COMMERICAL LANGUAGES SIG

RPG Sprechen sie DIBOL Parlez-vous BASIC Habla COBOL

**JANUARY 1985**

# COMMERCIAL LANGUAGES SIG

**Replaces**

**BASIC SIG**


COMMERICAL
LANGUAGES
SIG
RPG Sprechen sie DIBOL Parlez-vous BASIC Habla COBOL

COMMERCIAL LANGUAGES SIG FORMED

by Ted Bear and Mary A. Feerick

DECEMBER 12, 1984--Anaheim DECUS-- The formation of the Commercial Languages SIG (CLSIG) was approved today by the DECUS SIG Council. The new SIG combines the former BASIC SIG, COBOL/RPG SIG and the language portion of the DIBOL SIG.

COMMERCIAL LANGUAGES SIG - The members of this SIG recognize the use of BASIC, COBOL, DIBOL, or RPG as an integral part of their company's Information Processing environment. They share their technical knowledge and expertise with each other and with the developers at DIGITAL during sessions at the semi-annual symposia and thru the SIG newsletter. The SIG provides one forum for all of these high level languages as well as their use with other programming tools such as screen handlers, data base managers, text editors, etc. Our interest covers all DIGITAL computers and operating systems in use in business, scientific, and educational environments.

The COMMERCIAL LANGUAGES SIG has a special interest in programmer productivity issues. The Commercial Applications Productivity (CAP) working committee works with the DIGITAL developers on future, related products while still in development. This committee works to identify tomorrow's wishlist items before the product is marketed. Another important function of this SIG is to promote changes through the ANSI Standards Committee in order to make BASIC, COBOL and DIBOL into even better languages.

This SIG allows DECUS to further accomplish its objectives by eliminating duplicate effort, expenses and overhead; increase influence in ANSI, DEC and DECUS; better representation of the user; better alignment with the DEC Languages group. The new SIG will also offer a more unified voice, influence DECUS to get a seat on ANSI X-3.

CLSIG should draw on a greater base for volunteers, gather unified productivity "tools" (such as code generation) and influence development of "4th generation" languages. The new SIG will give better service to the user with a combined newsletter, less confusion at symposia and a unified theme at symposia. We expect to pick up those users who were interested in commericial languages, but didn't know which SIG to join as well as the COBOL/RPG BASIC and DIBOL SIG members.

Commercial Languages SIG Steering Committee members are:

Chair
Ted A. Bear
NCA Corporation
3250 Jay Street
Santa Clara, CA 95054
(408) 986-1800 X 3416

Operation and Communication Coordinator
Jim Wilson
Pfizer Quality Control Division
Vigo Plant
Terre Haute, IN 47808
(812) 299-2121

Coordinator
Phil Hunt
Systems Industries
1855 Barber Lane
P.O. Box 789
Milpitas, CA 95035
(408) 942-1212

Marketing Coordinator
Mary Anne Feerick
Greyhound Temporary Personnel
1050 Wall Street West
Lyndhurst, NJ 07071
(201) 438-8000 x 33

Symposia Coordinator
Ray Strackbein
6000 State College Parkway
San Bernandinoi, CA 92407
(714) 887-0916

Project Coordinator
Ken Shay
Melvin Simon & Associates
P.O. Box 7033 Merchants Plaza
Indianapolis, IN 46207
(317) 636-1600 X 6035

Seminar Coordinator
Mark Miller
The Pennsylvinia State University
J. Orvis Keller Bldg.
University Park, PA 16802
(814) 865-1379

BASIC Language Coordinator
Bill Tabor
Computer Products
2900 Gateway Road
Pompano Beach, FL 33069
(305) 974-5500 X 269

COBOL Language Coordinator
Bob Markwith
4660 Charlton Ct.
Woodbridge, VA 22193
(703) 690-4658

DIBOL Language Coordinator
Becky Burkes
Finanical Insurance Consultant
P.O. Box 77
Covington, LA 70434
(504) 892-7428

RPG Language Coordinator
Keith Batzel
Crowe Chizek & Company
330 E. Jefferson Blvd. P.O. Box 7
South Bend, IN 46624
(219) 232-3992

Member at large
Rocky Hayden
UserWare International
2235 Meyers Avenue
Escondido, CA 92025
(619) 745-6006

Member at large
Bruce Gaarder
Macalester College
1600 Grand Avenue
Paul, MN 55105
(612) 696-6170

Member at large
Dan Esbensen
Touch Technologies
609 South Escondido Blvd. Suite 101
Escondido, CA 92025
(619) 743-0494

ARTICLE I - NAME

1.0. The name of the organization is the Commericisl Languages Special Interest Group. Herein referred to as "The SIG".

ARTICLE II - PURPOSE AND AFFILIATION

2.0. The SIG is established as a Special User Group under the bylaws of the DECUS/U.S. Chapter.

2.1 The SIG is established, as empowered by the DECUS/U.S. Chapter Bylaws, to serve members having a common interest to promote the interchange of information concerning the utilization of software and computers manufactured and marketed by Digital Equipment Corporation (DIGITAL).

2.2 The members of this SIG recognize the use of BASIC, COBOL, DIBOL, or RPG as an integral part of their company's Information Processing environment. They share their technical knowledge and expertise with each other and with the developers DIGITAL during sessions at the semi-annual symposia and thru the SIG newsletter. The SIG provides one forum for all of these high level languages as well as their interface with other programming tools such as screen handlers, data base managers, language specific text editors, etc. Our interest covers all DIGITAL computers and operating systems in use in business, scientific, and educational environments.

2.3 The SIG has a special interest in programmer productivity issues. The Commercial Applications Productivity (CAP) working group works with the DIGITAL developers on future, related products while still in development. This working group works to identify tommorrow's wishlist items before the product is marketed. Another important function of this SIG is to promote changes through the ANSI Standards Committee in order to make BASIC, COBOL, DIBOL, RPG and other languages into even better languages.

2.4 Another important function of this SIG is to promote language standards via ANSI Committee in order to make BASIC, COBOL, DIBOL, RPG and other languages of interest to SIG membership even better.

ARTICLE III - MEMBERSHIP

3.0. SIG membership shall be open to any DECUS/U.S. Chapter member using or interested in a commericial language including, but not limited to: BASIC, COBOL, DIBOL and RPG.

3.1 Rights of Members:

3.1.1. Members shall be elisible to participate in SIG activities and be members of the Steering Committee.

3.1.2. Five or more members of the SIG may, by written petition, bring a motion before a meeting of the SIG Steering Committee.

3.2. The Steering Committee shall have the right to revoke SIG membership for cause.

ARTICLE IV - SIG STRUCTURE

4.0. The overall structure of the SIG shall be:

**4.0.1. Steering Committee**

The Steering Committee is responsible for governing the activities and direction of the SIG.

4.0.2. Chairman

The Chairman shall be chief executive and operational officer of the SIG.

4.0.3. Working Groups

Working Groups shall consist of SIG members who shall be responsible for projects and activities assigned to them by the Steering Committee. These groups fulfill the need for 'sub-SIGs'.

4.0.4. Ad Hoc Committees

The Chairman may from time to time establish such ad hoc committees as the business of the SIG requires. These committees shall serve for a maximum of two years or until a the new SIG Chairman is in office.

ARTICLE V - STEERING COMMITTEE

5.0. The Steering Committee shall function as the governing body of the SIG.

5.1. Responsibilities

5.1.1. The Steering Committee shall establish procedures and policies for orderly operation and development of the group.

5.1.2. The Steering Committee shall appoint working groups and committees on an ad-hoc or permanent basis, as required.

5.1.3. The Steering Committee shall appoint Working Group Coordinators to oversee continuing projects and participate in DIGITAL product activities.

5.1.4. The Steering Committee shall represent SIG members with respect to DECUS and DIGITAL.

5.1.5. The Steering Committee shall effect liaison with the DECUS Functional Units and such other DECUS committees as may from time to time be established by the DECUS/US Chapter board.

5.1.6. The Steering Committee shall effect liaison with any other DECUS groups with respect to its areas of special interest for the purpose of service to its members.

5.2. Composition and Voting

5.2.1. The four officers (section VI) of the SIG and one Digital Representative shall comprise the SIG Executive Committee.

5.2.2. At least one of the DIGITAL representatives, appointed by Digital Equipment Corporation, shall be a member of the Steering Committee. This representative shall be known as the "DIGITAL Counterpart". This representative shall represent the view of DIGITAL on issues of concern to the SIG. Other DIGITAL representatives for each commericial language are encoursed and shall be accepted as members of the Steering Committee.

5.2.3. The Steering Committee, at its option, may invite any SIG member to attend meetings as a non-voting participant.

5.2.4. The DECUS/U.S. Chapter Executive Secretary or authorized representative shall be a non-voting member.

5.2.5. The SIG Chairman shall be the chairman of the Steering and Executive Committees.

5.2.6. The Chairman may act independently on all matters, and shall inform and consult with the Steering Committee as (s)he sees fit. A majority vote of the remaining members shall be required to override decisions of the chairman.

5.3. Nominations and Committee Membership

5.3.1. Any SIG member in good standing may petition to join the Steering Committee.

5.3.2. Acceptance of an applicant shall be deemed complete upon vote of the Executive Committee.

5.3.3. The Executive Committee may vote to maintain an individual member's standing within the Steering Committee.

5.3.4. Normal term for Steering Committee membership shall be two years, renewable at the Chairman's option.

5.4. Vacancy in Office

In the event of a vacancy on the Steering Committee caused by resignation or incapacity, the Chairman shall appoint a replacement from amongst SIG members.

