SYNTAX VERIFIER, PUT, GET, AND BGI 11-18-66 (STV,27)

```
/SYNTAX VERIFIER VARIABLES
SUA=BTEM+6
SUD=BTEM+7
SUE=BTEM+10
SUF=BTEM+11
SUG=BTEM+12
SUK=BTEM+13
SUP=BTEM+14
SUQ=BTEM+15
SUR=BTEM+16
SUS=BTEM+17
SUT=BTEM+20
```

```
/STV IOT AND SUBR
STV,      LAW IR1
.STV,     LIO I (USERAC)
.STVJ,    DIO D
          DAP I (BTEM)
           LAC I (USERPC)
           DAC I (BTEM+1)
           LAC I (FSA)
          DAC G        /TEXT PTR
           DAC I (SUD)
          LAC BO
           DAC I (SUT) /MULTIPLIER
           LAC I (SUQ)
           DAC I (SUK)
           DAC C       /OUTPUT PTR
           LAC I (SUP)
           DAC I (SUF) /PUSHDOWN PTR

/MAIN LOOP FOR INTERPRETATION
STVML,    LIO I D      /GET PSEUDO INSTR
          ISP I (SUT)
           IDX D       /IDX PC CONTINGENT ON MULTIPLIER
          DIO E       /SAVE CURRENT PSEUDO INSTR
          CLA
           RCL 3S
           ADD (LAC STVDTB)
           DAC A
B2,        XCT A       /CONSTANT 100000. DISPATCH LOOKUP
          SZL I
           IDA         /ADD 1 IF IN WIN MODE
           DAC A
           CLA"U"SWP"U"CLF 7          /CLEAR ALL FLAGS EXCEPT LINK
           RCL 3S
           IFI         /SET FLAGS 4,5,6 FROM BITS 3,4,5
           LIO E
           JMP A       /DISPATCH JMP

/IDX OPCODE
STV44,    DZM I (BTEM+5) /GROUP COUNTER
           JSP STVTRA  /GET TEXT PTR INTO E
          LAC G
           DAC I (BTEM+4) /SPACE FOR RESTORING
          DAC I (BTEM+2) /FARTHEST SUCCESSFUL SO FAR
STV4A,    JSP STVLE   /LCH I E; DAC A; SKIP ON NOT EOM
          JMP STV4C   /EOM; SUCCEEDED
STV4A1,   LAC G
           SAD I (BTEM+2)
           STF 1       /THIS WILL BE A NEW RECORD IF IT WINS
          JSP ATM     /LCH I G; DAC F
          LAC A
           SAD F
           JMP STV4A   /CHAR.S AGREE, LOOP
STV4B,    JSP STVLE   /FAILURE, WAIT FOR EOM
          JMP STV4D   /FOUND, TRY NEXT CASE
          JMP STV4B
```

```
STV4C,    SZF I 1       /SUCCESS; IS THIS A NEW RECORS?
          JMP STV4D     /NO, IGNORE
          STF 2         /SET WIN FLAG
          LAC G
          DAC I (BTEM+2) /REMEMBER HIS TEXT PTR
          LAC I (BTEM+5)
          DAC I (BTEM+3) /REMEMBER HIS GROUP
STV4D,    LAC I (BTEM+4) /ROUTINE TO TRY NEXT CASE
          DAC G
          CLF 1
          JSP STVLE
          JMP STV4D1    /2 EOM'S IN A ROW; NEW GROUP
          JMP STV4A1    /GO INTO CHAR. LOOP
STV4D1,   IDX I (BTEM+5)
          JSP STVLE
          JMP STV4E     /3 EOM'S IN A ROW; FINISHED
          JMP STV4A1

STV4E,    SZF I 2       /WAS THERE A WINNER?
          JMP STVLOS
          LAC I (BTEM+2) /YES, REMEMBER HIS PTR
          DAC G
          LAC I (BTEM+3) /SAVE HIS GROUP #
STV14,    DAC I C
          IDX C
STV4,     JMP STVML     /ALL DONE; ALSO LOSE MODE
          SZF 4
          JMP STV44     /IDX OPCODE
          LAI           /?
          SCM"U"IDA     /ADD OPCODE, CALCULATE MULTIPLIER
          DAC I (SUT)
STV0,     JMP STVML
          JSP ATM
          JSP REATM     /CHECK CHAR. AGAINST ATOM TABLE
          JMP STVML     /OK
STVLOS,   CLL"U"CML     /NG, ENTER LOSE MODE
          LAC BO
          DAC I (SUT)   /CLOBBER MULTIPLIER
STV5,     JMP STVML
STV5L,    JSP ATM
          JSP REATM     /CHECK CHAR. AGAINST ATOM TABLE
          JMP STV5L     /OK, KEEP CYCLING
STV5UL,   LAC I (SUD)
          IDC
          DAC E
STV5UM,   LAW I 1
          ADD G
          IDC
          IDC
          DAC G
          IDC
          SAD E
          JMP STVML
          LCH G
          SAD ML6
          JMP STV5UM
```

```
STV1,     JMP STVML
          JSP STVTRA  /GET TEXT PTR
          LAC G
          DAC I (FSA) /UPDATE FSA SO IT CAN BE REFERRED TO
          LAC I E     /GET ARG
          SZF 4
          JMP STV14   /DCH OPCODE; SAVE ARG
STV1L,    JSP STVLE
           JMP STVML  /EOM, XCT OPCODE SUCCEEDED
          JSP ATM
          LAC A
          SAD F
          JMP STV1L   /KEEP INSISTING ON EXACT MATCH
          JMP STVLOS  /NO MATCH; FAIL

STV2,     JMP STV2L   /LAC IN LOSE MODE
          LAW 2       /LAC IN WIN MODE
          AND E
          SZA
          JMP STV2SP  /SUPER POP
STVPOP,   LAC I (SUF) /NORMAL POP
          SAD I (SUP)
          JMP STVNPP  /TOP LEVEL, CAN'T POP
          SUB C3
          DAC I (SUF) /UPDATE PUSHDOWN PTR
          DAC A       /TEMPORARY PUSHDOWN PTR
          LAC I A
          DAC I (SUD) /POP SUD
          IDX A
          LAC I A
          CLI         /PREPARE TO UNPACK POP SUT;SUK; PSEUDO PC
          SCR 6S
          SZL
          CLA         /NO MULTIPLIER IN LOSE MODE
          XOR B0
          DAC I (SUT)
          RIL 6S
          IDX A
          LAC I A
          RCL 6S
          DIO I (SUK)
          RAR 6S      /PSEUDO PC TO AC
          JMP STV7I   /PSEUDO GO
```

```
STVNPP,   LAC I (BTEM)
          SZL
          JMP STVRUR  /GIVE R1 ON CAN'T LOSE POP
          JSP ATM
          LAC F
          SAS ML4
          JMP STVWL   /NOT EOM, LOSE THEN LAC
STV2SP,   IDX I (BTEM)   /GIVE R2
STVRUR,   DAP H        /SUBR TO RESET USER'S REGISTERS
          LAC C
          DAC I (SUG) /OUTPUT PTR
          LAC D
          DAC I (SUA) /PSEUDO PC
          LAC G
          DAC I (FSA) /TEXT PTR
          LAC I (BTEM+1)
          DAC I (USERPC) /REAL PC (SOMETIMES REDUNDANT)
          JMP HEXIT

/DON'T LAW IN LOSE MODE
STV7L,    SZF I 4
          JMP STV7II
          DZM E        /LOOK LIKE LAC 0
/WIN ENTRY TO LAC IN LOSE MODE
STVWL,    CLL"U"CML    /ENTER LOSE MODE
STV2L,    LAC I (SUK) /RESTORE PTRS TO BEG OF LEVEL
          DAC C
          LAC I (SUD)
          DAC G
          LAC E
          SAD PUD      /(LAC H)
          JMP STV2SP
          LAW 5
          AND E
          SAS B17      /STAY IN LOSE MODE ON LAC 1
          CLL         /ENTER WIN MODE ON LAC 0; LAC 5
          LIO B17
          DIO E        /LOOK LIKE LAC 1 ON RECYCLE (LAC 5)
          SZA
          JMP STVPOP  /POP ON LAC 1; LAC 5
```

```
STV3,     JMP STVML
          LAC I (SUF) /PUSH OPCODE (DIP)
          SAD I (SUE)
          JMP STVLOS  /CAN'T PUSH, LOSE
          DAC A       /TEMPORARY PUSHDOWN PTR
          LAC I (SUD) /PUSH SUD
          DAC I A
          IDX A
          LAC I (SUT) /PACK PUSH SUT; SUK; PSEUDO PC
          LIO I (SUK)
          RIL 6S
          RCL 6S
          DAC I A
          IDX A
          LAC D
          RCL 6S
          RAR 6S
          DAC I A
          IDX A
          DAC I (SUF) /UPDATE PUSHDOWN PTR
          LAC C
          DAC I (SUK) /UPDATE "BEGINNING OF LEVEL" PTRS
          LAC G
          DAC I (SUD)
          LAC B0
          DAC I (SUT) /CLOBBER MULTIPLIER
          JMP STV7II  /NOW DO A "LAW" OPCODE

STV7,     JMP STV7L
STV7II,   JSP STVTRA
          LAC E
STV7I,    DZM D
          DAP D
STV6,     JMP STVML
          JSP STVTRA
          JSP STVRUR
          LAC E
          JMP GO

/STVL AND STVW IOTS
STVL,     CML
          LAC B0
          DAC I (SUT)
STVW,     LAC I (FSA) /UNRESTORE USER REGISTERS
          DAC G
          LAC I (SUG)
          DAC C
          LAC I (SUA)
          JMP STV7I
```

```
/ATOM TABLE SUBROUTINES
/ENTRY TO GET A CHAR. AND SKIP ALL IRRELEVANT (SUS) CHAR.S
ATM,      SUB B17
          DAP H
          LIO I (SUS) /PUT MASK IN IO
          LCH I G
          SAD ML6
          JMP ATMW    /WARNING
ATMRO,    DAC F       /SAVE CHAR.
          RAL 6S
ATMI,     ADD (LAC ATOM)
          DAC B
          XCT B       /LOOK UP ATOM TABLE ENTRY
          SZA I
          JMP XHEXIT  /R2, EOM ALWAYS FAILS
ATMWC,    DIO B       /SAVE MASK
          RIL 3S      /PUT CLASS BIT IN SIGN OF IO
          XAI
          SPA
          JMP ATMD    /DIFFERENT CLASSES
          XAI         /RESTORE AC
          AND B
          AND MR14
          SZA I
XHEXIT,   IDX H       /NO BITS THE SAME, FAIL (R2)
          JMP HEXIT   /OK, R1

ATMD,     RIL 1S      /MOVE "OTHER CLASS" BIT INTO SIGN OF IO
          SPI I
          IDX H
          JMP HEXIT

ATMW,     LCH I G
          SAD ML4
          JMP ATMRO   /TREAT RUBOUT LIKE EOM
          RAR 6S
          IOR ML6
          DAC F       /12BIT CHAR. IN F
ATMWI,    LAC (404000)  /ATOM TABLE ENTRY FOR 12 BIT CHAR.S
          JMP ATMWC

/ENTRY TO TEST CHAR. IN F AGAINST MASK IN E
REATM,    DAP H
          CLA
          LIO F
          RCL 6S
          LIO E
          SNI
          SAS (74)
          JMP REATM1
          JMP STV5UL
REATM1,   SAS C77
          JMP ATMI
          JMP ATMWI
```

```
/ROUTINE TO LCH I E, DAC A AND SKIP ON NOT EOM
STVLE,    DAP H
          LCH I E
          SAD ML6
          JMP STVLEW  /WARNING
STVLEI,   DAC A
          SAS ML4
          IDX H       /NOT EIM
          JMP HEXIT
STVLEW,   RCL 6S
          LCH I E
          RCR 6S
          JMP STVLEI  /12 BIT CHAR. IN A

/RELOCATABLE TRACING SUBROUTINE FOR STV
STVTRA,   DAP H
          LAW 7777    /TOP 6 BITS 0 FIRST TIME THROUGH
          AND E
          LIO E
STVTRL,   RIL 4S
          SPI
          ADD I (SUR)
          DAC E
          RIL 1S
          SPI I
          JMP HEXIT
          LAC I E
          AND (607777
          LIO I E
          JMP STVTRL

/OPCODE DISPATCH TABLE
STVDTB,   JMP STV0
          JMP STV1
          JMP STV2
          JMP STV3
          JMP STV4
          JMP STV5
          JMP STV6
          JMP STV7
```

```
/ATOM TABLE, SIGN BIT HAS CLASS - 0 MEANS EOM
ATOM,B13,  20              /SPACE
           400010          /!
B5,        10000           /"
           410000          /#
           410000          /$
           410000          /%
           410000          /&
B8,        1000            /'
           400020          /(
           400040          /)
           400001          /*
B11,       100             /+
B7,        2000            /,
B12,       40              /-
B10,       200             /.
           400002          //
B16,       REPEAT 10., 2   /DIGITS
           400004          /:
B6,        4000            /;
           400400          /<
           401000          /=
           402000          />
B14,       10              /?
