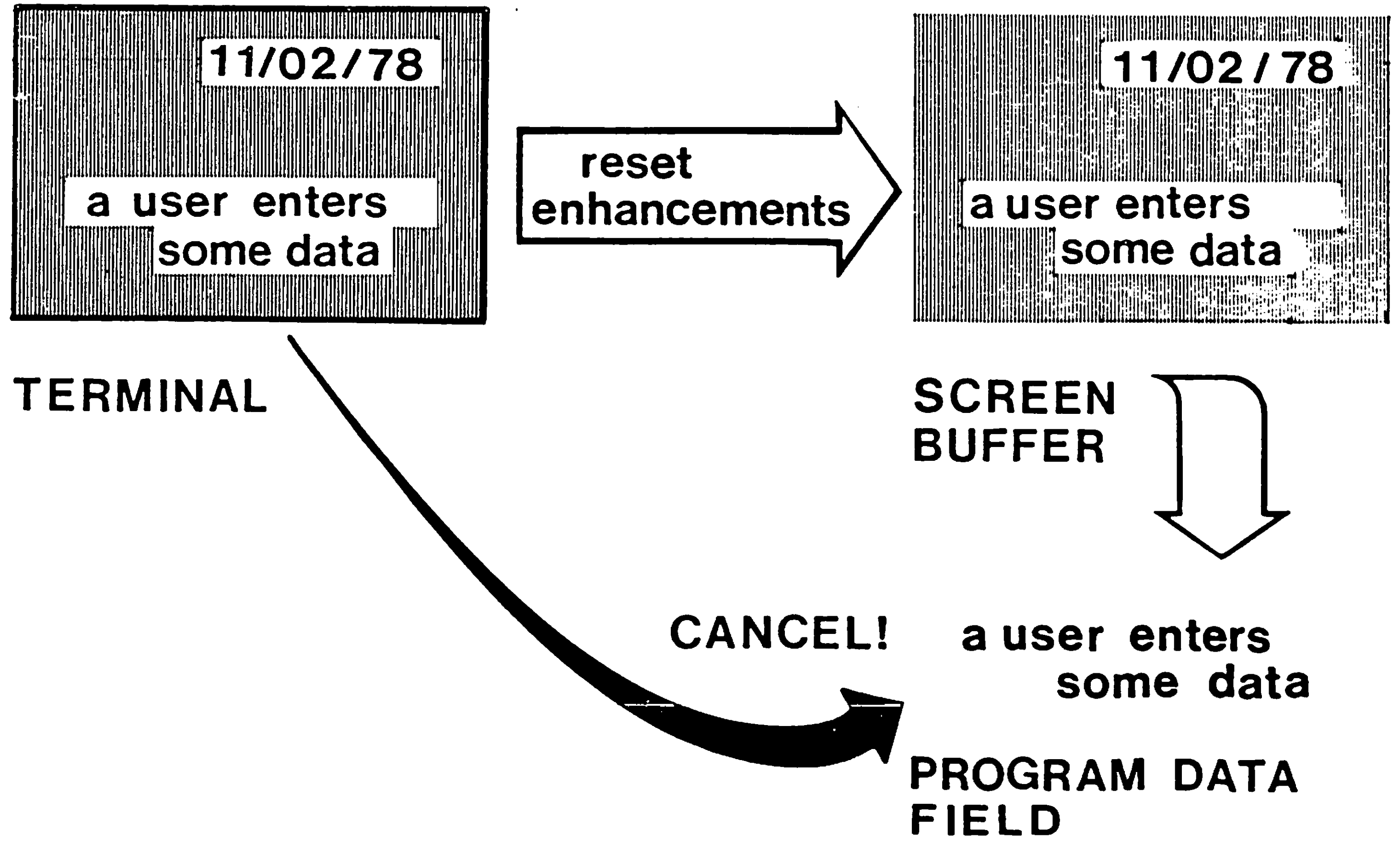
display screen
&
position cursor

TERMINAL DISPLAY

G-Ø9.38

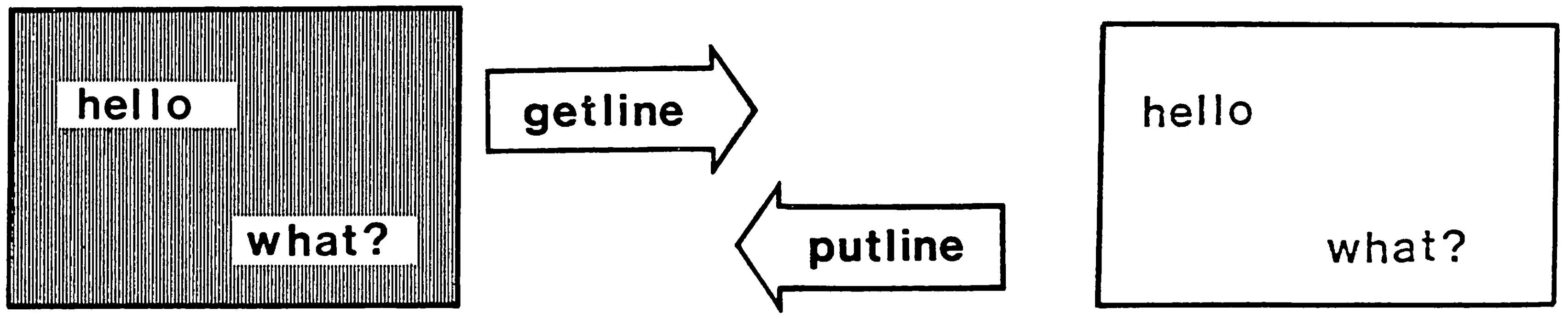
G-Ø9.39

# GETERM



G-Ø9.4Ø

# GETLINE & PUTLINE

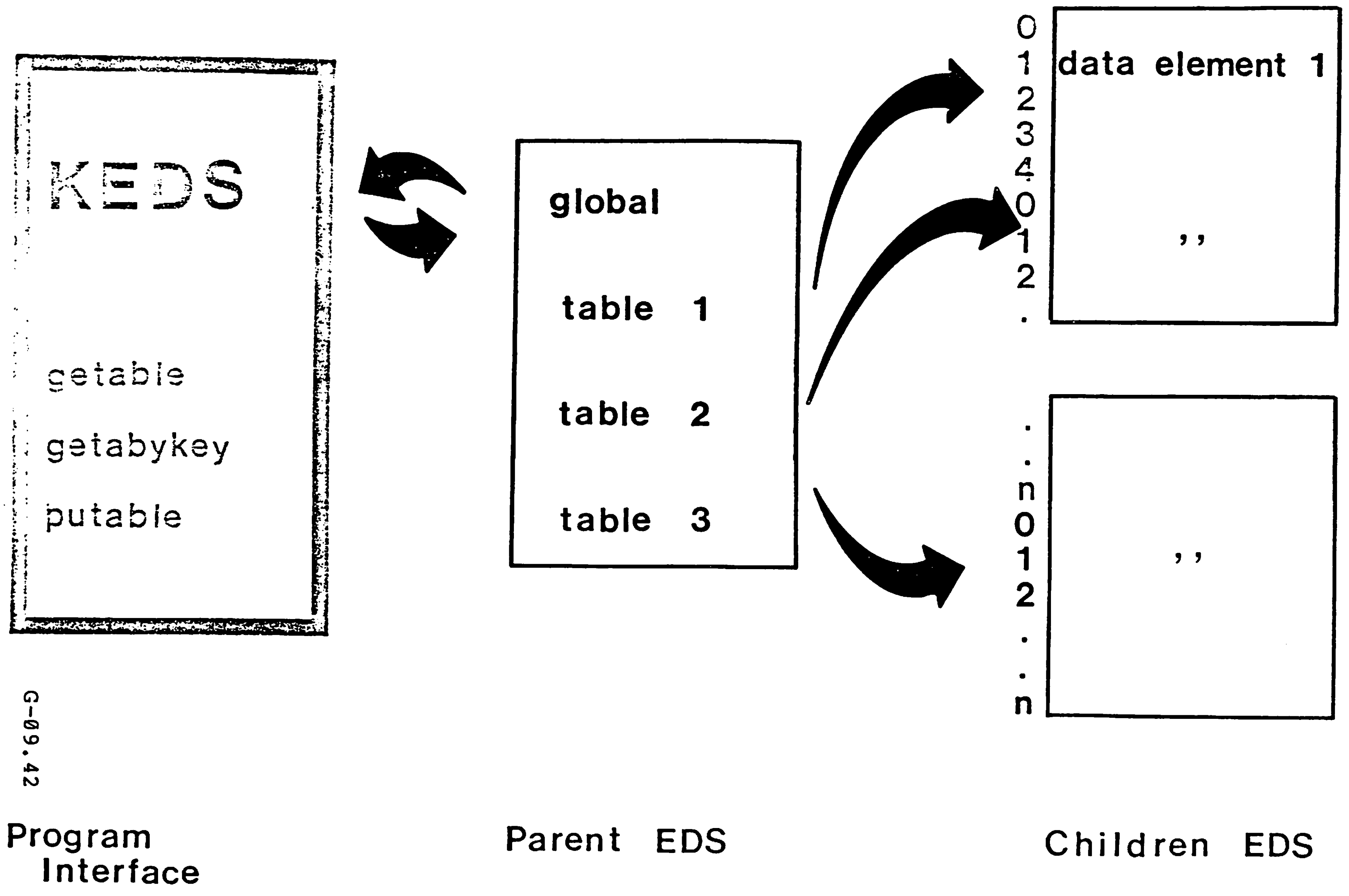


TERMINAL SCREEN PROGRAM DATA

# KEDS

- **MANAGES TABLES IN VIRTUAL MEMORY**

  - **Builds tables**
  - **Shares tables**
  - **Reads elements**
  - **Writes/updates elements**
  - **Reads elements by keyvalues**
  - **Purges tables**


KEDS
getable
getabykey
putable
global
table 1
table 2
table 3
0
1
2
3
4
0
1
2
.
data element 1
,,
.
.
n
0
1
2
.
.
n
,,
G-09.42
Program Interface
Parent EDS
Children EDS

# A GUIDE TO SYSTEMS DEVELOPMENT

JOHN M. GRILLOS

AMERICAN MANAGEMENT SYSTEMS, INC.

## INTRODUCTION

Developing a large scale computer application is an expensive effort, requiring careful, planned coordination of many different activities to be successful. This paper outlines the overall approach and key activities essential to the orderly development of a large system.

Not all management problems require computer-based solutions; careful analysis can sometimes yield significant improvements simply through new policies or procedures. However, if the problem is regular and recurring, involves large amounts of data, or requires special monitoring and control, automation should be considered.

This paper segments the development of a new system into three major phases:

-- System Concept (Phase I)

-- System Design (Phase II)

-- System Implementation (Phase III)

These phases reflect the idea that development is a learning and refining process, in which a few good ideas are expanded and detailed as more information becomes available to the point where millions of orderly pulses racing around in a computer perform the few functions intended when the development process began.

Good planning is essential to the whole process. Each project requires a carefully designed plan, giving consideration to task dependencies, available resources, non-system constraints, and management requirements. The guidelines presented here are a starting point for a task plan and a measure of task completion and project status. The milestones defined in this paper are usually accomplished regardless of the size of the system. However, smaller systems quite often allow collapsing of two phases or subphases of development. These are individual considerations which go into developing a plan.

Good documentation standards are also critical to the development process. Each project task must be documented. Documentation reduces potential confusion in verbal communication, broadens exposure to and makes explicit decisions, and allows measurement of successful completion of most non-programming tasks. This paper gives an indication of the type of information to document. Other areas in which standards are useful include: Naming conventions for data elements, segments, files, data bases, data glossary, and so on; Programming techniques for structure, frequency of comments, and naming of COBOL paragraphs; Formatting for reports, forms layouts and so on; and Environmental considerations such as operating standards, turnover standards, programming languages, use of data base, and so on.

A procedure for review and revision of all task documentation by both technical project management and concerned users should be in place. This is supplemented by substantive progress briefings periodically delivered to management. Disagreements and misunderstandings which persist through system development because of poor review procedures are very expensive to resolve in an operational system.

Beyond the process and ideas above are the following considerations which are also keys to the total success of a system development.

1. Personnel Assignments -- Beyond quality of people assigned is their orientation and training. Generally, business analysis skills are most useful in doing concept and general design work, and of course technically qualified professionals dominate staffing of detail design and implementation phases -- with business types once again involved in training and installation. However, involvement of technical talent in early project phases can head off problems caused by bad technical assumptions upon which important application functions may depend. Involvement of business-oriented personnel in late stages of development provides project continuity and, thereby, insures that user requirements stated during General Design are faithfully adhered to in subsequent project phases.

2. User Involvement -- System projects have little hope of achieving intended benefits without adequate user involvement. Users must be the primary definers of system functionality, and must be involved in critical review processes that occur at points throughout the development cycle. These requirements mean two things -- users must be actively interested in the project so that they are willing to spend the time to stay involved, and project planning must deal explicitly with user level of effort and timing considerations so that users can schedule project involvement into their routines.

3. Budgeting Process -- With good planning it is reasonable to expect reliable cost estimates for one phase beyond the current phase (e.g., reliable Detail Design costs can be prepared at the end of General Design). However, budgeting for an entire development cycle in advance or budgeting two phases into the future cannot be done with much reliability. Vagaries in staff productivity or costs, need to do tasks not foreseen in plans, personnel turnover, and a variety of other unforeseen problems seem to always overtake project cost estimates. Since full project budgets cannot be made accurately in advance, two things should be done -- budget estimates should be revised after each phase for management go/no-go decision making, and projects should not be hamstrung by attempted enforcement of premature and inadequate budgets. Such enforcement can result in no system or a system which does not meet important project objectives.

The following discussion breaks each development phase into several types of tasks. Those tasks that would be repeated during the same phase for each of several subsystems comprising a larger overall system are indicated by an asterisk (*),

## PHASE I -- SYSTEM CONCEPT

The purpose of the System Concept is to analyze a problem, define its cause, and create an economic solution, which may utilize the computer. The emphasis is on economic feasibility, but technical considerations are recognized in order to analyze the cost of any solution. However, these considerations are only examined in enough detail to derive rough measures. This phase is the first pass at a complete design.

### A. Define Scope and Effort

The first task is to make estimates of the level of effort to be expended in examining the problem. A plan must be developed scheduling the tasks described in the remainder of this section.

### B. Review of Current System

The current system should be examined to provide a base line against which the value of a new system can be measured. The current system includes all manual and computer systems considered for change. Some necessary information to be documented includes a description of processing, inputs, outputs, processing volumes, personnel utilization, computer utilization, and costs.

User review of all documentation should be done. Documentation should be reviewed quickly, yet in detail. The outputs and functions defined here provide the base line for measurement and minimum requirments of any new system.

## C. Determine Analysis Areas

Based on the above review, several key areas usually stand out as being points where additional analysis is critical for system success. These "leverage points" may have any of the following characteristics to distinguish them from other potential concerns:

1. High Volume Processes -- where small improvement can yield high return.

2. Simple Clerical Tasks -- are usually easy to mechanize.

3. Timelines and Quality of Input Data -- the cost of processing bad data for correction is usually excessive.

4. Cash Management Benefit -- earlier billing and later payment have measurable savings.

5. Timelines of Reports -- untimely reports usually force a secondary system to be created for control.

6. Reduction of Codes -- several codes used to define the same characteristic are very frustrating to users.

7. Management Interest -- any point where a concern has been specially expressed should be analyzed.

## D. Analysis

Analysis is the key step. The areas defined in the previous task are investigated in detail and alternatives for providing the required service in an efficient manner are considered. Analysis content will vary widely from project to project, and the process must be documented. The goal is to define a feasible solution which is <u>economically</u> superior to the present system and other alternatives. Each alternative must be described in functional terms, stressing the comparison with the present system and other alternatives. The cost of operating the system <u>once it is in place and operating</u> must be determined. The level of detail should not be too great; if the economic feasibility of an alternative is so marginal that it hinges on a very detailed analysis, the alternative should be avoided. All benefits should be quantified if possible. Personnel reductions, cash flow, and inventory reductions are obvious examples.

Reports can be evaluated in terms of the action that could be taken if the information were available. However, it is usually best not to attempt to quantify their value. Any future benefit possibilities should be assigned an expected value. Cost avoidance should also be quantified and stressed.

## E. New System Concept

The results of the analysis areas are integrated into a new total system concept which also incorporates the definition and analysis of common system components which serve the entire system, such as report generators or data bases which were not explicitly considered before. Other elements of the concept are a system flowchart, narrative description which emphasizes the flow of data and processing of major exceptions, a list of features, a list of remaining issues and criteria for resolving them, and a system operating cost/benefit summary.

## F. Phase II Task List

The next task is to create a task list for the design of the system. Depending upon the size of the project, the task list may include General Design or both General and Detail Design. The task list includes the amount of resources required for completion.

## G. System Development Cost and Schedule

Development cost for the second phase is derived from the task list by assigning a cost to the resources required. In order to estimate the cost of later phases, an estimating procedure based on the number of major files, programs, reports and forms can be used. Table 1 shows some planning factors which may be adjusted by experience to the personnel actually involved. The resources required for Phases IIIB and IIIC can only be estimated by personnel familiar with the user's environment.

Since the system is very sketchy at this time, the number of files, programs, reports and forms may not be accurately estimated. To correlate the costs developed this way, Table 2 can be used as a rough measure. If results generated from Table 1 differ greatly, well-defined reasons should be required.

TABLE 1

Development Cost Estimation
(Man-Days)

| | Major Files | Major Programs | Minor Programs | Reports and Forms | Project[1/] Management |
|---|---|---|---|---|---|
| General Design | Task List Expansion | | | | 20% |
| Detailed Design | 15 | 10 | 5 | 3 | 20% |
| Programming and Unit Test | | 35 | 12 | | 20% |

To correlate the estimate generated from the "bottom up" approach, the following guidelines can be used as indicators of distribution of cost between phases.

TABLE 2

| | |
|---|---|
| Phase I | 5% |
| Phase IIA | 10% |
| Phase IIB | 20% |
| Phase IIIA | 35% |
| Phase IIIB | 15% |
| Phase IIIC | 15% |

1/ Project Management is calculated as a percent of man-days on other tasks. As the size of the project grows above 20-25 personnel assigned, extra management above this may also be required.

Schedule should be defined at this time. The development sequence of each subsystem including precedences and implementation priorities can be illustrated by a staging chart. A required implementation date, a limit on the maximum personnel assigned to a project, or a known team are possible constraints to the schedule. The goal is a rough schedule for initial planning purposes.

Finally, return on investment or other financial viability measures should be computed using quantified system benefits, operating costs and development cost schedule.

## H. Management Presentation

As stated earlier, user involvement and management review is critical. The review at the end of the system concept is the key to beginning the actual design project.

## PHASE II -- SYSTEM DESIGN

The goal of the design phase is to devise the most effective and efficient means to provide the services which were earlier defined as economically desirable. Where the System Concept analyzed economic feasibility within wide limits of technical capacity, the System Design phase analyzes technical feasibility of alternative methods to provide the desirable features. If the project is large, the design phase is broken into two subphases: General and Detailed Design.

### A. General Design (Phase IIA)

The General Design phase is user oriented. All user issues should be resolved. Manual interfaces and processes, code types to be used and reporting needs are defined. The end product of the General Design is a specification of user requirements and interfaces, and a defined solution to the major technical problems of the system. The technical feasibility of the design must be thoroughly analyzed and alternative technical approaches devised and evaluated. The following tasks are usually required:

1. Planning

The task list from System Concept must be expanded into a working plan for this phase. Personnel must be matched to tasks, tasks grouped and assigned for effective continuity, and task interdependencies considered. Since integration of the subsystems is a primary goal of General Design, significant time must be reserved for analysts to review each other's work. Review procedures for both users and technical analysts should be precisely defined here, as well as procedures for resolving stalemates.

2. Issue Analysis

The issues to be resolved were defined in the System Concept phase. This analysis is quantified as much as possible, though these issues are generally harder to assign dollar values. A mechanism for resolving issues must be defined. Usually, a specific person is designated to approve issue resolutions. It is also critical to impose and document deadlines for issue resolution.

*3. Define Inputs and Outputs

The problem of which functions must be defined first (input, output or process) continually arises. It is often a restatement of the "chicken or the egg" riddle. The key here is to pick a sequence, and proceed with efficient timely analysis. In general though, a firm grasp of user reporting needs serves as a good starting point.

<u>Codes</u> to be used in the system must be defined as the design progresses. This task is completely overlapped with file design, and is also sometimes a major issue. There must be some explicit addressing of the use and purpose of each major code.

<u>Report definitions</u> should include: narrative descriptions, contents, purpose, users, frequency, and media. <u>Files</u> (both computer and manual) must be adequately described, including:

- Type of file.
  - System Master Files. Most of the subsystems use these files, though they are normally built and maintained in a separate subsystem. Designers of master file subsystems are often key personnel who insure system integration and finalize code definition. Master file definition is especially difficult to schedule since master file requirements may change several times during the design stage because of issues arising in processing subsystem design or technical problems which surface during the design.

- Subsystem Master Files. Used totally within a single subsystem; and contain relatively stable information to be updated and maintained by the subsystem.

- Input Files. Information from other subsystems is defined by the subsystem requiring the information, except, of course, at the initial entry of a source document.

- Description of files.

  - Description of Use. Narrative format.

  - Data Elements. Description, rough size estimate and frequency.

  - Records. Grouping of elements.

  - Processing Intent. A description of what the file is used for and how it is to be processed (e.g., direct or sequential).

  - Organization. Definition of how the file may be organized, e.g., IMAGE, KSAM.

Input documents must be described. The level of detail varies with importance to the system, and the system operating environment. Critical input documents should be roughed out and approved by the user. Occasionally, field testing is required, but usually critical documents should not be finalized in this phase.

Communciations requirements should be defined. Aspects to be considered and documented include rough estimates of line loading, location and types of terminals, transmission rates, recovery feasibility, fall back procedures, and alternative approaches.

*4. Computer Processing

Computer processing is described in terms of subsystem functions, not programs. The division of these functions into programs does not take place until Detail Design.

Subsystem flowchart definitions describe process functions connecting inputs and outputs. The functions are simply logical groupings of data manipulation, and definition of these is the key talent of a system designer.

Function definitions are detailed narrative descriptions of what each function will do. This does not mean a decision block level of flowchart or description. It must be in terms understandable to a non-technical system user.

Computer resource estimates based on the frequency of processing cycle and a profile of the inputs should be made. If technical feasibility of any alternative appears uncertain as a result of this analysis, further refinement by the use of simulation, emulation or more detailed analysis may be required. Establishment of technical feasibility is a requirement of this stage.

*5. Manual Procedures

Manual procedures related to system function should be documented. A processing flowchart should be prepared depicting operations to be performed and organizations which will handle inputs and outputs. Detailed narrative descriptions should be used to describe coding information on forms, balancing needs, error correction, data entry and verification or validation. Personnel requirements over daily and monthly cycles, including profiles of activity over time (peaks and averages) should be defined.

6. Environmental Conditions (System Configuration)

The tasks in this area are concerned with technical support for the development and operation of the system.

- Software Environment -- or system support utilities as well as more general installation support is analyzed here. Alternatives must be considered when differences in cost or maintainability exist. Objects of analysis may include: data base approach, use of a report generator, inquiry facilities, use of a test data generator, and programming languages.

- Training -- of programmers and analysts for later development phases should be planned and begun, if possible.

- Hardware Environment Analysis -- should be performed at the end of the General Design since it requires evaluation of all subsystems. If alternative hardware is under consideration, system resources or needs must be analyzed, including on-line storage, CPU utilization, (total hours required per processing cycle), I/O elapsed time, order lead time, and communications support needs.

7. System Flowchart

This is a key ouput from the phase. It should be understandable to both user and computer personnel, and provide an interface for discussion purposes. Two levels should be produced. A summary level chart reworks the Phase I flowchart, with the same features, issues, descriptions, etc. An integrated detail chart links subsystem flowcharts and is used to check for integration. All detail subsystem charts are included on this consolidated version.

8. Cost Benefit

This is reworking of operating cost and benefits in light of new detail to insure continued economic feasibility.

9. Phase IIB Task List

A detailed task list and resource estimate is now required. This task list may still be in terms of process functions rather than detail components, depending upon staffing uncertainties, start date for next phase, etc. If so, the list will be expanded early in Detailed Design, as discussed later.

10. Priorities for Development and Implementation

This is a refinement of the initial discussions with user groups, and must be reviewed with them. Precedences may be clearer and costs of partial implementation can now be considered. Priorities may not indicate a rigid scheduling procedure, due to alternative distributions of project resources (e.g., if one subsystem can only use four analysts and six are available, then a second subsystem can be worked on). Priorities act as guidelines in assigning scarce resources to the project. Some factors to consider are extra bridge (or conversion) programs, revision to high priority programs when the full system is in place, early realization of benefits and progress visibility.

11. Development Cost and Schedule

The development cost can now be re-evaluated with the detailed Phase IIB task list. A further refinement of Phase III costs is also possible since a firmer estimate of computer programs is possible. A rough schedule must also be defined utilizing the priorities and precedences known.

12. Management Review

The management review process has only to cover changes in scope, technique or benefits from the Phase I presentation. A progress report may be all that is required if changes are minimal. However, technical feasibility must be firmly established at this stage and should not be in doubt upon management review.

## B. Detailed Design (Phase IIB)

The Detailed Design phase of a systems development project develops system specifications in sufficient detail to accomplish the programming and implementation tasks. If the system is small, then separating the design into two sub-phases may not be necessary. In any case the outputs should be the same. If the implementation plan places development priorities on the various subsystems, the higher priority subsystems should have their designs completed before the lower priority subsystems.

In this phase, process functions are grouped into programs, and detailed specifications are developed for these programs. Rough report formats are developed into report layout specifications, and all reports, including activity lists, are specified. Input forms are finalized. File contents are refined to detailed file specifications, and layouts are prepared for all files. Manual procedures are further detailed to the level of detailed job specifications.

Non-application activites of this phase include making final arragements to insure the availability of hardware and software (including communications facilities) to provide a testing and implemention environment. Also, a series of final volumes are prepared which serve as the system documentation for implementation and later maintenance.

The key aspect of Phase IIB then is a further refinement of design and planning work done up to this point.

The level of detail required for a detailed program specification is determined by two things:

-- The level of qualification of the programmers who will have to program from these specifications; the more senior, the less detail required.

-- The staff continuity between Phase II and III. That is, if the same people who are designing the subsystem will also be programming the subsystem there is a reduced need for detail.

The following sections discuss the items to be accomplished during Detailed Design.

### 1. Detailed Project Plan

A detailed project plan must be prepared to control the Detailed Design Phase. This plan is based on the task list prepared at the end of Phase IIA, and reflects the priorities of subsystem development.

### 2. Resolve Remaining Issues

There will surely be some issues remaining from Phase IIA. These should receive the highest priority for resolution before proceeding, and be settled very rapidly.

3. Codes

It is necessary to define what new codes will represent. Consideration may have to be given to alternative coding types (structured codes and random coding values), specifying the length and composition of each alternative. Give consideration to the integration of new codes; that is, if a similar code is used elsewhere in the same business, try to maintain a consistency of the structure, and assigned values.

4. Detail Data Base Design

Although this section is primarily devoted to considerations of a data base system, some of the concepts are useful and should be considered even when not in a "formal" data base system:

- Produce a glossary of terms for data elements for the project. This glossary will be constantly maintained throughout the project (and possibly throughout the use of the facility). It should include the element name, a description, its "picture", the source of the element, (eventually) which files it is in, and where its (subsystem) maintenance responsibility lies.

- For large projects, data base design is a further (but not final) refinement of the concepts and data needs studied in Phase IIA, and includes developing an overall data base design, examining data relationships both inside and outside the project, and developing initial data base size estimates. Consideration should be given to redundancy and efficiency tradeoffs and integration with other system DB requirements. Size estimates should attempt to deal with overhead storage for system flags and pointers.

- The final data base design is done later in Phase IIB, and should include finalized physical and logical layouts of the data base, access methods, estimates of the frequency and timings for backing-up, recovery, and reorganizing the data base, and an outline of recovery procedures.

- If necessary, select SYSGEN options to optimize the performance of the data base facility for your project or system. TOTAL has relatively few SYSGEN options, whereas IMS has many.

*5. Detailed Subsystem Flowchart

The starting point for all detailed design work is a detailed subsystem flowchart, showing all inputs, programs, files, reports and other outputs of the subsystem, and how they connect. There must be firm agreement between technical staff and the users regarding this subsystem flowchart before proceeding with Detailed Design.

*6. Inputs/Outputs

For each subsystem, the inputs and outputs must be fully specified in detail -- this includes reports, files (data bases), and input (source) documents. Each of these are treated below:

- <u>Reports</u> must be specified in detail. Such things as a detailed report layout (with descriptions of the contents of each field on the report), a general description of the contents and purpose of each report, an estimate of the volume and frequency of generation, output medium, and special code values (e.g., error codes) should be indicated in detail.

- <u>File</u> specifications (layouts) consist of at least the details of the data elements and their groupings (e.g., COBOL-like layout or data base layout). The data element lengths, type relative positions, and frequency (of specific groupings or record types) should be given, along with record lengths and blocking factors.

- <u>Input (source) document</u> formats should be finalized at this point. Besides form layouts, keying formats, revised volume and timing estimates, and a chart which cross-checks the data elements with the source documents to insure that all data required is being captured should be prepared.

- <u>Communications</u> steps include revising volume estimates, and other data, determining detailed back-up and recovery procedures, and, if there are dedicated facilities, detailing the entire communications network (lines, their locations, line type, conditioning, maximum transmission rates, modem requirements, multiplexing facilities, and any other data communciations control equipment requirements). A detailed cost analysis should be performed on this planned network, if not already done.

*7. Computer Processing

The initial task plan for Phase IIB which was prepared at the end of Phase IIA was process function oriented. However, the system is really program oriented. The relationships between these process functions and the programs to be specified have been given in #5 above. Now, detailed program specifications must be prepared, including:

- The program name, program identification, subsystem, inputs, and outputs.
- A brief description of the purpose of each program.
- Frequency of running and the estimated "stand-alone" run time.
- Recovery in the event of an external malfunction.
- A detailed program specification and corresponding detailed description should be prepared. As noted previously, the level of detail is primarily determined by the skills of intended implementors. Flowcharts will serve as the principal documentation of the programs for maintenance purpose. Therefore, accuracy is vital. A written narrative should be keyed to each box on each flowchart to elaborate on the meaning of that step.

*8. Manual Procedures

The need to have clear, workable manual procedures at the design stage cannot be overemphasized. General ideas of the manual procedures have been specified during Phase IIA, but these must be reviewed and detailed at this time. These procedures must be thoroughly reviewed and approved by the users. If possible, field tests of these procedures should be accomplished before their finalization.

9. Environmental Conditions

Technical support and planning must be provided in the software and hardware areas to insure that the necessary facilities are available for implementation of the system.

- <u>Software</u> support must be provided as indicated below:
  - Software decisions made in Phase IIA (e.g., data base management systems, reporting packages, etc.) must be reviewed and finalized. If they are no longer adequate, specific alternatives must be analyzed and recommendations made.
  - The software to support testing and implementation must be purchased and made available. The detailed implementation plan should be a guide as to when the various pieces of software must be available.

- Basic technical support must be provided to the implementation, e.g., test data generators, standard error handling routines, program generators, data dictionaries, or special purpose routines. Because these utilities need to be available prior to programming, their development or installation should be started as soon as possible to insure that they are available when needed.

- Test plans for the system software should be evolved and executed.

- A detailed training program should be developed for the programmers, which would include familiarization not only with the new software systems (e.g., data base management systems), but also with the use of the software support utilities being developed.

● <u>Hardware</u> support must also be provided as follows:

  - The system resource requirements must be redone with the availability of the detailed design, and the CPU utilization, channel requirements, disks, tapes, terminals, and communications needs must be detailed. An order or modification of previous orders must be made. The delivery of hardware should be in phase with the needs for a testing, then an implementation environment.

  - Detailed plans for the physical facilities for the hardware (e.g., electricity, air conditioning, floor space, false flooring) must be made.

10. System Flowchart

Final subsystem documentation must be tied together to provide for easy availability for programming and maintenance. The items to be prepared are:

● A detail flowchart at the individual program level must be prepared for the entire system. This should be a combination of the flowcharts for the individual subsystem as done in #5 above. This flowchart should be backed up by a descriptive narrative, and supporting tables of programs and files. This flowchart is also very useful in ascertaining whether all subsystems interfaces are covered.

Final subsystem volumes (one per subsystem) should be prepared at the end of Phase IIB. These will serve as the final documentation for the subsystem, and should be a compendium of all files, programs, reports, and procedures previously prepared. Any later changes made during programming and conversion should be reflected in these volumes so that they can also be used to support maintenance.

11. System Implementation Project Plan Task List

A detailed task list must be prepared to cover the programming, systems testing, and conversion and training phases of the system development. The items to be covered in this plan are acceptance and performance criteria, coding and unit testing tasks, a detailed plan for program and system testing, and implementation phasing.

12. Management Review

A system overview presentation should be made to higher-level management and the users at the end of Phase IIB to gain specific approval to proceed to Phase III.

## PHASE III- SYSTEM IMPLEMENTATION

Phases I and II accomplish the key tasks of definition and detailed specification of system scope, functions, and components. Phase III proceeds with these specifications to develop an integrated system of computer programs and manual procedures which is ready for installation in a production environment.

System Implementation consists of programming and unit testing, system testing, and conversion and training.

### A. Phase IIIA -- Programming and Unit Testing

During this phase, detailed program specifications are analyzed, and individual programs are coded, compiled and tested.

1. Revise the Project Plan and Specify Detail Personnel Allocations and Work Schedules

2. Familiarize Programming Personnel with the Project

To ensure smooth project continuity, it is important to instruct programmers on system functions and components before they begin detailed coding. This is done by having them review detail specifications including relevant charts, file layouts and functional narratives, and then having programmers attend any formal training sessions planned for in Phase IIB (e.g., data base training courses).

*3. Program Coding

Detailed program specifications must be interpreted and translated into computer programs.

- Review detail program specifications.
- Where necessary, elaborate on program specifications in order to transform them into the working documents needed for program coding. At this point, it is useful to analyze input files, output files, processes, detailed calculations, and error conditions for correctness, consistency and potential use in program testing.
- Produce and compile program source code.
- Develop a plan for testing individual programs.

*4. Unit Testing

Individual computer programs are tested to insure that they meet program specifications.

- Install and checkout unit test software. This software, including utilities and aids to facilitate program testing, was defined and specified during system design.
- Generate unit test data. Files needed for program testing can be produced by both programmers, who generate files for a single program and a central technical support staff which creates system test master files used by groups of programs.
- Test programs and revise where necessary.
- Evaluate initial test volumes and timings in relation to the estimates developed during system design.
- Review unit test results and certify programs as ready for system testing.

B. Phase IIIB -- System Testing

Phase IIIA produces individual working computer programs which meet detail program specifications. System testing ensures that these programs and manual procedures operate together in the integrated manner necessary to accomplish system functions.

*1. Develop system acceptance criteria and test plan. The test plan normally covers such areas as input data to be used, expected outputs, interaction between computer system and manual procedures, system acceptance criteria, test plans for logical group programs, and a plan for phased, integrated system test.

2. Install system test software. These programs including utilities and aids to facilitate system testing were defined and specified during system design.

*3. Generate necessary input, master and intermediate system test data files.

*4. Conduct system test and revise programs where necessary.

- Analyze known inputs, resultant outputs, and the interaction between computer programs and manual procedures.
- Compare system test volumes and timings with benchmarks developed during system design and unit testing.
- Update system and programming documentation.
- Review system test in relation to system acceptance criteria, and obtain user approval for final implementation.

C. Phase IIIC -- Conversion and Training

1. Conversion

- Complete systems, operations and user documentation.
- Convert required master and systems files.
- Conduct pilot test as the first step of the phased system implementation plan. Steps include completing training of user personnel, installing any necessary pilot hardware of software, running the pilot test as a parallel run against an existing system (if applicable), and evaluating the pilot test in light of systems acceptance criteria. If necessary, revise programs, procedures, documentation and manuals based on results of the pilot run.

- Obtain system acceptance and begin full scale system implementation.
- Final implementation and signoff.

2. Training

   - Develop a plan for training users, operations personnel and systems maintenance personnel.
   - Conduct training sessions for users, operations personnel and systems maintenance personnel.

# DECISION TABLES - AN EFFECTIVE PROGRAMMING TOOL

DANIEL F. LANGENWALTER

INTEGRATED BUSINESS SYSTEMS, INC.

Manufacturing businesses have many information processing problems such as the rules for ordering materials, the rules for granting credit, the logic of what to build to meet a customers requirements, the rules for union dues, deductions for savings and pension plans, tax requirements, etc.

Decision Tables have been used to express the logic in a manner which people can understand and also can be used as a computer program. Both General Electric and B. F. Goodrich Chemical have reported a three-to-one increase in programming productivity when using Decision Tables. Here are some examples of applications which illustrate the power and simplicity of Decision Tables.

Turbine blades are up to five feet long and must be machined to airfoil shapes. A FORTRAN program produced the magnetic tape necessary to control the Excello contour milling machines. Unfortunately, some bug caused the milling cutter to gouge a blade once in a while.

The decision was made to do the program over using Decision Tables along with calculations. The analysis and tables were completed in five weeks. The subroutines were completed in two more weeks. The program was independently checked and debugged in three weeks and has been running successfully for ten years. The people involved say that this success in solving a difficult logical problem is due to the use of Decision Tables.

We started using Decision Tables when we were challenged to automate an entire business.

The business chosen for this work built electric indicating instruments for industrial panels. These included D.C. voltmeters and ammeters and AC voltmeters, ammeters, wattmeters, frequency meters and power factor meters. To get a bill of materials for each instrument we decided to use Decision Tables to generate the bills. One of the Decision Tables is like TABLE 1000 - ARMATURE ASSEMBLY, Figure 1, next page.

```
1000 TABLE - ARMATURE ASSEMBLY
1010C FREQ-   APPLICA-  RATING  NO. OF   SCALE   TYPE OF   QUAN.   ARM
1020C UENCY   TION      UNITS   PHASES   DIST.   ARM       COILS   DWG
1030C
1040   FRQ  &  APPL  &   RU    & NP    &  SD   #  TA=    & QAC=  & AD=     & GOTO #
1050 EQ  0  &EQ DC   &EQ --    &EQ -- &EQ --   # DCMC    &   1   & 17614   & 1210 #
1060 NE  0  &  RECT  &   --    &   -- &   --  #DCMC     &   1   & 17614   & 1210 #
1070    --  &  DCMR  &  AMPS   &    1 &   --  #  IV     &   0   & 17617   & 1310 #
1080 EQ 60  &    AC  &  AMPS   &    1 &   --  #  IV     &   0   & 27617   & 1310 #
1090    60  &    AC  &  VOLTS  &    1 &  REG  #  IV     &   0   & 17617   & 1310 #
1100 GE 25  &    AC  &  VOLTS  &    1 &  EXP  #  ED     &   1   & 17615   & 1310 #
1110    25  &    W   &  WATTS  &    1 &   --  #  ED     &   1   & 17615   & 1220 #
1120    25  &    V   &  VARS   &    1 &   --  #  ED     &   1   & 17615   & 1220 #
1130    25  &    W   &  WATTS  &LE  3 &   --  #  ED     &   2   & 17616   & 1220 #
1140    25  &    V   &  VARS   &    3 &   --  #  ED     &   2   & 17616   & 1220 #
1150    25  &  FREQ  &  HZ     &EQ  1 &   --  #  ED     &   1   & 17615   & 1220 #
1160 EQ 60  &    PF  &  PA     &LE  3 &   --  #  PV     &   0   & 17618   & 1310 #
1170 ENDT
```

FIGURE 1.

The Decision Tables expressed very complex product logic. As a result we were able to generate the bill of materials for any one of several million instruments. The cost of storing and processing bills of material was reduced by 10 to 1. We proceeded to try using Decision Tables on many other product lines and found considerable success if:

1. The product was made up of a variety of parts and assemblies used in different combinations.

2. There was a large variety of final products.

The work of capturing the logic would not be justified for a simple product line or for a single one-of-a-kind product.

After this success in the order entry - bill of materials application, we looked at the manufacturing instructions for making parts and inspection.

A business making industrial controls made extensive use of Decision Tables to convert a panel layout into the factory instructions for shearing, punching, and bending sheet metal. The enclosure design was captured in TABLE 2100 - BACK DIMENSION and TABLE 3010 - HOLE SPACING - VS on next page, Figure 2.

The system required a very large amount of logic for bend allowances, tool designation, etc. The logic was done in Decision Tables and then programmed. This system has been up and running successfully for over ten years.

```
2100 TABLE - BACK DIMENSIONS
2110C                        MATERIAL  BACK      BACK     FLANGE  MTG    ASSM
2120C ENCLOSURE HEIGHT       THICK     WIDTH     HEIGHT   WIDTH   HOLE   HOLE
2130     PH    &    PH    #   MT= &   BW=    &   BH=    &  BA= &  BB= &  BG= & GOTO #
2140 GE 10.00 &LT 28.00 # 0.090 & PW-0.16 & PH-0.16 & 1.36 & 1.11 & 3.42 & 3010 #
2150    28.00 &LE 56.00 # 0.120 & PW-0.25 & PH-0.25 & 1.42 & 1.07 & 3.38 & 3010 #
2160       -- &      -- #    -- &      -- &      -- &   -- &   -- &   -- & 2120 #
2110 ENDT
2120 PRINT: ERROR IN ENCLOSURE HEIGHT

3010 TABLE - HOLE SPACING - VS
3020C                       NUMBER
3030C ENCLOSURE HEIGHT      OF HOLES
3040     PH    &    PH    #  BD=   #
3050 GE 10.00 &LT 15.00 #  2     #
3060    15.00 &   31.00 #  3     #
3070    31.00 &   45.00 #  4     #
3080    45.00 &LE 56.00 #  5     #
3100 ENDT
```

FIGURE 2.

In a computer business, the quality control people had set up a punched card system for recording the type of defect on a circuit board, but found that manual analysis of the data required extensive time and had many errors. They used Decision Tables like TABLE 4100 - OPERATOR ERROR, Figure 3, next page.

Their comments were that the program runs and is easily expandable, if necessary.

Thus, we have a number of cases where complex logic was expressed in Decision Tables and used to control a computer.

In several businesses as diverse as an insurance company, a construction company and a builder of nuclear power plants, payroll and labor cost analysis logic has been done in Decision Tables. One such table selects the correct tax table depending on marital status (M=married; S=Single), and pay requency.

WK = weekly

BW = bi-weekly

SM = semi monthly

MN = monthly

TABLE 2880 - MARITAL STATUS, PAY FREQUENCY, Figure 4, shows the Decision Table.

The single monthly federal tax table would be like TABLE 4380 - SINGLE MONTHLY FEDERAL TAX, Figure 5., where:

FED is the federal withholding tax

TXGR is the taxable gross income

FE is the number of exemptions

```
4100 TABLE - OPERATOR ERROR
4110C DEFECT
4120C CODE    TYPE OF OPERATOR ERROR
4130  DEFC #  ND  &  NE= &  NF= & NG= & NX=  & GOTO #
4140 EQ 28 # ND+1 &   -- &  --  &  -- & NX+1 & 4200 #
4150    29 #   -- & NE+1 &  --  &  -- & NX+1 & 4200 #
4160    30 #   -- &   -- & NF+1 &  -- & NX+1 & 4220 #
4170    40 #   -- &   -- &  --  & NG+1& NX+1 & 4200 #
4180 ENDT
```

FIGURE 3.

```
2880 TABLE - MARITAL STATUS, PAY FREQUENCY
2890C MARITAL  PAY
2900C STATUS   FREQ
2910    MST   &  PF   # GOTO #
2920 EQ M     &  WK   # 3010 #
2930     M    &  BW   # 3180 #
2940     M    &  SM   # 3320 #
2950     M    &  MN   # 3510 #
2960     S    &  WK   # 3770 #
2970     S    &  BW   # 4020 #
2980     S    &  SM   # 4200 #
2990     S    &  MN   # 4370 #
3000 ENDT
```

FIGURE 4.

```
4370 FINC=TXGR-62.50*FE
4380 TABLE - SINGLE MONTHLY FEDERAL TAX
4390C FEDERAL          BASE       DEDUCT        PERCENT-
4400C TAXABLE          TAX        AMOUNT        AGE
4410C INCOME
4420          FINC  #    BT=   &   DAMT=    &  PRCT=   #
4430 LE    142.00  #    0.00 &    0.00    &   0.00   #
4440       329.00  #    0.00 &  142.00    &   0.16   #
4450       621.00  #   29.92 &  329.00    &   0.18   #
4460       788.00  #   82.48 &  621.00    &   0.22   #
4470       954.00  #  119.22 &  788.00    &   0.24   #
4480      1288.00  #  159.06 &  954.00    &   0.28   #
4490      1538.00  #  252.58 & 1288.00    &   0.32   #
4500         --    #  332.58 & 1538.00    $   0.36   #
4510 ENDT
4520 FED=(FINC-DAMT)*PRCT+BT
```

FIGURE 5.

Another problem is the different union deductions. A table can be written giving each deduction depending on the union involved. These tables show the logical relationship so clearly that changes are easy to make.

The conclusion, from the twenty years of experience with Decision Tables, is that they are very useful in writing complex logical relationships. They are, in effect, a powerful programming language which is useful in many functions of a business. Several users have reported a doubling or tripling in the productivity of their programmers.

Based on this experience, MUSCL - a Decision Table language was developed. The best ideas from five other languages were used, READ and WRITE commands for file processing and a REPORT writer to format reports, were included.

MUSCL has been implemented on a Hewlett-Packard 3000 to do payroll, order entry, and executive compensation system.

# Microprocessor Based Product Design on an HP3000

a Presentation for the
7th International Meeting of the
HP General Systems Users Group
Denver Colorado, November 1978

by
Jack C. Armstrong
Los Altos Research Center

Hardly a day passes without the announcement of a new product or new application based on micro-processors. Concurrent with these announcements are constantly reducing prices for microprocessor hardware. Unfortunately, as applications become more complex, the cost of software development has soared.[1]

Microprocessor manufacturers were quick to respond to complaints about the high cost of developing products around their devices, by introducing microprocessor development systems. [2] These systems, which were valuable in debugging both software and hardware, were immediately placed into wide-spread use. Not unreasonably, microprocessor vendors offered development systems tailored very specifically to their own products, which as a not-coincidental side effect locked their customers into using their chips to the inclusion of other vendors. This effect increased as the cost of the development systems increased. Recently, a few development systems have been introduced (not by microprocessor vendors!) which accommodate several manufacturers lines, but these systems often lack features available in the vendor's custom systems.[3]

Another response to the need for software development aids was the development of so-called "cross software", which allowed the use of a larger, or at least different, computer to compose and assemble/compile programs for microprocessors.[4] This software was made available on all major commercial time sharing services, and organizations with in-house computer systems purchased cross assemblers and cross compilers for use on their own machines. The trend towards cross software has been accelerated by the rapid acceptance of higher level programming languages with the need for larger compilers. The use of another machine for software development has much to recommend it, including freedom to choose from many different microprocessors at minimal incremental expense, use of a common, and usually better, text-editor, etc. The snag in this process comes when the software is written and compiled without syntax errors...now comes actual debugging of the program logic. This, in addition to check-out of the prototype hardware, generally requires transferring the software to a microprocessor development system, with its capabilities for in circuit emulation, hardware break-points, single stepping thru instructions, etc.

Using the cross software approach frequently means a cost savings in the purchase of development systems, both because fewer of them will be necessary, and the less sophisticated models (without text editors, assemblers, compilers, etc.) may be used. However, it is rare that development systems of some sort will not be required.

One company which found themselves involved in developing a product around a microprocessor chose to stand back and look at the entire process of developing the product, and all that that entailed.[5] This was Scientific Micro Systems, of Mountain View, California. The following steps were identified in the complete project of developing, testing, and supporting a new microprocessor based product:

1. System specification and functional descriptions.

2. Detail design specifications.

3. System implementation, (hardware and software), including construction of prototypes, coding software, etc.

4. System test and evaluation.

5. System documentation, including technical reference manuals, user's guides, manufacturing instructions, sales literature, etc.

6. Product support and maintenance.

The system which evolved at SMS utilized an HP3000, several microprocessor development systems, and a prom burner, all integrated in one system for the development of a microprocessor based product.[6] The total system involved the use of a sophisticated text editor and word processing system (LARC EDITOR/SCRIBE) which included features designed to aid program development, documentation, and tracking changes in both source programs and associated literature.[7] Cross assemblers and cross compilers were implemented on the same system, which produced object code files on the system disc. These files were then "down loaded" onto either the microprocessor development system (for testing) or onto a prom burner to fuse proms for the completed system.

As consultants to SMS, we developed two programs, one for interfacing to the development system, and the other to interface directly to a prom burner. These devices were plugged into HP3000 RS232 terminal ports, and interface programs written in SPL. Once developed, the interface programs were not terribly complex, but the thought and design took considerable effort.

Two approaches were considered in attaching the development system to the HP3000. The first would be to connect an RS232 port on the development system to a terminal port on the HP3000. This would require software in the development system which would allow a user on the development system console to talk "through" the development system to the HP3000, which would then be tricked into thinking that the development system is a user terminal. A user could then log onto the HP3000 and run software to access the object files created by the cross software and transmit then back to the development system. At SMS, the development system being used had its entire operating system in firmware, so no modification was possible. Instead, we chose to fool the development system rather than the HP3000. This was done by connecting the development system console line to a dedicated terminal port on the HP3000. Users may utilize any terminal to log onto the HP3000, and then run an interface program which reads the terminal, looks for a few special commands, and otherwise sends any line typed directly to the development system. The special commands cause disc files (of object code) to be transmitted as though they were being entered by hand directly into the development system. Thus, the development system thinks it is talking to a user console, not an HP3000. (A user who always types at 240 characters per second!).

In other, more flexible development systems, either technique could be used, and each have some advantages. In the case of the prom burner, we again made the device think it was attached to a terminal, not a computer. This program is actually more complex than the interface to the development system, as the prom burner may be both read or written, and we incorporated a facility to compare two blocks of object code, compute check sums, fill unused addresses with a user selectable constant, etc. Also, the program must deal with proms of several different sizes and types.

