

```
40154 04178 1.   3 USERDIC  SI  MINTSYS
              3.   1 USERDIC      SI  MINTCNF
              4.   1 USERDIC      .
              4.   2 USERDIC      .      Create MINT compiler system
              4.   3 USERDIC      .
              4.   4 USERDIC      .      Standard MINT elements
              4.   5 USERDIC      .
              4.   6 USERDIC      MACRO IOSBS: ' '
04180        4.   7 USERDIC      MACRO SUBS:  'SI MINTSUBS '
04184        4.   8 USERDIC      MACRO COMP:  'SI MINTCOMP '
04188        4.   9 USERDIC      MACRO ASUBS: 'SI MINTAUTO '
04192        4.  10 USERDIC      MACRO PFOUT: 'SI MINTPFOT '
04196        4.  11 USERDIC      .
              4.  12 USERDIC      .      System dependent elements
              4.  13 USERDIC      .
              4.  14 USERDIC      MACRO PRIM:  'SI MINTOPRM '
04200        4.  15 USERDIC      MACRO ODIR:  'SI MINTODIR ' .
04204        4.  16 USERDIC      .
              4.  17 USERDIC
              3.   2 USERDIC      PAGE
```

\$

T

T

```

3. 3 USERDIC      UNLOCK INTDIC
3. 4 USERDIC      ASUBS
4. 1 USERDIC      .
4. 2 USERDIC      .   MINT auto-compiler procedures
4. 3 USERDIC      .   -----
4. 4 USERDIC      .
4. 5 USERDIC      .   Introduce directives
4. 6 USERDIC      .
4. 7 USERDIC      DIR PO                      . PF output file def
4. 8 USERDIC      DIR AUTO                    . Set PF generate mode
4. 9 USERDIC      DIR GENPF                   . Create PF of code and data
4. 10 USERDIC     DIR GENSYS                   . Create PF system (incl dictionaries)
04277 4. 11 USERDIC     DICT MAUTO:HDICT        . public dict for auto
04350 4. 12 USERDIC     DICT IAUTO:HDICT        . internal dict for auto
4. 13 USERDIC     .
4. 14 USERDIC     .   Introduce internal functions
4. 15 USERDIC     .
4. 16 USERDIC     BLOCK
4. 17 1           FN DICBUILD                  . dictionary build facility
4. 18 1           FN RELOCD                    . compute relocated address
4. 19 1           FN ADDROF                    . find current address of id
4. 20 1           FN COPYREC                   . copy dic rec into data area
4. 21 1           FN NEWCLASS                  . set up new class rec addr
4. 22 1           FN RELADDR                   . relocate addr field
4. 23 1           FN SCOPY                     . dictionary string copy
4. 24 1           .
4. 25 1           .   Data
4. 26 1           .
4. 27 1           VAR CADIC:0                  . current active dict
4. 28 1           VAR SAVDIC:0
04351 4. 29 1           VAR NEWDIC:HDICT
04425 4. 30 1           VAR SCAP:0 VAR SCOL:0
04426 4. 31 1           VAR TCAP:0 VAR TCOL:0
04428 4. 32 1           .
4. 33 1           .   Dictionary entry MAP
4. 34 1           .
4. 35 1           ICON NDCURPOS  DCURPOS
04429 4. 36 1           ICON NDLEN  DLEN
4. 37 1           FN @NDSHUNT  EQV @@DSHUNT
4. 38 1           FN @NDSTATUS EQV @@DSTATUS
4. 39 1           FN @NDCLASS  EQV @@DCLASS
4. 40 1           FN @NDADDR   EQV @@DADDR
4. 41 1           FN @NDLINK   EQV @@DLINK
4. 42 1           FN @NDNAME   EQV @@DNAME
4. 43 1           FN NDLENGTH  EQV @DLENGTH
4. 44 1           MACRO NDCLASS: ' VAL(@NDCLASS) '
04434 4. 45 1           PAGE

```

```

4. 46 1 . SCOPY(<addr of source CAP>, <addr of target CAP>)
4. 47 1 .
4. 48 1 VAR CAPADI:0 VAR CAPADO:0
04435 4. 49 1 SCOPY:ENTRY
40155 04436 4. 50 1 ->@CAPADO, ->@CAPADI, VAL(VAL(CAPADI)),
40165 4. 51 1 DUP ->VAL(CAPADO) WHILE DUP(-1) GE 0 START
40180 4. 52 1 PUTCH(CAPADO, GETCH(CAPADI))
40185 4. 53 1 ADV(1 FROM CAPADI), ADV(1 FROM CAPADO),
40198 4. 54 1 REPEAT LOSE
40201 4. 55 1 EXIT
40203 4. 56 1 .
4. 57 1 . Address accessing function
4. 58 1 .
4. 59 1 ADDROF:ENTRY
40204 4. 60 1 SWINPUT( ,@STEOS), NEXTELT, DADDR, DO EOSTRCT
40210 4. 61 1 EXIT
40214 4. 62 1 .
4. 63 1 . relocate address field
4. 64 1 .
4. 65 1 RELADDR:ENTRY
40215 4. 66 1 DUP(DSTATUS) MASK 256 NE 0 THEN<
40227 4. 67 1 OUTST('Unset identifier: ') OUTST(@DNAME) OPNL>
40233 04442 4. 68 1 MASK 512 EQ 0 THEN <RELOCD(DADDR)->@DADDR>
40247 4. 69 1 EXIT
40248 4. 70 1 .
4. 71 1 . Calculate relocated address value
4. 72 1 .
4. 73 1 RELOCD:ENTRY DUP -->15 CHOOSE(PBIAS, DBASE), ADIFF EXIT
40260 4. 74 1 .
4. 75 1 . new class record search
4. 76 1 .
4. 77 1 NEWCLASS:ENTRY
40261 4. 78 1 CUR, IDOFADDR(NDCLASS, BASESTRT, [SLKP, EXIT]),
4. 79 1 CUR, IDOFADDR(NDCLASS, BASESTRT, [ ->@CUR, LOSE, SLKP, EXIT]),
40281 4. 80 1 ADDROF(@DNAME),
40283 4. 81 1 CUR EQ DICNMR THEN<LOSE, ->@CUR,
40295 4. 82 1 OUTST(@DNAME), OUTST(' undefined.'), OPNL, EXIT>
40302 04446 4. 83 1 <=>, ->@CUR, . restore new dic rec
40306 4. 84 1 RELOCD()->@NDCLASS, . set new class pointer
40309 4. 85 1 EXIT
40310 4. 86 1 PAGE