5.5. Vote of the Steering Committee

Except as expressly required by these Procedures or by the DECUS/U.S. Chapter Bylaws, motions shall be carried by a majority vote of the members of the Committee.

5.6. Meetings of the Steering Committee

The Steering Committee shall meet at least once at each symposium. Sufficient notice shall be given to all Steering Committee members of any Steering Committee meetings held other than at a symposium.

ARTICLE VI - SIG OFFICERS

6.0. The Chairman is the chief executive and operational officer of the SIG. The other officers shall be the Symposium Coordinator, Operations and Communications Coordinator and Library Coordinator. The Executive Committee shall elect one of its number to serve as a Chairman Pro-tem in the absence of the Chairman. Should the Chairman vacate the office by resignation, disability, or inelisibility or impeachment, a new Chairman shall be elected by a majority vote of the remaining members of the steering committee. Should any other officer vacate any office by resignation, disability, or inelisibility or impeachment, the Chairman shall appoint a replacement. Any member of the Steering Committee can be impeached by a majority vote of the remaining member of the Steering Committee.
Should all Steering Committee positions become simultaneously vacant, the SIG should be considered no longer viable and disbanded.

6.1. Chairman

6.1.1 The Chairman shall serve for a two-year, renewable term. The Chairman shall be elected by a vote of the Steering Committee. The Chairman is the chief executive officer of the SIG, and shall chair all steering committee meetings and be responsible for directing all activities of the SIG between meetings of the steering committee. The Chairman is subject to the review of the Steering Committee, or recall by a majority vote of the members. The Chairman will propose, and recommend for adoption, to the Steering Committee, a resolution to dissolve the SIG when interest in the SIG's activities becomes too small to justify the SIG's existance.

6.1.2 The responsibilities of the Chairman shall be:

6.1.2.1. To perform normal administrative functions.

6.1.2.2. To insure SIG interface with DIGITAL and DECUS.

6.1.2.3. To adopt interim procedures and policies when necessary on behalf of the Steering Committee.

6.1.2.4. To execute various functions as assigned by vote of the Steering Committee.

6.1.2.5. To participate in DECUS/U.S. Chapter fiscal planning and preparation of SIG Business Plans.

6.1.2.6. To maintain a stable steering committee by finding replacements, adding or deleting positions, as needed, other than required positions.

6.1.2.7. To encourage and direct the growth and development of SIG member participation in SIG activities, and to improve the overall benefits of DECUS to SIG members.

6.1.2.8. To Represent the SIG on the DECUS/U.S. Chapter SIG Council.

6.1.2.9. To execute various functions so assigned by vote of the Steering Committee.

6.2. Symposium Coordinator

6.2.1. The Symposium Coordinator shall be appointed for a two year, renewable term.

6.2.2. The responsibilities of the Symposium Coordinator shall be:

6.2.2.1. To serve on and represent the interests of the SIG to the DECUS/U.S. Chapter Symposium Committee.

6.2.2.2. To solicit Symposium input from SIG members.

6.2.2.3. To organize SIG-related Symposium submissions and prepare a Symposium schedule.

6.2.2.4. To execute various functions so assigned by vote of the Steering Committee.

6.3. Operations and Comminucations Coordinator

6.3.1 The O & C Coordinator shall be appointed for a two year, renewable term.

6.3.2. The responsibilities of the O & C Coordinator shall be:

6.3.2.1. To arrange for publication of a SIG newsletter on a timely basis.

6.3.2.2. To arrange for publication of the SIG session notes.

6.3.2.3. To arrange for publication of the SIG proceedings.

6.3.2.4. To arrange for publication of the SIG wishlist.

6.3.2.5. To represent the SIG on the DECUS/U.S. Chapter Operations & Communications Committee.

6.3.2.6. To arrange for SIG representation to the DECUS/U.S. Chapter Professional Relations Committee.

6.3.2.7. To execute various functions so assigned by vote of the Steering Committee.

6.4. Library Coordinator

6.4.1. The Library Coordinator shall be appointed for a two-year, renewable term.

6.4.2. The responsibilities of the Library Coordinator shall be:

6.4.2.1. To promote the submission of programs to the DECUS library.

6.4.2.2. To promote the exchange of programs, documents, and other such tangible materials among members.

6.4.2.3. To organize a library working group with the approval of the SIG Chairman.

6.4.2.4. To be the primary interface with and representative on the DECUS U.S. Chapter Library Committee.

6.4.2.5 To generate and start distribution of the SIG Symposia tape generated during Spring and Fall DECUS Symposia.

6.4.2.6. To execute various functions so assigned by vote of the Steering Committee.

6.5. Marketing Coordinator

6.5.1. The Marketing Coordinator shall be appointed for a two-year, renewable term.

6.5.2. The responsibilities of the Marketing Coordinator shall be:

6.5.2.1 Submission of DECUScope Articles.

6.5.2.2 Maintenance of New user kit information.

6.5.2.3 Organization and execution of New user orientation.

6.5.2.4 Recruiting and public relations.

6.5.2.5 Promotions.

6.5.2.6 Campgrounds.

6.5.2.7 To execute various functions so assigned by vote of the Steering Committee.

6.6. Project Coordinator

6.6.1. The Project Coordinator shall be appointed for a two-year, renewable term.

6.6.2. The responsibilities of the project Corrdinator shall be:

6.6.2.1 Standards.

6.6.2.2 CAP/development.

6.6.2.3 To execute various functions so assigned by vote of the Steering Committee.

6.7 Seminar Coordinator

6.7.1. The Seminar Coordinator shall be appointed for a two-year, renewable term.

6.7.2. The responsibilities of the Seminar Coordinator shall be:

6.7.2.1 Pre-symposia seminars

6.7.2.2 Other seminars

6.7.2.3 To execute various functions so assigned by vote of the Steering Committee.

6.8 BASIC language Coordinator

6.8.1. The BASIC language Coordinator shall be appointed for a two-year, renewable term.

6.8.2. The responsibilities of the BASIC language Coordinator shall be:

6.8.2.1 Items specific to the BASIC language.

6.8.2.2 To execute various functions so assigned by vote of the Steering Committee.

6.9 COBOL language Coordinator

6.9.1. The COBOL language Coordinator shall be appointed for a two-year, renewable term.

6.9.2. The responsibilities of the COBOL language Coordinator shall be:

6.9.2.1 Items specific to the COBOL language.

6.9.2.2 To execute various functions so assigned by vote of the Steering Committee.

6.10 DIBOL language Coordinator

6.10.1. The DIBOL language Coordinator shall be appointed for a two-year, renewable term.

6.10.2. The responsibilities of the DIBOL language Coordinator shall be:

6.10.2.1 Items specific to the DIBOL language

6.10.2.2 To execute various functions so assigned by vote of the Steering Committee.

6.11 RPG language Coordinator

6.11.1. The RPG language Coordinator shall be appointed for a two-year, renewable term.

6.11.2. The responsibilities of the RPG language Coordinator shall be:

6.11.2.1 Items specific to the RPG language

6.11.2.2 To execute various functions so assigned by vote of the Steering Committee.

6.12 Member at large

6.12.1. The members are appointed by the chairman and are intended to assist the chairman in discharging his or her duties. They will serve until discharged by the current chairman.

6.12.2. The responsibilities of a Member at large shall be:

6.12.2.1 To interface with LUGs.

6.12.2.2 To execute various functions so assigned by vote of the Steering Committee.

ARTICLE VII - MEETINGS

7.0 General meetings

7.0.1. Business meetings shall be scheduled at the Spring and Fall DECUS U.S. Chapter Symposia.

7.1 Steering Committee meetings

7.1.1. The Steering Committee shall communicate by DCS or phone prior to each general meeting, or at the Chairman's request, and shall also meet at each DECUS U.S. Chapter symposium.

## ARTICLE VIII - AMENDMENTS

### 8.0 General

8.0.1. Amendments to these Operating Procedures shall not conflict with any provisions of the DECUS Bylaws or the DECUS/U.S. Chapter Bylaws.

8.0.2. A motion to amend these Operating Procedures may be initiated by any member of the Steering Committee or by written petition of 15 SIG members.

8.0.3. Should any dispute arise from the interpretation of these operating procedures, the Chairman of the SIG shall be considered the final authority for any such interpretation.

### 8.1. Vote

8.1.1. Amendments to these operating procedures shall be made by a majority vote of the Steering Committee.

8.1.2. Amendments introduced by the Steering Commitee during an official meeting may be voted on and be immediately effective.

8.1.3. Amendments introduced by general SIG membership must be conveyed in writing to the Steering Committee. An official notice of meeting shall be sent to a representative of the petitioning group. An amendment will be proposed at one meeting, and voted on in a future meeting with the proposed amendment published in the Newsletter in the interim.


# ANSI STANDARDS

By

Dan Esbensen

Touch Technologies

609 South Encondido Blvd

Suite 101

Escondido CA 92025

(619) 743-0494

The ANSI X3J2 standardization of BASIC committee meetins took place in San Diego California. The meetins started Tuesday, October 23rd and ended Friday October 26th.

This was an interim meetins. The bis push risht now is to set out the ANSI LEVEL I document as a national standard. To finish off the standard all public comments must be addressed. This meetins took place in the middle of the second public comment period. Without all comments yet in, we focused the meetins on ANSI LEVEL II and beyond.

Committees were set up to explore the followins expansion areas:

- Screen manasement
- Data types and structures
- Prosrammins environment

Because of the kind of work I do at Touch Technologies, Inc., I choose to be on the Prosrammins Environment committee. Where the current standard is concerned with "Portins prosrams", the environment issues deal with "Portins prosrammers".

