General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

CR151908

SIGMA CORPORATION

TN 76-121

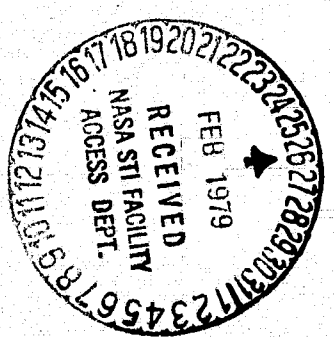
TECHNICAL NOTE

(NASA-CR-151908) UTILITIES (Sigma Corp., Houston, Tex.) 54 p HC A04/MF A01 CSCL 09B

N79-17574

Unclas
G3/61 13514

UTILITIES



By: Walter N. Colquitt

Prepared for:

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
Johnson Space Center
Houston, Texas 77058

September 1976

TABLE OF CONTENTS

UTILITIES

By: Walter N. Colquitt - Sigma Corporation

INTRODUCTION

This document is in several sections. Each section describes a set of related Adage utility programs. There are, in general, four parts to each section with the last part having two sub-parts. The first three parts are a general description of this software group, followed by instructions on how to use programs with the third part being a programmers description of the theory of operation. The final part is a printed listing with the first a printed example of the program in use and the second is a listing of the program.

DISK TO DISK TRANSFERS

The Univac side of disk-to-disk operations in the form of Adage supplied AGSCRD and AGSELT leave a great deal to be desired from the viewpoints of efficiency, stability, size, ease of use, ease of understanding and generality.

Two new Univac computer programs, UTA and ATU do much towards meeting these criteria. UTA transfers lines of data from the Univac computer to the Adage computer. ATU transfers lines of data from the Adage to the Univac.

The absolute (executable) versions of these computer codes are found in file EX42-N00002*UR. UTA is invoked as a processor and ATU is invoked as an executable (XQT) program.

Usage

In both cases after the program is invoked on the Univac side, the escape (ESC) key is hit and the Adage returns to the AMRMX monitor in the ordinary Adage mode. At this time, either AGSCRD or AGSELT is invoked on the Adage (for UTA and ATU respectively).

Use of the Univac to Adage (UTA) Routine

The rewrite of the Univac routine AGSCRD was carefully designed to overcome major shortcomings. It is no longer necessary to modify the element being sent, nor is it necessary to type more than one Exec 8 command card. In addition, the new version will transmit images up to a length of 132 characters. Core requirements are approximately 25% of the earlier version.

To use the routine type:

```
@UR.UTA          Your-File.Your-Element
```

When the message "USER DATA MESSAGE PENDING" begins to flash on the lower left of the screen, press the escape (ESC) key. This causes the AGT to cease functioning as a terminal and returns control to the AMOS 2 monitor.

Type "AGSCRD." The Adage will then request the pack/volume (ppvv) where to place the element as it comes across. When the transfer is complete and the Adage disk is closed out, the screen will go blank. At this time, type SYM11 to return to demand mode. An example of the use of UTA is shown in figure 2. Note that by default, the element being sent will be displayed on the Adage screen. Transfer time may be reduced by 60% by suppressing the

print. Print suppression is accomplished by the "N" option invoked as follows:

```
@UR.UTA,N              Your-File.Your Element
```

To transmit from the Univac to the Adage, an element which is more than 72 columns wide, then UTA should be invoked with the "T" (two pass) option, i.e.,

```
@UR.UTA,T              ur-file.ur-elem.
```

For an example, see figure 1-B.

Use of the Adage to Univac (ATU) Routine

The new Adage to Univac (ATU) routine has the capability of writing the image received from the Adage directly into an Univac program file as an element. Hence, no @ELT calls are necessary, and as tabbing is completed, no calls to the Editor are required.

To use the routine, type:

```
@XQT  UR.ATU
```

When the message "USER DATA MESSAGE PENDING" begins to flash in the lower left of the screen, press the escape (ESC) key. This causes the AGT to return control to the AMOS monitor.

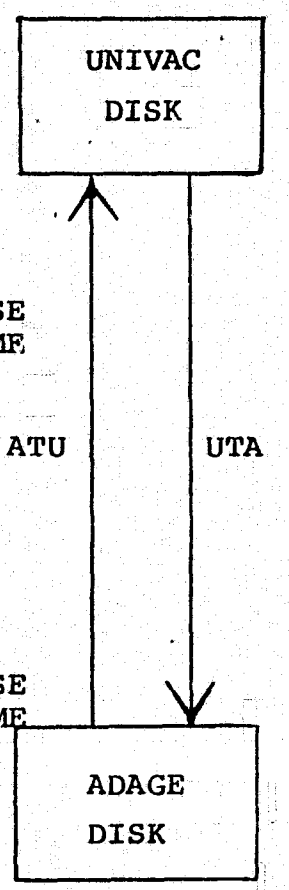
Type "AGSELT." The user should then answer the prompting questions and Adage file will be transferred over to the 1110 to become an Univac element of a program file.

For an example, see figure 1-A.

DISK-DISK COMMUNICATIONS

```
@XQT UR.ATU
ESC
AGSELT
ENTER DESIRED 1100 FILE NAME (
  C/R FOR TPS$) ...
FILE NAME
**ENTER C/R TO CONTINUE ANYTHING ELSE
  TO EXIT...ENTER DESIRED PACK/VOLUME
  NO (     PVV)...
110
ENTER STARTING FILE NUMBER...
6
ENTER ENDING FILE NUMBER...
14
**ENTER C/R TO CONTINUE ANYTHING ELSE
  TO EXIT...ENTER DESIRED PACK/VOLUME
  NO (     PVV)...
A
Screen Goes Blank
SYM11
```

FIGURE 1-A




```
@UR.UTA  urfile.elem
ESC
AGSCRD
ENTER DESIRED PACK/VOL NO (     PPV) ...

110
PROGRAM WILL APPEAR HERE
Screen Goes Blank
SYM11
```

FIGURE 1-B

FIGURE 1 DISK-DISK COMMUNICATIONS.

```
EXAMPLE OF ATU & UTA


TTY

AGS1Ø2
ENTER USERID/PASSWORD:
*DESTROY USERID/PASSWORD ENTRY
*UNIVAC 1100 OPERATING SYSTEM  VER. 31.244.211B (RSI)*
@RUN WC110,3039D-1088-C,EX42-NØØØØ2
DATE: Ø62576       TIME: 145540
@ED,I UTA-EXAMPLE
CASE UPPER ASSUMED
ED 14.02-06/25-14:56-(,Ø)
INPUT
1I:
FIRST LINE OF UNIVAC-TO-ADAGE EXAMPLE
2I:
LINE TWO
3I:
LAST LINE
4I:
@EOF
LINES:3 FIELDATA
@UR.UTA,N TPFS.UTA-EXAMPLE

*ESCAPE FROM UNIVAC 1100 DEMAND OPERATION*

AGSCRD

** START ON LINE CARD TO DISK PROGRAM **

  ENTER DESIRED PACK/VOL NO. (PVV) ...
110

**OUTPUT 'KR2' IN VOLUME 110
   FILE NO. = 16

SYM11


*SYM11  (VERSION 1, REV D,  SEPT 16, 1975)*

*START OF UNIVAC 1100 DEMAND OPERATION*
```

FIGURE 2 EXAMPLES OF ATU AND UTA.

```
@XQT UR.ATU

*ESCAPE FROM UNIVAC 1100 DEMAND OPERATION*

AGSELT

**START ON LINE ATEXT TO ELEMENT FILE PROGRAM**

ENTER DESIRED 1100 FILE NAME (C/R FOR TPF$) ...
TPF$

**ENTER C/R TO CONTINUE, ANYTHING ELSE TO EXIT ...

ENTER DESIRED PACK/VOLUME NO. (PVV) ...
110
ENTER STARTING FILE NUMBER ...
16
ENTER ENDING FILE NUMBER ...
16

**ENTER C/R TO CONTINUE, ANYTHING ELSE TO EXIT ...
DONE

SYM11


*SYM11  (VERSION 1, REV D,  SEPT 16, 1975)*

*START OF UNIVAC 1100 DEMAND OPERATION*

@PRT,T
FURPUR 0026-06/25-15:01
EX42-N00002*TPF$
ELT  UTA-EXAMPLE(0)
SYM  UTA-EXAMPL(0)
@FIN
  RUNID: WC110     ACCT: 3039D-1088-C   PROJECT: EX42-N00002
  TIME:     TOTAL: 00:00:08.672     CHARGE: 00:00:17.345
            CAU:   00:00:00.007     I/O:    00:00:00.717
            CC/ER: 00:00:07.948     WAIT:   00:02:59.176
  SRC:      PS --  6876               ES -- 10035
  START:     14:55:40 JUN 25,1976     FIN:  15:01:58 JUN 25,1976
*TERMINAL INACTIVE*
@@TERM

*END OF UNIVAC 1100 DEMAND OPERATION*
```

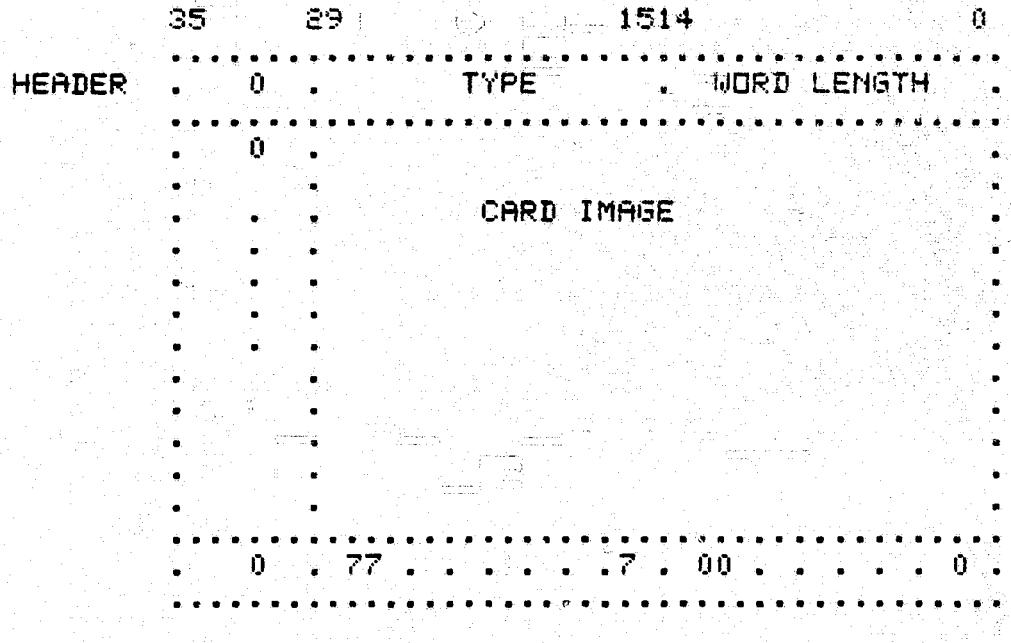
FIGURE 2 (CONTINUED)

Theory of Operation

The program was written by duplicating the logic in the old AGSCRD and AGSELT but expanding the I/O and executive interface to yield the effects desired. Much extraneous code was dropped. In general, to transmit data from the Univac to the Adage, information is received by a routine that blocks the data into 54 line pages. The page, with appropriate control words attached, is passed to IOG$ for transmittal to the Univac as a single data block. Data transmission in the opposite direction is exactly the same except the Univac side does deblocking rather than blocking. Note that AGSCRD and AGSELT on the Adage side were not modified at all and that transmission protocol is identical to the way it was in the past.