```

```

4. 87 1 . copy dictionary record
4. 88 1 .
4. 89 1 COPYREC:ENTRY
40311 4. 90 1 DSHUNT, DSTATUS, DCLASS, DADDR, DLINK, @DNAME->@SCAP,
40325 4. 91 1 NDCURPOS FROM DLOC ->@CUR, . new rec address
40333 4. 92 1 ->@NDLINK, ->@NDADDR, ->@NDCLASS, ->@NDSTATUS, ->@NDSHUNT,
40343 4. 93 1 DUP(0)->@SCOL,->@TCOL, @NDNAME->@TCAP,
40356 4. 94 1 SCOPY(@SCAP, @TCAP),
40361 4. 95 1 NDLENGTH FROM DLOC ->@DLOC,
40368 4. 96 1 DLOC GE MAXDLOC THEN <OUTST('Insufficient D-space.'), OPNL, ESTOP>
40381 04453 4. 97 1 ADDTODIC,
40382 4. 98 1 NEWCLASS, . link and relocate class pntr
40383 4. 99 1 EXIT
40384 4. 100 1 .
4. 101 1 . dictionary build function
4. 102 1 .
4. 103 1 DICBUILD:ENTRY
40385 4. 104 1 DIC->@SAVDIC, DICDLST->@DICP
40392 4. 105 1 WHILE DICP LT DICND START DICNMR->DICP,
40407 4. 106 1 1 FROM DICP->@DICP
40412 4. 107 1 REPEAT
40418 4. 108 1 PUSHODIC(DIC), ENVLIST->@ENVSTRT,
40426 4. 109 1 DUP->@CADIC, ->@DIC, DICDLST->@DICP,
40438 4. 110 1 WHILE CADIC->@DIC, DICP LT DICND START
40450 4. 111 1 VAL(DUP(DICP))->@CUR, DICNMR,
40459 4. 112 1 WHILE DUP NE CUR START
40466 4. 113 1 CUR, COPYREC, VAL()->@CUR,
40473 4. 114 1 REPEAT LOSE,
40477 4. 115 1 ->@DICP, 1 FROM DICP->@DICP
40485 4. 116 1 REPEAT
40491 4. 117 1 SAVDIC->@DIC, DICDLST->@DICP
40498 4. 118 1 WHILE DICP LT DICND START VAL(DICP)->@CUR, DICNMR
40515 4. 119 1 WHILE DUP NE CUR START
40523 4. 120 1 RELOCD(DUP(VAL(CUR)))->CUR, . relocate dlink
40531 4. 121 1 RELADDR . reloc addr field
4. 122 1 ->@CUR, . next dic rec
40535 4. 123 1 REPEAT LOSE,
40539 4. 124 1 RELOCD(VAL(DICP))->DICP, 1 FROM DICP->@DICP
40551 4. 125 1 REPEAT POPUPDIC LOSE
40558 4. 126 1 EXIT
40560 4. 127 1 PAGE

```

```

4. 128 1          PFOUT
5. 1 1          .
5. 2 1          . Portable format dump functions
5. 3 1          . -----
5. 4 1          .
5. 5 1          FN OUTPUT          . Output words with addr range
5. 6 1          FN OPOUTPUT        . Initialization for PF output
5. 7 1          FN CLOUTPUT        . Close PF output file
5. 8 1          BLOCK
5. 9 2          FN OUTCHC          . Output character unconditionally
5. 10 2         FN OUTNUM          . Output number
5. 11 2         FN OUTADD          . Output address value
5. 12 2         FN ROUTPUT        . Output words with rel addr range
5. 13 2         FN PFOUT          . Output words with abs or rel addresses.
5. 14 2         FN NEWLINE        . Output current line.
5. 15 2         VAR RELOCF:0       . Current relocation factor
5. 16 2         VAR SEQNR:0        . Sequence number
04454 5. 17 2         VAR POUNIT:0 . unit for portable output
04455 5. 18 2         VAR SOSAVE:0  . Save of SOUNIT
04456 5. 19 2         VAR CHKSUM:0  . Image checksum (DIFFER of numerics)
04457 5. 20 2         VAR CNT: 0, VAR ADD:0,
04460 5. 21 2         VAR AFLG:2, VAR END:0, VAR PRADD:0, VAR ARMK:0,
04464 5. 22 2         ICON cr 13
5. 23 2         ICON dc3 19
5. 24 2         .
5. 25 2         . Open file for portable format output.
5. 26 2         .
5. 27 2         REF PO:ENTRY OPENF(2,FNAME) ->@POUNIT, 1 ->@SEQNR EXIT
40574 5. 28 2         .
5. 29 2         . Initialize portable format output.
5. 30 2         .
5. 31 2         OPOUTPUT:ENTRY
40575 5. 32 2         SOUNIT ->@SOSAVE, POUNIT->@SOUNIT, SETBSE(#0,0,16)
40591 5. 33 2         EXIT
40593 5. 34 2         .
5. 35 2         . Close output file.
5. 36 2         .
5. 37 2         CLOUTPUT:ENTRY
40594 5. 38 2         NEWLINE
5. 39 2         OUTCHC(#0) OUTCHC(/) OUTCHC(#0) . Output load terminator
40603 5. 40 2         NEWLINE OUTCHC(dc3) OUTCHC(cr) . Close current line, output 'eof'
40610 5. 41 2         SETBSE(#0,5,10), SOSAVE ->@SOUNIT,
40623 5. 42 2         CLOSEF(POUNIT)
40625 5. 43 2         EXIT
40627 5. 44 2         PAGE