Each committee was asked to define a list of committee goals. The goals that were chosen were as follows:

Programming Environment
---
Build off of current chapter on editing. Underlaying functionality to include source code modification, debug facilities, and program execution.

Screen Management
---
Treat the form system as a type of I/O driver. Try to take what we can from the CODASYL approach.

Data types and Structures
---
Pattern after VAX BASIC

A number of hours were spent in committee discussing the goals and issues. The discussions will be continued at the next ANSI MEETING. If anyone would like to contribute suggestions on extensions, please call or write.

The next topic was the packaging and processing of ANSI Standards. We discussed the steps necessary to get ANSI BASIC LEVEL I to be a national standard. More important, we came up with a set of guidelines about what BASIC enhancements should look like. What makes BASIC BASIC?

1. New facilities should require only a minimum understanding in order to be used.

2. New facilities should promote rapid problem solving.

3. Facilities should be defined to allow for maximum 'extensability'.

4. Extensions should be conceptually independant. They should work without requiring other extensions.

5. Enhancements should avoid introducing additional key words.

6. When facilities are enhanced, the enhancements must be upward compatable.

After the last meeting, the entire graphics section of the manual was replaced with a GKS (Graphics Kernal Standard) compatable scheme. At this meeting, after studying the GKS package, it was concluded that maybe we were a bit too quick in adopting the GKS scheme. The graphics section will be discussed further at the next meeting.

Two presentations were made on program modules (ala PASCAL). Modules are named sections of routines. By including a given module, a program can access all routines specified in that module.

The next ANSI BASIC X3J2 meeting will be held in Miami, Florida. At this meeting we will discuss any objections to the standard. We will then draft and approve responses to objections. If there aren't any objections that can be overcome, we will take the next step in finalizing this ANSI BASIC level I standard.



Till Miami,

## A Generalized BASIC Error Processor

by

Donald L. Hallberg

State of California
The Resources Agency
Department of Fish and Game
3124 Nimbus Road, Suite E
Rancho Cordova, CA 95670


California Department of Fish and Game began using Basic-Plus-2 in 1978. At that time BP2 was in its infancy, version 1.1. We quickly learned that reading a file to EOF required a user written error-handling routine. The process involved a rather complex sequence of events: 1) error trapping must be enabled and BP2 told where the error-handler is located, 2) the error handler must be smart enough to figure out what the error was and take the appropriate action, 3) ther the routine must know where to go in order for normal processing to continue.

Digital provided a few handy variables (ERR, ERL and ERN$) that are set when BP2 detects an error and unconditionally branches. The error-handler typically will make use of these variables to determine where to resume program execution.

We were bothered by the necessary use of GOTO-like constructs which did not conform to our structured coding conventions. We were further confounded by the fact that the error-handler had to figure out where to resume processing. We began looking for a way to make error trapping simple and straightforward. We discovered that using the RESUME statement without a trailing line number allowed us to write one generalized error-handler routine that could be used without modification in every program. The routine was actually a status processing routine that did nothing more than assign the value of ERR to a status variable before resuming program execution by re-executing the statement that caused the error.

The status variable is used to allow conditional branching and error processing at the place the error occured rather than in the error-handling routine. The following example illustrates the basic technique involved.

```
1     ON ERROR GOTO 32720 ! Status Processor
...
1230 IF   STATUS.% = 0%
     THEN GET  1%
     ELSE IF   STATUS.% = 11%
          THEN EOF% = -1%
               STATUS.% = 0%
          ELSE UNEXPECTED.ERROR% = STATUS.%
               STATUS.% = 0%
...
32720 !* * * * *   STATUS PROCESSOR
32732 STATUS.% = ERR
32734 RESUME
```

The status processor error-handling routine can be enhanced with many features without destroying its generality. The ABORT program illustrates use of a generalized error-handler for status processing, user error notification, control C trapping and setting of exit status under RSX11M.

```
                                    ABORT.B2S
10-Oct-84   09:31 PM                                  Fish and Game   Page  1
================================================================================

1     ON ERROR GOTO 32730 ! Status Processor
2     CTRLC.% = CTRLC     ! Enable Control C trapping
900   !
901   !Standard Constants
902         TRUE.%              =  -1%       \ FALSE.%              =   0%
903         OK.%                =   0%
904         ILLEGAL.FILE.NAME.% =   2%       \ FILE.NOT.FOUND.%     =   5%
905         DUPLICATE.KEY.%     = 134%       \ RECORD.NOT.FOUND.%   = 155%
906         RECORD.LOCKED.%     = 154%       \ EOF.%                =  11%
907         CONTROL.C.%         =  28%       \ DATA.FORMAT.ERROR.%  =  50%
908         INTEGER.ERROR.%     =  51%       \ ILLEGAL.NUMBER.%     =  52%
909   !
910   !Initialize variables
911         ABORT.% = FALSE.%                \ STATUS.% = OK.%

1000  ! * * * * *    PROGRAM ABORT
1010  ZERO% = 0%
1020  UNTIL ZERO% > 0%
1030        IF   STATUS.% = OK.%                                          &
            THEN DIVISION.BY.ZERO% = ZERO% / ZERO%  ! Force an error!     &
            ELSE PRINT STATUS.MSG.$                                     \ &
                 STATUS.% = OK.%
1040        PRINT "Type Control C at anytime to terminate program "
1050        NEXT
1060  GOTO 32767 ! END

32730 !* * * * *    STATUS PROCESSOR
32732 CTRLC.% = RCTRLC ! Stop Control C trapping
32734 STATUS.% = ERR
32736 STATUS.MSG.$ = ERT$(ERR) + " (" + EDIT$(STR$(ERR),2%) + ")" +       &
                     " at line" + STR$(ERL) + "in " + ERN$ + BEL
32738 IF    STATUS.% <> DUPLICATE.KEY.%     AND                           &
            STATUS.% <> DATA.FORMAT.ERROR.% AND                           &
            STATUS.% <> RECORD.NOT.FOUND.%  AND                           &
            STATUS.% <> CONTROL.C.%                                       &
      THEN GOSUB 32750 ! Check For Looping!                               &
      ELSE !Skip
32740 IF    IN.LOOP.%  OR                                                 &
            STATUS.% = CONTROL.C.%                                        &
      THEN GOTO 32760 ! Terminate program!                                &
      ELSE CTRLC.% = CTRLC ! Start Control C Trapping!                  \ &
            RESUME
```

```
32750 ! * * * * *     STATUS PROCESSOR -- CHECK FOR LOOPING
32752 IF   PREV.STATUS.% = STATUS.%                                    &
      THEN IF   PREV.ERL.% = ERL                                       &
           THEN ERR.TALLY.% = ERR.TALLY.% + 1%                       \ &
                IF   ERR.TALLY.% > 20%                                 &
                THEN IN.LOOP.% = TRUE.%                                &
                ELSE ! Skip !                                          &
           ELSE PREV.ERL.% = ERL                                     \ &
                ERR.TALLY.% = 1%                                       &
      ELSE PREV.STATUS.% = STATUS.%                                  \ &
           PREV.ERL.% = ERL                                          \ &
           ERR.TALLY.% = 1%                                          \ &
           IF   STATUS.% = RECORD.LOCKED.%                             &
           THEN SLEEP 1%                                             \ &
                STATUS.% = OK.%                                        &
           ELSE ! Skip
32754 RETURN

32760 !* * * * *     STATUS PROCESSOR -- TERMINATE PROGRAM
32762 IF IN.LOOP.%                                                     &
         THEN PRINT "Looping at or about "; ERL; "in "; ERN$    ELSE   &
      IF STATUS.% = CONTROL.C.%                                        &
         THEN PRINT LF; "Control C termination at"; ERL; "IN "; ERN$   &
         ELSE PRINT STATUS.MSG.$
32764 CALL EXST(4%) !Severe error
32767 END
```

This method of error handling allows the programmer to process errors where they occur by simply checking STATUS.% as illustrated at statement 1030. The example program will loop until the Check For Looping routine detects the loop. The program will then terminate with exit status. Typing a control C at anytime will also terminate the program.

# A Generalized BASIC Error Processor

by

Donald L. Hallberg

State of California
The Resources Agency
Department of Fish and Game
3124 Nimbus Road, Suite E
Rancho Cordova, CA 95670

California Department of Fish and Game began using Basic-Plus-2 in 1978. At that time BP2 was in its infancy, version 1.1. We quickly learned that reading a file to EOF required a user written error-handling routine. The process involved a rather complex sequence of events: 1) error trapping must be enabled and BP2 told where the error-handler is located, 2) the error handler must be smart enough to figure out what the error was and take the appropriate action, 3) then the routine must know where to go in order for normal processing to continue.

Digital provided a few handy variables (ERR, ERL and ERN$) that are set when BP2 detects an error and unconditionally branches. The error-handler typically will make use of these variables to determine where to resume program execution.

We were bothered by the necessary use of GOTO-like constructs which did not conform to our structured coding conventions. We were further confounded by the fact that the error-handler had to figure out where to resume processing. We began looking for a way to make error trapping simple and straightforward. We discovered that using the RESUME statement without a trailing line number allowed us to write one generalized error-handler routine that could be used without modification in every program. The routine was actually a status processing routine that did nothing more than assign the value of ERR to a status variable before resuming program execution by re-executing the statement that caused the error.