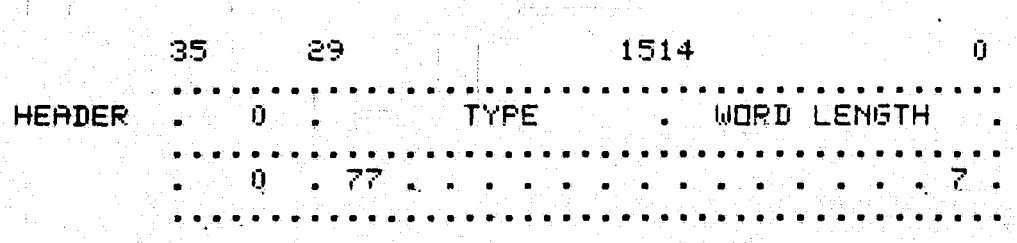
```
$(0)
.
. THIS IA A COMMUNICATION PROGRAM TO SEND
. ELEMENTS OF EXEC 8 PROGRAM FILES OVER TO ADAGE
. VIA IOG$. IT IS DESIGNED TO BE XQT'D AS A PROCESSOR
. WITH THE FILE AND ELEMENT TO BE SENT AS SPEC1 OF THE
. CALLING CARD. ONLY ONE ELEMENT IS TO BE TRANSMITTED
. FOR EACH TIME IT IS ENVOKED. THE ADAGE NAME IS THE FIRST
. 10 CHARACTERS OF THE ELEMENT NAME.
.
. AUTHOR - WALTER N COLQUITT
.
. DATE - FALL 1975
.
. CONTRACTOR - SIGMA CORP.
.
. LOCATION - NASA JOHNSON SPACE CENTER/HOUSTON,TEXAS
.
.  EXAMPLE
.
. TO SEND ELEMENT 'UTA/SYM' OF FILE 'DBINIT2'
.  OVER TO PPVV 333 OF THE ADAGE
.
.  @UR.UTA DBINIT2.UTA/SYM
.  ESC
.  AGSCRD
.  333
.  SYM11
.
. MESSAGE FORMATS
. NOTE ARE REALLY BITS 18 AND 17 NOT 15 AND 14 AS STATED HERE
.
. (1) START MESSAGE
.
.          35      29              1514                0
.          ..........................................
. HEADER   .   0   .       TYPE     .  WORD LENGTH    .
.          ..........................................
.          .   0   .              TITLE1               .
.          ..........................................
.          .   0   .              TITLE2               .
.          ..........................................
.          .   0   .  REV AND VERS  .      DATE        .
.          ..........................................
.
. TYPE = 0
```

(2) DATA MESSAGE



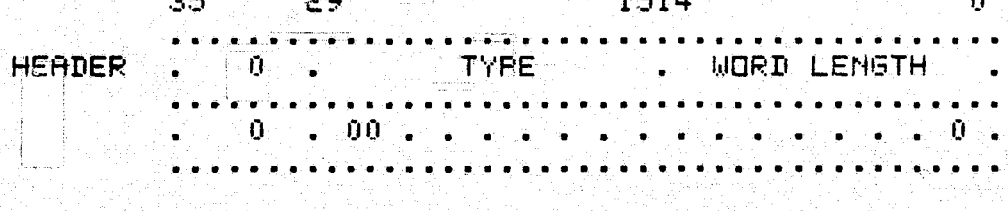
TYPE = 1

(3) EOF, EOF AND STOP MESSAGE



TYPE = 2 FOR EOF
TYPE = 3 FOR EOF AND STOP

(4) ERROR STOP MESSAGE



TYPE = 77777

```
/           .
PF          FORM        12,6,18
FLAGS       +           0
MAXSEND     EQU         12
IMAGELENGTH             EQU             22
HEADER      +           4
            +           0D
            +           0
TAIL        +           03414100000
ENDEOF      +           3,2
            +           1*/30-1
END         +           07777700000
INBUF       RES         2
IOGBUF      +           1,0
            RES         (55*(MAXSEND*6))//4+1
IMG         RES         IMAGELENGTH
BLANK       +           '   '
MSG1        'ERROR WHILE READING UNIVAC ELEMENT-CLOSING OUT'
MSG2        'FORMAT ERROR IN MESSAGE FROM ADAGE'
MSG3        'ERROR MADE ON ADAGE SIDE'
/           .
$(1)        AXR$
            .
            . INITIALIZATION SECTION
            .
GO          .
            LMJ         X11,BEGIN
            ER          EXIT$
            ER          ERR$
            -           0
            LA,U        A3,1
            LA          A0,PARTBL
            TOP,U       A0,1*/('Z'-'T')
            SZ          A3
            SA          A3,FLAGS
            LMJ         X11,INITG
            LA          A0,BLANK
            LR          R2,(-07777)
            MLU         A0,PARTBL+4
            LA          A0,PARTBL+3
            DS          A0,INBUF
            LMJ         X11,CNV8TA
            +           2,INBUF
            +           0,HEADER+1
            LA,S6       A0,PARTBL+12
            LSSL        A0,4
            AA,S4       A0,PARTBL+12
            LSSL        A0,5
            AA,S5       A0,PARTBL+12
            OR          A0,TAIL
            SA          A1,HEADER+3
            LMJ         X11,SENDG
            +           4,HEADER
```

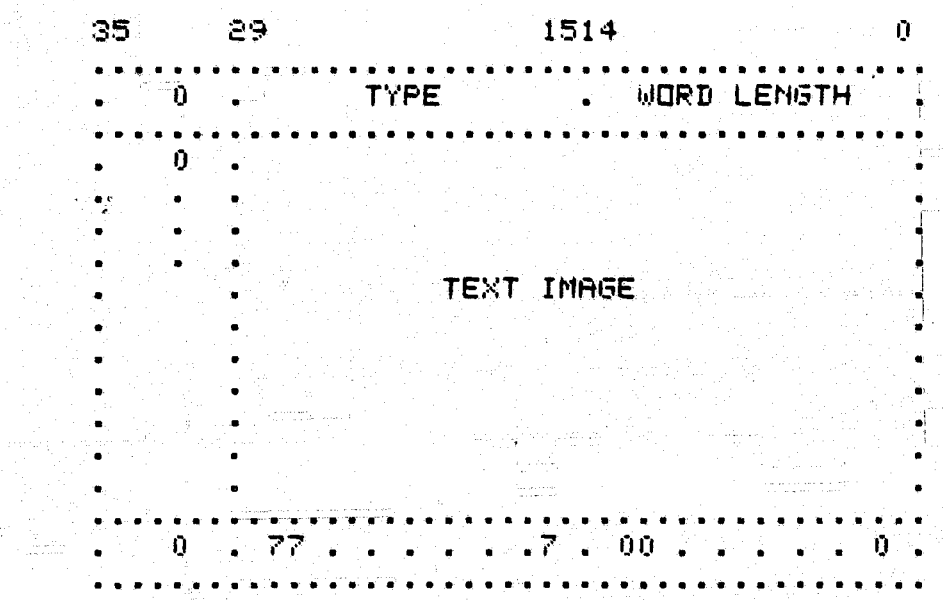
```
         .
         . THIS SECTION IS MAIN LOOP OUTER IS FOR EACH PAGE(54 LINES)
         . GET A LINE,COUNT NON-BLANK WORDS,CONVERT IT,AND PLACE IN BUFF
ER
NXTPAG   LX,U      X10,IOGBUF+1
         LA,U      A6,53
MORE     LMJ       X11,SIREAD
         J         ERR
         J         EOF
         +         IMG
         +         BLANK
         -         0
         LA        A2,BLANK
         LR,U      R1,IMAGELENGTH
         LXI,XU    A0,-1
         LXM,U     A0,IMG
         SNE       A2,IMAGELENGTH-1,*A0
         LR,XU     R1,-1
         SR        R1,A0
         LA,U      A0,1,A0
         TZ        FLAGS
         SLJ       RGTHLF
         TG,U      A0,MAXSEND
         LA,U      A0,MAXSEND
         SA,H1     A0,$+3
         SX,H2     X10,$+3
         LMJ       X11,CNVBTA
OUTACW   +         0,IMG
         +         0,0
         AX,H1     X10,$-1
         JGD       A6,MORE
         . ACTUALLY SEND THE PAGE VIA IOG$
SENDPG   LA        A0,END
         SA        A0,,X10
         ANX,U     X10,IOGBUF-1
         SX,H1     X10,$+3
         SX,H2     X10,IOGBUF
         LMJ       X11,SENDG
         +         0,IOGBUF
         J         NXTPAG
```

```
          .
          . THIS IS THE HANDLING OF THE 2ND PASS
          .   WHERE THE RIGHT HALF,WITH TWO WRDS OF BLANKS
          .     IS TRASMITTED
RGTHLF    +         0
          J         *RGTHLF
          +         0
          TNZ       FLAGS
          J         EOF+2
          LA,U      A0,DO2HLF
          SA,H2     A0,RGTHLF+1
          LA,U      A0,FCTI
          LMJ       X11,SDFIC
          LA        A0,PARTBL+11
          SA        A0,FCTI+5
          LA,U      A0,FCTI
          LMJ       X11,SDFIO
          +         0
          LA,U      A0,FCTI
          LMJ       X11,SDFI
          +         0
          +         0
          J         SENDPG
DO2HLF    ANA,U     A0,MAXSEND
          JN        A0,ALLBLKS
          JZ        A0,ALLBLKS
          AA,U      A0,2
          LA,U      A1,IMG
          SA        A2,MAXSEND-2,A1
          SA        A2,MAXSEND-1,A1
          AU,U      A1,MAXSEND-2
          SA        A2,OUTACW
          J         *RGTHLF
ALLBLKS   LA,U      A0,1
          LA,U      A1,BLANK
          SA        A1,OUTACW
          J         *RGTHLF
```

```
          .
          . CLOSE OUT SECTION
ERR       P$RINT    (PF 2,8,MSG1)
EOF       SLJ       $
          SLJ       RGTHLF+2
          LA        A0,END
          . SEND LAST PARTIAL PAGE
          SA        A0,,X10
          ANX,U     X10,IOGBUF-1
          SX        X10,A0
          TNE,U     A0,2
          J         SNDEOF
          SX,H1     X10,$+3
          SX,H2     X10,IOGBUF
          LMJ       X11,SENDG
          +         0,IOGBUF
          . SEND CLOSE OUT MESSAGE
SNDEOF    LMJ       X11,SENDG
          +         2,ENDEOF
          . CHECK TO SEE IF ANY ERROR RETURNS FROM ADAGE
          LMJ       X11,RECVG
          +         INBUF
          LA,H1     A1,$-1
          LA        A0,(PF 2,7,MSG2)
          TE,U      A1,1
          ER        PRINT$
          LA        A1,INBUF
          LSSL      A1,6
          LA        A0,(PF 2,6,MSG3)
          TP        A1
          ER        PRINT$
          . CLOSE IOG$
          LMJ       X11,CLOSEG
          . CLOSE IF PROCESSOR AND GO HOME
          LMJ       X11,DONE
          ER        EXIT$
          -         0
          ER        EXIT$
          END       GO
```

