```
MIDAS LINKING LOADER
/10-14-64 WFM, COMMENTS MOSTLY 5-28-64 CRM

6501/
TYO=TYO"U"I RPB=RPB"U"I
PPA=PPA"U"I PPB=PPB"U"I

START
```

```
MIDAS RELOCATING LINKING LOADER
/10-14-64  WFM

EQUALS CHAR,CHARACTER

ENTRY:      LAP
            AND (177777 /GET CORE FIELD THAT LOADER IS IN
            DIP TOP     /SET UP TOP AND BOT SO THEY REFER TO THE SAME C
            DIP BOT
            DIP BLK1
            LAC SHEREJ  /GET ENTRY POINT AND STORE IT IN LOC. 7776-SUBR
            DAC 7776
            LAC SYMPCA
            DAC 7775    /SET ENTRY TO SYMBOL PACKAGE PUNCH
            LAC .+2     /GET ENTRY POINT AND STORE IT IN LOC. 7777-REST
            DAC 7777
            JMP SA
SHEREJ,     JMP SHERE
SYMPCA,     JMP SYMPCH
            0           /CHECK FOR BOTTOM OF THE SYMBOL TABLE
SA,         LAC BOT
            DAC LIST
            HLT"U"CLA"U"CLI"U"CLF 7

SHERE,      SZS 20      /IF SWITCH ON USE TEST WORD TO FIND STARTING LO
            LAT
            AND (177777
            DAC MEMORY  /MEMORY ALWAYS REFERS TO NEXT STORAGE LOCATION
            DAC RFACTOR /RFACTOR IS THE RELOCATION FACTOR
            EEM         /LET US GO TO EXTEND MODE
            SZS 10      /PRINT OUT STARTING POSITION IF SWITCH IS OFF
            JMP BLOCK
            JDA OPT     <- dyn value
            JSP CR

BLOCK,      LAW BLOCK   /BLOCK IS STARTING POINT FOR READING BLOCK
            DAP GWS1    /SET THE END OF BLOC RETURN TO RETURN TO READ A
                        /THIS GWS1 IS CHANGED BY JUMP BLOCK AND LIBRARY
            LAW BTABLE
block+3,    DAP BLJ     /SAVE ADDRESS OF TRANSFER TABLE FOR BLOCKS,CHAN
            LAW LAST
            DAP BLK1
            DAP BLK2
            RPB
            DIO FIRST
            DIO BLK3
            DIO CHECKSUM
BLKA,       RPB
            DIO I BLK1
            LAC I BLK1
            ADD CHECKSUM
            DAC CHECKSUM
            IDX BLK1
            DAP BLK4
            IDX BLK3
            SUB ONE
            SAS LAST
            JMP BLKA
```

```
                RPB
                RCL 9S
                RCL 9S
                SAS CHECKSUM
                HLT
                DZM BITSC
                CLA"U"CLF 7
                RCL 2S
BSW,            ADD BLJ
                DAP BLX
                LAC FIRST
                AND (177777
BLX,            JMP .
BTABLE,         JMP ABLOCK   /ABSOLUTE BLOCK
                JMP RBLOCK   /RELOCATABLE BLOCK
                JMP LBLOCK   /LIBRARY BLOCK
JBLOCK,         STF 4        /JUMP BLOCK-SET FLAG 4 AS AN INDICATOR
                JSP BITS     /GO GET VALUE OF THE JUMP LOCATION
                DAC VALUE    /SAVE JUMP LOCATION IN VALUE
                LAW .+3      /SETUP RETURN FROM GWORD SO THAT CAN CHECK THE S
                DAP GWS1
                JMP GWORD+2
LJP,            SZS 10       /IF OFF PRINT OUT THE LAST LOCATION PPROCESSED
                JMP SKPP
                LAC MEMORY
                JDA OPT
                JSP CR
                JSP CR
SKPP,           LAC LIST     /SET UP TO SEARCH THE TABLE FOR UNDEFINED SYMBOS
JBACK,          DAP QLAC
                LAC MEMORY   /SET UP COMPUTER PANEL FOR HALT
                LIO VALUE
                SZS I 60     /IF SWITCH 6 ON SKIP THE HALT
                HLT"U"CLF 7 /NORMAL HALT FIRST TIME-SAME HALT AFTER UNDEFIN
                JSP QD       /GO SEARCH TABLE
                JMP JFOO     /UNDEFINED SYMBOL
                LAC MEMORY   /DONE WITH SEARCH GET READY TO TRANSFER TO PROGR
                LIO VALUE
                HLT
                SZS 50       /IF OFF GO PUNCH SYMBOL PACKAGE
                JMP I VALUE /GO....