The status variable is used to allow conditional branching and error processing at the place the error occured rather than in the error-handling routine. The following example illustrates the basic technique involved.

```
1     ON ERROR GOTO 32720 ! Status Processor
...
1230 IF   STATUS.% = 0%
     THEN GET  1%
     ELSE IF   STATUS.% = 11%
          THEN EOF% = -1%
               STATUS.% = 0%
          ELSE UNEXPECTED.ERROR% = STATUS.%
               STATUS.% = 0%
...
32720 !* * * * *   STATUS PROCESSOR
32732 STATUS.% = ERR
32734 RESUME
```

The status processor error-handling routine can be enhanced with many features without destroying its generality. The ABORT program illustrates use of a generalized error-handler for status processing, user error notification, control C trapping and setting of exit status under RSX11M.

```
                                    ABORT.B2S
10-Oct-84   09:31 PM                                    Fish and Game   Page  1
================================================================================

1     ON ERROR GOTO 32730 ! Status Processor
2     CTRLC.% = CTRLC     ! Enable Control C trapping
900   !
901   !Standard Constants
902         TRUE.%               =  -1%       \ FALSE.%              =   0%
903         OK.%                 =   0%
904         ILLEGAL.FILE.NAME.%  =   2%       \ FILE.NOT.FOUND.%     =   5%
905         DUPLICATE.KEY.%      = 134%       \ RECORD.NOT.FOUND.%   = 155%
906         RECORD.LOCKED.%      = 154%       \ EOF.%                =  11%
907         CONTROL.C.%          =  28%       \ DATA.FORMAT.ERROR.%  =  50%
908         INTEGER.ERROR.%      =  51%       \ ILLEGAL.NUMBER.%     =  52%
909   !
910   !Initialize variables
911         ABORT.% = FALSE.%                 \ STATUS.% = OK.%

1000  ! * * * * *    PROGRAM ABORT
1010  ZERO% = 0%
1020  UNTIL ZERO% > 0%
1030        IF   STATUS.% = OK.%                                              &
            THEN DIVISION.BY.ZERO% = ZERO% / ZERO%  ! Force an error!         &
            ELSE PRINT STATUS.MSG.$                                          \ &
                 STATUS.% = OK.%
1040        PRINT "Type Control C at anytime to terminate program "
1050        NEXT
1060  GOTO 32767 ! END

32730 !* * * * *    STATUS PROCESSOR
32732 CTRLC.% = RCTRLC ! Stop Control C trapping
32734 STATUS.% = ERR
32736 STATUS.MSG.$ = ERT$(ERR) + " (" + EDIT$(STR$(ERR),2%) + ")" +          &
                     " at line" + STR$(ERL) + "in " + ERN$ + BEL
32738 IF    STATUS.% <> DUPLICATE.KEY.%      AND                             &
            STATUS.% <> DATA.FORMAT.ERROR.% AND                              &
            STATUS.% <> RECORD.NOT.FOUND.%  AND                              &
            STATUS.% <> CONTROL.C.%                                          &
      THEN GOSUB 32750 ! Check For Looping!                                  &
      ELSE !Skip
32740 IF    IN.LOOP.%  OR                                                    &
            STATUS.% = CONTROL.C.%                                           &
      THEN GOTO 32760 ! Terminate program!                                   &
      ELSE CTRLC.% = CTRLC ! Start Control C Trapping!                      \ &
            RESUME
```

```
32750 ! * * * * *     STATUS PROCESSOR -- CHECK FOR LOOPING
32752 IF   PREV.STATUS.% = STATUS.%                                    &
      THEN IF   PREV.ERL.% = ERL                                       &
           THEN ERR.TALLY.% = ERR.TALLY.% + 1%                       \ &
                IF   ERR.TALLY.% > 20%                               &
                THEN IN.LOOP.% = TRUE.%                              &
                ELSE ! Skip !                                        &
           ELSE PREV.ERL.% = ERL                                     \ &
                ERR.TALLY.% = 1%                                     &
      ELSE PREV.STATUS.% = STATUS.%                                  \ &
           PREV.ERL.% = ERL                                          \ &
           ERR.TALLY.% = 1%                                          \ &
           IF   STATUS.% = RECORD.LOCKED.%                           &
           THEN SLEEP 1%                                             \ &
                STATUS.% = OK.%                                      &
           ELSE ! Skip
32754 RETURN

32760 !* * * * *     STATUS PROCESSOR -- TERMINATE PROGRAM
32762 IF IN.LOOP.%                                                     &
        THEN PRINT "Looping at or about "; ERL; "in "; ERN$    ELSE   &
      IF STATUS.% = CONTROL.C.%                                        &
        THEN PRINT LF; "Control C termination at"; ERL; "IN "; ERN$    &
        ELSE PRINT STATUS.MSG.$
32764 CALL EXST(4%) !Severe error
32767 END
```

This method of error handling allows the programmer to process errors where they occur by simply checking STATUS.% as illustrated at statement 1030. The example program will loop until the Check For Looping routine detects the loop. The program will then terminate with exit status. Typing a control C at anytime will also terminate the program.

# A Generalized BASIC Error Processor

by

Donald L. Hallberg

State of California
The Resources Agency
Department of Fish and Game
3124 Nimbus Road, Suite E
Rancho Cordova, CA 95670

California Department of Fish and Game began using Basic-Plus-2 in 1978. At that time BP2 was in its infancy, version 1.1. We quickly learned that reading a file to EOF required a user written error-handling routine. The process involved a rather complex sequence of events: 1) error trapping must be enabled and BP2 told where the error-handler is located, 2) the error handler must be smart enough to figure out what the error was and take the appropriate action, 3) then the routine must know where to go in order for normal processing to continue.

Digital provided a few handy variables (ERR, ERL and ERN$) that are set when BP2 detects an error and unconditionally branches. The error-handler typically will make use of these variables to determine where to resume program execution.

We were bothered by the necessary use of GOTO-like constructs which did not conform to our structured coding conventions. We were further confounded by the fact that the error-handler had to figure out where to resume processing. We began looking for a way to make error trapping simple and straightforward. We discovered that using the RESUME statement without a trailing line number allowed us to write one generalized error-handler routine that could be used without modification in every program. The routine was actually a status processing routine that did nothing more than assign the value of ERR to a status variable before resuming program execution by re-executing the statement that caused the error.

The status variable is used to allow conditional branching and error processing at the place the error occured rather than in the error-handling routine. The following example illustrates the basic technique involved.

```
1     ON ERROR GOTO 32720 ! Status Processor
...
1230 IF   STATUS.% = 0%
     THEN GET  1%
     ELSE IF   STATUS.% = 11%
          THEN EOF% = -1%
               STATUS.% = 0%
          ELSE UNEXPECTED.ERROR% = STATUS.%
               STATUS.% = 0%
...
32720 !* * * * *   STATUS PROCESSOR
32732 STATUS.% = ERR
32734 RESUME
```

The status processor error-handling routine can be enhanced with many features without destroying its generality. The ABORT program illustrates use of a generalized error-handler for status processing, user error notification, control C trapping and setting of exit status under RSX11M.

```
                                    ABORT.B2S
10-Oct-84   09:31 PM                                    Fish and Game   Page  1
================================================================================

1     ON ERROR GOTO 32730 ! Status Processor
2     CTRLC.% = CTRLC     ! Enable Control C trapping
900   !
901   !Standard Constants
902         TRUE.%              =  -1%       \ FALSE.%              =   0%
903         OK.%                =   0%
904         ILLEGAL.FILE.NAME.% =   2%       \ FILE.NOT.FOUND.%     =   5%
905         DUPLICATE.KEY.%     = 134%       \ RECORD.NOT.FOUND.%   = 155%
906         RECORD.LOCKED.%     = 154%       \ EOF.%                =  11%
907         CONTROL.C.%         =  28%       \ DATA.FORMAT.ERROR.%  =  50%
908         INTEGER.ERROR.%     =  51%       \ ILLEGAL.NUMBER.%     =  52%
909   !
910   !Initialize variables
911         ABORT.% = FALSE.%                \ STATUS.% = OK.%

1000  ! * * * * *    PROGRAM ABORT
1010  ZERO% = 0%
1020  UNTIL ZERO% > 0%
1030        IF   STATUS.% = OK.%                                          &
            THEN DIVISION.BY.ZERO% = ZERO% / ZERO%  ! Force an error!     &
            ELSE PRINT STATUS.MSG.$                                     \ &
                 STATUS.% = OK.%
1040        PRINT "Type Control C at anytime to terminate program "
1050        NEXT
1060  GOTO 32767 ! END

32730 !* * * * *    STATUS PROCESSOR
32732 CTRLC.% = RCTRLC ! Stop Control C trapping
32734 STATUS.% = ERR
32736 STATUS.MSG.$ = ERT$(ERR) + " (" + EDIT$(STR$(ERR),2%) + ")" +      &
                     " at line" + STR$(ERL) + "in " + ERN$ + BEL
32738 IF   STATUS.% <> DUPLICATE.KEY.%     AND                            &
           STATUS.% <> DATA.FORMAT.ERROR.% AND                            &
           STATUS.% <> RECORD.NOT.FOUND.%  AND                            &
           STATUS.% <> CONTROL.C.%                                        &
      THEN GOSUB 32750 ! Check For Looping!                               &
      ELSE !Skip
32740 IF   IN.LOOP.%  OR                                                  &
           STATUS.% = CONTROL.C.%                                         &
      THEN GOTO 32760 ! Terminate program!                                &
      ELSE CTRLC.% = CTRLC ! Start Control C Trapping!                  \ &
           RESUME
```

```
32750 ! * * * * *     STATUS PROCESSOR -- CHECK FOR LOOPING
32752 IF   PREV.STATUS.% = STATUS.%                                    &
      THEN IF   PREV.ERL.% = ERL                                       &
           THEN ERR.TALLY.% = ERR.TALLY.% + 1%                       \ &
                IF   ERR.TALLY.% > 20%                               &
                THEN IN.LOOP.% = TRUE.%                              &
                ELSE ! Skip !                                        &
           ELSE PREV.ERL.% = ERL                                     \ &
                ERR.TALLY.% = 1%                                     &
      ELSE PREV.STATUS.% = STATUS.%                                  \ &
           PREV.ERL.% = ERL                                          \ &
           ERR.TALLY.% = 1%                                          \ &
           IF   STATUS.% = RECORD.LOCKED.%                           &
           THEN SLEEP 1%                                             \ &
                STATUS.% = OK.%                                      &
           ELSE ! Skip
32754 RETURN

32760 !* * * * *     STATUS PROCESSOR -- TERMINATE PROGRAM
32762 IF IN.LOOP.%                                                     &
         THEN PRINT "Looping at or about "; ERL; "in "; ERN$    ELSE   &
      IF STATUS.% = CONTROL.C.%                                        &
         THEN PRINT LF; "Control C termination at"; ERL; "IN "; ERN$   &
         ELSE PRINT STATUS.MSG.$
32764 CALL EXST(4%) !Severe error
32767 END
```