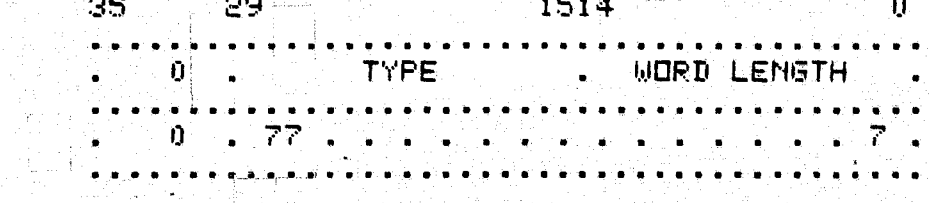
```
          THIS IS A COMPLETE REWRITE OF AGSELT
          THE PURPOSE OF THE REWRITE IS TO AVOID BOTH ELT AND ED CALLS
          AFTER XMISSION OF AN ELEMENT FROM ADAGE TO UNIVAC

        CONTRACTOR: SIGMA CORP.
        AUTHOR: WALTER N COLQUITT
        DATE: EARLY WINTER 1975

          TO EXECUTE THIS PROGRAM
          JUST TYPE @XQT FN.ATU
          THEN HIT ESC ON THE ADAGE AND DO REGULAR THING WITH AGSELT
          ON THE ADAGE SIDE

MESSAGE FORMAT FROM ADAGE
 NOTE THAT THESE ARE DIVIDED ON BIT 18 AND 17 NOT 15,14 SO COMPATABLE
(1) FILE NAME MESSAGE

          35      29                1514                  0
          .....................................................
          .   0   .       TYPE        .  WORD LENGTH        .
          .....................................................
          .   0   .       FILE NAME 1                       .
          .....................................................
          .   0   .       FILE NAME 2                       .
          .....................................................

TYPE = 07

(2) TITLE MESSAGE

          35      29                1514                  0
          .....................................................
          .   0   .       TYPE        .  WORD LENGTH        .
          .....................................................
          .   0   .              TITLE 1                    .
          .....................................................
          .   0   .              TITLE 2                    .
          .....................................................
          .   0   . TYPE, VER, REV   .      DATE            .
          .....................................................

TYPE = 0
```

(3) DATA MESSAGE



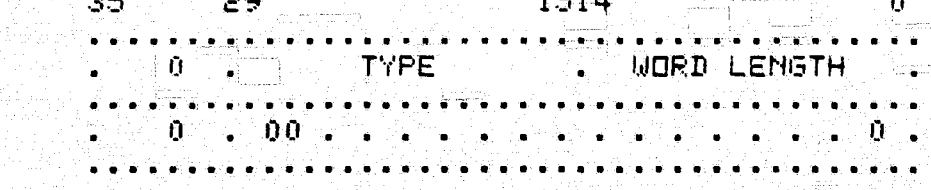
TYPE = 01

(4) EOF, EOF AND STOP MESSAGE



TYPE = 02 FOR EOF
TYPE = 03 FOR EOF AND STOP

(5) ERROR STOP MESSAGE



TYPE = 77777

```
.
. MESSAGE FORMAT TO ADAGE
.
. (1) ACK WORD
.
.          35      29                                        0
.          ..........................................
.          .   0   . 00 . . . . . . . . . . . . . . 0 .
.          ..........................................
.
. (2) NACK WORD
.
.          35      29                                        0
.          ..........................................
.          .   0   . 77 . . . . . . . . . . . . . . 7 .
.          ..........................................
/          .
PF         FORM      12,6,18
IMAGESIZE             EQU              22
           .
/          .
$(1)       AXR$
C4005◆     PROC      0
           LA,U      A1,9                .
           EX        FETCH,A1            . '0' TO SPACE
           JNZ       A0,$+2
           LA,U      A0,040
           EX        STORE,A1
           JGD       A1,$-4
           END
.
ATU        LMJ       X11,INITG           . INITIALIZE IOG$
           LMJ       X11,SENDG
           +         1,RBUF
RCVNAM     LMJ       X11,RECVG           . RECEIVE FILE NAME
           +         CNTRL               .
           LA        A0,CNTRL            . FETCH HEADER
           TE        A0,NAMHDR           .
           J         ERR1
```

```
          C4005
          LA,U      A13,1
          LMJ       X11,CNVAT8
          +         2,CNTRL+1
          +         ASG+2
          DL        A0,ASG+2
          DS        A0,USE+2
          LA        A0,(4,ASG)
          ER        CSF$
          JN        A0,ERRASG
          LA        A0,(5,USE)
          ER        CSF$
          JN        A0,ERRUSE
SDAKWD    LMJ       X11,SENDG
          +         1,ACKWD
          LMU       X11,RECVG
          +         CNTRL
          LA        A0,CNTRL
          TE        A0,CTLHDR
          J         EOFSTP
          C4005
          LMJ       X11,CNVAT8
          +         2,CNTRL+1
          +         PFIPKT+2
          LA,U      A3,11
          LA,S1     A0,PFIPKT+2
          TE,U      A0,005
          J         $+5
          DL        A0,PFIPKT+2
          LDSC      A0,6
          DS        A0,PFIPKT+2
          JGD,      A3,$-6
          LA        A1,DATE
          SZ        A0
          LSSL      A1,21
          LDSL      A0,6
          SA,S6     A0,PFIPKT+11
          SZ        A0
          LDSL      A0,4
          SA,S4     A0,PFIPKT+11
          SZ        A0
          LDSL      A0,5
          SA,S5     A0,PFIPKT+11
          ER        TDATE$
          SA,H1     A0,PFIPKT+11
          LMJ       X11,START
          +         PFIPKT+2
          +         BLANK
          +         0
          SZ        A13
```

```
RCVRCD   LMJ       X11,RECVG          . RECEIVE 1 RECORD DATA
         +         RBUF               .
         LA,H1     A0,RBUF            . FETCH MESSAGE HEADER
         TE,U      A0,1               .
         J         NEXT               . IF NOT DATA MESSAGE
         LA,H2     A4,RBUF            . FETCH WORD COUNT
         ANA,U     A4,1               .
         TNE,U     A4,1
         J         RCVRCD
         LA,U      A5,RBUF+1          .
FTLINE   LMJ       X11,DEVIDE         . FETCH 1 LINE DATA
         +         0                  .
         LA        A0,BLANK
         LA,U      A1,A0
         LXI,U     A1,0
         LA,U      A2,PBUF
         LXI,U     A2,1
         LR,U      R1,IMAGESIZE
         BT        A2,,*A1
         LA        A0,FTLINE+1        .
         SZ        A13                . CLEAR TAB FLAG
         SA        A0,$+2             .
         LMJ       X11,CNVAT8         .  AMOS TO FIELDATA
         +         0                  .
         +         PBUF               .
         DS        A4,SA4
         LMJ       X11,PUTCRD
         +         PBUF
         +         0
         DL        A4,SA4
         TNE,U     A4,1
         J         RCVRCD
         J         FTLINE
NEXT     LA        A0,RBUF
         TE        A0,EOFHDR
         J         EOFSTP
         LMJ       X11,FINIS
         +         0
         J         SDAKWD
```

```
EOFSTP   .
         TE        A0,STPHDR
         J         ERR2
         LMJ       X11,CLOSEG
         LA        A0,(3,FREA)
         ER        CSF$
         ER        EXIT$
ERR1     TNE       A0,ERRHDG
         J         AGSERR
ERR2     P$RINT    (PF 2,4,MSG1)
SDNAK    LMJ       X11,SENDG
         +         1,NAKWD
CLOUT    LMJ       X11,CLOSEG
         ER        ERR$
AGSERR   P$RINT    (PF 2,3,MSG4)
         J         CLOUT
ERRASG   P$RINT    (PF 2,2,MSG2)
         J         SDNAK
ERRUSE   P$RINT    (PF 2,3,MSG3)
         J         SDNAK
'        .
$(0).
MSG1     'ΦMESSAGE FORMAT ERROR'
MSG2     'ΦCAN''T ASG,A'
MSG3     'ΦCAN''T USE $WRITE'
MSG4     'ΦADAGE ERROR'
USE      +         '@USE $'
         +         'WRITE,'
         RES       2
FREA               '@FREE,A $WRITE .'
ASG      +         '@ASG,A'
         +         'X       '
         RES       2
CNTRL    RES       3
DATE     +         0
```

```
SA4       +         0D
BLANK     +         005050505050
          +         005050505050
CTLHDR    +         0,4
ERRHDG    +         07777700002
NAMHDR    +         7,3
EOFHDR    +         2,2
STPHDR    +         3,2
PBUF      RES       27
ACKWD     +         0
NAKWD     -         0
FETCH     LA,S2     A0,CNTRL+1          . FETCH EX INSTRUCTION
          LA,S3     A0,CNTRL+1          .
          LA,S4     A0,CNTRL+1          .
          LA,S5     A0,CNTRL+1          .
          LA,S6     A0,CNTRL+1          .
          LA,S2     A0,CNTRL+2          .
          LA,S3     A0,CNTRL+2          .
          LA,S4     A0,CNTRL+2          .
          LA,S5     A0,CNTRL+2          .
          LA,S6     A0,CNTRL+2          .
STORE     SA,S2     A0,CNTRL+1          . STORE EX INSTRUCTION
          SA,S3     A0,CNTRL+1          .
          SA,S4     A0,CNTRL+1          .
          SA,S5     A0,CNTRL+1          .
          SA,S6     A0,CNTRL+1          .
          SA,S2     A0,CNTRL+2
          SA,S3     A0,CNTRL+2          .
          SA,S4     A0,CNTRL+2          .
          SA,S5     A0,CNTRL+2
          SA,S6     A0,CNTRL+2
RBUF      +         06000300000
          RES       4099
          END       ATU
```

CHARS - CHARS is to meet the needs of a user who must manipulate characters. This is a series of six (6) routines - 3 pairs. The first two pack and unpack AMOS characters; the second pair pack and unpack ASCII character; and finally, the remaining two translate from one character set to the other. The first four routines are of the type "subroutines" while the latter two are of the type "integer function." All six require or produce characters that are right justified and zero filled on the left.