SYMPCH,         LAW I 200    /SYMBOL PUNCH--PUNCH BLANK LEADERR
                JDA PBT
                LIO (JMP 7750           /PUNCH NEEDED JMP INSTRUCTION
                JSP PBC2
                LAC LIST
                DAP SYM1

SYMPCB,         LAW I 20
                JDA PBT
                JSP PBC2     /PUNCH FIRST WORD OF BLOCK=0
                LAW 63.
                JDA PBC      /PUNCH SECOND WORD OF BLOCK AS LAST ENTRY IN BL0
                LAW I 21.
                DAC AD       /NUMBER OF SYMBOLS IN BLOCK
```

```
SYM1,       LAC .
            SZA I
            JMP SYMEND   /WE ARE DONE
            RAL 1S
            SPA
            JMP SMUDF    /BIT 1=1*420*SYMBOL UNDEFINED, DO NOT PUNCH
            SAR 1S
            DAC GWORD
            XCT SYM1
            AND (400000
            SZA I
            JMP SYM2     /SIGN BIT=0*420*ONLY ONE WORD IN NAME
            IDX SYM1
            XCT SYM1
SYM2,       IOR (400000
            JDA PBC      /PUNCH FIRST HALF OF SYMBOL
            LIO GWORD
            JSP PBC2     /PUNCH SECOND HALF
            IDX SYM1
            XCT SYM1     /PUNCH VALUE OF SYMBOL
            JDA PBC
            IDX SYM1
            ISP AD
            JMP SYM1
            LIO PBT
            JSP PBC2     /PUNCH CHECK SUM
            JMP SYMPCB

SYMEND,     JDA PBC
            JSP PBC1
            JSP PBC1
            ISP AD
            JMP SYM1
            LIO PBT
            JSP PBC2
            LAW I 20
            JDA PBT
            LIO (JMP 7750
            JSP PBC2
            LAW I 200
            JDA PBT
            LAC MEMORY
            LIO VALUE
            HLT
            JMP I VALUE
```

```
/SKIP OVER LIST OF USES

SMUDF,      RAR 1S
            SPA
            IDX SYM1
            IDX SYM1
            DAP .+1
            ADD .
            ADD ONE
            DAP SYM1
            JMP SYM1

/PRINT UNDEFINED SYMBOL
JF00,       XCT QLAC    /GET MASKED RIGHT HALF OF THE SYMBOL
            AND (-600000
            DAC SYMRM
            DZM SYMLM
            XCT QLAC    /MINUS *420*TWO WORDS TO NAME
            SMA
            JMP . 4
            IDX QLAC    /GET LEFT HALF OF THE WORD
            XCT QLAC
            DAC SYMLM
            IDX QLAC
            XCT QLAC    /GET THE NUMBER OF TIMES NEEDED-THE LIST COUNTER
            JDA SPT     /PRINT NAME AND NUMBER OF TIMES NEEDED
            JSP CR
            IDX SPT     /SPT CONTAINS THE LIST COUNTER
            ADD QLAC    /RESET THE POSITION FOR NEXT SYMBOL
            JMP JBACK
/SEARCH OF TABLE FOR UNDEFINED SYMBOLS
QD,         DAP QDX
QLAC,       LAC .       /STARTS AT BOTTOM OF TABLE,TO PICK UP RIGHT HAL
            SZA I       /IF ZERO WE ARE DONE, INDEX THE RETURN TO SECON
            IDX QDX     /SECOND BIT OF JMP IS 1 SO WILL FALL THROUGH
            RAL 1S      /1 BIT=0*420*DEFINED
            SPA
QDX,        JMP .       /UNDEFINED RETURN +1,DEFINED AND DONE RETURN +2
            RAR 1S
            SPA
            IDX QLAC
            IDX QLAC    /INDEX OVER LEFT HALF OF THE WORD
            IDX QLAC    /INDEX OVER THE VALUE
            JMP QLAC    /PROCESS NEXT SYMBOL
```

```
/GET WORD, CALLING SEQUENCE JDA GWORD
/ENTER WITH NUMBER IN AC, RETURN WITH NEXT VALUE FORM TAPE IN IO, THIS
/CHECKS CHECKSUM AND END OF BLOCK
GWORD,      0
            DAP GWX
            IDX BLK2
            SAD BLK4
GWS1,       JMP BLOCK          switch
BLK2,       LIO .