The success of this installation led to another company acquiring an HP3000 for similar purposes. Caere Corporation, also of Mountain View California, interfaced not only development systems, but single board computers (SBCs) to terminal ports on their HP3000. The SBCs actually have very simple operating systems in firmware which allow down loading via an RS232 port, so programs on the HP3000 can load and run test programs on prototype devices, which then transmit back test results for later analysis on the larger machine. They are using the same text editor and word processing system as SMS, and for similar purposes. They also use cross software, but have a rather unique twist to their use of a development system. One high level microprocessor language currently has a more efficient compiler implemented as a resident compiler on the development system than those available as cross compilers on larger systems. Caere intends to use the HP3000 for text editing and compostion of source programs, but will then ship the <u>source</u> to the development system for compilation!. The resulting object code will then be transferred back to the HP3000's disc, for subsequent down loading onto SBC's connected to other ports. Thus, the development system becomes a "compiling box", and rarely if ever will be used for anything else.

The advantages of directly interfacing to microprocessor devices, via standard RS232 interfaces, are that each component of the complete system is doing whatever it is best at, with a maximum of flexibility for future changes. The major improvement in this scheme would be to upgrade the cross compilers available. Ideally, one should be able to chose from several high level languages to develop software with, and <u>then</u> select the optimum microprocessor for the application. This selection would be considerably aided by a choice of code emitters all working off the same compiler, so code density and instruction counts could be compared. Finally, a direct interface to a development system would allow final testing of the competed system. The HP3000 has proven to be an ideal nucleus for such a total system, and most of the software is available today.

---

[1] McDermott, Jim, "Experts Tell How to Hold Down the High Cost of Microprocessor Programs", Electronic Design 26, vol. 23, (Dec. 20, 1975) pp20-26.

[2] Bursky, Dave, "It's Getting Easier to Program Microprocessors With the New Software Design Aids", Electronic Design 2, vol. 25, (Jan. 18, 1977) pp20-26.

[3] Snigier, Paul, "Microprocessor Development Systems - Which One is 'Best'", EDN, vol. 27, no. 5, (Mar. 5, 1977) pp68-78.

[4] Nauful, Eli S., "Software Support for Microprocessors Poses New Design Choices", Computer Design, vol. 15, no. 10, (Oct. 1976) pp93-98.

[5] Ivie, Evan L., "The Programmer's Workbench -- A Machine for Software Development", Communications of the ACM vol. 20, no. 10, (Oct. 1977) pp746-753.

[6] Auclair, Dan, "A Minicomputer-Based Microprocessor Development System", Compcon 78 Spring, 16th IEEE Computer Society International Conference, pp334-337.

[7] Rochkind, M. J., "The Source Code Control System", IEEE Transactions on Software Engineering, SE-1, 4 (Dec. 1975) pp364-369.

SYSTEM PERIPHALS

Series "H"

|  |  |
|---|---|
| PRESENTATION TITLE: | UT200 RJE Subsystem |
| INDIVIDUAL(S) NAME(S): | Donald Klett |
| ADDRESS: | Sangamon State University<br>Springfield, Illinois 62708 |

ABSTRACT:

A 200 User Terminal (UT200) simulator has been developed for the HP3000 Series II that supports remote job entry service to large Control Data Computer systems. The design and implementation of the subsystem will be presented. Input/output options and the user interface will also be discussed.

# HP 3000/IBM 1403 COUPLING EXTENDS SYSTEM UTILIZATION

RAY LORENZ
SPUR PRODUCTS COPR.

The HP series 33 gives the user the option of a 180-characters-per-second or 400-lines-per-minute dot matrix printer. Series II and III give choices of solid-character printers with speeds of 300, 600 or 1250 lines per minute. Being solid character printers they have advantages over dot matrix printers. However, all three are drum printers which lack operating features and options inherent in chain or train printers.

In fact, a chain or train printer can greatly expand the utilization of any data processing system compared with a drum printer. For example, an operator using a chain or train printer can quickly interchange character sets, and even individual characters, so that a number of fonts (even custom-designed type faces), special symbols, numbers and letters can be selected. Furthermore, printing quality is greatly improved, a factor that also makes it usable for some functions not desirable for a drum printer.

Spur Products Corp. has developed a controller that makes an IBM 1403 printer plug-compatible with the series II and III computers. (Figure 1) It replaces any of the three drum printers offered by H.P. The IBM 1403-2, or -3 or -N1 can be driven. Rated print speed of the -3 and -N1 is 1100 lines per minute, and that of the -2 is 600 lines per minute. Of the five models of 1403 printers available, we selected these three for use with the HP 3000 because they have 132 print positions.

The 1403-series printer was selected for this marriage because it can correctly be called the best item of electro-mechanical equipment ever produced. As such, dependability and versatility also are among its attributes.

A comparson of the operating features can best be understood by comparing the mechanical features of the drum and the chain or train printer:

In the drum printer a cylinder rotating at constant speed has a complete set of printing characters embossed around it at each print position. Hammers strike the desired characters as they rotate into print position. (Figure 2)

While a limited selection of fonts on the drum is available from the manufacturer, the drums cannot normally be changed by the operator. Furthermore, individual characters cannot be changed under any conditions. This not only completely takes away an option available with train or chain printers, but makes it necessary to replace the entire drum if a single letter is worn or broken.

Furthermore, no drum printer has been able to time the impact on the rotating drum at the precise moment that the character is in position. The result is that printed lines are wavy across the page. The HP printers, which actually are made by Dataproducts, are not as bad as some in this regard, but they also will go out of adjustment. Wavy lines discourage the use of the printer for sales solicitation letters or most printed communications that reach the public. Printed communications are often the only media on which the company is judged, so they should have the highest quality possible. It is natural that the 1403 is the standard of the direct marketing industry.

In such applications two other characteristics of the 1403 are definite economic advantages:

1. Paper is slewed from the bottom of one form to the beginning of the next at approximately 80 inches per second.

2. A vertical format unit can enable the paper to be advanced to any pre-selected position, such as slewing from the bottom of one form to the top of the next. The carriage control commands are punched into a paper forms control tape built into the printer.

In a chain or train printer an array of character slugs moves horizontally at constant speed past a set of hammers--one hammer for each print position. In chain printers the slugs are connected and pull each other around a track. In train printers the slugs are not connected and push each other around the track. (Figure 3) The model 1403-2 uses a chain and the 1403-3 and 1403-N1 use trains.

Our respect for the 1403-series printer is so great that we chose it as the printer to be driven by the Spur controller even though it has two marketing disadvantages for us:

1. It has been out of production since 1970. There still are hundreds of thousands around, but the market obviously is not growing, and any manufacturer likes a growing market.

2. Model 1403 printers cost as much as $30,000, which

is only about $8,000 less than it cost when it was new. Its cost per printed line is about the same as the latest model IBM printer. Still, you have to wait at least three months for delivery of a reconditioned 1403 that may be 20 years old.

Whenever there is an unusual computer printing job it invariably is done with a 1403. One company uses it to print on mylar for labeling products stored outdoors. The Atomic Energy Commission uses it to print its Nuclear Science Abstracts because it has the print quality that allows the copy to be read after it is reduced photographically and printed by offset lithography. Both the AEC and Stanford Research Institute use it because special chemical and mathematical symbols can quickly be added to the print train. SRI even creates Greek and other foreign letters by overprinting standard characters. In all of these applications, incidentally, the IBM 1403 printer is used with a computer other than an IBM.

The Spur controller is software-compatible with programs supporting the HP 3000 series II or III. To accomplish this while making it possible to permit the operator quickly to interchange character sets it was necessary to incorporate a memory in the controller that makes the formatting adjustments necessary for each type train.

A random access memory enables the operator to table load the memory. To make reprogramming as easy as possible when a single character is changed, the Spur controller memory has positions for 240 characters. Each has a direct relationship to the 240 character positions on the train (normally 48 characters repeated five times around the track for speed in aligning the chracter with the print position).

Optionally, the program can be permanently stored at the factory in programmable read-only memories. When this option is selected three PROMs are included with the controller, one for each of three print trains selected by the user. A front-panel switch enables the operator to change the program when the train is changed. Any number of additional PROMs can be ordered for as many trains as the user plans to use.

Naturally if PROMs are requested the operator is relieved of the problem of programming, but does not have the option of changing the computer language at a later date, which would be possible with table loading.

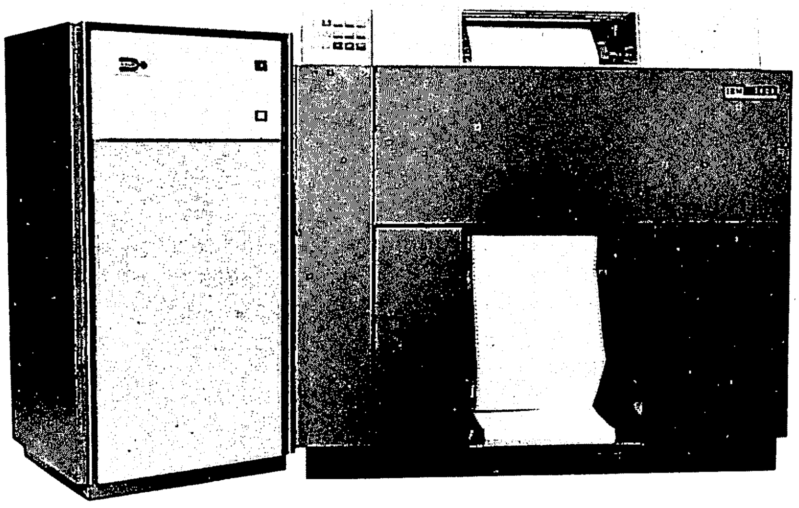
Even without considering the increased use of the system made possible, costs of a 1403 printer and Spur controller system are competitive with similar HP equipment. An HP 2618A printer with a speed of 1250 lpm has a list price of

$35,400, and the HP controller adds $1,275 to the total price. An IBM 1403-N1 printer costs somewhere between $20,000 and $30,000 and the Spur controller costs $17,500.

When comparing costs it is important to recognize that an increase in utility affects the comparable economics. If, for example, sales letters now can be printed with a 1403 printer while they previously could not and had to be sent to a service bureau, the comparative costs would have to be re-done, especially if the additional printing job can be done when the computer otherwise would be idle.

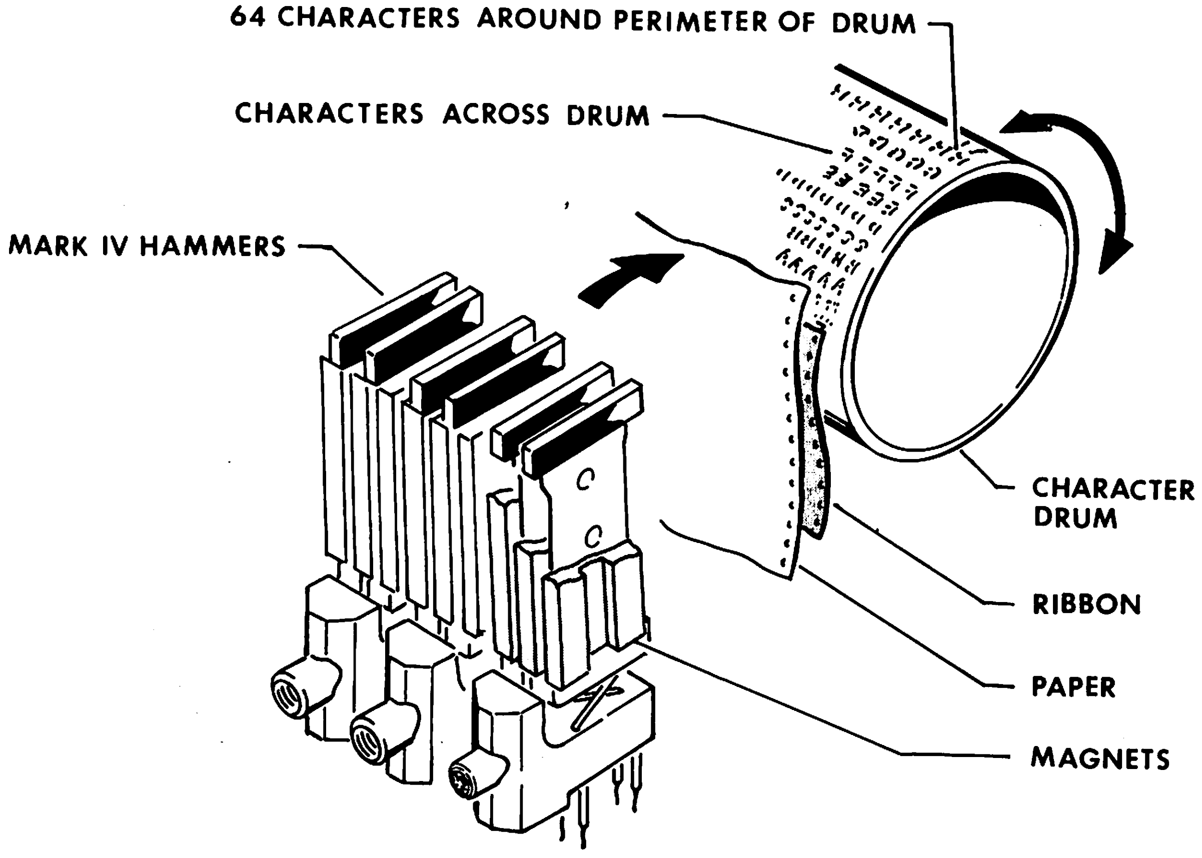
## CONCLUSION

Just as it can truthfully be said that the HP 3000 "ends the computer compromise," so the Spur controller ends the printer compromise. In some ways the printer is the most critical part of the data processing system. It produces the final product of the system. Therefore, its printing quality is the standard by which some unsympathetic boss, not to mention the public, may judge your entire effort. And most important, the printer can be a limiting factor in the variety of jobs that can be done. For these reasons some operators of HP 3000 general systems have welcomed this new product, and many more are expected to join them.



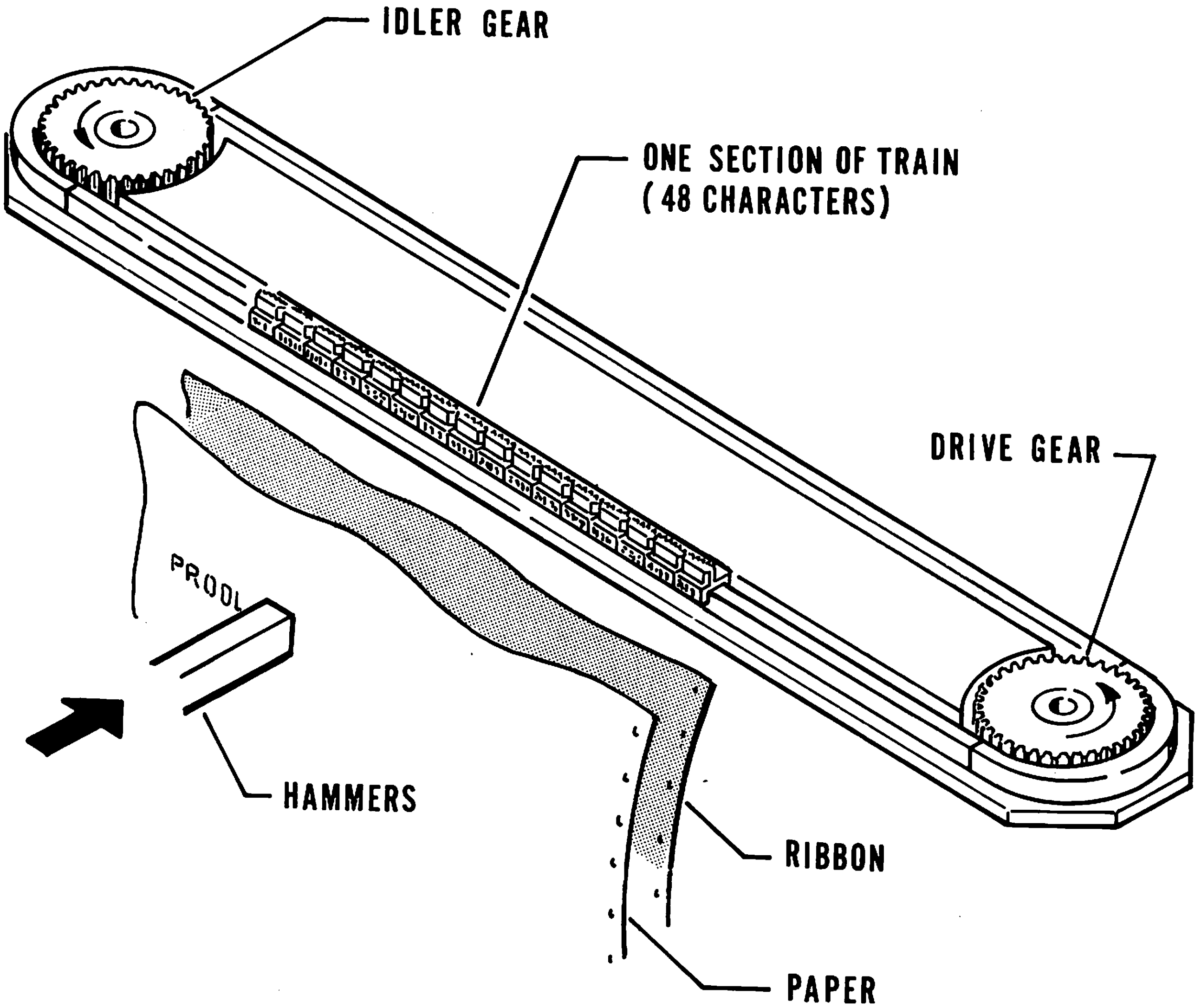
Spur Controller    IBM 1403 Printer

FIGURE 1



DRUM PRINT MECHANISM

FIGURE 2



IBM TRAIN PRINTER MECHANISM

FIGURE 3

PRESENTATION TITLE: A Real-Time Instrument Interface System for the HP3000

INDIVIDUAL(S) NAME(S): Gordon R. Symonds

ADDRESS: Environmental Health Centre
Tunneys Pasture, Ottawa, Ontario

ABSTRACT:

A real-time, terminal-oriented interface which permits acquisition of data from laboratory instruments and other real-time devices is described. The terminal/interface system can be used with any device having a serial ASCII output port, either RS232 or 20 ma current loop. It performs control functions, baud rate conversion, etc; in addition to serving as a normal session mode terminal. Current applications include automatic thermoluminescent dosimeter (TLD) readers, blood analysis equipment and an animal weighing system.

The system has been used in production for over one year, and has proven to be a low cost, easy-to-use solution to interfacing instrumentation to the HP3000.

## DISK

## SUBSYSTEMS

## SOFTWARE

## CONSIDERATIONS

"Foreign trade" enhances the quality of life of any nation. In similar fashion, "foreign devices" attached to a computer's central processor unit (CPU) can often enhance the performance and cost effectiveness of the computer installation.

In both cases, protectionist attitudes can limit the potential benefits. In the world of computers, the first instinct of the user is to protect his system software. The concern is legitimate. Investment in software can easily exceed the value of computer hardware.

System software has been designed, in most cases, to operate within the context of a specific hardware configuration. Even minor alterations in the hardware characteristics can have far-reaching and often unpredictable effects on the operating software.

Yet the fact is that nearly every computer installation is limited not by the capabilities of its CPU, but by the throughput and capacity of the input/output and mass-storage devices attached to the CPU. The performance of nearly every computer installation can be enhanced, therefore, by taking advantage of state-of-the-art advances in the design of these peripheral elements.

But this is acceptable <u>only</u> if the new equipment is "transparent" to the existing softward. Neither the CPU (for technical reasons) nor the user (for emotional reasons) should be disturbed by the switch to a new type of I/O or mass-storage device.

## Software Transparency

These comments apply to any type of equipment attached to a computer--including terminals, printers, tape transports, data-communication lines, and disks.

This paper will concentrate, however, on disk drives--for several reasons. Disks represent the most economical method for providing fast-access mass storage. Moreover, since disks are usually treated as an extension of main memory, they are intimately linked to the CPU and its operating softward. And if these facts were not enough to give pause, disk technology has been progressing at a very rapid rate during the past few years.

The challenge, then, is to realize the potential of the new technology (e.g., the new 3330 and Winchester drives) and still remain "software transparent." Techniques must be found to attach an advanced (but "foreign") drive, without altering a single instruction in the operating system software or application programs.

Three different methods have been developed to accomplish this objective:

a) Fixed plug-compatibility

b) Dynamic plug-compatibility

c) Virtual transparency

As one of the industry's leading suppliers of disk systems for computer enhancement, CalComp uses all three techniques for achieving software transparency. Each method has its place -- depending on the type of system and the volatility of the hardware and software.

## Fixed Plug-Compatibility

The original technique was fixed plug-compatibility--dating back to the 1960's. It was, in fact, the basis for the first effective penetration by independent disk suppliers, such as CalComp, into the monolithic IBM marketplace.

Small but significant changes in the mechanical design of disk drives provided major improvements in reliability and ease of maintenance--plus a modest increase in system throughput.

Despite these advantages, however, the new products could not have found a market unless they appeared (to the CPU) to be identical to equivalent products offered by IBM.

The "foreign" drives were provided with cables that could be plugged directly into the IBM mainframe. They also had electronic logic circuits that could respond, with absolute fidelity, to IBM's disk-drive commands.

But the fixed nature of the interface was a severe handicap. Independent disc suppliers were inhibited from developing performance characteristics beyond those that could be controlled by the IBM disk protocol. They would be, by definition, "transparent"--and of no practical value. Suppliers had to wait for IBM to make the improvements, and were therefore in a constant state of catch-up.

Of much greater concern was the fact that IBM could, at any time, make minor changes in its own operating software. These could render, overnight, all "foreign" disks inoperative. Sometimes the changes could be anticipated and allowances made in the control circuitry. Just as often, the only solution was an emergency retrofit of existing units in the field.

Despite these difficulties, the cost savings and performance benefits were sufficient to create a viable plug-compatible market for independent disk suppliers. Their sales grew at an accelerating pace and soon extended beyond IBM to systems produced by other mainframe and minicomputer manufacturers. But the vulnerability to change has remained as a constant threat to both users and suppliers.

## Dynamic Plug Compatibility

One solution to this problem is what can be referred to as dynamic plug-compatibility. This is the capability of "mapping" or making one type of disk look like another type which is recognized by the host system.

Two factors have contributed to the development of this concept. Of major importance, of course, has been the introduction of LSI and microprocessor circuitry. A microprocessor can be readily adapted to the type of control functions required in a disk interface. Equally important has been the expanding role of mass storage facilities as the principal method for enhancing the efficiency and throughput of computer installations. Disk storage facilities, taking advantage of new, high-capacity drives, have grown to the billion-byte level--as a starting point.

In addition to size, there is also a new emphasis on variety. Disk facilities may include drives with removable or non-removable media with fixed or moving heads. Small-capacity units may be added for private files, while maximum-capacity units serve as basic storage to provide a minimum average cost per byte. The configuration of the facility may, in fact, change from month to month, or even hour to hour.

With the new emphasis on size and capacity, the additional cost of a dynamic, microprocessor-based disk interface can be easily justified. And as an added bonus, the interface can be easily altered to meet any changes in the disk hardware or operating system software.

The functions of a dynamic plug-compatible interface are dictated by the microprocessor program--stored in easily interchanged PROMs or floppy disks attached directly to the interface controller. Or the operating system itself can define its own plug-compatibility by downloading a suitable interface program at the time of system generation. Response to changes can be, for all practical purposes, instantaneous.

## Virtual Transparency

Fixed plug-compatibility is still the most direct method for achieving software transparency in applications where the system software is static and little advantage can be gained by changing the mass storage facilities. Dynamic plug-compatibility can be justified when the storage facilities are large, or when there is a high degree of volatility in the system software and physical makeup of the storage facility.

In both of these cases, an assumption is made that the "plug-compatible" device is recognized by the existing system software. But this leaves a third situation in which neither technique is truly applicable. For example, the system software may have been written without any provision for the newer types of disk drives (e.g., 300 MB drives with a 1.2 MB/second transfer rate). There is, therefore, no "plug" to be compatible with. Situations could also exist in which the hardware and software are evolving at a rapid rate, yet the scope of the mass storage facility cannot justify the use of a dynamic, microprocessor-based "mapping" approach.

In both of these instances, a technique which can be referred to as "virtual transparency" can be an effective solution. CalComp is using this method to interface a variety of different capacity Trident disk drives to CPU's produced by many different minicomputer manufacturers, some offering many distinctly different operating systems. All of this is accomplished, moreover, with a microprocessor-based disk-controller design--produced in volume and thoroushly tested by hundreds of successful applications in the field.

## A Logical Answer

Virtual transparency can best be understood in terms of its origin: Virtual memory. Originated by IBM and now adopted by most minicomputer and mainframe operating systems, virtual memory was developed as a way to free programmers from any need to allocate and keep track of the computer's memory resources. Application programs could be written in abstract terms. The computer's operating system would translate "logical" virtual-memory addresses into physical addresses--taking advantage of any memory space available.

As programs grew in size and larger volumes of information were processed, much of the data (including application programs and portions of the operating system itself) were transferred to mass-storage devices like magnetic tapes and disks. But these were treated as I/O peripherals, and soon the computers were spending a majority of their time on the transfer of files between mass-storage and memory.

The virtual-memory concept again came to the rescue. Just as the use of logical addresses, independent of physical locations, simplified the life of the programmer, an extension of the virtual memory technique to mass storage devices served to relieve the main operating software from an equivalent concern for the physical configuration of the system. Subsidiary modules could accomplish the necessary mapping and address translations.

The benefits are manifold. The programmer and his application program can ask for data stored at a logical location. The executive portion of the operating system passes the request along to the appropriate memory-management module. The subsidiary software determines whether the requested data is in main memory--immediately accessible to the application program--or is remotely stored on disk or tape. If the latter is the case, the transfer can be initiated by the lower-level software while the operating-system executive moves on to other, more demanding tasks.

## Device Handler

The first step in the transfer operation would be for the memory-management module to pass the physical address along to an even more subsidiary software unit: the device handler for a specific disk-drive controller. Only now, three steps removed from the application program and two steps removed from the main body of the operating-system software, would the address request take a form that relates to a specific, physical device.

Virtual transparency takes advantage of the fact that there are, in truth, two interfaces between the disk storage and the system software. One is the physical interface at the plug connecting the disk-drive controller with the CPU hardware. The second is the software transition between the virtual addresses of the system software and the physical location of the stored data.

Either one of the interfaces can be used to maintain "software transparency." The plug-compatible techniques use the physical interface. CalComp's virtual-transparency method takes advantage of the modular structure of nearly all operating systems. When the memory-management module "calls" for a specific device-handler module, it can just as easily invoke a CalComp-supplied segment of software as one supplied by the computer manufacturer. In neither case is the main body of the system software affected. Not a single line of the existing application programming must be changed. Yet the user has the advantage of the latest technology disks, with capacities that far exceed those of the largest disks anticipated by the designers of the original software.

## System Generation

The simplicity of the virtual-transparency concept is evident each time the operator "generates" the computer system for a specific group of application programs. The complete set of operating-system modules is rarely used. Instead, to conserve main-memory space, the operator invokes a system-generation program that allows him to specify only the modules required for the particular applications.

A CalComp-supplied module is included in his choice of options. The software itself is supplied on tape or disk, depending on the system configuration. The operator also has a sheet of load instructions, written in exactly the same format as that used by the computer manufacturer. At an appropriate time in the system-generation procedure, the operator loads the CalComp device-handler module--or leaves it out. Moreover, if there is ever a problem with the CalComp disk hardware or software, the disk-controller cable can be simply unplugged and the system regenerated without the CalComp module.

## Summary

The true test of "transparency" is whether a foreign device can be added to a computer system to enhance its performance and capabilities without affecting, in any way, the user's investment and confidence in his operating software and application programming.

At least three different methods can be used to achieve this result. CalComp has used all three techniques to enhance both existing and new computer installations.

SPECIAL

Series "I"

# ORGANIZING A LOCAL GROUP OF COMPUTER USERS

Douglas J. Mecham
Hughes Aircraft Company, Ground Systems Group
P.O. Box 3310, Fullerton, California 92634

## Abstract

This presentation concerns itself with the organization, development, and management of local/regional user groups in support of communication among computer system users. Additionally, considerations for the relationship among RUGs, with the HP General Systems Users Group, and with Hewlett-Packard is also presented. The objective of a local users group is to promote the active interchange of ideas, techniques, and software among the users. This interchange can take place with only two people. The topics discussed below are some ideas that may be used to promote the success of users in a local group.

## A Simple Approach

Undoubtedly you have one or more friends that you confer with regarding your computer system problems. If you and your computer friends have met over lunch to discuss a computer technique you have the beginnings of a local users group. Most likely you will return to work with a new piece of software or computer technique to try out. Let us identify the basic aspects of such a users' meeting:

Meeting Arrangements--one of the group called the others and invited them to join him for lunch.
Meeting Opening--The objective of getting together was, in some fashion, stated and any new persons were introduced.
Technical Discussion--One of the group explained how he had solved a system problem and discussed associated software used. As the discussion progressed questions were asked and comments made by the others in the group. During the discussion reproduced copies of the software listing were handed to the group.
Computer Products--Since each day a new product for the computer market is introduced, one of the group discussed his research of the new product that may be useful to the others.
Social Hour--After all the involved technical material the group relaxed over coffee.
Next Time--Because members of this group had several other computer problems they decided to meet again to share more information.

As you can tell by the description a local users group meeting took place--simply and easily.

Needless to say with more and more users becoming involved with HP systems such a small meeting could rapidly grow. The basic meeting elements will not change but a little more planning and organizing is required. A meeting of computer users need not constitute an official group; but it is useful to have a spokesman to convene the group and communicate with the larger users group. The following paragraphs present the considerations involved in organizing a group of local computer users.

## Orientation

The object of such a computer users meeting is to share one's own knowledge and experiences of computer systems with others who may not yet understand the system. This, of course, provides self-esteem, a key psychological requirement in a group. Also, such sharing certainly provokes others to share their knowledge and experiences. Additionally the presentation of two items of information often times results in a synthesis providing a third and different item of information. For example, development of a file handling routine by one user may be put with another user's indexing routine to form a data management module. Through the users meeting, a user may discover a simple solution during a brief discussion with another user that he had been battling with for a week.

The collective of computer users can often provide the resources necessary to solve a problem not easily solved by an individual. For instance, comparison of tests performed with different data communication equipment or under different situations may yield enough data to solve a difficult problem. Or, you may just need the ideas from several other users to formulate a solution to your problem.

$\longrightarrow$ *[A users group does not make sense without user involvement]*

Likewise, individuals in the group may be able to contribute a mail listing program for distribution of mail to the group members. Economically, the formation of a group of users makes sense because software, an expensive item to develop, may be shared. An objective of a group may include one or more services such as publication and distribution of a newsletter and/or maintenance of a software library. In particular, a formalized users group can often arrange for technical experts to make presentations to the group as well as sponsor training seminars.

When a local group of users relates to a larger group such as RUGs to HPGSUG, it is important frequent communication take place between them. The larger group may be able to find solutions for users in your group who have problems. Certainly the larger group provides a larger base for sharing software. Such a relationship is a two way street, however, and contribution by your local group to the larger group is necessary. For instance, a local group could sponsor a technical session at an international meeting. By spreading such a work load the larger task is accomplished without over burdening any individual.

Thus, the formal orientation of a group of users is any objective which, by definition, must meet their needs. Before describing the technical aspects of organizing a group of users consider the individual user who would make up such a group.

The User

The objective for most users of computer systems is to solve their problems and enhance their software/hardware to perform more efficiently with greater capability to do their job. The success of a users group is dependent upon understanding the memberuser's perspective; the perspective is to solve his problem in a simple expedient fashion. If a users group can meet this requirement both the user and the group will be a success.

The contribution a user makes to a users group depends upon his participation. Certainly a user's expertese and experience will allow him to participate easily in technical functions. For the inexperienced user participation may be in the areas of administration and support of the group functions, such as maintaining mailing lists. Together a users group can function. The inexperienced users will advance to the experienced levels, the experienced to more sophisticated problems, and new users will join at the inexperienced stage. As professionals we have an obligation to contribute; each user taking a small responsibility can produce a useful benefit for all users.

The most important user attribute to pay attention to in a group is the user's attitude. Since most groups of users depend upon volunteers to keep the group active particular attention needs to be paid to keeping the atmosphere around the user's involvement positive and self-satisfying. The dedication of a user varies over a wide range as does his sensibilities. Recognizing these traits and managing them can be an asset. For the most part this involves common sense, responsiveness, common courtesy, being observant, and giving recognition.

A user's expectation is easier to meet in a volunteer organization than in a commercial organization. In the former the user is usually intimately involved and the destiny of his expectations is a function of his own effort. In the latter the user always expects more when he is paying money and is not involved in the "product".

Watch out for users who promise a great deal but contribute little. It is difficult to prevent users from taking advantage of a volunteer group for their own ends; on the other hand, there are most likely a few individuals who contribute a great deal. While it is impossible to satisfy the latter's desires or repay them, public recognition is most fulfilling. Psychological satisfaction is often an important and satisfying reward.

→*[Apathy is the worse enemy of a volunteer group.
Enthusiasm is a group's greatest asset.]*

Who should be included? Too often with larger computer systems only the computer center specialists get involved. While they have an indepth enthusiasm the users at the terminals should not be overlooked. A special local users group program for clerical personnel who interface with the computer might help dispel "computer fears." The "upper" management who must make key decisions regarding your system might be interested in a limited involvement to better educate themselves in the way of the computer system as well as meet other managers making similar decisions.

Now that some awareness of individual users has been brought out and objectives made clear where does one start to organize a group of computer users?

<u>Where/How to Start</u>

Why organize? It would be nice if the interface of all people just fit together to arrive at the desired goal. Since this is not the case in the real world someone needs to plan the sequence of events to arrive at that goal. Any organization is predicated on an agreed-upon group objective. This may come about through an informal conversation with a few friends with a common product or application interest who decide to interface among themselves in the future. The user interface may manifest itself through a meeting, newsletter, software library, or a combination of each. The user meeting is one of the easiest to start with since people like to socialize. The particular subject matter may be around a special interest such as word processing or a particular vendor product, or both.

An effective method is to get a few "key" friends involved who you can count on to at least carry out an initial meeting of users. Make sure the communications between the key group is free and easy since a time lag and a hinderance of communicatiions can postpone activity and dampen any enthusiasm. The next item of business is to establish communications with other users.

A first meeting at your company or local hotel facilitates easy arrangements and control. By organizing the basic meeting aspects described in the opening of this presentation your meeting will be a success. Depending upon how ambitious the key group is the meeting advertisement may be by word of mouth or a formal, mailed, invitation.

$\longrightarrow$*[Frequènt user meetings over lunch are easy and effective.]*

If the meeting sparks some interest and a definable need to interface among users in the future exists then a more formal group may be organized; although, several meetings may take place prior to serious organization of a group. A key to a successful first meeting is to arrange for all attendees to return home a "winner," that is, make a contribution to the user's future success. Such a "prize" may be a software program or inside information on a vendor's product. Before the first meeting has adjourned the next meeting or user interface mechanism needs to be planned and volunteers designated who will implement it, thus the next "key group" is formed. Note, if there are no volunteers either abort the group organization or be prepared to perform a GREAT deal of work.

For the first meeting keep expenditures and meeting operations to a minimum. To offset any dollar costs collect a donation at the door. An alternative is to find a sponsor such as a product manufacturer or user company whose interests are served by such a meeting. Most likely both clerical assistance and operations support, such as reproduction of notices and mailing, for the first meeting will come from the "key" group companies. Thereafter a more formal mechanism for funding and incentive for manpower needs to be found. Some ideas will be discussed later.

## User Education

One of the most attractive elements of a users group is its capability to educate users. Informal education is done through user meetings and technical publications; while this is rewarding the more organized/formal technical seminar is popular. Corporations seem to support the specific seminar since managers can easily relate specific seminar topics to their projects. Both the area of special application interest and computer system (or subsystem) product have their "gurus" who are willing to speak. The vendor usually has specialists who can present such a seminar. A users group could make the necessary arrangements while charging each attendee a required fee to pay the specialist for his preparation and any handouts. For example, a RUG on the West Coast sponsored a very successful seminar on HP3000 peripheral user maintenance given by Hewlett-Packard. This is a win-win seminar since HP benefits from more knowledgeable users and users can solve their problems faster. Such a seminar is an easy success.

Note that computer professions are hungry for computer knowledge. This need makes it easy for a users group to establish an educational objective. It just remains to focus in on a topic.

## Communications

During the formation and subsequent operation of a users group the most important element is communication. If those in the key group do not relate their problems and successes then plans are difficult to make, let alone adhere to. Likewise constitutants of the group probably will not respond if they do not receive some form of communication from the key group. In fact users who have contributed to the group who do not receive communication when expected can become bitter to the detriment of the group. If advice is requested in the communication then accepted when it is received individual cooperation is increased.

$\longrightarrow$ *[Communication is a terrific professional pacifier.]*

Individual cooperation in a volunteer group is valuable; the greater the individual member communication the greater chance for group success.

The information that must flow in a technical organization is tips, techniques, hints, kinks, along with meeting times and places. Communication needs to take place as often as necessary. This means if you communicate while you are thinking about it the link is made and progress can take the next step. A number of simple and easy acts of communication can build an effective network.

$\longrightarrow$*[Lack of effective communication has been the downfall of many groups.]*

Different requirements dictate different methods. For instance, the post card or note is easy and quick especially if mailing labels are already printed. Letters are more formal but do show more thought for a professional approach and they serve as a good audit trail so new ideas can build on past actions. A good secretary can make all the difference in the world in getting communication out.

The newsletter of course is a more formal but easy way to communicate with a number of users. However, a newsletter requires a dedicated volunteer and considerable user contribution. Special publications are very useful since they are specific in nature and usually treat technical material in-depth. Such a publication results from the dedication of a very few users but unlike the periodic newsletter it is usually a one-time task. Both of these methods of communication can be very effective and simple to produce. These publications may, however, be quite formal and require a complex publishing operation. The most effective newsletter/publication is not necessarily the "slickest," easy and simple techniques for production of these items is found in the literature [1].

The telephone is quick and easy for most users. While it is difficult to communicate many details over the telephone this instrument is very effective for directing a group of users by the key group and receiving status on activity. An automatic answering device can provide an easy method to communicate to the members and collect brief comments; such a device can be made available for communication 24 hours a day, unattended.

If the espectations of member users are to be met then mechanisms for user feedback need to be devised. This may take the form of a questionnaire or telephone campaign. Keep in mind that talleying responses and performing an analysis on the results may overwhelm the uninitiated; so keep responses very simple. Also, keep in mind that the number and accuracy of responses is directly related to the complexity of the questionnaire. Again, keep the questionnaire short and simple. It is better to have several short questionnaires than one long one.

If your group warrants it, effective communication may be supported by clerical assistance; it is well worth hiring a professional secretary part time for typing tasks. Of course, micro computers, simple printers, and an elementary text editing program can serve well as a secretary (except for spelling).

Handling the information flow and keeping it moving is important since newsletter editors thrive on new items and the key group needs the information to formulate plans to support the group. A central address and desk is useful for this purpose. Information must be appropriately redistributed in a timely fashion as well as just accepted. The formal information flow by mass mailings requires some serious thought. The key member responsible for such mailings needs to be aware of the considerable effort required to stuff, seal, address, stamp, and mail a large set of letters, not to mention printing. The cost of mailing can be optimized by planning the weight just below a postage price break and by using photo reduction techniques.

With the advent of word/text processing on micro computers the tasks of producing mailing labels, letters, etc. are easy. Even simple FORTRAN programs and simple text files on the HP3000 computer system can effectively support these tasks. Multiple, personalized letters, can easily be printed. Also, computer data communications can be used most effectively to get users to contribute articles; it is easy for a computer system editor to "type" an article. For instance, many HP3000 computer installations have a terminal dial-up facility. Thus, a newsletter editor could dial up such a system and copy the contributor's text file to a terminal tape cartridge or printer for subsequent use in producing a newsletter.

The local users group can perform a real service to its members by communicating with other local users groups and the international groups; the communication is two ways. You might consider exchange of newsletters and notices as well as software. Good seminar speakers can also be found this way.

## The Users Meeting

Computer users need to get together to discuss their successes and problems of their systems. In most instances a local group can have a very successful but simple meeting by considering the items discussed in the introduction of this presentation. Consider the aspects of a more organized meeting, although these aspects also apply to the small simple meeting where appropriate.

There needs to be a central managing committee to insure consistency and follow-through before and after the users meeting. Be very aware of the effort required by each committee member; each member should commit themselves in a formal manner as a warantee on dedication to complete the job. Make sure the meeting tasks are well defined and assigned. The magnitude of each task needs to be evaluated in light of the committment made by the committee member to perform it; the work effort is double that of any reasonable estimate.

→*[Large meetings supported by weak committees can turn a meeting into chaos.]*

Smaller informal user meetings can take place often at convenient times such as over lunch or in the evening. However, the larger full day meetings require planning a day when there is a minimum of conflicts with member users. Multiple day meetings should be avoided due to accommodation problems, competition for a member's time, and user saturation levels. The location of a meeting should be central to the group; however, special locations are attractive. Depending upon the budget and nature of the gathering meetings are easily arranged at hotels, schools, homes, or vendor facilities. The city to hold a users meeting is most likely obvious for a local group of users...except in Los Angeles.

Economics are always difficult for local groups since "seed" money is often lacking; asking for pre-registration fees or vendor contributions can be helpful. Group projects to raise money are also helpful. The larger the meeting the more difficult to budget since all cost factors must be analyzed and scrutinized since profit margins are very narrow. A simple meeting fee is to have each member contribute a fixed amount towards a single budget item such as a meal/refreshments.

Since the computer field is expanding at a phenomenal rate there are a great number of resources for technical presentations. The simplest and easiest is the group membership; many computer professionals have a speciality topic he/she could discuss. The meeting committee can be very helpful in encouraging a user speaker and helping him prepare. Vendors of audio visual equipment often have booklets on making technical presentations. Vendors and universities are also two good sources for speakers. Keep in mind any audio visual support required by speakers. It is often enlightning and fun to have a special speaker during the meeting talk about a non-computer related topic; it relaxes the minds of the listeners. Be sure to recognize the speakers in some manner; a free meal ticket is usually easy while a speaker's gift is nice. In any case send a letter to each speaker before and after the meeting to give him meeting particulars and thank him. A key to a successful meeting is to give sufficient lead time for users to plan to come and speakers to prepare; plenty of mailings help. Each potential attendee should be made to feel that if he/she comes to the meeting they will return home a winner and a greater success.

Proceedings and technical papers require a definite dedication on the part of a meeting committee. Remember each submittal needs to be edited and evaluated not to mention all the mechanics necessary to publish them. There are some techniques and guidelines, however, that make this choice easier. For local group meetings a simple Xerox reproduction or offset printing will suffice for handouts at the meeting. Attendees seem to like even the briefest of descriptions on paper.

Vendors can often be a great source of support for a users meeting. They may even be willing to sponsor or underwire costs, refreshments, facilities,... provided they can display their wares. This approach works even on a small scale by contacting a vendor salesman.

Don't forget to have some fun at user meetings by having a drawing for a "crazy" door prize. If funds permit a more serious door prize is good and/or a small gift for the meeting host, presented before the whole group. Kudos for all who contributed to and participated in the meeting is a necessity. Then, end the meeting on a high note with the attendees thinking about the next "great" meeting.

## Software Library

All computer users like to get a hold of another user's software contribution but often does not have time to contribute himself. An easy rule for this situation is, tit-for-tat, only those who contribute have access to the other contributions.

The form and format of a software contribution must be simple and easy but consistent so a user can easily compare, search for and retrieve entries. The quality of contributions is difficult to maintain but if the user has documented his source code well, and provided a working example and a simple user's guide then the recipient should be delighted. Assuring quality is very time consuming but a minimum quality can be maintained if the thorough examples can easily be proven to work.

Collection and distribution of software library entries needs serious thought. The media is the first consideration (floppy, paper tape, etc.), organization of supplementary documentation second, and third (but most difficult) how to get the job of reproduction/distribution done. There are many approaches but the more the contributing user does in formatting and testing his contribution the easier collecting and organizing the software becomes. Indexing a software library is not easy but a permuted title index is one easy method, provided each title is meaningful.

## Managing and Administration

For small local groups this aspect is relatively easy since the atmosphere is informal and activities are minimum. As the group grows a serious effort on the part of the leaders needs to be put towards the business aspects, i.e., the transition from a "club" to a "business" is considerable. Consider,

1. How to best meet the needs and expectations of the users.
2. Volunteerism and getting the tasks done.

When computer professionals become involved and passionate towards a group they may tend to show emotion and be sensitive to personal reactions. That is, leaders need to pay attention and be sensitive to personal politics enough to make most of the users successful most of the time.

→*[Cater to active enthusiasts since they are the heart of a users group]*

Only volunteers who are responsive and serious about contributing can be asked to perform a task. The task requested to be completed must be well defined and feasible within the necessary time frame. Keep in mind that such a volunteer type task is probably low on the volunteer's priority list. Thus, be generous in designating a time frame but be definite about the required completion date. Make sure the volunteer knows when he is done, i.e., recognizes an end point. Do not be vague nor overwhelm the contributor if you wish the task completed.

Unfortunately the family related to the enthusiastic professional may suffer and account should be taken of this fact. There needs to be some recognition or activity planned to involve them. An individual's contribution to the group can be adversely effected by his family and vice versa.

Administration of a larger users group requires establishing well-defined objectives, a guideline for operation, and specification of particular services to users for particular costs.

## Vendor Interface

If a users group is focussed on a particular vendor product such as the HP3000 computer system establishing an interface with the vendor is important. Such a relationship is mutually beneficial since the vendor has an indepth knowledge about how his product works and a successful user assists in promoting vendor sales and provides valuable feedback for the vendor.

It is important to recognize that a vendor company is made up of individuals, each giving a different response to a users group. Recognize also that the vendor's priorities may not be the same as yours. A users group, however, can assist the vendor in clarifying items of common interest to users that merit priority consideration. Clearly a users group, as a group, could possibly provide alternative solutions. An example might be support of a user who maintains a particularly useful utility program.

Since the vendor is in business to make a profit users group activities that promote sales are more easily supported by the vendor. Thus user meetings open to vendor sales prospects are a success. Successful vendor customers support sales and thus vendors tend to support software libraries and technical communication media such as journals.

A good, friendly working relationship between the users group and the vendor is necessary if both are to be successful. Caution, however, only a minimum of requests should be made of the vendor; if the users group is a success the vendor should volunteer more than adequate support. If the users group remains independent from the vendor the group has a freedom needed by enthusiastic computer system users and the users group need only moderate eccentrics to keep progress in motion.

## Business Approach

If a group of users can function as an informal group and accomplish their objectives then that is the easy and simple club approach. When the group grows to a state where a significant amount of organization is required, people are hired, and considerable money is processed. The club needs to become a business.

This transformation is considerable and requires a business plan, legal status, financing, and the like. There is legal counsel available that will assist in such preparation. Of course, when a users group becomes a "business" the user services, operations, and resources, along with costs and expenses need to be well defined. The "official" users group needs to plan for carrying on each day, for a directorship to make decisions, and a plan for passing the group on to the future. At this level the users group must be run as a business and not the informal club if it is to be a success.

## Structure

The more formal structure of a users group is not required unless it is needed. The need comes when the key group cannot administrate the group easily and/or it is necessary to form a consolidation of communication channels. For volunteer groups this threshold is very low. In a volunteer group each subgroup can more easily complete a clear limited task. The sum of such efforts if coordinated well, can produce useful results.

The subgroups formed may be around the group's operational tasks such as meetings and publications. Other subgroups may form around application areas, vendor products or geographical areas. The important point here is that each subgroup must have a specific task to perform with a definable end point; the subgroup must commit to accomplishing that task. Continual follow-up, encouragement, and assistance is required by users group management. If this effort cannot be put forth then just lists of committees are a burden and the subgroup should be dissolved.

## Conclusion

Becoming involved in a users group is a learning experience. You must realize that while you will gain a great deal of information your rewards will never match your contributions. There is, however, a great latitude for self satisfaction. The alternative to involvement is stagnation, an unchallenging position. If you do not belong to a group of users then join one or form your own, it is easy and simple.

## Acknowledgements

Several key people have assisted my development with computer groups. The first is Alan Mitchell of Tandem Computer Corporation who was co-founder of the HP3000 Users Group. Bill Bryden of Inland Systems Engineering has assisted me in developing several users group meetings, local and international. A great deal of reflection upon my ideas was done by Richard Nelson, Editor, PPC Journal. My thanks to them for supporting my efforts. The most effective support I have had for my user group involvement has come from my very talented secretary, Lynda Schenet.

## References

1. Nelson, Richard J.; Editing and Publishing a Technical Newsletter, Proceedings of the Southern California Regional Users Group Meeting, March 1978.

2. After Proposition 13, Volunteers Needed, Time, August 7, 1978, p. 34.

3. Proceedings, Southern California Regional Users Group Meeting, March 1 and 2, 1978, Douglas J. Mecham, Chairman.

4. Mecham, Douglas J., Founder and First President, HP3000 Users Group, 1974-1976.

## KEYNOTE ADDRESS
## and
## SESSION REVIEWS

Keynote address: "Future Possibilities--Hardware, Software and People".
CPT Grace Murray Hopper, USN

The outstanding event of the conference was a memorable keynote address by Captain Grace M. Hopper of the U S Navy. Before an audience of 500 conferees in the ballroom of the Denver Hilton, CPT Hopper gave a one hour and forty minute talk during which no one was seen leaving and a pin drop would have been clearly heard (except during the many applauses).

The address is difficult to summarize because of its substantial content, but could possibly be best described as "an eyewitness account of the evolution of electronic computers, plus advice and predictions concerning their future".

The entire speech was recorded on videotape by the Users Group and will soon be made available to members. Some highlights are:

-- CPT Hopper's involvement in the development of COBOL - the several deaths of COBOL, including a tombstone.
-- The first computer "bug".
-- Why we in the computer profession must begin considering the value of the information we're processing rather than only its quantity or cost.
-- What a nanosecond looks like.
-- Why the future will be in distributed computer systems based on mini's and micro's rather than large centralized systems.

Viewing a videotape of the address or seeing CPT Hopper in person is very highly recommended.

-Bill Gates

Note: Further information regarding the videotape will be forthcoming in an edition of the Newsletter.

## COMPUTER AIDED INSTRUCTION

Diane Christopherson of the University of Wisconin, River Falls discussed the work they have been doing to convert the Computer Aided Instruction package designed for HP 2000 for use on the HP 3000. After some comments of the difficulties they have had they she went into a full discussion of how the user takes advantage of the various features of the CAI program.

Computer Aided Instruction is divided into three sections: The Instructional Dialogue Facility (IDF), the Math Drill and Practice section, and the Instructional Management Facility (IMF).