ATOM+40,   410000          /@
B9,        400             /A
B17,       REPEAT 3, 1     /BCD
400        REPEAT 3, 1     /E FGH
400        REPEAT 5, 1     /I JKLMN
400        REPEAT 5, 1     /O PQRST
400        REPEAT 3, 1     /U VWX
400        1               /Y Z
           400100          /[
           0               /EOM
           400200          /]
B15,       4               /CRLF
ATOM+77,   /NO ENTRY FOR 77
```

```
/BGI, GET, AND PUT IOTS
/JULY 1964 WFM, MOD SRW 9-21-65

/BEGIN ITEM

BGI,    LAC I (USERAC) /GET MAP ADDRESS
        DAC A
        LAW 17
CANDI,  AND I A     /CONSTANT 30000. LENGTH OF OVERHEAD
        ADD I (USERIO) /ITEM ADDRESS
        DAC B
        DAC C       /BEGINNING OF DATA
        LAW 1760
        AND I A
        RAR 4S      /LENGTH OF PRIMARY INFO
        DAC I C     /POINTER TO END
        SUB B16
        LIA         /RELATIVE POINTER FROM TREE BEG. TO TREE
        IDX B
        LAC B1      /(200000)
        DAC I B
        LAW 7
BGA,    ADD A
        DAC A       /FIRST TREE POINTER
        LAC I A
        AND CJ      /(600000
        SZA
        JMP BGB
        IDX I C     /ONE WORD MORE BEFORE VARIABLES
        IDX B       /SET TO FIRST TREE
        DIO I B     /SET POINTER TO TREE
        LAW 3
        JMP BGA     /GO TO NEXT TREE POINTER

BGB,    IDX B
        AAI
        DAC D
        CLI"U"CMI
BGC,    DIO I B
        IDX B
        SAS D
        JMP BGC
        LAC ML4     /EOM
        DAC I B
        JMP R1
```

```
/GET A FIELD

PUT3=I (ATEM)
GET1=I (ATEM+1)
GET2=I (ATEM+2)
UPUSH=ATEM+3

GET,      LAC I (USERAC)
          DAC A        /AC (ERRORS POINTER)
          LAC I A
          DAC PUT3     /SAVE INFO ADDRESS FOR PUT
          DAC GET1     /SAVE ERRORS AREA
          IDX A        /AC+1 (FIELD ADDRESS)
          ADD B16      /(2
          DAC B        /AC+3 (MAP POINTER)
          LAC I B
          DAC C        /MAP
          ADD I A
          DAC D        /FIELD LOCATION
          LAW I 1777
          AND I C
          RAL 8S       /MAX. FIELD ADDRESS
          SUB I A
          SPA
          JMP FTE      /FIELD ADDRESS TOO HIGH - ERROR 0
          IDX GET1
          IDX A        /AC+2 (ITEM POINTER)
          LAW 17
          AND I C      /OVERHEAD LENGTH
CHARLA,   ADD I A
          DAC GET2     /BEG. OF INFO
          LIO I D
          SPI
          JMP FCD      /SECONDARY INFO
          DAC C        /PRIMARY INFO
FTB,      RIL 1S
          IDX GET1     /SUBSCRIPTS NOT EXCEEDED
          IDX D        /FIELD LOCATION+1
          DAC B
          IDX B        /FIELD LOCATION+2
          LAW I 7600
          AND I D      /POINTER TO INFO
          ADD C
          SPI I
          JMP FTE1     /NOT INFO - ERROR 2
          LIO I B
          SPI
          JMP FTA      /FIXED DATA
          DAC B        /VARIABLE DATA
          LAC GET2
          DAC A        /BASE OF POINTER TO VARIABLES
          ADD I A      /POINTER TO VARIABLES
          ADD I B      /POINTER TO DATA
FTA,      SZF 1
          JMP FPR      /PUT
          DAC I (USERAC) /GET, RETURN
          DIO I (USERIO)
          JMP R1
```

```
FCD,    LAW UPUSH
        DAC E          /BEG. OF PUSH-DOWN LIST (USER'S CORE)
       LAC D          /TOP FIELD ADDRESS
        DAC I E
FCG,    IDX E          /PUSH
        SAD (UPUSH+6)
        C16RET+1       /TOO LOW - PUNT
        CLA
        RCR 8S
        RAL 8S         /POINTER TO NEXT LEVEL
        ADD C          /POINTER BASE
        DAC I E        /PUSH DOWN INFO
        DAC F
        LIO I F        /TREE BASE?
        SPI
        JMP FCG        /NO, GO ON DOWN TREE
        IDX F          /TO TREE POINTER
        LAW 177
        AND I F        /TREE POINTER
        ADD GET2
        DAC C          /TREE LOCATION
        ADD I C
        DAC G          /NEXT LEVEL LOCATION
        IDX B
        LAC I B        /AC+4 (SUBSCRIPT POINTER)
        DAC B          /SUBS. AREA
        LAW I 1
        ADD E
FCC,    DAC H          /POP 1 LEVEL
        LAC I H
        DAC A          /FIELD POINTER
        LAC (77400)
        AND I A        /LEVEL LENGTH
        DAC A
        MUL I B        /SUBSCRIPT
        SCL 9S
        ADD I C        /ADD BASE
        ADD C
        DAC C          /NEXT LEVEL
        SUB I G
        SUB G
        SMA            /TOO HIGH?
        JMP FTEP       /ERROR 2 - PUT MAYBE
       LAC A
        RAR 8S
        ADD C
        DAC G          /NEXT LEVEL FOR CHECKING
       IDX B          /NEXT SUBSCRIPT
        LAW I 1
        ADD H
        SAS (UPUSH-1)
        JMP FCC        /GO ANOTHER LEVEL
FCF,    LIO I D        /DONE, RESTORE IO
        JMP FTB
```

```
FTE1,   SZF 1
        JMP R1
        LIO I B
        DIO I (USERIO)

FTEP,   SZF 1
        JMP FPE     /GET ERROR IN PUT, GO MAKE ROOM
FTE,    SZF 1
        C16RET+1    /PUT ERROR; PUNT
        DAC I (USERAC) /GET ERROR; SETUP AC
        LAC GET1
        DAC B
        LAC I B
        JMP GO

/PUT A FIELD

FPR,    SPI
        JMP PTF     /FIXED INFO
        LAW 1
        ADD A
        DAC C       /VARIABLE LENGTH INFO
        LAC I C
        DAC I B     /SET NEW VARIABLES LOCATION
        ADD A
        ADD I A
        DAC B       /NEW CORE LOCATION
        LAC PUT3
        DAC D
PTO,    LCH I D     /MOVE DATA TILL EOM
        DCH I B
        SAS C74
        JMP PTO
        LAC B
        SUB A
CHARLC, SUB I A
        DAC I C     /SET NEW VARIABLES LENGTH
        JMP R1
```

```
PTF,    DZM A       /FIXED DATA
        DAP A
        AND ML5     /(760000
        RAL 5S
        DAC B       /BIT ADDRESS
        ADD (LAC PTS)
        DAC C       /LAC LEFT MASK
        ADD (XCT PTU-[LAC PTS])
        DAC D       /LAC ROTATE
        XCT C
        DAC C       /LEFT MASK
        LAC PUT3
        DAC G
        CLA
        RCR 5S
        RAL 5S
        ADD B       /EXCESS BITS
        SUB (18.)
        SZM
        STF 2       /MORE THAN ONE WORD EXCESS
       ADD (AND PTT)
        SZF I 2
        ADD (18.)
        DAC E       /AND RIGHT MASK
       CLA
        RCR 4S
        RAL 4S
        IDA
        SZF 2
        IDA
        CMA
        DAC F       /-WORD COUNT
        JMP PTL

PTM,    LAC I G     /GET DATA
        RCR 9S
        XCT D       /ROTATE IT
        XOR I A     /INSERT IT THROUGH THE MASK IN C
        AND C
        XOR I A
        DAC I A
        LIO I G     /GET PART OF NEXT WORD
        IDX G       /MOVE UP POINTERS
        IDX A
CHARLY, LAW I 0
        DAC C       /RESET MASK
```

```
PTL,    ISP F       /COUNT F
        JMP PTM
        SZA
        JMP R1      /COMPLETELY DONE
        LAC C       /NEXT TO LAST
        XCT E
        DAC C       /FINAL MASK
        JMP PTM

/MAKE SPACE IN TREE

FPE,    SZA
        C16RET+1    /ERROR IF PTR OR >1 OFF
        LAC H
        SAS (UPUSH)
        C16RET+1    /ERROR IF NOT AT TOP LEVEL
        LAC GET2
        DAC B
        DAC H       /VARIABLES BEG. POINTER
        IDX H       /VARIABLES LENGTH POINTER
        LAC A
        RAR 8S
        DAC A       /ADJUST GROUP LENGTH
        ADD I B
        DAC I B     /NEW VARIABLES BEGINNING
        ADD I H
        AND MR12    /(7777
        ADD B       /NEW ENS LOCATION
PUJ,    DAC H       /MOVE INFO UP
        SUB A
        DAC B
        LIO I B     /OLD
        DIO I H
        SAD C       /STOP AT INSERTION POINT
        JMP PUIC    /DONE
        LAW I 1
        ADD H       /MOVE NEXT WORD
        JMP PUJ

PUIC,   LIO I D
        RIL 2S
        SPI I       /LOWEST LEVEL IN TREE?
        JMP PUIA    /NO
PUIB,   CLC
        DAC I H
PUIA,   LAW I 1
        ADD H
        DAC H
        SAS C
        JMP PUIB

PUI,    LAW 177
        AND I F
        ADD GET2
        DAC B
        DAC H       /TREE POINTER LOCATION
        LAW 2
        JMP PUB
```

```
PUC,    LAC I B     /INCREMENT TREE POINTER
        ADD A
        DAC I B
        LAW 3
PUB,    ADD F       /NEXT MAP FIELD
        DAC F
        IDX B       /NEXT ITEM WORD
        LAC I F
        AND CJ      /(600000
        SZA I       /ANOTHER TREE POINTER?
        JMP PUC     /YES
        LAW UPUSH
        SUB E
        DAC E       /-NUMBER OF LEVELS

PUD,    LAC H       /GO DOWN WITH SUBSCRIPTS 0; USED AS CONSTANT
        ADD I H
        DAC H
        ISP E
        JMP PUD
        LAC I (UPUSH+1)          /POINTER TO 1 LEVEL FROM LOWEST
        DAC E
        LAC (77400
        AND I E
        RAR 8S
        DAC E       /LENGTH OF GROUP ABOVE LEVEL

PUE,    LAC I G     /INCREMENT POINTER OF ABOVE LEVEL
        ADD A
        DAC I G
        LAC G
        SAD H       /LOWEST LEVEL?
        JMP PUG     /YES, DONE
        ADD E       /NO, GO TO NEXT POINTER
        DAC G
        JMP PUE

PUG,    SPI I       /LOWEST LEVEL IN TREE?
PUH,    SAD C       /OR LAST POINTER ON LEVEL?
        JMP FCF     /YES, BACK TO GET
        ADD A
        DAC G       /GO TO NEXT POINTER
        LAC I G
        ADD A       /INCREMENT IT
        DAC I G
        LAC G
        JMP PUH
```

```
/TABLES FOR FIXED TRANSFERS

PTT,        0
BO,         400000
            600000
            700000
ML4,        740000
ML5,        760000
ML6,        770000
            774000
            776000
            777000
ML10,       777400
            777600
            777700
ML13,       777740
ML14,       777760
            777770
ML16,       777774
ML17,       777776

PTS,        777777
MR17,       377777
MR16,       177777
            77777
MR14,       37777
MR13,       17777
MR12,       7777
            3777
            1777
C777,       777
C377,       377
            177
C77,        77
            37
C17,MR4,    17
C7,         7
C3,MR2,     3
            1

PTU,        RCL 9S
            RCL 8S
            RCL 7S
            RCL 6S
            RCL 5S
            RCL 4S
            RCL 3S
            RCL 2S
            RCL 1S
CHARLT,     SKP             /NOP
            RCR 1S
            RCR 2S
            RCR 3S
            RCR 4S
            RCR 5S
            RCR 6S
            RCR 7S
            RCR 8S

START
```

```
FLOATING ARITHMETIC ROUTINES  11-10-66  (FARITH,12)

/APM 8-64, MOD SRW 12-65

FPOVFL,    IDX I (USERPC) /FLOATING OVERFLOW
           DAP I (FOV+1)  /SETUP FOV AREA
           LAC CJ
            DIP I (FOV+1)
            LAC (ADD FOV)
            DAC I (USERPC)
            C16RET

FPNORM,    DAP H          /NORMALIZE 36-BIT MANTISSA IN G AND IO
           DZM F          /SHIFT COUNTER
           CLL            /ROUTINE SETS LINK FOR +@- 0
FPNOR1,    LAC G          /HI MANTISSA
FPNOR2,    ADD B1         /(200000)
           SPQ            /SKIPS IF AC WASN'T NORMALIZED
          JMP HEXIT      /EXIT WITH NORMALIZED MANTISSA IN G AND IO
           LAC G          /SETUP FOR NORMALIZING SHIFT
          SCL 1S
           DAC G
           IDX F          /INCREMENT DECREMENT
          SAS C44        /(36.) DO WE HAVE +@- 0?
          JMP FPNOR1     /NO.  GO AROUND AGAIN.
          CLL"U"CML      /+@- 0 INDICATOR
           JMP HEXIT

/FLOATING ROUND SUBROUTINE
FROUND,    DAP H          /MANTISSA IN G AND IO
           SZL            /+@- 0?
           JMP H          /YES, RETURN, H HAS A JMP FROM FPNORM
          LAW I 1        /DECREMENT THE DECREMENT TO THE CHARACTERISTIC
           ADD F
           DAC F
           LAC G          /UN-NORMALIZE TO PROTECT SIGN BIT
          SCR 1S
           SWP
           TAD B11        /ROUNDING ADD
           SWP
           TAD CZ         /ADD IN ANY POSSIBLE CARRY
          DAC G          /SET UP TO JUMP INTO FPNORM
           JMP FPNOR2     /TO RE-NORMALIZE

/FLAC IOT
FLAC,      JSP TRACE
           LIO I A
           IDX A
           LAC I A
           DIO I (FAC)
           DAC I (FAC+1)
           JMP R1
```

```
/FDAC IOT
FDAC,     JSP TRACE
          LAC I (FAC)
          DAC I A
          IDX A
          LAC I (FAC+1)
          DAC I A
          JMP R1

/FLOAT IOT
FLOAT,    LAC I (USERAC) /GET NUMBER TO BE FLOATED
          SCR 9S         /CONVERT TO 36-BIT UN-NORM. MANTISSA
          SCR 9S
FPFLT2,   DAC G          /SET UP NORMALIZE SUBROUTINE
          IDX I (USERPC) /GET PTR. TO BINARY POINT SPEC.
          DAC A
          LAW 35.        /CHARACTERISTIC
          ADD I A
          DAC C          /STORE CHARACTERISTIC IN C, LIKE FADD DOES
          JMP FADD4      /GO NORMALIZE AND PUT THE NUMBER TOGETHER

/FLOAT2 IOT
FLOAT2,   LAC I (USERAC) /GET THE NUMBER TO BE FLOATED
          LIO I (USERIO) /AS A 36 BIT MANTISSA
          JMP FPFLT2     /GO HANDLE IT JUST LIKE FLOAT1 DOES

/FIX IOT
FIX,      JSP FPFIX      /CONVERT TO FIXED POINT MANTISSA
          ADD CZ         /TEST AC FOR +@- 0
          SZA            /IS IT?