            LAC GWORD
            JDA AD
GWX,        JMP .

BLK1,       0           /SET TO CORE
BLK4,       LIO .
/ADD WITHOUT CHANGING -0 TO +0, ADD AC TO IO AND LEAVE IN AC
AD,         0
            DAP ADX
            LAC AD
            DIO AD
            CMA
            SUB AD
            CMA
ADX,        JMP .
/SUBROUTINE BITS, CALLING SEQUENCE JSP BITS, PROCESSES RELOACTION BITS
/RETURNS VALUE OF NEXT POSITION IN AC
BITS,       DAP BITR    /SAVE RETURN
            DZM VALUE   /ZERO SYMBOL VALUE WORD- RETURN HERE FROM SYMBO
            ISP BITSC   /INDEX RELOCATION BITS COUNTER-RETURN HERE AFTE
            JMP BIH     /NOT OUT OF BITS
            JDA GWORD   /GET ANOTHER WORD OF RELOCATION BITS
            DIO CODES
            LAW I 9
            DAC BITSC   /RESET THE BIT COUNTER
BIH,        CLA"U"CLF 3
            LIO CODES
            RCL 2S      /GET NEXT TWO RELOCATION BITS AND SAVE REST
            DIO CODES
            SZA I
            JMP XSYM    /CODE BITS 00*420*SYMBOL GO PROCESS
            ADD .+1
            DAP BIX     /SET JUMP FOR CALCULATING RELOCATION
            LAW 7777
            AND RFACTOR /GET CORE VALUE OF RFACTOR IN AC SO WE CAN USE
BIX,        JMP .
```

```
                CLC             /ABSOLUTE WORD-SET SO RELOCATION COMES OUT ZERO
                CMA             /MINUS RELOCATION
                LIO VALUE       /PLUS RELOCATION,GET VALUE TO ADD
                JDA AD
                JDA GWORD       /GO GET WORD WHICH BELONGS TO THESE CODE BITS A
BITR,           JMP .

EGO,            STF 3           /NOT IN TABLE SET INDICATOR
                LAC SYMR        /IN THE TABLE IN SOME FORM
                AND (600000     /GET CODE BITS OF WORD IN LOWER AC
                RAL 2S
                ADD .+1         /SET UP JUMP BLOCK TO CALCULATE THE VALUE OF TH
                DAP .+5
                LAC RFACTOR
                LIO SYML        /CHECK EXTEND MODE(IF EXTEND MODE SIGN BIT OF I
                SPI I
                AND (7777       /COMMA, GET RID OF CORE BITS
                JMP .
                CLC             /ABSOLUTE WORD
                CMA             /MINUS RELOCATION
                JDA GWORD       /GO GET THIRD WORD IN THE DEFINITION AND ADD IT
                DAC VALUE       /SAVE THIS VALUE IN VALUE
                SZF 3
                JMP ENTERV      /NOT ALREADY DEFINED IN TABLE-GO DEFINE
                SZF 6
                JMP COMPARE     /ALREADY DEFINED-GO SEE IF SAME
EGOL,           LAC .
                AND (-200000                /CHANGE CODE BITS SO IT HAS DEFINED
EGOD,           DAC .
                LAC LKL
                DAP VLIO
                DAP EPT         /SET UP ADDRESSES TO MOVE THE TABLE TO REMOVE R
                XCT LKL
                DAC BLK3
                CMA
                DAC GWORD
                SZS 10          /IF FLAG OFF,PRINT OUT VALUE OF THE SYMBOL
                JMP . 4
                LAC VALUE
                JDA SPT
                JSP CR
                IDX BLK3
```

```
EBLL,       IDX EPT
            DAP ELDA
            DAP ESTO
EPT,        LIO .
            LAC VALUE
            SPI         /TO PLACE SYMBOL VALUE WHERE NEEDED,ARE WE TO A
            CMA
ELDA,       LIO I .
            JDA AD
ESTO,       DAC I .