USAGE - The calling sequences are symmetric, i.e., all are of the form

"NAME"

Call "NAME" (input array, size of input, output array) where "NAME" is built as follows:

The first three characters of the name on PAK or UNP for pack or unpack respectively while the remaining three are ASC or AMS for obviously, Ascii or Amos type characters. Size of input is in units of words if unpacking, and in number of characters (which equals number of words) if doing a pack.

The two integer functions AMSTASC and ASCTAMS have one calling argument - the character to be converted. The value of the function is the translated character. Characters which are in the Ascii character set but not in the Amos set are returned or question marks.

The names are from the mnemonic phrase Amos to Ascii and Ascii to Amos. Attached are listings. These routines are resident on ppvv 201 of the four-people pack.

PROGRAMMING DESCRIPTION

All of the pairs (by function) of the routines work in similar fashion. For the routines that do packing, a working word is built up a character at a time. When full (four with Ascii, five with Amos), the word is stored back into the users array via the calling sequence and another working word is started. This continues until the number of input characters is exhausted.

In the unpacking sequence, the working word is one word of the input from the calling sequence producing several output words, each containing one character right justified and zero filled.

The translate programs both use the character's collating sequence value as an index into a translate table, possibly after an offset adjustment. Characters with no correspondence are arbitrarily given the value '?'.

Printed Example

PAKASC -

```
 1:        SUBROUTINE PAKASC(IN,NCHARS,OUT)
 2:        IMPLICIT INTEGER (A-Z)
 3:        DIMENSION IN(1),OUT(1)
 4:        DIMENSION SHIFT(4)
 5:        DATA SHIFT/23,16,8,1/
 6:        DATA MSK/177B/
 7:        OWDKT=NCHARS/4
 8:        IF(OWDKT.LE.0) GOTO 500
 9:        DO 10 I=1,OWDKT
10:        K=(I-1)*4
11:        WORK=           (IN(K+1).AND.MSK).L.23
12:        WORK=WORK.OR.((IN(K+2).AND.MSK).L.16)
13:        WORK=WORK.OR.((IN(K+3).AND.MSK).L.8)
14:        WORK=WORK.OR.((IN(K+4).AND.MSK).L.1)
15:        OUT(I)=WORK
16:  10    CONTINUE
17:  500   REM=NCHARS/%4
18:        IF(REM.LE.0) RETURN
19:        WORK=0
20:        IOFF=OWDKT*4
21:        DO 20 J=1,REM
22:        K=SHIFT(J)
23:        WORK=WORK.OR.((IN(IOFF+J).AND.MSK).L.K)
24:  20    CONTINUE
25:        OUT(OWDKT+1)=WORK
26:        RETURN
27:        END
```

UNPASC

```
29:        SUBROUTINE UNPASC(IN,NWDS,OUT)
30:        IMPLICIT INTEGER (A-Z)
31:        DIMENSION IN(1),OUT(1)
32:        DO 10 I=1,NWDS
33:        J=(I-1)*4
34:        WORD=IN(I)
35:        OUT(J+1)=((WORD.R.1).AND.3760000000B).R.22
36:        OUT(J+2)=(WORD.AND.37600000B).R.16
37:        OUT(J+3)=(WORD.AND.77400B).R.8
38:        OUT(J+4)=(WORD.AND.376B).R.1
39:  10    CONTINUE
40:        RETURN
41:        END
```

UNPAMS

```
42:
43:         SUBROUTINE UNPAMS(IN,NWDS,OUT)
44:         IMPLICIT INTEGER (A-Z)
45:         DIMENSION IN(1),OUT(1)
46:         DO 10 I=1,NWDS
47:         WORD=IN(I)
48:         J=(I-1)*5
49:         OUT(J+1)=((WORD.R.1).AND.3740000000B).R.23
50:         OUT(J+2)=(WORD.AND.77000000B).R.18
51:         OUT(J+3)=(WORD.AND.770000B).R.12
52:         OUT(J+4)=(WORD.AND.7700B).R.6
53:         OUT(J+5)=WORD.AND.77B
54:  10     CONTINUE
55:         RETURN
56:         END
```

PAKAMS

```
58:         SUBROUTINE PAKAMS(IN,NCHARS,OUT)
59:         IMPLICIT INTEGER (A-Z)
60:         DIMENSION IN(1),OUT(1)
61:         DATA MSK/77B/
62:         DIMENSION SHIFT(5)
63:         DATA SHIFT/24,18,12,6,0/
64:         WDKNT=NCHARS/5
65:         IF(WDKNT.LE.0) GOTO 500
66:         DO 10 I=1,WDKNT
67:         J=(I-1)*5
68:         WORK=           (IN(J+1).AND.MSK).L.24
69:         WORK=WORK.OR.((IN(J+2).AND.MSK).L.18)
70:         WORK=WORK.OR.((IN(J+3).AND.MSK).L.12)
71:         WORK=WORK.OR.((IN(J+4).AND.MSK).L.6)
72:         OUT(I)=WORK.OR.(IN(J+5).AND.MSK)
73:  10     CONTINUE
74:  500    CONTINUE
75:         REM=NCHARS//5
76:         IF(REM.LE.0) RETURN
77:         WORK=0
78:         IOFF=WDKNT*5
79:         DO 20 J=1,REM
80:         K=SHIFT(J)
81:         WORK=WORK.OR.((IN(IOFF+J).AND.MSK).L.K)
82:  20     CONTINUE
83:         OUT(WDKNT+1)=WORK
84:         RETURN
85:         END
```

AMSTASC

```
87:       INTEGER FUNCTION AMSTASC(I)
88:       INTEGER XLATE(64)
89:       DATA XLATE/1K[,1K%,1K],1K!,1K&,1K◆,1K:,1K\,1K+,1K?,1K?,1K^
90:       1          ,1K',1K?,1K(,1K),1K0,1K1,1K2,1K3,1K4,1K5,1K6,1K7,
91:       2          ,1K8,1K9,1K;,1K=,1K,,1K-,1K.,1K/,1K ,1KA,1KB,1KC
92:       3          ,1KD,1KE,1KF,1KG,1KH,1KI,1KJ,1KK,1KL,1KM,1KN,1KO
93:       4          ,1KP,1KQ,1KR,1KS,1KT,1KU,1KV,1KW,1KX,1KY,1KZ,1K$
94:       5          ,1K#,1K?,1K",1K?/
95:       AMSTASC=((XLATE(I+1).R.1).AND.3760000000B).R.22
96:       RETURN
97:       END
```

ASCTAMS

```
99:       INTEGER FUNCTION ASCTAMS(I)
100:      INTEGER XLATE(64)
101:      DATA XLATE/' ','!','^','#','$','%','&','''','(',')',
102:      1          '◆','+',',','-','.','/','0','1','2',
103:      2          '3','4','5','6','7','8','9',':',';',
104:      3          '?','=','?','?','@','A','B','C','D',
105:      4          'E','F','G','H','I','J','K','L','M',
106:      5          'N','O','P','Q','R','S','T','U','V',
107:      6          'W','X','Y','Z','[','\',']','"','?'/
108:      J=I.AND.177B
109:      IF(J.GT.140B .AND. J.LT.173B) J=J.AND.137B
110:      ASCTAMS=XLATE(J-37B)
111:      IF(J.GT.137B .OR. J.LT.040B) ASCTAMS='?'
112:      ASCTAMS=((ASCTAMS.R.1).AND.3740000000B).R.23
113:      RETURN
114:      END
```

A very common function of computer code is moving a group of computer words from one location to another. Mover has been designed to this function.

It is used as a Fortran callable subroutine; for example, CALL MOVER(FRM,FINC,TO,TOINC,NWDS). FRM is the input array, and FINC is the increment to move through this array. Both zero and negative values are allowed. TO is the receiving array, and TOINC is the increment to add after each store. NWDS is the total number of words to transfer.

To zero out an array of 100 words:

```
CALL MOVER(0,0,A,1,100)
```

To move every 10th element of 10 X 10 array into a 10 word array:

```
CALL MOVER(BIG(1),10,SMALL,1,10)
```

The listing:

```
                SUBROUTINE MOVER(FRM,FINC,TO,TOINC,NWDS)
                IKNT=-NWDS
                IF(IKNT.GE.0) RETURN
                IPTR=@FRM
                OPTR=@TO
                IINC=FINC
                OINC=TOINC
A LP:           MDBR'I          IPTR
A               BRMD'I          OPTR
A               MDAR            IPTR
A               MDAE            IINC
A               ARMD            IPTR
A               MDAR            OPTR
A               MDAE            OINC
A               ARMD            OPTR
A               MDAR'N'X        IKNT
A               JPLS            LP
                RETURN
                END
```

An identical routine exists on the 1110 which utilizes the "block transfer" instruction. The use of it is absolutely identical. A listing follows.

```
. THIS IS A BLOCK TRANSFER PROGRAM USEING THE 1108 BT INSTRUCTION
.    FORTRAN CALLING SEQUENCE
.    CALL MOVER(FROM,INCREMENT FOR FROM ARRAY,TO,INCREMENT FOR TO ARRAY
.
.                NUMBER OF WORDS TO TRANSFER)
.
.       THIS SLOWER THAN DO-LOOP IF 5 OR LESS WORDS TO TRANSFER
.
.    EXAMPLE
.       ZERO 100 WORDS OF THE ARRAY C
.    CALL MOVER(0,0,C,1,100)
.
.       MOVE EVERY 10 WORD OF AN ARRAY A(DIMENSION(10,1000)) INTO EVERY
.          CONSECUTIVE WORD OF B, AND DO THIS FOR 1000 WORDS
.    CALL MOVER(A,10,B,1,1000)
.
$(1) AXRS .
MOVER*   L,U   A0,*0,X11 .
         LXI   A0,*1,X11 .
         L,U   A1,*2,X11 .
         LXI   A1,*3,X11 .
         L     R1,*4,X11 .
         BT    A1,,*A0 .
         J     6,X11 .
         END .
```


A series of routines have been written to assist the user in interfacing his application software with the Exec 8 file maintenance software. With these interfaces it is possible to read from and write directly into program files. The elements that are thus created may be manipulated with ED and ELT processor and are in all ways compatible with the Exec 8 system.