Within the IDF the teacher can defien any lesson desired in almost any subject imaginable in any one of seven languages including Portugese, Swahili, and the other modern European languages. Also included in the language menu is Concise an abbreviated form of English. Lesson preparation does not required that the teacher know or understand a proramming language because the Author is led through the lesson preparation by a series of prompts which give a choice of selections such as the text to precede the lesson, questions to be asked and possible correct and incorrect answers with programmed responses to each. The teacher can insert a number of hints which the students may request as they progress through the lesson. The author can even insert a response for the unexpected answer. By specifying, the number of tries the student may make on any given question the teacher can decide when the student has failed the lesson and needs to return to other review material before going forward with the lesson. For this purpose failure messages are prepared within each lesson.

The correct and wrong answers can be specified within ranges as well as specific numeric or literal strings. Here again the programmed response can tell the student he is close to the right answer, give it another try. There is even a provision to accept a misspelled answer called the 'don't care character'.

As the student works through a program lesson he is permitted to branch to other functions, call up the calculator program for arithmetic steps, skip ahead in the lesson or back to a previous step. The author can specify how long the student is permitted to work on a geven question or lesson or may leave it open. In either case the computer gathers statistics on the students' performance. These statistics when produced through the IMF aid in charting the students' progress and in determining the students' grades.

It was estimated that the instructional lessons require about 15 hours of teacher preparation time per hour of student use but since the preparation only has to be done once and the students can use the programmed lesson for an indefinite time it does become an effective too. At any time after creating the lesson the author can go into n IDF function which permits changes in the questions, answers, repsonses or complete rearrangement of the sections or sequence of the lessons.

The Math Drill and Practice is set up on two levels. M1 is designed as 6 grades of 24 blocks each. Each block is made up of a pretest, 5 lessons, and a final test. If the student scores a 100% on the pretest he is skipped ahead to the next lesson in the block. The M2 level is designed for junior high or remedial senior high school math work up to the beginning level of algebra.

In addition to the instructional programs above the CAI also has the IMF through which the student logs on and off the system, which keeps track of student records of participation and if you like school enrollment. It is this facility which prepares the reports as requested by the teacher or school administration.

C. Mallette - Student
Metropolitan State College

## COMPUTER AIDED DESIGN
Residential Energy Audit

In view of the worsening oil shortage the work being done at the University of Wisconsin, River Falls, by Dr. N. H. Prochnow and his associates, to aid in evaluating home energy uses becomes more and more valid. With oil consumption mear an all time peak conservation of the energy we have is the only means of postponing the time of complete depletion of our resourses. Marlys Nelson described how, with the state Health and Energy agency the University is conducting surveys of low income housing and then bringing these homes up to an acceptable level for heat conservation purposes.

Young people in the youth opportunity corps are used to take measurements of the building and report the data to a central location where it is entered into the computer and run through the analysis program. The program then prints an output with recommendations for upgrading, the approximate cost of the improvements and the amount that the improvement should save the home owner on his heating bill. This can be calculated on the basis of any one of nine feating fuels and their current retail prices.

The data collected consists of building size, condition of walls, windows, ceilings, and roof, some calculated R values which the investigator can easily get from the tables in the guide, and the condition of the foundation walls. The style of building is also considered because the existence of basement, crawl space, and additional floors effects the heat loss patterns for the house. Using a degree day determined for the zone of the state the computer approximated the homeowner's heating bill which he can compare with what he has actually been paying to see if the computer's heat loss estimate is accurate.

Since education of the young is the only way we can begin to train people to conserve what energy we have this program has also been extended to the 45 high schools who are connected with the University computer so that Social Issues classes, general math and general science classes can use it. In addition to teaching the conservation of energy and the related course material needed to collect the data it also begins to teach the student something about computer applications and has created interest in learning to program the computer as well.

C. Mallette
Student
Metropolitan State College

A-06

## Corporate Modeling

## Design and Implementation of Financial Planning Systems for the HP 3000

Using a hypothetical manufacturing company, Mr. John Gewecke presented an overview of a Financial Planning System which may be purchased or leased to the HP 3000. USER FORESIGHT is a computerized financial analysis, planning and modeling language developed by United Computing Systems, Inc., Business Information Products, a subsidiary of United Telecommunications, Inc. In use for over a decade, it was the first user-oriented interactive financial planning language to be placed on a computer system and has been continually upgraded and modified in response to changing business trends and computer technology.

FORESIGHT is based on the concept of a matrix. The matrix is made by the intersections of a series of lines and columns (both variable) which each house a data element. For example, a monthly sales forecast by marketing might be developed into the matrix with the sales forecast for each product for each of twelve months as the data elements. A number of descriptive fields are added to this matrix such as the date, sales division, report title, and descriptions unique to the report. With a price per unit for each product determined by marketing, a Planned Product Sales report may be produced. This report will show the forecast number of units sold and gross sales by product for the division on a monthly and annual basis.

Based on these results the manufacturing department may determine the necessary lead time for production and indirect cost and overhead may be forecast. From this input, a Cost of Goods Manufactured report is produced. At the Finance Director's level, most of the detailed information is not required so a Profit and Loss report is produced using only those totals he wishes to see and including additional corporate level items of interest (Depreciation, General and Administrative expenses, Bonuses).

A similar set of reports is produced for each division. From divisional outputs, a series of higher level reports can be easily attained. By the use of the FORESIGHT "consolidate and merge" command a Consolidated Profit and Loss statement is produced. At higher corporate levels a variety of additional reports may be required using the exiting data base of information. A Comparative Profit and Loss statement is produced using the "consolidate and select" command.

At each higher level, additional logic can be added to specified results extracted from the original data base utilizing FORESIGHT's format file capability. For the final presentation to the President, the advanced report writer capability may be called on to produce a more finished report in a format designed by the user.

Ann Minzer
Metropolitan State College Student

## FINANCIAL MANAGEMENT SYSTEM DESIGN

The scope of the design of a financial management system should not stop with general accounting applications, according to C.E. McVaney of J.D. Edwards & Company, but should encompass all facets of financial management reporting in such a way that each level of management is provided with the most timely and appropriate reports for decision-making.

The system designer should begin with the general ledger accounts, then using the top-down approach, determine what data is necessary to "beef-up" the general ledger to create an accounting report data base. Main components of this data base should include: the general ledger, the cost ledger, the chart of accounts, the general journal, sample trial balances, and audit lead schedules.

Three basic report types should be generated by the system: 1) general accounting reports for the accounting and budgeting levels of management, 2) budget plans and 3) various management summaries which should be set up so that the lower the level of management, the higher the amount of detail in the report.

Reports generated by a complete financial management system ideally would include most of the following: Key Variable Reports, Return on Investment analyses, Responsibility Reports, Planned vs Actual Reports, Prior Year Comparisons, Prior Month Comparisons, Financial Ratios, Per Cent of Completion Reports, Analysis of Seasonal Fluctuations and Cost and Profit Center Reports in addition to the common general accounting reports. The system designer should also consider including applicable statistical analyses and exception reports which would aid management.

In summary, a financial management system designer should strive to meet management's needs for timely information through enhanced reporting which will serve both management and operations.

Mary Farris
Metropolitan State
College Student

## A P P L I C A T I O N S

### GRAPHICS IN BUSINESS APPLICATIONS

The use of graphics in scientific applications has long been an accepted way of showing information pictorially. Mr. Cooper suggests that graphics in business can also have more impact than common rows of numbers and statistical data.

Graphics is a better way of reporting because it more clearly illustrates both good and bad points which numerical data alone obscures. Another reason to increase use in business applications is the relative ease of programming a graphics problem (I.E. a pie chart fortran program accepting up to ten variables with approximately two hundred statements).

Two CRT terminals are presently available from H-P with graphic capabilities. These are the models 2647A and 2648A. They feature A 32K bit memory (A bit for each dot on the matrix screen). These can stand alone or be used with a computer. The autoplot feature is available which allows the user to make graphs by simply entering parameters for the X and Y axis. Columnar data can come from three sources; (1) typed in; (2) computer; (3) cassette. Loading the plot takes less than one escape sequence.

Graphic software is now available for the 2647A.

J. A. Vuletich
Student
Metropolitan State College

## GRAPHICS IN BUSINESS APPLICATIONS

Mr. Paul Cooper, an HP Systems Engineer from Tulsa, Oklahoma, started with a slide presentation on the graphic applications available with the 2648 terminal. He pointed out that graphics have been used for some years in scientific and engineering work but are just beginning to be used in the business world. With the growth of distributed data processing putting more terminals at the disposal of more managers some of the mystery is being stripped from data processing and managers are finding there are better tools at their disposal than ever before. Speaking of five principle levels of managers, Mr. Cooper showed how data normally presented in tabular form to each could be quickly translated into a line graph showing actual production levels, or efficiency levels or net worth figures on a comparative basis. Through use of interactive terminals throughout a manufacturing facility management has the capability of accessing timely data shown on a time continum from any terminal as soon as the production data has been generated by the manufacturing unit. There is no longer any need to wait for the analysis of a weekly production report. They have the picture NOW. In this manner trends can be diagnosed quickly, forecasts projected and corrective action taken if necessary before the major problem has time to develop.

The Autoplot Menu leads the user through the steps one by one to build the graphic display he needs. It is a function imbedded in the firmware of the terminal, includes eight different text sizes, and will create the appropriate grid with increments designed as specified by the user when he supplies the maximum and minimum values for each axes. When supplied with the data required the line graph is produced on the terminal screen but hard copy can be obtained by using the 2648 to drive the 4 pen plotter, Plot 21 or a matrix printer.

Mr. Cooper has also included in his paper which will be published with the proceedings the source listing of the 200 statement FORTRAN program which will generate a pie chart.

At this point Mr. Cooper feels that the technology of computer graphic representation has outrun the imagination of business to utilize what could be produced, and they are looking for input from users in business to show them direction for further development in this field.

C. Mallette
Student
Metropolitan State College

# D A T A   M A N A G E M E N T

## Transaction Processing

In the development of new methods of transaction processing, Decision Strategy Corporation of New York has developed a new processor intended for use with HP hardware.

The new processor is called a Terminal Application Processing System (TAPS). Some business objectives of TAPS are listed below:

1) To replace manual processing of orders
2) To cut the invoice to shipment response time
3) To provide order management tools
4) To standardize proceedures in processing
5) To create a vehicle for decentralization

In a typical manufacturing operation, TAPS allows many operational areas within the firm to initiate inquiry and update functions with a main computer, typically the HP-3000.

Management benefits include information by-products and information summaries stemming from automated transactional analysis.

Operationally, TAPS is sub-divided into three main functional areas:

1) TAPS/CM - the communications monitor
2) TAPS/AM - the applications manager
3) TAPS/DM - the data manager

As basically a table driven system, TAPS offers many unique benefis over manual and other automated processing systems. Some basic features include increased data security, accelerated transaction processing, and an inquiry/update function. Other important and unique features warrant more complete explanation.

TAPS assigns a numeric value to variables which enables it to become language independant. This will allow TAPS to become compatible with virtually any hardware and software packages. In addition, TAPS is also format and device independant allowing the user to define new areas of data and not have to rewrite the program. Independance from language and device restrictions will allow TAPS to be adaptable to different applications; a must for future developments in hardware and software. In application development, TAPS will reduce changeover from 95% required under normal circumstances to 5%; generally only input/output changes.

The Terminal Application Processing System can be a valuable tool if your business organization utilizes large amounts of transactional data. TAPS, as an aggregate tool, replaces 30% of a typical transaction system and tabulizes 25% of the development work. In addition, statistics indicate that 30% of the development effort can be saved. Savings on maintenance exceed 50%, because the nature of most maintenance is simply a table change to TAPS. An orientation meeting with a representative of Decision Strategy Corporation may well be time well spent for an organization with large transactional requirements to process.

Richard Rehm
Student
Metropolitan State College

## IMAGE DATA BASE DESIGN AND PERFORMANCE MEASUREMENT

Mr. Orland J. Larsen, IMAGE 3000 Product Manager, addressed this session on IMAGE. IMAGE 3000 and IDEA were the topics discussed.

The steps to design a data base included determination of the information needed to make decisions, the actual design, the data base dictionary and activity against the data base. This last step is where IDEA comes in. IDEA stands for IMAGE Data Base Evaluation Analyzer. The service that this software provides include:

1. A calculation of the estimated load time
2. Response time
3. Throughput
4. Provides initial design feedback
5. Provides future change impact

IDEA is now available for Series I and II. Series III is expected to be announced soon. The main concern is that IDEA does not always work with MPE III. HP is currently working on this problem. HP also expressed the desire for input from users as to its usefullness to determine the future of IDEA.

IMAGE 3000 has been named to the Data Pro Software Honor Roll. It has received the highest user ratings in all categories of any other data base. This included a 3.8 overall satisfaction rating.

Kelly J. Patterson
Metropolitan State College
Student

## Data Management

Mr. D.C. Dummer of D.C. Dummer and Associates, Calgary, AB spoke on "Data Management - Information Management - An Investment for the Future." He discussed the problems and advantages of using the data base approach.

The definition of data base, often a misued term according to Dummer, is an accumulation of raw data which you can use for decision making. In decision making you do not make the right decision, you make the best decision.

In introducing a data base management system (DBMS) to an organization, there may be resistance to change in methods and procedures, loss of data ownership and data control, change in power structure, change in the status of EDP users and in staff requirements. However, there are corporate opportunities which off set the resistance to change. These include improved data utilization, methods and procedures, reduction in cost and improved control.

Information must be brought together from different departments to form a centralized information system. Benefits derived from centralized record files include improved training facilities, controlled introduction of new technology, protective mechanisms and procedures for data resources and easier transition from system to system.

Some practical requirements of data sharing that must be considered are:

1) Data Integrity
   -accuracy of data
   -currency of data
   -usability of data
2) Data Security
   What to Protect?
   -data items and data files
   -computer memory and data storage devices
   -data listings and information reports
   -data transmissions
   -security system(s) and procedures
   Protecting from what?
   -human and system errors
   -environmental accidents and catastrophes
   -mischievousness and fraudulence
   -industrial and political espionage
3) Data Dictionary and Directory
   They are used to define data and structures. The data dictionary and directory should reflect the relationship between data administration, maintenance group, systems group and the user.
4) Backup and Recovery Procedures
   Backup is a utility that restores the data base or some portion of it to a particular state after a situation occurs that causes the data base to lose its integrity. A recovery procedure uses the backup copy of the data base at the checkpoint plus the system journals to generate a new copy of the data base to the point just prior to the failure.

5) Audit Trail
The design of the system so that any transaction, total, or resulting output may be traced back to the original source.

Before selecting a data base management system there are several inital considerations. One important point to consider - is a DBMS really necessary? Other considerations are: Avoid home built DBMS, concentrate on the immediate payoffs, and access risks involved in committing to a partial DBMS. Also you should compare the record of the vendor and the data base with other users.

The ability to access a data base by an individual within an organization offers great potential for the efficient management of a business organization. Data base management systems will play an increasingly greater role in system design because of the advantages they offer in terms of cost, control, and the ability to store and access data.

Beverly Kelman
Metropolitan State College

DATA MANAGEMENT
IMAGE
TIPS AND TECHNIQUES FOR THE NEW USER
B-11

THIS SEMINAR WAS DESIGNED FOR USERS WHO HAVE PURCHASED THE HP IMAGE DATA BASE.

ONE PARTICULAR TIP WAS TO AVOID THE USE OF SORTED CHAINS WHERE THERE ARE A NUMBERS OF "PUTS" AND "DELETES". SINCE QUERY ONLY REPORTS 136 RECORD SIZE POSITIONS SOME USERS RELATED THE NEED FOR MORE. ONE POSSIBLE SOLUTIONS WAS TO USE THE EDITOR AND ADD ONE FILE TO THE OTHER.

QUESTIONS CONCERNING MANUAL AND AUTOMATIC MASTER WERE BROUGHT UP. IN AN AUTOMATIC MASTER IT IS IMPOSSIBLE FOR VALUES TO BE V VERIFIED SINCE IT IS AUTOMATICALLY ASSUMED TO BE CHECKED. A MANUAL MASTER IS NOT SO VOLATILE. IT CREATES A TABLE OF LEGITIMATE VALUES. IF A VALUE IS NOT FOUND IN THE TABLE OF LEGITIMATE VALUES IT IS DISCARDED.

STAND-ALONE DETAILS GIVE THE IMPRESSION OF A MPE FILE SO QUERY CAN BE USED AND SOME SECURITY CAN BE ASSIGNED. STAND-ALONE DETAILS ARE USED FOR TRANSACTION LOGGING.

IF A USER HAS A REQUIREMENT FOR MORE THAN 255 FIELDS (THE FIELD LIMITATION FOR THE DATA BASE) EVERY EFFORT SHOULD BE MADE TO DISCUSS WITH THE USER THE REASON WHY THE THINK THEY NEED MORE FIELDS AND WHAT EXACTLY WHY THEY WANT TO USE QUERY.

FOR LISTING PURPOSES IT WAS SUGGESTED TO USE "**" INSTEAD OF "@". THE FORMER IS SEVEN TO NINE TIME FASTER FOR EACH DATA SET. THE LATTER WILL LIST ALL ITEMS. ANOTHER ADVANTAGE OF "**" IS THAT IT USES THE PREVIOUS LIST THAT HAS BEEN PROCESSED AND KEPT RATHER THAN CHECKING THE ALGORITHM.

THE QUERY COMMAND "NUMBERS" WAS SUGGESTED TO FIND AND SELECTIVELYY GO DOWN AND COMPARE THE QSLIST TO A DISK FILE. "NUMBERS" WILL SET UP A QSLIST FILE THAT CAN BE BROKEN DOWN.

NANCY L. YELLOTT
METROPOLITAN STATE COLLEGE
DENVER, COLORADO

## VIEW/3000: A NEW TOOL

THE VIEW/3000 SYSTEM, AS PRESENTED BY JUTTA KERNKE OF HEWLETT-PACKARD'S GENERAL SYSTEMS DIVISION, IS A VERSATILE NEW SOFTWARE PACKAGE FOR DATA ENTRY. VIEW/3000 CONSISTS OF FOUR MAJOR FACILITIES: FORMS DESIGN, SOURCE DATA ENTRY, DATA REFORMATTING, AND PROGRAM INTERFACE. THE FACILITIES ALLOW THE USER TO CREATE INTERACTIVE DATA ENTRY SCREENS IN A "FILL-IN-THE-BLANKS" MANNER WITHOUT COMPLICATED PROGRAMMING EFFORT. THE FORMS DESIGN FACILITY WILL ALLOW THE USER TO INCLUDE DATA EDITING SUCH AS LENGTH CHECKS, RANGE CHECKS, TABLE CHECKS, EQUALITY CHECKS, PATTERN MATCH, AND CHECK DIGIT VERIFICATION. VIEW/3000 WILL ALSO JUSTIFY, FILL, UPSHIFT, AND STRIP DATA INTO A PRE-DEFINED FORMATTED FORM. VIEW/3000 WILL ALSO ALLOW DATA TO BE MOVED BETWEEN FIELDS ON A SINGLE FORM OR BETWEEN SEVERAL FORMS. ARITHMETIC AND CONDITIONAL PROCESSING ON THIS SYSTEM CAN BE USED TO TRIGGER CUSTOM ERROR MESSAGES AND LINK SEVERAL FORMS TOGETHER. MS. KERNKE POINTED OUT THAT VERY SOPHISTICATED FORMS COULD BE CREATED VERY EASILY. MODIFICATIONS TO EXISTING FORMS ARE ALSO VERY SIMPLE PROCESSES.

VIEW/3000 CAN BE USED INTERACTIVELY WITH AN APPLICATION PROGRAM OR IT CAN ACT AS A STAND-ALONE DATA ENTRY FACILITY. USERS CAN USE THE SYSTEM TO CREATE EDITED AND FORMATTED FILES OF DATA. THE DATA ENTRY OPERATOR CAN "BROWSE" EXISTING DATA FILES AND VERIFY THEIR CONTENTS. THESE DATA FILES CAN ALSO BE REFORMATTED TO MEET THE APPLICATION PROGRAM'S REQUIREMENTS. FOR EXAMPLE, DATA FROM SEVERAL FORMS CAN BE COMBINED INTO ONE BATCH FILE. THE REFORMATTING IS A BATCH PROCESS WHICH CAN BE DONE AFTER ALL THE DATA HAS BEEN ENTERED.

MS. KERNKE NOTED THAT VIEW/3000 MUST BE USED ON AN HP/3000 SERIES COMPUTER WITH AT LEAST 256KB OF MEMORY AND OPERATING UNDER THE MPE-III OPERATING SYSTEM. ALSO, THE KSAM (KEYED SEQUENTIAL ACCESS METHOD) SOFTWARE MUST BE AVAILABLE FOR VIEW/3000'S USE. THIS SYSTEM IS DESIGNED FOR USE WITH THE HP264X SERIES OF CRT TERMINALS AND AT LEAST 4K OF TERMINAL MEMORY IS ADVISED. GIVEN ITS VERSATILITY AND SOPHISTICATION, VIEW/3000 SHOULD PROVIDE ITS USERS WITH A GREAT DATA ENTRY TOOL.

WILLIAM L. BLANKENSHIP
STUDENT
METROPOLITAN STATE COLLEGE
DENVER, COLORADO

## Process Optimization - On-Line Programs

Optimizing on-line programs is an important issue for all users of any computer system. This view is clearly one Robert M. Green of Robelle Consulting LTD. took in the formation of his firm. The goal of optimization is to get the most productivity from your system's resources. This is a rather broad statement and goal to achieve, it also requires a broad perspective to deal with it. To aid in this goal, Mr. Green has outlined five basic ideas or principles for on-line programs. They are:

1) Make each disc access count
2) Maximize the value of each terminal input
3) Minimize the run-time program size
4) Avoid constant demands for execution
5) Optimize for the common event

These are general rules which should be kept in mind in designing a program or the structure in which the program will work. The key to optimization is make the system resources open for all users of the system and not to overload the system by doing so. Consquently, the user should establish a priority system to do jobs and in that priority system optimize for the common event in a particular job. This sorting process should be rational in that optimizing a high priority job that doesn't occur frequently would in essence be a waste of resources.

Perhaps the effectiveness or usefulness of any of these basic principles must be weighed on a cost vs. benifit analysis. An arguement against optimization or more precisely one against delegating much time to it, is that optimization is a fuction of memory. What I mean here is that a system will be suitable for the user's need if there is enough memory. This belief is furthered by the fact that memory will get cheaper as time passes. I believe this view will cause a tangled mess as poor documentation does. Optimization intergrates processes with the computer as well as within the computer enhancing throughput and reliability. In a burgeoning data and data processing environment optimization is still an important issue.

John L. Fong
Metro State College

INSTALLATION MANAGEMENT
SYSTEM SECURITY
D-08

BILL GATES, WHO GAVE THE SPEECH, IS ASSOCIATED WITH LONG DRUG STORES, INCORPORATED LOCATED ON THE WEST COAST.

LONGS IS UNIQUE IN THAT ITS OPERATIONS ARE DECENTRALIZED AND IT HAS NO CENTRAL WAREHOUSE OR SYSTEM. LONGS LACKED A GOOD SECURITY SYSTEM UNTIL THEY HAD A DATA PROCESSING AUDIT A FEW YEASRS AGO. THE AUDITORS SUGGESTED THAT LONGS MAKE CHANGES TO THEIR EDP OPERATIONS.

ALL PROGRAMS ARE RUN FROM APPLICATIONS GROUPS. THEY CAN ONLY ACCESS DATA WITHIN THE GROUP EXCEPT FOR CERTAIN DATA BASES.

FOR THE BACK-UP PROCEDURE LONGS ELECTED TO USE "STORE" RATHER THAN "SYSDUMP". THE REASONS FOR THIS IS THAT "STORE" MAY BE SELECTIVE THAT IS, ONLY FILES THAT ARE NECESSARY IN CASE OF DISASTEER ARE STORED. "SYSDUMP" HAS THE DISADVANTAGE OF NOT DUMPING FILES THAT ARE CURRENTLY IN USE. "STORE" MAY BE RUN DURING OTHER PROCESSING.

MOST BACKUP IS DONE BY THE APPLICATION GROUP. ONLY FILES NECESSARY FOR RECOVERY, IN CASE OF CRASH, ARE STORED. THE FREQUENCY OF SYSTEM DUMPS DEPENDS ON THE APPLICATION GROUP. A COMPLETE SYSTEM BACK-UP IS DONE EACH EVENING AT 6 P.M.

A "CONSOLE OPERATIONS PROGRAM" CAN ACCESS THE PRODUCTION JCL FI FILE. IT READS JOB STREAMS INTO A TEMPORARY FILE AND ALLOWS THE CONSOLE OPERATOR TO ENTER PROGRAM "PARAMETER CARDS". IT THEN STREAMS FROM THE TEMPORARY FILE. THE PROGRAM DOES AN INTRINSIC CALL RIGHT INTO THE DIRECTORY TO FIND THE APPROPRIATE PASSWORD AND INSERTS IT IN THE JOB RECORD.

FOR SYSTEM MAINTENANCE THE "GOLD BOOK" IS KEPT UP TO DATE. THIS LOG CONTAINS THE SYSTEM FAILURE LOG, MAINTENANCE LOG, COPY OF SERVICE CONTRACTS AND A COPY OF THE CURRENT CONFIGURATION.

A "SYSDATA" JOB IS RUN EACH MONDAY MORNING. THIS JOB HAS NUMEROUS LISTINGS OF FILES, FILE SPACE, AND A DATA BASE UTILITY LISTING. THESE ARE ALL REVIEWED AT DEPARTMENT MEETINGS.

THE CURRENT PHYSICAL SECURITY AT LONGS IS SIMPLY RESTRICTED ACCESS TO THE COMPUTER ROOM. STANDARD APPLICATIONS CONTROLS INCLUDES DIVISION OF RESPONSIBILITY, EXTERNAL INPUT AND OUTPUT BALANCING BY USERS, AND USER APPROVAL OF PROGRAM CHANGES. ACCESS TO THE DATA IS RESTRICTED BY THE USE OF PASSWORDS, USER CAPABILITIES WITH SOME USERS ONLY ABLE TO "READ", AND EDP DEPARTMENT RESTRICTIONS WITH PROGRAMMERS ACCESS LIMITED.

DATA ACCESS IS AUDITABLE. THE AUDITOR REVIEWS EVERYTHING THAT GOES ON. WHEN A JOB IS TO BE RUN THINGS MUST BE TIED TOGETHER WITH A JOB REQUEST SHEET, $STLIST, CONSOLE AND SYSTEM LOG.

DISASTER CONTINGENCY PLANS HAVE BEEN IMPLEMENTED. A HALON FIRE PREVENTION HAS BEEN INSTALLED. THERE IS ALSO OFF-SITE BACK-UP OF FILES. AFTER A DISASTER HAS OCCURED AN EXPENSIVE AND TIME CONSUMING SOLUTION WAS INSTIGATED: BACK-UP SITES FOR THE COMPUTER AND A BACK-UP LOCATION FOR USERS.

MR. GATES STRESSED THAT SECURITY IMPLEMENTATION IS A GRADUAL PROCESS. THE COMPUTER ROOM WAS AT FIRST WIDE OPEN AND IT WAS THEN GENERALLY CLOSED OFF. PASSWORDS ARE CHANGED WEEKLY AND THE SYSTEM MANAGER PASSWORD IS KNOWN ONLY BY THREE PEOPLE. THIS PASSWORD IS CHANGED AFTER IT IS GIVEN OUT.

ANOTHER POINT BROUGHT OUT WAS TO INVOLVE EDP AND USER PERSONNEL

IN THE SECURITY PROCESS. EXPLAIN THE TRADE-OFFS, CHALLENGE PERSONNEL TO DEVELOP GOOD SECURITY REQUIREMENTS AND EFFICIENT PROGRAMMING TECHNIQUES. ALSO VARYING DEGREES OF COMPLIANCE WITH SECURITY MEASURES ARE FELT BE PERSONNEL. THESE SHOULD DECREASE OVER TIME AND FAMILIARITY WITH THE NEW PROCEDURES.

NANCY L. YELLOTT
METROPOLITAN STATE COLLEGE
DENVER, COLORADO

## WHAT'S AHEAD IN COBOL

Mr. Greg Gloss of HP's General Systems Divsion conducted this session on the changing world of COBOL using three main topics to guide the presentation and discussion. First he discussed some of the new features proposed for the '74 COBOL followed by suggestions for the conversion from '68 COBOL and finally a few comments on '80 COBOL.

Users were urged to start programming now with the conversion in mind so that the transition will cause a minimum of disruption to the operation. The major changes in programming '74 COBOL will involve changes in some COBOL statements, changes in the reserved word list and changes in the treatment of comments. These changes are covered in detail in the Conversion Guide which was available in a preliminary form.

New in the standard '74 COBOL include indexed I/O, relative I/O, which checks for the presence of data on the file record, enhanced SORT and MERGE, STRING and UNSTRING verbs to concatinate data with delimiters and unconcatinate fields, and Multiple Copylib Files. MERGE becomes a part of the language and the SORT verb will permit multiple input files. The EXAMINE verb is replaced by INSPECT and REMARKS and NOTE paragraphs are eliminated to be replaced by the * in column 7.

There is now a conversion guide program available which will flag any COBOL statement in a program of '68 COBOL if that statement requires reprogramming before the program can be used after the conversion. It does not make the correction it just shows what will have to be changed giving the user a chance to plan the stages of hs conversion.

Improvements in the COBOL language standards are attempting to eliminate some of the error prone statements, making programs faster to debug and more reliable to run. The changes are being aimed not at improving compilation time but in shortening runtime once the program is compiled. This makes for greater efficiency in the production environment and it is hoped will allow production to run 30 to 40 percent faster.

As a member of the ANSI COBOL committee, HP is seeking input from users about what they would like to see in the next version of COBOL, currently being called COBOL 80. So far suggestions include: CODASYL DATA BASE FACILITY, reference modification, 48 levels of subscripting and a USAGE BIT. Plans include gradual phasing out of 77 and 66 level data items, Label Records clauses, Value of clauses, Add/Subtract Corresponding, Alter statements, the picture character 'A', most of the Identification Division paragraph and the INSPECT TALLYING . . . REPLACING verb which is coming in as part of '74 COBOL. Some file related clauses would be moved from the environment to the data division.

During the question period questions included the ability to use SET for a condition value, a method of tying nexted IF's to their own ELSE's, debugging features and the creation of permanent files.

C. Mallette
Student
Metropolitan State College

## SPL/3000: SYSTEM PROGRAMMING LANGUAGE PART I OF II

R. SCROGGS, OF NOEISIS IN SAN FRANCISCO, PRESENTED AN INTRODUCTION TO SPL/3000. SPL/3000 IS THE IMPLEMENTATION LANGUAGE ON THE HEWLETT-PACKARD SERIES 3000 COMPUTERS. IT IS A HIGH LEVEL LANGUAGE OF MUCH VERSATILITY. THE LANGUAGE ALLOWS THE USER TO CREATE PROCEDURES, TO UTILIZE SUBROUTINES, TO PERFORM LOGIC UNDER CONDITIONS, TO CODE NESTED IF-THEN-ELSE SEQUENCES, AND MANY OTHER SOPHISTICATED LOGIC SEQUENCES. WHILE SPL/3000 INCORPORATES MANY HIGH LEVEL LANGUAGE FACILITIES, IT ALSO ALLOWS THE USER TO DIRECTLY MANIPULATE THE HARDWARE REGISTERS, TEST HARDWARE CONDITIONS, PERFORM BIT MANIPULATION, AND GENERATE MACHINE INSTRUCTIONS.

MR. SCROGGS DEVOTED THE FIRST OF HIS TWO PART PRESENTATION TO THE GENERAL SYNTAX OF SPL/3000 AND THE HP/3000 ENVIRONMENT. SPECIFIC TOPICS COVERED INCLUDED THE FOLLOWING:

1) DISTINCTION BETWEEN GLOBAL, LOCAL, LITERAL, AND PARAMETRIC DATA
2) DEFINITION OF THE STACK OR DATA SEGMENT
3) THE VARIOUS FORMS OF SCALAR VARIABLE STORAGE--I.E. BYTE, INTEGER, LOGICAL, DOUBLE, REAL, AND LONG
4) THE USE OF ARRAYS
5) THE GENERAL FORMAT OF AN SPL PROGRAM
6) ARITHMETIC AND LOGICAL OPERATORS
7) THE EQUATE AND DEFINE STATEMENTS
8) ASSIGNMENT STATEMENTS
9) IF-THEN-ELSE STATEMENTS
10) THE VARIOUS FORMS OF THE DO STATEMENT

MR. SCROGGS SAID VARIOUS UNFOUNDED FEARS OF SPL/3000 HAVE CAUSED IT TO BE IGNORED BY MANY USERS. HE SUGGESTED THAT GIVEN THE INTRODUCTION AND THE REFERENCE MANUAL, MOST APPLICATION PROGRAMMERS SHOULD BE ABLE TO CODE IN SPL. MR. SCROGGS SUGGESTED THAT THE BEGINNER START BY CONVERTING SOME SIMPLE BASIC OR FORTRAN PROGRAMS TO SPL/3000.

WILLIAM L. BLANKENSHIP
STUDENT
METROPOLITAN STATE COLLEGE
DENVER, COLORADO

## ADVANCED BASIC

E-09

Mr. Warren Kuehner, HP Systems Engineering Supervisor, gave his presentation on various techniques of interfacing SPL with basic to overcome slight misrepresentations and design flaws in the HP 3000. However extreme flexibility in calling SPL routines within a HP 3000 made correction easier.

Some unique characteristics of Basic called SPL routines were also presented. The stack marker retains information relating to the state of the machine when the subroutine is called. On top of the stack, information is also kept pertaining to the parameters passed to the subroutine in the order they are passed. The number of parameters passed is totaled and codes are developed to determine the type of parameter passed, both features of use in diagnostics and both unique to Basic. It was suggested that these features be used as a "Cookbook" in developing subroutines.

Other important characteristics of Basic were:

1.) Basic calls an intermediate routine which in turn calls the SPL routine.
2.) Basic calls by reference, not by value.

To actually run the routines it is necessary to use a segmenter built SL file. If a error is detected it was suggested that the SL file be purged by use of the procedure call.

Specific routines discussed were:

1.) To overcome a flaw in DEL which limited its formated display screen for data entry to 255 characters. The routine would provide for longer strings.
2.) To increase double-precision accuracy on the HP 3000 series 1 to 48 digits.
3.) To obtain the MPE file number by just giving the Basic number.

Comments:

It was pointed out that even though Basic is a beginners language that it is second only to SPL in data base application code efficiency.

Craig Morris
Metropolitan State College

## SPECIAL PURPOSE LANGUAGES

Mr. M.J. Badlander of B & B Computer Company addressed this session on the design and implementation of special purpose languages. His presentation covered the following:

1. How to determine when a special purpose language is needed. If (a) the problem area can be well-defined, and (b) existing languages are unsatisfactory, and (c) there will be frequent usage of the special language, design and implementation of a special purpose language should be considered.

2. Considerations in designing a special purpose language. These include power, flexibility and naturalness of the language, as well as an awareness that designing a special purpose language is an art, not merely a technical job.

3. Alternate methods of implementation. A special purpose language can be implemented as (a) a preprocessor: this method is the least difficult as well as the least costly, but offers limited applications and power; (b) an interpreter: a medium difficulty factor is involved, the cost is moderate to high, and applications are limited; or (c) a compiler: the most complex and usually the most costly method, but provides the most flexibility and power.

4. Advantages of a special purpose language. The increased power of the language should result in a need for fewer and less complex programs. Reduced complexity would lead to greater reliability. Both the increased power and reduced complexity should simplify program maintenance. All three factors should result in lowered software costs.

Mr. Badlander then presented a case involving the design of a special purpose language - DEC/3000 - to solve the following needs for a user: 1) terminal and form management, 2) off-line batch development, 3) hide terminal details from programmers, 4) provide adequate documentation, 5) support non-HP terminals and 6) dovetail with existing software. DEC/3000 was implemented through a compiler, simplifying the language, compiles into USL files, and uses a host language concept. It provides terminal management, data editting, checking and conversion, and handles input/output as part of solving the user's problem.

Mary Farris
Metropolitan State
College Student

# A P P L I C A T I O N S   D E V E L O P M E N T

## PAST PRESENT AND FUTURE

Due to the increasing technological improvements occurring in the computer industry the use of compilers in the future will change making applications programming the direction of development according to Mr. Couch.

There are basically two types of compilers; algorythmic and specification. Algorythmic languages provide step by step procedures to break down a job into several individual and specific procedures (I.E. FORTRAN and COBOL). Specification languages are not interested in a major breakdown of a problem. They deal with a more general subject area leaving the user free of minute details such as specific input and output definitions (I.E. RPG).

Historically, this is what has occurred. From 1955 - 65 a rapid development of algorythmic languages came about. In 1965 and up to 1970 high level algorythmic languages found wide acceptance. This was due largely to a strong push from the government and everyone else getting on the "band wagon". Also applications programming was being pushed in FORTRAN and COBOL. From 1970 and up until today, high level algorythmic languages are still prevelent. Also most applications are being written in high level languages.

Mr. Couch sees that for the future algorythmic languages will still be used but specification languages will also be in wide use. The reason for this being the increasing disparity between hardware and software prices. The small user can now afford a computer but can't afford the software. In this way, a non-programmer can carry out most needs without technical expertise.

Applications programs could provide users with packages for inputting, outputting and accessing data. The major characteristics of this system are:

* INTEGRATION OF EDP FUNCTIONS
* PROVIDE ADDITIONAL FUNCTIONS FOR THE NON-PROGRAMMER
* USE INTERACTIVE POTENTIAL OF CRT

The environment that applications programs should be used in is described as follows:

* FORM/DOCUMENT ORIENTED
* SCRIPT AND INTERACTIVE DIALOG MODES
* KEYSTROKE RESPONSIVE
* LIMITED PURPOSE - TRANSACTION/BUSINESS PROGRAMMING
* INFORMATION USERS SYSTEM

J. A. Vuletich
Student
Metropolitan State College

## DATA COMMUNICATIONS

Eric Pennala presented this seminar on data communications developing, from a management viewpoint, a planning and problem solving guide.

I PLANNING

A. The first step in selecting a communications system is to define the volume.

B. Contact Hewlett-Packard salesman and S.E. once you have a basic layout.

1. When making a selection, layout the total needs and let the salesman propose a solution and let him back his proposal.

C. Determine if you are capable of handling the system, if not, secure the services of someone who does know the system.

1. If you do not have someone who can debug the systemhave someone trained.

D. Determine the terminal you will need and the specifications of it.

E. Select a vendor, one that is reputable, can offer full service and as fast as possible. Stay with one vendor for the entire system if possible.

F. Set up a meeting for everyone involved.

1. The primary comcern here is to get everyone to sgree.
2. Determine the HP CPU configuration.
3. Determine and write down the proper strapping functions.
4. Determine the cable arangements and electrical needs.

A. Site preparation is important and a good area to look for ways to cut costs. Put pressure on the building inspectors to eliminate waste.

5. Have a clear cost understanding covering all aspects of your communication system.
6. Draw up a formal installation plan including dates and times.
7. Put the results of the meetings on paper and send copies to the appropriate people; a precoution that might prove valuable when someone tries to do a little backsliding.

II PROBLEMS

A. HP system configured incorrectly.

B. Communication system is configured incorrectly.

C. Expect everything to be wrong, expect it!

1. Suggestions to overcome these problems.

A. When they install boxes and lines, label them! Include vendor and repair phone numbers on label.

B. Develope communication book- label cables, ports and record circuit numbers. Write down recommended trouble-shooting procedures.

C. Have a testout planned.

1. Have everyoue on hand for testout, make sure they bring their data scopes and data analysis machines the first time!

D. No cooperation or no solution.

1. Go through normal channels first.

2. After you have exausted every other alternative.
   A. Get MAD, Be MEAN!
   B. Work your way up through supervisors until you reach the person that will make something happen. Let them know exactly what your feelings are, chances are that person does not like problems and does not want to be bothered. You will get results.

E. Every way to get system up fails.
   1. Get the DAYTECH Division of Bell Systems on the job, they have the reputation of being the elite of the communications field.

III SUMMARY FOR MANAGEMENT
   A. Put pressure on Hewlett-Packard.
   B. Willingness to get consultant, know and accept you will have problems.
   C. Planning and Knowledge are the key words in communication systems.

Michael L. Hooks
MSC Student
DENVER, CO

# DATA COMMUNICATIONS ON THE HP 3000

The first session of a two-session series of presentations by Charles J. Villa Jr. from Alter*Ability of San Francisco, addressed the rudiments of data communications. Even so, the session was, due to time constraints, limited to what Mr. Villa called "a cram course in HP 3000 Datacom". So, a limited background in the datacom field was a necessary prerequisite for a professional understanding of the material. At one point during the discussion, Mr. Villa queried the group as to how many present had basic knowledge of datacom fundamentals. Suprisingly, about half of the audience held up their hands.

At the outset, it was said that the field of datacom was one that was fascinating and rapidly changing.

An adequate datacom network can gather data on personnel very quickly and efficiently. Also, development of safeguards to deter invasion of privacy of data has advanced considerably in recent times.

A basic concept of datacom is that data transmission is accomodated over voice networks that are modified in that the data signals (digital) are converted to analog signals (voice) before transmission. This process of modification is done by a <u>modem</u>, or a modulator-demodulator.

Certain communications disciplines called protocols are used to provide a set of procedures for establishing and controlling transmissions in datacom. The HP 3000 recognizes three asynchrnous line disciplines:

1. Hardwired, using no modem; the HP 3000 provides no control signals whatsoever, whereby control characters are inserted into a data stream for signalling the receiving station to perform a function or to identify the structure of the message.

2. 103 protocol, full-duplex; a communications link is established that performs a prescribed sequence of events each time a mainframe calls a terminal or vice-versa. Communications is allowed in both directions at the same time.

3. 202 protocol, half-duplex, reverse-channel; communications is allowed in one direction at a time here. But, FSK-type modulation (frequency shift keying), is normally used with this protocol; FSK is a form of modulation using two separate frequencies to represent the two binary digits (0 and 1).

The state of the art in modems is presently the Dataphone Data Set 212A produced by the Bell System, Mr. Villa stated. When the terminal operator dials the computer, communications is established between the modem and the computer, when the 212A modem is utilized.

A discussion of <u>pin numbers</u>, which are used in interface curcuitry in modems when connecting cables ensued, in which a series of questions from the user portion of the audience arose. The technicality of this concept appeared to be above that of the rudiments of datacom.

A hand-out entitled "Term Type Table" depicted in tabular form the individual characteristics of 14 kinds of communications terminals applicable for use with the HP 3000 computer. A significant portion of the table was devoted to the requirement or lack thereof that the terminals have sync characters and how many were required for each terminal. A sync character is a character of a defined bit pattern that is used by the receiving terminal to adjust its clock and achieve synchronization. Other characteristics listed were response to delete function and forms feed, line feed and carriage return options, and speed in characters per second.

A noteworthy statement from the handout was "the definition of the remote terminal's function determines the desired speed of the terminal and the volume of data to be handled, which determines the type of datacom line and equipment needed, which determines not only the required hardware for the HP 3000, but also the configured terminal subtype. And, we must not forget the log-on terminal type, which determines the control characteristics for the log-on terminal for the duration of the current datacom link".

Leased lines have the aspects of a hardwired connection to the HP 3000, and no real documentation available.

In datacom jargon, a signal that is "high" is one that is on.

The second session covered the equipment and elements required in setting up a datacom network. Three major types of equipment contribute to a datacom network: Datasets (modems), data distributors, and data convertors.

Probably the most dollars can be saved in network structures by choosing the proper dataset. Factors such as operational distance, minimum speed, type of line service (leased or switched), line discipline, and configuration are all to be considered in determining cost/effectiveness of a given dataset.

Operational distance was expounded upon as follows:

1. <u>line drivers</u> are not designed to work on phone lines (5 to 10 miles); also, they require a dc continuity from end to end. If two points are within the same <u>wire center</u>, a modified 3002 circuit is achieved.

2. <u>short haul modems</u> work over a 3002 circuit (10 to 50 miles).

3. <u>long haul modems</u> work over voice grade lines with <u>DAA</u> interface.

Data distributors can also reduce network costs by choosing wisely. Data distributors are more widely known as multiplexers, which are devices that allow the use of one facility for two or more simultaneous data paths. FDM (frequency division multiplexing) works at low speeds and costs less. TDM (time division multiplexing) works at high speeds and costs more.

Statistical multiplexers are the state of the art, and more efficient use of communications lines is realized through their use.

Data convertors are devices that convert asynchronous signals to synchronous signals.

As to types of line services, it was mentioned that switched lines experience more noise but cost less, while leased lines have a fixed cost (higher) but noise is not a significant factor to contend with.

Another new concept in networks is the Value Added Network. An example of this concept is Telenet, which uses packet switching. In Telenet, a call can be made to long distances at local call rates. In packet switching, the terminal delivers up to 128 characters to the network; the network takes the packet and sends it through the most efficient routing. Term type 13 supports packet switching on the Telenet network.

The last concept covered was digital circuits, which regenerate and amplify signals instead of converting them to analog signals for transmission. The signal stays in a digital state and provides for low-bid circuits.

Presentation by: Charles J. Villa Jr.

Summarized bv: Patrick G. Dulskv
Metropolitan State College Student

G-04

## Program Scripting for Custom Transaction Entry

Mr. D. D. Brown of the Nice Corporation in Ogden, Utah addressed this session on System Design. He discussed how his company developed SPICE/3000 (System Processor Instant Consumer Exchange on a HP 3000 computer).

The software used is COBOL for transferability. The programs are modular, that is they are transaction driven. This was necessary because when the company started development, they didn't know what the software would be required to do.

The data file system is composed of a master file, stat file, procs file, and a utility file.

1. Master file: each record is 256 bytes long. It was stressed the buffers should end on even multiples. If not, if the computer is being shared, the system will assume that there are no buffers and this multiples the time needed by the number of users.

2. Stat file: provides pointers for each item. It also shows the first transaction to take place each day.

3. Procs file: provides all the dialogue that comes out on the CRT screens. This file contains all the routines and subroutines that are used.

4. Utility file: contains all the embellishments that are in the system: messages like "Good Morning" and messages telling the people that are operating the CRT's what to look out for.

The features of SPICE/3000 are: total modulability, on-line evaluation, security, data base integrity is continuously tested, and there is Mag tape backup. Also, if the system goes down, the most that would be lost is one transaction.

The Nice Corporation handles 20 to 25% of all calls in response to television commercials advertising products such as record albums in the country. They presently have 28 phone/CRT operators and are planning to expand to 45 by the first of next year.

Bill Brunson
Metropolitan State College Student

## SYSTEMS DEVELOPMENT
### System Testing & Reliability
### Software Quality Control

Mr. C. E. McVaney of J. D. Edwards & Co. addressed this session on "Computer Systems Software Quality" and described in his presentation what might be called the three phases of data software.

First, there was the "age of survival" and in this time period programs were written just to get the job done with no foresightedness as to modification or changing perimeters. There was no such thing as documentation and if there was, it was of a very poor quality. Do everything for today and let tomorrow take care of itself.

Next there came the "age of high quality". In this second phase the original program had proven themselves to a certain extent and now comes the time for modification as well as an extensive look as to the maintenance cost in keeping this system of starting from scratch. There is now more foresight as to what the future will hold for these programs and documentation is being upgraded. A new word is introduced-standards. With the introduction of standards every programmer is able to write a program that is structural and uninformal so that someone else may work on it and understand it after he has left that to converse about some data element and everybody knows what the other person is talking about.

The third phase is where we are today, "The El Cheapo Software". The time and money invested in developing your own software may be less profitable due to the fact that there are so many firms today who can or have produced the systems that you may need. Watch out for what looks extremely cheap today may cause you extensive headaches and high maintenance cost tomorrow. As with any commodity, shop around for the best deal that will not only benefit you today but will require less work and money to modify it in the future.

Programs today are written with seventy-five percent "grunt work" and twenty-five percent genius. The term "grunt work" refers to the fact that there is only a certain number of ways in writing particulars types of software for a system. i.e.-Accounts Receivable-there is only a limited number of ways to write a program to do an add, deletion or an updating. The twenty-five percent genius comes into play when adapting the program to an user's system and in doing, use the least amount of storage and maintain data integrity.

By not taking advantage as to what program and services are being offered, management may view the problem as one concerning itself with the hardware. If this happens, management may acquire more hardware than is really needed and this is referred to as the "Iceberg Effect". The problem appears to be one of hardware but upon a closer look, the real problem or problems may be deep down. Such things to look at are: computer programs, system documentation, input control, procedures-both operating and clerical, data bank information and lastly the reports for both management and operating systems. So by only seeing the top of the iceberg more hardware was added on and the problem went away. You were just able to bury it deeper into the system and given time it will raise it's ugly head again only this time much higher than ever before.

From the age of survival, through the search for quality and lastly the el cheapo phase, the software reliance and capability has grown at an ever increasing rate. It is up to management to properly use this computer tool to its fullest extent and not look so much at the tip of the iceberg by acquiring more hardware but by evaluating the whole system for its merits or demerits and there openly make its judgements after all the data has been gathered.

Walter Anderson Jr.
Metropoltan State College

## ON-LINE APPLICATION MANAGEMENT

Mr. Dennis Dinam and Mr. Glenn Entis, from Morgan Guaranty and Trust Company of New York addressed this session on Screen Management Off-line Generation package (SMOG). They discussed the needs, development and use of this system.

The system was designed to work in an HP 3000 Series II, on-line and some batch and a Multiuser environment. Morgan Guaranty and Trust Company comprises four banking functions:

1. Automated Banking System
2. Euro-currency Automation
3. Global Exposure System
4. New York Profitability System

The requirements of the system were:

1. Self contained system
2. Computer Management
3. User and function Oriented Security
4. Terminal Management
5. Full screen Support

The system that was developed to meet all of the above conditions was SMOG. It helps the HP 2645A Interactive Display Terminal to create and maintain files of formated data. It is extremely useful in instances where screens are changed frequently. All procedures are written in SPL but are accessed through COBOL, FORTRAN and SPL.

The Terminal Application Management Program (TAMP), a part of this system, generally defines the application environment.

Another component of the package is Key Extra Data Segments (KEDS). They felt KSAM was too slow for a system that contained many table valuations. KEDS was the answer and is similar to KSAM and manages tables in virtual memory.

The design of the package has taken about two years at Morgan Guaranty and Trust. It is a system for the management and development of on-line uses and not directly for the end user.

Kelly J. Patterson
Metropolitan State College
Student

## The Software Development Cycle

Mr. J.M. Grillo of HP's Genenral System Division addressed this session on 'The Software Development Cycle." He discussed the problems an the keys to success in the Development System.