This method of error handling allows the programmer to process errors where they occur by simply checking STATUS.% as illustrated at statement 1230. The example program will loop until the Check For Looping routine detects the loop. The program will then terminate with exit status. Typing a control C at anytime will also terminate the program.

# BASIC *MAGIC*

```
! program to give easier access to video attributes on VT100 terminal
! Jan/83   John McGuire


2 gosub 32700
3!                                                                          &
 !                                                                          &
 !      ACTION                    TURN ON                   TURN OFF        &
 !   ***************************************************************      &
 !                                                                          &
 !   CLEAR SCREEN                 CLS$                      (N/A)           &
 !   REVERSE VIDEO                REVID$                    RTN$            &
 !   MAKE STUFF BLINK             BLINK$                    UNBLINK$        &
 !   GRAPHICS MODE                GRAPH$                    UNGRAPH$        &
 !   FILL SCREEN WITH E's         EE$                                       &
 !   SCREEN WIDTH 132             WIDE$                     NARROW$         &
 !   SET PRINT WIDTH              N=FNWIDTH(N)                              &
 !   BACKGROUND COLOR             BLACK$                    WHITE$          &
 !   LIGHT LED's                  LED1$                     LED0$           &
 !                                LED2$                                     &
 !                                LED3$                                     &
 !                                LED4$                                     &
 !                                                                          &
 !   MOVE CURSOR                  M=FNMOVE(x,y)                             &
 !         (NOTE: Any variable can be used instead of 'M'.                  &
 !                The number put in 'x' designates which row,               &
 !                and the number in 'y' designates which column             &
 !                the cursor moves to.                                      &
 !                                                                          &
 !   SET SCROLLING AREA           S=FNSCROLL(t,b)                           &
 !         (NOTE: As above the 'S' variable can be changed.                 &
 !                The number in 't' is the top row,                         &
 !                the number in 'b' is the bottom row.                      &
 !                                                                          &
 !   REVERSE SCROLLING            REVSCROL$                                 &
 !                                                                          &
 !         (NOTE: This command moves the cursor up one line. If             &
 !                the cursor is at the top of the page, a                   &
 !                reverse scrolling occurs. Also, it is a                   &
 !                'one-shot' occurance rather than one that                 &
 !                continues until reset. To have multiple lines             &
 !                move down, a loop is needed.                              &
 !                                                                          &
 !   ERASING                                                                &
 !     LINE                                                                 &
 !        Cursor to end of line              ERLE$                         &
 !        Beginning of line to cursor        ERLB$                         &
 !        All of line                        ERLA$                         &
 !     SCREEN                                                               &
 !        Cursor to end of screen            ERSE$                         &
 !        Beginning of screen to cursor      ERSB$                         &
 !        All of screen                      ERSA$                         &
 !                                                                          &
 !                                                                          &
 !   CHARACTER SIZE                                                         &
 !       DOUBLE WIDTH,                                                      &
 !        DOUBLE HEIGHT           DWDH.T$    (TOP HALF)                    &
 !                                DWDH.B$    (BOTTOM HALF)                 &
 !       DOUBLE WIDTH,                                                      &
 !        SINGLE HEIGHT           DWSH$                                     &
```

```
!                                                                    &
!       SINGLE WIDTH,                                                &
!        SINGLE HEIGHT          SWSH$                                &
!                                                                    &
!          (NOTE: Character size is affected for one line ONLY)      &
!                                                                    &
!   MASTER RESET                RESET$                               &
!                                                                    &
!                                                                    &
!
32699 GOTO 32767
32700 esc$=chr$(155)                                                 &
\               print esc$;'<';                                      &
\               cls$ = esc$ + '[r' + esc$ + '[2J'                    &
\               def* fnmove(x,y)                                     &
\                print 'off screen error' if x<1 or x>24 or y<1 or y>80 &
\                print esc$+ '[' + num1$(x) + ';' + num1$(y) + 'f' ; &
\               fnend                                                &
\               revid$ = esc$ + '[7m'                                &
\               rtn$ = esc$ + '[0m'                                  &
\               blink$ = esc$ + '[5m'                                &
\               unblink$ = esc$ + '[0m'                              &
\               grapn$ = esc$ + '(0'                                 &
\               ungraph$ = esc$ + '(B'                               &
\               DWDH.T$ =esc$ + '#3'                                 &
\               DWDH.B$ = esc$ + '#4'                                &
\               SWSH$ = esc$ + '#5'                                  &
\               DWSH$ = esc$ + '#6'                                  &
\               black$ = esc$ + '[?51'                               &
\               white$ = esc$ + '[?5h'                               &
\               EE$ = esc$ + '#8'                                    &
\               reset$ = esc$ + '[2;1y'                              &
\               LED1$ = esc$ + "[1q"                                 &
\               LED2$ = esc$ + "[2q"                                 &
\               LED3$ = esc$ + "[3q"                                 &
\               LED4$ = esc$ + "[4q"                                 &
\               LED0$ = esc$ + "[0q"                                 &
\               narrow$ = esc$ + '[?31'                              &
\               wide$ = esc$ + '[?3h'                                &
\               revscrol$ = esc$ + 'M'                               &
\               ERLE$ = esc$ + '[0K'                                 &
\               ERLB$ = esc$ + '[1K'                                 &
\               ERLA$ = esc$ + '[2K'                                 &
\               ERSE$ = esc$ + '[0J'                                 &
\               ERSB$ = esc$ + '[1J'                                 &
\               ERSA$ = esc$ + '[2J'                                 &
\               def* fnwidth(n)                                      &
\               V$=SYS(CHR$(6%)+CHR$(16%)+CHR$(0%)+CHR$(255%)+CHR$(N+1))&
\               fnend                                                &
\               def* fnscroll(t,b)                                   &
\                print 'off screen error' if t<1 or t>24 or t<1 or t>24 &
\                print esc$+ '[' + num1$(t) + ';' + num1$(b) + 'r' ; &
\               fnend
32725           return
32767 end
```

## DECUS PROCEEDINGS

For your convenience and information listed below are the current DECUS Proceedings that are available and can be ordered through the DECUS office in Marlboro, Massachusetts. As availability changes this list will be updated.

| | | | DECUS Part No. | Media Service Codes |
|---|---|---|---|---|
| Canada | 1982 | Toronto, Canada | PROC-82/V08.3 | YA |
| U.S. Spring | 1982 | Atlanta, Georgia | PROC-82/V08.4 | YA |
| Europe | 1982 | Warwick, UK | PROC-EUR-82 | YA |
| U.S. Fall | 1982 | Anaheim, California | PROC-ANA-82 | YA |
| U.S. Spring | 1983 | St. Louis, Missouri | PROC-STLO-83 | YA |
| U.S. Fall | 1983 | Las Vegas, Nevada | PROC-FALL-83 | YA |
| U.S. Spring | 1984 | Cincinnati, Ohio | PROC-SPR-84 | YA |

PLEASE NOTE: The DECUS Proceedings are no longer grouped together in one volume; they are each listed separately. European, Canadian and Australian Proceedings will be listed by the year, date and place of the symposium. U.S. Proceedings will be listed by the year, season (Spring or Fall) and place of the symposium.

new
CPM-157

Vol. 80 Cromemco Structured BASIC Programs

Version: April 1984

Author: Various

Submitted By: Digital Equipment Corporation

Operating System: CP/M-80

Source Language: Cromemco BASIC

Memory Required: 64KB-128KB

The following is a brief description of the programs to found on the diskette:

This package contains programs written in the Cromemco Structured BASIC language. Source code is included, executables are not. Most have hardware dependency on a Hazeltine 1500 terminal.

The programs included are a spelling checking program, a mailing list program, utility to convert Microsoft and C-BASIC programs into structured BASIC, a statistical package and Startrek.

This package was developed on a Z80 chip. It was not developed on a Digital Equipment Corporation personal computer. In some cases, the source code might make specific calls to the hardware which would require changes to the sources.

There are no guarantees that this software will run "AS IS" across the Rainbow, the DECmate II (with CP/M option), or the Professional-300 (with CP/M option), series of computers.

Documentation on magnetic media.