            ISP GWORD   /KEEP COUNT TO SEE WHEN DONE WITH LIST
            JMP EBLL

            LIO VALUE
            LAW I 2
            ADD LIST    /SET UP TO REMOVE LIST FROM TABLE, I MEAN REFER
            DAP EPT     /USE EPT AS END CHECK
            JMP VLIO+1

RBLOCK,     ADD RFACTOR
ABLOCK,     DAC MEMORY  /SET LOWER BOUND OF BLOCK

RWORD,      JSP BITS    /GET A DATA WORD
            DAC I MEMORY            /STORE IT IN PROGRAM
            IDX MEMORY  /INDEX MEMORY LOCATION AND CHECK F HIT LOADER T
            SUB LIST
            SPA
            JMP BITS+1  /OK-GO PROCESS NEXT WORD (RETURN ALREADY SET TO
            LAC MEMORY
            SUB TOP
            SPA
            HLT"U"CLI"U"STF 7       /HALT-OVERFLOW THE LOADER
            JMP BITS+1
/LOOKUP, ROUTINE TO LOOKUP OR STORE VALUE OF SYMBOL IN REFERENCE TABLE
LOOKUP,     DAP LKX
            JDA GWORD
            DIO SYML    /GET FIRST HALF (LEFT HALF) OF SYMBOL
            CLF 5
            LAC SYML
            AND (-600000
            DAC SYMLM   /REMOVE CODE BITS AND SAVE NAME
            SZA
            STF 5       /NO LEFT OF NAME-ONLY ONE WORD IN TABLE FOR THI
            JDA GWORD   /GET RIGHTOF SYMBOL
            DIO SYMR
            LAC SYMR
            AND (-600000            /GET THE RIGHT HALF NAME,REMOVE VOD
            DAC SYMRM
            SPI I
            RIL 1S
            LAW EGO
            SPI
            DAP LKX     /IF NOT AN EXIT CHANGE RETURN TO EGO, TO DEFINE
            LAC LIST
LKB,        DAP LKL
            DAP EGOL
            DAP EGOD
```

```
LKL,    LAC .       /PICK UP NEXT TERM OF TABLE
        SZA I
LKX,    JMP .       /EXIT, END OF TABLE
        RAL 1S      /SET FLAG 6 IF SYMBOL IN TABLE IS ALREADY DEFINED
        CLF 6
        SMA
        STF 6
        RAR 1S
        AND (-600000            /GET NAME AND COMPARE RIGHT HALF
        SAD SYMRM
        JMP LKM     /RIGHT HALF MATCHES -CHECK LEFT HALF
        XCT LKL
        SPA         /CHECK IF NAME IN TABLE HAD TWO WORDS
        IDX LKL     /YES-INCREMENT EXTRA TIME
LKR,    IDX LKL
        DAP .+2     /SET UP TO ADD PROPER NUMBER TO GET TO NEXT SYMB
        SZF I 6
        ADD .       /ITS NOT DEFINED SO ADD LENGTH OF ADDRESS LIST
        ADD ONE
        JMP LKB     /ALL SET TRY NEXT SYMBOL

LKM,    XCT LKL     /CHECK RIGHT HALF TO SEE IF THERE IS LEFT HLAF
        SPA         /SIGN BIT 1*420*TWO PARTS TO THIS NAME IN TABLE
        JMP . 4
        SZF 5       /FLAGE 5=0*420*ONE PART TO THIS NAME ON TAPE
        JMP LKR     /NOT RIGHT MATCH BECAUSE OF NAME LENGTH GO TRY
        JMP LKF     /RIGHT MATCH-GO PROCESS
        IDX LKL     /TWO PARTS GO GET SECOND PART AND COMPARE
        XCT LKL
        SAS SYMLM
        JMP LKR
LKF,    IDX LKX     /CORRECT INDEX RETURN TO SECOND POSITION
        IDX LKL     /INDEX TABLE POINTER TO COUNTER TO NAME LIST OR
        JMP LKX
```

```
/ENTER VALUE OF SYMBOL NOT IN TABLE
ENTERV,     LAW I 3      /CHANGE LIST BY 3 IF 2 WORD TO NAME, 2 IF ONE
            SZF I 5
            LAW I 2
            ADD LIST
            DAP LIST
            DAP EN1V
            LAC SYMRM
            SZF I 5      /ONLY ONE WORD TO NAME, GO PROCESS AS SUCH
            JMP EN3
            IOR (400000 /OR IN BITS TO INDICATE TWO WORD NAME
            XCT EN1V
            IDX EN1V     /STORE IN TABLE AND INCREMENT COUNTER
            LAC SYMLM
EN3,        XCT EN1V     /STORE OTHER HALF
            IDX EN1V
            SZS 10       /IF OFF PRINT NAME AND VALUE STORED
            JMP .+4
            LAC VALUE
            JDA SPT
            JSP CR
            LAC VALUE
EN1V,       DAC .        /STORE VALUE OF THE SYMBOL
            JMP BITS+1   /GO PROCESS MORE SYMBOLS (VALUE IS ZEROED)
/MULTIPLE DEFINED COMPARE
COMPARE,    XCT LKL      /GET STORED VALUE
            SAD VALUE
            JMP BITS+1   /SAME VALUE-FORGET IT AND GO ON
            LAC (FLEXO MDG          /PRINT MULTIPLY DEFINED INFORMATION
            JDA TYS
            LAC (CHAR LS+36
            JDA TYS
            LAC MEMORY
            JDA OPT
            JSP TAB
            XCT LKL
            JDA SPT
            JSP TAB
            LAC VALUE
            JDA OPT
            JSP CR
            HLT"U"CLI"U"CLI          /WAIT AND SEE WHATTO DO
            JMP BITS+1   /IF GO ON-USE FIRST VALUE DEFINED
```

```
VBACK,      DAP VLIO    /DECREASE ALL REGISTERS BY ONE IN MOVING THE TAB
            ADD BLK3
            DAP .+1
            DIO .