There are three routines in this interface package. With one, a program is involked on an Exec 8 processor, i.e., @URFILE. ABSELT name1,name2. The behavior of a program involked is similar to the ELT processor. The input stream may be an Exec 8 element or the runstream, via the "I" option. Output is optional but if desired then the second name must be present.

This routine was then sectioned out into two separate routines, one for reading and one for writing. When using either or both and then the program is involked in the normal manner for an Exec 8 user program - that is via the @XQT URFILE.ABSELT control card.

Since the involking control card of a processor contains the file and element names, this information is not required when calling the entry points of the IF program. However, since there is no such information on a XQT card, the desired file name and element name must be passed the reading (or writing) routine via coded calls within the user program. For reasons of simplicity of development, it was decided that the file to be read from would or always be named $READ and one written into would have the name $WRITE. These files must be attached to the run with an @ASG,AX card and the name of either (or both) $READ/$WRITE use-attached to the file.

Program IF

Usage is rather simple. There are four entry points to this routine, three major and one minor. The major entries are:

```
CALL SIREAD($ERR,$EOF,BUFFER,FILCHR)
```

and

```
CALL PGMOUT($ERR,$EOF,BUFFER,FILCHR)
CALL DONE ($ERR)
```

The first time either SIREAD or PGMOUT is called, the program is initialized. This must be done prior to any read from Unit 5 - the input stream device. The arguments are:

- $ERR - Fortran statement to receive control if an error is detected.
- $EOF - Fortran statement to receive control if an End-of-File is read. Note that this is a dummy argument in the output routine.
- BUFFER - This is the image that will be output or the array to receive is incoming data.
- FILCHR - Normally a 1H5 would be used here. It was originally included so that trailing zeros, rather than trailing blanks could be dropped. Its use other than a word of blanks is not encouraged.

SIREAD and PGMOUT read in a 14 word image or write a 14 image. The image size is variable at assembly time in the routine IF. The values 14 or 22 are the only recommended values.

DONE is an entry that must be called when I/O is complete to drain all buffer and update the files table of contents.

If a user elects to use the IF interface program, note that the program can only be involked as a processor.

There is one other entry:

```
CALL BEGINN($ERR,$EOF)
```

that is used to initialize the program so that unit 5 reading may take place prior to actually wanting pass an image.

Example:

A very simplified example of the use of IF. Suppose a Fortran program as follows:

```
                DIMENSION IMG(14)
      1         CALL SIREAD($99,$99,IMG,1HB)
                CALL PGMOUT($99,$99,IMG,1HB)
                GO TO 1
     99         CONTINUE
                CALL DONE($100)
    100         END
```

which was collected in TPF$. The map used was:

```
@MAP,I A,B
    IN TPF$.
  END
```