Media (Service Charge Code): 5 1/4" Floppy Diskette (JA)

Keywords: CP/M - BASIC
Operating System Index:
CP/M-80

October 1, 1984

new
DM-106

COS-310 ISAM and Screen Handler

Version: July 1984

Author: Rudi Stange

Operating System: COS-310 V9.0 or later

Source Language: DIBOL

Memory Required: 16KW

Two subroutines are supplied: 1) demonstrating the index sequential access method, 2) enabling easy screen handling of 1...100 separate fields on the screen. Each field is controlled for alpha and/or numerical input/output as well as decimal point control. The cursor can be started in any desired fields, skipping desired fields, backspacing fields and/or restricting access to certain field(s). A sample customer data base is included to demonstrate the operation. File READ.AS gives hints to programmers as well as to users. Operation is initiated via a menu. A blank floppy must be inserted into DZ1: to operate the Demos.

Changes and Improvements: Converted from PDP-8 to Decmate II. Improvements to the Screen Handler.

Restrictions: User must have a valid COS-310 license.

Documentation on magnetic media.

Media (Service Charge Code): 5 1/4" Floppy Diskette (JA)

Format: OS/8

Keywords: ISAM
Operating System Index:
COS-310

October 1, 1984

new
11-753

RSTS/E Whittier College Package, Part I

Version: July 1984

Author: David Garland, Whittier College, Whittier, CA

Operating System: RSTS/E

Source Language: BASIC-PLUS

Memory Required: 16KW

The following is a brief description of the programs to be found on the tape:

JUMP - allows certain users to change accounts without knowing the passwords.
KILLO - will delete files with Ø blocks.
BIGFIL - locates and optionally deletes large files.
PROBIT - can set and clear the "protect" bit on a file. When the protect bit is set on a file, that file can not be deleted or renamed even by a privileged user.
CLASS - creates or deletes a range of accounts.
ACCLST - gives a list of all the accounts on the system in a compact format.
PACKID - allows you to change the ID of a disk pack.
PRIV - gives a list of all files which have a privileged protection code.
LAST - accesses the "last logged in" data in the GFD for specific accounts, keyboards, and dates.
PROTEC - reserves a terminal for a short time.
MFDBIT - will set the "marked for deletion" bit on a file, making it partially invisible. FIND will list all the files which have this bit set.
GFD and UFD - access the data in the GFD and UFD. BFD is included just for fun.
DCN and RETRIE both print out the contents of the Retrieval Blockette in the UFD.
DATETI - accesses the data in the Date/Time blockette of the GFD.

Documentation on magnetic media.

Media (Service Charge Code): 600' Magtape (MA)

Format: DOS-11

Keywords: RSTS/E - System Management, Utilities - RSTS/E Operating System Index: RSTS/E

October 1, 1984

new
11-754

RSTS/E Whittier College Package, Part II

Version: July 1984

Author: David Garland, Whittier College, Whittier, CA

Operating System: RSTS/E

Source Language: BASIC-PLUS

Memory Required: 16KW

The following is a brief description of the programs to be found on the tape:

ENCODE and DECODE - encrypt and decrypt files.
DEVCNT - accesses information in the monitor tables having to do with devices.
KBJOB - looks through the job tables to find out which job (if any) is associated with a given keyboard.
DDB - accesses information in the Device Date Block for a keyboard.
PLB - prints the information in the Pack Label Block of a disk.
PIP2KB - is a program designed to display specified columns of files.
LOGMSG - will append to existing login messages (or create new ones) in a range of accounts.
ERROR - prints error messages by number or by partial contents.
MODE8 - demonstrates Mode 8 input.
RAD5Ø - converts three character strings into their Rad5Ø representation.
WORDS - counts the number of lines, words, and sentences in a text file.
ALARM and ALARMO - let users set alarms at their terminals.
ODDNAM - allows you to have files with "illegal" names.
DATE - changes creation and access dates for files.
SPY - prints out the contents of someone else's input and output buffers.
WCWP - is just for fun.

Documentation on magnetic media.

Media (Service Charge Code): 6ØØ' Magtape (MA)

Format: DOS-11

Keywords: Utilities - RSTS/E,
RSTS/E - System Management
Operating System Index:
RSTS/E

October 1, 1984

new
V-SP-29

Symposium Tape from the VAX SIG, Spring 1984, Cincinnati

Version: Spring 1984

Author: Various

Submitted By: J. L. Bingham, Mantech International,
Alexandria, VA

Operating System: VAX/VMS V3.x

Source Language: VAX-11 BASIC, BLISS-32, C, VAX-11 COBOL, DCL, VAX-11 FORTRAN, MACRO-11, MACRO-32, PASCAL, TECO, SNOBOL, STOIC

This package contains material submitted for the Tapecopy project at the Spring 1984, Cincinnati, DECUS symposium. It nearly fills two 2400 foot reels of tape. The first reel contains two backup save sets, VAX000 which contains general information and indexes into the VAX SIG Symposium tapes and VAX84A (about 45% by volume of the submitted material). The second tape contains VAX84B (the four largest packages submitted.)

This symposium tape, as usual, contains a potpourri of new and revised programs, command procedures and other valuable material. Much of the bulk of this tape consists of updates to programs which have appeared on other VAX SIG tapes. Some new items are: Reece BASIC (from the RSX SIG), the SLIDES used by the VAX/VMS developers during their talks at the Cincinnati symposium, and HEX to manipulate ASCII hex formatted files. For more specific content, the reader is advised to obtain a copy of the tape and read the AAAREADME.TXT files.

Note: Release notes are distributed with each order.

Complete sources not included. Documentation on magnetic media.

Media (Service Charge Code): 2400' Magtape (PB)

Format: VMS/BACKUP (Blocked at 7952)

Keywords: Symposia Tapes - VMS
Operating System Index:
VAX/VMS

October 1, 1984

new
11-SP-69

Symposium Tape from the German RSX SIG, Spring 1984, Darmstadt

Version: Spring 1984

Author: Various

Submitted By: Klaus Centmayer, TU Muenchen, Lst.f. Datenverarbeitung, Munich, West Germany

Operating System: IAS, RSX-11D, RSX-11M

Source Language: BASIC, FORTRAN IV, FORTRAN IV-PLUS, MACRO-11, TECO

This tape contains the programs submitted by users at the DECUS Munich Symposium, 1984. The following is a very brief summary of the programs and routines on the tape. This collection also includes some revised versions of other RSX SIG tapes and a summary of available DECUS-SIG-tapes.

German RUNOFF, RECFIL, File Recover, Disk File Change, Editors, Text-processing, FORTRAN-debug, BASIC, Games, Common Region Modify, Device online/offline, Eventflag Report, Monitor Console Emulator, Computer Link, Catch All, AR11-driver, ZX-driver, Driver Database Dump, TT Status, Sysgen-modifications, TECO-macros, CAMAC Support, Plot-routines, Pictures on LP, Fast File I/O Routines, div. MT-routines (QIO, foreign tapes, EBCDIC-ASCII), Variable send/receive, Program-AST, IAS-utilities; Batch Level, STD.

No guarantees are made as to the completeness, usability, or quality of the programs on the tape and the material has not been checked or reviewed.

Documentation may or may not be included on the magnetic media.

Media (Service Charge Code): 2400' Magtape (PS)

Format: DOS-11 (1600 bpi ONLY)

Keywords: Symposia Tapes - RSX-11
Operating System Index: RSX-11/IAS

October 15, 1984

new
11-SP-70

Symposium Tape from the RSTS/BASIC SIG, Spring/Fall 1983

Version: V1.0, July 1984

Author: Various

Submitted By: Philip Hunt, System Industries, Milpitas, CA

Operating System: RSTS/E V8.0

Source Language: BASIC-PLUS, BASIC-PLUS2, MACRO-11

This tape contains the entries to the RSTS and BASIC SIGS Tape Copy Project for the Spring and Fall 1983 U.S. Chapter DECUS Symposia. The tape includes but is not limited to the following items: an implementation of the VAX program phone for RSTS, a 'standard' EDT initialization file, an editor runtime system, a general pseudo-keyboard driver, a version of CB for BP2 V1.6, a set of HASP utilities, PortaCalc, APL runtime systms, a set of APL and FORTRAN utilities, ANSI BASIC standard draft, TEDIT and others.

No guarantees are made as to the completeness, usability, or quality of the programs on the tape and the material has not been checked or reviewed.

Documentation may or may not be included on the magnetic media.

Media (Service Charge Code): 2400' Magtape (PS)

Format: DOS-11

Keywords: Syposia Tapes - RSTS/E, PORTACALC, APL
Operating System Index: RSTS/E

October 15, 1984

new
11-SP-71

Symposium Tape from the RSTS/BASIC SIG, Spring 1984, Cincinnati

Version: V1.Ø, July 1984

Author: Various

Submitted By: Philip Hunt, System Industries, Milpitas, CA

Operating System: RSTS/E V8.Ø, VAX/VMS V3.4, V3.5, V3.6

Source Language: VAX-11 BASIC, BASIC-PLUS, BASIC-PLUS2, MACRO-11, MACRO-32

Memory Required: Varies

This tape contains the entries to the RSTS/E and BASIC SIGS Tape Copy Project for the Spring 1984 U.S. Chapter DECUS Symposia. The tape includes, but is not limited to the following: A SYSTAT patch, some editing files, description of some BP2 bugs and examples, a terminal spy system built into the monitor, the latest version of RSTS/KERMIT with sources, a job dump and display program, a COBOL program to generate large calendars, a RSTS/E tape management system, a disk quota checking and report program, the latest version of CB for V2 of BP2.

No guarantees are made as to the completeness, usability, or quality of the programs on the tape and the material has not been checked or reviewed.

Documentation may or may not be included on magnetic media.