Key Problems in Development System:

A. Implementing new management and system processes

B. Skill problems
   1. Common problems
      Lack of communication between technicians and users
   2. The use of outsiders
      The risk is that the outsider might not know the Business
   3. Expertise problems

C. User involvement

D. Cost overruns

E. Lack of detailed requirement
   1. Constant changes that the user will run into, because it is hard to tell when you are done.
   2. Inefficiency in programs and decumentation

Keys to successeful system development

A. User involvement
   1. Interest and support
   2. Involvement in all phases
   3. Develop acceptance test

B. Personnel utilization
   1. Business analysts
      a. Define system with user
      b. Install and train
      c. Review design for conformance
         what the system should do and what the user is asking for.

C. Technical Personnal
   1. Define and implement
      the technicians in business study should go over requirements with user.
   2. Review definitions for feasability
   3. Define/Develop technical environment
      How to structure Data Base Management

Standards

A. Documentation
   1. What must be documented and thought through
   2. Exposure decisions for people to make judgment on.

B. Naming
   Standarized the vocabilary

C. Programming techniques
   difficulty of language selection

D. Formats for printed media

E. Environmental standard

Keys to success

- A. Review and Revision Procedures
- B. Phase to phase budgeting
- C. Phased development
  1. System concept phase (5% of costs of the project)
     - a. Define scope and effort
     - b. Review current system
     - c. Identify leverage/risk areas
     - d. Perform analysis
     - e. Develop new system concept
     - f. Task & cost estimate for General Design (cost & task timing)
     - g. Estimate system development costs
  2. General design phase (10% of cost)
     - a. Develp & detailed work program
     - b. Resolve remaining issues
     - c. Define inputs & outputs
     - d. Define processing function
     - e. Define manual procedure, personnel requirement & volumes
     - f. Define technical support requirements
     - g. Define an integrated system flowchart
     - h. Refine cost/benefit analysis
     - i. Order development priorities
     - j. Develop task list and costs for detailed design
     - k. Refine estimate of system develpement cost
  3. Detailed Design Phase (20% of cost)
     - a. Develop detailed work program task allocation
     - b. Resolve remaining issues
     - c. Define & desing codes
     - d. Design data base
     - e. Develop detailed subsystem flowcharts
     - f. Specify inputs & outputs
     - g. Develop program specifications
     - h. Develop manual procedures
  4. Programming and unit testing (35% of cost)
  5. System testing (15% of cost)
  6. Conversion & training (15% of cost)

Ann Sawaged
Metropolitan State College

## ABSTRACT

Microprocessors: Product Design Using Microprocessors

Presented by: J. C. Armstrong, Los Altos Research

Presented at: Hewlett-Packard Users' Meeting,
Denver, Colorado,
October, 1978

A discussion of useage of the HP 3000 for development of software for microprocessor product design.

The technique of using the HP 3000 in developing software in various developmental systems was presented along with the problems encountered and the solutions developed.

The use of the HP 3000 as a large, full-scale editor overrides the cost of setting it up. The advantages are a large editor, sophisticated editing tools, cross-reference facilities and multiple users. The development system box can be selected after programming. Efficiencies are obtained as only one editor is learned--not a different one for each development system. The program documentation, reference manuals and user manuals can also be done on the HP 3000 rather than the microprocessor.

One user writes microprocessor software on the HP 3000, including program entry, cross-assemblies, debugging and ouputs Intel hex to the development system. A CRT input of assembler language to the HP 3000 goes through a SPL program and out to a Z80 development system. The HP 3000 is "invisible" in this application.

Another development uses the HP 3000 editor with PLM language and tells the development system that the HP 3000 is a teletype.

[illegible]

## A Real-Time Instrument Interface

Mr. G. R. Symonds of Env. Health Directorate, Ottowa, has developed a viable real-time interface with H. P. 3000 computers. His 'black box' is an alternative to the H. P. 1000, which he considers to be overpriced. In his development of this interface, Mr. Symonds determinded that certain characteristics would be necessary. They are: immediate response, event driven, asynchronous, implied data lost or disaster, immediate acceptance or rejection, something controlling something, and subject to the vagaries of Murphy's Law. He concluded that this interface was similar to a data entry clerk and has referred to it as a impersonal one. With these requirements Mr. Symonds went on to try to interface with the H. P. 3000. His first step in this approach was to determine where to branch into the system . Almost by default he choose the Asynchronous terminal multiplexer. Frequency restrictions and hampered flexibility were some of the problems this outdated multiplexer has. But these problems were nothing compared with a Selector channel or Multiplexer channel choice. The H. P. configuration and channel choice acts like a modem, therefore anything that interfaces with it must be or appear to be a terminal. The 'black box' is set in line between the terminal and the H. P. 3000. As a backup, Mr Symonds recommends that a printer be hooked up in parallel with his 'black box'. Because the 'black box' is not a terminal certain dialogue characteristics that the H. P. 3000 has with a terminal must be turned off. In general there must be some customizing of hardware and software to make this interface work.

Opinions are varied for this hardware solution. Microprogramming of the terminal is possible and can achieve the same results. Costs are probably the biggest factor in choosing the 'black box' solution. Mr. Symonds calculates around $ 800 for design and manufacture of his interface. This is compared with a programmer working five days at $ 325. There was considerable skepticism whether this goal could be accomplished in that time. Mr. Symonds' hardwire approach is cost effective and available in a otherwise limited area of data processing

John L. Fong
Metro State College

## Data Management on the HP 3000
## MPE File System

Four panelists, led by Mr. John Couch, recently appointed Project Director of the MPE File System, held a "Round Table" discussion with HP 3000 users. There was no formal presentation and the emphasis was placed on interaction, as members of the panel answered specific questions pertaining to the MPE File System.

Users felt that KSAM reliability should be increased and that the manual should be changed to properly describe its use. It was specifically suggested that the manual should have an improved description of carriage control. When it was suggested that ordinary MPE utilities should work on IMAGE files, the Project Director for KSAM and IMAGE commented that he felt that IMAGE and KSAM should not be separate entities.

In response to a question concerning the difficulties encountered with KSAM when a crash occurs, one of the panelists said that HP is aware of the problems and has identified a number of things that might be done. Among those considered is the possibility of a pointer to the area at the crash point.

John Couch encouraged users to send their suggestions concerning MPE to him. When he asked how many of those present hesitated to turn in SMR's because they felt it went into the "bit bucket", two-thirds answered affirmatively. He then assured them that all suggestions were filed by classification and scrutinized carefully when appropriate changes were being considered. A member of the User's Group Interface Committee encouraged the users to contribute input to HP by filling out the annual questionnaires they receive.

As a wrap-up to the discussion, Mr. Couch outlined the objectives and strategy he is following in his new position:

1. Reliability: Isolate as many bugs as possible (200 MPE problems have been identified) and assign the personnel to solve them.
2. Performance: Double the performance of the operating system. 1970 decisions led to scarce memory and as this is changed, so should the performance.
3. Increase capabilities.
4. Form an MPE 3 group responsible for enhancements to the present MPE File System.
5. Form an MPE 4 group responsible for "look ahead" projects such as re-writing the file system and restructuring the operating system.

Ann Minzer
Metropolitan State College Student

## DATA COMMUNICATIONS

I-07

Mr. J. Alderete, a HP Project Manager, identified three specific areas of data communications currently being developed by HP. He also made note that the projects he would be discussing are not in production but are in the developmental stage only.

The three general areas are:

1.) An intelligent controller program
2.) IBM data communications
3.) HP networks

The first concerns a mirco-processor based, intelligent data communications controller, specific benefits of the input/output subsystem would be:

1.) Reduce CPU overhead
2.) Simple extension
3.) Supportability
4.) Diagnostic capabilities

New IBM system support was the second announcement. Interactive communication with the IBM host are to be improved by:

1.) Improving console message handling
2.) Making job submission more flexible
3.) Expanding output

Benefits of these efforts are to be realized in the area of distributed data base management, especially inventory control.

The third area invloves standardization of the data communications within HP networks. Extensive polling of HP users and HP management will be necessary to develop such a system. Three specific advantages of the above efforts were mentioned.

1.) Packet switching
2.) Use of A,T and T's Advanced Communication Service
3.) Satillite communication

Questions were mostly on IBM data communications. Remote Job Entry and Multiple Remote Job Entry were discussed extensively. RJE through an IBM 2780 Data Transmission Terminal or 3780 Data Communication Terminal was compared to MRJE.

One-way protocall, limited compression, and limited console features were some of the drawbacks of RJE. MRJE, a more powerful full duplex line, offers a possible line cost advantage, has high compression, and a live console.

In response to other questions:

1.) MRJE is to be enhanced later through the CLP (6 months to a year).
2.) MRJE is a high overhead feature as compared to a RJE system.
3.) Data Systems Inc. is working on a dial up line for transmission within an HP group.
4.) MRJE will not support RES efficiently.
5.) The CLP will eliminate 10 percent of CPU overhead.
6.) Asynchronous transmission is not yet being developed.
7.) X-25 status is currently being developed in Europe.
8.) MTS currently supports 14 terminals on one site. Possible release soon.
9.) Nothing is currently being done to adapt peripherals to new HP 3000 system 3.

Comments:

Content of the round table soon became too envolved for over half of participants who left at various times throughout the block. Definition of the subjects to be covered were wholly misunderstood as beneficiaries of the discussion were confined to those with an extremely advanced knowledge of the material.

Craig Morris
Metropolitan State College

This is a summary of the special "Round Tables" session, Machine Utilization: System Monitoring and Tuning Performance Measurement Tools, held on November 2, 1978, at the 7th International Meeting of the Hewlett Packard General System Users Group.

The session was moderated by Tom Rawson of the University of Washington. Two HP lab persons, Roy Clifton and Ken Spalding were in attendance, and addressed questions and topics which were brought up.

The purpose of the session was to give feedback to HP as to what kinds of things are available, mainly in the Contributed Library, that the users would like to see HP produce.

Reference was made to the paper in the proceedings by Jim Squires and Ed Splinter, System Performance Measurement and Optimization, and the following tools were mentioned and described:

| | | |
|---|---|---|
| 1) | MONITOR | A major tool provided by HP which is part of the operating system of the Series I, II, and III HP3000. The system activities are recorded and a window of that period is chosen for analysis which provides information concerning resource utilization, system workload and the programs used. Available only to SEs. |
| 2) | TUNER2 | Provides a display of what the CST is doing, For use with MPEII and MPEIII. In the Contributed Library. |
| 3) | SOO (SON OF OVERLORD) | For use with MPEII. Has a bug when used with MPEIII. To correct it, one must set the define statement with PCB entries to what is needed. In the Contributed Library. |
| 4) | SHOWVM | Gives a snapshot of virtual memory to show how much memory the MPE is using. In the Contributed Library. |
| 5) | SHOWPIE | Shows processes and what the system is doing. In the Contributed Libary. |
| 6) | SHOWQ | An MPE command provided by HP which lists all processes in the system as dormant, waiting or running. |

|  |  |  |
|---|---|---|
| 7) | IDLE/IDLE PRINT | Used to show how busy the CPU is hourly. Some problem exists in that it gets itself in EM and should be in priority 254. In the Contributed Library. |
| 8) | OVERLORD | For use with MPEIII. Does not give CPU percentage as does SOO. |
| 9) | DREEACTG | Thirty to forty programs in FORTRAN and SPL which analyze the log file to give information concerning system use. In the Contributed Library. |

Additional tools that were addressed were SAMPLER and TRACER, which will probably be available late next year and are expected to provide more evaluation at the application level than MONITOR, such as segment histograms. SAMPLER presently will require the use of an additional clock board.

It was stated that one of HP's objectives is to produce tools to be sold to users which the users can use themselves, and that HP wants to know what information the users need. The following user desires were expressed:

1) That each district office should have a clockboard to rent to users who wish to use SAMPLER.

2) Tools that the user can use without having to go to his SE every time.

3) User does not want an SE in his system because of classified material.

4) Would like to see something from HP like SOO, and does not want to have to go to an SE every time something goes wrong.

5) Would like to see a CPU utilization guage in the HP3000.

6) A set of tools to measure user running time to give him a histogram of what the user is doing.

7) Wants MONITOR to be broken down into separate tools so that he can choose what information he needs.

8) A need to give the operator better control of the system:

    a) The ability to suspend sessions in order to get the system out of thrash.
    b) The ability to change the subqueues.
    c) The ability to alter priorities.

9) A tool to tune an application program when developing a system:

   a) To determine how the program is written affects usage time.
   b) To determine what is affecting performance.
   c) To give information on segmentation vs. design efficiency.

10) To be able to change queues according to the time of day because certain users use the system at the same times each day.

11) Wants segmentation information provided for FORTRAN programmers to improve program performance.

12) Wants to see who is running what, how much CPU time is being used, and where a job is in the system.

13) Wants snapshots of memory over periods of time.

14) Wants HP to provide some configuration guidelines.

The most generally agreed upon desires of the users seemed to be for tools that the users can use themselves which will not require any additional hardware, and for the provision of some basic configuration guidelines.

By David Schwaab
Metropolitan State College

ATTENDEE AND EXHIBITOR

NAME LISTS

CONFERENCE ATTENDEE LISTING INDEXED BY ATTENDEE NAME

| NAME | TITLE | COMPANY | ADDRESS | CITY | STATE | ZIP | COUNTRY |
|---|---|---|---|---|---|---|---|
| ADAMS BOB | SYSTEMS EDP MANAGER | SCOTT PAPER LTD | P O BOX 760 | NEW WESTMINST | BC | | CANADA |
| ADAMS KEARNEY A | SR PROGRAMMER ANALYST | SMITH KLINE INST | 880 W MAUDE AVENUE | SUNNYVALE | CA | 94086 | USA |
| ADAMS RAY | CONTROLLER | GRACELAND COLLEGE | | LAMONI | IA | 50140 | USA |
| ADDIS JOHN | MGR CONSULTING SVCS | UNITED COMPUTING SYS | 1901 AVE OF STARS #585 | LOS ANGELES | CA | 90067 | USA |
| AGRUSTI RAYMOND J | | WAYNE BOARD OF ED | 50 NELLIS DRIVE | WAYNE | NJ | 07470 | USA |
| ALBARRAN RAUL F | GTE DE PROCESAMIENTO | CARDANES, SA | APARTADO POSTAL 591 | QUERETARO | QR | 591 | MEXICO |
| ALDERETE JOHN R | PROJECT MANAGER | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| ALEXANDER BYRON L | CONSULTANT | COLLIER-JACKSON ASSOC | 1805 N WESTSHORE BLVD | TAMPA | FL | 33607 | USA |
| ALEXY MICHAEL J | DATA SYSTEMS STAFF | STORER BROADCASTING | 1177 KANE CONCOURSE | BAY HARBOR IS | FL | 33154 | USA |
| ALLAN WILLIAM D | INFO SERVICES MANAGER | KAISER FOUNDATION | 2005 FRANKLIN ST | DENVER | CO | 80205 | USA |
| ALVAREZ MANUEL A | COORDINATOR | UNIVERSITY IDAMETROPOLITAN | APDO POSTAL 55-503 | MEXICO | DF | 13 | MEXICO |
| AMBLER BURT | | HEWLETT-PACKARD | 815 14 SW | LOVELAND | CO | 80537 | USA |
| ANDERSEN SVEND | | EBI COMPANIES | 1290 NORTH FIRST ST | SAN JOSE | CA | 95112 | USA |
| ANDERSON EDWARD W | SR SYSTEMS SPECIALIST | WEYERHAEUSER CO | AFB 4 | TACOMA | WA | 98402 | USA |
| ANDERSON GARY D | ASSOC PROF OF BIOSTAT | MC MASTER UNIVERSITY | 1200 MAIN ST WEST | HAMILTON | ON | L8S4J9 | CANADA |
| ANKERS, JR ALFRED H | MANAGER DATA PROCESSING | MARITIME TERMINALS | 7737 HAMPTON BLVD | NORFOLK | VA | 23505 | USA |
| ARANDA JORGE A | SYSTEMS ENGINEER | UNIVERSIDAD BAJA CAL | OBREGON Y F | MEXICALI BAJA | | | MEXICO |
| ARCAYA PEDRO M | MANAGER | PROCESASEG, SA | TORRE LA PREVISORA | PISO SABANA G | | | VENEZU |
| ARMSTRONG JACK C | PARTNER | LOS ALTOS RESEARCH CENTER | 339 S SAN ANTONIO ROAD | LOS ALTOS | CA | 94022 | USA |
| ARNOLD PAT | SYSTEM ENGINEER | PETRO CANADA | P O BOX 2844 | CALGARY | AL | T2P2M7 | CANADA |
| ARTHOFER ROBERT J | DATA PROCESSING MANAGER | FOSECO INC | 20200 SHELDON ROAD | BROOKPARK | OH | 44142 | USA |
| AUSLANDER RICHARD C | DB DC TECH SPECIALIST | THE GAP STORES INC | 900 CHERRY AVE | SAN BRUNO | CA | 94066 | USA |
| AUSTIN DAVID J | ACTING DATA CENT COORD | DENVER EMPLOY & TRAIN | 1421 ELATI STREET | DENVER | CO | 80204 | USA |
| AUSTIN DONALD (CANCEL) | MGR SYSTEMS & OPERATIONS | SYRACUSE BOARD OF EDUC | 409 W GENESEE ST | SYRACUSE | NY | 13202 | USA |
| BADGER PATRICK S | DIRECTOR ACADEMIC COM | XAVIER UNIVERSITY | | NEW ORLEANS | LA | 70125 | USA |
| BAKST LAWRENCE E | SENIOR CONSULTANT | SOFTWARE SYSTEMS TECH | 39 BROADWAY 32ND FL | NEW YORK | NY | 10006 | USA |
| BALANDER MATTHEW J | SYSTEM PROGRAMMER | CCSC | 2640 PINE STREET | ST LOUIS | MO | 63103 | USA |
| BAMBACH THOMAS H | PROJECT LEADER | ALLIED CHEMICAL COPR] | COLUMBIA RD | MORRISTOWN | NJ | 07960 | USA |
| BANDI DENNIS | TECHNICAL ANALYST | DOMTAR INC | 395 DE MAISONNEUVE BLVD | MONTREAL | QU | J6K2E6 | CANADA |
| BANSAL A K | | SOHIO | | CLEVELAND | OH | 44115 | USA |
| BARKER DAVID A | MANAGER SYSTEMS & PROG | PURITAN INSURANCE CO | 1515 SUMMER ST | STAMFORD | CT | 06905 | USA |
| BARKER RONALD E | MANAGING DIRECTOR | CLAYBROOK COMPUTING | 70 BROOK ST | LONDON | | W1Y2HN | ENGLAN |
| BARMAN MARC | DEVELOPMENT ENGINEER | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| BARTOLI JR CHESTER T | OPERATIONS SUPERVISOR | HP AVONDALE DIV | ROUTE 41 & STARR RD | AVONDALE | PA | 19311 | USA |
| BASSELLIER JEAN-PAUL | DIRECTEUR INFORMATIQUE | VILLE SAINT-LAURENT | 777 BOUL LAURENTIEN | SAINT-LAURENT | QU | H4M2M7 | CANADA |
| BEACH JOHN R | HEAD OF OPERATIONS | ADRIA LABORATORIES | P O BOX 16529 | COLUMBUS | OH | 43216 | USA |
| BEADLE, JR RAY | SR SYSTEMS ANALYST | GRUMMAN AEROSPACE | SOUTH OYSTER BAY ROAD | BETHPAGE | NY | | USA |
| BEASLEY DAVID R | | ANDERSON COLLEGE | | ANDERSON | IN | 46011 | USA |
| BECHER ANTHONY H | HEAD C.A.M. ENG | HUGHES AIRCRAFT CO | BOX 3310 BLDG 607/B329 | FULLERTON | CA | 92634 | USA |
| BEDET ROB | | GOV'T OF N W TERRITORIES | | YELLOWKNIFE | NW | X0E1H0 | CANADA |
| BEHELER ANN F | PROJECTS MANAGER | ROCKWELL INTL-COLLINS | 1200 N ALMA RD | RICHARDSON | TX | 75081 | USA |
| BEHNKEN H PAUL | SUPVR DATA PROCESSING | SMITHS INDUSTRIES | P O BOX 5389 | CLEARWATER | FL | | USA |
| BELLIS STEPHEN | PROGRAMMER | DEPT OF FISHERIES & OCEANS | BRANDY COVE | ST ANDREWS | NB | E0G2X0 | CANADA |
| BEMAN ROGER | V P MARKETING | INFORMATION RESOURCES | 7935 E PRENTICE | ENGLEWOOD | CO | 80111 | USA |
| BENJAMIN ROBERT A | SR SYSTEMS PROGRAMMER | KEYDATA CORP | 1400 WILSON BLVD | ARLINGTON | VA | 22209 | USA |
| BENNETT WALTER W | DEV INFO COORDINATOR | HARVARD | HOLYOKE | CAMBRIDGE | MA | 02138 | USA |
| BENOIT WAYNE F | MANAGER PRODUCT DEV | EPSILON DATA MGMT | 24 NE EXECUTIVE PK | BURLINGTON | MA | 01803 | USA |
| BERAM ALFRED J | EXECUTIVE VICE PRESIDENT | FINANCIAL DATA PLANNING | 2670 TIGERTAIL AVENUE | MIAMI | FL | 33133 | USA |
| BERGER ROBERT(CANCEL) | ASSISTANT VICE PRESIDENT | BANKERS TRUST CO | 1 BANKERS TRUST PLAZA | NEW YORK | NY | 10006 | USA |

| NAME | TITLE | COMPANY | ADDRESS | CITY | STATE | ZIP | COUNTRY |
|---|---|---|---|---|---|---|---|
| BERGOLD THEODORE A | MANAGER INFO-SYSTEMS | GATX LEASING | ONE EMBARCADERO CNTR | SAN FRANCISCO | CA | 94111 | USA |
| BERND WALTER | ASSISTANT SYSTEM OFF | BANKERS TRUST CO | 1 BANKERS TRUST PLAZA | NYC | NY | 10006 | USA |
| BHUTA MEHENDRA | MGR SYSTEMS PLANNING | ALLIED CHEMICAL CORP | COLUMBIA RD | MORRISTOWN | NJ | 07960 | USA |
| BILLS DANIEL G | PRESIDENT | GRANVILLE-PHILLIPS CO | 5675 ARAPAHOE | BOULDER | CO | 80303 | USA |
| BILMER,EARL | | DOMTAR INC | 395 DE MAISONNEUVE BLVD | MONTREAL | QU | J6K2E6 | CANADA |
| BINDEWALD THOMAS L | SR TECHNICAL ANALYST | GTE DATA SERVICES | FIRST FINANCIAL TOWER | TAMPA | FL | 33611 | USA |
| BIRKWOOD ILENE M | S E MANAGER | HEWLETT-PACKARD | 5303 STEVENS CREEK BLVD | SANTA CLARA | CA | 95050 | USA |
| BISHOP LARRY | DIRECTOR SYSTEMS DEVELOP | NPD RESEARCH, INC | 15 VERBENA AVENUE | FLORAL PARK | NY | 11001 | USA |
| BIVENS FREDERICK(CANC) | DATA SPECIALIST | STANDARD OIL COMPANY | 101 W PROSPECT AVENUE | CLEVELAND | OH | 44115 | USA |
| BLACK DAVID T | SYSTEMS ANALYST | JOHN HENRY COMPANY | P O BOX 17099 | LANSING | MI | 48981 | USA |
| BLAND JOHN J | SYSTEM DP ANALYST | BROOKEHAVEN NATL LABOR | 32 BROOKEHAVEN AVE | UPTON | NY | 11973 | USA |
| BLIESNER ROBERT G | SYSTEMS ANALYST | BCS | PO BOX 24346 | SEATTLE | WA | 98124 | USA |
| BOOTH BUD | PRESIDENT | AUTOMATED ANALYSIS | 3105 DONA SOFIA DRIVE | STUDIO CITY | CA | 91604 | USA |
| BORDEN JOHN (CANCELED) | SYSTEM MANAGER | BALTIMORE GAS & ELEC | BOX 1475 | BALTIMORE | MD | 21203 | USA |
| BORG CHARLES J | SYSTEMS PROGRAMMER | HEWLETT-PACKARD SRD | 1400 FOUNTAIN GROVE PK | SANTA ROSA | CA | 95404 | USA |
| BOSCO SUSAN M | PROJECT MANAGER | THE GAP STORES INC | 900 CHERRY AVE | SAN BRUNO | CA | 94066 | USA |
| BOTTEGAL THOMAS M | | GEORGE WASHINGTON UNIVERSI | 725 23 STREET NW | WASHINGTON | DC | 20052 | USA |
| BOWNES GORDON | SYSTEMS ANALYST | SYNCRUDE CANADA LTD | 10030 107TH ST 7TH ST PLAZA | EDMoNTON | AL | T5J3E5 | CANADA |
| BOYER CONNIE Y | PROGRAMMER/ANALYST | MARY WASHINGTON CLG | P O BOX 1081 CLG STAT | FREDERICKSBUR | VA | 22401 | USA |
| BRAUN GUENTHER K | DIRECTOR DATA PROC | SPRECKELS SUGAR DIV | 50 CALIFORNIA ST | SAN FRANCISCO | CA | 94111 | USA |
| BRICKER HARLEY G | SUPT MATERIALS | SYNCRUDE CANADA LTD | P O BAG 4009 | FT MC MURRAY | AL | T9H2L1 | CANADA |
| BRINOLE JIM | | BRANT COMPUTER SERVICES | 615 BRANT STREET | BURLINGTON | ON | L7R2G6 | CANADA |
| BRODOWSKI RICHARD E | DIRECTOR INFO SYSTEMS | IND PRESS-TELEGRAM | 604 PINE AVE | LONG BEACH | CA | | USA |
| BROOKE ROGER G | MGR SUPPORT SERVICE | WESTINGHOUSE ELEC CORP | P O BOX 8839 | PITTSBURGH | PA | 15221 | USA |
| BROOKS ROY | | SYNCRUDE CANADA LIMITED | 10030 107TH STREET | EDMONTON | AL | T5J3E5 | CANADA |
| BROWN D DAVID | PRESIDENT | NICE CORPORATION | 4357 AIRPORT PARK PLAZA | ODGEN | UT | 84403 | USA |
| BROWN DALE A | DIR OF INST RESEARCH | MARY WASHINGTON CLG | P O BOX 1081 CLG STAT | FREDERICKSBUR | VA | 22401 | USA |
| BROWN DANA | ADMINISTRATOR INFO | EBI COMPANIES | 1290 NORTH FIRST ST | SAN JOSE | CA | 95112 | USA |
| BROWN EDWIN J | PRESIDENT | WOOD,BROWN & ASSOCIATES | 1673 CARLING AVE | OTTAWA | ON | K2A1C4 | CANADA |
| BROWN ROBERT A | DATA PROCESSING SYSTEM M | B.K. SWEENEY MAN. CO. | 6300 STAPLETON S. DR. | DENVER | CO | 80216 | USA |
| BRUCE CLIFF | D P ASST MANAGER | SAN JOSE MERCURY-NEWS | 750 RIDDER PARK DR | SAN JOSE | CA | 95190 | USA |
| BRUNK GREGG A | ADMINSTRATIVE ASSISTANT | HOUSTON INSTRUMENT | 8500 CAMERON ROAD | AUSTIN | TX | 78753 | USA |
| BRYDEN WILLIAM | PRESIDENT | INLAND SYSTEMS ENGINEERING | | REDLANDS | CA | | USA |
| BRYSON GARY J | CONSULTANT | SYSTEM HOUSE | 560 ROCHESTER | OTTAWA | ON | K1B3B8 | CANADA |
| BUELL DIANE | ANALYST/PROGRAMMER | AURORA PUBLIC SCHOOLS | 1085 PEORIA STREET | AURORA | CO | 80011 | USA |
| BUTLER KEITH C | ENGINEER | HEWLETT-PACKARD | RT 41 & STAR ROAD | AVONDALE | PA | 19311 | USA |
| BUTLER STEPHEN M | DIRECTOR DATA PROCESSING | PARADISE VALLEY HOSPITAL | 2400 E 4TH STREET | NATIONAL CITY | CA | 92050 | USA |
| BYFORD WENDY K | SYSTEMS ENGINEER | H P | 275 HYMUS BLVD | PTE CLAIRE | QU | | CANADA |
| CALLANAN BILL | S E SUPERVISOR | HEWLETT-PACKARD LTD | KING ST LANE WINNERSH | WOKINGHAM BER | | | ENGLAN |
| CAMARILLO MARIO R | DIR DE SISTEMAS | INFORMATICA DESC, SC | THIERS 248 ANZUREZ | MEXICO | DF | | MEXICO |
| CAMERON LOUISE | PROGRAMMER | DEPT REG ECO EXPANSION | 200 RUE PRINCIPAL | HULL | QU | K1A0M4 | CANADA |
| CANTWELL FRANK E | DIRECTOR PRODUCT DEV | EPSILON DATA MGMT | 24 NE EXECUTIVE PK | BURLINGTON | MA | 01803 | USA |
| CARLSON LEE A | PROF MATH & COMP SCIENCE | VALPARAISO UNIVERSITY | ACADEMIC COMPUTER CENTER | VALPARAISO | IN | 46383 | USA |
| CARR CHARLES E | MAN ANALYST PROG II | SANGAMON STATE UNIVERSITY | SHEPHERD ROAD | SPRINGFIELD | IL | 62708 | USA |
| CARRUTHERS ALEX R | SYSTEMS ENGINEER | HEWLETT-PACKARD | 210 7220 FISHER ST | CALGARY | AL | T2H2H8 | CANADA |
| CARVALHO MARCOS A X | SYSTEMS ANALYST | PROMON ENGENHARIA S A | NOVE DE JULHO 4939 | SAO PAULO | SP | 01407 | BRAZIL |
| CASEY CHRIS J | SYSTEMS SUPPORT MGR | HEWLETT-PACKARD | 1501 PAGE MILL RD | PALO ALTO | CA | 94304 | USA |
| CAYLOR LARRY W | DATA PROCESSING MANAGER | JOHANSON DIELECTRICS | 2210 SCREENLAND DRIVE | BURBANK | CA | 91505 | USA |
| CELLI JOHN | BUS & MKTING DEV MGR | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| CHADWICK GRAHAM D | COMPUTER MANAGER | DE ZOETE D BEVAN | THROGMORTON STREETS | LONDON | | | ENGLAN |
| CHARD MILES M | DATA PROCESSING MANAGER | CLA-VAL CO | P O BOX 1325 | NEWPORT BEACH | CA | 92663 | USA |
| CHASE LARRY M | SUPV SYS & PROGM | MULTNOMAH COUNTY ESD | 220 SE 102 | PORTLAND | OR | 97216 | USA |

| NAME | TITLE | COMPANY | ADDRESS | CITY | STATE | ZIP | COUNTRY |
|---|---|---|---|---|---|---|---|
| CHATFIELD DENNIS C | SYSTEM MANAGER | R J FRISBY | 1500 CHASE | ELK GROVE | IL | 60007 | USA |
| CHEN CHANG L | SYSTEM ANALYST | HP AVONDALE DIV | ROUTE 41 & STARR RD | AVONDALE | PA | 19311 | USA |
| CHITWOOD RICK L | ASST V P INFO SYS | UNITED PRESIDENTIAL LIFE | 217 SOUTHWAY BLVD E | KOKOMO | IN | 46901 | USA |
| CHIU WUYAN | PROJECT ENGINEER | BOEING COMM AIRPLANE CO | P O BOX 3707 36-02 | SEATTLE | WA | 98124 | USA |
| CHRISTOPHERSON DIANE | | UNIVERSITY OF WISCONSIN | | RIVER FALLS | WI | 54022 | USA |
| CLABORN GEORGE H | PROGRAMMER ANALYST | ENVIRONMENTAL ELEMENTS | 3700 KOPPERS ST | BALTIMORE | MD | 21227 | USA |
| CLARK KIM | | MC MASTER UNIVERSITY | 1200 MAIN STREET WEST | HAMILTON | ON | L8S4J9 | CANADA |
| CLEEVER KAREN J | SYSTEMS PROGRAMMER | DEPT REGIONAL ECONOMIC EXP | 601 SPADINA CRES E | SASKATOON | SA | S7K3G8 | CANADA |
| CLEMENT RUBY J | DIRECTOR OF DP | UMPQUA COMM COLLEGE | P O BOX 967 | ROSEBURG | OR | 97470 | USA |
| CLEMENTSON GERHARDT C | DIR OF ACAD COMP CNTR | METROPOLITAN ST COLLEGE | 1006 11TH STREET | DENVER | CO | 80204 | USA |
| CLEVELAND WALTER | | UNITED COMPUTING SYS | 2525 WASHINGTON | KANSAS CITY | MO | 64108 | USA |
| CLIFTON ROY A | MPE SUPPORT MANAGER | HEWLETT-PACKARD | 972 BUCKEYE CT | SUNNYVALE | CA | 94086 | USA |
| COLDREN J DAVID | ASSOCIATE DIRECTOR | IL LAW ENFORCEMENT CO | 120 S RIVERSIDE PLAZA | CHICAGO | IL | 60606 | USA |
| CONNOR JACK | SR CONSULTANT | SYSTEMS & COMP TECH | 7 N 5 POINT RD | WEST CHESTER | PA | 19380 | USA |
| COOK LINDA L | DATA PROC SPEC/PRGMR | ALLEGHENY INTER UNIT | SUITE 300 TWO ALLEGH CENTER | PITTSBURGH | PA | 15212 | USA |
| COOKSEY WILLIAM P | PROGRAMMER/ANALYST | COPCO | 4905 LIMA STREET | DENVER | CO | 80239 | USA |
| COOPER PAUL D | SYSTEMS ENGINEER | HEWLETT-PACKARD | 4110 S 100 E AVE | TULSA | OK | 74145 | USA |
| COOPER STEVEN M | CONSULTANT | AMERICAN MANAGEMENT SYSTEM | 561 PILGRIM DRIVE SUITE D | SAN MATEO | CA | 94404 | USA |
| COPLIN ROBERT G | D P DIRECTOR | CITY OF KENNEWICK | 210 W 6TH | KENNEWICK | WA | 99336 | USA |
| CORDELLE YANN | ENGINEER | COGELOG | 1 QUINCONCES | GIF | | 91190 | FRANCE |
| CORRELL STEVEN | ENGINEER | HEWLETT-PACKARD | RT 41 & STAR ROAD | AVONDALE | PA | 19311 | USA |
| COUCH JOHN D | SECTION MANAGER | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| COVITT MARC L | TECHNICAL SVCS MGR | HEWLETT-PACKARD SDD | 16399 W BERNARDO DR | SAN DIEGO | CA | 92127 | USA |
| COX W PHIL | SYSTEMS ANALYST | PROCTOR & REDFERN | 75 EGLINTON AVE E | TORONTO | ON | M4P1H3 | CANADA |
| CRAWFORD JEFFREY D | PRESIDENT | OMEGA SYSTEMS, INC. | 3720 SO ROCKWELL ST | CHICAGO | IL | 60632 | USA |
| CROCKETT DAVID | PRG MGR HP300 SYS | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| CROW WILLIAM M | MGR TECHNICAL SYSTEMS | AUSTIN INFORMATION SYS | 450 WEST 1ST AVENUE | ROSELLE | NJ | 07203 | USA |
| CULPEPPER BRITTON B | SYSTEMS ANALYST | UNION CAMP CORP | | FRANKLIN | VA | 23851 | USA |
| CURBELO RALPH | COST ANALYST | NEW YORK TELEPHONE CO | 1095 AVE OF THE AMERICAS | NEW YORK | NY | 10036 | USA |
| CURRERI JOSEPH A | | | 9142 EDMONSTON CT 202 | GREENBELT | MD | 20770 | USA |
| CURTIS LINDA | ANALYST/PROGRAMMER | PETRO CANADA | P O BOX 2844 | CALGARY | AL | T2P2M7 | CANADA |
| D'ANGELO RICHARD J | SYSTEMS ENGINEER | HEWLETT-PACKARD CO | 32 HARTWELL AVE | LEXINGTON | MA | 02173 | USA |
| DAILEY JOSEPH D | MANAGER DATA PROCESSING | MASTERCRAFT IND | 4881 IRONTON ST | DENVER | CO | 80239 | USA |
| DAMM JACK A | PRINCIPAL/OWNER | THE PALO ALTO GROUP | 790 LUCERNE DRIVE | SUNNYVALE | CA | 94086 | USA |
| DAUGHERTY ROGER | DIRECTOR DATA PROCESSING | AMERICAN SUBSRIPT TV | 8383 WILSHIRE BLVD | BEVERLY HILLS | CA | 90211 | USA |
| DAVISON GERALD W | PROJECT MANAGER | ARGONNE NATIONAL LAB | 9700 SOUTH CASS AVENUE | ARGONNE | IL | 60439 | USA |
| DAVY CHRIS | MANAGER | KEYDATA CORP | 1400 WILSON BLVD | ARLINGTON | VA | 22209 | USA |
| DAVY CHRISTOPHER | PRODUCT MANAGER | KEYDATA CORP | 1400 WILSON BLVD | ARLINGTON | VA | 22209 | USA |
| DEBOK LOWELL W | PROGRAMMER/ANALYST | MITCHELL BROS TRUCK | 3841 N COL BLVD | PORTLAND | OR | 97217 | USA |
| DELMAGE A R(CANCELED) | | GOV'T OF N W TERRITORIES | | YELLOWKNIFE | NW | X0E1H0 | CANADA |
| DELONG DAN G | PROGRAMMER | FEDERAL HOME LOAN BANK | 600 STEWART ST | SEATTLE | WA | 98101 | USA |
| DENEVSY JIM | | SILTON DATA INC | 2407 E 38TH STREET | VERNON | CA | 90058 | USA |
| DINAN DENNIS M | SENIOR PROGRAMMER ANALYS | MORGAN GUARANTY TRUST | 23 WALL STREET | NEW YORK | NY | 10015 | USA |
| DOLAN JR DOUGLAS C | SPECIAL PROJECTS MGR | VICTOR O SCHINNERER | 5028 WISCONSIN AVE NW | WASHINGTON | DC | 20016 | USA |
| DONHAM DAN | SYSTEMS ENGINEER | HEWLETT-PACKARD | 5600 DTC PKWY | ENGLEWOOD | CO | 80110 | USA |
| DOUGLAS GORDON R | FACILITY INFO SYS MGR | HEWLETT-PACKARD | 11000 WOLFE ROAD | CUPERTINO | CA | 95014 | USA |
| DOWLING JAMES F | DP OPERATIONS MANAGER | BOSE CORPORATION | 100 MOUNTAIN ROAD | FRAMINGHAM | MA | 01701 | USA |
| DREILING CHERYL B | COMPUTER SCIENTIST | LAWRENCE LIVERMORE LABOR | P O BOX 808 | LIVERMORE | CA | 94550 | USA |
| DROBNY RONALD G | SYSTEM MANAGER | GATES & SONS | 90 SOUTH FOX | DENVER | CO | 80226 | USA |
| DRYNAN GILBERT W | SYSTEMS DEV MANAGER | BOEING AEROSPACE CO | PO BOX 3999 | SEATTLE | WA | 98124 | USA |
| DUMAS ROBERT J | MARKETING SUPPORT | SYSTEMS RESEARCH INC | 2400 SCIENCE PARKWAY | OKEMOS | MI | 48864 | USA |
| DUMMER DAVID C | PRESIDENT | D C DUMMER & ASSOCIATES | 40 LK LUCERNE CL SE | CALGARY | AL | T2J3H5 | CANADA |

| NAME | TITLE | COMPANY | ADDRESS | CITY | STATE | ZIP | COUNTRY |
|---|---|---|---|---|---|---|---|
| DUMMER SHEILA T | MANAGER FINANCIAL SYSTEM | QUME CORP | 2323 INDUSTRIAL PARKWAY | HAYWARD | CA | 94545 | USA |
| DUNCAN HANNAH F | | | 1601 LEHMBERG BLVD | COLORADO SPRI | CO | 80915 | USA |
| DUNCAN JAMES J | MANAGER OEM SALES | KAPPA SYSTEMS INC | 1409 POTTER DR | COLORADO SPRI | CO | 80909 | USA |
| DUNCOMBE BRIAN C | | MOHAWK COLLEGE | P O BOX 2034 | HAMILTON | ON | L8N3T2 | CANADA |
| DURHAM PAUL H | SYSTEMS ANALYST PROG | PETRO CANADA INC | 400 4 AVE SW | CALGARY | AL | T2P2M7 | CANADA |
| EARLS JOHN D | STAFF ENGINEER | ARTHUR A COLLINS INC | 13601 PRESTON ROAD | DALLAS | TX | 75240 | USA |
| EATON JOHN | DIR COMPUTER SERVICES | LONDON BUS SCH GRAD STUDIES | | LONDON | EN | | ENGLAN |
| ECCLES SHARON L | DATA PROCESSING MANAGER | ECKANKAR | 120 SCOTT DRIVE | MENLO PARK | CA | | USA |
| ECKARD DAVID L | SYSTEM ANALYST | WESTINGHOUSE ELEC C | P O BOX 9175 | PHILADELPHIA | PA | 19113 | USA |
| EDWARDS BENJAMIN E | ASST MGR DATA PROCESSING | UNION CAMP CORP | | FRANKLIN | VA | 23851 | USA |
| EDWARDS BETH | INFO SYSTEMS ADM | GENERAL TELEPHONE-MI | 455 E ELLIS ROAD | MUSKEGON | MI | 49443 | USA |
| EDWARDS CONSTANCE E | MANAGER COMPUTER CTR | ST FRANCIS XAVIER UNIV | PO BOX 67 ST.F.X.U. | ANTIGONISH | NS | B2G1C0 | CANADA |
| EDWARDS JON L | | JENNISON ASSOCIATES | 270 PARK | NEW YORK | NY | 10017 | USA |
| EDWARDS ROBERT M | SPEC ASST FOR DP | NATL CONF ST LEGISLATURES | 444 N CAPITOL ST NW | WASHINGTON | DC | 20001 | USA |
| ELLIOTT HARRY A | MANAGER MIS DEPARTMENT | THE CANADA STARCH CO | 1 PLACE DU COMMERCE | NUNS' ISLAND | QU | H3E1A7 | CANADA |
| ENGBERG TONY I | SYSTEMS ENGINEER | HEWLETT-PACKARD | 19855 GIESENDORFER RD | COLFAX | CA | 95713 | USA |
| ENGLANDER ALICE | D P CONSULTANT | ALICE ENGLANDER | 1966 TITUS STREET | SAN DIEGO | CA | 92110 | USA |
| ENGLANDER WILLIAM R | D P CONSULTANT | WILLIAM R ENGLANDER | 1966 TITUS STREET | SAN DIEGO | CA | 92110 | USA |
| ENGLEBRECHT MICHAEL L | RESEARCH ENGINEER | ARMCO INC | 703 CURTIS STREET | MIDDLETOWN | OH | 45043 | USA |
| ENTIS GLENN M | PROGRAMMER ANALYST T | MORGAN GUARANTY TRUST | 23 WALL STREET | NEW YORK | NY | 10015 | USA |
| ERICSCON GORDY O | ANALYST | 3M CO | 3M CENTER BLDG 224-4N | ST PAUL | MN | 55101 | USA |
| EXCOFFON MARGOT M | PROJECT LEADER | LYNES UNITED SERVICES | 309 2ND AVE SW | CALGARY | AL | T2P0C5 | CANADA |
| FAIRCHILD JAMES B | SUPERVISOR OPERATIONS | DOMTAR INC CHEM GROUP | BP 7212 | MONTREAL | QU | H3C3M3 | CANADA |
| FAIRFIELD STEVE | INFO SYSTEMS ANALYST | GENERAL TELEPHONE | 455 E ELLIS ROAD | MUSKEGON | MI | 49443 | USA |
| FARIA JOHN | MGR COMPUTER SERVICES | PRUDENTIAL REINSURANCE | 213 WASHINGTON ST | NEWARK | NJ | 07101 | USA |
| FARKAS GEORGE J | SYSTEMS ANALYST | CHRYSLER CORPORATION | PO BOX 1919 CIMS:4162725 | DETROIT | MI | 48288 | USA |
| FARRAGHER MICHAEL J | PROG SUPERVISOR | HEWLETT-PACKARD | 175 WYMAN STREET | WALTHAM | MA | 02154 | USA |
| FARRELL JIM G | DIRECTOR MARKET RESEARCH | WM C BROWN PUB CO | 2460 KERPER BLVD | DUBUQUE | IA | 52001 | USA |
| FAWKES GERALD | PROGRAMMER | DEPT OF FISHERIES & OCEANS | BRANDY COVE | ST ANDREWS | NB | E0G2X0 | CANADA |
| FEHR GENE I | D P MANAGER | COHEN FURNITURE CO | 336 SW ADAMS STREET | PEORIA | IL | 61602 | USA |
| FELDMAN DAN | | KEYDATA CORP | 108 WATER ST | WATERTOWN | MA | 02172 | USA |
| FERENSEN DANIEL E | SYSTEMS ANALYST | ADRIA LABORATORIES | P O BOX 16529 | COLUMBUS | OH | 43216 | USA |
| FIELDS JAMES A | VICE PRESIDENT | APEX DATA PROCESSING | 950 SO EASTERN AVE | LOS ANGELES | CA | 90022 | USA |
| FIGUEROA LUIS V | GTE DE PROCESAMIENTO | EJES TRACTIVOS SA | APARTADO POSTAL 14820 | MEXICO | DF | 14820 | MEXICO |
| FINE BRIAN T | TECH STAFF | ESL INC | 495 JAVA DR | SUNNYVALE | CA | 94086 | USA |
| FINNEY BILLIANNA M | PROGRAMMER | SANTABARBARA RESEARCH | 75 COROMAR DRIVE | GOLETA | CA | 93017 | USA |
| FISCHER LEE H | MANAGER DATA PROCESSING | QUME CORP | 2323 INDUSTRIAL PARKWAY | HAYWARD | CA | 94545 | USA |
| FISHER M ROGER | MANAGER COMPUTER SERV | ENVIRONMENTAL ELEMENTS | 3700 KOPPERS ST | BALTIMORE | MD | 21227 | USA |
| FLEET VICTOR W | CONTROLLER | B&S FINANCIAL SERVICES | N COUNTY ROAD 25A | PIQUA | OH | 45356 | USA |
| FLOWERS CURTIS J | DP ANALYST II | SANGAMON STATE UNIV | SHEPHERD ROAD | SPRINGFIELD | IL | 62708 | USA |
| FLOYD TERRY H | ACCNT SYSTEMS ANALYST | THERMON | 100 THERMON DR | SAN MARCOS | TX | 78666 | USA |
| FLYNN DOUGLAS C | D P MANAGER | LEXINGTON HERALD LEAD | 239 WEST SHORT ST | LEXINGTON | KY | 40507 | USA |
| FORSEE ANN T | INSTRUCTOR | ANDERSON COLLEGE | | ANDERSON | IN | 46011 | USA |
| FOSTER RICHARD M | MANAGER SYSTEM S/W | SYSTEM DEVELOPMENT CORP | 2500 COLORADO AVENUE | SANTA MONICA | CA | 90406 | USA |
| FRAGH EDWARD A | SYSTEMS & PROG SUPVSR | CITY OF SANTA MONICA | 1685 MAIN STREET | SANTA MONICA | CA | 90401 | USA |
| FRAME DARYL A | SYSTEM MANAGER | COMARCO INC | 227 W HUENEME ROAD | OXNARD | CA | 93030 | USA |
| FRANSEN CRAIG S | SYSTEM ANALYST | REICHHOLD CHEM | 2340 TAYLOR | TACOMA | WA | 98401 | USA |
| FRASER TOM | | SYSTEMS RESEARCH INC | 2400 SCIENCE PARKWAY | OKEMOS | MI | 48864 | USA |
| FRATUS WILLIAM P | | FUTURA INC | 1714 S CONGRESS | AUSTIN | TX | 78704 | USA |
| FRECH JOHN J | MANAGER OF STIC | ADRIA LABORATORIES | PO BOX 16529 | COLUMBUS | OH | 43216 | USA |
| FREDRICKSON GUNNARD A | PROJECT LEADER | RELIANCE ELECTRIC | 150 CANTERBURY DR | ATHENS | GA | | USA |
| FREDRICKSON STEVE | DISTRICT SALES MGR | UNITED COMPUTING SYS | 4544 POST OAK PL #146 | HOUSTON | TX | 77027 | USA |