```

```

5. 45 2 . Output a new line
5. 46 2 .
5. 47 2 NEWLINE:ENTRY
40628 5. 48 2 OUTNUM(CHKSUM, #.), OUTNUM(SEQNR, # ), OUTCHC(cr),
40641 5. 49 2 ADV(@SEQNR), DUP(0 )->@CHKSUM, ->@CNT,
40653 5. 50 2 EXIT
40654 5. 51 2 .
5. 52 2 . Check if new line is to be started (max of 80 char's per line)
5. 53 2 . therefore must stop at 57 to allow full field plus checksum.
5. 54 2 .
5. 55 2 FN CNEWLINE:ENTRY CNT GE 57 THEN <NEWLINE, OUTADD(1 FROM ADD)> EXIT
40671 5. 56 2 .
5. 57 2 . Output character to POUNIT and update count
5. 58 2 . OUTCHC (<character>)
5. 59 2 .
5. 60 2 OUTCHC:ENTRY OPCH(<=>POUNIT) ADV(@CNT) EXIT
40680 5. 61 2 .
5. 62 2 . Output number (to POUNIT)
5. 63 2 . OUTNUM(<number>,<terminator>)
5. 64 2 .
5. 65 2 . This 32 bit version writes unsigned HEX because in full
5. 66 2 . 32 bit implementation in a 32 bit machine signs cannot
5. 67 2 . be distinguished.
5. 68 2 .
5. 69 2 OUTNUM:ENTRY
40681 5. 70 2 <=>, DIFFER(DUP, CHKSUM)->@CHKSUM,
40689 5. 71 2 DUP NE 0 THEN< OPINTD(), CWIDTH+CNT->@CNT ELSE LOSE>
40709 5. 72 2 OUTCHC() . Output terminator
5. 73 2 EXIT
40711 5. 74 2 .
5. 75 2 . Output address value
5. 76 2 . OUTADD(<address value>)
5. 77 2 .
5. 78 2 OUTADD:ENTRY
40712 5. 79 2 DUP EQ PRADD THEN<LOSE, EXIT> . If address already output, forget it
40721 5. 80 2 CNT EQ 0 THEN<DUP->@PRADD> . Save address on start of line
40733 5. 81 2 OUTNUM( ,ARMK)
40735 5. 82 2 EXIT
40737 5. 83 2 PAGE

```

```

5. 84 2 . Create portable format image
5. 85 2 . PFOUT(<from address>,<to address>,<reloc. factor>,<#/ or #;>)
5. 86 2 .
5. 87 2 . Format: Absolute address followed by "/",
5. 88 2 . Relative address followed by ";"
5. 89 2 . Data followed by "," (Neg. foll. by "-")
5. 90 2 . Checksum followed by "." and line sequence number
5. 91 2 . Switch from absolute to relative loading ":"
5. 92 2 .
5. 93 2 PFOUT:ENTRY
40738 5. 94 2 ->@ARMK, ->@RELOCf, ->@END, 2->@AFLG,
40752 5. 95 2 WHILE DUP NE END START
40759 5. 96 2 DUP-RELOCf->@ADD, . Set current address
40766 5. 97 2 DUP(VAL(DUP)) EQ 0 THEN< . Check if zero value
40775 5. 98 2 LOSE, ADV(@AFLG), . Don't output zero
40779 5. 99 2 . If more than 2 successive
5. 100 2 . zero's output new address value
5. 101 2 ELSE DUP(AFLG) GT 1 THEN <LOSE, OUTADD(ADD), 0>
40797 5. 102 2 EQ 1 THEN <OUTNUM(0, #,)> . Output terminator only
40808 5. 103 2 OUTNUM( #, ), CNEWLINE, 0->@AFLG>. Output the value
40817 5. 104 2 1, <=>, FROM,
40821 5. 105 2 REPEAT LOSE
40824 5. 106 2 EXIT
40826 5. 107 2 .
5. 108 2 . Output for absolute load
5. 109 2 . OUTPUT(<from address>,<to address>,<reloc. factor>)
5. 110 2 .
5. 111 2 OUTPUT:ENTRY PFOUT( #/ ) EXIT
40831 5. 112 2 .
5. 113 2 . Output for relative load
5. 114 2 . ROUTPUT(<from address>,<to address>,<reloc. factor>)
5. 115 2 .
5. 116 2 ROUTPUT:ENTRY OUTNUM(0, #:) PFOUT( #; ) EXIT
40841 5. 117 2 ENDBLOCK
5. 118 1 PAGE