Then the card:

```
@.B AFILE.BELM,CFILE.DELM
```

does exactly the same as

```
@COPY,S AFILE.BELM,CFILE.DELM
```

IF

```
$(0)      AXR$
          .     THIS PROGRAM IS DESIGNED TO ALLOW A FORTRAN PROGRAM TO BE
          .          INVOKED AS A PROCESSOR
          .
          .     USEAGE IS
          .          CALL SIREAD($ERR,$EOF,BUFFER,FILCHR) FOR INPUT
          .        AND
          .          CALL PGMOUT($ERR,$EOF,BUFFER,FILCHR) FOR OUTPUT
          .        AND
          .          CALL DONE($ERR) FOR CLOSE-OUT
          .
          .     IF MAY ALSO BE INITIALIZED WITH THE FOLLOWING CALL
          .     CALL BEGINN($ERR,$EOF)
          .     THIS CALL DOES NOT READ AN IMAGE
          .
          .    TOTALLY REWRITTEN FOR SIGMA CORP
          .          SUMMER '75
          .
          .               AUTHOR WALTER N COLQUITT
          .
          .   REGISTERS USED AND NOT RESTORED ARE THE 'VOLATILE' SET
          .
          .
          .
PRINT*    PROC      1,2
          LA        A0,PRINT(1,1)
          ER        PRINT$
          END
          .
PCW*      PROC      1,1
          PF        1,$-PCW(1,1),PCW(1,1)
          END
          .
PF        FORM      12,6,18
BUFLEN    EQU       1792              AN EFFICIENT I/O SIZE
IMAGELENGTH EQU     14                USER'S RECIEVING LENGTH * * * IMPORTAN
T * * *
          .
INBUF     RES       2*BUFLEN          WHERE INPUT IS PLACED
OUTBUF    RES       2*BUFLEN          WHERE OUTPUT IS TAKEN FROM
PARTBL*.
 DO 14 , + 0D
PFIPKT    +         0D
          +         0D
          +         0
          +         1,0
          +         0D
          +         1,0,1
          +         0,0
          +         0
          +         0
```

```
FLAGS    +          0
SVX11    +          0                    X11 SAVE AREA
PFISAT   +          0
BGNFLG   +          0
LABEL    +          0500130,0            INITIAL 2 IMAGES IN FILE TO BE READ BY
         +          '♦SDFF♦'             SIRASM
FCTI     +          0D
         +          0
         +          0002000,0
         +          BUFLEN,0
         +          0
         +          INBUF,INBUF+BUFLEN
         +          BUFLEN/28,IMAGELENGTH
         +          1,0
         +          1,0
         +          0
FCTO     +          0D
         +          0
         +          0001000,0
         +          BUFLEN,0
         +          0
         +          OUTBUF,OUTBUF+BUFLEN
         +          BUFLEN/28,0
         +          1,0
         +          1,0
         +          0
/
         .
SI       EQUF       PARTBL+1             'USE' ATTACHED INPUT NAME
INELM    EQUF       PARTBL+3
INVER    EQUF       PARTBL+7
SILOC    EQUF       PARTBL+11            SEE PARTBL FORMAT(UP-4144,CHAPTER 9)
SO       EQUF       PARTBL+14            'USE' ATTACHED OUTPUT NAME
OUTELM   EQUF       PARTBL+16
OUTVER   EQUF       PARTBL+20
SOLOC    EQUF       PARTBL+26
ASCII    EQUF       FLAGS,,S1
NONAME   EQUF       FLAGS,,S2
STARTED  EQUF       FLAGS,,S3
IOPFLG   EQUF       FLAGS,,S4
LOPFLG   EQUF       FLAGS,,S5
OOPFLG   EQUF       FLAGS,,S6
INADDR   EQUF       FCTI+5
INBUFR   EQUF       FCTI+8,,H2
ICWI     EQUF       FCTI+10
DELETED  EQUF       FCTI+10,,S4
OUTADR   EQUF       FCTO+5
OUTBFR   EQUF       FCTO+8,,H2
ICWO     EQUF       FCTO+10
IOP      EQU        1♦/('Z'-'I')
LOP      EQU        1♦/('Z'-'L')
OOP      EQU        1♦/('Z'-'O')
ELNAM    EQUF       PFIPKT+2
ELVER    EQUF       PFIPKT+6
SLOT     EQUF       PFIPKT+9,,H2
TXTADR   EQUF       PFIPKT+10
```

```
/
$(1)
OCTAL
/
$(1)
SIREAD→
FILEIN
CONTLIMG
```

```
        .
        .
        +       0
        LA,U    A8,11
        LR,XU   R2,-7
        DL      A6,('000000000000')
        MLU     A5,A6
        SSL     A5,3
        DSC     A6,6
        JGD     A8,$-3
        DSC     A6,36
        J       ←OCTAL
        .
        .
        .   THIS IS THE MAINLINE SECTION FOR INPUT
        .
        .
        SX      X11,SVX11       SAVE X11
        SLJ     $               ONLY 1ST TIME WILL
        SLJ     INITIALIZE      THIS SLJ BE EXECUTED
        SZ      A3              ALL THESE CLEARINGS OF A3 FOR ADR=IND
        LA,U    A0,←2,X11       ADDR OF RECIEVING BUFFER
        SA      A0,INBUFR       USER'S RECIEVING ADDRESS
        TNZ     IOPFLG
        J       FILEIN          INPUT IS FROM MASS STORAGE
        LXI,U   A0,INEOF        A0 NOW PREP'ED FOR THE CARD READ
        ER      READ$
        SA      A0,ICWI         A DEBUGGING AID IF BAD READ
        LXI,U   A0,0
        J       FILOUT
        LA,U    A0,FCTI         INPUT FCT ADDRESS
        LMJ     X11,SDFI        GET SOURCE DATA FILE INPUT IMAGE
        J       INERR           MSG AND TAKE ERROR RETURN
        J       INEOF           TAKE EOF RETURN
        TP      ICWI            IS THIS A CONTROL IMAGE
        J       CONTLIMG        YES, SO SEE IF OUTPUT IS ALLOWED
        TZ      DELETED         NO. HAS IMAGE BEEN DELETED?
        J       FILEIN          YES IT HAS
        J       TESTASCI        NOT DEL., NOT LABEL IMAGE SO PRESS ON
        LA      A3,ICWI
        DSL     A3,30
        SSL     A4,6
        TNE,U   A3,040
        J       FILEIN
        TNE,U   A3,051
        J       NOCONT
        TNE,U   A3,042
        SA      A4,ASCII
        TZ      NONAME          A LABEL IMG,BUT IS OUTPUT ALLOWED?
        J       FILEIN          NO IT IS NOT
        LA      A0,ICWI         GET CONTROL WORD
        SA      A0,ICWO         USE FOR OUTPUT
        LA,U    A0,FCTO
        LMJ     X11,SDFO        WRITE OUT THE DELETED LINE
        J       OUTERR
        J       FILEIN          GET NEXT IMAGE
```

```
TESTASCI TNZ       ASCII            DO WE NEED TO CONVERT..
         J         CNTL             NO,ALREADY IN F.D.
         LA        A1,INBUFR        INPUT ADDR
         LA        A0,ICWI          CONTROL WORD
         SSL       A0,24            NOW ASCII WORD COUNT
         LA        A2,INBUFR        OUTPUT BUFFR ADDR
         LMJ       X11,ASCFD        CONVERT VIA SYS ROUTINE
         SA,S2     A0,ICWI          NEW COUNT TO CONTROL IMAGE
CNTL     LX        X11,SVX11        RESTORE X11
         LA        A0,ICWI          GET THE CONTROL WORD
         SSL       A0,24            MOVE IMAGE COUNT OVER TO THE RIGHT
FILOUT   AU        A0,INBUFR        WHERE TO BEGIN THE FILL
         LXI,U     A1,1             INCREMENT BY 1
         ANA,U     A0,IMAGELENGTH   NUMBER OF WORDS TO FILL
         SMA       A0,R1            PLACE COMPUTED REPEAT COUNT INTO R1
         SZ        A3
         LA,U      A0,*3,X11        GET THE CHARACTER TO BE AS 'FILL' CHAR
         BT        A1,,*A0          ZINNNING!!
         TNZ       LOPFLG           L-OPTION IS ON?
         J         5,X11            NO, SO RETURN
         LA        A0,(PF 1,IMAGELENGTH,0)
         LXM       A0,INBUFR        PRINT CNTL WORD NOW FORMED
         ER        PRINT$
         J         5,X11            AND RETURN TO FORTRAN CALLING PROGRAM
         .
         .
         .   HERE WE PREFORM THE OUTPUT FUNCTION
         .
PGMOUT*  SX        X11,SVX11        SAVE GOOD OLE X11
         SLJ       $                JUMPS TO $+1 IF 1ST PASS,OTHERWISE $+2
         SLJ       INITIALIZE       THIS IS DONE ONLY 1ST TIME
         SZ        A3
         LA,U      A0,*2,X11        ADDR OF IMAGE TO OUTPUT
         SA        A0,OUTBFR        PLACE INTO THE CONTROL PACKET
         LXI,U     A0,0100++IMAGELENGTH
         TZ        DOPFLG           OUTPUT PRINT FLAG ON?
         ER        PRINT$
         LA        A1,*3,X11        WORD OF CHARS TO BE DELETED
         LR,U      R1,IMAGELENGTH          .
         LXI,XU    A0,-1                   . THESE SIX LINES OF REALLY
         SNE       A1,IMAGELENGTH-1,*A0    . EXCELLENT CODE WERE DESIGNED
         LR,XU     R1,-1                   . BY MESSRS. FLOYD LINN,
         SR        R1,A0                   .  KIRK YARINA, AND
         LA,U      A0,1,A0                 .  WALT COLQUITT
         LSSL      A0,24
         SA        A0,ICWO
         LA,U      A0,FCTO          ADDR OF OUTPUT FCT
         LMJ       X11,SDFO         OUTPUT TO FILE
         J         OUTERR
         LX        X11,SVX11        RESTORE GOOD OLE X11
         J         5,X11            AND RETURN TO CALLING PROGRAM
```

```
*
*    HERE FILES ARE CLOSED AND RETURNED TO ORIGINAL STATUS.
*        THIS SUB-PROGRAM MAY NOT BE AGAIN REFERENCED
*
DONE←   SX      X11,SVX11
        LA      A0,(J NOMORE)    ERROR RETURN
        SA      A0,PGMOUT       LOCK OUT ANY MORE CALLS
        SA      A0,SIREAD           TO THIS SUB-PROGRAM
        SA      A0,DONE
        TZ      IOPFLG          WAS INPUT FROM CARDS
        J       $+3             YES, SO NO NEED TO CLOSE SDFI
        LA,U    A0,FCTI         CLOSE INPUT
        LMJ     X11,SDFIC
        LA,U    A0,FCTO         CLOSE OUTPUT
        LA      A1,NONAME
        JNZ     A1,FINOFF       OUTPUT IS NOT BEING ALLOWED
        LMJ     X11,SDFOC
        J       BADLASTWRITE
        DL      A0,SD           FILE NAME FOR PFI$ PACKET
        DS      A0,PFIPKT
        DL      A0,OUTELM
        DL      A2,OUTVER
        JNZ     A0,$+3
        DL      A0,INELM
        DL      A2,INVER
        DS      A0,ELNAM
        DS      A2,ELVER
        LA      A0,OUTADR
        ANA     A0,SDLOC
        SA      A0,SLOT         SECTOR SIZE OF THE ELEMENT
        LA      A1,OUTADR
        LN,U    A0,PFIPKT
        ER      PFI$
        SA      A2,PFISAT
FINOFF  LMJ     X11,POSTPR      CLOSE AND PROPERLY RELEASE FILES
        J       FINALERR
        LX      X11,SVX11
        J       2,X11
```

```
          .
          .
          .    INITIALIZATION OF INPUT AND OUTPUT OCCURS HERE
          .
INITIALIZE          .
          +         0
          TZ        STARTED         HERE BEFORE?
          J         *$-2            YES
          LA,U      A0,1
          SA        A0,STARTED
          LMJ       X11,PREPRM      OPEN THE FILES
          J         OPNER           FATAL ERROR - CAN'T OPEN
          LA        A0,PARTBL       OPTION LETTER MASK
          LA,U      A1,1
          TEP,U     A0,LOP
          SA        A1,LOPFLG
          TEP,U     A0,OOP          'O' OPTION ON?
          SA        A1,OOPFLG       YES,SO SET FLAG TO 'ON'
          TEP,U     A0,IOP          IS THE I OPTION ON?
          J         IOPON           YES
          LA,S3     A0,PARTBL+6
          LA,U      A1,0
          LA,U      A2,4
          TW        A1,A0
          J         BADTYPE
          SA,S3     A0,PFIPKT+5
          LA,S1     A0,PARTBL+10
          SA,S1     A0,PFIPKT+9
          DL        A0,SI           NO SO MUST BUILD THE FCTI
          DS        A0,FCTI
          LA        A0,SILOC
          SA        A0,INADDR
          LA,U      A0,FCTI
          LMJ       X11,SDFIO       OPEN INPUT FILE
          J         FTO
          LA,U      A0,FCTI
          LMJ       X11,SDFI        GET INITAL LABEL IMAGE
          J         INERR           INPUT READING ERROR
          J         INEOF           EOF AT THE VERY BEGINNING
          LA        A0,ICWI         GET CONTROL WORD
          SA        A0,ASCII        STORE AWAY ASCII FLAG
          SSL       A0,30           IMAGE TYPE TO LOWER 6
          TE,U      A0,050
          J         NO1STLAB
          TZ        SO              WAS SO FIELD NAMED?
          J         IOPON+2         YES BUILD FCTO
          LA        A0,(ER ERR$)    NO SO PREVENT OUTPUT
          SA        A0,PGMOUT
          LA,U      A0,1
          SA        A0,NONAME       DISALLOW ANY OUTPUT
          J         EXIT
```

```
IOPON    LA,U     A0,1
         SA       A0,IOPFLG
         DL       A0,SO
         DS       A0,FCTO
         LA,U     A0,LABEL+1
         SA       A0,OUTBFR
         LA       A0,LABEL
         SA       A0,ICWO
         LA,U     A0,SO
         ER       PFWL$
         SA       A1,OUTADR
         SA       A1,TXTADR
         LA,U     A0,FCTO
         TNZ      NONAME
         LMJ      X11,SDFOO       OPEN OUTPUT VIA SDFOO
         LA,U     A0,FCTO
         LA       A1,NONAME
         JNZ      A1,EXIT
         LMJ      X11,SDFO        WRITE THE LABEL IMAGE NEEDED BY SIRASM
         J        FTWL
EXIT     LX       X11,SVX11
         TNZ      BGNFLG
         J        *INITIALIZE
         LX       X11,BGNFLG
         SZ       BGNFLG
         J        3,X11
BEGINN*  SX       X11,BGNFLG
         J        INITIALIZE+3
         .
         .    THE ERROR HANDLING IS HANDLED IN THIS SECTION
         .
OUTERR   LX       X11,SVX11
         PRINT    OUTERMSG
         J        0,X11

INERR    LX       X11,SVX11
         PRINT    INERRMSG
         J        0,X11
         .
INEOF    LX       X11,SVX11
         J        1,X11
         .
BADLASTWRITE .
         PRINT    LSTMSG
         J        FINOFF
         .
FINALERR PRINT    DNER
         LX       X11,SVX11
         J        0,X11
         .
OPNER    PRINT    NOSTRT
         ER       ERR$
NOMORE   PRINT    NOTAGN
         ER       ERR$
FTO      SLJ      OCTAL
         DS       A6,ST1
         PRINT    NOOPEN
         ER       ERR$
```

```
FTWL      SLJ       OCTAL
          DS        A6,ST2
          PRINT     NOLABEL
          ER        ERR$
BADTYPE   PRINT     WRNGTYPE
          ER        ERR$
NOCONT    PRINT     CANTCONT
          ER        ERR$
NO1STLAB  PRINT     FIRSTLAB
          ER        ERR$
          .
          .              .   MESSAGE STACK
          .
$(0)      .
MSG1      '◆IF◆ ERROR READING FROM THE INPUT FILE'
INERRMSG  PCW       MSG1
          .
MSG2      '◆IF◆ ERROR WHILE WRITING TO OUTPUT FILE'
OUTERMSG  PCW       MSG2
          .
MSG3      '◆IF◆ FINAL WRITE HAD AN ERROR'
LSTMSG    PCW       MSG3
          .
MSG4      '◆IF◆ ERROR IN CLOSING FILES'
DNER      PCW       MSG4
MSG5      '◆IF◆ THE INTERFACE SUBPROGRAM MAY NOT BE CALLED';
          ' AGAIN AFTER CALLING ''DONE'' '
NOTAGN    PCW       MSG5
          .
MSG6      '◆IF◆ FAILED TO INITALLY OPEN VIA PREPRM'
NOSTRT    PCW       MSG6
MSG7      '◆IF◆ FAILED TO OPEN INPUT FILE.ELEMENT'
ST1       RES       2
NOOPEN    PCW       MSG7
MSG8      '◆IF◆ FAILED TO WRITE INITIAL LABEL IMAGE'
ST2       RES       2
NOLABEL   PCW       MSG8
MSG9      '◆IF◆ INPUT ELEMENT IS NOT SYMBOLIC'
WRNGTYPE  PCW       MSG9
MSG10     '◆IF◆ CAN''T HANDLE CONTINUATION IMAGES'
CANTCONT  PCW       MSG10
MSG11     '◆IF◆ ELEMENT FORMAT ERROR-1ST IMG IS NOT A LABEL IMAGE'
FIRSTLAB  PCW       MSG11
          END
```

READALL and WRITEALL have two modes of operation where in one mode images are passed back to the calling routine one line at a time. In the other mode, an entire element is loaded into memory with but one call.