| NAME | TITLE | COMPANY | ADDRESS | CITY | STATE | ZIP | COUNTRY |
|---|---|---|---|---|---|---|---|
| FREED JAMES F | INFO SYSTEMS MANAGER | HEWLETT-PACKARD | ROUTE 41 | AVONDALE | PA | 19311 | USA |
| FRENCH DEBORAH J | PROGRAMMER/ANALYST | HEWLETT-PACKARD | 3400 E HARMONY ROAD | FORT COLLINS | CO | 80525 | USA |
| FRYER WILLIAM R | PRESIDENT | S.B.D.P. | 4208 AIRPORT ROAD | CINCINNATI | OH | 45226 | USA |
| FUNK CHRISTOPHER M | PRESIDENT | C M FUNK & CO INC | 22 N 2ND STREET BOX 1249 | LAFAYETTE | IN | 47902 | USA |
| GALLARDO PEDRO A | SYSTEM ENGINEER | UNIVERSIDDO BAJA CAL | OBREGON Y F C CALCOLO | MEXICALI MAJA | | | MEXICO |
| GALUSKY DIANE M | SUPERVISOR APPL DEV | HUGHES AIRCRAFT CO | P O BOX 3310 607/E331 | FULLERTON | CA | 92634 | USA |
| GARDINER JAMES A | MIS MANAGER | PILKINGTON BROS | 685 WARDEN AVE | SCARBOROUGH | ON | M1L3Z8 | CANADA |
| GARDNER LYNN | CUSTOMER RELATIONS | HEWLETT-PACKARD | 5303 STEVENS CREEK BLVD | SANTA CLARA | CA | 95050 | USA |
| GATES BILL | DATA PROCESSING MGR | LONGS DRUG STORES, INC | 141 NORTH CIVIC DR | WALNUT CREEK | CA | 94596 | USA |
| GEER ROSS E | DIRECOTR OF SYS ENGR | TEXAS MUN POWER AGENCY | 2225 E RANDOL MILL RD | ARLINGTON | TX | 76011 | USA |
| GENTRY THOMAS L | MGR ENG'G COMPUTER CTR | MICROWAVE ASSOCIATES | SOUTH AVE | BURLINGTON | MA | 01803 | USA |
| GEWECKE JOHN W | DISTRICT SALES MGR | UNITED COMPUTING SYS | 1901 AVE OF STARS #585 | LOS ANGELES | CA | 90067 | USA |
| GEYER SANFORD A | SYSTEM MANAGER | ROLM CORP | 4900 OLD IRONSIDES DR | SANTA CLARA | CA | 95050 | USA |
| GILCHRIST DON | | MC MASTER UNIVERSITY | 1200 MAIN ST WEST | HAMILTON | ON | L8S4J9 | CANADA |
| GILL RICK | SYSTEMS ANALYST | VANDERBILT UNIVERSITY | 521 KIRKLAND HALL | NASHVILLE | TN | 37235 | USA |
| GIMPLE BILL | R & D MGR HP300 PRG | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| GLOSS GREGORY C | COBOL PROJECT MANAGER | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| GOJENOLA BEN | SR SOFTWARE DESIGNER | WEYERHAEUSER CO | 10TH & A STREETS RMB-2 | TACOMA | WA | 98401 | USA |
| GOLDBERG MICHAEL C | PRESIDENT | FINANCIAL DATA PLANNING | 2670 TIGERTAIL AVENUE | MIAMI | FL | 33133 | USA |
| GOLDMANN ROGER M | SYSTEMS ANALYST | COMARCO INC | 227 W HUENEME ROAD | OXNARD | CA | 93030 | USA |
| GOMEZ VICTOR | SYSTEM ENGINEER | ELECTRONIC DATA SYS | ONE WAYNE MALL | WAYNE | NJ | 07470 | USA |
| GOODMAN ROBERT A | PRESIDENT | PROF COMP SERV | 2821 E 28TH STREET | LONG BEACH | CA | 92648 | USA |
| GOOLSBY MONTY P | SYSTEMS MANAGER | UNION CAMP CORP | P O BOX 570 | SAVANNAH | GA | 31402 | USA |
| GORFINKEL MARTIN | PARTNER | LOS ALTON RESEARCH CENTER | 339 S SAN ANTONIO ROAD | LOS ALTOS | CA | 94022 | USA |
| GOSSELIN JEANNINE G | EDP MANAGER | DOMTAR INC CHEM GROUP | BP 7212 | MONTREAL | QU | H3C3M3 | CANADA |
| GOT TERRENCE | D P MANAGER | SAN JOSE MERCURY-NEWS | 750 RIDDER PARK DR | SAN JOSE | CA | 95190 | USA |
| GRACE JAYNIA | | UNITED COMPUTING SYS | 2525 WASHINGTON | KANSAS CITY | MO | 64108 | USA |
| GRADY MICHAEL K | SYSTEMS CONSULTANT | ILLINOIS DEPT LAW & ORDER | 200 W WASHINGTON | SPRINGFIELD | IL | | USA |
| GREEN ANNABELLE H | SECRETARY/TREASURER | ROBELLE CONSULTING LTD | #130-5421 10TH AVENUE | DELTA | BC | V4M3T9 | CANADA |
| GREEN CHARLES R | DATA PROCESSING MANAGER | ADAMS COUNTY SCHOOLS | 602 E 64TH AVE | DENVER | CO | 80229 | USA |
| GREEN ROBERT M | PRESIDENT | ROBELLE CONSULTING LTD | #130-5421 10TH AVENUE | DELTA | BC | V4M3T9 | CANADA |
| GREENE MARK L | SUPERVISOR | 3M CO | 3M CENTER BLDG 224-4N | ST PAUL | MN | 55101 | USA |
| GREGORY KENT A | ACTUARY/PROGRAMMER | ZISCHKE ORGANIZATION | 1 POST STREET | SAN FRANCISCO | CA | 94104 | USA |
| GRICE FRANK H | PRESIDENT | INTERTEC | 2625 PARK BLVD | PALO ALTO | CA | 94306 | USA |
| GRIFFIN MARY | MARKETING ENGINEER | HEWLETT-PACKARD | 5303 STEVENS CREEK BLVD | SANTA CLARA | CA | | USA |
| GRIFFIN RICK | PRESIDENT | I C S SERVICES, INC | 2131 W EULESS BLVD | EULESS | TX | | USA |
| GRILLOS JOHN M | VICE PRESIDENT | AMERICAN MGMT SYSTEMS | 561 PILGRIM DRIVE SUITE D | SAN MATEO | CA | 94404 | USA |
| GROFF JAMES R | PRODUCT MANAGER | HEWLETT-PACKARD | 1342 STAYNER RD | SAN JOSE | CA | 95121 | USA |
| GROSSCUP LORIN | PROGRAMMER | MERCHANDISING METHODS | 274 BRANNAN STREET | SAN FRANCISCO | CA | 94107 | USA |
| GRUBER DIANNE | | HEWLETT-PACKARD | 1900 GARDEN GODS ROAD | COLORADO SPRI | CO | 80907 | USA |
| GUERRERO JORGE | S E INSTRUCTOR | HEWLETT-PACKARD | 4 CHOKE CHERRY RD | ROCKVILLE | MD | 20850 | USA |
| GUISINGER ERIC L | PROGRAMMER/ANALYST SR | KAISER FOUNDATION | 2005 FRANKLIN ST | DENVER | CO | 80205 | USA |
| GULICK LLOYD R | COMPUTING SPECIALIST | HUGHES AIRCRAFT CO | P O BOX 3310 607/E331 | FULLERTON | CA | 92634 | USA |
| GUNBY D L (CANCELED) | SUPV SYSTEM DESIGN & PRO | BONAR & BEMIS LTD | 2380 MC DOWELL ROAD | BURLINGTON | ON | L7R4A1 | CANADA |
| GUPTA RAMESH | SYSTEMS & OR CONSULTANT | DOMTAR INC CHEM GROUP | BP 7212 | MONTREAL | QU | H3C3M3 | CANADA |
| GURUPRASAD CHOULGERE | DIRECTOR CORP SYSTEMS | DEPT OF SUPPLY CANADA | 11 LAURIER | HULL | QU | | CANADA |
| GWALTHNEY DAVID L | DIRECTOR-CCIS | RUTGERS UNIVERSITY | 311 N FIFTH STREET | CAMDEN | NJ | 08102 | USA |
| HACKMAN LINFORD B | ADMIN SOFTWARE MGR | VYDEC INC | 9 VREELAND ROAD | FLORHAM PARK | NJ | 07932 | USA |
| HAINES OLIN R | DATA PROCESSING MANAGER | WICHITA EAGLE & BEACON | 825 E DOUGLAS | WICHITA | KS | 67202 | USA |
| HAISCH KEN R | PRESIDENT | CASCADE COMPUTER SYSTEMS I | P O BOX 1666 | LONGVIEW | WA | 98632 | USA |
| HALL CRAIG T | PRESIDENT | INFO TRONIC SYSTEMS | 449 HOWARD AVE | HOLLAND | MI | 49423 | USA |
| HALLOCK JIM | | HEWLETT-PACKARD | 815 14 SW | LOVELAND | CO | 80537 | USA |

| NAME | TITLE | COMPANY | ADDRESS | CITY | STATE | ZIP | COUNTRY |
|---|---|---|---|---|---|---|---|
| HALSEY GREGG | SYSTEM ANALYST | BOEING COMPUTER SERVICE | P O BOX 24346 | SEATTLE | WA | 98124 | USA |
| HAMAN VINCE J | DIRECTOR DP | JOHNSON COUNTY | P O BOX 2510 | IOWA CITY | IA | 52240 | USA |
| HAMPTON JEAN K | PROGRAMMER | WM C BROWN PUB COMPANY | 2460 KERPER BLVD | DUBUQUE | IA | 52001 | USA |
| HANSEN WILLIAM R | SR SYSTEM CONSULTANT | | 5560 CAMERFORD DR | DAYTON | OH | 45424 | USA |
| HANSON FRITZ M | LOGISTICS SYSTEM ENGINEE | BOEING AERO SPACE CO | PO BOX 3999 | SEATTLE | WA | 98124 | USA |
| HARBAUGH JERRY E | MGR DATA PROCESSING | ELFAB | 4200 WYLEY POST | ADDISSON | TX | 75001 | USA |
| HARBRON THOMAS R | CHR COMPUTER SCIENCE | ANDERSON COLLEGE | | ANDERSON | IN | 46011 | USA |
| HARMAN LAWRENCE K | GROUP LEADER | LAWRENCE LIVERMORE LAB | P O BOX 808 MC L-389 | LIVERMORE | CA | 94550 | USA |
| HARRIS DAVID W | MANAGER | NBS FINANCIAL SERVICES | 500 W WILSON BDG RD | COLUMBUS | OH | 43285 | USA |
| HATCH JAMES L | DIRECTOR | GM SAGINAW DATA CENTER | 3900 HOLLAND ROAD | SAGINAW | MI | 48605 | USA |
| HATCHER BETH | | HEWLETT-PACKARD | 5301 STEVENS CREEK BLVD | SANTA CLARA | CA | 95050 | USA |
| HAUDERT GERARD A | OPERATIONS SUPERVISOR | BROOKEHAVEN NATL LABOR | 32 BROOKEHAVEN AVE | UPTON | NY | 11973 | USA |
| HAYS CARRY R | SR DATA COMMUNICATION | UNION OIL CO | 461 S BOYLSTON ST | LOS ANGELES | CA | 90017 | USA |
| HEDA SHARAD | MANAGER ADM SYS | VYDEC INC | 9 VREELAND ROAD | FLORHAM PARK | NJ | | USA |
| HEIN NORM | MANAGER INFO SERVICES | WM C BROWN PUB COMPANY | 2460 KERPER BLVD | DUBUQUE | IA | 52001 | USA |
| HEINEN TERRY M | E D P MANAGER | ST CLOUD HOSPITAL | 1406 6TH AVENUE NO | ST CLOUD | MN | 56301 | USA |
| HELLAMS CAROLE | D P PROG ANALYST | WHARTON CO JR COLLEGE | 911 BOLING HIGHWAY | WHARTON | TX | 77488 | USA |
| HENRY WENDELL A | MEMEBERT OF TECH STAFF | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| HERBEL ERIC S | MEDICAL SYS ANALYST | HOECHST-ROUSSEL PHARM | RT 202-206 NORTH | SOMERVILLE | NJ | 08876 | USA |
| HICKS DAVID L | PROGRAMMER ANALYST | PACIFIC MUTUAL | 700 NEWPORT CENTER DRIVE | NEWPORT BEACH | CA | 92660 | USA |
| HINES RELLA M | EXECUTIVE DIRECTOR | HP GEN SYS USERS GROUP | P O BOX 18813 | BALTIMORE | MD | 21240 | USA |
| HISKES GEORGE J | DATA PROCESSING MANAGER | BEALL'S DEPARTMENT ST | 3923 MANATEE AVE W | BRADENTON | FL | 33505 | USA |
| HODSON DICK | DP MANAGER | VALTEC | BOX 2200 | SPRINGVILLE | UT | 84663 | USA |
| HOKE ROBERT D | MARKETING MANAGER | HEWLETT-PACKARD | DISC MEMORY DIVISION | BOISE | ID | 83707 | USA |
| HOLLING ERNEST(CANCEL) | DIRECTOR-DATA SYSTEMS | STORER BROADCASTING | 1177 KANE CONCOURSE | BAY HARBOR IS | FL | 33154 | USA |
| HOLMAN JAMES R | HEAD STATISTICS LAB | OHIO AGRIC R&D CENTER | | WOOSTER | OH | 44691 | USA |
| HOLT WAYNE E | DIRECTOR OR D P | WHITMAN COLLEGE | 345 BOYER AVE | WALLA WALLA | WA | 99362 | USA |
| HOUY DAVID J | FINANCIAL SYSTEMS MGR | SMITH INTERNATIONAL | 4343 VON KARMAN AVE | NEWPORT BEACH | CA | 92660 | USA |
| HUDSON MEL | SYSTEMS ANALYST | COLO DEPT OF EDUC | 201 E COLFAX | DENVER | CO | 80203 | USA |
| HUNTER JOHN C | SYSTEMS SUPERVISOR | OKANAGAN HELICOPTERS | 4391 AGAR DRIVE | VANCOUVER | BC | V7B1A5 | CANADA |
| HUTCHINSON PHILIP L | PRODUCT ENGINEER | HEWLETT-PACKARD FCD | 3400 E HARMONY RD | FORT COLLINS | CO | 80525 | USA |
| HUTTUNEN HEIKKI J | ADP MANAGER | HELSINKI SCHOOL OF ECON | RUNEBERGINK 14-16 | HELSINKI | | 00100S | FINLAN |
| HUXHAM BASIL C | MANAGER DATA PROCESSING | PORT OF VANCOUVER | 1300 STEWART ST | VANCOUVER | BC | V5L4X5 | CANADA |
| HUXHOLD PHIL W | DP MANAGER | MERCHANDISING METHODS | 274 BRANNAN STREET | SAN FRANCISCO | CA | 94107 | USA |
| HWA CHIH S | PRESIDENT | COMPUSYS INC | 789 SHERMAN 560 | DENVER | CO | 80203 | USA |
| ICKLER AL | SENIOR PROGRAMMER | COLO DEPT OF EDUC | 201 E COLFAX | DENVER | CO | 80203 | USA |
| IMBEAU ANDRE | ACTING CHIEF | DEPT REG ECO EXPANSION | 200 RUE PRINCIPAL | HULL | QU | K1A0M4 | CANADA |
| IRIYE DICK | SYSTEMS ANALYST | BOETTCHER & CO | 828 17TH STREET | DENVER | CO | 80202 | USA |
| IZETT CRAIG N | DIRECTOR TECHNOLOGY | MARTIN MARIETTA | 300 EAST JOPPA RD | BALTIMORE | MD | 21204 | USA |
| JACKSON CHARLES W | PRESIDENT | COLLIER-JACKSON ASSOC | 1805 N WESTSHORE BLVD | TAMPA | FL | 33607 | USA |
| JACKSON ELAINE T | MINICOMPUTER ANALYST | HARTFORD INSURANCE | HARTFORD PLAZA | HARTFORD | CT | 06033 | USA |
| JACKSON JOHN A | SYSTEMS ANALYST | DOMINION CONSTRUCTION | 3100 3 BENTALL CENTRE | VANCOUVER | BC | V7X1B1 | CANADA |
| JACOB HARRY O | MGR DISTRIBUTED SYSTEMS | GM SAGINAW DATA CENTER | 3900 HOLLAND ROAD | SAGINAW | MI | 48605 | USA |
| JAMES ROBERT M | ANALYST/PROGRAMMER | SANDIA LABS | P O BOX 5800 | ALBUQUERQUE | NM | 87185 | USA |
| JAMISON CARL L | DATA PROCESSING MANAGER | CRAIG HOSP RESEARCH OFFC | 3460 SO CLARKSON | ENGLEWOOD | CO | 80110 | USA |
| JARAMILLO JR NARCISO | DB ADMIN | BOURNS | 1200 COLUMBIA AVE | RIVERSIDE | CA | 92507 | USA |
| JARVIS RAY D | PROGRAMMER | UMPQUA DATA FACTORY | 727 SE CASS | ROSEBURG | OR | 97470 | USA |
| JEANS KENNETH E | ASST MGR OF DATA PROC | NOOTER CORP | P O BOX 451 | ST LOUIS | MO | 63166 | USA |
| JEFFRIES WILLIAM F | COORDINATOR ICS | STARK COUNTY DEPT ED | 7800 COLUMBUS RD | LOUISVILLE | OH | 44641 | USA |
| JELLIFFE ARLENE M | LEAD ENGINEER | BOEING COMM AIRPLANE CO | BOX 3707 | SEATTLE | WA | 98124 | USA |
| JENSEN HAROLD E | SYSTEMS ANALYST | WILLAMETTER VALLEY CO | 660 MC KINLEY | EUGENE | OR | 97402 | USA |
| JESSEN TIM D | COMPUTER SCIENTIST | LAWRENCE LIVERMORE LAB | P O BOX 808 L-414 | LIVERMORE | CA | 94550 | USA |

| NAME | TITLE | COMPANY | ADDRESS | CITY | STATE | ZIP | COUNTRY |
|---|---|---|---|---|---|---|---|
| JEWEL MARTIN D | MGR COMPUTER SERVICES | NATL SCI DATA CENTER | 1130 E MC DOWELL RD | PHOENIX | AZ | 85006 | USA |
| JOERGER STEVEN G | | ARMAMENT SYSTEMS INC | 712-F N VALLEY ST | ANAHEIM | CA | 92801 | USA |
| JOHNSON BOB | | GOV'T OF N W TERRITORIES | | YELLOWKNIFE | NW | X0E1H0 | CANADA |
| JOHNSON GARY L | INFO SYSTEMS MGR | HEWLETT-PACKARD | 3800 HARMONY ROAD | FORT COLLINS | CO | 80524 | USA |
| JOHNSON PAMELA | SYSTEM ANALYST | BOEING COMPUTER SERVICE | P O BOX 24346 | SEATTLE | WA | 98124 | USA |
| JOHNSON RAYMOND E | HP 3000 PRGM SERV MGR | HEWLETT-PACKARD GSD | 5303 STEVENS CRK BLVD | SANTA CLARA | CA | | USA |
| JOHNSON RON H | DISTRICT SALES MGR | HEWLETT-PACKARD | 5600 DTC PKWY | ENGLEWOOD | CO | 80110 | USA |
| JOHNSTON CHARLES F | DATA PROCESSING MANAGER | FERRIS BSSCHER LOHMA | 339 E 16TH STREET | HOLLAND | MI | 49423 | USA |
| JOHNSTON FRANK H | DIRECTOR DATA PROCESS | MACON TELEGRAPH PUB CO | P O BOX 4167 | MACON | GA | 31208 | USA |
| JOHNSTONE SHIRLEY J | SOFTWARE RELIABILITY ENG | HEWLETT-PACKARD | 19400A HOMESTEAD RD | CUPERTINO | CA | 95014 | USA |
| JONES DANNY A | COMPUTER PROGRAMMER | LAWRENCE LIVERMORE LABOR | P O BOX 808 | LIVERMORE | CA | 94550 | USA |
| JONES MORGAN | | TYMDATA CORP | 44 W JEFFERSON ST | BROWNSVILLE | TX | 78520 | USA |
| JONES THOMAS O | EXEC V P | EPSILON DATA MGMT | 24 NE EXECUTIVE PK | BURLINGTON | MA | 01803 | USA |
| JORGENSON DANIEL M | PRODUCT SUPPORT MGR | HEWLETT-PACKARD | 5303 STEVENS CREEK BLVD | SANATA CLARA | CA | 95050 | USA |
| KAM POLLY | PROGRAMMER/ANALYST | HEWLETT-PACKARD | 5301 STEVENS CREEK BLVD | SANTA CLARA | CA | 95050 | USA |
| KANAGA MIKE | SYSTEM ANALYST | BOEING COMPUTER SERVICE | P O BOX 24346 | SEATTLE | WA | 98124 | USA |
| KASUN ELLEN | | HEWLETT-PACKARD | 815 14 SW | LOVELAND | CO | 80537 | USA |
| KELLY KENT | | FUTURA INC | 1714 S CONGRESS | AUSTIN | TX | 78704 | USA |
| KENDALL JOHN | SYSTEMS PROGRAMMER | SO MISSIONARY COLLEGE | | COLLEGEDALE | TN | 37315 | USA |
| KENFIELD JOHN E | INFO SYS MANAGER | H-P SANTA ROSA DIV | 1400 FOUNTAIN GROVE PK | SANTA ROSA | CA | 95404 | USA |
| KENNEDY DOUGLAS | VICE PRESIDENT | FINANCIAL DATA PLANNING | 2670 TIGERTAIL AVENUE | MIAMI | FL | 331331 | USA |
| KENNEDY JACK | MANAGER SYS & PROG | A G BECKER | 55 WATER STREET | NEW YORK | NY | 10041 | USA |
| KERNKE JUTTA C | PRODUCT MANAGER | HEWLETT-PACKARD | 5303 STEVENS CREEK BLVD | SANTA CLARA | CA | | USA |
| KESSEL JIM (CANCELED) | INDIVIDUAL | | P O BOX 124 | CHAGRIN FALLS | OH | | USA |
| KILLCOMMONS PETER F | GENERAL COST SUPV | NEW YORK TELEPHONE CO | 1095 AVE OF THE AMERICAS | NEW YORK | NY | 10036 | USA |
| KILMON JR LIN E | SENIOR ENGINEER | WESTERN ELECTRIC CO | 2500 BROENING HWY | BALTIMORE | MD | 21224 | USA |
| KING GAIL | PROJECT MANAGER | BOEING COMPUTER SERVICE | P O BOX 24346 | SEATTLE | WA | 98124 | USA |
| KING NEIL R | PROGRAMMING SUPV | SCOTT PAPER LTD | P O BOX 760 | NEW WESTMINST | BC | | CANADA |
| KING STEPHEN E | PROGRAMMER/ANALYST | U S A F | LOS ANGELES AIR FORCE STA | EL SEGUNDO | CA | | USA |
| KIRK TIM L | DIRECTOR INFO SYSTEMS | SACRED HEART GEN HOSP | P O BOX 10905 | EUGENE | OR | 97401 | USA |
| KLEIN JAMES D | SYSTEMS PROGRAMMER | INFORMATION TERMINALS | 323 SOQUEL WAY | SUNNYVALE | CA | 94086 | USA |
| KLEIN THOMAS C | MGR INFO SYSTEMS | HOOVER-NSK BEARING | 5400 S STATE RD | ANN ARBOR | MI | 48106 | USA |
| KLETT DONALD S | DIRECTOR UNIVERSITY LAB | SANGAMON STATE UNIVERSITY | SHEPHERD ROAD | SPRINGFIELD | IL | 62708 | USA |
| KLEVER JOHN | ANALYST/PROGRAMMER | AURORA PUBLIC SCHOOLS | 1085 PEORIA STREET | AURORA | CO | 80011 | USA |
| KNIGHT JR WILLIAM J | MANAGER INFORMATION | MULTIVEST INC | 6452 N FEDERAL HWY | FT LAUDERDALE | FL | 33308 | USA |
| KOBUCHI KENT K | PROJECT SUPERVISOR | BECHTEL CORP | P O BOX 3965 | SAN FRANCISCO | CA | 94119 | USA |
| KOLSTO ELLIOT L | MANAGEMENT ENGINEER | ARGONNE NATIONAL LAB | 9700 SOUTH CASS AVENUE | ARGONNE | IL | 60439 | USA |
| KOMAR JAMES V | DIRECTOR DATA PROCESSING | THE H W WILSON CO. | 950 UNIVERSITY AVENUE | BRONX | NY | 10452 | USA |
| KORNEK KEN | | COMPUTER SERVICE DIVISION | 19310 PRUNERIDGE AVE | CUPERTINO | CA | 95014 | USA |
| KRIGER WINSTON A | MGR TECHNICAL SALES | HOUSTON INSTRUMENT | 8500 CAMERON ROAD | AUSTIN | TX | 78753 | USA |
| KROESEN JACOBUS A | MANAGER O & A | MAKRO INTL | CHURCHILL LAAN 11 | 3525 GVUTRECH | | | NETHER |
| KROPP MICHAEL E | SYSTEMS PROGRAMMER | NORWCH-EATON PHARM | 13-17 EATON AVENUE | NORWICH | NY | 13815 | USA |
| KUEHNER WARREN | S E DISTRICT MGR | HEWLETT-PACKARD | 5600 DTC PKWY | ENGLEWOOD | CO | 80111 | USA |
| KUKUDA, JR. JOHN | VICE PRESIDENT | BUSINESS COMPUTER CONCEPTS | RD #1 BOX 131-B | BURGETTSTOWN | PA | 15021 | USA |
| KWAVNICK MYER | | MONTREAL CHILDRENS HOSPITA | 2300 TUPPER | MONTREAL | QU | H3H1P3 | CANADA |
| LACY LEROY P | COMPUTER SCIENTIST | LAWRENCE LIVERMORE LAB | P O BOX 808 MC L-389 | LIVERMORE | CA | 94550 | USA |
| LAIR STEVE | S E DSITRICT MGR | HEWLETT-PACKARD | 19310 PRUNERIDGE RD | CUPERTINO | CA | 95014 | USA |
| LANCASTER HENRY C | V P ENG DEV & COMP SCI | ENVIRONMENTAL ELEMENTS | 3700 KOPPERS ST | BALTIMORE | MD | 21227 | USA |
| LANGE JOHN M | PROGRAMMER ANALYST | THE TORO COMPANY | 5825 JASMINE STREET | RIVERSIDE | CA | 92504 | USA |
| LARSON ORLAND J | IMAGE PRODUCT MANAGER | HEWLETT-PACKARD | 5303 STEVENS CREEK BLVD | SANTA CLARA | CA | 95050 | USA |
| LARSON TERRY | PROJECT LEADER | LYNES UNITED SERVICES | 309 2ND AVE SW | CALLARY | AL | T2P0C5 | CANADA |
| LASLEY MICHAEL A | MIS MANAGER | HINDERLITER | 1240 N HARVARD | TULSA | OK | 74115 | USA |

| NAME | TITLE | COMPANY | ADDRESS | CITY | STATE | ZIP | COUNTRY |
|---|---|---|---|---|---|---|---|
| LAVIOLA ANTHONY | SYSTEMS MANAGER | N Y TELEPHONE CO | 375 PEARL ST | NEW YORK | NY | 10038 | USA |
| LAW JACK | DATABASE ADMINSTRATOR | HEWLETT-PACKARD SDD | 16399 W BERNARDO DR | SAN DIEGO | CA | 92127 | USA |
| LEE HAROLD | | FAIRFAX CNTY PUBLIC SCHOOL | DATA SERVICES DIVISION | SPRINGFIELD | VA | 22151 | USA |
| LEE LANCE | SENIOR PROGRAMMER | GEORGE WIMPEY CANADA | 80 NORTH QUEEN ST | TORONTO | ON | M8Z2C9 | CANADA |
| LEIGHT BETSY(CANCELED) | DIRECTOR | LEIGHT AND ASSOCIATES | 14200 SHOLES COURT | LOS ALTOSHILL | CA | 94022 | USA |
| LEMLEY JOHN | | HEWLETT-PACKARD | 815 14 SW | LOVELAND | CO | 80537 | USA |
| LESSEY KEN W | ASSOCIATE | DATACOM | 50 WEST STREET | ST HELENS | OR | 97051 | USA |
| LEVIN GREGG B | DESIGN ENGINEER | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| LEWIN ROBERT E | THIRD PARTY PGM MGR | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| LEWIS GERALD | VICE PRESIDENT | APPLIED ANALYSIS INC | 615 SOUTH FLOWER STREET | LOS ANGELES | CA | 90017 | USA |
| LEWIS KERMON | V P DATA PROCESSING | ROBERT JAMES CO | | BIRMINGHAM | AL | | USA |
| LIENARD JAMES B | PROGRAMMER | NATL SCI DATA CENTER | 1130 E MC DOWELL RD | PHOENIX | AZ | 85006 | USA |
| LIGHTHEART TED M | ANALYST/PROGRAMMER | THE WILLAMETTE VALLEY CO | 660 MC KINLGY ST | EUGENE | OR | 97402 | USA |
| LINDSTROM EDWARD R | DIRECTOR-AGRI DATA CT | UNIVERITY OF KENTUCKY | 5107 AG SCI CTR N | LEXINGTON | KY | 40506 | USA |
| LOCHNER CHRIS | SYSTEM MANAGER | FORD BOX 1599B | SOUTHFIELD AT ROTUNDA | DEARBORN | MI | 48121 | USA |
| LOCKHEED ALLAN H | | EXXON MINERAL CO USA | 601 JEFFERSON | HOUSTON | TX | 77601 | USA |
| LONG LYNDA J | D P MANAGER | E S T | 765 CALIFORNIA STREET | SAN FRANCISCO | CA | 94108 | USA |
| LOWRY GLEN H | INFO SYS MANAGER | HEWLETT-PACKARD | P O BOX 15 | BOISE | ID | 83707 | USA |
| LUFT MARKUS F | SYSTEMS & PROG SUPRV | DOMTAR CONSTRUCTION MTRLS | 2001 UNIVERSITY ST | MONTREAL | QU | HSA2A6 | CANADA |
| LUISI WILLIAM F | SOFTWARE ANALYST | UNION CAMP CORP | 1600 VALLEY RD | WAYNE | NJ | 07470 | USA |
| LUITHLE WILLIARD H | COMPUTER OPS SUPV | KAISER FOUNDATION | 2005 FRANKLIN | DENVER | CO | 80205 | USA |
| LULICH LEO J | PROGRAMMER | NORTHERN SPECIALTY | 6635 N BALTIMORE | PORTLAND | OR | 97203 | USA |
| LUMB ARTHUR C | CONSULTANT | PROCTER & GAMBLE CO | 7162 READING RD | CINCINNATI | OH | 45208 | USA |
| LUNDBERG ERIK | SCIENTIFIC PROGRAMMER | CASS UNIV OF WASH | 1107 NE 45TH ROOM 530 | SEATTLE | WA | 98105 | USA |
| LYNN DEAN E | COMPUTER SCIENTIST | LAWRENCE LIVERMORE LABOR | P O BOX 808 | LIVERMORE | CA | 94550 | USA |
| MACHIN JOHN | MANAGING PARTNER | INTERCOMP SERVICES | 459 COLLINS STREET | MELBOURNE | VI | 3000 | AUSTRA |
| MAGDALENO JESUS | GTE DE PROYECTOS | INFORMATICA DESC, SC | THIERS 248 ANZUREZ | MEXICO | DF | | MEXICO |
| MAGNUS ANN M | SUPV COMPUTER OPER | MULTNOMAH COUNTY ESD | 220 SE 102 | PORTLAND | OR | 97216 | USA |
| MAHONEY LARRY | SUPERVISOR SEATTLE COMP | R W BECK & ASSOCIATES | 200 TOWER BUILDING | SEATTLE | WA | 98101 | USA |
| MAIER EDWARD F | INFO SYSTEMS EXECUTIVE | IL LAW ENFORCEMENT CO | 120 S RIVERSIDE PLAZA | CHICAGO | IL | 60606 | USA |
| MALACHOWSKI ERNEST S | CONTROLLER | GRANVILLE-PHILLIPS CO | 5675 ARAPAHOE | BOULDER | CO | 80303 | USA |
| MALUS JOSEPH T | SENIOR PROGRAMMER ANALYS | MORGAN GUARANTY TRUST | 37 WALL STREET | NEW YORK | NY | 10015 | USA |
| MANEWAL SANDRA S | SYSTEMS MANAGER | LIBERTY COMMUNICATIONS INC | 2225 COBURG ROAD | EUGENE | OR | 97401 | USA |
| MANIES RALPH G | CUSTOMER RELATIONS | HEWLETT-PACKARD GSD | 5303 STEVENS CREEK BLVD | SANTA CLARA | CA | 95050 | USA |
| MARCHESE BUZZ | SYSTEM MANAGER | COMPUTER COMPOSITION SALES | 2640 PINE ST | ST LOUIS | MO | 63103 | USA |
| MARQUEZ MATEO(CANCEL) | FIELD ENGINEER | HEWLETT-PACKARD MEXICO | AVENIDA PERIFERICO SUR 6501 | TEPEPAN | XO | 22 | MEXICO |
| MARSICEK R G | | BOEING COMPUTER SERVICE | PO BOX 24346 | SEATTLE | WA | 98124 | USA |
| MARTIN STEVE T | SYSTEMS ANALYST | SANDIA LABORATORIES | DIVISION 2627 | ALBUQUERQUE | NM | 87185 | USA |
| MASSLINGER BOB | SYSTEM ANALYST | BOEING COMPUTER SERVICE | P O BOX 24346 | SEATTLE | WA | 98124 | USA |
| MATHIAS TIMOTHY E | | ANDERSON COLLEGE | | ANDERSON | IN | 46011 | USA |
| MATT RICHARD G | HARDWARE PLANNER | GENERAL MILLS INC | 9200 WAYZATA BLVD | MINNEAPOLIS | MN | 55426 | USA |
| MAUS JOHN A | SYSTEMS ANALYST | M.H. GOLDEN CO | 123 CAMINO DE LA REINA | SAN DIEGO | CA | 92108 | USA |
| MAY NORMAN F | SR SYSTEMS ANALYST | IL LAW ENFORCEMENT CO | 120 S RIVERSIDE PLAZA | CHICAGO | IL | 60606 | USA |
| MAYHEW JOHN | | AMERICAN SUBSRIP TV | 8383 WILSHIRE BLVD | BEVERLY HILLS | CA | 90211 | USA |
| MC AFEE WILLIAM K | PRESIDENT | FUTURA INC | 1714 S CONGRESS | AUSTIN | TX | 78704 | USA |
| MC CLAIN MALCOLM E | MANAGER DATA PROCESSING | NORTH IDAHO COLLEGE | 1000 W GARDEN AVENUE | COEUR D' ALEN | ID | 83814 | USA |
| MC CLURE HERSCHEL D | MGR SYSTEMS DEPT | UNION CAMP CORP | | FRANKLIN | VA | 23851 | USA |
| MC CORMICK DOUGLAS B | MGR INFORMATION SYS | REPUBLIC GEOTHERMAL | 11823 E SLAUSON #1 | SANTA FE SPRI | CA | 90670 | USA |
| MC CRACKEN ED | GEN MGR G S D | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| MC CREA ROBERT B | CHIEF FINANCIAL OFFICER | PORT OF VANCOUVER | 1300 STEWART ST | VANCOUVER | BC | V5L4X5 | CANADA |
| MC GEOY JOHN G | MGR DATA SERVICES | WESTINGHOUSE ELEC CORP | P O BOX 8839 | PITTSBURGH | PA | 15221 | USA |
| MC GRATH JOSEPH | | TYMDATA CORP | 44 W JEFFERSON ST | BROWNSVILLE | TX | 78520 | USA |

| NAME | TITLE | COMPANY | ADDRESS | CITY | STATE | ZIP | COUNTRY |
|---|---|---|---|---|---|---|---|
| MC INNIS, JR. A MARVI | | MINI/MICRO SYSTEMS INC | 3315 N SHARTEL AVE | OKLAHOMA CITY | OK | 73118 | USA |
| MC LEMORE JIM | SYSTEMS MANAGER | DEPT OF ENERGY | 1330 BROADWAY | OAKLAND | CA | | USA |
| MC LEOD PAT | PROGRAMMER | SCOTT PAPER LTD | P O BOX 760 | NEW WESTMINST | BC | | CANADA |
| MC MURRAY JACK M | DP MANAGER | DOMINION CONSTRUCTION | 3100 3 BENTALL CENTRE | VANCOUVER | BC | V7X1B1 | CANADA |
| MC SHANE MICHAEL G | ASSOC DIRECTOR COMPUTING | ASSOC AMERICAN MED COLLEGE | ONE DUPONT CIRCLE SUITE 200 | WASHINGTON | DC | 20036 | USA |
| MECHAM DOUGLAS J | SYSTEMS COORDINATOR | HUGHES AIRCRAFT CO | P O BOX 3310 601-4219 | FULLERTON | CA | 92634 | USA |
| MEDD RANDY | HP3000 SYSTEMS MGR | BOETTCHER & CO | 828 17TH STREET | DENVER | CO | 80202 | USA |
| MEINZ MICHAEL J | SUPERVISOR,TECH SERVS | GENERAL MILLS, INC | PO BOX 1113 | MINNEAPOLIS | MN | 55440 | USA |
| MELVIN PETER S | DIRECTOR OF MARKETING | CMC ASSOCIATES | 3 MARGARET ST | BOSTON | MA | 02021 | USA |
| MENOLD BEN | S E DISTRICT MGR | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| MERSHON ROBERT C | PROJECT MANAGER | ITT FINANCIAL | P O BOX 250 | CHIPPEWA FALL | WI | 54729 | USA |
| METZNER BERNIE | OPERATIONS MGR | PILKINGTON GLASS LTD | 685 WARDEN AVE | TORONTO | ON | M1L3X7 | CANADA |
| MEYER DIANNE M | OPERATIONS COORDINATOR | PROCTER & GAMBLE CO | 6105 CENTER HILL RD | CINCINNATI | OH | 45220 | USA |
| MEYER JAMES J | COMPUTER OPERATIONS | COPCO | 4905 LIMA STREET | DENVER | CO | 80239 | USA |
| MEYERS BARRY | NATIONAL SALES MGR | UNITED COMPUTING SYS | 1901 AVE OF STARS #585 | LOS ANGELES | CA | 90067 | USA |
| MEYERS LAWRENCE | DIRECTOR OF MARKETING | SYSTEMS RESEARCH INC | 2400 SCIENCE PARKWAY | OKEMOS | MI | 48864 | USA |
| MILLARD MICHAEL J | PROGRAMMING SUPERVISOR | OKANAGAN HELICOPTERS | 4391 AGAR DRIVE | VANCOUVER | BC | V7B1A5 | CANADA |
| MILLER DANIEL J | SYSTEMS ANALYST | ENVIRONMENTAL ELEMENTS | 3700 KOPPERS ST | BALTIMORE | MD | 21227 | USA |
| MILLER LEROY M | D P MANAGER | NORTHERN SPECIALTY SALES | 6635 N BALTIMORE | PORTLAND | OR | 97203 | USA |
| MILLER STEPHEN L | COMP PROD SUPVR II | IL DEPT OF CORRECTION | 200 W WASHINGTON | SPRINGFIELD | IL | | USA |
| MILLER WILLIAM J | MGR CORP SYS PLANNING | THE BOVAIRD SUPPLY CO | 823 S DETROIT | TULSA | OK | 74102 | USA |
| MINOR TERRY | SYSTEMS MGR | DONNELLY MIRRORS INC | 49 W 3RD ST | HOLLAND | MI | 49423 | USA |
| MOCK MADISON O | MGR INFO SYS BAG DIV | UNION CAMP CORP | P O BOX 1825 | SAVANNAH | GA | 31402 | USA |
| MOIRAO DAVID E | PROGRAMMING MANAGER | LONGS DRUG STORES, INC | 141 NORTH CIVIC DR | WALNUT CREEK | CA | 94596 | USA |
| MONAHAN WILLIAM | DEVELOPMENT MGR | 3 M | 3M CTR 223-5N | ST PAUL | MN | 55001 | USA |
| MOODY JERRY R | SYSTEMS SUP | DEPT OF ARMY | BLDG 12500 ATTN DXRMC-C-C | FORT LEE | VA | 23801 | USA |
| MOORE GEORGE E | OPERATIONS MANAGER | INTERACTIVE APPL INC | 505 HAMILTON AVE #103 | PALO ALTO | CA | 94301 | USA |
| MOORE ROBERT E | PRESIDENT | EDUCATIONAL COMPUTER SYS | 1313 KEMPER RD | CINCINNATI | OH | 45246 | USA |
| MORAIN OLEN | COMPUTER SERVICES DIV | HEWLETT-PACKARD | 19310 PRUNERIDGE AVE | CUPERTINO | CA | 95014 | USA |
| MORRISON JAMES C | MANAGER D P | SUN HYDRAULICS CORP | 1817 57TH STREET | SARASOTA | FL | 33580 | USA |
| MOWER MONTE J | SYSTEM MANAGER | PROVO SCHOOL DISTRICT | 280 WEST 800 NORTH | PROVO | UT | 84601 | USA |
| MULLER MEREDITH A | PROGRAMMER/ANALYST | BOEING | P O BOX 3707 MS 3N-03 | SEATTLE | WA | 98124 | USA |
| MULVIHILL GARY | | R SHRIVER ASSOCIATES | 1530 CHESTNUT SUITE 714 | PHILADELPHIA | PA | 19102 | USA |
| MURPHY DONALD C | SYSTEM MANAGER | BOEING COMPUTER SERVICES | P O BOX 3707 M/S36-01 | SEATTLE | WA | 98124 | USA |
| MURPHY JR ROBERT | DIRECTOR ADMINISTRATION | TEXAS MUNICIPAL POWER | 600 ARLINGTON DOWNS | ARLINGTON | TX | 76011 | USA |
| NAGEL PATRICK M | SYSTEMS PROGRAMMER | UNION OIL COMPANY | 135TH ST & NEW AVE | LEMONT | IL | 60439 | USA |
| NEAL RANDOLPH H | PRESIDENT | AUTOMATED BUSINESS SV | 1649 W BROAD ST | RICHMOND | VA | | USA |
| NEIBERGS GEORGE J | IS MANAGER | ESB-EXIDE | 101 GIBRALTOR RD | HORSHAM | PA | 19044 | USA |
| NELSON MARLYS | | UNIVERSITY OF WISCONSIN | | RIVER FALLS | WI | 54022 | USA |
| NEMETH LOUIS E | DIRECTOR ENG COMP CTR | PHILA WATER DEPT | 1270 MSB | PHILADELPHIA | PA | 19107 | USA |
| NEUMYER RICHARD D | SENIOR ENGINEER | WESTERN ELECTRIC CO | 2500 BROENING HWY | BALTIMORE | MD | 21224 | USA |
| NEWELL RUSSELL L | SYSTEMS ANALYST | VOFM-DEARBORN | 4901 EVERGREEN | DEARBORN | MI | 48128 | USA |
| NEWMAN ALAN T | INFO SERVICES MGR | TOWER MANAGEMENT | 1779 TRIBUTE ROAD #H | SACRAMENTO | CA | 95815 | USA |
| NICHOLS MARTIN D | TECHNICAL ANALYST | UNION CAMP CORP | 1600 VALLEY RD | WAYNE | NJ | 07470 | USA |
| NOLAN VINCENT | | UNITED MC GILL CORP | 2400 FAIRWOOD AVE | COLUMBUS | OH | 43207 | USA |
| NONAMAKER LAURA S | PROGRAMMER/ANALYST | UNION OIL COMPANY | 461 BOYLSTON RM M327 | LOS ANGELES | CA | 90017 | USA |
| NORRIS BOB | PROGRAMMER ANALYST | SAN JOSE MERCURY-NEWS | 750 RIDDER PARK DR | SAN JOSE | CA | 95190 | USA |
| NORTH CARL H | DIRECTOR COMPUTER CTR | THOMAS NELSON COM COL | PO BOX 9407 | HAMPTON | VA | 23670 | USA |
| OCHI YOSHIAKI | EDP MANAGER | MATSUSHITA ELEC INDUST CO | #2 MATSUSHITA MACHI | MORIGUCHI-SHI | OS | | JAPAN |
| OLSEN MARY ANN(CANCEL) | PROGRAMMER | MITCHELL BROS TRUCK | 3841 N COLUMBIA BLVD | PORTLAND | OR | 97217 | USA |
| OLSON BRIAN | SENIOR ANALYST | APPLIED ANALYSIS INC | 615 SOUTH FLOWER STREET | LOS ANGELES | CA | 90017 | USA |
| OMI G KEVIN | MANAGER | WELLS FARGO BANK | 420 MONTGOMERY ST | SAN FRANCISCO | CA | 94111 | USA |

| NAME | TITLE | COMPANY | ADDRESS | CITY | STATE | ZIP | COUNTRY |
|---|---|---|---|---|---|---|---|
| ONEY JAMES B | DATA PROCESSING MANAGER | RCM | ONE MARKET PLAZA 3910 | SAN FRANCISCO | CA | 94105 | USA |
| ONIZUKA SANDRA M | SYSTEM MANAGER | CANCER CENTER OF HI | 1150 SOUTH KING ST | HONOLULU | HA | 96814 | USA |
| ONYX ROBERT P | D P OPER MANAGER | FOSECO INC | 20200 SHELDON ROAD | BROOKPARK | OH | 44142 | USA |
| OSBORNE LEE K | DEVELOPMENT ENGINEER | HEWLETT-PACKARD | 1046 S WINCHESTER #9 | SAN JOSE | CA | 95128 | US |
| OSCARSSON HANS O | MANAGER | AB TRAV OCH GALOPP | BOX 1733 | STOCKHOLM | SW | 11187 | SWEDEN |
| OUELLETTE MARC E | SOFTEWARE SPECIALIST | INCO LIMITED | 1 FIRST CANADIAN PL | TORONTO | ON | M5X1C4 | CANADA |
| PACHALY FRED A | PROGRAMMER | PURITAN INSURANCE CO | 1515 SUMMER ST | STAMFORD | CT | 06905 | USA |
| PAGE JOHN | PROJECT MANAGER | HEWLETT-PACKARD | 5303 STEVEN CREEK | SANTA CLARA | CA | 95050 | USA |
| PAPPAS STEVE |  | AMERICAN SUBSCRIP TV | 8383 WILSHIRE BLVD | BEVERLY HILLS | CA | 90211 | USA |
| PARELLA MIKE | PRESIDENT | DECISION STRATEGY | 708 THIRD AVENUE | NEW YORK | NY | 10017 | USA |
| PARKER KENNETH | DIRECTOR E D P | PARISIAN INC | 1101 26TH STREET N | BIRMINGHAM | AL | 35234 | USA |
| PASHAK JAMES W | COMPUTER SPECIALIST | DEPT OF ENERGY | 1333 BROADWAY | OAKLAND | CA | 94161 | USA |
| PENNALA ERIC M | MANAGER PROG & OPS | SPRECKELS SUGAR DIV | 50 CALIFORNIA ST | SAN FRANCISCO | CA | 94111 | USA |
| PERKINS ALAN L | SYSTEMS MGR/PGMR | WHITE HOUSE COMM | THE WHITE HOUSE | WASHINGTON | DC | 22070 | USA |
| PETERS HAROLD J | PRESIDENT | EDUCATIONAL SOFTWARE PRODU | 9 GEORGETOWN CIRCLE | IOWA CITY | IA | 52240 | USA |
| PETERSON DON | SYSTEMS PROGRAMMER | US CIVIL SERV COMM | 4685 LOG CABIN DR | MACON | GA | 31210 | USA |
| PETERSON DONALD C | MGR SYSTEMS ANALYST | COLO DEPT OF HIGHWAYS | 4201 E ARKANSAS | DENVER | CO | 80222 | USA |
| PETERSON MERRILL A | STAFF ENGINEER | JEFFERSON CHEMICAL | P O BOX 847 | PORT NECHES | TX | 77651 | USA |
| PETTERCHAK JOHN J | INFO SYST EXECUTIVE | IL DEPT OF CORRECTION | 200 N WASHINGTON | SPRINGFIELD | IL |  | USA |
| PHILLIPS TERRY | DEVELOPMENT MANAGER | SACRED HEART GEN HOSP | P O 10905 | EUGENE | OR | 97401 | USA |
| PICK V MICHAEL | RESEARCH COMPUTER MGR | GRUMMAN AERO SPACE CORP | A 08-35 | BETHPAGE | NY | 11714 | USA |
| PIEKARSKI DAVID M | SUPVR COMPUTER UTILIZ | CHRYSLER CORPORATION | PO BOX 1919 CIMS:4162725 | DETROIT | MI | 48288 | USA |
| PIERCE ANTHONY | SR SYSTEMS ANALYST | BOEING AEROSPACE | P O BOX 3999 | SEATTLE | WA | 98124 | USA |
| PIPER MICHAEL |  |  | 19601 NORDHOFFF | NO RIDGE | CA |  | USA |
| PLEASANTS JOYCE B | DIRECTOR DATA PROCESSING | AURORA PUBLIC SCHOOLS | 1085 PEORIA STREET | AURORA | CO | 80011 | USA |
| POLAKOWSKI KEN | GRP MGR SOFTWARE SUPPORT | VYDEC INC | 9 VREELAND RD | FLORHAM PARK | NJ |  | USA |
| POLHEMUS JAN R | MGR BUSINESS SYSTEMS | AUSTIN INFORMATION SYS | 800 SW 16TH STREET | RENTON | WA | 98055 | USA |
| POLO FRANK | SYSTEMS SPECIALIST | NORTHROP CORP | 2301 WEST 120 STREET | HAWTHORNE | CA | 90250 | USA |
| PORTERFIELD STEPHEN T | HP3000 SYSTEM MANAGER | BECHTEL CORPORATION | P O BOX 3965 | SAN FRANCISCO | CA | 94119 | USA |
| POWERS DENNIS D | DIRECTOR INFO SERVICE | WM C BROWN PUB COMPANY | 2460 KERPER BLVD | DUBUQUE | IA | 52001 | USA |
| PREDMORE LIONEL A | INFORMATION SYSTEMS SUPV | CITY OF TEMPE | 31 E FIFTH BOX 5002 | TEMPE | AZ | 85281 | USA |
| PRELLE JEAN | PROFESSOR | ECOLE SUPERIEURE DE COMMER | 23 ROUTE DE DARDILLY | ECULLY |  |  | FRANCE |
| PRICE RICHARD J | MGR TECH SUPPORT | NATIONWIDE FINANCE | 700 OFFICE PARKWAY | CREVE COEUR | MO | 63141 | USA |
| PRICE ROGER | SYSTEMS ANALYST | THE BOVAIRD SUPPLY CO | 823 S DETROIT | TULSA | OK | 74102 | USA |
| PRILL BOB (CANCELED) | OPERATIONS MANAGER | HEWLETT-PACKARD SDD | 16399 W BERNARDO DR | SAN DIEGO | CA | 92127 | USA |
| PRITCHARD G BRIAN | SR PROGRAMMER | CANADA BLDG MATERIALS | 55 INDUSTRIAL ST | TORONTO | ON | M4G3W9 | CANADA |
| PROCHNOW NEAL |  | UNIVERSITY OF WISCONSIN |  | RIVER FALLS | WI | 54022 | USA |
| PUTEGNAT MICHAEL(CANC) | PRESIDENT | CONTROL SYSTEMS | 44 W JEFFERSON ST | BROWNSVILLE | TX | 78520 | USA |
| RADOFF IRVING | SUPVR SYS/PROG | VSI CORPORATION | 8463 HIGUERA ST | CULVER CITY | CA | 90230 | USA |
| RASMUSSEN BENT | VICE PRESIDENT | R SHRIVER ASSOCIATES | 120 LITTLETON ROAD | PARSIPPANY | NJ | 07054 | USA |
| RAUH JOSEPH E | TECHNICAL SHOWMAN | ISSCO | 4186 SORRENTO VALLEY BLVD | SAN DIEGO | CA | 92121 | USA |
| RAWSON TOM | SYSTEMS PROGRAMMER | CASS UNIV OF WASH | 1107 NE 45TH ROOM 530 | SEATTLE | WA | 98105 | USA |
| REGO F ALFREDO | PROFESSOR | UNIV F MARROQUIN | 6A AVE 0-28 ZONA 10 | GUATEMALA |  |  | GUATEM |
| REITHNER JR ROBERT M | MANAGER | N.E.R.A. | 80 BROAD STREET | NEW YORK | NY | 10004 | USA |
| RICE ROBERT J | SYSTEMS ANALYST | ADAMS COUNTY SCHOOLS | 602 E 64TH AVE | DENVER | CO | 80229 | USA |
| RICKARD C ALLEN | MGR SYSTEMS DEVELOPMENT | NATIONWIDE FINANCE | 700 OFFICE PARKWAY | CREVE COEUR | MO | 63141 | USA |
| RIEGER DENNIS E | PRODUCT MANAGER - MPE | HEWLETT-PACKARD | 5303 STEVENS CREEK BLVD | SANTA CLARA | CA | 95050 | USA |
| RITCHIE CLIFFORD E | D P MANAGER | BENEFIT TECHNOLOGY | 900 LAFAYETTE STREET | SANTA CLARA | CA | 95050 | USA |
| ROBERTS RICHARD A | MGR SYSTEM & PROGRAMMING | B&S FINANCIAL SERVICES | N COUNTY ROAD 25A | PIQUA | OH | 45356 | USA |
| ROBERTSON DENNIS L | MGR DIST PROC TECH SP | WEYERHAEUSER CO | 10TH & A STREETS RMB-1 | TACOMA | WA | 98401 | USA |
| ROBINSON JOEL H | MGR INFORMATION SYS | GEORGE WIMPEY CANADA | 80 NORTH QUEEN ST | TORONTO | ON | M8Z2C9 | CANADA |
| RODGER JOHN D |  | NORTHERN TELECOM | 8200 DIXIE RD | BRAMPTON | ON | L6V2M6 | CANADA |
| RODRIGUEZ DAN R | PROJECT LDR SUPPLY | STANDARD OIL CO | 1090 GUILDHALL BLDG | CLEVELAND | OH | 44115 | USA |