```

```

4. 129 1 . Relocate Reference List and output to PLOC and DLOC.
4. 130 1 .
4. 131 1 FN RELOCPF:ENTRY
40842 4. 132 1 0->DBASE,
40847 4. 133 1 RELOCD(DLOC)->(@IDLOC$ FROM DBASE), . Set initial DLOC.
40856 4. 134 1 WHILE RLIST NE 0 START . relocation addr list
40864 4. 135 1 DUMP(POPPEDUP(@RLIST)), . loc to be relocated
40868 4. 136 1 RELOCD(VAL(TEMP)) ->TEMP, . set relocated contents
40875 4. 137 1 REPEAT,
40878 4. 138 1 OOUTPUT,
40879 4. 139 1 OUTPUT(DBASE, DLOC, DBASE),
40886 4. 140 1 OUTPUT(PBASE, PLOC, PBIAS),
40893 4. 141 1 CLOUTPUT,
40894 4. 142 1 OUTST('New system generated. Space used (PROG/DATA): '),
40897 04477 4. 143 1 OPINT(PLOC-PBASE), OPINT(DLOC-DBASE), OPNL,
40910 4. 144 1 PBASE->@PLOC, DBASE->@DLOC, DUP(0)->@PBASE->@DBASE
40926 4. 145 1 EXIT
40930 4. 146 1 .
4. 147 1 . auto-compile directive
4. 148 1 .
4. 149 1 REF AUTO:ENTRY
40931 4. 150 1 COMPILE('LAB DT EQV @DATE,') . CLEARLST(@STRLIST),
40933 04483 4. 151 1 PLOC->@PBASE, DLOC->@DBASE,
40944 4. 152 1 MASK(32767, PLOC)->@PBIAS, ENVLIST ->@CURRSTRT,
40957 4. 153 1 REF MAUTO,
40960 4. 154 1 COMPILE('DATA, RESERVE 1, VAR MAXDS$:0, VAR MAXPS$:0, '),
40963 04496 4. 155 1 SIUNIT EQ 0 THEN<OUTST('Specify VSTORE data size: ')>
40974 04504 4. 156 1 READ, ININT-->7, DUP, ->(MINUS 2 FROM DLOC), OPNL,
40987 4. 157 1 OUTST(' VSTORE data size set to '), *128, OPINTD, OPNL,
40995 04513 4. 158 1 SIUNIT EQ 0 THEN<OUTST('Specify VSTORE program size: ')>
41006 04522 4. 159 1 READ, ININT-->7, DUP, ->(MINUS 1 FROM DLOC), OPNL
41018 4. 160 1 OUTST(' VSTORE program size set to '), *128, OPINTD, OPNL,
41027 04531 4. 161 1 COMPILE('DATA, VAR IDLOC$:0,
4. 162 1 -VAR CONT$ : LAB ERROR ERROR,
4. 163 1 -LAB SYSDAT$:6, &(VAL(1 FROM DT)), &(VAL(2 FROM DT)), 0,
4. 164 1 -VAR EXOPT$:0, 0, VAR DREM:0,
4. 165 1 -LAB DATE:6, &(VAL(1 FROM DT)), &(VAL(2 FROM DT)), 0,
4. 166 1 -RESERVE 4, LAB CARRYB:0, ')
41029 04584 4. 167 1 SIUNIT EQ 0 THEN<OUTST('Specify system ID: ')> READ, OPNL,
41043 04590 4. 168 1 OUTST(' Old ID was:'), OUTST(SYSID$), OPNL
41049 04595 4. 169 1 OUTST(' New ID is :'), OUTST(CURCHS), OPNL
41056 04600 4. 170 1 COMPILE('DATA, LAB SYSID$:FORGET DT,')
41059 04608 4. 171 1 LOSE(FNAME), 81 FROM DBASE->@DLOC,
41070 4. 172 1 EXIT
41071 4. 173 1 PAGE

```



```

4. 174 1 .
4. 175 1 . Required if string format to be changed
4. 176 1 .
4. 177 1 FN STRCONV:ENTRY WHILE STRLIST NE 0 START
41080 4. 178 1 DUP(POPPEDUP(@STRLIST))->@SCAP,->@TCAP,
41090 4. 179 1 DUP(0)->@SCOL,->@TCOL, SCOPY(@SCAP, @TCAP)
41103 4. 180 1 REPEAT
41107 4. 181 1 EXIT
41108 4. 182 1 .
4. 183 1 . Create Portable Format of new code and data.
4. 184 1 .
4. 185 1 REF GENPF:ENTRY POPDIC POPDIC RELOCPF, REF USERDIC, EXIT
41116 4. 186 1 .
4. 187 1 . Create complete new system.
4. 188 1 .
4. 189 1 VAR CITEM:0
4. 190 1 REF GENSYS:ENTRY
41117 04609 4. 191 1 ENVLIST->@ENVSTRT, . STRCONV,
41122 4. 192 1 VAL(ADDROF('BASESTRT'))->@CITEM,
41129 04612 4. 193 1 VAL(1 FROM CITEM) ->@DIC, DICDLST, DICNMR,
41142 4. 194 1 @NEWDIC->@DIC, ->@DICNMR, ->@DICDLST, DICBUILD(@IAUTO),
41154 4. 195 1 VAL(CITEM)->@CITEM, VAL(1 FROM CITEM)->@DIC, DICDLST,
41171 4. 196 1 @NEWDIC->@DIC, ->@DICDLST, DICBUILD(@MAUTO),
41181 4. 197 1 POPDIC POPDIC, ENVLIST->@ENVSTRT, RELOCPF
41188 4. 198 1 EXIT
41190 4. 199 1 ENDBLOCK
4. 200 USERDIC .
3. 5 USERDIC PO MINTPFA.NEW
3. 6 USERDIC AUTO

```

VSTORE data size set to 32768

VSTORE program size set to 32768

Old ID was:3.0

New ID is :3.0

```

32768 00081 3. 7 MAUTO PROG
3. 8 MAUTO FN SYSSTART , SYSSTART,
32769 3. 9 MAUTO IOSBS
3. 10 MAUTO PAGE