Both of these subprograms were developed from the coding of "IF." They are to serve the purpose of allowing program execution without using the "PARTBL" and yet still allowing direct access to Exec 8 program files for data storage.

```
0:>P!
$(0)      AXR$
          .
          .  THIS FORTRAN CALLABLE SUBROUTINE WILL READ IN ALL OF
          .    AN ELEMENT FROM A FILE WHICH HAS THE 'USE' ATTACHED
          .    NAME OF '$READ$'. ALL IMAGES ARE EXPECTED TO BE 14 WDS
          .    LONG,THROUGH AN ASSEMBLY TIME PARAMETER
          .
          .  CONTRACTOR:  SIGMA CORPORATION
          .
          .  AUTHOR:      WALTER N COLQUITT
          .
          .  DATE  :      SPRING 1975
          .
          .    FORTRAN LINKING SEQUENCE
          .      FIRST: @@USE $READ,UR-FILE-NAME    ;   OR VIA NERTRAN ;
          .             OR JUST AN @USE $READ,UR-FILE-NAME PRIOR TO
          .             INVOLKING THE PROGRAM
          .        THEN,IN THE EXECUTABLE CODE:
          .
          .              CALL READAL('EN','VN',BUFFER,KRDKNT)
          .                NOTE 'EN' & 'VN' ARE 12 CHARS(2 WORDS) LONG
          .
          .         'EN':12 CHAR ELEMENT NAME THAT IS TO BE READ
          .
          .         'VN':12 CHAR VERSION-2WDS OF ZERO ALLOW ANY VERSION
          .
          .       BUFFER:THE AREA TO RECIEVE THE 14 WORD CARD IMAGES
          .
          .
          .       KRDKNT: AN ACTUAL COUNT OF THE CARD IMAGES PLACED
          .               IN THE BUFFR
          .
          .
          .    THE LINE-AT-A-TIME ENTRIES ARE
          .
          .    CALL BEGIN('EN','VN') - VN=0D OR @@@@@(12TIMES) WILL CAUSE
          .                            VERSION NAME TO BE IGNORED
          .
          .    CALL GETCRD(BUF,$EOF)
          .
          .    CALL END
          .
          .     NEARLY ALL OF THIS CODE WAS DEVELOPRD FROM THE INTERFACE
          .     PROGRAM CALLED 'IF'
          .
          .
          .
          .
          .
          .    PRINT◆   PROC      1,2              . THESE PROCS STOLEN FROM PREPRM
          .  LA          A0,PRINT(1,1)
          .  ER          PRINT$
          .  END
          .
```

```
PCW*      PROC      1,1
          PF        1,$-PCW(1,1),PCW(1,1)
          END
          .
PF        FORM      12,6,18
/         .
IMAGELENGTH EQU     14              USERS RECIEVING LENGTH
BUFLEN    EQU       1792
INBUF     RES       2*BUFLEN
FLAGS     RES       1
SX11      RES       1
FCTI      +         'SREAD '        FILE CONTROL TABEL FOR INPUT
          +         '      '
          +         0
          +         0002000,0
          +         BUFLEN,0
          +         0
          +         INBUF,INBUF+BUFLEN
          +         BUFLEN/28,IMAGELENGTH
          +         1,0
          +         1,0
          +         0
PFSPKT    +         'SREAD '        USE-ATTACHED FILE NAME
          +         '      '
          +         0D              ELEMENT NAME
          +         0
          +         1,0
          +         0D              VERSION NAME
          +         0D
          +         0               TEXT ADDRESS
          +         0
BLANK     +         '      '
FLAG      +         0
ELEMNAME  EQU       PFSPKT+2
FLAGWORD  EQU       PFSPKT+5
VERNAME   EQU       PFSPKT+6
TEXTADDR  EQU       PFSPKT+10
DISKADDR  EQU       FCTI+5
COREADDR  EQUF      FCTI+8,,H2
ICWI      EQU       FCTI+10
DELIMAGE  EQUF      ICWI,,S4
ASCIBIT   EQUF      FLAG,,S1
ONELINE   EQUF      FLAG,,S2
STARTED   EQUF      FLAG,,S3
/         .
$(1)      .
```

```
READAL*  .
                TZ          ONELINE
         J      ERRAMES
         SX     X11,SX11
         SZ     A3                  ALLOW FORTRAN (ADR=IND)
         .
         .  BUILD THE PFS$ PACKET
         .
         DL     A0,*0,X11           GET ELEMENT NAME
         DS     A0,ELEMNAME         PLACE INTO PACKET
         DL     A0,*1,X11           VERSION NAME
         DS     A0,VERNAME          TO PACKET
         TNZ    ONELINE
         SZ     *3,X11              CLEAR CARD COUNT
         .
         .  BEGIN THE SEARCH THRU THE TOC

PFS      LA,U   A0,PFSPKT           LOOK FOR ONE
         ER     PFS$
         TZ     A2
         J      LOOKERR
         TP     FLAGWORD
         J      PFS
         .
         .  NOW BUILD THE INPUT FCT AND INITIALIZE INPUT
         .
         LA     A0,TEXTADDR         EXTERNAL DEVICE ADDRESS
         SA     A0,DISKADDR         TO THE I/O PACKET
         TZ     ONELINE
         J      3,X11
         LA,U   A0,*2,X11           CORE LOCATION
         SA     A0,COREADDR         INTO THE PACKET
OPEN     .
         LA,U   A0,FCTI
         LMJ    X11,SDFIO
         J      OPENERR
         LA,U   A0,FCTI
         LMJ    X11,SDFI
         J      READERR
         J      NOLABEL
         LA     A0,ICWI
         SA     A0,ASCIBIT
         SSL    A0,30
         TE,U   A0,050
         J      NOLABEL
```

```
.
.  GET IMAGES,ONE AT A TIME,AND STORE AWAY AT FULL LENGTH
.
GETONE  LA,U    A0,FCTI
        LMJ     X11,SDFI
        J       READERR
        J       EOF             ELEMENT NOW FULLY READ IN
        TN      ICWI            DID WE READ A CONTROL IMAGE
        J       CHECK           NO,SO GO DO REGULAR THING
        LA      A0,ICWI         GET THE CNTL WORD FOR THIS IMAGE
        DSL     A0,30           TYPE OF CONTROL IMAGE TO S1
        $SL     A1,6            VALUE OF SWITCH TO S1 OF NEXT A-REG
        TNE,U   A0,051
        ER      ABORT$
        TNE,U   A0,042          IS THIS AN ASCII/FD SWITCH?
        SA      A1,ASCIBIT      YES,IT IS
        J       GETONE
.
.  SKIP IF IMAGE MARKED AS DELETED
.
CHECK   TZ      DELIMAGE
        J       GETONE
.
.  CONVERT TO FIELDATA IF IN ASCII MODE
.
        TNZ     ASCIBIT         ARE WE IN ASCII MODE?
        J       PASIMG          NO
        LA      A1,COREADDR
        LA      A0,ICWI
        $SL     A0,24
        SA      A1,A2
        LMJ     X11,ASCFD
        SA,S2   A0,ICWI
.
.  BLANK FILL IMAGE IF NEEDED
.
PASIMG  LA      A0,ICWI
        $SL     A0,24
        AU      A0,COREADDR
        LXI,U   A1,1
        ANA,U   A0,IMAGELENGTH
        $MA     A0,R1
        LA,U    A0,BLANK
        BT      A1,,*A0
                                LX      X11,SX11
        TZ      ONELINE
        J       3,X11
        LA      A1,*3,X11
        AA,U    A1,1
        SA      A1,*3,X11
        LA      A0,COREADDR
        AA,U    A0,IMAGELENGTH
        SA      A0,COREADDR
        J       GETONE
.
```

```
          .  ALL IMAGES HAVE BEEN READ IN,SO CLOSE OUT
          .
EOF       LA,U      A0,FCTI
          LMJ       X11,SDFIC
          LA        A0,SX11
          J         5,A0
          .
          .   OPEN ONE LINE AT A TIME ?
          .
BEGIN◆    LA,U      A0,1
          SA        A0,ONELINE
          J         READAL+3
          .
EXIT      SZ        ONELINE
          SZ        STARTED
          LA        A0,(J EOF)
          SA        A0,GETONE+3
          LA,U      A0,FCTI
          LMJ       X11,SDFIC
          LX        X11,SX11
          J         1,X11          $EOF RETURN
          .
          .  CLOSE OUTPUT AND CLEAR ONE-LINE FLAGS ?
          .
END◆      TNZ       ONELINE
          J         1,X11
          SZ        ONELINE
          SZ        STARTED
          SX        X11,SX11
          LA        A0,(J EOF)
          SA        A0,GETONE+3
          LA,U      A0,FCTI
          LMJ       X11,SDFIC
          LA        A0,SX11
          J         1,A0
          .
          .  RETURN THE NEXT CARD IMAGE
          .
GETCRD◆   TNZ       ONELINE
          ER        ERR$
          LA,U      A0,◆0,X11
          SA        A0,COREADDR
          SX        X11,SX11
                                TZ        STARTED
                                                  J         GETON

          LA,U      A0,1
          SA        STARTED
          LA        A0,(J EXIT)
          SA        A0,GETONE+3
          J         OPEN
          .
          .
```

```
$(0)
OPENERR  PRINT CANTOPEN
         ER        ERR$
MSG1     'FAILED ON ATTEMPT TO OPEN I/O'
CANTOPEN PCW       MSG1
READERR  PRINT     FATALREADERR
         ER        ERR$
MSG2     'A FATAL READ ERROR HAS OCCURED-ABORTING'
FATALREADERR       PCW          MSG2
LOOKERR  PRINT     NOFIND
         DL        A0,PFSPKT+2
         DS        A0,E
         DL        A0,PFSPKT+6
         DS        A0,V
         PRINT     IDIT
         ER        ERR$
MSG31    'TRYING TO FIND ELEMENT:'''
E        +         0D
         +         '    ''
V        +         0D
                              +        ''''
IDIT     PCW       MSG31
MSG3 'NO ELEMENT OF REQUIRED NAME,I/O ERROR,OR ''$READ'' NOT ATTACHED'
NOFIND   PCW       MSG3
NOLABEL  PRINT     NOLAB
         ER        ERR$
MSG4     'ELEMENT NOT PROPERLY FORMATTED-ABORTING'
NOLAB    PCW       MSG4
ERRAMES  PRINT     OUCH
         ER        ERR$
MSG5 'CAN''T HAVE ALL-AT-ONCE AND LINE-AT-A-TIME OPEN SIMULTANEOUSLY'
OUCH     PCW       MSG5
                              END
```