| NAME | TITLE | COMPANY | ADDRESS | CITY | STATE | ZIP | COUNTRY |
|---|---|---|---|---|---|---|---|
| ROGERS THOMAS D | ASSISTANT DIRECTOR | NORFOLK STATE COLLEGE | 2401 CORPREW AVE | NORFOLK | VA | 23504 | USA |
| ROSENBERG IVAN M | GENERAL PARTNER | SYSTEMS DESIGN ASSOC | P O BOX 1144 | SAN LUIS OBIS | CA | 93406 | USA |
| ROSS PATRICK D | SR MEMBER TECHNICAL STAF | BSL INC | 495 JAVA | SUNNYVALE | CA | 94086 | USA |
| ROUSSEAU ALLEN F | VP PRODUCT DEV | KEYDATA CORP | 108 WATER ST | WATERTOWN | MA | 02172 | USA |
| ROWE THOMAS C | V P SYSTEMS | NATIONWIDE FINANCE | 700 OFFICE PARKWAY | CREVE COEUR | MS | 63141 | USA |
| RUSSELL BRIAN G | PROGRAMMER/ANALYST | WEST FRASER MILLS | BOX 6000 | QUESNEL | BC | V2J3J5 | CANADA |
| RUSSELL JIM | PRESIDENT | COMPUTER SERVICES CORP | 75 MANHATTAN DR #107 | BOULDER | CO | 80303 | USA |
| RUSSELL KENT A | PRESIDENT | NATIONAL COMPUTER CORP | 3000 34TH ST | METAIRIE | LA | 70001 | USA |
| RUTHERFORD CLYDE R | SUPV MSS COMPUTING | BOEING COMMERCIAL AIR | P O BOX 3707 M/S 5F-09 | SEATTLE | WA | 98124 | USA |
| RYAN GEOFFREY |  | CONTROL SYSTEMS | 44 W JEFFERSON ST | BROWNSVILLE | TX | 78520 | USA |
| SALINS GARY E | DATA PROCESSING MANAGER | DARE PAFCO, INC | 11353 REED HARTMAN HY | CINCINNATI | OH | 45040 | USA |
| SARBAUGH JAY C | VICE PRESIDENT | SMALL BUSINESS DATA PROC | 4208 AIRPORT ROAD | CINCINNATI | OH | 45226 | USA |
| SCHICK JOHN A | PROGRAMMER | WM C BROWN PUB COMPANY | 2460 KERPER BLVD | DUBUQUE | IA | 52001 | USA |
| SCHLOSSER JR ROBERT J | COMPUTER SCIENTIST | PEPCO | 1900 PENNA AVE | WASHINGTON | DC | 20068 | USA |
| SCHULER KURT J | DIR / COMP SERVICES | UNIVERSITY OF DALLAS |  | IRVING | TX | 75061 | USA |
| SCHWARTZ RICHARD T | PRINCIPAL | AMERICAN MANAGEMENT SYSTEM | 561 PILGRIM DRIVE SUITE D | SAN MATEO | CA | 94404 | USA |
| SCHWARTZ RICK A | DIST CE MANAGER | HEWLETT-PACKARD | 6877 GOREWAY DRIVE | MISSISSAUGA | ON | L4V1M8 | CANADA |
| SCHWARZ RAYMOND T | MINI-OPERATIONS MGR | HEWLETT-PACKARD | 1501 PAGE MILL RD | PALO ALTO | CA | 94304 | USA |
| SCOTT GEORGE B | MANAGER | REICHHOLA CHEM | 2340 TAYLOR WAY | TACOMA | WA | 98401 | USA |
| SCROGGS ROSS E |  | ALTER*ABILITY | 534 ROSAL AVENUE | OAKLAND | CA | 94610 | USA |
| SEIFRIED W BRIAN | SUPERVISOR SYSTEMS APPL | BONAR & BEMIS LTD | 2380 MC DOWELL ROAD | BURLINGTON | ON | L7R4A1 | CANADA |
| SELLERS HARRY P | ADM DP MANAGER | VA WESTERN COM COL | 3095 COLONIAL AVE | ROANOKE | VA | 24015 | USA |
| SEVEBORG KARL E | CIVIL ENGINEER | AB BOFORS | BOX 500 | BOFORS |  | S69020 | SWEDEN |
| SEWELL DAVID | DIRECTOR OF DATA PROCESS | SII SERVCO | PO BOX 880 | GARDENA | CA | 90247 | USA |
| SHAULIS MICHAEL |  | FAIRFAX CNTY PUBLIC SCHOOL | DATA SERVICES DIVISION | SPRINGFIELD | VA | 22151 | USA |
| SHEFFIELD REBECCA L | STUDENT | ANDERSON COLLEGE |  | ANDERSON | IN | 46011 | USA |
| SHELLEY NANCY | SYSTEMS PROGRAMMER | SCOTT PAPER LTD | P O BOX 760 | NEW WESTMINST | BC |  | CANADA |
| SHOUP BRYAN | DATA PROCESSING MGR | SII DYNA-DRILL | 1771 DEERE AVENUE | IRVINE | CA | 92713 | USA |
| SHROADS, JR JAMES C | DIRECTOR COMP OPER | NPD REASEARCH, INC | 15 VERBENA AVENUE | FLORAL PARK | NY | 11001 | USA |
| SHUMATE D CRAIG | MANAGER C S D | WARREN & VAN PRAAG | 1276 NORTH WARSON RD | ST LOUIS | MO | 63132 | USA |
| SILVER GAYE L | SYSTEMS ANALYST | TRW COLORADO ELECT | 3450 N NEVADA AVE | COLORADO SPRI | CO | 80907 | USA |
| SIMONSEN J LAWRENCE | COMPUTER SYSTEMS ENGR | VALTEK INC | MOUNTAIN SPRINGS PRKWY | SPRINGVILLE | UT | 84663 | USA |
| SIMS RICH | EXEC V P | A T I | 3532 S HILLCREST DR #3 | DENVER | CO |  | USA |
| SINHA DEEPAK | PROGRAMMER | MC MASTER UNIVERSITY | 1280 MAIN ST W | HAMILTON | ON | L8S4J9 | CANADA |
| SISOIS MIKES N | DIRECTOR INFO SYSTEMS | UNIV OF SANTA CLARA | BANNAN HALL 113 | SANTA CLARA | CA | 95053 | USA |
| SJOGREN DAVID R | SYSTEM SPECIALIST | GENERAL MILLS | PO BOX 1113 | MINNEAPOLIS | MN | 55440 | USA |
| SLATER TED | SALES REP | HEWLETT-PACKARD | 10691 SHELLBRIDGE WAY | RICHMOND | BC | V6X2W7 | CANADA |
| SLOAN DAVID L | CONSULTANT | COLLIER-JACKSON ASSOC | 1805 N WESTSHORE BLVD | TAMPA | FL | 33607 | USA |
| SMART JOHN E | VICE PRESIDENT | SILTON DATA INC | 2407 E 38TH STREET | VERNON | CA | 90058 | USA |
| SMELSER LINDA C | ASST GENERAL MGR | STATE CONTROLLERS OFFICE | 504 E MUSSER | CARSON CITY | NV | 89701 | USA |
| SMITH BARRY |  | PRUDENTIAL INSURANCE | 213 WASHINGTON ST | NEWARK | NJ | 07101 | USA |
| SMITH HOWARD J | LAB MANAGER | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| SMITH JR HARRY R | DIRECTOR, COMPUTER CTR | HOWARD COMMUNITY COLLEGE | LITTLE PATUXENT PKWY | COLUMBIA | MD | 21044 | USA |
| SMITH RAYMOND E | DATA PROCESSING MANAGER | POWERS REGULATOR | RR #1 | BEETON | ON | L0G1A0 | CANADA |
| SMITH ROGER | DP MANAGER | DONNELLY MIRRORS INC | 49 W 3RD ST | HOLLAND | MI | 49423 | USA |
| SMITH TERRY B | CHIEF OF DATA PROCESSING | MILO BEAUTY SUPPLY CO | 4670 ALLEN ROAD | STOW | OH | 44224 | USA |
| SMEED JAMES E | CORPORATE MGR D P | RELIANCE ELECTRIC | P O BOX 5065 STATION B | GREENVILLE | SC | 29606 | USA |
| SNELLINGS FRANK W | PROJECTS DEV MANAGER | MEDIA GENERAL INC | 301 E GRACE ST | RICHMOND | VA | 23219 | USA |
| SNESRUD MARGARET J | PROJECT DIRECTOR | WM C BROWN PUB COMPANY | 2460 KERPER BLVD | DUBUQUE | IA | 52001 | USA |
| SNESRUD WALLY M | PROGRAMMER | WM C BROWN PUB COMPANY | 2460 KERPER BLVD | DUBUQUE | IA | 52001 | USA |
| SOCIE LARRY J | PROGRAMMER | HOOVER-NSK BEARING | 5400 S STATE RD | ANN ARBOR | MI | 48106 | USA |
| SOHMLE RONALD C | MGR COMPUTING SERVICES | REGIONAL ECONOMIC EXPANSIO | 601 SPADINA CRESCENT | SASKATOON | SA | S7K3G8 | CANADA |

| NAME | TITLE | COMPANY | ADDRESS | CITY | STATE | ZIP | COUNTRY |
|---|---|---|---|---|---|---|---|
| SOMERS PETER | | CAPE ISLAND MARINA | OCEAN DRIVE | CAPE MAY | NJ | 08204 | USA |
| SPAHN CARL P | DIRECTOR ADSS | US DEPT AGR-APHIS | 6525 BELCREST RD #853 | HYATTSVILLE | MD | 20782 | USA |
| SPALDING KENNETH G | DEVELOPMENT ENGINEER | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| SPERLE GLENN M | COMPUTER SPECIALIST | US DEPT OF AGR-APHIS | 6525 BELCREST RD #853 | HYATTSVILLE | MD | 20782 | USA |
| SPIELER CHARLES W | DIRECTOR MIS | SIMCO | 21001 CABOT BLVD | HAYWARD | CA | 94545 | USA |
| SPORKEN HEIN P | | NOVA AUTOMATION CONS | 30 NEDEREIND | NIEUWEGEIN | | | NETHER |
| SQUIRES JIM | SYSTEMS ENGR | HEWLETT-PACKARD | 1430 E ORANGETHORPE | FULLERTON | CA | 92631 | USA |
| ST PIERRE JEAN | DP MANAGER EMD | SCOTT PAPER LTD | P O BOX 760 | CRABTREE | PQ | | CANADA |
| STAMBAUGH JAN R | DIRECTOR DP | MULTNOMAH COUNTY ESD | 220 SE 102 | PORTLAND | OR | 97216 | USA |
| STARCK RICHARD E | PRESIDENT | BUSINESS COMPUTER CONCEPTS | RD #1 BOX 131-B | BURGETTSTOWN | PA | 15021 | USA |
| STARK JOHN A | SYSTEMS ENGINEER | HEWLETT-PACKARD | 1203 114TH SE | BELLEVUE | WA | 98004 | USA |
| STEIN SALLYSUE | PARTNER | T.O.A.D. | 19855 GIESENDORFER RD | COLFAX | CA | 95713 | USA |
| STOCKDALE WALLACE L | DP MANAGER | PORT OF OAKLAND | 66 JACK LONDON SQ | OAKLAND | CA | 94606 | USA |
| STOVER DAVID W | DIR OF INFO SERVICES | TELEPHONE EMPLOYEES CO | 639 S NEW HAMPSHIRE | LOS ANGELES | CA | 90005 | USA |
| STOVER RAY J | | INFORMATION RESOURCES | 4905 LIMA | DENVER | CO | | USA |
| STRANDHAGEN DEBRA A | SYSTEM CONTROL | WILLIAM M MERCER INC | 409 GRISWOLD | DETROIT | MI | 48226 | USA |
| STUMP DALE K | DATA PROCESSING MGR | STATE CONTROLLERS OFFICE | 504 E MUSSER | CARSON CITY | NV | 89701 | USA |
| SULLIVAN DENNIS J | MGR D P OPERATIONS | NATIONWIDE FINANCE | 700 OFFICE PARKWAY | CREVE COEUR | MO | 63141 | USA |
| SULLIVAN EARLENE | DATA COORDINATOR | CITY OF KENNEWICK | 210 W 6TH | KENNEWICK | WA | 99336 | USA |
| SURR T | | BOEING COMPUTER SERVICE | PO BOX 24346 | SEATTLE | WA | 98124 | USA |
| SWEARER DALE F | SYSTEMS ANALYST | ENVIRONMENTAL ELEMENTS | 3700 KOPPERS ST | BALTIMORE | MD | 21227 | USA |
| SYMONDS GORDON R | HEAD COMPUTER APPLIC | ENVIRONMENTAL HEALTH | TUNNEYS PASTURE | OTTAWA | ON | K1A0L2 | CANADA |
| SYNOLD PRISCILLA J | MANAGER TECHNICAL SPT | ISSCO | 4186 SORRENTO VALLEY BLVD | SAN DIEGO | CA | 92121 | USA |
| TANKERSLEY JAMES J | SR SYSTEMS ANALYST | PROCTER & GAMBLE | P O BOX 599 | CINCINNATI | OH | 45201 | USA |
| TANTZEN ROBERT G | CHIEF PROGRAMMING | | 658 5TH TEST GROUP/AD | HOLLOMAN AFB | NM | 88330 | USA |
| TARENS MICHAEL R | REGIONAL MKTG ENGINEER | HEWLETT-PACKARD | 19400 HOMESTEAD RD | CUPERTINO | CA | 95014 | USA |
| TATMAN PAUL E | SYSTEMS MANAGER | HMO SYSTEMS INC | 1235 RIVERSIDE | FT COLLINS | CO | 80521 | USA |
| TAYLOR DAVID W | SYSTEMS ANALYST | BOEING COMPUTER SERVICES | P O BOX 24346 | SEATTLE | WA | 98124 | USA |
| TAYLOR MIKE | PROGRAMER | A T I | 3532 S HILLCREST DR #3 | DENVER | CO | | USA |
| TAYLOR PAUL M | COMPUTER SPECIALIST | US DEPT AGRICULTURE | 6525 BELCREST RD #853 | HYATTSVILLE | MD | 20782 | USA |
| TEARE ROBERT F | ASST DIR BIBL SERV | THE CLAREMONT COLLEGES | HONNOLD LIBRARY | DARTMOUTH AT | CA | 91711 | USA |
| TEKLITS ROBERT S | DATA MANAGEMENT SUPVR | COLORADO DEPT EDUC | 201 E COLFAX | DENVER | CO | 80203 | USA |
| TENBROCT JOE R | DP MANAGER | HEWLETT-PACKARD | 5301 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| TETI FRANK A | SYSTEM ANALYST | MORGAN GUARANTY TRUST | 23 WALL STREET | NEW YORK | NY | 10015 | USA |
| THEISSEN WILHELM J | MANAGER BUSINESS SYST | HUGHES AIRCRAFT CO | P O BOX 3310 607/B318 | FULLERTON | CA | 92634 | USA |
| THOMASSON GARY | FAC INFO SYS MANAGER | HEWLETT-PACKARD | P O BOX 16 | BOISE | ID | 83707 | USA |
| THOMPSON JIM | | GOV'T OF N W TERRITORIES | | YELLOWKNIFE | NW | X0E1H0 | CANADA |
| THOMPSON JR CHARLES H | INFORMATION MGMT CHF | SPACE & MSL SYS DIV | 2805 MAPLE AVENUE | MANHATTAN BEA | CA | 90266 | USA |
| THOMPSON RANDY(CANCEL) | ANALYST | A T I | 3532 S HILLCREST DR #3 | DENVER | CO | | USA |
| THOMSON RON | DATA PROCESSING MGR | THE BOVAIRD SUPPLY CO | 823 S DETROIT | TULSA | OK | 74102 | USA |
| THORMAN BEN | ASST D P MGR | FLINT INDUSTRIES INC | P O BOX 490 | TULSA | OK | 74101 | USA |
| THORNTON WILLIAM L | PROGRAMMER/ANALYST | MASTERCRAFT IND | 4881 IRONTON ST | DENVER | CO | 80239 | USA |
| TICHAUER MARIO F W | ENGINEER | PROMON ENGENHARIA S A | NOVE DE JULHO 4939 | SAO PAULO | SP | 01407 | BRAZIL |
| TIER PAUL | MARKETING MANAGER | D C DUMMER & ASSOCIATES | 40 LK LUCERNE CL SE | CALGARY | AL | T2J3H8 | CANADA |
| TIGELMAN ROBERT J | DATA PROCESSING MGR | BEACON JOURNAL PUBL | 44 E EXCHANGE ST | AKRON | OH | 44328 | USA |
| TONNESEN LARRY D | SYSTEMS MANAGER | GUARDSMAN LIFE INS | 1025 ASHWORTH ROAD | WEST DESMOINE | IA | 50265 | USA |
| TOWNSEND RICHARD | PROGRAMMER/ANALYST | UNITED PRESIDENTIAL LIFE | 217 SOUTHWAY BLVD E | KOKOMO | IN | 46901 | USA |
| TRAPP ROBERT E | TEST ENGINEER | HUGHES AIRCRAFT CO | 1901 W MALVERN AVE | FULLERTON | CA | 92634 | USA |
| TRAPPE CLARENCE J | SYSTEMS PROGRAMMER | BECHTEL CORP | 350 MISSION STREET | SAN FRANCISCO | CA | 94105 | USA |
| TROWBRIDGE VERN H | MGR DATA PROCESSING | TIDEWATER COMM COLL | RT 135 | PORTSMOUTH | VA | 23701 | USA |
| TRUE JOHN F | DIR COMPUTING SERVICE | UNIV OF TENNESSEE | 615 MC CALLIE AVENUE | CHATTANOOGA | TN | 37402 | USA |
| TSUKISHIMA LLOYD M | PROGRAMMER/ANALYST | WEST FRASER MILLS | P O BOX 6000 | QUESNEL | BC | V2J3J5 | CANADA |

| NAME | TITLE | COMPANY | ADDRESS | CITY | STATE | ZIP | COUNTRY |
|---|---|---|---|---|---|---|---|
| TURNER WILLIAM A | PROF ASSOCIATE | WILLIA M MERCER INC | 409 GRISWOLD | DETROIT | MI | 48226 | USA |
| TWYMAN DONALD R | DIRECTOR INFO SYSTEMS | AMERICAN RED CROSS | 100 E MACK | DETROIT | MI | 48201 | USA |
| UTTER ROGER S | COMPUTER APPLICATIONS | EXXON NUCLEAR CO | 777-106 TH AVE NE | BELLEVUE | WA | 98009 | USA |
| VALERIO MARC N | GENERAL MANAGER | DIVERSIFIED COMP SYS | PO BOX 1098 | GREELEY | CO | 80631 | USA |
| VAN AUSDALL C. R. | MANAGER DATA PROCESSING | COPCO | 4905 LIMA STREET | DENVER | CO | 80239 | USA |
| VAN BUITENEN PETER | SYSTEMS DEDMAN | MAKRO INTL | CHURCHILL LAAN 11 | 352F GUUTRECH | | | NETHER |
| VAN DEN KIEBOOM AAD | | HEWLETT-PACKARD | 90- EPHARTL 121 | AMSTERDAM | | | NETHER |
| VAN KURAN PETER | PRODUCT MANAGER | HEWLETT-PACKARD | 5303 STEVENS CREEK BLVD | SANTA CLARA | CA | 95050 | USA |
| VEENEMAN WILLIAM E | SYSTEMS ANALYST | PROCTER & GAMBLE CO | 6105 CENTER HILL RD | CINCINNATI | OH | 45220 | USA |
| VELLANKI RAO C | SR PROGR/ANALYST | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| VILLA JR CHARLES J | PRESIDENT | PANTECHNIC | 5805 OCEAN VIEW DR | OAKLAND | CA | 94618 | USA |
| VILLARREAL RAMON V | | TYMWARE CORP | 44 W JEFFERSON ST | BROWNSVILLE | TX | 78520 | USA |
| VIOHL KEN | | SATELLITE COMPUTING, INC | PO BOX 2015 | NORFOLK | VA | 23501 | USA |
| VOILES DUANE A | SR SYSTEMS ANALYST | ITT FINANCIAL | P O BOX 250 | CHIPPEWA FALL | WI | 54729 | USA |
| WADE GERRY T | PRODUCT SPECIALIST | HEWLETT-PACKARD | 5600 S DTC PARKWAY | ENGLEWOOD | CO | 80110 | USA |
| WADE RON | SYSTEMS ANALYST | INFORMATION RESOURCES | 7935 E PRENTICE AVE | ENGLEWOOD | CO | 80111 | USA |
| WAGNER DAVID L | MANAGER DATA PROCESSING | MOORE & CO | 300 E SPEER BLVD | DENVER | CO | 80203 | USA |
| WALESKI JR WALTER L | DIR INFO SYSTEMS | MEDIA GENERAL INC | 301 E GRACE ST | RICHMOND | VA | 23219 | USA |
| WALLACE DARL L | DIR EDUC COMP CTR | WALLA WALLA COLLEGE | | COLLEGE PLACE | WA | 99324 | USA |
| WANDERMAN KENNETH A | MANAGER COMPUTER SYS | ITEL CORP | 1 EMBARCADERO | SAN FRANCISCO | CA | 94114 | USA |
| WARP CRAIG | | SHUGART ASS | 435 OAKMEAD PARKWAY | SUNNYVALE | CA | 94086 | USA |
| WATERS FRED | INFO SYSTEMS MANAGER | HEWLETT-PACKARD SDD | 16399 W BERNARDO DR | SAN DIEGO | CA | 92127 | USA |
| WATSON DAVID J(CANCEL) | | ARMAMENT SYSTEMS INC | 712-F N VALLEY ST | ANAHEIM | CA | 92632 | USA |
| WEBER WILLIAM | SYSTEMS ANALYST | NPD RESEARCH, INC | 15 VERBENA AVENUE | FLORAL PARK | NY | 11001 | USA |
| WEBSTER THAD N | MARKETING ENGINEER | HEWLETT-PACKARD | 2104 SUNSET AVE | BOISE | ID | 83702 | USA |
| WEISMAN JOE | MGR SYSTEM DEVELOPMENT | COMPUTERS FOR MARKETING | 215 MARKET | SAN FRANCISCO | CA | 94105 | USA |
| WEISS JAMES R | VICE PRESIDENT | E M KLEIN & ASSOC | 1000 SUPERIOR BLDG | CLEVELAND | OH | 44114 | USA |
| WELSH ROD R | SYSTEMS ANALYST | DOMINION CONSTRUCTION | 3100 3 BENTALL CENTRE | VANCOUVER | BC | V7X1B1 | CANADA |
| WETTER WAYNE | | ALUM ROCK SCHOOL DIST | 2930 GAY AVENUE | SAN JOSE | CA | 95127 | USA |
| WHEELER J LADD | DATA PROCESSING MANAGER | C M FUNK & CO INC | 22 N 2ND STREET BOX 1249 | LAFAYETTE | IN | 47902 | USA |
| WHEELER ROBIN C | EDP MANAGER | DOMTAR CONSTRUCTION MTRL | 2001 UNIVERSITY ST | MONTREAL | QU | H3A2A6 | CANADA |
| WHIDDON ROY L | SPECIALIST SOFTWARE | NORTHERN TELECOM LTD | 33 CITY CENTRE DRIVE | MISSISSAUGA | ON | L5B2N5 | CANADA |
| WHITE RUSS | PRESIDENT | R SHRIVER ASSOCIATES | 120 LITTLETON ROAD | PARSIPPANY | NJ | 07054 | USA |
| WICKHAM GAIL O | MGR MARKETING SERVICES | SYSTEMS RESEARCH INC | 2400 SCIENCE PARKWAY | OKEMOS | MI | 48864 | USA |
| WILKINSON JIM L | GENERAL PARTNER | UMPQUA DATA FACTORY | 222 E 11 | EUGENE | OR | 97401 | USA |
| WILLARD JIM | | HEWLETT-PACKARD | 815 14 SW | LOVELAND | CO | 80537 | USA |
| WILLIAMS RITA W | SUPPORT ENGINEER | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| WILLIAMSON ROBERT M | MFG DATA SYSTEMS MGR | BOEING AEROSPACE CO | PO BOX 3999 MIS 8M69 | SEATTLE | WA | 98124 | USA |
| WILOCK JAMES M | SYSTEMS ANALYST | FORD BOX 1599B | SOUTHFIELD AT ROTUNDA | DEARBORN | MI | 48121 | USA |
| WILSON ROBERT L | SALES MGR | R SHRIVER ASSOCIATES | 120 LITTLETON ROAD | PARSITPANY | NJ | 07054 | USA |
| WINTON HUGH | D P MANAGER | HOERBIGER CORP | 35 LUMBER ROAD | ROSLYN | NY | 11576 | USA |
| WOMBACHER ERNEST J | CLERK OF DISTRICT COURT | JOHNSON COUNTY IOWA | 400 S CLINTON STREET | IOWA CITY | IA | 52240 | USA |
| WONG DY-YEE | DIRECTOR PROGRAMMING | AUTOMATED ANALYSIS | 3105 DONA SOFIA DR | STUDIO CITY | CA | 91604 | USA |
| WOOD GLEN R | MANUF SYS PROJECT LDR | BOSE CORPORATION | 100 MOUNTAIN ROAD | FRAMINGHAM | MA | 01701 | USA |
| WOOD WALTER A | VICE PRESIDENT | WOOD BROWN & ASSOCIATES | 1673 CARLING SUITE 105 | OTTAWA | ON | K2A1C4 | CANADA |
| WOOLPERT BRUCE W | PRODUCT MANAGER | HEWLETT-PACKARD | 16399 W BERNARDO DR | SAN DIEGO | CA | 92127 | USA |
| WRIGHT DALE | | PANTECHNIC | 5805 OCEAN VIEW DR | OAKLAND | CA | 94618 | USA |
| WRIGHT JAMES C | ELECTRICAL ENGINEER | TEXAS MUN POWER AGENCY | 2225 E RANDOL MILL RD | ARLINGTON | TX | 76011 | USA |
| WRIGHT NORMAN B | EXAMINING SYSTEMS COORD | US CIVIL SERVICE COMM | 4685 LOG CABIN DR | MACON | GA | 31206 | USA |
| WRIGHT STEVE T | MANAGER HOME OFFICE | REPUBLIC MORTGAGE INS | P O BOX 2514 | WINSTON SALEM | NC | 27102 | USA |
| YARBROUGH CHARLES | VICE PRESIDENT | COMPUTERS FOR MARKETING | 215 MARKET | SAN FRANCISCO | CA | 94105 | USA |
| YEO MARGE | | HEWLETT-PACKARD | 5301 STEVENS CREEK BLVD | SANTA CLARA | CA | 95050 | USA |

| NAME | TITLE | COMPANY | ADDRESS | CITY | STATE | ZIP | COUNTRY |
|---|---|---|---|---|---|---|---|
| YOUNG CHUCK W | PROGRAMMER/ANALYST | ASARCO INC | P O BOX 98 | HAYDEN | AZ | 85235 | USA |
| ZABOR ELIAS | CUSTOMER RELATIONS | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |

## CONFERENCE ATTENDEE LISTING INDEXED BY ATTENDEE COMPANY

| NAME | TITLE | COMPANY | ADDRESS | CITY | STATE | ZIP | COUNTRY |
|---|---|---|---|---|---|---|---|
| KESSEL JIM (CANCELED) | INDIVIDUAL | | P O BOX 124 | CHAGRIN FALLS | OH | | USA |
| DUNCAN HANNAH F | | | 1601 LEHMBERG BLVD | COLORADO SPRI | CO | 80915 | USA |
| HANSEN WILLIAM R | SR SYSTEM CONSULTANT | | 5560 CAMERFORD DR | DAYTON | OH | 45424 | USA |
| CURRERI JOSEPH A | | | 9142 EDMONSTON CT 202 | GREENBELT | MD | 20770 | USA |
| TANTZEN ROBERT G | CHIEF PROGRAMMING | | 658 5TH TEST GROUP/AD | HOLLOMAN AFB | NM | 88330 | USA |
| PIPER MICHAEL | | | 19601 NORDHOFFF | NO RIDGE | CA | | USA |
| MONAHAN WILLIAM | DEVELOPMENT MGR | 3 M | 3M CTR 223-5N | ST PAUL | MN | 55001 | USA |
| ERICSCON GORDY O | ANALYST | 3M CO | 3M CENTER BLDG 224-4N | ST PAUL | MN | 55101 | USA |
| GREENE MARK L | SUPERVISOR | 3M CO | 3M CENTER BLDG 224-4N | ST PAUL | MN | 55101 | USA |
| KENNEDY JACK | MANAGER SYS & PROG | A G BECKER | 55 WATER STREET | NEW YORK | NY | 10041 | USA |
| SIMS RICH | EXEC V P | A T I | 3532 S HILLCREST DR #3 | DENVER | CO | | USA |
| TAYLOR MIKE | PROGRAMER | A T I | 3532 S HILLCREST DR #3 | DENVER | CO | | USA |
| THOMPSON RANDY(CANCEL) | ANALYST | A T I | 3532 S HILLCREST DR #3 | DENVER | CO | | USA |
| SEVEBORG KARL E | CIVIL ENGINEER | AB BOFORS | BOX 500 | BOFORS | | S69020 | SWEDEN |
| OSCARSSON HANS O | MANAGER | AB TRAV OCH GALOPP | BOX 1733 | STOCKHOLM | SW | 11187 | SWEDEN |
| GREEN CHARLES R | DATA PROCESSING MANAGER | ADAMS COUNTY SCHOOLS | 602 E 64TH AVE | DENVER | CO | 80229 | USA |
| RICE ROBERT J | SYSTEMS ANALYST | ADAMS COUNTY SCHOOLS | 602 E 64TH AVE | DENVER | CO | 80229 | USA |
| BEACH JOHN R | HEAD OF OPERATIONS | ADRIA LABORATORIES | P O BOX 16529 | COLUMBUS | OH | 43216 | USA |
| FERENSEN DANIEL E | SYSTEMS ANALYST | ADRIA LABORATORIES | P O BOX 16529 | COLUMBUS | OH | 43216 | USA |
| FRECH JOHN J | MANAGER OF STIC | ADRIA LABORATORIES | PO BOX 16529 | COLUMBUS | OH | 43216 | USA |
| ENGLANDER ALICE | D P CONSULTANT | ALICE ENGLANDER | 1966 TITUS STREET | SAN DIEGO | CA | 92110 | USA |
| COOK LINDA L | DATA PROC SPEC/PRGMR | ALLEGHENY INTER UNIT | SUITE 300 TWO ALLEGH CENTER | PITTSBURGH | PA | 15212 | USA |
| BAMBACH THOMAS H | PROJECT LEADER | ALLIED CHEMICAL COPR] | COLUMBIA RD | MORRISTOWN | NJ | 07960 | USA |
| BHUTA MEHENDRA | MGR SYSTEMS PLANNING | ALLIED CHEMICAL CORP | COLUMBIA RD | MORRISTOWN | NJ | 07960 | USA |
| SCROGGS ROSS E | | ALTER*ABILITY | 534 ROSAL AVENUE | OAKLAND | CA | 94610 | USA |
| WETTER WAYNE | | ALUM ROCK SCHOOL DIST | 2930 GAY AVENUE | SAN JOSE | CA | 95127 | USA |
| COOPER STEVEN M | CONSULTANT | AMERICAN MANAGEMENT SYSTEM | 561 PILGRIM DRIVE SUITE D | SAN MATEO | CA | 94404 | USA |
| SCHWARTZ RICHARD T | PRINCIPAL | AMERICAN MANAGEMENT SYSTEM | 561 PILGRIM DRIVE SUITE D | SAN MATEO | CA | 94404 | USA |
| GRILLOS JOHN M | VICE PRESIDENT | AMERICAN MGMT SYSTEMS | 561 PILGRIM DRIVE SUITE D | SAN MATEO | CA | 94404 | USA |
| TWYMAN DONALD R | DIRECTOR INFO SYSTEMS | AMERICAN RED CROSS | 100 E MACK | DETROIT | MI | 48201 | USA |
| PAPPAS STEVE | | AMERICAN SUBSCRIP TV | 8383 WILSHIRE BLVD | BEVERLY HILLS | CA | 90211 | USA |
| MAYHEW JOHN | | AMERICAN SUBSRIP TV | 8383 WILSHIRE BLVD | BEVERLY HILLS | CA | 90211 | USA |
| DAUGHERTY ROGER | DIRECTOR DATA PROCESSING | AMERICAN SUBSRIPT TV | 8383 WILSHIRE BLVD | BEVERLY HILLS | CA | 90211 | USA |
| BEASLEY DAVID R | | ANDERSON COLLEGE | | ANDERSON | IN | 46011 | USA |
| FORSEE ANN T | INSTRUCTOR | ANDERSON COLLEGE | | ANDERSON | IN | 46011 | USA |
| HARBRON THOMAS R | CHR COMPUTER SCIENCE | ANDERSON COLLEGE | | ANDERSON | IN | 46011 | USA |
| MATHIAS TIMOTHY E | | ANDERSON COLLEGE | | ANDERSON | IN | 46011 | USA |
| SHEFFIELD REBECCA L | STUDENT | ANDERSON COLLEGE | | ANDERSON | IN | 46011 | USA |
| FIELDS JAMES A | VICE PRESIDENT | APEX DATA PROCESSING | 950 SO EASTERN AVE | LOS ANGELES | CA | 90022 | USA |
| LEWIS GERALD | VICE PRESIDENT | APPLIED ANALYSIS INC | 615 SOUTH FLOWER STREET | LOS ANGELES | CA | 90017 | USA |
| OLSON BRIAN | SENIOR ANALYST | APPLIED ANALYSIS INC | 615 SOUTH FLOWER STREET | LOS ANGELES | CA | 90017 | USA |
| DAVISON GERALD W | PROJECT MANAGER | ARGONNE NATIONAL LAB | 9700 SOUTH CASS AVENUE | ARGONNE | IL | 60439 | USA |
| KOLSTO ELLIOT L | MANAGEMENT ENGINEER | ARGONNE NATIONAL LAB | 9700 SOUTH CASS AVENUE | ARGONNE | IL | 60439 | USA |
| JOERGER STEVEN G | | ARMAMENT SYSTEMS INC | 712-F N VALLEY ST | ANAHEIM | CA | 92801 | USA |
| WATSON DAVID J(CANCEL) | | ARMAMENT SYSTEMS INC | 712-F N VALLEY ST | ANAHEIM | CA | 92632 | USA |
| ENGLEBRECHT MICHAEL L | RESEARCH ENGINEER | ARMCO INC | 703 CURTIS STREET | MIDDLETOWN | OH | 45043 | USA |
| EARLS JOHN D | STAFF ENGINEER | ARTHUR A COLLINS INC | 13601 PRESTON ROAD | DALLAS | TX | 75240 | USA |
| YOUNG CHUCK W | PROGRAMMER/ANALYST | ASARCO INC | P O BOX 98 | HAYDEN | AZ | 85235 | USA |
| MC SHANE MICHAEL G | ASSOC DIRECTOR COMPUTING | ASSOC AMERICAN MED COLLEGE | ONE DUPONT CIRCLE SUITE 200 | WASHINGTON | DC | 20036 | USA |
| BUELL DIANE | ANALYST/PROGRAMMER | AURORA PUBLIC SCHOOLS | 1085 PEORIA STREET | AURORA | CO | 80011 | USA |

| | | | | | | |
|---|---|---|---|---|---|---|
| KLEVER JOHN | ANALYST/PROGRAMMER | AURORA PUBLIC SCHOOLS | 1085 PEORIA STREET | AURORA | CO 80011 | USA |
| PLEASANTS JOYCE B | DIRECTOR DATA PROCESSING | AURORA PUBLIC SCHOOLS | 1085 PEORIA STREET | AURORA | CO 80011 | USA |
| POLHEMUS JAN R | MGR BUSINESS SYSTEMS | AUSTIN INFORMATION SYS | 800 SW 16TH STREET | RENTON | WA 98055 | USA |
| CROW WILLIAM M | MGR TECHNICAL SYSTEMS | AUSTIN INFORMATION SYS | 450 WEST 1ST AVENUE | ROSELLE | NJ 07203 | USA |
| BOOTH BUD | PRESIDENT | AUTOMATED ANALYSIS | 3105 DONA SOFIA DRIVE | STUDIO CITY | CA 91604 | USA |
| WONG DY-YEE | DIRECTOR PROGRAMMING | AUTOMATED ANALYSIS | 3105 DONA SOFIA DR | STUDIO CITY | CA 91604 | USA |
| NEAL RANDOLPH H | PRESIDENT | AUTOMATED BUSINESS SV | 1649 W BROAD ST | RICHMOND | VA | USA |
| FLEET VICTOR W | CONTROLLER | B&S FINANCIAL SERVICES | N COUNTY ROAD 25A | PIQUA | OH 45356 | USA |
| ROBERTS RICHARD A | MGR SYSTEM & PROGRAMMING | B&S FINANCIAL SERVICES | N COUNTY ROAD 25A | PIQUA | OH 45356 | USA |
| BROWN ROBERT A | DATA PROCESSING SYSTEM M | B.K. SWEENEY MAN. CO. | 6300 STAPLETON S. DR. | DENVER | CO 80216 | USA |
| BORDEN JOHN (CANCELED) | SYSTEM MANAGER | BALTIMORE GAS & ELEC | BOX 1475 | BALTIMORE | MD 21203 | USA |
| BERGER ROBERT(CANCEL) | ASSISTANT VICE PRESIDENT | BANKERS TRUST CO | 1 BANKERS TRUST PLAZA | NEW YORK | NY 10006 | USA |
| BERND WALTER | ASSISTANT SYSTEM OFF | BANKERS TRUST CO | 1 BANKERS TRUST PLAZA | NYC | NY 10006 | USA |
| BLIESNER ROBERT G | SYSTEMS ANALYST | BCS | PO BOX 24346 | SEATTLE | WA 98124 | USA |
| TIGELMAN ROBERT J | DATA PROCESSING MGR | BEACON JOURNAL PUBL | 44 E EXCHANGE ST | AKRON | OH 44328 | USA |
| HISKES GEORGE J | DATA PROCESSING MANAGER | BEALL'S DEPARTMENT ST | 3923 MANATEE AVE W | BRADENTON | FL 33505 | USA |
| KOBUCHI KENT K | PROJECT SUPERVISOR | BECHTEL CORP | P O BOX 3965 | SAN FRANCISCO | CA 94119 | USA |
| TRAPPE CLARENCE J | SYSTEMS PROGRAMMER | BECHTEL CORP | 350 MISSION STREET | SAN FRANCISCO | CA 94105 | USA |
| PORTERFIELD STEPHEN T | HP3000 SYSTEM MANAGER | BECHTEL CORPORATION | P O BOX 3965 | SAN FRANCISCO | CA 94119 | USA |
| RITCHIE CLIFFORD E | D P MANAGER | BENEFIT TECHNOLOGY | 900 LAFAYETTE STREET | SANTA CLARA | CA 95050 | USA |
| MULLER MEREDITH A | PROGRAMMER/ANALYST | BOEING | P O BOX 3707 MS 3N-03 | SEATTLE | WA 98124 | USA |
| HANSON FRITZ M | LOGISTICS SYSTEM ENGINEE | BOEING AERO SPACE CO | PO BOX 3999 | SEATTLE | WA 98124 | USA |
| PIERCE ANTHONY | SR SYSTEMS ANALYST | BOEING AEROSPACE CO | PO BOX 3999 | SEATTLE | WA 98124 | USA |
| DRYNAN GILBERT W | SYSTEMS DEV MANAGER | BOEING AEROSPACE CO | P O BOX 313 | WOODINVILLE | WA 98072 | USA |
| WILLIAMSON ROBERT M | MFG DATA SYSTEMS MGR | BOEING AEROSPACE CO | PO BOX 3999 M1S 8M69 | SEATTLE | WA 98124 | USA |
| CHIU WUYAN | PROJECT ENGINEER | BOEING COMM AIRPLANE CO | P O BOX 3707 36-02 | SEATTLE | WA 98124 | USA |
| JELLIFFE ARLENE M | LEAD ENGINEER | BOEING COMM AIRPLANE CO | BOX 3707 | SEATTLE | WA 98124 | USA |
| RUTHERFORD CLYDE R | SUPV MSS COMPUTING | BOEING COMMERCIAL AIR | P O BOX 3707 M/S 5F-09 | SEATTLE | WA 98124 | USA |
| HALSEY GREGG | SYSTEM ANALYST | BOEING COMPUTER SERVICE | P O BOX 24346 | SEATTLE | WA 98124 | USA |
| JOHNSON PAMELA | SYSTEM ANALYST | BOEING COMPUTER SERVICE | P O BOX 24346 | SEATTLE | WA 98124 | USA |
| KANAGA MIKE | SYSTEM ANALYST | BOEING COMPUTER SERVICE | P O BOX 24346 | SEATTLE | WA 98124 | USA |
| KING GAIL | PROJECT MANAGER | BOEING COMPUTER SERVICE | P O BOX 24346 | SEATTLE | WA 98124 | USA |
| MARSICEK R G | | BOEING COMPUTER SERVICE | PO BOX 24346 | SEATTLE | WA 98124 | USA |
| MASSLINGER BOB | SYSTEM ANALYST | BOEING COMPUTER SERVICE | P O BOX 24346 | SEATTLE | WA 98124 | USA |
| SURR T | | BOEING COMPUTER SERVICE | PO BOX 24346 | SEATTLE | WA 98124 | USA |
| MURPHY DONALD C | SYSTEM MANAGER | BOEING COMPUTER SERVICES | P O BOX 3707 M/S36-01 | SEATTLE | WA 98124 | USA |
| TAYLOR DAVID W | SYSTEMS ANALYST | BOEING COMPUTER SERVICES | P O BOX 24346 | SEATTLE | WA 98124 | USA |
| IRIYE DICK | SYSTEMS ANALYST | BOETTCHER & CO | 828 17TH STREET | DENVER | CO 80202 | USA |
| MEDD RANDY | HP3000 SYSTEMS MGR | BOETTCHER & CO | 828 17TH STREET | DENVER | CO 80202 | USA |
| GUNBY D L (CANCELED) | SUPV SYSTEM DESIGN & PRO | BONAR & BEMIS LTD | 2380 MC DOWELL ROAD | BURLINGTON | ON L7R4A1 | CANADA |
| SEIFRIED W BRIAN | SUPERVISOR SYSTEMS APPL | BONAR & BEMIS LTD | 2380 MC DOWELL ROAD | BURLINGTON | ON L7R4A1 | CANADA |
| DOWLING JAMES F | DP OPERATIONS MANAGER | BOSE CORPORATION | 100 MOUNTAIN ROAD | FRAMINGHAM | MA 01701 | USA |
| WOOD GLEN R | MANUF SYS PROJECT LDR | BOSE CORPORATION | 100 MOUNTAIN ROAD | FRAMINGHAM | MA 01701 | USA |
| JARAMILLO JR NARCISO | DB ADMIN | BOURNS | 1200 COLUMBIA AVE | RIVERSIDE | CA 92507 | USA |
| BRINDLE JIM | | BRANT COMPUTER SERVICES | 615 BRANT STREET | BURLINGTON | ON L7R2G6 | CANADA |
| BLAND JOHN J | SYSTEM DP ANALYST | BROOKEHAVEN NATL LABOR | 32 BROOKEHAVEN AVE | UPTON | NY 11973 | USA |
| HAUDERT GERARD A | OPERATIONS SUPERVISOR | BROOKEHAVEN NATL LABOR | 32 BROOKEHAVEN AVE | UPTON | NY 11973 | USA |
| ROSS PATRICK D | SR MEMBER TECHNICAL STAF | BSL INC | 495 JAVA | SUNNYVALE | CA 94086 | USA |
| KUKUDA, JR. JOHN | VICE PRESIDENT | BUSINESS COMPUTER CONCEPTS | RD #1 BOX 131-B | BURGETTSTOWN | PA 15021 | USA |
| STARCK RICHARD E | PRESIDENT | BUSINESS COMPUTER CONCEPTS | RD #1 BOX 131-B | BURGETTSTOWN | PA 15021 | USA |
| FUNK CHRISTOPHER M | PRESIDENT | C M FUNK & CO INC | 22 N 2ND STREET BOX 1249 | LAFAYETTE | IN 47902 | USA |

| NAME | TITLE | COMPANY | ADDRESS | CITY | STATE | ZIP | COUNTRY |
|---|---|---|---|---|---|---|---|
| WHEELER J LADD | DATA PROCESSING MANAGER | C M FUNK & CO INC | 22 N 2ND STREET BOX 1249 | LAFAYETTE | IN | 47902 | USA |
| PRITCHARD G BRIAN | SR PROGRAMMER | CANADA BLDG MATERIALS | 55 INDUSTRIAL ST | TORONTO | ON | M4G3W9 | CANADA |
| ONIZUKA SANDRA M | SYSTEM MANAGER | CANCER CENTER OF HI | 1150 SOUTH KING ST | HONOLULU | HA | 96814 | USA |
| SOMERS PETER | | CAPE ISLAND MARINA | OCEAN DRIVE | CAPE MAY | NJ | 08204 | USA |
| ALBARRAN RAUL F | GTE DE PROCESAMIENTO | CARDANES, SA | APARTADO POSTAL 591 | QUERETARO | QR | 591 | MEXICO |
| HAISCH KEN R | PRESIDENT | CASCADE COMPUTER SYSTEMS I | P O BOX 1666 | LONGVIEW | WA | 98632 | USA |
| LUNDBERG ERIK | SCIENTIFIC PROGRAMMER | CASS UNIV OF WASH | 1107 NE 45TH ROOM 530 | SEATTLE | WA | 98105 | USA |
| RAWSON TOM | SYSTEMS PROGRAMMER | CASS UNIV OF WASH | 1107 NE 45TH ROOM 530 | SEATTLE | WA | 98105 | USA |
| BALANDER MATTHEW J | SYSTEM PROGRAMMER | CCSC | 2640 PINE STREET | ST LOUIS | MO | 63103 | USA |
| FARKAS GEORGE J | SYSTEMS ANALYST | CHRYSLER CORPORATION | PO BOX 1919 CIMS:4162725 | DETROIT | MI | 48288 | USA |
| PIEKARSKI DAVID M | SUPVR COMPUTER UTILIZ | CHRYSLER CORPORATION | PO BOX 1919 CIMS:4162725 | DETROIT | MI | 48288 | USA |
| COPLIN ROBERT G | D P DIRECTOR | CITY OF KENNEWICK | 210 W 6TH | KENNEWICK | WA | 99336 | USA |
| SULLIVAN EARLENE | DATA COORDINATOR | CITY OF KENNEWICK | 210 W 6TH | KENNEWICK | WA | 99336 | USA |
| FRAGH EDWARD A | SYSTEMS & PROG SUPVSR | CITY OF SANTA MONICA | 1685 MAIN STREET | SANTA MONICA | CA | 90401 | USA |
| PREDMORE LIONEL A | INFORMATION SYSTEMS SUPV | CITY OF TEMPE | 31 E FIFTH BOX 5002 | TEMPE | AZ | 85281 | USA |
| CHARD MILES M | DATA PROCESSING MANAGER | CLA-VAL CO | P O BOX 1325 | NEWPORT BEACH | CA | 92663 | USA |
| BARKER RONALD E | MANAGING DIRECTOR | CLAYBROOK COMPUTING | 70 BROOK ST | LONDON | | W1Y2HN | ENGLAN |
| MELVIN PETER S | DIRECTOR OF MARKETING | CMC ASSOCIATES | 3 MARGARET ST | BOSTON | MA | 02021 | USA |
| CORDELLE YANN | ENGINEER | COGELOG | 1 QUINCONCES | GIF | | 91190 | FRANCE |
| FEHR GENE I | D P MANAGER | COHEN FURNITURE CO | 336 SW ADAMS STREET | PEORIA | IL | 61602 | USA |
| ALEXANDER BYRON L | CONSULTANT | COLLIER-JACKSON ASSOC | 1805 N WESTSHORE BLVD | TAMPA | FL | 33607 | USA |
| JACKSON CHARLES W | PRESIDENT | COLLIER-JACKSON ASSOC | 1805 N WESTSHORE BLVD | TAMPA | FL | 33607 | USA |
| SLOAN DAVID L | CONSULTANT | COLLIER-JACKSON ASSOC | 1805 N WESTSHORE BLVD | TAMPA | FL | 33607 | USA |
| HUDSON MEL | SYSTEMS ANALYST | COLO DEPT OF EDUC | 201 E COLFAX | DENVER | CO | 80203 | USA |
| ICKLER AL | SENIOR PROGRAMMER | COLO DEPT OF EDUC | 201 E COLFAX | DENVER | CO | 80203 | USA |
| PETERSON DONALD C | MGR SYSTEMS ANALYST | COLO DEPT OF HIGHWAYS | 4201 E ARKANSAS | DENVER | CO | 80222 | USA |
| TEKLITS ROBERT S | DATA MANAGEMENT SUPVR | COLORADO DEPT EDUC | 201 E COLFAX | DENVER | CO | 80203 | USA |
| FRAME DARYL A | SYSTEM MANAGER | COMARCO INC | 227 W HUENEME ROAD | OXNARD | CA | 93030 | USA |
| GOLDMANN ROGER M | SYSTEMS ANALYST | COMARCO INC | 227 W HUENEME ROAD | OXNARD | CA | 93030 | USA |
| HWA CHIH S | PRESIDENT | COMPUSYS INC | 789 SHERMAN 560 | DENVER | CO | 80203 | USA |
| MARCHESE BUZZ | SYSTEM MANAGER | COMPUTER COMPOSITION SALES | 2640 PINE ST | ST LOUIS | MO | 63103 | USA |
| KORNEK KEN | | COMPUTER SERVICE DIVISION | 19310 PRUNERIDGE AVE | CUPERTINO | CA | 95014 | USA |
| RUSSELL JIM | PRESIDENT | COMPUTER SERVICES CORP | 75 MANHATTAN DR #107 | BOULDER | CO | 80303 | USA |
| WEISMAN JOE | MGR SYSTEM DEVELOPMENT | COMPUTERS FOR MARKETING | 215 MARKET | SAN FRANCISCO | CA | 94105 | USA |
| YARBROUGH CHARLES | VICE PRESIDENT | COMPUTERS FOR MARKETING | 215 MARKET | SAN FRANCISCO | CA | 94105 | USA |
| PUTEGNAT MICHAEL(CANC) | PRESIDENT | CONTROL SYSTEMS | 44 W JEFFERSON ST | BROWNSVILLE | TX | 78520 | USA |
| RYAN GEOFFREY | | CONTROL SYSTEMS | 44 W JEFFERSON ST | BROWNSVILLE | TX | 78520 | USA |
| COOKSEY WILLIAM P | PROGRAMMER/ANALYST | COPCO | 4905 LIMA STREET | DENVER | CO | 80239 | USA |
| MEYER JAMES J | COMPUTER OPERATIONS | COPCO | 4905 LIMA STREET | DENVER | CO | 80239 | USA |
| VAN AUSDALL C. R. | MANAGER DATA PROCESSING | COPCO | 4905 LIMA STREET | DENVER | CO | 80239 | USA |
| JAMISON CARL L | DATA PROCESSING MANAGER | CRAIG HOSP RESEARCH OFFC | 3460 SO CLARKSON | ENGLEWOOD | CO | 80110 | USA |
| DUMMER DAVID C | PRESIDENT | D C DUMMER & ASSOCIATES | 40 LK LUCERNE CL SE | CALGARY | AL | T2J3HS | CANADA |
| TIER PAUL | MARKETING MANAGER | D C DUMMER & ASSOCIATES | 40 LK LUCERNE CL SE | CALGARY | AL | T2J3H8 | CANADA |
| SALINS GARY E | DATA PROCESSING MANAGER | DARE PAFCO, INC | 11353 REED HARTMAN HY | CINCINNATI | OH | 45040 | USA |
| LESSEY KEN W | ASSOCIATE | DATACOM | 50 WEST STREET | ST HELENS | OR | 97051 | USA |
| CHADWICK GRAHAM D | COMPUTER MANAGER | DE ZOETE D BEVAN | THROGMORTON STREETS | LONDON | | | ENGLAN |
| PARELLA MIKE | PRESIDENT | DECISION STRATEGY | 708 THIRD AVENUE | NEW YORK | NY | 10017 | USA |
| AUSTIN DAVID J | ACTING DATA CENT COORD | DENVER EMPLOY & TRAIN | 1421 ELATI STREET | DENVER | CO | 80204 | USA |
| MOODY JERRY R | SYSTEMS SUP | DEPT OF ARMY | BLDG 12500 ATTN DXRMC-C-C | FORT LEE | VA | 23801 | USA |
| MC LEMORE JIM | SYSTEMS MANAGER | DEPT OF ENERGY | 1330 BROADWAY | OAKLAND | CA | | USA |
| PASHAK JAMES W | COMPUTER SPECIALIST | DEPT OF ENERGY | 1333 BROADWAY | OAKLAND | CA | 94161 | USA |
| BELLIS STEPHEN | PROGRAMMER | DEPT OF FISHERIES & OCEANS | BRANDY COVE | ST ANDREWS | NB | E0G2X0 | CANADA |