          JMP FPOVFL     /NO.  NUMBER WON'T FIT INTO AC.  OVERFLOW.
          LAI            /SIGN BITS OF BOTH HALVES MUST ALSO AGREE
          XOR C
          SPA            /DO THEY?
          JMP FPOVFL     /NO.  NUMBER NEEDS 19 BITS.  OVERFLOW.
          DIO I (USERAC) /PUT LOW ORDER PART IN AC
          JMP R1

/FIX2 IOT
FIX2,     JSP FPFIX
          DIO I (USERIO) /GIVE IT TO USER
          DAC I (USERAC)
          JMP R1
```

```
/SUBROUTINE TO GET A FIXED POINT MANTISSA
FPFIX,    DAP A
          LAC CJ
          DIP A          /SETUP EXIT
          IDX I (USERPC) /GET PTR. TO BINARY PT. SPECS.
         DAC B          /B POINTS AT USER'S CORE
         LAC I (FAC+1)  /GET CHARACTERISTIC OF FAC ALONE
          RAR 8S
          SAR 9S
          SAR 1S
          ADD I B        /ADD IN USER'S BINARY POINT SPECIFICATION
          SUB C44        /SHIFT BIN. PT. TO FAR RIGHT AND 1 EXTRA
          /FOR FIRST ISP AT FPFIX2+1
          SMA            /TEST CONVERTED CHARACTERISTIC
          JMP FPOVFL     /TOO BIG FOR A 36-BIT INTEGER.  OVERFLOW.
          DAC B          /SAVE NUMBER TO ISP ON LATER
          LAC I (FAC)    /GET 36-BIT MANTISSA
          LIO I (FAC+1)
          SCR 8S         /GET RID OF CHARACTERISTIC
         SCL 8S
          JMP FPFIX2     /ISP FIRST IN CASE B CONTAINS -1
FPFIX1,   LAC C          /UN-NORMALIZING LOOP
         SCR 1S
FPFIX2,   DAC C
          ISP B          /DONE UN-NORMALIZING?
         JMP FPFIX1     /NO.  GO AROUND AGAIN.
         LAC C          /SET UP FOR RETURN
          SAS PTS        /(-0)  IS HIGH ORDER HALF MINUS ZERO?
         JMP A          /NO, RETURN RIGHT AWAY
         SWP            /YES.  GET LOW ORDER HALF IN AC FOR TESTING.
          SAD PTS        /(-0)  IS LOW ORDER HALF MINUS ZERO ALSO?
          CLA"U"CLI      /YES.  BOTH HALVES ARE -0.  MAKE THEM +0.
          SWP            /GET THE HALVES BACK IN NORMAL ORDER
         DAC C
          JMP A          /RETURN WITH INTEGER IN AC AND IO
```

```
/FADD AND FSUB IOTS
FADD,     JSP TRACE    /FLAG 1 SET FOR FSUB
          LIO I A      /GET THE ADDEND
          IDX A
          LAC I A      /GET 18-BIT CHARACTERISTIC ALONE
         RAR 8S
          SAR 9S
          SAR 1S
          ADD B17      /A 1-BIT UN-NORM.ING SFT IS DONE LATER
          DAC C
          LAC I A      /GET 36-BIT MANTISSA ALONE
          SWP
          SCR 8S       /GET RID OF CHARACTERISTIC
         SCL 7S       /1 BIT UN-NORM.ING SHIFT
          SZF 1        /FSUB?
          CMA"U"CMI    /COMPLEMENT ADDEND
          DAC A        /STORE MANTISSA
          DIO B
          LAC I (FAC+1)
          RAR 8S       /GET 18-BIT CHARACTERISTIC ALONE
          SAR 9S
          SAR 1S
          ADD B17      /A 1-BIT UN-NORM.ING SFT IS DONE LATER
         DAC F
          LAC I (FAC) /GET 36-BIT MANTISSA ALONE
          LIO I (FAC+1)
          SCR 8S       /GET RID OF CHARACTERISTIC
         SCL 7S       /1-BIT UN-NORMALIZING SHIFT, AS BEFORE
         DAC D
          DIO E        /LEAVES CONTENTS OF E IN I.O., FOR USE LATER
          LAC C        /COMPARE CHARACTERISTICS
          SUB F
          SMA          /IF C<F, SWITCH A,B,C WITH D,E,F
          JMP FADD1    /IF C=F OR C>F, GO AROUND
          LAC A        /SWAP NUMBERS.  SEMI-EXEC-MODE CROCK.
          LIO D
          DAC D
          DIO A
          LAC C
          LIO F
          DAC F
          DIO C
          LAC E
          LIO B
          DAC B
          DIO E        /LEAVES CONTENTS OF E IN I.O.
FADD1,    LAC A        /LARGER # IN A,B,C
          SZA          /TEST FOR NORM.ED +@- 0
         SAD PTS      /POSSIBLE COMPLEMENTED ZERO (FSUB)
         JMP FADD9    /FOUND A PLUS-OR-MINUS ZERO
```

```
FADD2,    LAC F         /UN-NORMALIZING LOOP
          SAD C         /END TEST FOR LOOP.  QUIT WHEN C=F.
         JMP FADD3
          IDX F         /COUNT SHIFTS
         LAC D         /SHIFT MANTISSA IN D AND E RIGHT 1 BIT
         SCR 1S
         DAC D
          JMP FADD2     /GO AROUND AGAIN

FADD3,    LAC D         /GET 36-BIT MANTISSA FROM D AND E
         SWP
          TAD B         /ADD IN LOW ORDER HALF OF OTHER MANTISSA
          SWP
          TAD A         /ADD IN HIGH ORDER HALF
         SWP
          TAD CZ
          SWP
          TAD CZ
          DAC G         /SET UP NORMALIZE SUBROUTINE
FADD4,    JSP FPNORM    /GO NORMALIZE 36-BIT MANTISSA
          JSP FROUND    /GO ROUND-OFF TO A 28-BIT MANTISSA
FADD5,    SZL I         /TEST FOR A PLUS-OR-MINUS ZERO MANTISSA
          JMP FADD7
FADD6,    CLA"U"CLI     /+@- 0
          JMP FADD8     /PUT ZERO IN FLOATING AC AND RETURN TO USER
FADD7,    LAC C         /CHARACTERISTIC IS IN C
          SUB F         /SUBTRACT DECREMENT
          SCR 7S        /GET CHARACTERISTIC INTO LEFT OF IO
          RAL 1S        /PUT SIGN OF CHARACTERISTIC INTO SIGN(IO)
          SCR 1S        /FOR OVERFLOW TESTING
          ADD G         /AC SHOULD NOW BE +@- 0, GET HI-ORD. MANT.
          SAS G         /WAS AC +@- 0
          JMP FPOVFL    /CHARACTERISTIC WON'T FIT, OVERFLOW
          RIL 8S        /GET CHARACTERISTIC ON RIGHT END OF I.O.
FADD8,    DAC I (FAC)   /STORE ANSWER
          DIO I (FAC+1)
          JMP R1

/FOUND IT WAS TRYING TO FADD 0 TO A NUMBER WITH 0@- CHARACTERISTIC
FADD9,    LAC F         /RETURN WITH NON-0 NUMBER AS ANSWER
          DAC C
          LAC D
          DAC G
          JMP FADD4
```

```
/FMUL IOT
FMUL,     JSP FPSBR1  /CODING SHARED BY FMUL & FDIV
          DIO I (ATEM)   /SAVE HIGH ORDER MANT1
          CLI         /GET LOW MANT1 ALONE AS +INTEGER IN AC
          RCR 8S
          MUL C       /MULTIPLY BY HIGH ORDER MANT2
          SCR 9S      /DIV BY 2↑10
          SCR 2S
          DIO H       /SAVE PARTIAL PRODUCT IN H
          LAC D       /GET LOW ORDER MANT2
          MUL I (ATEM)   /MUL BY HIGH MANT1
          SCR 9S      /TRUNCATE LOW ORDER 10 BITS AS BEFORE
          SCR 2S
          DIO G       /SAVE PARTIAL PRODUCT IN G
          LAC I (ATEM)   /MULTIPLY HI ORD PARTS OF MANTS TOGETHER
          MUL C       /THIS YEILDS THE LARGEST PARTIAL PRODUCT
          SCR 1S      /UN-NORM.ING SFT, COMPENSATED FOR LATER
          SWP         /GET LO PARTIAL PRODUCT INTO AC
          TAD G       /ADD IN OTHER PARTIAL PRODUCTS
          SWP
          TAD CZ      /(0)
          SWP
          DAC D       /SET UP FPSBR2
          DZM F
          JSP FPSBR2  /FPSBR2 IS MORE CODING SHARED BY FMUL AND FDIV
          ADD B       /ADD IN CHAR1
          IDA         /1 BIT UN-NORM.ING SFT BONE BEFORE
FMUL1,    DAC C       /SET UP FOR JMP INTO FADD
          LIO H
          JMP FADD5   /GO PUT RESULTS IN FLOATING AC AND RETURN

/FDIV IOT
FDIV,     JSP FPSBR1
          SWP         /SET UP FOR DIVIDE
          SCR 8S      /GET RID OF CHARACTERISTIC
          SCL 6S      /EFFECTIVE RIGHT SHIDT OF 2 OR 3 BITS
          DIV C       /2↑31<=N<2↑32, DIVIDED BY 2↑16<=C<2↑17
          JMP R1      /FIRST RETURN, FOR DIVISION BY ZERO
          DAC G       /YIELDS 2↑14<AC<2↑16
          IDX I (USERPC) /SETUP FOR 2ND RETURN TO USER
          CLA"U"SWP   /MULTIPLY REMAINDER BY 2↑17
          DIV C       /((R<C)"T"2↑17), DIVIDED BY 2↑16<=C<2↑17
C43,      35.
          RAL 1S      /YIELDS AC<2↑17
          DAC H       /TACK IT ON AS AN 18-BIT EXTENSION OF G
          LAC D       /COMPUTE CORRECTION TERM
          RAL 6S
          CLI         /DELETABLE IF IO DOESN'T AFFECT ACCURACY
          DIV C
C44,      36.
          MUL G
          RCL 2S      /CORRECTION TERM IN AC WITH PROPER SCALE
          CMA         /CORRECTION TERM IS NEGATIVE
```

```
          LIO G          /SET UP FPSBR2
          DAC D
          CLC            /-0 INTO F, SINCE CORR. TERM IS NEG.
          DAC F
          JSP FPSBR2
          CMA            /SUBTRACT CHARACTERISTICS
          ADD B
          ADD B16        /CORRECT FOR UN-NORMALIZING SFT OF 2 PLACES
          JMP FMUL1      /GO NORMALIZE AND RETURN

FPSBR1,   DAP E          /SUBROUTINE SHARED BY FMUL AND FDIV
          LAC CJ
          DIP E          /RETURN IS IN E
          JSP TRACE
          LAC I A        /GET SIGN OF RESULT AND SAVE IN B
          XOR I (FAC)
          DAC B
          LIO I A        /GET HIGH ORDER MANT2
          IDX A
          LAC I A        /GET LOW ORDER MANT2
          SPI            /MAKE MANTISSA POSITIVE
          CMA"U"CMI
          DIO C          /SAVE HIGH ORDER MANT2 IN C
          CLI            /GET LO ORDER MANT2 ALONE AS A +INTEGER
          RCR 8S
          DAC D          /SAVE LOW ORDER MANT2 IN D
          LIO I (FAC)    /GET MANT1
          LAC I (FAC+1)
          SPI            /MAKE MANTISSA POSITIVE
          CMA"U"CMI
          JMP E          /RETURN

FPSBR2,   DAP E          /SUBR SHARED BY FMUL & FDIV
          LAC D          /RESTORE AC PREVIOUS TO JSP
          TAD H          /CONTINUE ADDING PARTIAL RESULTS TOGETHER
          SWP
          TAD F          /ADD IN POSSIBLE CARRY, -0 FOR FDIV
          DAC G          /SET UP NORMALIZE ROUTINE
          JSP FPNORM     /GO NORMALIZE
          JSP FROUND     /ROUND TO A 28-BIT MANTISSA
          LAC B          /RESTORE SIGN OF RESULT
          SPA
          CMI
          DIO H
          LIO G
          SPA
          CMI
          DIO G          /RESULT IS NOW IN G AND H WITH PROPER SIGN
          LAC I (FAC+1)  /GET CHAR1 ALONE
          RAR 8S
          SAR 9S
          SAR 1S
          DAC B          /SAVE CHAR1 IN B
          LAC I A        /GET CHAR2 ALONE
          RAR 8S
          SAR 9S
          SAR 1S
          JMP E          /RETURN

START
```

```
FLOATING IN-OUT ROUTINES  11-10-66  (FINOUT,12)

/APM 8-64, MOD SRW 12-65

/FLIP IOT
FLIP,      LAC I (FSA)
FLIP+1,    DAC A        /KEEP TEXT PTR IN A
           DZM G        /HIGH ORDER MANTISSA IN G
           DZM H        /LOW ORDER MANTISSA IN H
           DZM C        /CHARACTERISTIC IN C
           DZM I (ATEM)   /CORRECTIVE SCALE FACTOR IN ATEM
           DZM B        /EXPONENT OF "E" IN B
CHLPLS,
FLIP1,     LCH I A      /MAIN LOOP, CLASSIFY CHARACTER
           SAD (CHARAC L.)
           JMP FLIP4
           SAS CHLPLS   /(CHARAC L+)
           SAD (CHARAC L-)
           JMP FLIP5
           SZA I
           JMP FLIP5
           SAD CHARLE   /(CHARAC LE)
           JMP FLIP6
           RAL 6S       /TEST FOR A DIGIT
           XOR B13      /(20)
           SUB C10.
           SPA          /WAS CHARACTER A DIGIT?
           JMP FLIP7    /YES.  GO PROCESS IT.
FLIP2,     SZF 4        /ANY DIGITS IN THIS FIELD?
           JMP FLIP13   /YES.  GO EVALUATE FOREGOING STRING.
FLIP3,     JSP FLIPX    /EXIT SUBROUTINE, FAILURE
           JMP R1       /GIVE R1 TO INDICATE FAILURE

FLIP4,     SZF I 1      /DECIMAL POINT, CHECK FOR REDUNDANCE
           SZF 2        /HAVE WE SEEN AN "E" ALREADY?