```

```

3. 11 MAUTO      SUBS
5.  1 MAUTO      .
5.  2 MAUTO      .   Introduction of Identifiers
5.  3 MAUTO      .   -----
5.  4 MAUTO      .
5.  5 MAUTO      VAR TEMP:0      . general temp storage
5.  6 MAUTO      VAR MAXDLOC     . end of data space
5.  7 MAUTO      VAR DLOC        . data-space pointer
00083 5.  8 MAUTO      VAR MAXILOC:&(448*1024) . top of Item space
00084 5.  9 MAUTO      VAR ILOC:  &(384*1024) . bottom of Item space
00085 5. 10 MAUTO      VAR MAXRLOC:&(511*1024) . top of POOL
00086 5. 11 MAUTO      VAR RLOC:  &(448*1024) . bottom of POOL
5. 12 MAUTO      FN GETREC       . acquire a record from pool
5. 13 MAUTO      FN RELREC       . release a record back to pool
5. 14 MAUTO      DIR POOL        . output current pool info
5. 15 MAUTO      FN MOVE         . move data
5. 16 MAUTO      FN DUMP         . dump object into TEMP
5. 17 MAUTO      FN DETACHED     . detach item from a list
5. 18 MAUTO      FN NEXTFREE     . obtain free space entity
5. 19 MAUTO      FN JOIN         . join item to a list
5. 20 MAUTO      FN PUSHD        . push item into free-space
5. 21 MAUTO      FN POPUP        . pop up item from free space
5. 22 MAUTO      FN POPPEDUP     . as POPUP but obtain item
5. 23 MAUTO      FN PUSHDREC     . push a record
5. 24 MAUTO      FN POPUPREC     . pop last record pushed
5. 25 MAUTO      FN FOREACH      . operate on all members of list
5. 26 MAUTO      FN CLEARLST     . POPUP all items in a list.
5. 27 MAUTO      FN NULL        . do nothing
5. 28 MAUTO      FN GETCH        . get a character
5. 29 MAUTO      FN PUTCH        . store a character
5. 30 MAUTO      FN FORCHS       . operate on each char in string
5. 31 MAUTO      .FN MATCH       . match two strings
5. 32 MAUTO      .FN DICMATCH    . match key string with list of strings
5. 33 MAUTO      FN GETSTR       . read string of characters
5. 34 MAUTO      FN READ         . read input image
5. 35 MAUTO      FN INSTRING     . build input string
5. 36 MAUTO      FN ADVCH        . advance character pointer
5. 37 MAUTO      FN CHAR         . obtain current character
5. 38 MAUTO      FN NEXTCH      . obtain next character
5. 39 MAUTO      FN FNAME        . read string until blank or CR
5. 40 MAUTO      FN ININT        . input decimal number
5. 41 MAUTO      FN INHEX       . input HEX number
5. 42 MAUTO      FN LETTER       . test for alpha-character
5. 43 MAUTO      FN DIGIT        . test for decimal number
5. 44 MAUTO      FN BLANKS       . space over blanks
5. 45 MAUTO      PAGE

```

```

5. 46 MAUTO . SIUNIT and LINENO form a 2 word record for use
5. 47 MAUTO . in source listings. INPST is the list pointer.
5. 48 MAUTO VAR SIUNIT:0 . input unit number
5. 49 MAUTO IAUTO
5. 50 IAUTO VAR LINENO:0 . input line number
00087 5. 51 IAUTO VAR INPST:0,2 . siunit and line number record
00089 5. 52 IAUTO VAR BUFFER:0,35 . 132 char input image buffer
00091 5. 53 IAUTO VAR BTCOUNT:0 . total active nodes in B-tree
00092 5. 54 MAUTO MAUTO
5. 55 MAUTO VAR NOCHS:0 . char count in INSTRING
00093 5. 56 MAUTO . The variables CURCHS, CURCOL, and EOSTRACK are
5. 57 MAUTO . managed as a 3 word record. Thus, they must be set
5. 58 MAUTO . consecutively as shown below.
5. 59 MAUTO .
5. 60 MAUTO VAR CURCHS:0 . input buffer cap
00094 5. 61 MAUTO VAR CURCOL:0 . current char pos.
00095 5. 62 MAUTO IAUTO
5. 63 IAUTO VAR EOSTRACK:READ, . end of string action
00097 5. 64 IAUTO MAUTO
5. 65 MAUTO FN OUTST . output a string
5. 66 MAUTO FN OPNL . output carriage-return
5. 67 MAUTO FN OPFF . output form feed
5. 68 MAUTO FN OPINT . output integer number
5. 69 MAUTO FN OB . output a blank character
5. 70 MAUTO FN OBW . output WIDTH+1 blanks
5. 71 MAUTO FN SETOPP . set opint parameters
5. 72 MAUTO FN OPINTD . output magnitude of int number
5. 73 MAUTO FN SETBSE . set base and opint parameters
5. 74 MAUTO IAUTO
5. 75 IAUTO VAR BASE:10 . opint conversion base
5. 76 IAUTO MAUTO
5. 77 MAUTO VAR BLANK6:' ' . 6 blanks for space fill
00101 5. 78 MAUTO VAR SOUNIT:0 . current output unit
5. 79 MAUTO VAR PAD:#0, . output pad character
00103 5. 80 MAUTO VAR WIDTH:5, . output width not counting
00104 5. 81 MAUTO . leading blank or -
5. 82 MAUTO VAR CWIDTH:0 . width of last int output
5. 83 MAUTO FN BTINIT . Initialize B-Tree.
5. 84 MAUTO FN BTDEL . Delete entire tree.
5. 85 MAUTO FN BTINSRT . Insert data item in tree.
5. 86 MAUTO FN BTREM . Remove data (all items for given key).
5. 87 MAUTO FN FORBTVAL . Apply a function to each data item for a
5. 88 MAUTO . given key.
5. 89 MAUTO FN BINLOC . Binary search routine.
5. 90 MAUTO BLOCK
5. 91 1 VAR ITEM:0,2 . item list for free space
00106 5. 92 1 PAGE