VLIO,       LIO .       /ORIGINALLY SET TO LIST COUNTER IN TABLE MOVES
            LAW I 1
            ADD VLIO
            SAS EPT     /CHECK FOREND OF MOVE
            JMP VBACK
            LAC VLIO-1  /CHANGE VALUE OF LIST TO NEW VALUE-WE ARE DONE
            DAP LIST
            JMP BITS+1
/XSYM CALCULATES VALUE OF SYMBOL IF AVAILABLE
XSYM,       JSP LOOKUP  /GO LOOKUP THE SYMBOL
            JMP ENTERS  /AND EXIT NOT FOUND IN TABLE-GO PLACE NAME IN TB
            SZF I 6     /FOUND IN TABLE IS IT DEFINED
            JMP INSERT  /NO-GO INSET THIS ADDRESS IN  ITS ADDRESS LIST
            XCT LKL     /DEFINED GET VALUE
            LIO SYML    /CHECK IF TO ADD OR SUBTRACT, ADD IF SIGN BIT 0
            SPI
            CMA
            LIO VALUE   /ADD THIS SYMBOL TO THOSE ALREADY PROCESSED
            JDA AD
            DAC VALUE
            JMP BITS+2  /GO BACK FOR MORE
/ENTER SYMBOL IN TABLE-UNDEFINED
ENTERS,     SZF 4       /IF JUMP BLOCK IGNORE
            JMP BITS+2
            LAW I 4
            SZF I 5
            LAW I 3     /IF TWO WORDS TO NAME RESET FOR 4 LOCATIONS IN
            ADD LIST
            DAP LIST
            DAP EN1
            LAC SYMRM   /GET RIGHT HALF AND SET UP BITS TO SHOW UNDEFINE
            IOR (200000
            SZF I 5     /IF ONLY ONE WORD NAME SAVE AND GO ON,OTHERWISE
            JMP EN2
            IOR (400000
            XCT EN1
            IDX EN1     /SAVE NAME(RIGHT HALF) AND INDEX TABLE POINTER
            LAC SYMLM   /GET LEFT HALF TO PLACE
EN2,        XCT EN1
            IDX EN1
            LAW 1       /SET COUNTER OF UNDEFINED ADDRESS LIST TO 1
            XCT EN1
            IDX EN1
IN1,        LAC SYML    /SET UPPER BITS OF ADDRESS LIST TO TELL IF ADD
                        /OF LEFT HALF ARE 0
            AND (400000
            IOR MEMORY
EN1,        DAC .
            JMP BITS+2  /GO BACK AND PROCESS MORE
```

```
/INSERT MEMORY LOCATION TO LISTOF UNDEFINED SYMBOLS
INSERT,     SZF 4          /IF THIS IS A JUMP BLOCK FORGET IT
            JMP BITS+21
            DAP EN1        /HAVE ENTERED WITH LOCATION OF COUNTER WORD IN A
            DAP .+1
            IDX .          /UP THE COUNTER BY ONE
            LAW I 1
            ADD LIST       /ADJUST LIST DOWN BY ONE AND SET ADDRESS TO MOVE
            DAP LIST
            DAP MDAC
            DAP MLAC
MLOOP,      IDX MLAC
MLAC,       LAC .
MDAC,       DAC .
            IDX MDAC
            SAS EN1        /CHECK IF AT END OF TABLE
            JMP MLOOP
            JMP IN1
/SOME TYPE OUT ROUTINES
CR,         DAP CRTABX
            LAW 77
            JMP .+3
TAB,        DAP CRTABX
            LAW 36
            JDA TOU
CRTABX,     JMP .