           JMP FLIP2    /YES.  THIS CHARACTER FAILS.
           STF 1        /FLAG 1 INDICATES "SEEN A DECIMAL POINT"
           JMP FLIP1    /GO GET NEXT CHARACTER

FLIP5,     SZF I 3      /SIGN.  CHECK FOR REDUNDANCE
           SZF 1        /HAVE WE SEEN A DECIMAL POINT ALREADY?
           JMP FLIP2    /YES.  THIS CHARACTER FAILS.
           SZF 4        /HAVE WE ALREADY SEEN A DIGIT?
           JMP FLIP13   /YES, THIS CHARACTER FAILS
           STF 3        /INDICATES "SEEN A SIGN" IN THIS FIELD
           SAS (CHARAC L-)
           JMP FLIP1    /NO.  GET NEXT CHARACTER.
           SZF 2        /SET FLAG 6 IF WE HAVE SEEN AN E
           STF 6        /FLAG 6 INDICATES A NEGATIVE EXPONENT OF "E"
           SZF I 2      /IF FLAG 2 IS NOT SET, SET FLAG 5
           STF 5        /FLAG 5 INDICATES A NEGATIVE MANTISSA
           JMP FLIP1    /GET NEXT CHARACTER
```

```
FLIP6,    SZF I 2      /FOUND AN E, CHECK FOR REDUNDANCY
          SZF I 4      /HAS THE E PRECEDED ALL DIGITS?
          JMP FLIP2    /YES.  THIS CHARACTER FAILS.
          CLF 1        /INITIALIZE FOR A NEW FIELD
          STF 2        /FLAG 2 INDICATES "SEEN AN "E""
          CLF 3        /DONE SO A SIGN WON'T FAIL
          CLF 4
          JMP FLIP1    /GET NEXT CHARACTER

FLIP7,    STF 4        /FOUND A DIGIT
          ADD C10.     /GETS VALUE OF DIGIT IN AC
          DAC E        /SAVE DIGIT IN E TEMPORARILY
          SZF 2        /HAVE WE SEEN AN "E"
          JMP FLIP12   /YES.  GO BUILD UP EXPONENT OF "E".
          JSP FPM10    /NO, BUILD UP MANTISSA
          JSP FPNORM   /PREPARE TO ADD IN THIS DIGIT
          DIO H
          LAW 4        /NEEDED TO FINISH MUL BY 10
          ADD C
          SUB F        /SUBTRACT DECREMENT LEFT BY NORMALIZE ROUTINE
          DAC C        /SAVE NEW CHARACTERISTIC
          LAC E        /MAKE THIS DIGIT INTO A MANTISSA
          RAR 5S
          SCI"U"SZL I  /TEST FOR 0 ^ CLEAR IO
          JMP FLIP8    /PREVIOUS MANTISSA WAS NOT ZERO.  GO AROUND.
          DAC G        /STORE CURRENT DIGIT AS MANTISSA
          LAW 4        /WITH 4 AS CHARACTERISTIC
          DAC C
          JMP FLIP11   /GO CHECK FOR DIGIT TO RIGHT OF DEC. PT.

FLIP8,    DAC E        /SAVE HI-ORD PART OF MANTISSA, LO-ORD=0
          LAW 4        /CHAR.=4, SET UP TO UN-NORMALIZE FOR ADDING
          DAC F
FLIP9,    SAD C        /UN-NORMALIZING LOOP.  FINISHED WHEN C=F.
          JMP FLIP10
          LAC E        /SHIFT MANTISSA RIGHT 1 BIT
          SCR 1S
          DAC E
          IDX F        /ADD 1 TO CHARACTERISTIC TO PRESERVE VALUE
          JMP FLIP9    /GO AROUND AGAIN
FLIP10,   LAC E        /ADD THE TWO 36-BIT MANTISSAS TOGETHER
          SWP
          TAD H
          SWP
          TAD G
          SCR 1S       /NUMBERS ARE +, CORRECT POSSIBLE OVERFLOW
          AND MR17     /(377777)
          DAC G        /SET UP NORMALIZE ROUTINE
          JSP FPNORM   /GO NORMALIZE
          DIO H        /SAVE LO-ORD MANTISSA, (MUST BE NON-0)
          IDX C        /OFFSET UN-NORMALIZING SHIFT
          SUB F        /SUBTRACT NORMALIZING DECREMENT
          DAC C        /STORE NEW CHARACTERISTIC
FLIP11,   SZF 1        /ARE WE TO THE RIGHT OF THE DECIMAL POINT?
          IDX I (ATEM)   /YES, IDX THE CORRECTIVE SCALE FACTOR
          JMP FLIP1    /GET NEXT CHARACTER
```

```
FLIP12, LAC B        /GET CURRENT EXPONENT OF E
        MUL C10.
        SWP          /GET LOW-ORDER PART OF PRODUCT
        SNI"U"SMA I  /OVERFLOWS IF LARGER THAN 17 BITS
        JMP FLIP18   /OVERFLOW
        SAR 1S       /MULTIPLY LEAVES THINGS SHIFTED LEFT A BIT
        ADD E        /ADD IN CURRENT DIGIT
        DAC B        /SAVE NEW EXPONENT OF "E"
        JMP FLIP1    /GET NEXT CHARACTER

FLIP13, IDX I (USERPC) /SUCCESS, SET UP R2
        LAC B        /GET EXPONENT OF "E"
        SZA"U"SZF 6  /RESTORE SIGN UNLESS 0
        CMA
        SUB I (ATEM)    /COMBINE IT WITH CORRECTIVE SCALE FACTOR
        DAC I (ATEM)    /POWER OF TEN TO TAKE NUMBER TO
FLIP14, LAC I (ATEM)    /GET CORRECTIVE SCALING FACTOR
        SZA I        /END TEST, QUIT WHEN CORRECTION = 0
        JMP FLIP17   /EXIT LOOP, NUMBER IS NOW OK
        SPA          /MUST WE MUL OR DIV TO GET CORRECTION TO 0?
        JMP FLIP16   /GO ADD 1 TO SCALE FACTOR AND DIVIDE BY TEN
        SUB B17      /SUB (1) AND DIV BY 10
        DAC I (ATEM)
        JSP FPM10    /FPM10 MULTIPLIES MANTISSA IN G AND H BY TEN
        JSP FPNORM   /GO NORMALIZE
        DIO H        /SAVE LOW-ORDER MANTISSA IN H
        LAW 4        /ADD (4) TO CHARACTERISTIC IS PART OF MULTIPLY
FLIP15, ADD C        /ADD IN PREVIOUS CHARACTERISTIC
        SUB F        /SUBTRACT DECREMENT LEFT BY NORMALIZE ROUTINE
        DAC C        /SAVE NEW CHARACTERISTIC
        SZL I        /TEST FOR A ZERO MANTISSA
        JMP FLIP14   /MANTISSA WAS NON-ZERO.  GO AROUND AGAIN.
        JSP FLIPX    /MANTISSA WAS ZERO.  PREPARE TO EXIT
        JMP FADD6    /GO PUT ZERO IN FLOATING AC AND RETURN TO USER

FLIP16, IDX I (ATEM)    /ADD 1 TO SCALE FACTOR ^ DIV BY 10
        JSP FPD10    /DIVIDE MANTISSA IN G AND H BY TEN
        JSP FPNORM   /GO NORMALIZE
        DIO H        /SAVE LOW-ORDER MANTISSA
        LAW I 3      /DECREMENT CHARACTERISTIC FINISHES DIV BY 10
        JMP FLIP15   /COMPUTE NEW CHARACTERISTIC ^ LOOP AROUND
FLIP17, LAC G        /NUMBER IS NOW OF PROPER MAGNITUDE
        LIO H        /GET MANTISSA
        SZF 5        /IF FLAG 5 IS SET, MANTISSA IS NEGATIVE
        CMA"U"CMI
        DAC G        /SAVE MANTISSA WITH CORRECT SIGN
        DIO H
        JSP FLIPX    /PREPARE TO EXIT
        LIO H        /GET LOW-ORDER MANTISSA IN I.O.
        JMP FADD4    /GO PACK # INTO FAC ^ RETURN
FLIP18, IDX I (USERPC) /OVERFLOW
        JSP FLIPX    /PREPARE TO EXIT
        JMP FPOVFL   /GO TO THE FLOATING OVERFLOW ROUTINE
```

```
FLIPX,    DAP D        /FLIP EXIT SUBROUTINE
          LAC CJ
          DIP D        /SAVE RETURN IN D
          LAW I 1      /GET PTR TO CHAR AND UNSTEP IT
          ADD A
          IDC
          IDC
          DAC I (FSA) /STORE PTR IN FSA
          JMP D        /RETURN

/FLOPE ^ FLOPF IOTS

FLOP,     LIO I (JMODE)  /CHECK FOR JBH
          SPI"U"SZF 6
          JMP R2
          LAC I (FAC+1)  /GET FL PNT #
          LIO I (FAC)
          SPI            /MAKE SURE MANTISSA IS POSITIVE
          CMA"U"CMI"U"STF 1          /NEGATIVE FLAG
          AND ML10     /(777400)  GET RID OF CHARACTERISTIC
          DIO G        /SAVE 36-BIT POSITIVE MANTISSA IN G AND H
          DAC H
          LAW I 1      /C CONTAINS EXPONENT OF 10 LESS 1
          DAC C
          LAC I (FAC+1)  /GET CHARACTERISTIC
          RAR 8S
          SAR 9S
          SAR 1S
FLOP1,    DAC E        /SAVE IT IN E
          SZM          /IS CHARACTERISTIC >0?
          JMP FLOP6    /YES, DIVIDE FL. PNT. # BY 10 ^ IDX C
          ADD C3       /(3)
          SMA          /TEST WHETHER NUMBER IS <-3
          JMP FLOP7    /EXIT.  CHARACTERISTIC IS -3, -2, -1, OR 0.
FLOP2,    JSP FPM10    /MUL FL. PNT. # BY 10 ^ DECREMENT C
          JSP FPNORM   /NORMALIZE
          DIO H        /LO-ORD MANTISSA
          SZL I        /ZERO?
          JMP FLOP4    /NO.  NON-ZERO MANTISSA.  GO AROUND.
FLOP3,    DZM C        /GO PRINT A 0
          JMP FLOP10   /EXIT WITH C, G, AND H ALL PLUS ZERO
FLOP4,    LAW I 1      /SUBTRACT 1 FROM EXPONENT OF TEN
          ADD C
          DAC C
          LAW 4        /PART OF MUL BY 10
FLOP5,    ADD E        /ADD IN CHARACTERISTIC
          SUB F        /SUBTRACT DECREMENT LEFT BY NORMALIZE ROUTINE
          JMP FLOP1    /GO AROUND AGAIN
```

```
FLOP6,    JSP FPD10    /DIVIDE NUMBER BY TEN
          JSP FPNORM   /NORMALIZE
          DIO H        /SAVE LOW ORDER HALF OF MANTISSA
          SZL          /WERE WE NORMALIZING PLUS ZERO?
          JMP FLOP3    /YES.  EXIT.  PRINT OUT A ZERO.
          IDX C        /ADD 1 TO EXPONENT OF TEN
          LAW I 3      /PART OF DIV BY 10
          JMP FLOP5    /COMPUTE NEW CHARACTERISTIC ^ GO AROUND AGAIN

FLOP7,    SZF 3        /HAVE WE BEEN HERE BEFORE?
          JMP FLOP8    /YES.  GO ON TO NEXT PART OF ROUTINE.
          STF 3        /REMEMBER THAT WE'VE BEEN HERE
          JMP FLOP2    /SOME -3 CHARACTERISTICS CAN GO TO 0, TRY

FLOP8,    LIO H        /FP # IN E,G,H; UN-NORMALIZE G,H TO GET E=0
          LAC E
FLOP9,    SZA I        /FINISHED?
          JMP FLOP10   /EXIT
          LAC G
          SCR 1S
          DAC G
          IDX E
          JMP FLOP9
FLOP10,   DIO H        /PUT LOW ORDER MANTISSA BACK IN H
          LAW ATEM
          DAC A        /A IS TEM STORAGE PTR
FLOP11,   JSP FPM10    /LOOP TO GENERATE 8 DIGITS
          LAC G        /HI-ORD RESULT OF MUL
          DAC B
          AND MR13     /(17777) REMOVE BITS TO LEFT OF BIN PT
          SCL 4S       /PUT BIN PT BETWEEN BITS 0^1
          DAC G        /STORE NEW MANTISSA
          DIO H
          LAC B        /GET NEW DIGIT ALONE IN LO END OF AC
          RAL 5S
          AND MR4      /(17)
          DAC I A      /STORE IN TEM AREA
          IDX A        /MAKE A POINT TO NEXT REGISTER
          SAS (ATEM+10)  /QUIT AFTER 8 DIGITS
          JMP FLOP11   /GO AROUND AGAIN
          LAC I (STS)
          DAC G        /SET UP BYTE PTR FOR TEXT
          IDX I (USERPC) /GET PTR TO FORMAT WORD
          DZM E
          DAP E        /SAVE PTR
          LCH I E      /1ST BYTE OF FORMAT WD IS # OF COLUMNS
          RAL 6S
          IDA
          CMA
          DAC I (ATEM+10)            /COLUMN COUNT WD
          LCH I E      /NEXT BYTE IS # OF SIGNIFICANT FIGURES
          RAL 6S
          DAC B
          LCH I E      /3RD BYTE IS # OF PLACES PAST DEC. PT.
          RAL 6S
          IDA
          DAC F        /1 + # OF SIG. FIGURES
```

```
        LAC C       /THIS SECTION CALCULATES WHICH DIGIT TO ROUND
        SZF 2       /SKIP ON F FORMAT
        CLA
        ADD F
        LIA
        SUB B
        SZM         /ROUND ON ACCURACY OR DIGITS?
        LIO B       /ACCURACY
        LAI         /# OF CHARACTERS TO BE TYPED
        SPA         /TEST FOR ROUND BEING TOO FAR LEFT
        STF 4       /TOO FAR LEFT, PF4 PREVENTS ROUNDING
        SUB B14     /(8.)