Media (Service Charge Code): 2400' Magtape (PS)

Format: DOS-11

Keywords: Symposia Tapes - RSTS/E, Kermit
Operating System Index: RSTS/E, VAX/VMS

October 15, 1984

new
V-SP-30

NOTIFY

Version: V1.0, July 1984

Author: Tim Steele, Peter Steele & Partners Ltd., Knowle, Solihull, West Midlands, UK

Operating System: VAX/VMS V3.6

Source Language: VAX-11 BASIC

This utility is for use under VAX/VMS. It sends a single line of text to one or more users on the system. It is intended as a more flexible substitute for use of REQUEST and REPLY for user to user communication. No privileges are required. The username may be specified ambiguously, for example JA would match all users whose username began with JA. For example, user ALPHA might type:

```
$ NOTIFY BRAVO "Seen CHARLIE lately?"
and user BRAVO would see:
***From Alpha: Seen CHARLIE lately?
```

The utility comes with an addition to the VMS Help library.

Note: Release notes are distributed with each magtape.

Documentation on magnetic media.

Media (Service Charge Code): 600' Magtape (MC)

Format: VMS/BACKUP (Blocked at 2048)

Keywords: Utilities - VMS
Operating System Index:
VAX/VMS

October 15, 1984

new
VAX-101

Comprehensive VAX Demonstration Package

Version: V1.0, July 1984

Author: Michael Powell, Juniata College, Huntingdon, PA

Operating System: VAX/VMS V3.5

Source Language: VAX-11 BASIC, CDD, VAX-11 COBOL, DBMS, DCL, FORTRAN IV, MACR0-11, PASCAL, VAX-11 PL/1

Memory Required: 70KB

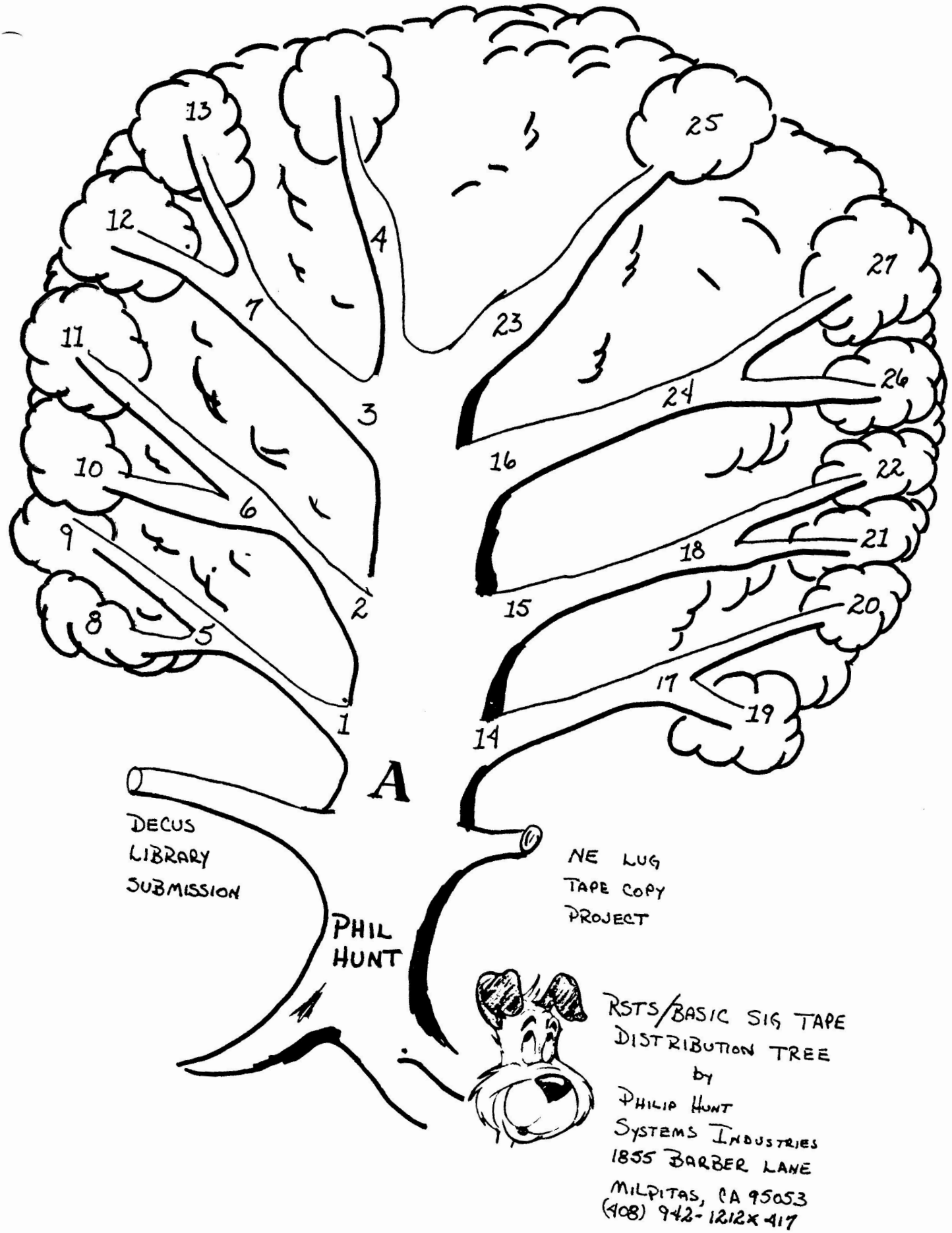
This demonstration package is for all VAX users. It will give a good idea of what the beast is made out of, as well as what software is available to be used. If you do not have a particular item in the demo list EDIT MODULE.FIL to change the demonstration modules available. For a more complete description of the functions of each file for this demonstration look at the file DEMODES.TXT.

Media (Service Charge Code): 600' Magtape (MA)

Format: VMS/BACKUP (Blocked at 2048)

Keywords: Utilities - VMS, Demonstrations
Operating System Index: VAX/VMS

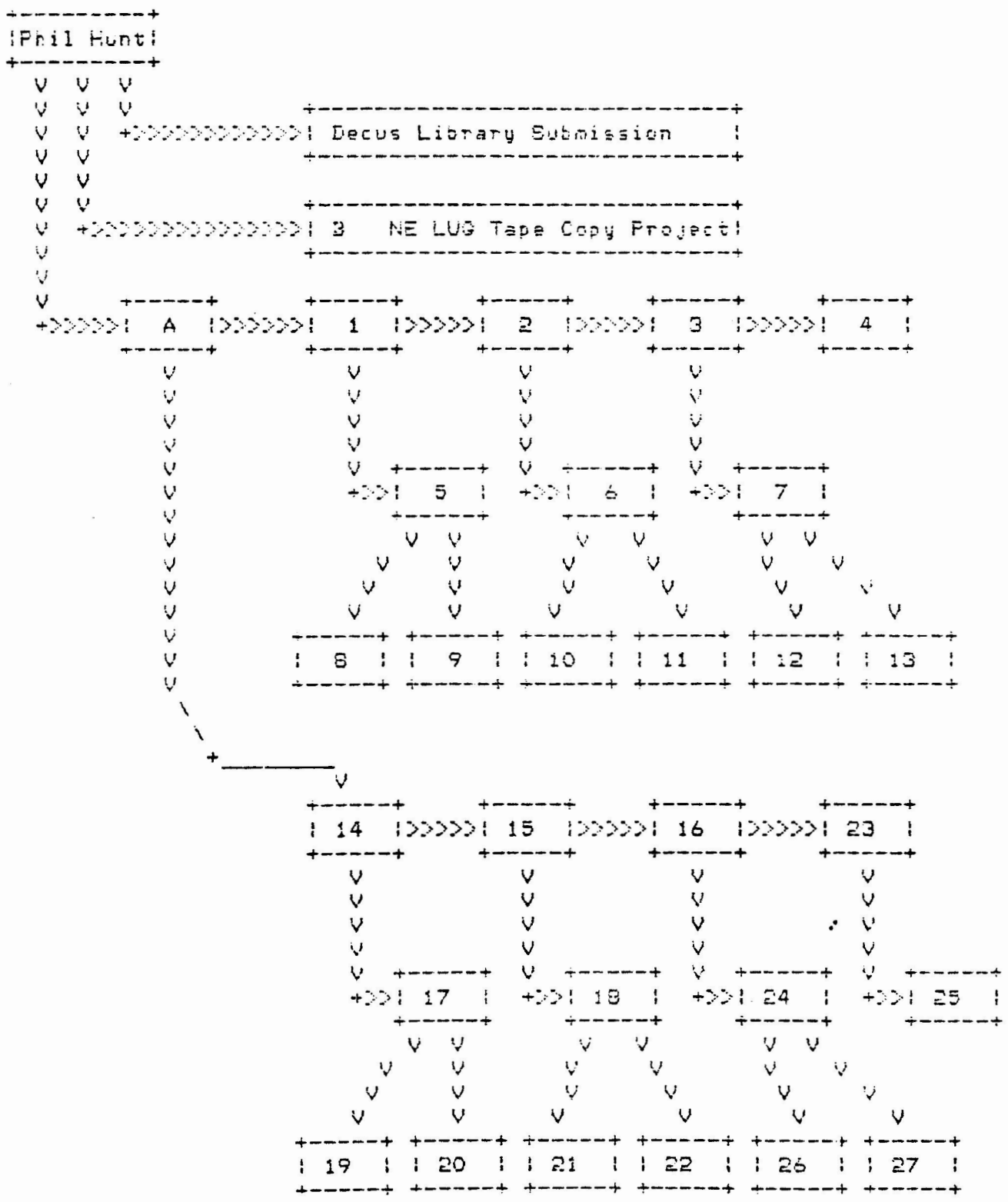
October 15, 1984


13
25
12
4
27
7
23
11
3
26
24
10
16
22
6
9
18
21
2
15
20
8
5
17
1
19
14
A
DECUS
LIBRARY
SUBMISSION
NE LUG
TAPE COPY
PROJECT
PHIL
HUNT
RSTS/BASIC SIG TAPE
DISTRIBUTION TREE
by
PHILIP HUNT
SYSTEMS INDUSTRIES
1855 BARBER LANE
MILPITAS, CA 95053
(408) 942-1212x417

RSTS/BASIC Sig Tape Distribution Tree 08/15/84

Prepared by Philip Hunt
Systems Industries
1855 Barber Ln
Milpitas, CA 95035
408-942-1212 x417

Distribution Tree diagram:



Directions for use:

Above this sheet is a diagram showing boxes with numbers and letters inside. Please send copies of enclosed tapes in DOE format at 1600 BPI to the people that are connected to your box by an arrow. PLEASE SEND A COPY (EXTRAS ENCLOSED) OF THE DISTRIBUTION LIST TO PEOPLE YOU SEND TAPES TO and let them know that this file is on the Spring '84 tape as [1,1]distr.lis.