TYS,        0
            DAP TYX
            LAW I 3
            DAC OPT

TYL,        LAC TYS
            RAL 6S
            DAC TYS
            AND (77
            SZA
            JDA TOU
            ISP OPT
            JMP TYL
TYX,        JMP .

TOU,        0
            DAP TOX
            CKS
            RIL 2S
            SPI I
            JMP .-3
            LIO TOU
            TYO-I
TOX,        JMP .
```

```
/PUNCH BINARY CODE-JDA WITH VALUE IN AC
PBC,        0
PBC1,       LIO PBC
PBC2,       DAP PCX
            REPEAT 3,PPB            RCL 6S
            ADD PBT     /CHECK CHECKSUM
            DAC PBT
PCX,        JMP .
/PUNCH BLANK TAPE-JDA WITH MINUS NO OF LINES IN AC
PBT,        0
            DAP PBX
            CLI
            PPA
            ISP PBT
            JMP .-2
PBX,        JMP .

LBLOCK,     LAC MEMORY
            DAC RFACTOR
            LAW LDONE
            DAP GWS1    /RESET RETURN FROM BLOCK READING GWORD TO LDONE
ANLIB,      JSP LOOKUP  /GO GET SYMBOL FROM TAPE-IS IT DEFINED
             JMP ANLIB  /NOT IN TABLE -TRY NEXT ONE
            SZF 6       /IN TABLE-IS IT DEFINED
            JMP ANLIB   /YES-TRY NEXT SYMBOL
            LAW BLOCK   /WE NEED NEXT ROUTINE -RESET BLOCK RETURN SO TH
            DAP GWS1
IGNORE,     JDA GWORD   /READ TO END OF BLOCK
            JMP .-1

LDONE,      JSP BLOCK+3 /END OF LIBRARY BOLCK-IGNORE NEXT ROUTINE,THIS
             JMP IGNORE
             JMP IGNORE
             JMP ANLIB  /LIBRARY BLOCK TRY AGANIN
            LAW LJP     /JMP BLOCK END OF LIBRARY TAPE
            JMP IGNORE-1            /SET RETURN SO THAT WILL RETURN TO

OPT,        0
            STF 1
            DAP OPX
            LAW I 6
            DAC AD

OP1,        LIO OPT
            CLA
            RCL 3S
            DIO OPT
            SZA
            CLF 1
            SZA I
            LAW CHAR RO
            SZF I 1
            JDA TOU
            ISP AD
            JMP OP1
            LAW CHAR RO
            SZF 1
            JDA TOU
OPX,        JMP .
```

```
SPT,            0
                DAP SPX
                LAW SYMLM
                DAP SPDAC
SPB,            DAP SPLAC
                LAW SPD
                DAP SPJ

SPN,            DZM AD
SPR,            IDX AD
SPLAC,          LAC .
SPJ,            SUB .
                SPA
                JMP SPP
SPDAC,          DAC .
                JMP SPR

SPP,            LAC AD
                SCR 1S
                SZA I
                JMP SPS
                ADD SPT+1
                DAP .+1
                LAC .
                SPI I
                RAR 6S
                JDA TOU
SPS,            IDX SPJ
                SAS (SUB SPD 3
                JMP SPN
                IDX SPDAC
                SAS (DAC SYMLM 2
                JMP SPB
                JSP TAB
                LAC SPT
                JDA OPT
SPX,            JMP .

SPL,            FLEXO  01    FLEXO  23
                FLEXO  45    FLEXO  67
                FLEXO  89    FLEXO  AB
                FLEXO  CD    FLEXO  EF
                FLEXO  GH    FLEXO  IJ
                FLEXO  KL    FLEXO  MN
                FLEXO  OP    FLEXO  QR
                FLEXO  ST    FLEXO  UV
                FLEXO  WX    FLEXO  YZ
                CHAR M.
SPD,            50"T"50      50          ONE,        1
SYMLM,          0            SYMRM,      0
BOT,            SA-1         TOP,        B
CONSTANTS
```

```
A,
BLJ,        0           FIRST,      0
LIST,       0           CHECKSUM,   0
BLK3,       0           VALUE,      0
BITSC,      0           CODES,      0
RFACTOR,    0           SYML,       0
SYMR,       0           MEMORY,     0
LAST,       LAST+104/
B,
START ENTRY
```