        SMA         /TEST FOR ROUND BEING TOO FAR RIGHT
        STF 4       /PREVENT ROUNDING IF SO
        ADD (ATEM+10)  /GETS PTR AT REG TO ROUND
        DAC A       /SAVE PTR FOR LOOP AT FLOP12
        DAC E       /SAVE PTR MORE PERMANENTLY FOR USE LATER
        LAW I 5     /SUB (5) ^ TEST WHETHER +v-
        ADD I A
        SZF I 4     /SKIP TO SUPPRESS ROUNDING
        SPA         /SKIP TO ADD CARRY TO NEXT CHARACTER
        JMP FLOP14  /QUIT ROUNDING
CHRLEQ,
FLOP12, DZM I A     /CARRY LOOP ON ROUND-UP
        LAW I 1     /UNSTEP TEM PTR
        ADD A
CHARL4, DAC A
        SAS (ATEM-1)   /TEST WHETHER ROUND CARRIED OFF THE LEFT END
        JMP FLOP13  /OTHERWISE, GO AROUND
        IDX I (ATEM)   /ATEM WAS 0, MAKE IT 1
        IDX C       /ADD 1 TO EXPONENT OF TEN
        IDX E
        JMP FLOP14  /EXIT FROM LOOP
CHARLE,
FLOP13, IDX I A     /ADD CARRY FROM PREVIOUS DIGIT
        SAD C10.    /DOES THIS CHARACTER CARRY LEFT NOW?
        JMP FLOP12  /GO AROUND CARRY LOOP AGAIN
FLOP14, CLI         /ROUNDOFF NOW COMPLETE, READY TO PRINT
        SZF 1       /FLAG 1 SET FOR NEGATIVE NUMBER
        LIO (CHARAC L-)
        IDX I (ATEM+10)
        SZF I 5
        JSP TYOIF
        LAW I 2     /INITIALIZE D FOR STORING DIGITS
        SZF I 2
        SUB C
        SMA
        LAW I 1
        DAC D       /D GETS IDX'ED FOR EACH DIGIT, STORE "." AT 0
        LAC C       /INITIALIZE B
        SZF I 2
        SMA
        LAW I 1
        ADD (ATEM)
        DAC B       /B ALWAYS POINTS TO NEXT DIGIT, AFTER IDX
```

```
FLOP15,   IDX D        /LOOP FOR PRINTING
          SZA          /SKIP IF TIME TO PRINT DECIMAL POINT
          JMP FLOP16   /OTHERWISE, GO AROUND
          LIO (CHARAC L.)          /PRINT DEC PT
          JMP FLOP20
FLOP16,   SAD F        /ALL DONE PRINTING DIGITS?
          JMP FLOP21   /QUIT PRINTING DIGITS, EXIT FROM LOOP
          IDX B        /GET PTR TO NEXT DIGIT TO BE PRINTED
          SUB (ATEM)   /FIGURE OUT IF TOO FAR LEFT
          SMA          /TOO FAR LEFT?
          JMP FLOP18   /NO.  GO AROUND.
FLOP17,   LAW 20       /YES, PRINT A 0
          JMP FLOP19
FLOP18,   LAC B        /FIGURE OUT IF TOO FAR RIGHT
          SUB E        /E IS PTR TO DIGIT ROUNDED AT
          SMA          /TOO FAR RIGHT?
          JMP FLOP17   /YES, PAST SIGNIFICANT FIGURES, PRINT A 0
          LAW 20       /NO.  GET INTERNAL CODE FOR THIS DIGIT.
          ADD I B
FLOP19,   RCR 6S       /POSITION INTERNAL CODE FOR PRINTING
FLOP20,   IDX I (ATEM+10)
          SZF I 5
          JSP TYOIF
          JMP FLOP15   /GO AROUND LOOP AGAIN
FLOP21,   SZF I 2      /FINISHED WITH DIGITS, TEST FOR E FORMAT
          JMP FLOP22   /"F" FORMAT.  GO AROUND.
          LAW 4
          ADD I (ATEM+10)
          DAC I (ATEM+10)
          SZF 5
          JMP FLOP22
          LIO CHARLE   /(CHARAC LE) E FORMAT, PRINT THE E
          JSP TYOIF
          LAC C        /GET EXPONENT OF E, TEST SIGN, AND TYPE IT
          LIO (CHARAC L-)
          SMA
          CLI"U"CMA
          DAC A
          JSP TYOIF
          LAC A
          CLI"U"CMA"U"SWP
          RCL 1S
          DIV C10.
C5,       5            /CONSTANT STORED AFTER DIVIDE
          RIR 6S
          RCL 6S
          IOR (2020)
          RCR 6S
          RCR 6S       /POSITION FIRST DIGIT
          JSP TYOIF
          JSP TYOIF
```

```
FLOP22,   LAC G        /UPDATE PTR
          SZF I 6
          DAC I (STS)
          SZF 5
          JMP FLOP23   /RIGHT ADJUST
          LAC ML4
          SZF I 6
          DCH I G
          JMP R1

FLOP24,   CLI
          JSP TYOIF
FLOP23,   ISP I (ATEM+10)
          JMP FLOP24
          CLF 5
          JMP FLOP14

/SUBR TO MUL A 36BIT +MANTISSA IN G,H BY 10 ^ SHIFT BIN PT 4 TO RIGHT
FPM10,    DAP D
          LAC CJ       /RETURN
          DIP D
          LAC H        /GET MANTISSA, WITH HALVES REVERSED
          LIO G
          AND ML14     /(777760) REMOVE LAST 4 BITS TO KEEP SIGN +
          RCR 3S       /UN-NORM.ING SHIFT TO PROTECT SIGN
          DIO G
          RAR 1S
          MUL C5       /(5), MUL (5) ^ SHIFT 1
          DAC F
          DIO H
          LAW 5
          MUL G
          RCR 1S
          LAI
          ADD F        /COMBINE THE TWO HALVES OF THE NEW MANTISSA
          DAC G
          LIO H        /LEAVES LO-ORD IN H ^ IO, HI-ORD IN G
          JMP D        /RETURN

/SUBR TO DIV A 36BIT +MANTISSA IN G,H BY 10 ^ SHIFT BIN PT 3 TO LEFT
FPD10,    DAP D
          LAC CJ
          DIP D
          LAC G        /SET UP FOR DIVIDE
          LIO H
          SCR 1S       /SIC.
          DIV CHARL4   /(240000)
          .
          DAC G
          CLA"U"SWP
          DIV CHARL4   /(240000)
          .
          RAL 1S
          DAC H
          LIA          /SAME EXIT FORMAT AS FPM10
          JMP D        /RETURN

START
```

```
JOBHUNTER  11-20-66  (JBH,24)
/WITH ↑N


/VARIABLES
JBUFP=DTEM              /6 TABLE PTRS.
/DTEM+6   /RESERVED FOR EXPANSION
JQN=DTEM+7              /CURRENT QUESTION NUMBER
JBF=DTEM+11             /BRIEF MODE FLAG
JANS=DTEM+12            /CORE ANSWER PTR
JDANS=DTEM+13           /DRUM ANSWER PTR
JDTR=DTEM+14            /DRUM TEXT RELOCATION FACTOR
JTTOCP=DTEM+15          /TEXT TOC PTR
JQTOCP=DTEM+16          /QN TOC PTR
JTBUFP=DTEM+17          /TEXT BUF PTR
JQBUFP=DTEM+20          /QN BUF PTR
JTBUFC=DTEM+21          /TEXT BUF CURRENT PTR
JQBUFC=DTEM+22          /QN BUF CURRENT PTR
JOJM=DTEM+23            /OLD JMODE ^ TEM STORAGE
JLGQN=DTEM+24           /LOCATION OF LAST GQN; EOM FOR NULLING
JLEVEL=DTEM+25          /LEVEL PTR INTO JQN REGISTERS
JLQNT=DTEM+26           /LAST QN TYPED
JORG=DTEM+30            /LOCATION OF USER'S INIT; NULLING MODE IN INST PAR
JHTSU=DTEM+31           /TRAP STARTUP LOCATION
.REASK=IOT I 5064       /IOT PUT IN TTTSU


/QN TABLE ENTRY - 4 WORDS CREATED AT GQN:
/ QN WORD 1
/ QN WORD 2
/ BITS AND "QUESTION POINTER" (POINTER TO BLOCK OF TEXT PTRS)
/ -0 OR ANSWER TEXT MOBY POINTER
/BITS: 0,RECONS; 1,LIST START;2,STOP


/JMODE:
/-3 NULL
/-1 SIMULATE
/0  NORMAL  (EXCLUDES KILLED QUESTIONS)
/1  TOC ( = TYPE OUT FOR CORRECTION = RECONSIDER)
/2  LIST STOP
/3  LIST CONTINUE
/4  DEAD (QUESTION ANSWERED THEN KILLED. ACTS LIKE 0)
```

```
/SEARCH SUBROUTINE; QN TO BE SEARCHED FOR IN B,C

SEARCH, CLL"U"CML   /SET FOR READTQ
        DAP H
        LAC I (JQBUFC) /START WITH SEG NOW IN CORE
        AND MR12
        DAC D       /FOR ENDTEST
        DAC F

SEAR8,  LAW 2       /BEGIN SEARCHING A SEG
        ADD I (JBUFP+3)          /PTR TO SEARCH WITH
SEAR2,  DAC E
        SAS I (JQBUFP)
        JMP SEAR6   /DIFFERENT FROM TOP OF LAST SEG
        LAW 7777    /SAME
        AND I (JQBUFC)
        SAD I (JQTOCP)
        JMP SEAR1   /WE'RE IN LAST SEG, START AT BEGINNING
        LAC E
SEAR6,  SAD I (JBUFP+4)          /CHECK FOR TOP OF ANY SEG
        JMP SEAR7   /GET NEXT SEG
SEAR3,  LAC I E
        SAD B       /CHECK FIRST WORD OF QN
        JMP SEAR4
        LAW 4
SEAR5,  ADD E
        JMP SEAR2
SEAR4,  IDX E
        LAC I E
        SAD C       /CHECK SECOND WORD
        JMP SEARF
        LAW 3
        JMP SEAR5
SEARF,  IDX H       /2ND RETURN
        IDX E
        JMP HEXIT

SEAR1,  LAC I (JBUFP+4)          /START AT BEGINNING OF SEGS
        JMP SEAR9
SEAR7,  IDX F       /GET NEXT SEG
SEAR9,  SAD D       /DID WE START HERE?
        JMP HEXIT   /YES, GIVE RETURN 1
        DAC F       /SETUP PTR TO DA
        LAW SEAR8   /CALL READTQ; RETURN TO SEAR8
```

```
/ROUTINE TO READ TEXT SEG OR QN SEG (LINK=1 FOR QN)
READTQ,   DAP A
          LAC I (JTBUFC)
          SZL
          LAC I (JQBUFC)
          LIA          /FIND OUT WHAT'S IN CORE
          AND MR12
          DIP F        /COMPARE ADR PARTS ONLY
          SAD F        /SAME AS WHAT'S WANTED?
          JMP AEXIT    /YES, RETURN
          SPI I
          JMP READ1    /VALID ON DRUM, DON'T WRITE OUT
          DAC G        /WRITE OUT
          LIO I G      /DRA
          LAC I (JBUFP)
          SZL
          LAC I (JBUFP+3)              /CORE ADR
          SNI          /NO DRA?
          JMP READWN   /WRITE NON-ADR
          WAI+22       /REWRITE
           JSP JIOPER
          JMP READ1    /NOW GO READ
READWN,   WNIH+2       /WRITE OUT
           JSP JIOPER
          DIO I G      /SAVE DRA

READ1,    LAC F        /READ,
          SZL I
          JMP READ1T
          DAC I (JQBUFC) /UPDATE CURRENT POINTER
READ2T,   LIO I F      /GET DRA
          SNI
          JMP AEXIT    /NO DRA, NOTHING IS DESIRED
          LAC I (JBUFP)
          SZL
          LAC I (JBUFP+3)              /GET CORE ADR
          RAI+2        /READ
           JSP JIOPER
          JMP AEXIT

READ1T,   DAC I (JTBUFC) /UPDATE CURRENT PTR
          SUB I (JBUFP+1)              /CALCULATE RELOCATION VALUE
          DAC G
          LAW I 2
          ADD I (JBUFP+1)
          SUB I (JBUFP)
          MUL G
          DIV B17
C65,       65          /CONSTANT 65
          SUB I (JBUFP)
          SUB B16
          DAC I (JDTR)   /SETUP RELOCATION REGISTER
          JMP READ2T
```

```
/TELETYPE SUBROUTINES. CLOBBER H.

/SPACE
SPAC,     LIO (SPAT)

/TOS FROM IO
JTOS,     DAP H
          LAI
          TOS
           JMP JTTER
          JMP HEXIT

SPAT:     TEXT / #/

/TT ERROR ROUTINE AND .REASK=SICKTT IOT
REASK.,
JTTER,    LAW 2
          SAS I (ERCODE) /INTERUPTED BY NULL?
          EOTHNG       /NO, HANG
          LIO (JINTT) /YES, COMMENT AND REASK
          JSP JTOS
          LIO C64      /(64)
          LAC I (JMODE)  /BUT IF LISTING,
          AND ML17
          SAD B16
          LIO C7       /JUST RESTART
          JMP KILLFI   /GO REASK
JINTT:    TEXT /
INTERRUPTED.#/

/ERROR PRINTOUT ROUTINE
JTERR,    AND MR16
          TOS
           JMP JTTER
          JMP TYIBM1   /GO HALT USER ON P 28 OR SO

/IOP ERROR ROUTINE
JIOPER,   LIA
          LAC (JIOPRT)
          TOS
C64,       64          /CONSTANT 64 IN TT ERROR RETURN
          Ø
JIOPRT:   TEXT /
JBH FASTR ERROR.#/


/SUBR TO COPY QN INTO B,C
GQUES,    DAP H
          LAC I (JQN)
          DAC B
          LIO I (JQN+1)
          DIO C
          JMP HEXIT
```

```
/INIT IOT
INIT,       DZM I (JLONT)
            LAC (.REASK)
            DAC I (TTTSU)  /FOR THE BENEFIT OF THOSE WHO TOS OR TIS
            LAW 2
            ADD I (JBUFP)
            DAC I (JTBUFP) /WHERE TO PUT TEXT
            CMA
            DAC I (JDTR)
            LAC I (JBUFP+1)
            DAC I (JTTOCP) /IN WHICH SEG TO PUT TEXT
            DAC A
            DZM I A      /DRA OF SEG
            DAC I (JTBUFC) /WHAT'S IN CORE
            ADD (JMP-13)
            DAC I (TISMAX) /FIR TISMAX RECOVERY
            LAW 2
            ADD I (JBUFP+3)
            DAC I (JQBUFP) /WHERE TO PUT QN ENTRIES
            LAC I (JBUFP+4)
            DAC I (JQTOCP) /IN WHICH SEG TO PUT QN ENTRIES
            DAC A
            DZM I A      /DRA OF SEG
            DAC I (JQBUFC) /WHAT'S IN CORE
            IDX I (USERPC)
            DAC I (JORG)   /WHERE TO RESTART
            CLA          /IN NORMAL MODE
            JMP INIT1    /RESTART
```

```
/KILL ^ RESTART IOT
GTYOQF,   LAC B        /FULL FAIL ENTRY FROM GTYOQ IOT
          DAC I (JQN) /RESTORE QN
          DIO I (JQN+1)
          LIO (74)     /BITS FOR FULL FAIL
KILLFI,   LFI          /FALL INTO KILL IOT. ENTRY FROM →ROUTINES
/IOT ENTRY
KILL,     SZF 2
          JMP KILL1
          SZF I 1      /PF 1, NOT PF2, RECONSIDER
          JMP RESTART /PF1^2 CLEAR, RESTART ONLY

KILL12,   SZF 3        /GET QN TO KILL OR RECONS. WHERE?