| NAME | TITLE | COMPANY | ADDRESS | CITY | STATE | ZIP | COUNTRY |
|---|---|---|---|---|---|---|---|
| FAWKES GERALD | PROGRAMMER | DEPT OF FISHERIES & OCEANS | BRANDY COVE | ST ANDREWS | NB | E0G2X0 | CANADA |
| GURUPRASAD CHOULGERE | DIRECTOR CORP SYSTEMS | DEPT OF SUPPLY CANADA | 11 LAURIER | HULL | QU | | CANADA |
| CAMERON LOUISE | PROGRAMMER | DEPT REG ECO EXPANSION | 200 RUE PRINCIPAL | HULL | QU | K1A0M4 | CANADA |
| IMBEAU ANDRE | ACTING CHIEF | DEPT REG ECO EXPANSION | 200 RUE PRINCIPAL | HULL | QU | K1A0M4 | CANADA |
| CLEEVER KAREN J | SYSTEMS PROGRAMMER | DEPT REGIONAL ECONOMIC EXP | 601 SPADINA CRES E | SASKATOON | SA | S7K3G8 | CANADA |
| VALERIO MARC N | GENERAL MANAGER | DIVERSIFIED COMP SYS | PO BOX 1098 | GREELEY | CO | 80631 | USA |
| JACKSON JOHN A | SYSTEMS ANALYST | DOMINION CONSTRUCTION | 3100 3 BENTALL CENTRE | VANCOUVER | BC | V7X1B1 | CANADA |
| MC MURRAY JACK M | DP MANAGER | DOMINION CONSTRUCTION | 3100 3 BENTALL CENTRE | VANCOUVER | BC | V7X1B1 | CANADA |
| WELSH ROD R | SYSTEMS ANALYST | DOMINION CONSTRUCTION | 3100 3 BENTALL CENTRE | VANCOUVER | BC | V7X1B1 | CANADA |
| WHEELER ROBIN C | EDP MANAGER | DOMTAR CONSTRUCTION MTRL | 2001 UNIVERSITY ST | MONTREAL | QU | H3A2A6 | CANADA |
| LUFT MARKUS F | SYSTEMS & PROG SUPRV | DOMTAR CONSTRUCTION MTRLS | 2001 UNIVERSITY ST | MONTREAL | QU | H5A2A6 | CANADA |
| BANDI DENNIS | TECHNICAL ANALYST | DOMTAR INC | 395 DE MAISONNEUVE BLVD | MONTREAL | QU | J6K2E6 | CANADA |
| BILMER,EARL | | DOMTAR INC | 395 DE MAISONNEUVE BLVD | MONTREAL | QU | J6K2E6 | CANADA |
| FAIRCHILD JAMES B | SUPERVISOR OPERATIONS | DOMTAR INC CHEM GROUP | BP 7212 | MONTREAL | QU | H3C3M3 | CANADA |
| GOSSELIN JEANNINE G | EDP MANAGER | DOMTAR INC CHEM GROUP | BP 7212 | MONTREAL | QU | H3C3M3 | CANADA |
| GUPTA RAMESH | SYSTEMS & OR CONSULTANT | DOMTAR INC CHEM GROUP | BP 7212 | MONTREAL | QU | H3C3M3 | CANADA |
| MINOR TERRY | SYSTEMS MGR | DONNELLY MIRRORS INC | 49 W 3RD ST | HOLLAND | MI | 49423 | USA |
| SMITH ROGER | DP MANAGER | DONNELLY MIRRORS INC | 49 W 3RD ST | HOLLAND | MI | 49423 | USA |
| WEISS JAMES R | VICE PRESIDENT | E M KLEIN & ASSOC | 1000 SUPERIOR BLDG | CLEVELAND | OH | 44114 | USA |
| LONG LYNDA J | D P MANAGER | E S T | 765 CALIFORNIA STREET | SAN FRANCISCO | CA | 94108 | USA |
| ANDERSEN SVEND | | EBI COMPANIES | 1290 NORTH FIRST ST | SAN JOSE | CA | 95112 | USA |
| BROWN DANA | ADMINISTRATOR INFO | EBI COMPANIES | 1290 NORTH FIRST ST | SAN JOSE | CA | 95112 | USA |
| ECCLES SHARON L | DATA PROCESSING MANAGER | ECKANKAR | 120 SCOTT DRIVE | MENLO PARK | CA | | USA |
| PRELLE JEAN | PROFESSOR | ECOLE SUPERIEURE DE COMMER | 23 ROUTE DE DARDILLY | ECULLY | | | FRANCE |
| MOORE ROBERT E | PRESIDENT | EDUCATIONAL COMPUTER SYS | 1313 KEMPER RD | CINCINNATI | OH | 45246 | USA |
| PETERS HAROLD J | PRESIDENT | EDUCATIONAL SOFTWARE PRODU | 9 GEORGETOWN CIRCLE | IOWA CITY | IA | 52240 | USA |
| FIGUEROA LUIS V | GTE DE PROCESAMIENTO | EJES TRACTIVOS SA | APARTADO POSTAL 14820 | MEXICO | DF | 14820 | MEXICO |
| GOMEZ VICTOR | SYSTEM ENGINEER | ELECTRONIC DATA SYS | ONE WAYNE MALL | WAYNE | NJ | 07470 | USA |
| HARBAUGH JERRY E | MGR DATA PROCESSING | ELFAB | 4200 WYLEY POST | ADDISSON | TX | 75001 | USA |
| CLABORN GEORGE H | PROGRAMMER ANALYST | ENVIRONMENTAL ELEMENTS | 3700 KOPPERS ST | BALTIMORE | MD | 21227 | USA |
| FISHER M ROGER | MANAGER COMPUTER SERV | ENVIRONMENTAL ELEMENTS | 3700 KOPPERS ST | BALTIMORE | MD | 21227 | USA |
| LANCASTER HENRY C | V P ENG DEV & COMP SCI | ENVIRONMENTAL ELEMENTS | 3700 KOPPERS ST | BALTIMORE | MD | 21227 | USA |
| MILLER DANIEL J | SYSTEMS ANALYST | ENVIRONMENTAL ELEMENTS | 3700 KOPPERS ST | BALTIMORE | MD | 21227 | USA |
| SWEARER DALE F | SYSTEMS ANALYST | ENVIRONMENTAL ELEMENTS | 3700 KOPPERS ST | BALTIMORE | MD | 21227 | USA |
| SYMONDS GORDON R | HEAD COMPUTER APPLIC | ENVIRONMENTAL HEALTH | TUNNEYS PASTURE | OTTAWA | ON | K1A0L2 | CANADA |
| BENOIT WAYNE F | MANAGER PRODUCT DEV | EPSILON DATA MGMT | 24 NE EXECUTIVE PK | BURLINGTON | MA | 01803 | USA |
| CANTWELL FRANK E | DIRECTOR PRODUCT DEV | EPSILON DATA MGMT | 24 NE EXECUTIVE PK | BURLINGTON | MA | 01803 | USA |
| JONES THOMAS O | EXEC V P | EPSILON DATA MGMT | 24 NE EXECTIVE PK | BURLINGTON | MA | 01803 | USA |
| NEIBERGS GEORGE J | IS MANAGER | ESB-EXIDE | 101 GIBRALTOR RD | HORSHAM | PA | 19044 | USA |
| FINE BRIAN T | TECH STAFF | ESL INC | 495 JAVA DR | SUNNYVALE | CA | 94086 | USA |
| LOCKHEED ALLAN H | | EXXON MINERAL CO USA | 601 JEFFERSON | HOUSTON | TX | 77601 | USA |
| UTTER ROGER S | COMPUTER APPLICATIONS | EXXON NUCLEAR CO | 777-106 TH AVE NE | BELLEVUE | WA | 98009 | USA |
| LEE HAROLD | | FAIRFAX CNTY PUBLIC SCHOOL | DATA SERVICES DIVISION | SPRINGFIELD | VA | 22151 | USA |
| SHAULIS MICHAEL | | FAIRFAX CNTY PUBLIC SCHOOL | DATA SERVICES DIVISION | SPRINGFIELD | VA | 22151 | USA |
| DELONG DAN G | PROGRAMMER | FEDERAL HOME LOAN BANK | 600 STEWART ST | SEATTLE | WA | 98101 | USA |
| JOHNSTON CHARLES F | DATA PROCESSING MANAGER | FERRIS BSSCHER LOHMA | 339 E 16TH STREET | HOLLAND | MI | 49423 | USA |
| BERAM ALFRED J | EXECUTIVE VICE PRESIDENT | FINANCIAL DATA PLANNING | 2670 TIGERTAIL AVENUE | MIAMI | FL | 33133 | USA |
| GOLDBERG MICHAEL C | PRESIDENT | FINANCIAL DATA PLANNING | 2670 TIGERTAIL AVENUE | MIAMI | FL | 33133 | USA |
| KENNEDY DOUGLAS | VICE PRESIDENT | FINANCIAL DATA PLANNING | 2670 TIGERTAIL AVENUE | MIAMI | FL | 33133] | USA |
| THORMAN BEN | ASST D P MGR | FLINT INDUSTRIES INC | P O BOX 490 | TULSA | OK | 74101 | USA |
| LOCHNER CHRIS | SYSTEM MANAGER | FORD BOX 1599B | SOUTHFIELD AT ROTUNDA | DEARBORN | MI | 48121 | USA |
| WILOCK JAMES M | SYSTEMS ANALYST | FORD BOX 1599B | SOUTHFIELD AT ROTUNDA | DEARBORN | MI | 48121 | USA |

| NAME | TITLE | COMPANY | ADDRESS | CITY | STATE | ZIP | COUNTRY |
|---|---|---|---|---|---|---|---|
| ARTHOFER ROBERT J | DATA PROCESSING MANAGER | FOSECO INC | 20200 SHELDON ROAD | BROOKPARK | OH | 44142 | USA |
| ONYX ROBERT P | D P OPER MANAGER | FOSECO INC | 20200 SHELDON ROAD | BROOKPARK | OH | 44142 | USA |
| FRATUS WILLIAM P | | FUTURA INC | 1714 S CONGRESS | AUSTIN | TX | 78704 | USA |
| KELLY KENT | | FUTURA INC | 1714 S CONGRESS | AUSTIN | TX | 78704 | USA |
| MC AFEE WILLIAM K | PRESIDENT | FUTURA INC | 1714 S CONGRESS | AUSTIN | TX | 78704 | USA |
| DROBNY RONALD G | SYSTEM MANAGER | GATES & SONS | 90 SOUTH FOX | DENVER | CO | 80226 | USA |
| BERGOLD THEODORE A | MANAGER INFO-SYSTEMS | GATX LEASING | ONE EMBARCADERO CNTR | SAN FRANCISCO | CA | 94111 | USA |
| SJOGREN DAVID R | SYSTEM SPECIALIST | GENERAL MILLS | PO BOX 1113 | MINNEAPOLIS | MN | 55440 | USA |
| MATT RICHARD G | HARDWARE PLANNER | GENERAL MILLS INC | 9200 WAYZATA BLVD | MINNEAPOLIS | MN | 55426 | USA |
| MEINZ MICHAEL J | SUPERVISOR,TECH SERVS | GENERAL MILLS, INC | PO BOX 1113 | MINNEAPOLIS | MN | 55440 | USA |
| FAIRFIELD STEVE | INFO SYSTEMS ANALYST | GENERAL TELEPHONE | 455 E ELLIS ROAD | MUSKEGON | MI | 49443 | USA |
| EDWARDS BETH | INFO SYSTEMS ADM | GENERAL TELEPHONE-MI | 455 E ELLIS ROAD | MUSKEGON | MI | 49443 | USA |
| BOTTEGAL THOMAS M | | GEORGE WASHINGTON UNIVERSI | 725 23 STREET NW | WASHINGTON | DC | 20052 | USA |
| LEE LANCE | SENIOR PROGRAMMER | GEORGE WIMPEY CANADA | 80 NORTH QUEEN ST | TORONTO | ON | M8Z2C9 | CANADA |
| ROBINSON JOEL H | MGR INFORMATION SYS | GEORGE WIMPEY CANADA | 80 NORTH QUEEN ST | TORONTO | ON | M8Z2C9 | CANADA |
| HATCH JAMES L | DIRECTOR | GM SAGINAW DATA CENTER | 3900 HOLLAND ROAD | SAGINAW | MI | 48605 | USA |
| JACOB HARRY O | MGR DISTRIBUTED SYSTEMS | GM SAGINAW DATA CENTER | 3900 HOLLAND ROAD | SAGINAW | MI | 48605 | USA |
| BEDET ROB | | GOV'T OF N W TERRITORIES | | YELLOWKNIFE | NW | X0E1H0 | CANADA |
| DELMAGE A R(CANCELED) | | GOV'T OF N W TERRITORIES | | YELLOWKNIFE | NW | X0E1H0 | CANADA |
| JOHNSON BOB | | GOV'T OF N W TERRITORIES | | YELLOWKNIFE | NW | X0E1H0 | CANADA |
| THOMPSON JIM | | GOV'T OF N W TERRITORIES | | YELLOWKNIFE | NW | X0E1H0 | CANADA |
| ADAMS RAY | CONTROLLER | GRACELAND COLLEGE | | LAMONI | IA | 50140 | USA |
| BILLS DANIEL G | PRESIDENT | GRANVILLE-PHILLIPS CO | 5675 ARAPAHOE | BOULDER | CO | 80303 | USA |
| MALACHOWSKI ERNEST S | CONTROLLER | GRANVILLE-PHILLIPS CO | 5675 ARAPAHOE | BOULDER | CO | 80303 | USA |
| PICK V MICHAEL | RESEARCH COMPUTER MGR | GRUMMAN AERO SPACE CORP | A 08-35 | BETHPAGE | NY | 11714 | USA |
| BEADLE, JR RAY | SR SYSTEMS ANALYST | GRUMMAN AEROSPACE | SOUTH OYSTER BAY ROAD | BETHPAGE | NY | | USA |
| BINDEWALD THOMAS L | SR TECHNICAL ANALYST | GTE DATA SERVICES | FIRST FINANCIAL TOWER | TAMPA | FL | 33611 | USA |
| TONNESEN LARRY D | SYSTEMS MANAGER | GUARDSMAN LIFE INS | 1025 ASHWORTH ROAD | WEST DESMOINE | IA | 50265 | USA |
| BYFORD WENDY K | SYSTEMS ENGINEER | H P | 275 HYMUS BLVD | PTE CLAIRE | QU | | CANADA |
| KENFIELD JOHN E | INFO SYS MANAGER | H-P SANTA ROSA DIV | 1400 FOUNTAIN GROVE PK | SANTA ROSA | CA | 95404 | USA |
| JACKSON ELAINE T | MINICOMPUTER ANALYST | HARTFORD INSURANCE | HARTFORD PLAZA | HARTFORD | CT | 06033 | USA |
| BENNETT WALTER W | DEV INFO COORDINATOR | HARVARD | HOLYOKE | CAMBRIDGE | MA | 02138 | USA |
| HUTTUNEN HEIKKI J | ADP MANAGER | HELSINKI SCHOOL OF ECON | RUNEBERGINK 14-16 | HELSINKI | | 00100S | FINLAN |
| VAN DEN KIEBOOM AAD | | HEWLETT-PACKARD | 90- EPHARTL 121 | AMSTERDAM | | | NETHER |
| BUTLER KEITH C | ENGINEER | HEWLETT-PACKARD | RT 41 & STAR ROAD | AVONDALE | PA | 19311 | USA |
| CORRELL STEVEN | ENGINEER | HEWLETT-PACKARD | RT 41 & STAR ROAD | AVONDALE | PA | 19311 | USA |
| FREED JAMES F | INFO SYSTEMS MANAGER | HEWLETT-PACKARD | ROUTE 41 | AVONDALE | PA | 19311 | USA |
| STARK JOHN A | SYSTEMS ENGINEER | HEWLETT-PACKARD | 1203 114TH SE | BELLEVUE | WA | 98004 | USA |
| HOKE ROBERT D | MARKETING MANAGER | HEWLETT-PACKARD | DISC MEMORY DIVISION | BOISE | ID | 83707 | USA |
| LOWRY GLEN H | INFO SYS MANAGER | HEWLETT-PACKARD | P O BOX 15 | BOISE | ID | 83707 | USA |
| THOMASSON GARY | FAC INFO SYS MANAGER | HEWLETT-PACKARD | P O BOX 16 | BOISE | ID | 83707 | USA |
| WEBSTER THAD N | MARKETING ENGINEER | HEWLETT-PACKARD | 2104 SUNSET AVE | BOISE | ID | 83702 | USA |
| CARRUTHERS ALEX R | SYSTEMS ENGINEER | HEWLETT-PACKARD | 210 7220 FISHER ST | CALGARY | AL | T2H2H8 | CANADA |
| ENGBERG TONY I | SYSTEMS ENGINEER | HEWLETT-PACKARD | 19855 GIESENDORFER RD | COLFAX | CA | 95713 | USA |
| GRUBER DIANNE | | HEWLETT-PACKARD | 1900 GARDEN GODS ROAD | COLORADO SPRI | CO | 80907 | USA |
| DOUGLAS GORDON R | FACILITY INFO SYS MGR | HEWLETT-PACKARD | 11000 WOLFE ROAD | CUPERTINO | CA | 95014 | USA |
| JOHNSTONE SHIRLEY J | SOFTWARE RELIABILITY ENG | HEWLETT-PACKARD | 19480A HOMESTEAD RD | CUPERTINO | CA | 95014 | USA |
| LAIR STEVE | S E DSITRICT MGR | HEWLETT-PACKARD | 19310 PRUNERIDGE RD | CUPERTINO | CA | 95014 | USA |
| MORAIN OLEN | COMPUTER SERVICES DIV | HEWLETT-PACKARD | 19310 PRUNERIDGE AVE | CUPERTINO | CA | 95014 | USA |
| TARENS MICHAEL R | REGIONAL MKTG ENGINEER | HEWLETT-PACKARD | 19400 HOMESTEAD RD | CUPERTINO | CA | 95014 | USA |
| DONHAM DAN | SYSTEMS ENGINEER | HEWLETT-PACKARD | 5600 DTC PKWY | ENGLEWOOD | CO | 80110 | USA |
| JOHNSON RON H | DISTRICT SALES MGR | HEWLETT-PACKARD | 5600 DTC PKWY | ENGLEWOOD | CO | 80110 | USA |

| NAME | TITLE | COMPANY | ADDRESS | CITY | STATE | ZIP | COUNTRY |
|---|---|---|---|---|---|---|---|
| KUEHNER WARREN | S E DISTRICT MGR | HEWLETT-PACKARD | 5600 DTC PKWY | ENGLEWOOD | CO | 80111 | USA |
| WADE GERRY T | PRODUCT SPECIALIST | HEWLETT-PACKARD | 5600 S DTC PARKWAY | ENGLEWOOD | CO | 80110 | USA |
| FRENCH DEBORAH J | PROGRAMMER/ANALYST | HEWLETT-PACKARD | 3400 E HARMONY ROAD | FORT COLLINS | CO | 80525 | USA |
| JOHNSON GARY L | INFO SYSTEMS MGR | HEWLETT-PACKARD | 3800 HARMONY ROAD | FORT COLLINS | CO | 80524 | USA |
| SQUIRES JIM | SYSTEMS ENGR | HEWLETT-PACKARD | 1430 E ORANGETHORPE | FULLERTON | CA | 92631 | USA |
| AMBLER BURT | | HEWLETT-PACKARD | 815 14 SW | LOVELAND | CO | 80537 | USA |
| HALLOCK JIM | | HEWLETT-PACKARD | 815 14 SW | LOVELAND | CO | 80537 | USA |
| KASUN ELLEN | | HEWLETT-PACKARD | 815 14 SW | LOVELAND | CO | 80537 | USA |
| LEMLEY JOHN | | HEWLETT-PACKARD | 815 14 SW | LOVELAND | CO | 80537 | USA |
| WILLARD JIM | | HEWLETT-PACKARD | 815 14 SW | LOVELAND | CO | 80537 | USA |
| SCHWARTZ RICK A | DIST CE MANAGER | HEWLETT-PACKARD | 6877 GOREWAY DRIVE | MISSISSAUGA | ON | L4V1M8 | CANADA |
| CASEY CHRIS J | SYSTEMS SUPPORT MGR | HEWLETT-PACKARD | 1501 PAGE MILL RD | PALO ALTO | CA | 94304 | USA |
| SCHWARZ RAYMOND T | MINI-OPERATIONS MGR | HEWLETT-PACKARD | 1501 PAGE MILL RD | PALO ALTO | CA | 94304 | USA |
| SLATER TED | SALES REP | HEWLETT-PACKARD | 10691 SHELLBRIDGE WAY | RICHMOND | BC | V6X2W7 | CANADA |
| GUERRERO JORGE | S E INSTRUCTOR | HEWLETT-PACKARD | 4 CHOKE CHERRY RD | ROCKVILLE | MD | 20850 | USA |
| WOOLPERT BRUCE W | PRODUCT MANAGER | HEWLETT-PACKARD | 16399 W BERNARDO DR | SAN DIEGO | CA | 92127 | USA |
| GROFF JAMES R | PRODUCT MANAGER | HEWLETT-PACKARD | 1342 STAYNER RD | SAN JOSE | CA | 95121 | USA |
| OSBORNE LEE K | DEVELOPMENT ENGINEER | HEWLETT-PACKARD | 1046 S WINCHESTER #9 | SAN JOSE | CA | 95128 | USA |
| JORGENSON DANIEL M | PRODUCT SUPPORT MGR | HEWLETT-PACKARD | 5303 STEVENS CREEK BLVD | SANATA CLARA | CA | 95050 | USA |
| ALDERETE JOHN R | PROJECT MANAGER | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| BARMAN MARC | DEVELOPMENT ENGINEER | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| BIRKWOOD ILENE M | S E MANAGER | HEWLETT-PACKARD | 5303 STEVENS CREEK BLVD | SANTA CLARA | CA | 95050 | USA |
| CELLI JOHN | BUS & MKTING MGR | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| COUCH JOHN D | SECTION MANAGER | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| CROCKETT DAVID | PRG MGR HP300 SYS | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| GARDNER LYNN | CUSTOMER RELATIONS | HEWLETT-PACKARD | 5303 STEVENS CREEK BLVD | SANTA CLARA | CA | 95050 | USA |
| GIMPLE BILL | R & D MGR HP300 PRG | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| GLOSS GREGORY C | COBOL PROJECT MANAGER | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| GRIFFIN MARY | MARKETING ENGINEER | HEWLETT-PACKARD | 5303 STEVENS CREEK BLVD | SANTA CLARA | CA | 95050 | USA |
| HATCHER BETH | | HEWLETT-PACKARD | 5301 STEVENS CREEK BLVD | SANTA CLARA | CA | 95050 | USA |
| HENRY WENDELL A | MEMEBERT OF TECH STAFF | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| KAM POLLY | PROGRAMMER/ANALYST | HEWLETT-PACKARD | 5301 STEVENS CREEK BLVD | SANTA CLARA | CA | 95050 | USA |
| KERNKE JUTTA C | PRODUCT MANAGER | HEWLETT-PACKARD | 5303 STEVENS CREEK BLVD | SANTA CLARA | CA | 95050 | USA |
| LARSON ORLAND J | IMAGE PRODUCT MANAGER | HEWLETT-PACKARD | 5303 STEVENS CREEK BLVD | SANTA CLARA | CA | 95050 | USA |
| LEVIN GREGG B | DESIGN ENGINEER | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| LEWIN ROBERT E | THIRD PARTY PGM MGR | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| MC CRACKEN ED | GEN MGR G S D | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| MENOLD BEN | S E DISTRICT MGR | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| PAGE JOHN | PROJECT MANAGER | HEWLETT-PACKARD | 5303 STEVEN CREEK | SANTA CLARA | CA | 95050 | USA |
| RIEGER DENNIS E | PRODUCT MANAGER - MPE | HEWLETT-PACKARD | 5303 STEVENS CREEK BLVD | SANTA CLARA | CA | 95050 | USA |
| SMITH HOWARD J | LAB MANAGER | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| SPALDING KENNETH G | DEVELOPMENT ENGINEER | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| TEMBROCT JOE R | DP MANAGER | HEWLETT-PACKARD | 5301 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| VAN KURAN PETER | PRODUCT MANAGER | HEWLETT-PACKARD | 5303 STEVENS CREEK BLVD | SANTA CLARA | CA | 95050 | USA |
| VELLANKI RAO C | SR PROGR/ANALYST | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| WILLIAMS RITA W | SUPPORT ENGINEER | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| YEO MARGE | | HEWLETT-PACKARD | 5301 STEVENS CREEK BLVD | SANTA CLARA | CA | 95050 | USA |
| ZABOR ELIAS | CUSTOMER RELATIONS | HEWLETT-PACKARD | 5303 STEVENS CREEK | SANTA CLARA | CA | 95050 | USA |
| CLIFTON ROY A | MPE SUPPORT MANAGER | HEWLETT-PACKARD | 972 BUCKEYE CT | SUNNYVALE | CA | 94086 | USA |
| COOPER PAUL D | SYSTEMS ENGINEER | HEWLETT-PACKARD | 4110 S 100 E AVE | TULSA | OK | 74145 | USA |
| FARRAGHER MICHAEL J | PROG SUPERVISOR | HEWLETT-PACKARD | 175 WYMAN STREET | WALTHAM | MA | 02154 | USA |
| D'ANGELO RICHARD J | SYSTEMS ENGINEER | HEWLETT-PACKARD CO | 32 HARTWELL AVE | LEXINGTON | MA | 02173 | USA |
| HUTCHISON PHILIP L | PRODUCT ENGINEER | HHEWLETT-PACKARD | 3400 E HARMONY RD | FORT COLLINS | CO | 80525 | USA |

| NAME | TITLE | COMPANY | ADDRESS | CITY | STATE | ZIP | COUNTRY |
|---|---|---|---|---|---|---|---|
| JOHNSON RAYMOND E | HP 3000 PRGM SERV MGR | HEWLETT-PACKARD GSD | 5303 STEVENS CRK BLVD | SANTA CLARA | CA | | USA |
| MANIES RALPH G | CUSTOMER RELATIONS | HEWLETT-PACKARD GSD | 5303 STEVENS CREEK BLVD | SANTA CLARA | CA | 95050 | USA |
| CALLANAN BILL | S E SUPERVISOR | HEWLETT-PACKARD LTD | KING ST LANE WINNERSH | WOKINGHAM BER | | | ENGLAN |
| MARQUEZ MATEO(CANCEL) | FIELD ENGINEER | HEWLETT-PACKARD MEXICO | AVENIDA PERIFERICO SUR 6501 | TEPEPAN | XO | 22 | MEXICO |
| COVITT MARC L | TECHNICAL SVCS MGR | HEWLETT-PACKARD SDD | 16399 W BERNARDO DR | SAN DIEGO | CA | 92127 | USA |
| LAW JACK | DATABASE ADMINSTRATOR | HEWLETT-PACKARD SDD | 16399 W BERNARDO DR | SAN DIEGO | CA | 92127 | USA |
| PRILL BOB (CANCELED) | OPERATIONS MANAGER | HEWLETT-PACKARD SDD | 16399 W BERNARDO DR | SAN DIEGO | CA | 92127 | USA |
| WATERS FRED | INFO SYSTEMS MANAGER | HEWLETT-PACKARD SDD | 16399 W BERNARDO DR | SAN DIEGO | CA | 92127 | USA |
| BORG CHARLES J | SYSTEMS PROGRAMMER | HEWLETT-PACKARD SRD | 1400 FOUNTAIN GROVE PK | SANTA ROSA | CA | 95404 | USA |
| LASLEY MICHAEL A | MIS MANAGER | HINDERLITER | 1240 N HARVARD | TULSA | OK | 74115 | USA |
| TATMAN PAUL E | SYSTEMS MANAGER | HMO SYSTEMS INC | 1235 RIVERSIDE | FT COLLINS | CO | 80521 | USA |
| HERBEL ERIC S | MEDICAL SYS ANALYST | HOECHST-ROUSSEL PHARM | RT 202-206 NORTH | SOMERVILLE | NJ | 08876 | USA |
| WINTON HUGH | D P MANAGER | HOERBIGER CORP | 35 LUMBER ROAD | ROSLYN | NY | 11576 | USA |
| KLEIN THOMAS C | MGR INFO SYSTEMS | HOOVER-NSK BEARING | 5400 S STATE RD | ANN ARBOR | MI | 48106 | USA |
| SOCIE LARRY J | PROGRAMMER | HOOVER-NSK BEARING | 5400 S STATE RD | ANN ARBOR | MI | 48106 | USA |
| BRUNK GREGG A | ADMINSTRATIVE ASSISTANT | HOUSTON INSTRUMENT | 8500 CAMERON ROAD | AUSTIN | TX | 78753 | USA |
| KRIGER WINSTON A | MGR TECHNICAL SALES | HOUSTON INSTRUMENT | 8500 CAMERON ROAD | AUSTIN | TX | 78753 | USA |
| SMITH JR HARRY R | DIRECTOR, COMPUTER CTR | HOWARD COMMUNITY COLLEGE | LITTLE PATUXENT PKWY | COLUMBIA | MD | 21044 | USA |
| BARTOLI JR CHESTER T | OPERATIONS SUPERVISOR | HP AVONDALE DIV | ROUTE 41 & STARR RD | AVONDALE | PA | 19311 | USA |
| CHEN CHANG L | SYSTEM ANALYST | HP AVONDALE DIV | ROUTE 41 & STARR RD | AVONDALE | PA | 19311 | USA |
| HINES RELLA M | EXECUTIVE DIRECTOR | HP GEN SYS USERS GROUP | P O BOX 18813 | BALTIMORE | MD | 21240 | USA |
| BECHER ANTHONY H | HEAD C.A.M. ENG | HUGHES AIRCRAFT CO | BOX 3310 BLDG 607/B329 | FULLERTON | CA | 92634 | USA |
| GALUSKY DIANE M | SUPERVISOR APPL DEV | HUGHES AIRCRAFT CO | P O BOX 3310 607/E331 | FULLERTON | CA | 92634 | USA |
| GULICK LLOYD R | COMPUTING SPECIALIST | HUGHES AIRCRAFT CO | P O BOX 3310 607/E331 | FULLERTON | CA | 92634 | USA |
| MECHAM DOUGLAS J | SYSTEMS COORDINATOR | HUGHES AIRCRAFT CO | P O BOX 3310 601-4219 | FULLERTON | CA | 92634 | USA |
| THEISSEN WILHELM J | MANAGER BUSINESS SYST | HUGHES AIRCRAFT CO | P O BOX 3310 607/B318 | FULLERTON | CA | 92634 | USA |
| TRAPP ROBERT E | TEST ENGINEER | HUGHES AIRCRAFT CO | 1901 W MALVERN AVE | FULLERTON | CA | 92634 | USA |
| GRIFFIN RICK | PRESIDENT | I C S SERVICES, INC | 2131 W EULESS BLVD | EULESS | TX | | USA |
| MILLER STEPHEN L | COMP PROD SUPVR II | IL DEPT OF CORRECTION | 200 W WASHINGTON | SPRINGFIELD | IL | | USA |
| PETTERCHAK JOHN J | INFO SYST EXECUTIVE | IL DEPT OF CORRECTION | 200 N WASHINGTON | SPRINGFIELD | IL | | USA |
| COLDREN J DAVID | ASSOCIATE DIRECTOR | IL LAW ENFORCEMENT CO | 120 S RIVERSIDE PLAZA | CHICAGO | IL | 60606 | USA |
| MAIER EDWARD F | INFO SYSTEMS EXECUTIVE | IL LAW ENFORCEMENT CO | 120 S RIVERSIDE PLAZA | CHICAGO | IL | 60606 | USA |
| MAY NORMAN F | SR SYSTEMS ANALYST | IL LAW ENFORCEMENT CO | 120 S RIVERSIDE PLAZA | CHICAGO | IL | 60606 | USA |
| GRADY MICHAEL K | SYSTEMS CONSULTANT | ILLINOIS DEPT LAW & ORDER | 200 W WASHINGTON | SPRINGFIELD | IL | | USA |
| OUELLETTE MARC E | SOFTEWARE SPECIALIST | INCO LIMITED | 1 FIRST CANADIAN PL | TORONTO | ON | M5X1C4 | CANADA |
| BRODOWSKI RICHARD E | DIRECTOR INFO SYSTEMS | IND PRESS-TELEGRAM | 604 PINE AVE | LONG BEACH | CA | | USA |
| HALL CRAIG T | PRESIDENT | INFO TRONIC SYSTEMS | 449 HOWARD AVE | HOLLAND | MI | 49423 | USA |
| CAMARILLO MARIO R | DIR DE SISTEMAS | INFORMATICA DESC, SC | THIERS 248 ANZUREZ | MEXICO | DF | | MEXICO |
| MAGDALENO JESUS | GTE DE PROYECTOS | INFORMATICA DESC, SC | THIERS 248 ANZUREZ | MEXICO | DF | | MEXICO |
| STOVER RAY J | | INFORMATION RESOURCES | 4905 LIMA | DENVER | CO | | USA |
| BEMAN ROGER | V P MARKETING | INFORMATION RESOURCES | 7935 E PRENTICE | ENGLEWOOD | CO | 80111 | USA |
| WADE RON | SYSTEMS ANALYST | INFORMATION RESOURCES | 7935 E PRENTICE AVE | ENGLEWOOD | CO | 80111 | USA |
| KLEIN JAMES D | SYSTEMS PROGRAMMER | INFORMATION TERMINALS | 323 SOQUEL WAY | SUNNYVALE | CA | 94086 | USA |
| BRYDEN WILLIAM | PRESIDENT | INLAND SYSTEMS ENGR | 424 BEVERLY DR | REDLANDS | CA | 92373 | USA |
| MOORE GEORGE E | OPERATIONS MANAGER | INTERACTIVE APPL INC | 505 HAMILTON AVE #103 | PALO ALTO | CA | 94301 | USA |
| MACHIN JOHN | MANAGING PARTNER | INTERCOMP SERVICES | 459 COLLINS STREET | MELBOURNE | VI | 3000 | AUSTRA |
| GRICE FRANK H | PRESIDENT | INTERTEC | 2625 PARK BLVD | PALO ALTO | CA | 94306 | USA |
| RAUH JOSEPH E | TECHNICAL SHOWMAN | ISSCO | 4186 SORRENTO VALLEY BLVD | SAN DIEGO | CA | 92121 | USA |
| SYNOLD PRISCILLA J | MANAGER TECHNICAL SPT | ISSCO | 4186 SORRENTO VALLEY BLVD | SAN DIEGO | CA | 92121 | USA |
| WANDERMAN KENNETH A | MANAGER COMPUTER SYS | ITEL CORP | 1 EMBARCADERO | SAN FRANCISCO | CA | 94114 | USA |
| MERSHON ROBERT C | PROJECT MANAGER | ITT FINANCIAL | P O BOX 250 | CHIPPEWA FALL | WI | 54729 | USA |
| VOILES DUANE A | SR SYSTEMS ANALYST | ITT FINANCIAL | P O BOX 250 | CHIPPEWA FALL | WI | 54729 | USA |

| NAME | TITLE | COMPANY | ADDRESS | CITY | STATE | ZIP | COUNTRY |
|---|---|---|---|---|---|---|---|
| PETERSON MERRILL A | STAFF ENGINEER | JEFFERSON CHEMICAL | P O BOX 847 | PORT NECHES | TX | 77651 | USA |
| EDWARDS JON L | | JENNISON ASSOCIATES | 270 PARK | NEW YORK | NY | 10017 | USA |
| CAYLOR LARRY W | DATA PROCESSING MANAGER | JOHANSON DIELECTRICS | 2210 SCREENLAND DRIVE | BURBANK | CA | 91505 | USA |
| BLACK DAVID T | SYSTEMS ANALYST | JOHN HENRY COMPANY | P O BOX 17099 | LANSING | MI | 48901 | USA |
| HAMAN VINCE J | DIRECTOR DP | JOHNSON COUNTY | P O BOX 2510 | IOWA CITY | IA | 52240 | USA |
| WOMBACHER ERNEST J | CLERK OF DISTRICT COURT | JOHNSON COUNTY IOWA | 400 S CLINTON STREET | IOWA CITY | IA | 52240 | USA |
| ALLAN WILLIAM D | INFO SERVICES MANAGER | KAISER FOUNDATION | 2005 FRANKLIN ST | DENVER | CO | 80205 | USA |
| GUISINGER ERIC L | PROGRAMMER/ANALYST SR | KAISER FOUNDATION | 2005 FRANKLIN ST | DENVER | CO | 80205 | USA |
| LUITHLE WILLIARD H | COMPUTER OPS SUPV | KAISER FOUNDATION | 2005 FRANKLIN | DENVER | CO | 80205 | USA |
| DUNCAN JAMES J | MANAGER OEM SALES | KAPPA SYSTEMS INC | 1409 POTTER DR | COLORADO SPRI | CO | 80909 | USA |
| BENJAMIN ROBERT A | SR SYSTEMS PROGRAMMER | KEYDATA CORP | 1400 WILSON BLVD | ARLINGTON | VA | 22209 | USA |
| DAVY CHRIS | MANAGER | KEYDATA CORP | 1400 WILSON BLVD | ARLINGTON | VA | 22209 | USA |
| DAVY CHRISTOPHER | PRODUCT MANAGER | KEYDATA CORP | 1400 WILSON BLVD | ARLINGTON | VA | 22209 | USA |
| FELDMAN DAN | | KEYDATA CORP | 108 WATER ST | WATERTOWN | MA | 02172 | USA |
| ROUSSEAU ALLEN F | VP PRODUCT DEV | KEYDATA CORP | 108 WATER ST | WATERTOWN | MA | 02172 | USA |
| HARMAN LAWRENCE K | GROUP LEADER | LAWRENCE LIVERMORE LAB | P O BOX 808 MC L-389 | LIVERMORE | CA | 94550 | USA |
| JESSEN TIM D | COMPUTER SCIENTIST | LAWRENCE LIVERMORE LAB | P O BOX 808 L-414 | LIVERMORE | CA | 94550 | USA |
| LACY LEROY P | COMPUTER SCIENTIST | LAWRENCE LIVERMORE LAB | P O BOX 808 MC L-389 | LIVERMORE | CA | 94550 | USA |
| DREILING CHERYL B | COMPUTER SCIENTIST | LAWRENCE LIVERMORE LABOR | P O BOX 808 | LIVERMORE | CA | 94550 | USA |
| JONES DANNY A | COMPUTER PROGRAMMER | LAWRENCE LIVERMORE LABOR | P O BOX 808 | LIVERMORE | CA | 94550 | USA |
| LYNN DEAN E | COMPUTER SCIENTIST | LAWRENCE LIVERMORE LABOR | P O BOX 808 | LIVERMORE | CA | 94550 | USA |
| LEIGHT BETSY(CANCELED) | DIRECTOR | LEIGHT AND ASSOCIATES | 14200 SHOLES COURT | LOS ALTOSHILL | CA | 94022 | USA |
| FLYNN DOUGLAS C | D P MANAGER | LEXINGTON HERALD LEAD | 239 WEST SHORT ST | LEXINGTON | KY | 40507 | USA |
| MANEWAL SANDRA S | SYSTEMS MANAGER | LIBERTY COMMUNICATIONS INC | 2225 COBURG ROAD | EUGENE | OR | 97401 | USA |
| EATON JOHN | DIR COMPUTER SERVICES | LONDON GRAD SCH OF BUS | REGENT'S PARK | LONDON | | NW14SA | ENGLAN |
| GATES BILL | DATA PROCESSING MGR | LONGS DRUG STORES, INC | 141 NORTH CIVIC DR | WALNUT CREEK | CA | 94596 | USA |
| MOIRAO DAVID E | PROGRAMMING MANAGER | LONGS DRUG STORES, INC | 141 NORTH CIVIC DR | WALNUT CREEK | CA | 94596 | USA |
| GORFINKEL MARTIN | PARTNER | LOS ALTON RESEARCH CENTER | 339 S SAN ANTONIO ROAD | LOS ALTOS | CA | 94022 | USA |
| ARMSTRONG JACK C | PARTNER | LOS ALTOS RESEARCH CENTER | 339 S SAN ANTONIO ROAD | LOS ALTOS | CA | 94022 | USA |
| EXCOFFON MARGOT M | PROJECT LEADER | LYNES UNITED SERVICES | 309 2ND AVE SW | CALGARY | AL | T2P0C5 | CANADA |
| LARSON TERRY | PROJECT LEADER | LYNES UNITED SERVICES | 309 2ND AVE SW | CALLARY | AL | T2P0C5 | CANADA |
| MAUS JOHN A | SYSTEMS ANALYST | M.H. GOLDEN CO | 123 CAMINO DE LA REINA | SAN DIEGO | CA | 92108 | USA |
| JOHNSTON FRANK H | DIRECTOR DATA PROCESS | MACON TELEGRAPH PUB CO | P O BOX 4167 | MACON | GA | 31208 | USA |
| KROESEN JACOBUS A | MANAGER O & A | MAKRO INTL | CHURCHILL LAAN 11 | 3525 GVUTRECH | | | NETHER |
| VAN BUITENEN PETER | SYSTEMS DEDMAN | MAKRO INTL | CHURCHILL LAAN 11 | 352F GUUTRECH | | | NETHER |
| ANKERS, JR ALFRED H | MANAGER DATA PROCESSING | MARITIME TERMINALS | 7737 HAMPTON BLVD | NORFOLK | VA | 23505 | USA |
| IZETT CRAIG N | DIRECTOR TECHNOLOGY | MARTIN MARIETTA | 300 EAST JOPPA RD | BALTIMORE | MD | 21204 | USA |
| BOYER CONNIE Y | PROGRAMMER/ANALYST | MARY WASHINGTON CLG | P O BOX 1081 CLG STAT | FREDERICKSBUR | VA | 22401 | USA |
| BROWN DALE A | DIR OF INST RESEARCH | MARY WASHINGTON CLG | P O BOX 1081 CLG STAT | FREDERICKSBUR | VA | 22401 | USA |
| DAILEY JOSEPH D | MANAGER DATA PROCESSING | MASTERCRAFT IND | 4881 IRONTON ST | DENVER | CO | 80239 | USA |
| THORNTON WILLIAM L | PROGRAMMER/ANALYST | MASTERCRAFT IND | 4881 IRONTON ST | DENVER | CO | 80239 | USA |
| OCHI YOSHIAKI | EDP MANAGER | MATSUSHITA ELEC INDUST CO | #2 MATSUSHITA MACHI | MORIGUCHI-SHI | OS | | JAPAN |
| ANDERSON GARY D | ASSOC PROF OF BIOSTAT | MC MASTER UNIVERSITY | 1200 MAIN ST WEST | HAMILTON | ON | L8S4J9 | CANADA |
| CLARK KIM | | MC MASTER UNIVERSITY | 1200 MAIN STREET WEST | HAMILTON | ON | L8S4J9 | CANADA |
| GILCHRIST DON | | MC MASTER UNIVERSITY | 1200 MAIN ST WEST | HAMILTON | ON | L8S4J9 | CANADA |
| SINHA DEEPAK | PROGRAMMER | MC MASTER UNIVERSITY | 1200 MAIN ST W | HAMILTON | ON | L8S4J9 | CANADA |
| SNELLINGS FRANK W | PROJECTS DEV MANAGER | MEDIA GENERAL INC | 301 E GRACE ST | RICHMOND | VA | 23219 | USA |
| WALESKI JR WALTER L | DIR INFO SYSTEMS | MEDIA GENERAL INC | 301 E GRACE ST | RICHMOND | VA | 23219 | USA |
| GROSSCUP LORIN | PROGRAMMER | MERCHANDISING METHODS | 274 BRANNAN STREET | SAN FRANCISCO | CA | 94107 | USA |
| HUXHOLD PHIL W | DP MANAGER | MERCHANDISING METHODS | 274 BRANNAN STREET | SAN FRANCISCO | CA | 94107 | USA |
| CLEMENTSON GERHARDT C | DIR OF ACAD COMP CNTR | METROPOLITAN ST COLLEGE | 1006 11TH STREET | DENVER | CO | 80204 | USA |
| GENTRY THOMAS L | MGR ENG'G COMPUTER CTR | MICROWAVE ASSOCIATES | SOUTH AVE | BURLINGTON | MA | 01803 | USA |