```

```

5. 93 1 . Utility routines
5. 94 1 . -----
5. 95 1 .
5. 96 1 . DUMP(<item>)
5. 97 1 .
32774 00107 5. 98 1 DUMP:ENTRY ->@TEMP EXIT
5. 99 1 .
5. 100 1 . Do nothing function
5. 101 1 .
5. 102 1 NULL:ENTRY EXIT
32776 5. 103 1 .
5. 104 1 . MOVE(<item count>, <source addr.>, <dest. addr.>)
5. 105 1 .
5. 106 1 VAR SOURCE:0 VAR DEST:0 VAR STEP:0 VAR LIM:0 VAR CNT:0
00111 5. 107 1 MOVE:ENTRY
32777 00112 5. 108 1 ->@DEST, ->@SOURCE, DUP LE 0 THEN< LOSE EXIT >
32792 5. 109 1 SOURCE GT DEST THEN< ()-1, MINUS 1 <=>, 1 ELSE 0, MINUS 1 >
32815 5. 110 1 ->@STEP, ->@LIM, ->@CNT,
32824 5. 111 1 WHILE CNT NE LIM START CNT+STEP->@CNT,
32840 5. 112 1 VAL(CNT FROM SOURCE) ->(CNT FROM DEST),
32852 5. 113 1 REPEAT
32855 5. 114 1 EXIT
32856 5. 115 1 .
5. 116 1 . Item List routines
5. 117 1 . -----
5. 118 1 .
5. 119 1 . Detach item from list
5. 120 1 . DETACHED(<addr of list pointer>)
5. 121 1 .
5. 122 1 DETACHED:ENTRY
32857 5. 123 1 DUP(VAL(DUP)) EQ 0 THEN< <=>, LOSE EXIT>,
32869 5. 124 1 <=>, VAL(VAL(DUP))<=>, ->,
32875 5. 125 1 EXIT
32876 5. 126 1 .
5. 127 1 . Obtain free space entity
5. 128 1 . NEXTFREE(<addr of free space pointer>)
5. 129 1 .
5. 130 1 NEXTFREE:ENTRY
32877 5. 131 1 VAL(DUP) EQ 0 THEN
32882 5. 132 1 <DUP(NEG(VAL(<=>1, FROM)) FROM MAXILOC) LT ILOC
32895 5. 133 1 THEN<OUTST('Max free-space exceeded.') OPNL ESTOP>
32906 00119 5. 134 1 DUP->@MAXILOC, ELSE DETACHED(>)
32914 5. 135 1 EXIT
32915 5. 136 1 PAGE

```

```

5. 137 1 . Join item to a list
5. 138 1 . JOIN(<addr of item>, <addr of list pointer>)
5. 139 1 .
5. 140 1 VAR JLISTP:0
5. 141 1 JOIN:ENTRY ->@JLISTP, DUP, <=>(VAL(JLISTP))->, ->JLISTP EXIT
32929 00120 5. 142 1 .
5. 143 1 . Push down item into free space
5. 144 1 . PUSHHD(<object>, <addr of list pointer>)
5. 145 1 .
5. 146 1 PUSHHD:ENTRY
32930 5. 147 1 DUP, JOIN(NEXTFREE(@ITEM), <=>), VAL<=>1, FROM, ->,
32942 5. 148 1 EXIT
32943 5. 149 1 .
5. 150 1 . Pop-up item from free space
5. 151 1 . POPUP(<addr of list pointer>)
5. 152 1 .
5. 153 1 POPUP:ENTRY JOIN(DETACHED(), @ITEM) EXIT
32949 5. 154 1 .
5. 155 1 . Pop-up item and obtain it
5. 156 1 . POPPEDUP(<addr of list pointer>)
5. 157 1 .
5. 158 1 POPPEDUP:ENTRY VAL(VAL(DUP)<=>1, FROM), POPUP(<=>), EXIT
32960 5. 159 1 .
5. 160 1 . Operate on all members of a list
5. 161 1 . FOREACH(<addr of list pointer>, [<func>])
5. 162 1 .
5. 163 1 VAR FEACHL:0,2 . record list.
00121 5. 164 1 VAR LISTP:0 VAR FACT:0
00123 5. 165 1 .
5. 166 1 FOREACH:ENTRY
32961 00124 5. 167 1 PUSHHDREC(@LISTP, @FEACHL), ->@FACT, VAL->@LISTP
32970 5. 168 1 WHILE LISTP NE 0 START DO FACT(VAL(1 FROM LISTP)),
32990 5. 169 1 VAL(LISTP)->@LISTP
32993 5. 170 1 REPEAT
32999 5. 171 1 POPUPREC(@LISTP, @FEACHL),
33004 5. 172 1 EXIT
33005 5. 173 1 .
5. 174 1 . Clear all entries in an Item List
5. 175 1 . CLEARLST(<addr of list pointer>)
5. 176 1 .
5. 177 1 CLEARLST:ENTRY
33006 5. 178 1 WHILE VAL(DUP) NE 0 START POPUP(DUP) REPEAT LOSE
33019 5. 179 1 EXIT
33021 5. 180 1 PAGE