          JMP KILLG1   /AC,IO
          IDX I (USERPC) /CALLING SEQUENCE
          DAC C
          LIO I C
          IDX I (USERPC)
          DAC C
          LAC I C
          JMP KILLG2

KILLG1,   LIO I (USERAC) /GET QN FROM AC,IO
          LAC I (USERIO)
KILLG2,   DIO B
          DAC C

KILL6,    JSP SEARCH  /GET QN TABLE SEG IN CORE, POINT E AT QUES PTR
           JMP RESTART    /JUST RESTART IF NO SUCH NUMBER
          LAC I (JMODE)
          SZF I 2
          JMP KILL11  /RECONS ON CLEAR PF2 (PF1 SET IF HERE)
          SZF 1       /KILL ON CLEAR PF1 (2 SET)
          SAS ML17    /FAIL, REASKS GET HERE, RECONS IF SIMULATING
          JMP KILL1Ø
KILL11,   LAC I E     /RECONSIDER
          IOR BØ
          DAC I E
          SKP I
KILL1Ø,   DZM I E     /KILL

RESTART,  LAW I 3
          SZF I 4
          JMP REST6
          SAD I (JMODE)
          DZM I (JMODE)  /FLUSH NULL MODE
          SZF 5       /RESTART WHERE?
          JMP REST1   /FROM GQN OR CALLING SEQ
          LAW 3       /FROM INIT
          SZF I 6     /DETERMINE MODE TO REST IN (3 FOR LIST IN AC)
          LAW I 1     /SIM MODE
INIT1,    DAC I (JMODE)
          LAC (LAC JQN)  /RESTORE QN TO Ø,Ø ^ LEVEL
          DAC I (JLEVEL)
          DZM I (JQN)
          DZM I (JQN+1)
          LAC I (JORG)   /GO BACK TO INIT
          JMP GO
```

```
KILL1,    SZF I 1       /COME HERE IF PF2 SET
          JMP KILL12    /KILL FROM AC,IO OR CALLING SEQ
          SZF I 3       /CURRENT QN: TYPE FIX FIRST?
          JMP KILL5
          LAC I (JMODE)  /FAIL IOT: TYPE "FIX" OR "FIX:"
          LIO (JTXFIX)
          SPA           /DEPENDING ON JMODE
KILL13,   LIO (JTXFXC)   /ENTRY FOR UNANS Q, NOT NULL MODE
          SAS ML16      /-3.  DON'T TYPE IN NULL MODE
          JSP JTOS      /TYPE-OUT ROUTINE. PTR IN IO.
KILL5,    JSP GQUES     /SETUP CURRENT QN IN B,C
          JMP KILL6

JTXFIX:   TEXT / FIX#/
JTXFXC:   TEXT /
FIX:#/



REST6,    SZF I 5
          JMP R1        /NO RESTART ON RESTAR+ 0,1
          DAC I (JMODE)  /RESTAR + 2,3 ENTER NULL MODE
REST1,    LAC I (JLGQN)
          SZF 6
          JMP REST3     /RESTART AT BEGINNING OF JOB
          JSP TRACE     /RESTART FROM CALLING SEQUENCE
          IDX I (USERPC)
          DAC C         /QN(1)
          LAC I C
          DAC I (JQN)
          IDX C         /QN(2)
          LAC I C
          DAC I (JQN+1)
          LAC A         /WHERE TO GO
REST3,    AND MR12      /GET RID OF EOM IN JLGQN
          DAC I (USERPC)
          DAC C
          LAW 77
          AND I C
          LIA
          LAW JQN       /SET UP LEVEL (FOR RESTART FROM CALL)
          DAC C
          IDC"U"LFI
          DAC D
REST2,    LCH I C
          LCH I D
          SZA           /SEARCH FOR 0
          JMP REST2
          LAC C         /POINTS TO LAST NON-0 BYTE (OR FIRST ONE)
          DAC I (JLEVEL)
          JMP GTYOQ9    /GO TO GQN
```

```
/GTYOQ IOT.  UPLV, DNLV, XDNLV PART:
GTYOQ,      JSP GQUES    /SAVE OLD QN. NOTE THAT 2ND WD IN IO.
            LAC I (JLEVEL)
            DAC A        /SETUP LEVEL PTR IN A
            SZF I 2      /SET FOR UPLV, DNLV
            JMP GTYOQ5   /CLEAR FOR NONE, XDNLV
            SZF I 1      /SET FOR XDNLV, DNLV
            JMP GTYOQ6   /CLEAR FOR UPLV, NONE
GTYOQ7,     CLF 1        /FOR NEXT TIME AROUND (XDNLV))
            LAC A
GTYOQ8,     IDC          /DNLV (OR END OF UPLV)
            SAS (JMP JQN-1)              /ERROR ON UPLV
            SAD (LAC JQN+2)              /ERROR ON DNLV
            JMP GTYOQF
            DAC A
            DAC I (JLEVEL)

/GQN PART; SOME ALSO USED BY XDNLV
GTYOQ5,     SZF 1        /FOR XDNLV, GO INCR ON CURRENT LEVEL
            JMP GTYQ51
            SZF I 3      /GQN BIT?
            JMP TYOQ
GTYQ51,     LCH A        /INCR QN
            ADD B5
            SAD B17
            JMP GTYOQF   /OVERFLOW
            DCH A
            SZF 1
            JMP GTYOQ7   /XDNLV
GTYOQ9,     LAC I (USERPC) /SET LOCATION OF LAST GQN
            IOR ML4      /AN EOM TO TYPE IN FOR NULLING
            DAC I (JLGQN)
            LAC I (JMODE)  /SAVE MODE BEFORE GENERATING NEW MODE
            DAC I (JOJM)   /REFERENCED IN TYOQ
            JSP GQUES    /COPY QN INTO B,C
            JSP SEARCH   /LOOKUP CURRENT QN
            JMP QNEWG    /GO MAKE NEW ENTRY
GTYOQ4,     JSP TRACE    /GET USER'S QUESTION POINTER
            LAW 7777
            AND A
            DAP I (JOJM)   /SAVE IN JOJM
            LIO I E      /OLD QUES PTR TO IO
            DAC I E      /NEW ONE TO TABLE
            XAI          /TEST IDENTITY OF ADDRESSES
            AND MR12
            SZF 3
            SZA
            JMP QNOT     /DIFFERENT PLACE OR NEW QUESTION
```

```
/SET UP MODE FOR PREVIOUSLY ASKED QUES - SIM, LIST, OR TOC
GMODE,    LAW I 7777
          NAI
          SZA I        /ANY BITS SET ON QUES PTR FROM TABLE?
          JMP GMOD3
          CLC
          DIP I (JQBUFC) /YES, MARK BUFFER MODIFIED
GMOD3,    LAW 1
          SPI          /RECONSIDER BIT SET?
          JMP GMOD4
          RIL 1S
          LAW 3
          SAS I (JMODE)  /IN LIST MODE?
          SPI          /OR LIST START BIT?
          JMP GMOD2    /YES, ENTER LIST MODE ^ TEST STOP BIT
          LAW I 1      /NOT IN L MODE, NO L START OR RECONS BIT,
                       /SO ENTER SIM MODE
GMOD4,    DAC I (JMODE)  /SET MODE
          IDX E
.TYOQ,    LAC I E      /NEW QN COME HERE
          DAC I (JDANS)  /MOBY PTR TO PREVIOUS ANS OR -0
```

```
/TYOQ PORTION OF GTYOQ IOT
TYOQ,     LAC I (JMODE)
          SZF 4          /TYOQ BIT OF IOT?
          SPA            /TEST FOR SIM OR NULL MODE
          JMP R1         /NO TYPE
          LAC I (JOJM)
          SPA            /WAS PREVIOUS QUESTION IN SIM. MODE?
TYOQ1,    CLF 7          /YES, CLEAR FLAGS 1^2
          LAC ML5
          SZF 5          /SUPPRESS CR BIT
          JMP TYOQ13
          TYO            /TYPE CR
           JMP JTTER     /TYO ERROR
TYOQ13,   SZF 6          /SUPPRESS QN BIT
          JMP TYOQ5B
          SZF 5          /SUPPRESS INDENT (CR)
          JMP TYOQ3
          LAC (LAC JQN)  /BEGIN ROUTINE TO INDENT
          DAC F
TYOQ2,    LCH I F
          SZA I
          JMP TYOQ3      /FINISHED INDENTING, GO TYPE QN
          JSP SPAC       /TYPE A SPACE
          JMP TYOQ2

TYOQ3,    LAW JQN
          DAC F
          LIO (11)       /FLAGS 3^6 FOR SNM
          LFI
          LCH I F
          JMP TYOQ12     /SO INITIAL 0 PRINTS

TYOQ9,    SUB B5         /ALPHA PRINT #
          RCL 7S         /IO WAS CLEAR
          DIV C26.       /LETTERS IN ALPHABET
C26.,      26.
          IDA
          CMA            /SETUP FOR ISP
          DAC H
          LAW CHARAC RA
          AAI
          RCR 6S         /CHAR INTO TOP(IO)
TYOQ10,   LAI
          TYO
           JMP JTTER
          ISP H
          JMP TYOQ10     /RE-TYPE LETTER
TYOQ4,    LCH I F
          SZA I
          JMP TYOQ5      /FINISHED, GO TYPE QUESTION
TYOQ12,   CML"U"SCI"U"SZL          /ENTRY. CHANGE ^ TEST ALPHA FLAG
          JMP TYOQ9      /IO CLEAR
          RCL 6S         /# TO BE TYPED TO IO
          LAW TYOQ4
          JMP .SNMJ      /CALL SNM WITH RETURN TO TYOQ4
TYOQ5,    JSP SPAC       /SPACE AFTER QN
```

```
                        /ROUTINE TO TYPE QUESTION
TYQQ5B,     LAC I (JQN) /DOES THIS QN = THE LAST 1 TYPED?
            STF 1       /SIGNIFIES NOTHING MODE
            LIO I (JBF)
            SAS I (JLQNT)
            JMP TYQQ5A
            LAC I (JQN+1)
            SAS I (JLQNT+1)
TYQQ5A,     SNI
            JMP TYQQ8   /IF QN= LAST QN TYPED, OR JBF=Ø
            LAW 7777
            AND I (JOJM)
            DAC A       /PTR TO SHORT MODE PTR
            LAW 1
            SAS I (JBF) /UNLESS SHORT MODE,
            IDX A       /MAKE I LONG
            LIO I A
            JSP JTOS    /TYPE QUESTION

            CLF 1       /MEANING NOT IN NOTHING MODE
TYQQ8,      JSP SPAC    /COME HERE FOR NOTHING QUESTION, PF1 SET
            LAC I (JQN) /MARK QN AS LAST 1 TYPED
            DAC I (JLQNT)
            LAC I (JQN+1)
            DAC I (JLQNT+1)
            LAW 3
            SZF I 1     /IF TYPING NOTHING QUESTION,
            SAS I (JBF) /OR NOT IN HOW MODE,
            JMP R1      /DONE
            IDX A
            LIO I A
            JSP JTOS    /TYPE HOW MODE
            JMP TYQQ1   /GO TYPE NOTHING QUESTION


/REMOTE ROUTINE FOR MODE SETTING
GMOD2,      RIL 1S      /3 IN AC FOR LIST CONTINUE MODE
            SPI         /IF LIST STOP BIT,
            LAW 2       /MAKE IT LIST STOP MODE
            JMP GMOD4

/REMOTE ROUTINE TO DO UPLV
GTYQQ6,     CLA
            DCH A       /CLEAR VALUE AT OLD LEVEL
            LAW I 1
            ADD A       /UNSTEP A
            IDC
            JMP GTYQQ8
```

```
/ROUTINE TO MAKE NEW QN TABLE ENTRY
/QN FROM B,C; LINK MUST BE SET

QNEWG,    LAC I (JQBUFP)
          SAS I (JBUFP+4)          /AT END OF BUFFER?
          JMP QNEW1   /NO
          IDX I (JQTOCP) /NEW SEGMENT
          SAD I (JBUFP+5)          /OUT OF ROOM?
          JMP QNEWBF  /TYPE ERROR ^ TRAP
          DAC E
          DZM I E     /NO DRA
          LAW 2
          ADD I (JBUFP+3)          /NEW CORE ADR FOR TABLE
QNEW1,    DAC E       /PTR TO BEG OF 4 WD TABLE
          ADD B15     /(4), NEXT TABLE
          DAC I (JQBUFP)
          LAC I (JQTOCP) /CURRENT SEG PTR
          DAC F
          JSP READTQ  /INTO CORE (LINK LEFT SET)
          LAC B       /QN TO TABLE
          DAC I E
          IDX E
          LAC C
          DAC I E
          IDX E
          CLF 3       /MEANING NEW QUESTION
          JMP GTYQQ4

QNEWBF,   JSP JTERR
          TEXT /
TOO MANY QUESTIONS.