| NAME | TITLE | COMPANY | ADDRESS | CITY | STATE | ZIP | COUNTRY |
|---|---|---|---|---|---|---|---|
| SMITH TERRY B | CHIEF OF DATA PROCESSING | MILO BEAUTY SUPPLY CO | 4670 ALLEN ROAD | STOW | OH | 44224 | USA |
| MC INNIS, JR. A MARVI | | MINI/MICRO SYSTEMS INC | 3315 N SHARTEL AVE | OKLAHOMA CITY | OK | 73118 | USA |
| DEBOK LOWELL W | PROGRAMMER/ANALYST | MITCHELL BROS TRUCK | 3841 N COL BLVD | PORTLAND | OR | 97217 | USA |
| OLSEN MARY ANN(CANCEL) | PROGRAMMER | MITCHELL BROS TRUCK | 3841 N COLUMBIA BLVD | PORTLAND | OR | 97217 | USA |
| DUNCOMBE BRIAN C | | MOHAWK COLLEGE | P O BOX 2034 | HAMILTON | ON | L8N3T2 | CANADA |
| KWAVNICK MYER | | MONTREAL CHILDRENS HOSPITA | 2300 TUPPER | MONTREAL | QU | H3H1P3 | CANADA |
| WAGNER DAVID L | MANAGER DATA PROCESSING | MOORE & CO | 300 E SPEER BLVD | DENVER | CO | 80203 | USA |
| DINAN DENNIS M | SENIOR PROGRAMMER ANALYS | MORGAN GUARANTY TRUST | 23 WALL STREET | NEW YORK | NY | 10015 | USA |
| ENTIS GLENN M | PROGRAMMER ANALYST T | MORGAN GUARANTY TRUST | 23 WALL STREET | NEW YORK | NY | 10015 | USA |
| MALUS JOSEPH T | SENIOR PROGRAMMER ANALYS | MORGAN GUARANTY TRUST | 37 WALL STREET | NEW YORK | NY | 10015 | USA |
| TETI FRANK A | SYSTEM ANALYST | MORGAN GUARANTY TRUST | 23 WALL STREET | NEW YORK | NY | 10015 | USA |
| KNIGHT JR WILLIAM J | MANAGER INFORMATION | MULTIVEST INC | 6452 N FEDERAL HWY | FT LAUDERDALE | FL | 33308 | USA |
| CHASE LARRY M | SUPV SYS & PROGM | MULTNOMAH COUNTY ESD | 220 SE 102 | PORTLAND | OR | 97216 | USA |
| MAGNUS ANN M | SUPV COMPUTER OPER | MULTNOMAH COUNTY ESD | 220 SE 102 | PORTLAND | OR | 97216 | USA |
| STAMBAUGH JAN R | DIRECTOR DP | MULTNOMAH COUNTY ESD | 220 SE 102 | PORTLAND | OR | 97216 | USA |
| LAVIOLA ANTHONY | SYSTEMS MANAGER | N Y TELEPHONE CO | 375 PEARL ST | NEW YORK | NY | 10038 | USA |
| REITHNER JR ROBERT M | MANAGER | N.E.R.A. | 80 BROAD STREET | NEW YORK | NY | 10004 | USA |
| RUSSELL KENT A | PRESIDENT | NATIONAL COMPUTER CORP | 3000 34TH ST | METAIRIE | LA | 70001 | USA |
| PRICE RICHARD J | MGR TECH SUPPORT | NATIONWIDE FINANCE | 700 OFFICE PARKWAY | CREVE COEUR | MO | 63141 | USA |
| RICKARD C ALLEN | MGR SYSTEMS DEVELOPMENT | NATIONWIDE FINANCE | 700 OFFICE PARKWAY | CREVE COEUR | MO | 63141 | USA |
| ROWE THOMAS C | V P SYSTEMS | NATIONWIDE FINANCE | 700 OFFICE PARKWAY | CREVE COEUR | MS | 63141 | USA |
| SULLIVAN DENNIS J | MGR D P OPERATIONS | NATIONWIDE FINANCE | 700 OFFICE PARKWAY | CREVE COEUR | MO | 63141 | USA |
| EDWARDS ROBERT M | SPEC ASST FOR DP | NATL CONF ST LEGISLATURES | 444 N CAPITOL ST NW | WASHINGTON | DC | 20001 | USA |
| JEWEL MARTIN D | MGR COMPUTER SERVICES | NATL SCI DATA CENTER | 1130 E MC DOWELL RD | PHOENIX | AZ | 85006 | USA |
| LIENARD JAMES B | PROGRAMMER | NATL SCI DATA CENTER | 1130 E MC DOWELL RD | PHOENIX | AZ | 85006 | USA |
| HARRIS DAVID W | MANAGER | NB5 FINANCIAL SERVICES | 500 W WILSON BDG RD | COLUMBUS | OH | 43285 | USA |
| CURBELO RALPH | COST ANALYST | NEW YORK TELEPHONE CO | 1095 AVE OF THE AMERICAS | NEW YORK | NY | 10036 | USA |
| KILLCOMMONS PETER F | GENERAL COST SUPV | NEW YORK TELEPHONE CO | 1095 AVE OF THE AMERICAS | NEW YORK | NY | 10036 | USA |
| BROWN D DAVID | PRESIDENT | NICE CORPORATION | 4357 AIRPORT PARK PLAZA | ODGEN | UT | 84403 | USA |
| JEANS KENNETH E | ASST MGR OF DATA PROC | NOOTER CORP | P O BOX 451 | ST LOUIS | MO | 63166 | USA |
| ROGERS THOMAS D | ASSISTANT DIRECTOR | NORFOLK STATE COLLEGE | 2401 CORPREW AVE | NORFOLK | VA | 23504 | USA |
| MC CLAIN MALCOLM E | MANAGER DATA PROCESSING | NORTH IDAHO COLLEGE | 1000 W GARDEN AVENUE | COEUR D' ALEN | ID | 83814 | USA |
| LULICH LEO J | PROGRAMMER | NORTHERN SPECIALTY | 6635 N BALTIMORE | PORTLAND | OR | 97203 | USA |
| MILLER LEROY M | D P MANAGER | NORTHERN SPECIALTY SALES | 6635 N BALTIMORE | PORTLAND | OR | 97203 | USA |
| RODGER JOHN D | | NORTHERN TELECOM | 8200 DIXIE RD | BRAMPTON | ON | L6V2M6 | CANADA |
| WHIDDON ROY L | SPECIALIST SOFTWARE | NORTHERN TELECOM LTD | 33 CITY CENTRE DRIVE | MISSISSAUGA | ON | L5B2N5 | CANADA |
| POLO FRANK | SYSTEMS SPECIALIST | NORTHROP CORP | 2301 WEST 120 STREET | HAWTHORNE | CA | 90250 | USA |
| KROPP MICHAEL E | SYSTEMS PROGRAMMER | NORWCH-EATON PHARM | 13-17 EATON AVENUE | NORWICH | NY | 13815 | USA |
| SPORKEN HEIN P | | NOVA AUTOMATION CONS | 30 NEDEREIND | NIEUWEGEIN | | | NETHER |
| SHROADS, JR JAMES C | DIRECTOR COMP OPER | NPD REASEARCH, INC | 15 VERBENA AVENUE | FLORAL PARK | NY | 11001 | USA |
| BISHOP LARRY | DIRECTOR SYSTEMS DEVELOP | NPD RESEARCH, INC | 15 VERBENA AVENUE | FLORAL PARK | NY | 11001 | USA |
| WEBER WILLIAM | SYSTEMS ANALYST | NPD RESEARCH, INC | 15 VERBENA AVENUE | FLORAL PARK | NY | 11001 | USA |
| HOLMAN JAMES R | HEAD STATISTICS LAB | OHIO AGRIC R&D CENTER | | WOOSTER | OH | 44691 | USA |
| HUNTER JOHN C | SYSTEMS SUPERVISOR | OKANAGAN HELICOPTERS | 4391 AGAR DRIVE | VANCOUVER | BC | V7B1A5 | CANADA |
| MILLARD MICHAEL J | PROGRAMMING SUPERVISOR | OKANAGAN HELICOPTERS | 4391 AGAR DRIVE | VANCOUVER | BC | V7B1A5 | CANADA |
| CRAWFORD JEFFREY D | PRESIDENT | OMEGA SYSTEMS, INC. | 3720 SO ROCKWELL ST | CHICAGO | IL | 60632 | USA |
| HICKS DAVID L | PROGRAMMER ANALYST | PACIFIC MUTUAL | 700 NEWPORT CENTER DRIVE | NEWPORT BEACH | CA | 92660 | USA |
| VILLA JR CHARLES J | PRESIDENT | PANTECHNIC | 5805 OCEAN VIEW DR | OAKLAND | CA | 94618 | USA |
| WRIGHT DALE | | PANTECHNIC | 5805 OCEAN VIEW DR | OAKLAND | CA | 94618 | USA |
| BUTLER STEPHEN M | DIRECTOR DATA PROCESSING | PARADISE VALLEY HOSPITAL | 2400 E 4TH STREET | NATIONAL CITY | CA | 92050 | USA |
| PARKER KENNETH | DIRECTOR E D P | PARISIAN INC | 1101 26TH STREET N | BIRMINGHAM | AL | 35234 | USA |
| SCHLOSSER JR ROBERT J | COMPUTER SCIENTIST | PEPCO | 1900 PENNA AVE | WASHINGTON | DC | 20068 | USA |

| NAME | TITLE | COMPANY | ADDRESS | CITY | STATE | ZIP | COUNTRY |
|---|---|---|---|---|---|---|---|
| ARNOLD PAT | SYSTEM ENGINEER | PETRO CANADA | P O BOX 2844 | CALGARY | AL | T2P2M7 | CANADA |
| CURTIS LINDA | ANALYST/PROGRAMMER | PETRO CANADA | P O BOX 2844 | CALGARY | AL | T2P2M7 | CANADA |
| DURHAM PAUL H | SYSTEMS ANALYST PROG | PETRO CANADA INC | 400 4 AVE SW | CALGARY | AL | T2P2M7 | CANADA |
| NEMETH LOUIS E | DIRECTOR ENG COMP CTR | PHILA WATER DEPT | 1270 MSB | PHILADELPHIA | PA | 19107 | USA |
| GARDINER JAMES A | MIS MANAGER | PILKINGTON BROS | 685 WARDEN AVE | SCARBOROUGH | ON | M1L3Z8 | CANADA |
| METZNER BERNIE | OPERATIONS MGR | PILKINGTON GLASS LTD | 685 WARDEN AVE | TORONTO | ON | M1L3X7 | CANADA |
| STOCKDALE WALLACE L | DP MANAGER | PORT OF OAKLAND | 66 JACK LONDON SQ | OAKLAND | CA | 94606 | USA |
| HUXHAM BASIL C | MANAGER DATA PROCESSING | PORT OF VANCOUVER | 1300 STEWART ST | VANCOUVER | BC | V5L4X5 | CANADA |
| MC CREA ROBERT B | CHIEF FINANCIAL OFFICER | PORT OF VANCOUVER | 1300 STEWART ST | VANCOUVER | BC | V5L4X5 | CANADA |
| SMITH RAYMOND E | DATA PROCESSING MANAGER | POWERS REGULATOR | RR #1 | BEETON | ON | L0G1A0 | CANADA |
| ARCAYA PEDRO M | MANAGER | PROCESASEG, SA | TORRE LA PREVISORA | PISO SABANA G | | | VENEZU |
| TANKERSLEY JAMES J | SR SYSTEMS ANALYST | PROCTER & GAMBLE | P O BOX 599 | CINCINNATI | OH | 45201 | USA |
| LUMB ARTHUR C | CONSULTANT | PROCTER & GAMBLE CO | 7162 READING RD | CINCINNATI | OH | 45208 | USA |
| MEYER DIANNE M | OPERATIONS COORDINATOR | PROCTER & GAMBLE CO | 6105 CENTER HILL RD | CINCINNATI | OH | 45220 | USA |
| VEENEMAN WILLIAM E | SYSTEMS ANALYST | PROCTER & GAMBLE CO | 6105 CENTER HILL RD | CINCINNATI | OH | 45220 | USA |
| COX W PHIL | SYSTEMS ANALYST | PROCTOR & REDFERN | 75 EGLINTON AVE E | TORONTO | ON | M4P1H3 | CANADA |
| GOODMAN ROBERT A | PRESIDENT | PROF COMP SERV | 2821 E 28TH STREET | LONG BEACH | CA | 92648 | USA |
| CARVALHO MARCOS A X | SYSTEMS ANALYST | PROMON ENGENHARIA S A | NOVE DE JULHO 4939 | SAO PAULO | SP | 01407 | BRAZIL |
| TICHAUER MARIO F W | ENGINEER | PROMON ENGENHARIA S A | NOVE DE JULHO 4939 | SAO PAULO | SP | 01407 | BRAZIL |
| MOWER MONTE J | SYSTEM MANAGER | PROVO SCHOOL DISTRICT | 280 WEST 800 NORTH | PROVO | UT | 84601 | USA |
| SMITH BARRY | | PRUDENTIAL INSURANCE | 213 WASHINGTON ST | NEWARK | NJ | 07101 | USA |
| FARIA JOHN | MGR COMPUTER SERVICES | PRUDENTIAL REINSURANCE | 213 WASHINGTON ST | NEWARK | NJ | 07101 | USA |
| BARKER DAVID A | MANAGER SYSTEMS & PROG | PURITAN INSURANCE CO | 1515 SUMMER ST | STAMFORD | CT | 06905 | USA |
| PACHALY FRED A | PROGRAMMER | PURITAN INSURANCE CO | 1515 SUMMER ST | STAMFORD | CT | 06905 | USA |
| DUMMER SHEILA T | MANAGER FINANCIAL SYSTEM | QUME CORP | 2323 INDUSTRIAL PARKWAY | HAYWARD | CA | 94545 | USA |
| FISCHER LEE H | MANAGER DATA PROCESSING | QUME CORP | 2323 INDUSTRIAL PARKWAY | HAYWARD | CA | 94545 | USA |
| CHATFIELD DENNIS C | SYSTEM MANAGER | R J FRISBY | 1500 CHASE | ELK GROVE | IL | 60007 | USA |
| RASMUSSEN BENT | VICE PRESIDENT | R SHRIVER ASSOCIATES | 120 LITTLETON ROAD | PARSIPPANY | NJ | 07054 | USA |
| WHITE RUSS | PRESIDENT | R SHRIVER ASSOCIATES | 120 LITTLETON ROAD | PARSIPPANY | NJ | 07054 | USA |
| WILSON ROBERT L | SALES MGR | R SHRIVER ASSOCIATES | 120 LITTLETON ROAD | PARSITPANY | NJ | 07054 | USA |
| MULVIHILL GARY | | R SHRIVER ASSOCIATES | 1530 CHESTNUT SUITE 714 | PHILADELPHIA | PA | 19102 | USA |
| MAHONEY LARRY | SUPERVISOR SEATTLE COMP | R W BECK & ASSOCIATES | 200 TOWER BUILDING | SEATTLE | WA | 98101 | USA |
| ONEY JAMES B | DATA PROCESSING MANAGER | RCM | ONE MARKET PLAZA 3910 | SAN FRANCISCO | CA | 94105 | USA |
| SOHNLE RONALD C | MGR COMPUTING SERVICES | REGIONAL ECONOMIC EXPANSIO | 601 SPADINA CRESCENT | SASKATOON | SA | S7K3G8 | CANADA |
| SCOTT GEORGE B | MANAGER | REICHHOLA CHEM | 2340 TAYLOR WAY | TACOMA | WA | 98401 | USA |
| FRANSEN CRAIG S | SYSTEM ANALYST | REICHHOLD CHEM | 2340 TAYLOR | TACOMA | WA | 98401 | USA |
| FREDRICKSON GUNNARD A | PROJECT LEADER | RELIANCE ELECTRIC | 150 CANTERBURY DR | ATHENS | GA | | USA |
| SNEED JAMES E | CORPORATE MGR D P | RELIANCE ELECTRIC | P O BOX 5065 STATION B | GREENVILLE | SC | 29606 | USA |
| MC CORMICK DOUGLAS B | MGR INFORMATION SYS | REPUBLIC GEOTHERMAL | 11823 E SLAUSON #1 | SANTA FE SPRI | CA | 90670 | USA |
| WRIGHT STEVE T | MANAGER HOME OFFICE | REPUBLIC MORTGAGE INS | P O BOX 2514 | WINSTON SALEM | NC | 27102 | USA |
| GREEN ANNABELLE H | SECRETARY/TREASURER | ROBELLE CONSULTING LTD | #130-5421 10TH AVENUE | DELTA | BC | V4M3T9 | CANADA |
| GREEN ROBERT M | PRESIDENT | ROBELLE CONSULTING LTD | #130-5421 10TH AVENUE | DELTA | BC | V4M3T9 | CANADA |
| LEWIS KERMON | V P DATA PROCESSING | ROBERT JAMES CO | | BIRMINGHAM | AL | | USA |
| BEHELER ANN F | PROJECTS MANAGER | ROCKWELL INTL-COLLINS | 1200 N ALMA RD | RICHARDSON | TX | 75081 | USA |
| GEYER SANFORD A | SYSTEM MANAGER | ROLM CORP | 4900 OLD IRONSIDES DR | SANTA CLARA | CA | 95050 | USA |
| GWALTHNEY DAVID L | DIRECTOR-CCIS | RUTGERS UNIVERSITY | 311 N FIFTH STREET | CAMDEN | NJ | 08102 | USA |
| FRYER WILLIAM R | PRESIDENT | S.B.D.P. | 4208 AIRPORT ROAD | CINCINNATI | OH | 45226 | USA |
| KIRK TIM L | DIRECTOR INFO SYSTEMS | SACRED HEART GEN HOSP | P O BOX 10905 | EUGENE | OR | 97401 | USA |
| PHILLIPS TERRY | DEVELOPMENT MANAGER | SACRED HEART GEN HOSP | P O 10905 | EUGENE | OR | 97401 | USA |
| BRUCE CLIFF | D P ASST MANAGER | SAN JOSE MERCURY-NEWS | 750 RIDDER PARK DR | SAN JOSE | CA | 95190 | USA |
| GOT TERRENCE | D P MANAGER | SAN JOSE MERCURY-NEWS | 750 RIDDER PARK DR | SAN JOSE | CA | 95190 | USA |
| NORRIS BOB | PROGRAMMER ANALYST | SAN JOSE MERCURY-NEWS | 750 RIDDER PARK DR | SAN JOSE | CA | 95190 | USA |

| NAME | TITLE | COMPANY | ADDRESS | CITY | STATE | ZIP | COUNTRY |
|---|---|---|---|---|---|---|---|
| MARTIN STEVE T | SYSTEMS ANALYST | SANDIA LABORATORIES | DIVISION 2627 | ALBUQUERQUE | NM | 87185 | USA |
| JAMES ROBERT M | ANALYST/PROGRAMMER | SANDIA LABS | P O BOX 5800 | ALBUQUERQUE | NM | 87185 | USA |
| FLOWERS CURTIS J | DP ANALYST II | SANGAMON STATE UNIV | SHEPHERD ROAD | SPRINGFIELD | IL | 62708 | USA |
| CARR CHARLES E | MAN ANALYST PROG II | SANGAMON STATE UNIVERSITY | SHEPHERD ROAD | SPRINGFIELD | IL | 62708 | USA |
| KLETT DONALD S | DIRECTOR UNIVERSITY LAB | SANGAMON STATE UNIVERSITY | SHEPHERD ROAD | SPRINGFIELD | IL | 62708 | USA |
| FINNEY BILLIANNA M | PROGRAMMER | SANTABARBARA RESEARCH | 75 COROMAR DRIVE | GOLETA | CA | 93017 | USA |
| VIOHL KEN | | SATELLITE COMPUTING, INC | PO BOX 2015 | NORFOLK | VA | 23501 | USA |
| ST PIERRE JEAN | DP MANAGER END | SCOTT PAPER LTD | P O BOX 760 | CRABTREE | PQ | | CANADA |
| ADAMS BOB | SYSTEMS EDP MANAGER | SCOTT PAPER LTD | P O BOX 760 | NEW WESTMINST | BC | | CANADA |
| KING NEIL R | PROGRAMMING SUPV | SCOTT PAPER LTD | P O BOX 760 | NEW WESTMINST | BC | | CANADA |
| MC LEOD PAT | PROGRAMMER | SCOTT PAPER LTD | P O BOX 760 | NEW WESTMINST | BC | | CANADA |
| SHELLEY NANCY | SYSTEMS PROGRAMMER | SCOTT PAPER LTD | P O BOX 760 | NEW WESTMINST | BC | | CANADA |
| WARP CRAIG | | SHUGART ASS | 435 OAKMEAD PARKWAY | SUNNYVALE | CA | 94086 | USA |
| SHOUP BRYAN | DATA PROCESSING MGR | SII DYNA-DRILL | 1771 DEERE AVENUE | IRVINE | CA | 92713 | USA |
| SEWELL DAVID | DIRECTOR OF DATA PROCESS | SII SERVCO | PO BOX 880 | GARDENA | CA | 90247 | USA |
| DEMEVSY JIM | | SILTON DATA INC | 2407 E 38TH STREET | VERNON | CA | 90058 | USA |
| SMART JOHN E | VICE PRESIDENT | SILTON DATA INC | 2407 E 38TH STREET | VERNON | CA | 90058 | USA |
| SPIELER CHARLES W | DIRECTOR MIS | SIMCO | 21001 CABOT BLVD | HAYWARD | CA | 94545 | USA |
| SARBAUGH JAY C | VICE PRESIDENT | SMALL BUSINESS DATA PROC | 4208 AIRPORT ROAD | CINCINNATI | OH | 45226 | USA |
| HOUY DAVID J | FINANCIAL SYSTEMS MGR | SMITH INTERNATIONAL | 4343 VON KARMAN AVE | NEWPORT BEACH | CA | 92660 | USA |
| ADAMS KEARNEY A | SR PROGRAMMER ANALYST | SMITH KLINE INST | 880 W MAUDE AVENUE | SUNNYVALE | CA | 94086 | USA |
| BEHNKEN H PAUL | SUPVR DATA PROCESSING | SMITHS INDUSTRIES | P O BOX 5389 | CLEARWATER | FL | | USA |
| KENDALL JOHN | SYSTEMS PROGRAMMER | SO MISSIONARY COLLEGE | | COLLEGEDALE | TN | 37315 | USA |
| BAKST LAWRENCE E | SENIOR CONSULTANT | SOFTWARE SYSTEMS TECH | 39 BROADWAY 32ND FL | NEW YORK | NY | 10006 | USA |
| BANSAL A K | | SOHIO | | CLEVELAND | OH | 44115 | USA |
| THOMPSON JR CHARLES H | INFORMATION MGMT CHF | SPACE & MSL SYS DIV | 2805 MAPLE AVENUE | MANHATTAN BEA | CA | 90266 | USA |
| BRAUN GUENTHER K | DIRECTOR DATA PROC | SPRECKELS SUGAR DIV | 50 CALIFORNIA ST | SAN FRANCISCO | CA | 94111 | USA |
| PENNALA ERIC M | MANAGER PROG & OPS | SPRECKELS SUGAR DIV | 50 CALIFORNIA ST | SAN FRANCISCO | CA | 94111 | USA |
| HEINEN TERRY M | E D P MANAGER | ST CLOUD HOSPITAL | 1406 6TH AVENUE NO | ST CLOUD | MN | 56301 | USA |
| EDWARDS CONSTANCE E | MANAGER COMPUTER CTR | ST FRANCIS XAVIER UNIV | PO BOX 67 ST.F.X.U. | ANTIGONISH | NS | B2G1C0 | CANADA |
| RODRIGUEZ DAN R | PROJECT LDR SUPPLY | STANDARD OIL CO | 1090 GUILDHALL BLDG | CLEVELAND | OH | 44115 | USA |
| BIVENS FREDERICK(CANC) | DATA SPECIALIST | STANDARD OIL COMPANY | 101 W PROSPECT AVENUE | CLEVELAND | OH | 44115 | USA |
| JEFFRIES WILLIAM F | COORDINATOR ICS | STARK COUNTY DEPT ED | 7800 COLUMBUS RD | LOUISVILLE | OH | 44641 | USA |
| SMELSER LINDA C | ASST GENERAL MGR | STATE CONTROLLERS OFFICE | 504 E MUSSER | CARSON CITY | NV | 89701 | USA |
| STUMP DALE K | DATA PROCESSING MGR | STATE CONTROLLERS OFFICE | 504 E MUSSER | CARSON CITY | NV | 89701 | USA |
| ALEXY MICHAEL J | DATA SYSTEMS STAFF | STORER BROADCASTING | 1177 KANE CONCOURSE | BAY HARBOR IS | FL | 33154 | USA |
| HOLLING ERNEST(CANCEL) | DIRECTOR-DATA SYSTEMS | STORER BROADCASTING | 1177 KANE CONCOURSE | BAY HARBOR IS | FL | 33154 | USA |
| MORRISON JAMES C | MANAGER D P | SUN HYDRAULICS CORP | 1817 57TH STREET | SARASOTA | FL | 33580 | USA |
| BROOKS ROY | | SYNCRUDE CANADA LIMITED | 10030 107TH STREET | EDMONTON | AL | T5J3E5 | CANADA |
| BOWNES GORDON | SYSTEMS ANALYST | SYNCRUDE CANADA LTD | 10030 107TH ST 7TH ST PLAZA | EDMoNTON | AL | T5J3E5 | CANADA |
| BRICKER HARLEY G | SUPT MATERIALS | SYNCRUDE CANADA LTD | P O BAG 4009 | FT MC MURRAY | AL | T9H2L1 | CANADA |
| AUSTIN DONALD (CANCEL) | MGR SYSTEMS & OPERATIONS | SYRACUSE BOARD OF EDUC | 409 W GENESEE ST | SYRACUSE | NY | 13202 | USA |
| FOSTER RICHARD M | MANAGER SYSTEM S/W | SYSTEM DEVELOPMENT CORP | 2500 COLORADO AVENUE | SANTA MONICA | CA | 90406 | USA |
| BRYSON GARY J | CONSULTANT | SYSTEM HOUSE | 560 ROCHESTER | OTTAWA | ON | K1B3B8 | CANADA |
| CONNOR JACK | SR CONSULTANT | SYSTEMS & COMP TECH | 7 N 5 POINT RD | WEST CHESTER | PA | 19380 | USA |
| ROSENBERG IVAN M | GENERAL PARTNER | SYSTEMS DESIGN ASSOC | P O BOX 1144 | SAN LUIS OBIS | CA | 93406 | USA |
| DUMAS ROBERT J | MARKETING SUPPORT | SYSTEMS RESEARCH INC | 2400 SCIENCE PARKWAY | OKEMOS | MI | 48864 | USA |
| FRASER TOM | | SYSTEMS RESEARCH INC | 2400 SCIENCE PARKWAY | OKEMOS | MI | 48864 | USA |
| MEYERS LAWRENCE | DIRECTOR OF MARKETING | SYSTEMS RESEARCH INC | 2400 SCIENCE PARKWAY | OKEMOS | MI | 48864 | USA |
| WICKHAM GAIL D | MGR MARKETING SERVICES | SYSTEMS RESEARCH INC | 2400 SCIENCE PARKWAY | OKEMOS | MI | 48864 | USA |
| STEIN SALLYSUE | PARTNER | T.O.A.D. | 19855 GIESENDORFER RD | COLFAX | CA | 95713 | USA |
| STOVER DAVID W | DIR OF INFO SERVICES | TELEPHONE EMPLOYEES CO | 639 S NEW HAMPSHIRE | LOS ANGELES | CA | 90005 | USA |

| NAME | TITLE | COMPANY | ADDRESS | CITY | STATE | ZIP | COUNTRY |
|---|---|---|---|---|---|---|---|
| GEER ROSS E | DIRECOTR OF SYS ENGR | TEXAS MUN POWER AGENCY | 2225 E RANDOL MILL RD | ARLINGTON | TX | 76011 | USA |
| WRIGHT JAMES C | ELECTRICAL ENGINEER | TEXAS MUN POWER AGENCY | 2225 E RANDOL MILL RD | ARLINGTON | TX | 76011 | USA |
| MURPHY JR ROBERT | DIRECTOR ADMINISTRATION | TEXAS MUNICIPAL POWER | 600 ARLINGTON DOWNS | ARLINGTON | TX | 76011 | USA |
| MILLER WILLIAM J | MGR CORP SYS PLANNING | THE BOVAIRD SUPPLY CO | 823 S DETROIT | TULSA | OK | 74102 | USA |
| PRICE ROGER | SYSTEMS ANALYST | THE BOVAIRD SUPPLY CO | 823 S DETROIT | TULSA | OK | 74102 | USA |
| THOMSON RON | DATA PROCESSING MGR | THE BOVAIRD SUPPLY CO | 823 S DETROIT | TULSA | OK | 74102 | USA |
| ELLIOTT HARRY A | MANAGER MIS DEPARTMENT | THE CANADA STARCH CO | 1 PLACE DU COMMERCE | NUNS' ISLAND | QU | H3E1A7 | CANADA |
| TEARE ROBERT F | ASST DIR BIBL SERV | THE CLAREMONT COLLEGES | HONNOLD LIBRARY | DARTMOUTH AT | CA | 91711 | USA |
| AUSLANDER RICHARD C | DB DC TECH SPECIALIST | THE GAP STORES INC | 900 CHERRY AVE | SAN BRUNO | CA | 94066 | USA |
| BOSCO SUSAN M | PROJECT MANAGER | THE GAP STORES INC | 900 CHERRY AVE | SAN BRUNO | CA | 94066 | USA |
| KOMAR JAMES V | DIRECTOR DATA PROCESSING | THE H W WILSON CO. | 950 UNIVERSITY AVENUE | BRONX | NY | 10452 | USA |
| DAMM JACK A | PRINCIPAL/OWNER | THE PALO ALTO GROUP | 790 LUCERNE DRIVE | SUNNYVALE | CA | 94086 | USA |
| LANGE JOHN M | PROGRAMMER ANALYST | THE TORO COMPANY | 5825 JASMINE STREET | RIVERSIDE | CA | 92504 | USA |
| LIGHTHEART TED M | ANALYST/PROGRAMMER | THE WILLAMETTE VALLEY CO | 660 MC KINLGY ST | EUGENE | OR | 97402 | USA |
| FLOYD TERRY H | ACCNT SYSTEMS ANALYST | THERMON | 100 THERMON DR | SAN MARCOS | TX | 78666 | USA |
| NORTH CARL H | DIRECTOR COMPUTER CTR | THOMAS NELSON COM COL | PO BOX 9407 | HAMPTON | VA | 23670 | USA |
| TROWBRIDGE VERN H | MGR DATA PROCESSING | TIDEWATER COMM COLL | RT 135 | PORTSMOUTH | VA | 23701 | USA |
| NEWMAN ALAN T | INFO SERVICES MGR | TOWER MANAGEMENT | 1779 TRIBUTE ROAD #H | SACRAMENTO | CA | 95815 | USA |
| SILVER GAYE L | SYSTEMS ANALYST | TRW COLORADO ELECT | 3450 N NEVADA AVE | COLORADO SPRI | CO | 80907 | USA |
| JONES MORGAN |  | TYMDATA CORP | 44 W JEFFERSON ST | BROWNSVILLE | TX | 78520 | USA |
| MC GRATH JOSEPH |  | TYMDATA CORP | 44 W JEFFERSON ST | BROWNSVILLE | TX | 78520 | USA |
| VILLARREAL RAMON V |  | TYMWARE CORP | 44 W JEFFERSON ST | BROWNSVILLE | TX | 78520 | USA |
| KING STEPHEN E | PROGRAMMER/ANALYST | U S A F | LOS ANGELES AIR FORCE STA | EL SEGUNDO | CA |  | USA |
| CLEMENT RUBY J | DIRECTOR OF DP | UMPQUA COMM COLLEGE | P O BOX 967 | ROSEBURG | OR | 97470 | USA |
| WILKINSON JIM L | GENERAL PARTNER | UMPQUA DATA FACTORY | 222 E 11 | EUGENE | OR | 97401 | USA |
| JARVIS RAY D | PROGRAMMER | UMPQUA DATA FACTORY | 727 SE CASS | ROSEBURG | OR | 97470 | USA |
| CULPEPPER BRITTON B | SYSTEMS ANALYST | UNION CAMP CORP |  | FRANKLIN | VA | 23851 | USA |
| EDWARDS BENJAMIN E | ASST MGR DATA PROCESSING | UNION CAMP CORP |  | FRANKLIN | VA | 23851 | USA |
| MC CLURE HERSCHEL D | MGR SYSTEMS DEPT | UNION CAMP CORP |  | FRANKLIN | VA | 23851 | USA |
| GOOLSBY MONTY P | SYSTEMS MANAGER | UNION CAMP CORP | P O BOX 570 | SAVANNAH | GA | 31402 | USA |
| MOCK MADISON O | MGR INFO SYS BAG DIV | UNION CAMP CORP | P O BOX 1825 | SAVANNAH | GA | 31402 | USA |
| LUISI WILLIAM F | SOFTWARE ANALYST | UNION CAMP CORP | 1600 VALLEY RD | WAYNE | NJ | 07470 | USA |
| NICHOLS MARTIN D | TECHNICAL ANALYST | UNION CAMP CORP | 1600 VALLEY RD | WAYNE | NJ | 07470 | USA |
| HAYS GARRY R | SR DATA COMMUNICATION | UNION OIL CO | 461 S BOYLSTON ST | LOS ANGELES | CA | 90017 | USA |
| NAGEL PATRICK M | SYSTEMS PROGRAMMER | UNION OIL COMPANY | 135TH ST & NEW AVE | LEMONT | IL | 60439 | USA |
| NONAMAKER LAURA S | PROGRAMMER/ANALYST | UNION OIL COMPANY | 461 BOYLSTON RM M327 | LOS ANGELES | CA | 90017 | USA |
| FREDRICKSON STEVE | DISTRICT SALES MGR | UNITED COMPUTING SYS | 4544 POST OAK PL #146 | HOUSTON | TX | 77027 | USA |
| CLEVELAND WALTER |  | UNITED COMPUTING SYS | 2525 WASHINGTON | KANSAS CITY | MO | 64108 | USA |
| GRACE JAYNIA |  | UNITED COMPUTING SYS | 2525 WASHINGTON | KANSAS CITY | MO | 64108 | USA |
| ADDIS JOHN | MGR CONSULTING SVCS | UNITED COMPUTING SYS | 1901 AVE OF STARS #585 | LOS ANGELES | CA | 90067 | USA |
| GEWECKE JOHN W | DISTRICT SALES MGR | UNITED COMPUTING SYS | 1901 AVE OF STARS #585 | LOS ANGELES | CA | 90067 | USA |
| MEYERS BARRY | NATIONAL SALES MGR | UNITED COMPUTING SYS | 1901 AVE OF STARS #585 | LOS ANGELES | CA | 90067 | USA |
| NOLAN VINCENT |  | UNITED MC GILL CORP | 2400 FAIRWOOD AVE | COLUMBUS | OH | 43207 | USA |
| CHITWOOD RICK L | ASST V P INFO SYS | UNITED PRESIDENTIAL LIFE | 217 SOUTHWAY BLVD E | KOKOMO | IN | 46901 | USA |
| TOWNSEND RICHARD | PROGRAMMER/ANALYST | UNITED PRESIDENTIAL LIFE | 217 SOUTHWAY BLVD E | KOKOMO | IN | 46901 | USA |
| REGO F ALFREDO | PROFESSOR | UNIV F MARROQUIN | 6A AVE 0-28 ZONA 10 | GUATEMALA |  |  | GUATEM |
| SISOIS MIKES N | DIRECTOR INFO SYSTEMS | UNIV OF SANTA CLARA | BANNAN HALL 113 | SANTA CLARA | CA | 95053 | USA |
| TRUE JOHN F | DIR COMPUTING SERVICE | UNIV OF TENNESSEE | 615 MC CALLIE AVENUE | CHATTANOOGA | TN | 37402 | USA |
| LINDSTROM EDWARD R | DIRECTOR-AGRI DATA CT | UNIVERITY OF KENTUCKY | 5107 AG SCI CTR N | LEXINGTON | KY | 40506 | USA |
| ARANDA JORGE A | SYSTEMS ENGINEER | UNIVERSIDAD BAJA CAL | OBREGON Y F | MEXICALI BAJA |  |  | MEXICO |
| GALLARDO PEDRO A | SYSTEM ENGINEER | UNIVERSIDDO BAJA CAL | OBREGON Y F C CALCOLO | MEXICALI MAJA |  |  | MEXICO |
| ALVAREZ MANUEL A | COORDINATOR | UNIVERSITY IDAMETROPOLITAN | APDO POSTAL 55-503 | MEXICO | DF | 13 | MEXICO |

| NAME | TITLE | COMPANY | ADDRESS | CITY | STATE | ZIP | COUNTRY |
|---|---|---|---|---|---|---|---|
| SCHULER KURT J | DIR / COMP SERVICES | UNIVERSITY OF DALLAS | | IRVING | TX | 75061 | USA |
| CHRISTOPHERSON DIANE | | UNIVERSITY OF WISCONSIN | | RIVER FALLS | WI | 54022 | USA |
| NELSON MARLYS | | UNIVERSITY OF WISCONSIN | | RIVER FALLS | WI | 54022 | USA |
| PROCHNOW NEAL | | UNIVERSITY OF WISCONSIN | | RIVER FALLS | WI | 54022 | USA |
| PETERSON DON | SYSTEMS PROGRAMMER | US CIVIL SERV COMM | 4685 LOG CABIN DR | MACON | GA | 31210 | USA |
| WRIGHT NORMAN B | EXAMINING SYSTEMS COORD | US CIVIL SERVICE COMM | 4685 LOG CABIN DR | MACON | GA | 31206 | USA |
| SPAHN CARL P | DIRECTOR ADSS | US DEPT AGR-APHIS | 6525 BELCREST RD #853 | HYATTSVILLE | MD | 20782 | USA |
| TAYLOR PAUL M | COMPUTER SPECIALIST | US DEPT AGRICULTURE | 6525 BELCREST RD #853 | HYATTSVILLE | MD | 20782 | USA |
| SPERLE GLENN M | COMPUTER SPECIALIST | US DEPT OF AGR-APHIS | 6525 BELCREST RD #853 | HYATTSVILLE | MD | 20782 | USA |
| SELLERS HARRY P | ADM DP MANAGER | VA WESTERN COM COL | 3095 COLONIAL AVE | ROANOKE | VA | 24015 | USA |
| CARLSON LEE A | PROF MATH & COMP SCIENCE | VALPARAISO UNIVERSITY | ACADEMIC COMPUTER CENTER | VALPARAISO | IN | 46383 | USA |
| HODSON DICK | DP MANAGER | VALTEC | BOX 2200 | SPRINGVILLE | UT | 84663 | USA |
| SIMONSEN J LAWRENCE | COMPUTER SYSTEMS ENGR | VALTEK INC | MOUNTAIN SPRINGS PRKWY | SPRINGVILLE | UT | 84663 | USA |
| GILL RICK | SYSTEMS ANALYST | VANDERBILT UNIVERSITY | 521 KIRKLAND HALL | NASHVILLE | TN | 37235 | USA |
| DOLAN JR DOUGLAS C | SPECIAL PROJECTS MGR | VICTOR O SCHINNERER | 5028 WISCONSIN AVE NW | WASHINGTON | DC | 20016 | USA |
| BASSELLIER JEAN-PAUL | DIRECTEUR INFORMATIQUE | VILLE SAINT-LAURENT | 777 BOUL LAURENTIEN | SAINT-LAURENT | QU | H4M2M7 | CANADA |
| NEWELL RUSSELL L | SYSTEMS ANALYST | VOFM-DEARBORN | 4901 EVERGREEN | DEARBORN | MI | 48128 | USA |
| RADOFF IRVING | SUPVR SYS/PROG | VSI CORPORATION | 8463 HIGUERA ST | CULVER CITY | CA | 90230 | USA |
| HACKMAN LINFORD B | ADMIN SOFTWARE MGR | VYDEC INC | 9 VREELAND ROAD | FLORHAM PARK | NJ | 07932 | USA |
| HEDA SHARAD | MANAGER ADM SYS | VYDEC INC | 9 VREELAND ROAD | FLORHAM PARK | NJ | | USA |
| POLAKOWSKI KEN | GRP MGR SOFTWARE SUPPORT | VYDEC INC | 9 VREELAND RD | FLORHAM PARK | NJ | | USA |
| WALLACE DARL L | DIR EDUC COMP CTR | WALLA WALLA COLLEGE | | COLLEGE PLACE | WA | 99324 | USA |
| SHUMATE D CRAIG | MANAGER C S D | WARREN & VAN PRAAG | 1276 NORTH WARSON RD | ST LOUIS | MO | 63132 | USA |
| AGRUSTI RAYMOND J | | WAYNE BOARD OF ED | 50 NELLIS DRIVE | WAYNE | NJ | 07470 | USA |
| OMI G KEVIN | MANAGER | WELLS FARGO BANK | 420 MONTGOMERY ST | SAN FRANCISCO | CA | 94111 | USA |
| RUSSELL BRIAN G | PROGRAMMER/ANALYST | WEST FRASER MILLS | BOX 6000 | QUESNEL | BC | V2J3J5 | CANADA |
| TSUKISHIMA LLOYD M | PROGRAMMER/ANALYST | WEST FRASER MILLS | P O BOX 6000 | QUESNEL | BC | V2J3J5 | CANADA |
| KILMON JR LIN E | SENIOR ENGINEER | WESTERN ELECTRIC CO | 2500 BROENING HWY | BALTIMORE | MD | 21224 | USA |
| NEUMYER RICHARD D | SENIOR ENGINEER | WESTERN ELECTRIC CO | 2500 BROENING HWY | BALTIMORE | MD | 21224 | USA |
| ECKARD DAVID L | SYSTEM ANALYST | WESTINGHOUSE ELEC C | P O BOX 9175 | PHILADELPHIA | PA | 19113 | USA |
| BROOKE ROGER G | MGR SUPPORT SERVICE | WESTINGHOUSE ELEC CORP | P O BOX 8839 | PITTSBURGH | PA | 15221 | USA |
| MC GEOY JOHN G | MGR DATA SERVICES | WESTINGHOUSE ELEC CORP | P O BOX 8839 | PITTSBURGH | PA | 15221 | USA |
| ANDERSON EDWARD W | SR SYSTEMS SPECIALIST | WEYERHAEUSER CO | AFB 4 | TACOMA | WA | 98402 | USA |
| GOJENOLA BEN | SR SOFTWARE DESIGNER | WEYERHAEUSER CO | 10TH & A STREETS RMB-2 | TACOMA | WA | 98401 | USA |
| ROBERTSON DENNIS L | MGR DIST PROC TECH SP | WEYERHAEUSER CO | 10TH & A STREETS RMB-1 | TACOMA | WA | 98401 | USA |
| HELLAMS CAROLE | D P PROG ANALYST | WHARTON CO JR COLLEGE | 911 BOLING HIGHWAY | WHARTON | TX | 77488 | USA |
| PERKINS ALAN L | SYSTEMS MGR/PGMR | WHITE HOUSE COMM | THE WHITE HOUSE | WASHINGTON | DC | 22070 | USA |
| HOLT WAYNE E | DIRECTOR OR D P | WHITMAN COLLEGE | 345 BOYER AVE | WALLA WALLA | WA | 99362 | USA |
| HAINES OLIN R | DATA PROCESSING MANAGER | WICHITA EAGLE & BEACON | 825 E DOUGLAS | WICHITA | KS | 67202 | USA |
| JENSEN HAROLD E | SYSTEMS ANALYST | WILLAMETTER VALLEY CO | 660 MC KINLEY | EUGENE | OR | 97402 | USA |
| TURNER WILLIAM A | PROF ASSOCIATE | WILLIA M MERCER INC | 409 GRISWOLD | DETROIT | MI | 48226 | USA |
| STRANDHAGEN DEBRA A | SYSTEM CONTROL | WILLIAM M MERCER INC | 409 GRISWOLD | DETROIT | MI | 48226 | USA |
| ENGLANDER WILLIAM R | D P CONSULTANT | WILLIAM R ENGLANDER | 1966 TITUS STREET | SAN DIEGO | CA | 92110 | USA |
| FARRELL JIM G | DIRECTOR MARKET RESEARCH | WM C BROWN PUB CO | 2460 KERPER BLVD | DUBUQUE | IA | 52001 | USA |
| HAMPTON JEAN K | PROGRAMMER | WM C BROWN PUB COMPANY | 2460 KERPER BLVD | DUBUQUE | IA | 52001 | USA |
| HEIN NORM | MANAGER INFO SERVICES | WM C BROWN PUB COMPANY | 2460 KERPER BLVD | DUBUQUE | IA | 52001 | USA |
| POWERS DENNIS D | DIRECTOR INFO SERVICE | WM C BROWN PUB COMPANY | 2460 KERPER BLVD | DUBUQUE | IA | 52001 | USA |
| SCHICK JOHN A | PROGRAMMER | WM C BROWN PUB COMPANY | 2460 KERPER BLVD | DUBUQUE | IA | 52001 | USA |
| SNESRUD MARGARET J | PROJECT DIRECTOR | WM C BROWN PUB COMPANY | 2460 KERPER BLVD | DUBUQUE | IA | 52001 | USA |
| SNESRUD WALLY M | PROGRAMMER | WM C BROWN PUB COMPANY | 2460 KERPER BLVD | DUBUQUE | IA | 52001 | USA |
| WOOD WALTER A | VICE PRESIDENT | WOOD BROWN & ASSOCIATES | 1673 CARLING SUITE 105 | OTTAWA | ON | K2A1C4 | CANADA |
| BROWN EDWIN J | PRESIDENT | WOOD,BROWN & ASSOCIATES | 1673 CARLING AVE | OTTAWA | ON | K2A1C4 | CANADA |

| NAME | TITLE | COMPANY | ADDRESS | CITY | STATE | ZIP | COUNTRY |
|---|---|---|---|---|---|---|---|
| BADGER PATRICK S | DIRECTOR ACADEMIC COM | XAVIER UNIVERSITY | | NEW ORLEANS | LA | 70125 | USA |
| GREGORY KENT A | ACTUARY/PROGRAMMER | ZISCHKE ORGANIZATION | 1 POST STREET | SAN FRANCISCO | CA | 94104 | USA |

# CONFERENCE VENDOR LISTING INDEXED BY COMPANY

| NAME | TITLE | COMPANY | ADDRESS | CITY | STATE | ZIP | COUNTRY |
|---|---|---|---|---|---|---|---|
| BEVERLY SHEPHERD | OPERATIONS SUPERVISOR | ACADEMIC COMP CENTER | UNIVERSITY OF WISCONSIN | RIVER FALLS | WI | 54022 | USA |
| ROSS SCROGGS | | ALTER*ABILITY | 534 ROSAL AVE | OAKLAND | CA | 94610 | USA |
| JOHN M GRILLOS | VICE PRESIDENT | AMERICAN MANAGEMENT SYSTE | 561 PILGRIM DRIVE, SUITE D | SAN MATEO | CA | 94404 | USA |
| J R DAUGHERTY | DIR DATA PROCESSING | AMERICAN SUBSCRIPTION TV | 8383 WILSHIRE BLVD-SUITE900 | BEVERLY HILLS | CA | 90211 | USA |
| BUD BOOTH | PRESIDENT | AUTOMATED ANALYSIS | 3105 DONA SOFIA DR | STUDIO CITY | CA | 91604 | USA |
| MARK ROBBINS | SALES REPRESENTATIVE | BARNHILL THREE INC | 1980 S QUEBEC - UNIT 4 | DENVER | CO | 80231 | USA |
| DONNA CALDWELL | BUSINESS MANAGER | BOEING COMPUTER SERVICES | 871 SO NASH ST - SUITE 101 | EL SEGUNDO | CA | 90245 | USA |
| D R DAVIDSON | MARKETING PROMOTION MGR | CALCOMP | 1270 N KRAEMER | ANAHEIM | CA | 92806 | USA |
| ANDREW JAMES | MARKETING MANAGER | CARROLL MFG CO | 1212 HAGAN | CHAMPAIGN | IL | 61820 | USA |
| PETER MELVIN | MARKETING DIRECTOR | CMC ASSOCIATES INC | 755 BOYLSTON ST | BOSTON | MA | 02021 | USA |
| CHARLES W JACKSON | PRESIDENT | COLLIER-JACKSON & ASSOC | 1805 N WESTSHORE BLVD - SUI | TAMPA | FL | 33607 | USA |
| ROGER GOLDMAN | SYSTEMS ANALYST | COMARCO INC | 227 W HUENEME RD | OXNARD | CA | 93030 | USA |
| ED ST AMOUR | NATIONAL SALES MANAGER | COMPUTER PERIPHERAL SVS | 3187 "F" AIRWAY AVE | COSTA MESA | CA | 92626 | USA |
| CHARLES YARBROUGH | VICE PRESIDENT | COMPUTERS FOR MARKETING | 215 MARKET ST - SUITE 1212 | SAN FRANCISCO | CA | 94105 | USA |
| DAVID C DUMMER | PRESIDENT | D C DUMMER & ASSOCIATES | 40 LAKE LUCERNE CLOSE SE | CALGARY | AL | T2J3H8 | CANADA |
| KEN LESSEY | ASSOCIATE | DATACON OF ST HELENS | 50 WEST STREET | ST HELENS | OR | 97051 | USA |
| DOUGLAS L MURRAY | SALES ENGINEER | GANDALF DATA INC | 8821 RUTGERS ST | WESTMINSTER | CO | 80030 | USA |
| RELLA HINES | EXECUTIVE DIRECTOR | HP GENERAL SYSTEMS USERS | P.O. BOX 18313, BWI BR. | BALTIMORE | MD | 21240 | USA |
| RICK GRIFFIN | PRESIDENT | I C S SERVICES INC | 2131 W EULESS BLVD | EULESS | TX | | USA |
| JOSEPH RAUH | TECHNICAL SHOWMAN | INTEGRATED SOFTWARE SYSTE | 4186 SORRENTO VALLEY BLVD | SAN DIEGO | CA | 92121 | USA |
| CHRISTOPHER DAVY | MANAGER GOVERNMENT SERV | KEYDATA CORPORATION | 1400 WILSON BLVD-SUITE 100 | ARLINGTON | VA | 22209 | USA |
| MARTIN GORFINKEL | PARTNER | LOS ALTOS RESEARCH CENTER | 339 SOUTH SAN ANTONIO ROAD | LOS ALTOS | CA | 94022 | USA |
| GARY D ANDERSON | ASSOC PROF OF BIOSTAT | MC MASTER UNIVERSITY | 1200 MAIN ST WEST | HAMILTON | ON | L8S4J9 | CANADA |
| MANAGER | MINI PERIPHERAL PRGRMS | MEMOREX CORP, BUS SYS DIV | 3015 DAIMLER STREET | SANTA ANA | CA | 92705 | USA |
| DAVID W PIDWELL | VICE PRESIDENT, SALES | MINICOMPUTER ACCESSORIES | 130 S WOLFE RD PO BOX 9004 | SUNNYVALE | CA | 94086 | USA |
| SUSAN FEINBERG | VICE PRESIDENT | NICHOLS AND COMPANY | 1900 AVENUE OF THE STARS ST | LOS ANGELES | CA | 90067 | USA |
| CHARLES J VILLA JR | PRESIDENT | PANTECHNIC | 5805 OCEAN VIEW DR | OAKLAND | CA | 94618 | USA |
| LARRY GRESS | PRESIDENT | PROGRESSIVE COMMUNICATION | 310 / ALAMO 128 S TEJON | COLORADO SPRI | CO | | USA |
| WILLIAM E MOORE | MARKETING MANAGER | R SHRIVER ASSOCIATES | 1530 CHESTNUT ST-SUITE 714 | PHILADELHIA | PA | 19102 | USA |
| ROBERT M GREEN | PRESIDENT | ROBELLE CONSULTING LTD | #130-5421 10TH AVE | DELTA | BC | V4M3T9 | CANADA |
| DONALD S KLETT | DIR UNIV LAB | SANGAMON STATE UNIVERSITY | SHEPHERD ROAD | SPRINGFIELD | IL | 62708 | USA |
| STEVE JAMISON | MANAGER/MARKETING SUPP | SATELLITE COMPUTING INC | 4530 PROFESSIONAL CIRCLE | VA BEACH | VA | 23455 | USA |
| LAWRENCE MEYERS JR | | SYSTEMS RESEARCH INC | 241 E SAGINAW ST | E LANSING | MI | 48823 | USA |
| PHILLIP G BEGICH | APPLICATIONS ENGINEER | TELEFILE COMPUTER PRODUCTS | 17131 DAIMLER ST | IRVINE | CA | 92714 | USA |
| MICHAEL B MARFES | ACCOUNT REPRESENTATIVE | TEXAS INSTRUMENTS | 9725 E HAMPDEN AVE | DENVER | CO | 80231 | USA |
| JOHN A DAMM JR | PRINCIPAL | THE PALO ALTO GROUP | 790 LUCERNE DR | SUNNYVALE | CA | 94086 | USA |
| JACK KENNEY | DISTRICT MANAGER | UARCO | 1805 SOUTH BELLAIRE ST - | DENVER | CO | 80222 | USA |
| JAYNIA LYNN GRACE | MARKETING MANAGER | UNITED COMPUTING SYSTEMS | 2525 WASHINGTON | KANSAS CITY | MO | 64108 | USA |
| R R MURRAY | MANAGER, MARKETING COMM | VERSATEC INC | 2805 BOWERS AVE | SANTA CLARA | CA | 95051 | USA |
| ROBERT A VASCONCELLOS | OWNER | WHERE ENDS MEET | 5926 WHITNEY ST | OAKLAND | CA | 94609 | USA |