```

```
5. 181 1 . Record List routines
5. 182 1 . -----
5. 183 1 .
5. 184 1 VAR FLISTP:0 VAR FSPOINT:0 VAR RECORD:0
00126 5. 185 1 .
5. 186 1 . Obtain free space pointer
5. 187 1 .
5. 188 1 FN FSP:ENTRY NEG(VAL(1 FROM FLISTP)) EXIT
33030 00127 5. 189 1 .
5. 190 1 . Obtain free space length
5. 191 1 .
5. 192 1 FN FSL:ENTRY VAL(1 FROM FSP) EXIT
33037 5. 193 1 .
5. 194 1 . Initialize free space list pointer
5. 195 1 .
5. 196 1 FN INITL:ENTRY
33038 5. 197 1 NEXTFREE(@ITEM)->@FSPOINT, 0->FSPOINT,
33049 5. 198 1 NEG(FSP)->1 FROM FSPOINT, NEG(FSPOINT)->1 FROM FLISTP
33062 5. 199 1 EXIT
33067 5. 200 1 .
5. 201 1 . Push record on list
5. 202 1 . PUSHREC(<addr of record>, <addr of list pointer>)
5. 203 1 . see INITL re list pointer
5. 204 1 .
5. 205 1 PUSHREC:ENTRY
33068 5. 206 1 ->@FLISTP, ->@RECORD, FSP LT 0 THEN<INITL>
33082 5. 207 1 MOVE(FSL, RECORD, PUSH(DUP(NEXTFREE(FSP)), FLISTP))
33091 5. 208 1 EXIT
33093 5. 209 1 .
5. 210 1 . Pop record of list
5. 211 1 . POPREC(<addr of record>, <addr of list pointer>)
5. 212 1 .
5. 213 1 POPREC:ENTRY
33094 5. 214 1 ->@FLISTP, ->@RECORD, POPPEDUP(FLISTP)->@FSPOINT,
33106 5. 215 1 MOVE(FSL, FSPOINT, RECORD), JOIN(FSPOINT, FSP)
33115 5. 216 1 EXIT
33117 5. 217 1 PAGE
```

```

5. 218 1 . Standard binary lookup function for B-tree.
5. 219 1 .
5. 220 1 FN ABOVE: ENTRY GT THEN<1 ELSE 0> EXIT
33130 5. 221 1 .
5. 222 1 . Binary search function.
5. 223 1 . BINLOC( key, @first entry, # of entries)
5. 224 1 . returns: @entry GE key
5. 225 1 .
5. 226 1 ICON STRIDE 2
5. 227 1 VAR KEY:0, VAR N:0, VAR STEP:0, VAR LOCF:0,
00131 5. 228 1 BINLOC: ENTRY
33131 5. 229 1 ->@N, ->@LOCF, ->@KEY,
33140 5. 230 1 N LT 0 THEN< OUTST('BINLOC ERR. '), OPNL >
33152 00135 5. 231 1 N GT 0 THEN< WHILE N -->1 ->@STEP,
33168 5. 232 1 ABOVE( KEY, VAL(STRIDE*STEP FROM LOCF)) THEN<
33183 5. 233 1 STRIDE*STEP FROM LOCF->@LOCF, N-STEP ELSE STEP> ->@N,
33207 5. 234 1 N GT 1, YES BACK,
33215 5. 235 1 ABOVE(KEY, VAL(LOCF)) THEN< STRIDE FROM LOCF->@LOCF >> .N=1 case
33232 5. 236 1 LOCF
5. 237 1 EXIT
33235 5. 238 1 .
5. 239 1 . B-TREE Access Routines
5. 240 1 .
5. 241 1 . These routines provide for storage and retrieval of data stored
5. 242 1 . by key in a B-tree structure (See Knuth, Searching and Sorting,
5. 243 1 . Section 6.2.4). This version accepts data items, with duplicate
5. 244 1 . items for same key pushed down in an item list.
5. 245 1 .
5. 246 1 ICON NMAX 12, . Max number of entries in a node.
5. 247 1 ICON KINCR 2, . Size of an entry in words.
5. 248 1 ICON NDSIZ (KINCR*(NMAX+3)), . Node size.
5. 249 1 VAR TREEP: 0, VAR BKEY:0
00136 5. 250 1 VAR NKLST: 0, 3,
00139 5. 251 1 VAR NEWKEY:0, VAR NODEP:0, VAR PARREC:0,
00142 5. 252 1 VAR BDATA: 0
5. 253 1 FN NPINC: ENTRY 1 FROM NODEP->@NODEP EXIT
33245 00143 5. 254 1 FN NODE?: ENTRY VAL(MINUS 3 FROM PARREC) NE MINUS 1 EXIT
33256 5. 255 1 FN @LEAFN: ENTRY MINUS 3 FROM VAL(NODEP) EXIT
33264 5. 256 1 FN @NODEPAR: ENTRY MINUS 4 FROM VAL(NODEP) EXIT
33272 5. 257 1 FN @NENTS: ENTRY MINUS 2 FROM PARREC EXIT
33279 5. 258 1 FN NENTS: ENTRY VAL(@NENTS) EXIT
33283 5. 259 1 FN @PARENT: ENTRY MINUS 4 FROM PARREC EXIT
33290 5. 260 1 FN PARENT: ENTRY VAL(@PARENT) EXIT
33294 5. 261 1 FN @LOCVR: ENTRY MINUS 3 FROM VAL(TREEP) EXIT
33302 5. 262 1 FN LOCVR: ENTRY VAL(@LOCVR) EXIT
33306 5. 263 1 FN LOCFN: ENTRY DO LOCVR EXIT
33310 5. 264 1 PAGE

```

```

5. 265 1 . Function to build new Node.
5. 266 1 . Returns node link pointer.
5. 267 1 .
5. 268 1 VAR TMPNODE:0
5. 269 1 FN NEWNODE: ENTRY
33311 00144 5. 270 1 GETREC(NDSIZ)->@TMPNODE, 4 FROM TMPNODE, ADV(@BTCOUNT)
33324 5. 271 1 EXIT
33326 5. 272 1 .
5. 273 1 . Function to build new Leaf.
5. 274 1 .
5. 275 1 FN NEWLEAF: ENTRY
33327 5. 276 1 NEWNODE->NODEP, PARREC->@NODEPAR, MINUS 1 ->@LEAFN
33337 5. 277 1 EXIT
33340 5. 278 1 .
5. 279 1 . Function to set parent address in nodes.
5. 280 1 .
5. 281 1 FN SETPAR: ENTRY
33341 5. 282 1 NENTS+1, MINUS 1 FROM NODEP->@NODEP,
33353 5. 283 1 WHILE DUP(-1) GE 0 START PARREC->@NODEPAR,
33367 5. 284 1 2 FROM NODEP->@NODEP REPEAT LOSE
33378 5. 285 1 EXIT
33380 5. 286 1 .
5. 287 1 . B-TREE search function.
5. 288 1 . BTRCH( key, @root)
5. 289 1 .
5. 290 1 FN BTRCH: ENTRY
33381 5. 291 1 ->@TREEP, ->@BKEY,
33387 5. 292 1 VAL(TREEP) EQ 0 THEN<OUTST('B-Tree not initialized. '), OPNL, EXIT>
33401 00151 5. 293 1 VAL(TREEP)->@PARREC, WHILE NODE? START
33411 5. 294 1 VAL(MINUS 1 FROM LOCFN(BKEY, PARREC, NENTS)) ->@PARREC,
33424 5. 295 1 REPEAT PARREC->@NODEP
33429 5. 296 1 EXIT
33433 5. 297 1 .
5. 298 1 VAR SPLITLST:0, VAR ND2:0, VAR CPT:0,
00154 5. 299 1 FN NODESPLIT: ENTRY
33434 5. 300 1 ->@PARREC, NENTS-->1->@NENTS, NODE? THEN<NEWNODE->@ND2,
33451 5. 301 1 PARREC->(MINUS 4 FROM ND2), NENTS->(MINUS 2 FROM ND2),
33466 5. 302 1 KINCR*(NMAX-->1)->@CPT,
33477 5. 303 1 MOVE(CPT+1, CPT-3 FROM PARREC, MINUS 1 FROM ND2),
33496 5. 304 1 MOVE(2, KINCR*(NMAX-1) FROM PARREC, CPT-2 FROM PARREC)
33517 5. 305 1 DUP(ND2)->(CPT-3 FROM PARREC), DUP->@NODEP, ->@PARREC, SETPAR,
33538 5. 306 1 ELSE
33541 5. 307 1 NEWNODE->@NODEP, NENTS->(MINUS 2 FROM NODEP),
33552 5. 308 1 MINUS 1->(MINUS 3 FROM NODEP),
33560 5. 309 1 PARENT ->(MINUS 4 FROM NODEP),
33567 5. 310 1 MOVE(KINCR*NENTS, KINCR*NENTS FROM PARREC, NODEP),
33581 5. 311 1 VAL(NODEP)-1->@NEWKEY, BKEY GE VAL(NODEP) THEN<NODEP->@PARREC>,
33604 5. 312 1 PUSHDREC(@NEWKEY,@NKLST)>
33609 5. 313 1 EXIT
33610 5. 314 1 PAGE