WRITEALL

```
$(0)    AXR$
.
.   THIS FORTRAN CALLABLE SUBROUTINE WILL WRITE OUT TO AN EXEC 8
.   PROGRAM FILE AN ELEMENT,EITHER ALL AT ONCE OR
.   ONE LINE-AT-A-TIME.
.
.   IMAGES TO BE WRITTEN OUT MUST BE 14 WORDS LONG AND BLANK
.   FILLED OUT TO THE RIGHT,IF NECESSARY
.
.   IN THE WRITE IT ALL OUT AT ONCE MODE THE NUMBER OF IMAGES TO
.   BE WRITTEN MUST BE PASSED TO THIS ROUTINE
.
.
.   CONTRACTOR:  SIGMA CORP
.
.   AUTHOR:      WALTER N COLQUITT
.
.                DATE:        SPRING 1975
.
.   FORTRAN LINKING SEQUENCE
.      FIRST: @@USE $WRITE,UR-FILE-NAME ; OR VIA NERTRN
.      THEN IN THE EXECUTABLE CODE
.
.   CALL WRTALL('EN','VN',BUF,#-OF-CARDS)
.    WHERE
.    'EN' IS TO BE THE ELEMENT NAME
.    'VN' WILL BE THE VERSION NAME; 0 IS THE SAME AS BLANKS
.
.
.    NOTE THAT EN AND VN ARE TWO(2) WORD ITEMS
.
.
.    FOR THE LINE-AT-A-TIME METHOD
.     CALL START('EN','VN')
.
.     CALL PUTCRD(BUF)
.
.      AND FINALLY
.     CALL FINIS
.
.
.
.
```

```
PRINT◆    PROC      1,2
          LA        A0,PRINT(1,1)
          ER        PRINT$
          END
          .
PCW◆      PROC      1,1
          PF        1,$-PCW(1,1),PCW(1,1)
          END
          .
PF        FORM      12,6,18
          .
IMAGELENGTH EQU     14
BUFLEN    EQU       1792
OUTBUF    RES       2◆BUFLEN
SX11      RES       1
REGSAV    RES       2
FLAGS     +         0
FCTO      +         'SWRITE'
          +         '      '
          +         0
          +         0001000,0
          +         BUFLEN,0
          +         0
          +         OUTBUF,OUTBUF+BUFLEN
          +         BUFLEN/28,0
          +         1,0
                                +         1,0
          +         0
LABEL     +         0500130,0
          +         '◆SDFF◆'
PFIPKT    +         'SWRITE'
          +         '      '
          +         0D
          +         0
          +         1,0
          +         0D
          +         5,0,1
          +         0,0
          +         0
                              +         0
IMAGEADR  +         0
CARDKNT   +         0
MASK      +         01777001013777777700
SINGLE    EQUF      FLAGS,,S1
```

```
$(1)    .
OCTAL   +       0
                                LR,U        R1,11
        LR,XU   R2,-7
        DL      A3,('000000000000')
        MLU     A2,A3
        SSL     A2,3
        DSC     A3,6
                                JGD         R1,$-3
        DSC     A3,36
        J       *OCTAL
        .
            $(1)    .
START*  .
        LA,U    A0,1
        SA      A0,SINGLE
        SZ      CARDKNT
        J       $+3
WRTALL* .
        TZ      SINGLE
        J       BUSY
        SX      X11,SX11
        SZ      A3
        .
        .
        .
        DL      A1,*0,X11
        DS      A1,PFIPKT+2
                                SLJ         CHARCHK
        DL      A1,*1,X11
        DS      A1,PFIPKT+6
        SLJ     VERCHK
        TZ      SINGLE
        J       PFI
                                LA,U        A0,*2,X11
        SA      A0,IMAGEADR
        LA      A0,*3,X11
        SA      A0,CARDKNT
```

```
PFI     LH,U    A0,PFIPKT
        ER      PFWL$
        TZ      A2
        J       BADPFIWL
                                        SA      A1,FCTO+5
        SA      A1,PFIPKT+10
        LA,U    A0,LABEL+1
        SA      A0,FCTO+8,,H2
        LA      A0,LABEL
        SA      A0,FCTO+10
        LA,U    A0,FCTO
        LMJ     X11,SDFOO
        LA,U    A0,FCTO
        LMJ     X11,SDFO
        J       OUTERR
        LX      X11,SX11
        TZ      SINGLE
        J       3,X11
        .
        .
        .
        DS      A6,REGSAV
        LA      A6,IMAGEADR
        LA      A7,CARDKNT
        J       JGD
        .
NXTLNE  SA      A6,FCTO+8,,H2
        LA      A1,(' ')
        LR,U    R1,IMAGELENGTH
        LXI,XU  A0,-1
        SNE     A1,IMAGELENGTH-1,*A0
        LR,XU   R1,-1
        SR      R1,A0
        LA,U    A0,1,A0
        LSSL    A0,24
        SA      A0,FCTO+10
        LA,U    A0,FCTO
        LMJ     X11,SDFO
        J       OUTERR
        LX      X11,SX11
        TZ      SINGLE
        J       2,X11
        AA,U    A6,IMAGELENGTH
JGD     JGD     A7,NXTLNE
        LA,U    A0,FCTO
        LMJ     X11,SDFOC
        J       OUTERR
        LA      A0,FCTO+5
        SA      A0,A1
        ANA     A0,PFIPKT+10
        SA      A0,PFIPKT+9,,H2
        LN,U    A0,PFIPKT
        ER      PFI$
        TZ      A2
        J       PFIBAD
```

```
RETURN   LX       X11,SX11
         TNZ      SINGLE
         J        $+3
         SZ       SINGLE
         J        1,X11
         .
         DL       A6,REGSAV
         J        5,X11
         .
         .
         .
PUTCRD+  TNZ      SINGLE
         J        NOTREADY
         SZ       A3
         LA,U     A0,+0,X11
         SA       A0,FCTO+8,,H2
         SX       X11,SX11
         J        NXTLNE+1
         .
FINIS+   SX       X11,SX11
         TNZ      SINGLE
         J        NOTREADY
         J        JGD+1
         .
VERCHK   +        0
         DJZ      A1,+$-1
         LA       A3,VERCHK
         SA,H2    A3,CHARCHK
         J        $+2
                       MCHARCHK   +          0
         LA,U     A5,11
J12      SZ       A0
         DL       A3,MASK
         LDSL     A0,6
         SSL      A1,6
         LDSL     A1,6
         DSC      A3,,A0
         JNB      A4,RSTBLKS
         JGD      A5,J12
         J        +CHARCHK
RSTBLKS  TE,U     A0,005
         J        NOTFINE
         SZ       A0
         LDSL     A0,6
         SSL      A1,6
         LDSL     A1,6
         JGD      A5,RSTBLKS
         J        +CHARCHK
```

```
/           .
            .
            .
NOTREADY    PRINT       NOSTART
            ER          ERR$
MSG0        'CAN''T ''PUTCRD'' OR ''FINIS'' UNLESS START HAS BEEN CALLED'
NOSTART     PCW         MSG0
BUSY        PRINT       NOTBOTH
            ER          ERR$
MSG1        'CAN''T DO WRTALL IF LINE-AT-A-TIME IS OPEN'
NOTBOTH     PCW         MSG1
BADPFIWL    .
            SLJ         OCTAL
                                -           DS          A3,STAT
            PRINT       NOOPEN
            ER          ERR$
MSG2        'CAN''T ACCESS FILE;F.N. ''$WRITE'' ATTACHED?'
STAT        RES         2
NOOPEN      PCW         MSG2
OUTERR      PRINT       IOERR
            ER          ERR$
MSG3        'FATAL I/O ERROR HAS OCCURED-ABORTING'
IOERR       PCW         MSG3
PFIBAD      .
            SLJ         OCTAL
            DS          A3,STATUS
            PRINT       PFINFG
            J           RETURN
MSG4        'PFI$(NAME TO TOC) FAILED-CONTINUING-STATUS WAS'
STATUS      RES         2
PFINFG      PCW         MSG4
NOTFINE     PRINT       BADNAME
            ER          ERR$
MSG5        'ILLEGAL CHARACTER IN A NAME'
BADNAME     PCW         MSG5
            END
```

This multiple key sort is Batcher's sort from Kunth's Volume III. The calling sequence is:

```
CALL BATCHR(DATA,LNG,NUM,KEY,KEYLNG)
```

DATA — A 2-D array of records to be sorted.

LNG — The length (in words) of each record.

NUM — The number of records in the array.

Note this means that DATA is dimensioned as follows:

```
DIMENSION DATA (LNG,NUM).
```

KEY — A singlely dimensional array that contains from major to minor order the relative word positions of the keys within the record. Note that all values in the array KEY must be between 1 and LNG.

KEYLNG — The size (in words) of the array KEY.

Example: To sort an array of 100 records, each 13 words long when the major key is word #12 and the minor key is word #2, the following code is illustrative:

```
KEY(1)=12

KEY(2)=2

CALL BATCHR(DATA,13,100,KEY,2)
```

BATCHER (ADAGE VERSION)

```
C.....
      SUBROUTINE BATCHR (DATA,LNG,NUM,KEY,KEYLNG)
      IMPLICIT INTEGER (A-Z)
      DIMENSION DATA(LNG,NUM), KEY(KEYLNG)
      IF(NUM.LT.2) RETURN
      IF(KEYLNG.LT.1 .OR. KEYLNG.GT.LNG) RETURN
      S=1
10    S=S*2
      IF(S.LT.NUM) GOTO 10
      S=S/2
      P=S
20    Q=S
      R=0
      D=P
30    LIM=NUM-D-1
      DO 40 I=0,LIM
      IF((I.AND.P).NE.R) GOTO 40
      IP1=I+1
      EXP=I+D+1
      DO 37 J=1,KEYLNG
      KEYH=KEY(J)
      IF(DATA(KEYH,IP1)-DATA(KEYH,EXP)) 40,37,38
37    CONTINUE
      GOTO 40
38    CONTINUE
      DO 39 J=1,LNG
      TMP=DATA(J,IP1)
      DATA(J,IP1)=DATA(J,EXP)
      DATA(J,EXP)=TMP
39    CONTINUE
40    CONTINUE
      IF(Q.EQ.P) GOTO 60
      D=Q-P
      Q=Q/2
      R=P
      GOTO 30
60    P=P/2
      IF(P.GT.0) GOTO 20
      RETURN
      END
```

REFERENCES

1. Anon, "Communication Control Subroutines." (Private Document).

2. Knuth, D. E., "The Art of Computer Programming," Volume 3 Sorting and Searching. Addison-Westley. 1973.

3. Anon, "NUK U1100-AGS Attachment Software, Summary of Adage Modifications, Revision A." Adage Incorporated.

4. Anon, "Univac 1100 Assembler, UP4040R4."

5. Anon, "Univac PRM, UP-4144R3." Chapters 9 and 24.