#/


/STUFF FOR QUES NOT ASKED BEFORE OR KILLED
/OLD QUES PTR IS IN IO

QNOT,     IDX E
          LAC I E     /OLD ANS PTR
          SAS PTS     /IF ANSWERED BEFORE,
          SNI"U"SZF I 3  /AND ASKED AND KILLED,
          JMP QMOD1
          LAW 4       /THEN ENTER DEAD MODE
          JMP QMOD2
QMOD1,    LAC I (JMODE)  /OTHERWISE,
          SAD MR16    /CONTINUE NULLING, OR
          CLA         /ENTER NORMAL MODE
QMOD2,    DAC I (JMODE)
          CLC
          DAC I E     /-0 ANS PTR
          DIP I (JQBUFC) /MARK BUFFER CHANGED
          JMP .TYOQ
```

```
/IOT TO CONVERT MOBY PTR INTO CORE PTR ^ GET INTO CORE
GTEXT,    LAC I (USERAC)
          DAC D
          LAW R1
/SUBR ENTRY
.GTEXT,   DAP H
          LAW I 2
          SUB I (JBUFP)
          ADD I (JBUFP+1)              /CALCULATE BUFFER LENGTH
          DAC F
          LAC D          /GET MOBY PTR
          CLL"U"SCI      /CLL FOR READTQ
          RCL 2S
          RAR 1S         /REMOVE TOP 2 BITS
          CLI"U"SWP      /POSITION FOR DIVIDE
          DIV F
C67,       67            /CONSTANT 67
          ADD I (JBUFP+1)              /ADD BASE OF TOC TO QUOTIENT
          DAC F          /SET UP FOR READTQ
          JSP READTQ     /GET THIS BUFFER INTO CORE
          LAC D
          SUB I (JBTR)
          DAC I (FSA) /SAVE IN FSA
          DAC D
          JMP HEXIT
```

```
/TYINV IOT
TYINV,     LAC I (JTBUFP)
           SUB I (TISMAX) /SET 13 WORDS BELOW END OF BUFFER BY INIT IOT
           RAL 2S         /BYTE PART OF PTR OUT OF THE WAY
           SMA            /IS PRESENT TEXT SEG FULL ENOUGH
           JSP JNTS       /YES, START ANOTHER
           LAC I (JTBUFP)
           DAC I (JOJM)   /CURRENT PTR TO END OF TEXT
           JSP GQUES      /GET CURRENT QN IN B,C
           JSP SEARCH
            C16RET+1      /ILLEGAL SEQUENCE OF IOTS
           LAC I (JQBUFC) /0 BUF SEG
           DAC I (ATEM+5) /SEE TYIBC+24
           IDX E          /GET PTR TO ANSWER
           STF 1          /FOR .EDITJ

/CHECK JMODE, TYPE OLD ANS AND THINGS IF NECESSARY
TYINV1,    SZF 4
           JMP TYINN      /FLAG 4 FORCES TYPEIN IN ALL CASES
           LAC I (JMODE)
           SAS ML16       /-3
           JMP TYINV3
           LAW JLGQN      /NULL MODE. GO TYPE EOM
           STF 6          /IN FROM CORE
           JMP TYINNN

TYINV3,    AND C3         /AND WITH 3 TO MAKE 4 LOOK LIKE 1
           SZF 6
           AND B16        /GET RID OF TOC MODE
           SZA I
           JMP TYINN      /NORMAL MODE; TOC MODE IF TYPEIN FROM CORE
           LAC I E        /ANSWER POINTER
           SAD PTS        /(-0)
           JMP TYINVK     /NOT ANS BUT NOT NORMAL - FAIL.
           DAC I (JDANS)
           DAC D
           JSP .GTEXT     /RETRIEVE ANSWER
           LIO D
           DIO I (JANS)

TYINV2,    LAC I (JMODE)
           SZF I 6
           SPA
           JMP TYIV       /DONE HERE IF IN SIM MODE OR FROM CORE
                          /LIST, TOC GET HERE
           JSP JTOS       /TYPE OLD ANSWER
           LAW 1
           SAS I (JMODE)
           JMP TYIV       /DONE IN LIST MODE
           JSP SPAC       /TOC MODE; SPACE
                          /FALL THROUGH TO GET NEW ANSWER
```

```
/TYPE-IN SECTION  FROM CORE ON PF6
/R1 RUBOUT (# TYPED) OR TOO LONG
/R2 WIN  JTBUFP POINTS TO BEG OF TEXT
/        JOJM TO END
/TRANS TO USERAC

/STORAGE
/A  SUBR RETS
/B  DCH I PTR FOR TEXT
/C  PTR TO BEG TEXT; PTR FOR COPY-BACK
/D  TEXT PTR ON PF6; OW TRANS
/E SAVED IN ATEM+4 (PTR TO ANS PTR)
/F  ARG FOR READTQ
/G  SUBRS USE

TYINN,     LAC I (USERAC) /FOR TYPE IN FROM CORE
TYINNN,    DAC D
           CLL
           LAC I (JTTOCP)
           DAC F
           JSP READTQ  /GET TEXT SEG BEING FILLED
           LAC I (JTBUFP)
           DAC B       /DCH I POINTER FOR TEXT TYPED IN
JRBTY1,    DAC C       /PTR TO BEGINNING FOR COPYBACK
           SZF 6
           JMP JTYIEL  /FROM CORE
           LAC B
           TIS
            JMP JTYIER /ERROR; IF TISMAX, START NEW BUF

JRBTY5,    LAC E       /SAVE PTR TO CURRENT ANS PTR
           DAC I (ATEM+4) /THROUGH EDIT AND →C
           SZF 6
           JMP JRBTY4
           LIO I (JTBUFP) /TEXT PTR FOR EDIT
           JH4IEJTY4,   DIO I (JOJM)   /PTR TO END OF TEXT
           LAC I (ATEM+4) /RESTORE PTR FOR →.
           DAC E       /NOT NEEDED IF EDIT CHANGED
           LCH I C     /LOOK FOR →
           RCL 6S
           LCH I C
           RCR 6S
           SAD (FLEXO → )
           JMP TYINBA  /HANDLE →
TYIBR,     LAC I (JTBUFP)
TYIBRB,    DAC I (FSA)
TYIBRC,    DAC I (JANS)
           ADD I (JDTR)
           DAC I (JDANS)
           LIO I (ATEM+4) /RESTORE PTR TO (OLD) ANSWER PTR
           DIO E
           SAS I E     /IS NEW PTR THE SAME (AS → AFTER RECONS)
           CLF 1
           DAC I E     /UPDATE PTR IN ON TABLE
           CLC
           SZF I 1
           DIP I (JQBUFC) /BUFFER IS CHANGED IF PTR CHANGED
```

```
/VERIFY AND FINISH UP
TYIV,     SZF I 5
          JMP TYIN     /DON'T VERIFY
          JSP TRACE    /GET SYNDEF ADDRESS IN A
          LIO A
          JSP .STVJ
           JMP TYINVF /LOSE
TYIN,     LAC I (JOJM)
          SUB I (JTBUFP)
          CLI"U"CMI"U"SWP
          SNI I
          DIP I (JTBUFC) /MARK TEXT BUF CHANGED IF PTR CHANGED
          LAC I (JOJM)   /MUST CONTAIN POINTER TO END OF TEXT
          DAC I (JTBUFP)
          JMP R1

TYINVK,   DZM I (JMODE)  /PREVENTS FIX: FIX ON UNANSWERED QUES
          STF 7
          JMP KILL13  /TYPE "FIX:" AND REASK

TYINVF,   LIO C77     /TYPE FIX AND REASK
          JMP KILLFI

TYINNR,   LAC MR14    /#
          TYO
           JMP JTTER
TYINNI,   LIO C67
          JMP KILLFI  /REASK
```

```
/TYPE IN FROM CORE AND TIS ERRORS
/NOTE THAT TISMAX IS 13 WORDS BELOW END OF BUFFER

JTYIER,   DAC B        /TIS ERROR.  SAVE POINTER
          LAW 3
          SAS I (ERCODE)
          JMP JTTER    /NOT A TISMAX ERROR
JTYIEL,   CLI          /COPY ANSWER FROM CORE AND TISMAX ERR RUTIN
          SZF I 6
          TYI 2-1      /EMPTY TT BUFFER AND DON'T HANG
          SZF 6
          LCH I D      /FROM CORE
          SPI
          JMP JTYIEH   /SHOULD HAVE HUNG. MAKE NEW BUF WHILE WE CAN
          DCH I B
          LIO B        /POINTER TO END, NEEDED IF EOM FROM CORE
          SAD (CHARAC R#)
          JMP JRBTY5   /EOM. DONE.
          LAC (JMP-1)
          ADD I (JBUFP+1)
          SAS B        /BUFFER COMPLETELY FULL?
          JMP JTYIEL   /NO, LOOP

/CREATE NEW TEXT SEG AND COPY TEXT BACK TO BEGINNING
JTYIEH,   LAC B
          DAC I (JOJM)    /SAVE POINTER TO END OF TEXT
          LAW 2
          ADD I (JBUFP)
          SAD C        /PTR TO BEG OF TEXT TYPED IN
           JMP JTYIEF /STARTED AT BEGINNING OF BUFFER, ANS TOO LONG
          DAC B        /PTR TO BEG OF ANSWER
          JSP JNTS     /SET UP POINTERS FOR NEW SEG
          JSP READTQ   /TO WRITE CURRENT SEG ON DRUM IF NECESSARY

JTYIEM,   LCH I C
          DCH I B      /COPY TEXT DOWN TO BEGINNING OF BUFFER
          LAC C
          SAS I (JOJM)
          JMP JTYIEM
          LAC I (JTBUFP) /NEW VALUE OF PTR TO BEGINNING
          JMP JRBTY1

JTYIEF,   SZF 6        /ANSWER TOO LONG
          JMP JTYIFC
          LIO (JTYIFT)
          JSP JTOS
          JMP TYINNI
JTYIFT:   TEXT /
ANSWER TOO LONG.#/

JTYIFC,   JSP JTERR
          TEXT /
INTERNAL ANSWER TOO LONG#./
```

```
/SUBR TO SET UP PTRS FOR NEW TEXT BUF SEG
JNTS,       DAP A
            LAW 2
            ADD I (JBUFP)
            DAC I (JTBUFP) /POINTS TO BEGINNING OF TEXT AREA
            IDX I (JTTOCP) /NEXT WORD IN TOC
            DAC F
            SAD I (JBUFP 2)
             JMP JNTSE  /ALL SEGS USED
            DZM I F     /0 DRA SINCE NEW
            JMP AEXIT
JNTSE,      JSP JTERR
TEXT /
TOO MUCH TEXT.
#/
```

```
/DECODE QUESTION NUMBER ROUTINE FOR → INTERPRETATION
DQN,    DAP H
        DZM I (ATEM)
        DZM I (ATEM+1) /FOR BUILDING UP QN
        LAW ATEM
        DAC F       /LEVEL PTR
        DZM D       /LEVEL COUNTER, EXCLUSIVE OF INITIAL Ø
        LCH G
        SAD B1      /(CHARAC LØ)
        JMP DQN8    /HANDLE NUMBERS LIKE ØA

DQN1,   CLL         /DECODE A NUMERIC LEVEL. LINK SET AFTER A DIGIT
        DZM B
        LCH G
        SAD B1      /(CHARAC LØ)
        JMP DQNO    /CAN'T BEGIN WITH Ø AT PRESENT
DQN3,   RAL 6S
        XOR B13
        DAC C
        SUB C1Ø.    /(1Ø.)
        SMA
        JMP DQN2
        LAW 1Ø.