Note: The most tapes anybody must send out is 2 copies, and some people do not have to send any. If the tape is returned or you would like to let the person know a tape is coming, I have included phone numbers when possible so you may contact them. Please send these tape AS SOON AS POSSIBLE!!!!!! Any questions, please contact me at the above number.

Any errors or ommissions or updates, please contact me and I will update the list. Thank-you.

A Gene Alpern
Saber Computer Services
1975 Johns Drive
Glenview, ILL 60025
312-998-5950

B Doug Eickford
Univ of Vermont
Academic Computing Center
Cook + Science Bldg
Burlington, VT 05401
802-656-3190

1 Hank Vander Waal
Prime Metals
3910 Roger Chaffee
G.R., MICH 49508
616-241-3451

2 Jim Bivins
TD Industries
13737 North Stemmons Fkwy
Dallas, TX 75234
214-620-1511

3 David Taft
USGS Box 25046 - MS 978
Denver Federal Center
Denver, CO 80225
303-234-5479

4 Gene Dugger
Harding University
Box 753
Searcy, ARKANSAS 72143
501-268-6161 x266

5 Steve Lorentzen
555 Fourth + Battery Bldg

Seattle, WA 98121
206-223-6453

6 Dan Tipton
ORFMA
117 Flint Road
Oakridge, Tn 37830
615-482-8000

7 Lee Gilbreath
SignAd, Inc.
PO Box 8585
Houston, TX 77249
713-861-6013

8 Guy Dunbar
Wm. E. Davis and Sons, Inc
73 NW 122nd Street
Oklahoma City, OK 73114
405-751-4660

9 Jerry Ethington
Federal Land Bank
201 W. Main St/PO BOX 32390
Louisville, Ky 40232

10 Robert Perry
Tektronix Inc
Box 500, M/S 19-333
Beaverton, Oregon 97077
503-627-5410

11 Tom Haase
SOHIO
4440 Warrensville CTR Rd
Cleveland, OH 44128
216-581-5724

12 Kathy Furtik - Computer Svcs
Lockwood Green Engineers
PO Box 491
Spartanburg, SC 29304
803-578-2000

13 Frank Atkinson
1900 E. Dublin-Granville Rd
Suite 100
Columbus, OH 43229
614-885-0841

14 Scott Matthews
Dow Chemical-Tx Div
A-2708 Railroad Dept
Freeport, Texas 77541
409-238-4816

15 Robert Walraven
Univ of California
Applied Science
Davis, CA 95616
916-752-0360

16 Micheal Newell
Florida Inst of Technology
150 W University Blvd
Melbourne, Fl 32901
305-723-3701 x255

17 Tom Gerhard
Advanced Data Management
15 Main St
Kingston, NJ 08528
609-799-4600

18 Marion Fauber
PO Box 1042 MS 4-07

Dayton, Oh 45401
513-258-7707

19 Roger Engleman
Computer Partners, Inc.
2995 N. Cole Suite 140
Boise, ID 83704
208-377-2070

20 Richard Wrenn
Washington Univ-Biochemistry
Campus Box 8094
St. Louis, MO 63110
314-362-3354

21 Robert Fairchild
Nebraska Wesleyan Univ
50th & St. Paul
Lincoln, NE 68504
402-466-2371

22 Virgil Larson
Univ of Minnesota
Box 732 Mayo, Surgery Dept
Minneapolis, MN 55455
612-376-9280

23 Pierre M. Hahn
SUNY - HSC - T10

Stony Brook, NY 11790-8101

24 Rocky Hayden
Userware International
2235 Meyers Ave.
Escondido, Ca 92025

25 Jim Reed
Federal Land Bank of Wichita
151 N. Main
Wichita, KS 67202

26 ***OPEN***

27 ***OPEN***

# HOW TO GET SIG TAPES PRODUCED AT THIS SYMPOSIUM

At each biannual DECUS symposium, several SIGs prepare a tape of free, user-contributed software. In the past, each SIG distributed its own tape via separate trees. However, during the Spring '84 Symposium the National LUG Organization (NLO) tested a new distribution method in the Northeast Region. Because of the success of that test, the NLO has undertaken, with the SIG Council's approval, national distribution of all SIG tapes. The two major features of this method are: 1) much flatter distribution trees, and 2) LUGs get all tapes from just 1 source.

## METHOD OF DISTRIBUTION

Distribution will be made to licensed LUGs in the U.S Chapter of DECUS, which is divided into seven regions. Individuals must obtain tapes through a LUG; the Regional Distributors cannot make copies for individuals. Each region has at least 1 Tape Distribution Coordinator who is a member of the National LUG Library Committee (NLLC). When ready, a SIG sends the NLO 2 copies of its tape; the NLO in turn distributes to the Regional Tape Distribution Coordinators. Before tapes are available, LUG chairs poll their membership for desired tapes and mail the necessary blanks, properly labeled with the tape name and return address, to the Regional Distributor. (Individuals should tell their LUG chair what tapes are wanted.) As soon as possible the Regional Distributor returns the copied SIG tapes to the LUGs. Finally, individuals obtain the SIG tapes through their LUG by whatever mechanism the LUG deems best.

Current NLO Regional Tape Distribution Coordinators (all are on DCS, the DECUS Leadership Communications System):

Northeast Region:
Douglas Bickford
Academic Computer Services
Cook Science Building
University of Vermont
Burlington, VT 05401 (802) 656-3190

NY/NJ Region:
Pierre Hahn
Dept. of Psychiatry
SUNY HSC-T10
Stony Brook, NY 11794-8101
(516) 444-1362

Central Region:
Bruce Mitchell
Machine Intelligence & Industrial Magic
P.O. Box 601
Hudson, WI 54016
(612) 736-2892

Western Region:
Robert Perry
Tektronix, Inc.
Box 500
MS 50-454
Beaverton, OR 97077
(503) 627-5410

Mid-Atlantic Region:
Rick Sharpe
Toledo Edison
300 Madison MS 3085
Toledo, OH 43652
(419) 259-5000 X470

Southern Region:
Dennis Clark
Oak Ridge National Lab
P.O. Box Y
Bldg. 9201-2 Room 209C
Oak Ridge, TN 37831
(615) 576-7384

Southwest Region:
Roger Jenkins
Wycliffe Bible Translators
19891 Beach Blvd.
Huntington Beach, CA 92647
(714) 536-9346

## WHICH SIGS PRODUCE TAPES, AND WHEN ARE THEY AVAILABLE?

Some SIGs produce a tape at every symposium; others produce a tape at some symposia, and some SIGs never produce a tape. (At the Spring '84 Symposium VAX, RSX, TOPS-10, TOPS-20, RSTS and RT tapes were produced.)

It is hard to predict just when a tape will be available, but usually distribution begins 8 to 16 weeks following the symposium. The NLLC will make available over DCS (the DECUS Communications System) a weekly status report of what tapes are being made and when they will be available. LUG chairs should consult this report or contact the Regional Distributor; individuals should contact their LUG chair for information. Do not try to contact the SIG tape maker (they are busy enough) or the DECUS Library (they will not know).

## HOW DO I GET COPIES OF OLD SIG TAPES?

The NLO Regional Distributors will maintain tapes at least until the next symposium's tapes are produced; they may optionally keep them longer. A LUG chair has three alternatives for obtaining archive tapes: another LUG willing to make a copy, the Regional Distributor if they still have it, or the DECUS library. Individuals' alternatives are: another site which has the tape, your LUG chair, or the DECUS Library. As above, the Regional Distributor will not copy tapes for individuals.

## WHAT IF I'M NOT A LUG MEMBER?

If you are not a LUG member, you need to find one near you geographically. Stop by the LUG Suite during symposium (you can find out where this is at the General Information Booth), or after the symposium contact:

LUG Administrator
DECUS
249 Northboro Road (BP02)
Marlboro, Mass. 01752

Either way, you'll be put in touch with a LUG near you.

## CAN I HELP THE NLO DISTRIBUTE SIG TAPES?

YES! If you are a LUG Chair or LUG Librarian with the hardware resources and time to copy 30 - 100 tapes, the NLLC wants you. Some regions need additional Regional Distributors where the number of LUGs in the region puts an undo burden on one site. All regions need backup sites to take over distribution in the event the Regional Distributor cannot continue. For more information contact the NLLC member in your region listed above.

Published by the NLO 12/84