        MUL B
        SIR 1S
        LAC C
        AAI"U"CLL"U"CML             /ACCUMULATE VALUE IN HI BITS
        DAC B       /OVERFLOW CARRIES AROUND TO LO BITS
        LCH I G
        JMP DQN3

DQN2,   SZL I
        JMP DQNO    /NO NUMBER
        LAC B
        RAR 6S

DQN7,   DCH I F     /STORE VALUE
        SAR 6S
        SZA         /OVERFLOW CHECK
         JMP HEXIT  /R1
        IDX D       /HAVE A GOOD LEVEL
        LAC F
        SAD (LAC ATEM+7)            /NUMBER OF LEVELS CHECK
         JMP HEXIT  /R1 LOSE
        SZL I       /LINK CLEAR IF NEXT FIELD NUMERIC
        JMP DQN1
        JMP DQN6

DQN8,   LCH I F     /STEP LEVEL FOR INITIAL Ø
        LCH I G     /AND STEP CHAR PTR
```

```
DQN6,   LCH G       /DECODE ALPHA LEVEL
        DAC B
        SUB (CHARAC LZ+1)
        LIA
        ADD (CHARAC LZ+1-CHARAC LA)
        SPI"U"SMA I /MUST BE LETTER
        JMP DQNO    /ISN'T
        DZM C       /LETTER COUNTER
        JMP DQN5
DQN4,   IDX C
DQN5,   LCH I G
        SAD B
        JMP DQN4    /COUNT # OF SAME LETTERS
        LAW 26.     /CONVERT TO NUMBER
        MUL C
        RIR 7S      /VALUE FOR ALL BUT FIRST LETTER TO HI IO
        LAC B
        XOR BØ      /VALUE FOR FIRST LETTER
        AAI"U"CLL
        JMP DQN7    /GO STORE VALUE, CHECK FOR OVERFLOW

DQNO,   LAC I (ATEM)   /DONE
        DAC B       /LEAVE NUMBER IN B,C
        LAC I (ATEM+1)
        DAC C
        LAC D       /ILLEGAL IF NO LEVELS (NOT A DIGIT FIRST, OR Ø ONE
        SZA         /R1
        IDX H
        JMP HEXIT
```

```
/SUBR TO TYPE DIAGNOSTIC ERRORS (QN AS TYPED IN)
/WITH MANY ENTRIES FOR VARIOUS ERRORS
TYIBCI,   CLL
          LAW TYIREA
TYIBIS,   LIO (TYIBIT)
TYIS,     DAP A        /MAIN ENTRY. TEXT ADR IN I0, RET IN AC
          LAC I (ATEM+2)
          DAC G
          LAC ML5
TYISL,    TYO          /TYPE CRLF, QN
          JMP JTTER
          LCH I G
          SAD ML4
          CLF 1
          SZL          /TERMINATE ON , IF LINK SET ONLY
          SAS (CHARAC L,)
          SAD ML4      /CHARAC L#
          JMP TYISC
          SZF 3        /TERMINATE ON - IF PF3 SET
          SAS (CHARAC L-)
          JMP TYISL
TYISC,    LAC G
          DAC I (ATEM+2)
          LCH I G
          JSP JTOS     /TYPE COMMENT WHOSE ADDRESS CAME IN I0
          JMP AEXIT

TYIBIT:   TEXT / IS NOT A LEGAL QUESTION NUMBER.#/

TYIBCA,   CLL
          LAW TYIREA
TYIBAS,   LIO (TYIBAT)
          JMP TYIS
TYIBAT:   TEXT / HAS NOT BEEN ANSWERED.#/

TYIBCK,   CLL
          LAW TYIREA
TYIBKS,   LIO (TYIBKT)
          JMP TYIS
TYIBKT:   TEXT / HAS NOT BEEN ASKED.#/
```

```
/BACK-ARROW INTERPRETING ROUTINES
/DISPATCH ON CHARACTER AFTER →
TYINBA, CLF 4
        CLI"Ü"CLF 6
        LAC C
        IDC             /STEP PTR BEYOND CONTROL LETTER
        DAC G
        DAC I (ATEM+2) /SAVE CORE PTR TO ANSWER
        LCH G
        SAD (CHARAC LC)
        JMP TYIBC
        SAD (CHARAC LH)
        JMP TYIBH
        SAD (CHARAC L/)
        JMP TYIBS
        SAD (CHARAC L-)
        JMP TYIBM
        SAD ML4         /CHARAC L#
        JMP TYIVOK
        SAD (CHARAC LN)
        JMP TYIBN
        SAD (CHARAC LL)
        JMP TYIBL
        SAD (CHARAC LR)
        JMP TYINB8
        SAD (CHARACTER LK)
        JMP TYINB6
        LIO C           /→NUMBER WITHOUT CONTROL LETTER, UNSTEP PTR
        DIO I (ATEM+2)
        JMP TYINB9  /TREAT AS →R
TYINB6, STF 6           /PF6 FOR KILLING
TYINB8, LCH I G
        JMP TYINB9
```

```
/→L(LIST) AND →#,→R,→K (RECONS AND KILL) INTERPRETING ROUTINE
/ACCEPT A LIST OF NUMBERS SEPERATED BY , AND - ;  MARK ON TABLE ENTRIES

/FLAGS: 1, CLEARED IF EOM SEEN; 2, CHANGE MADE, FULL RESTART;
/3, →L NOT →# ETC. (AFFECTS TYIS)
/  4, RESTART IN LIST MODE; 5, HAVE SEEN A - AND MAYBE A # SINCE , ;
/  6, →K NOT →R
/STORAGE:
/  ATEM+2: CORE PTR TO BEG CURRENT ON IN TEXT BEING INTERP
/    +3: PTR TO END OF SAME
/    +6: ON TOC PTR FOR QUESTION WHICH IS TO GET START BIT
/    +7: CORE POINTER TO QUESTION TEXT PTR IN SAME
/  E: PTR INTO ON TABLE, PARTICULARLY TO ENTRY TO GET STOP BIT
/  G: BYTE PTR OVER TEXT BEING INTERPRETED


TYIBL5,   STF 4        /COME BACK HERE ON INITIAL COMMA
TYIBL,    STF 3        /ENTRY FOR →L
          LCH I G
          SAD (CHARAC L,)
          JMP TYIBL5
          SAD ML4
          JMP TYIBLU   /EOM, GO RESTART
          SAD (CHARAC L-)
          JMP TYIBL8
TYINB9,   RAL 6S       /ENTRY FOR →#, →R, →K
          XOR B13      /DIGIT?
          SUB C10.     /(10.)
          SMA
          JMP TYIBR    /NO, FALL THROUGH TO USER
TYIBL6,   CLF 5        /ERRORS COME BACK HERE
          JMP TYIBL7

/MAIN LOOP
TYIBLE,   STF 4        /COME HERE ON - AT BEGINNING OF "ELEMENT"
TYIBLJ,   LIO G        /AND HERE ON - AFTER #
          STF 5
          LCH I G
          SAD (CHARAC L,)
          JMP TYIBLF
          SAS ML4
          JMP TYIBL4   /LOOK FOR ANOTHER NUMBER (PF5 SET)
          CLF 1
TYIBLF,   DZM E        /NO QUES IS TO BE MARKED STOP
TYIBLL,   LAC G
          DAC I (ATEM+3) /SAVE THRU READTQ
TYIBL9,   CLA"U"CLF 5 /LOOP ENDS WITH FOLLOWING
          SAD E
          JMP TYIBL1   /0, NO QUES TO PUT STOP BIT ON
          LAC B0       /RECONS BIT
          SZF 3
          LAC B2       /LIST STOP BIT
          IOR I E
          SZF I 3
          SZF I 6
          SKP I
          CLA          /KILL ON PF6 AND NOT PF3
          DAC I E      /SET ON THIS QUES
          CLC"U"STF 2
          DIP I (JQBUFC)
```

```
TYIBL1,   LAC I (ATEM+6) /MARK START QUES IF ANY
          SZA I
          JMP TYIBL2
          DAC F
          JSP READTQ  /GET QN SEG INTO CORE
          LAC I (ATEM+7)
          DAC E
          LAC B1      /MARK IT LIST START
          IOR I E
          DAC I E
          CLC"U"STF 2
          DIP I (JQBUFC) /MARK SEG CHANGED
/ENTRIES TO LOOP
TYIBL2,   LIO I (ATEM+3) /RESTORE G ETC
TYIBL4,   DIO I (ATEM+2) /PTR BEG THIS # FOR TYIS
          LAI
          IDC
          DAC G
TYIBL7,   LCH G
          SAS ML4     /NEEDED FOR →L#-EOM WHERE # FAILS
          SZF I 1     /CLEARED WHEN EOM SEEN (OTHER CASES)
          JMP TYIREA  /GO RESTART IF PF1 CLEAR
          SZF 5
          JMP TYIBLQ  /IF LOOKING FOR A # AFTER A -
TYIBL8,   DZM I (ATEM+6) /NO # TO MARK START
          SZF I 3
          JMP TYIBLQ  / - LEGAL ONLY IF →L
          SAD (CHARAC L-)
          JMP TYIBLE  /LIST "ELEMENT" STARTS WITH -
TYIBLQ,   CLL"U"CML   /LINK SET FOR TYIS
          JSP DQN     /DECODE QUESTION NUMBER
           JMP TYIBLI /FORMAT ERROR
          LAC G
          DAC I (ATEM+3)
          LCH G
          SZF 3
          SZF 5
          JMP TYIBLS  /SECOND - NOT ALLOWED
          SAD (CHARAC L-)
          JMP TYIBLM  / # FOLLOWED BY -
TYIBLS,   SAD (CHARAC L,)
          JMP TYIBLM
          SAS ML4
           JMP TYIBLI /FORMAT ERROR
          CLF 1
TYIBLM,   JSP SEARCH
           JMP TYIBLK  /NO ENTRY
          LAC E
          DAC D
          IDX D
          LAC I D     /GET ANSWER PTR
          SAD PTS
           JMP TYIBLV  /NOT ANSWERED.
          SZF 3       /ON →# ETC,
TYIBLB,   SZF 5       /OR IF THIS WAS THE # AFTER - ,
          JMP TYIBL9  /GO TO THE END OF THE LOOP, MARK THIS QUES STOP
```

```
        LIO I (ATEM+3) / # AT START OR AFTER , IN -L IF HERE
        DIO G
        LAC E
        DAC I (ATEM+7) /SAVE PTRS FOR MARKING ON ENTRY
        LAC I (JQBUFC) /  AFTER WHOLE LIST EL. VERIFIED
        DAC I (ATEM+6)
        LCH G
        SAD (CHARAC L-)
        JMP TYIBLJ
        JMP TYIBL9  /LOOP BACK. NOTE THAT E NOT ZEROED, SO
                    /THIS QUES WILL BE MARKED BOTH START AND STOP

TYIBLI, LAW TYIBL6  /ILLEGAL FORMAT
        JMP TYIBIS  /TYPES # AND COMMENT

TYIBLK, LAW TYIBL6  /QUES NOT ASKED
        JMP TYIBKS  /ERROR COMMENT

TYIBLV, LAW TYIBL6  /NOT ANSWERED (EG CURRENT QUESTION)
        JMP TYIBAS
```

```
/→/
TYIBS,    LCH I G
          SAD ML4
          JMP TYIBSN
          SAD (CHARAC LS)
          LIO B17
          SAD (CHARAC LL)
          LIO B16
          SAD (CHARAC LH)
          LIO MR2       /(3) INSERT TAG IF NOT THERE
          LCH I G
          SNI I         /IO Ø MEANS NO ACCEPTABLE CHAR FOUND
          SAS ML4       /INSIST ON EOM
          JMP TYIBR
TYIBSN,   LAC I (JBF)
          DIO I (JBF)
          CMI
          AAI
          SPA
          DZM I (JLONT)  /RETYPE QUESTION IF JBF DOESN'T DECREASE

/→COMMANDS COME HERE TO RESTART
TYIREA,   LIO C67       /(67) FAST RESTART FLAGS
          SZF 2
          LIO C64       /(64). FULL RESTART IF A CHANGE HAS BEEN MADE
          SZF 4
          SZF I 3
          SKP I
TYIBLU,   LIO C65       /IN LIST MODE ON PF3^4 OR INITIAL EOM
          JMP KILLFI    /JMP TO KILL IOT



/→N RESTART FROM GQN IN NULLING MODE
TYIBN,    LCH I G
          SAS ML4       /EOM REQUIRED
          JMP TYIBR
          LIO (63       /RESTART IN NULL MODE
          JMP KILLFI
```

```
/→H
TYIBH,    LCH I G
          RCL 6S
          LCH I G
          RCL 6S
          LCH I G
          AAI          /PUTS 3 CHAR IN AC IN ORDER 3-1-2
          SAS (FLEXO #OW)          /CHECK FOR EXACTLY "→HOW#"
          JMP TYIBR
          JSP SPAC     /SEPERATING SPACE
          LAC I (JLGON)  /SP0+LOC(LAST GON)
          ADD (IOR)    /MAKES PTR TO LOC(LAST GON)+1
          LIA
          JSP TRACEI
          IDX A
          IDX A        /GET HOW PTR
          LIO I A
          JSP JTOS
          JMP TYIREA

TYIBM,    LCH I G
          SAS ML4      /CHECK FOR EXACTLY "→-#"
          JMP TYIBR
TYIBM1,   LAW JHTSU    /USED BY ERROR ROUTINE TOO
          JMP GO

/→#
TYIVOK,   LAW 1
          SAS I (JMODE)  /TOC MODE?
          JMP TYINVF   /NO, ILLEGAL, TYPE FIX, REASK
          LAC I (JTBUFP) /TOC MODE, PRESERVE OLD ANSWER
          DAC I (JOJM)   /SO JTBUFP DOESN'T CHANGE
          LAC I E
          DAC D
          JSP .GTEXT
          LAC D
          JMP TYIBRB
```

```
/→C
TYIBC,    LCH I G
          SAD (CHARACTER LR)
          JMP TYIBCR
          RAL 6S
          XOR B13       /DIGIT?
          SUB C10.      /(10.)
          SMA
          JMP TYIBR     /NO, GIVE IT TO USER
          JSP DQN
           JMP TYIBCI
          LCH G
          SAS ML4
           JMP TYIBCI
          JSP SEARCH
           JMP TYIBCK
          IDX E
          LAC I E
          SAD PTS
           JMP TYIBCA   /NOT ANSWERED
          DAC D         /MOBY PTR
          LAC I (ATEM+5) /SEG
          DAC F
          JSP SPAC
          JSP READTQ    /LINK SET FROM SEARCH
          JSP .GTEXT    /GET OLD ANSWER
          LAC I (JTBUFP)
          DAC I (JOJM)
          LIQ D         /CORE PTR TO OLD ANSWER
          JSP JTOS
          LAC D
          JMP TYIBRC    /RETURN TO TYINV IOT

TYIBCR,   LCH I G       /CHECK FOR →CRASH
          RCL 6S
          LCH I G
          RCL 6S
          LCH I G
          AAI
          SAS (FLEXO HAS)           /ASH BUT OUT OF ORDER
          JMP TYIBR
          LCH I G
          SAS ML4       /IS IT FOLLOWED BY #
          JMP TYIBR
          765432        /EXECUTE ILLEGAL INSTRUCTION
```

```
/TYOC IOT
TYOC,       JSP TRACE
            LIO A
            SZF I 6
            JMP TYOCC
            LIO I (JBF) /JBF DEPENDENT OPTION
            RIR 2S
            SPI
            IDX A
            LIO I A
TYOCC,      LAC I (JMODE)
            SMA
            JSP JTOS    /DON'T TYPE IN SIM OR NULL MODES
            JMP R1


CON,        CONSTANTS
FOO,        FLEXO FOO

START
```

CORE 14 ENTRIES TO CORE 16 11-10-66 (16TO14,21)

```
7600/     TAD DDTGO
          TAD EDIT
          TAD RPB
          TAD ERIM
          TAD PPB
          TAD TTON
          TAD DNM
          TAD DUN
          TAD SNM
          TAD SUN
          TAD TDNUM
          TAD STD
          TAD DTM
          TAD DDT
          TAD DCDTD
          TAD STV
          TAD STVW
          TAD STVL
          TAD BGI
          TAD GET
          TAD HALT

7626/     TAD FLIP
          TAD FADD
          TAD FMUL
          TAD FDIV
          TAD FLAC
          TAD FDAC
          TAD FLOAT
          TAD FLOAT2
          TAD FIX
          TAD FIX2
          TAD FLOP
          TAD SUPGO
          TAD INIT
          TAD GTYOQ
          TAD TYINV
          TAD KILL
          TAD GTEXT
          TAD TYOC
          TAD REASK.

7700/     TAD BKPNT
          OPR BKKEY
          TAD HHORR
          OPR STARTUP
          TAD STARTUP
          TAD PSTART
          TAD C16FIN
          TAD C16FOU
          TAD C16FSW

START HLT-JMP
```

CORE 17 ENTRIES TO CORE 16 (16TO17,2)

```
37720/     LAC LISSUP
           LAC 16LSSP
           TAD C16SWF

START XX-JMP
```