```



```

5. 315 1 . Function to split nodes on overflow.
5. 316 1 . BKEYINST
5. 317 1 .
5. 318 1 FN BINSRT
5. 319 1 FN BKEYINST: ENTRY
33611 5. 320 1 WHILE (PARENT NE 0) AND (NENTS EQ NMAX) START
33623 5. 321 1     PUSH(D(PARREC, @SPLITLST) PARENT->@PARREC, REPEAT
33635 5. 322 1     NENTS EQ NMAX THEN< PUSH(D(PARREC, @SPLITLST)>
33647 5. 323 1     FOREACH(@SPLITLST, [NODESPLIT]), CLEARLST(@SPLITLST),
33661 5. 324 1     WHILE NKLST NE 0 START POPUPREC(@NEWKEY, @NKLST), NODEP,
33676 5. 325 1     PARENT->@NODEP, NEWKEY->@BKEY, BINSRT([->(1 FROM NODEP)]),
33699 5. 326 1     REPEAT
33702 5. 327 1     EXIT
33703 5. 328 1 .
5. 329 1 . Function to insert into found node.
5. 330 1 . BINSRT( [insert action])
5. 331 1 .
5. 332 1 VAR BFACT:0
5. 333 1 BINSRT: ENTRY
33704 00155 5. 334 1     ->@BFACT, NODEP->@PARREC, LOCFN(BKEY, NODEP, NENTS) ->@NODEP,
33721 5. 335 1     BKEY EQ VAL(NODEP) THEN< DO BFACT EXIT>
33734 5. 336 1     NENTS LT NMAX THEN<
33741 5. 337 1     MOVE( KINCR*(NENTS+1)- (NODEP ADIFF PARREC), NODEP,
33756 5. 338 1     KINCR FROM NODEP), BKEY->NODEP, 0->(1 FROM NODEP),
33775 5. 339 1     DO BFACT, ADV(@NENTS),
33780 5. 340 1     ELSE
33783 5. 341 1     BKEY, BKEYINST, ->@BKEY, BTINSRT(BDATA, BKEY, TREEP) >
33796 5. 342 1     EXIT
33797 5. 343 1 .
5. 344 1 . B-TREE initialization function.
5. 345 1 . BTINIT( lower, upper, [lookup function], @tree)
5. 346 1 .
5. 347 1 BTINIT: ENTRY
33798 5. 348 1     ->@TREEP, NEWNODE->TREEP, ->@LOCVR, DUP(VAL(TREEP))
33810 5. 349 1     ->@NODEP, ->@PARREC, 2->@NENTS, <=>,
33822 5. 350 1     (MINUS 1 FROM NODEP)->@NODEP, NEWLEAF, NPINC, ->NODEP,
33835 5. 351 1     NPINC, NEWLEAF, NPINC, ->NODEP, NPINC, NEWLEAF,
33843 5. 352 1     EXIT
33844 5. 353 1 .
5. 354 1 . B-TREE insert function.
5. 355 1 . BTINSRT( data, key, @root)
5. 356 1 .
5. 357 1 BTINSRT: ENTRY
33845 5. 358 1     BTRCH, ->@BDATA, BINSRT([PUSH(D(BDATA, 1 FROM NODEP)]))
33864 5. 359 1     EXIT
33866 5. 360 1 PAGE

```

```

5. 361 1 VAR FACT:0, VAR VLSTP:0
00156 5. 362 1 FORBTVAL: ENTRY
33867 00157 5. 363 1 ->@TREEP, ->@FACT, ->@BKEY, BTRCH(BKEY, TREEP),
33881 5. 364 1 NODEP->@PARREC, LOCFN(BKEY, NODEP, NENTS)->@NODEP,
33895 5. 365 1 BKEY EQ VAL(NODEP) THEN< VAL(1 FROM NODEP)->@VLSTP, GLKP,
33914 5. 366 1 WHILE VLSTP NE 0 START DO FACT(VAL(1 FROM VLSTP)),
33931 5. 367 1 VAL(VLSTP)->@VLSTP
33934 5. 368 1 REPEAT LOSE>
33941 5. 369 1 EXIT
33942 5. 370 1 .
5. 371 1 . B-TREE remove data function. Removes all occurrences of data value.
5. 372 1 . BTREM( data, key, @root)
5. 373 1 .
5. 374 1 BTREM: ENTRY
33943 5. 375 1 BTRCH, ->@BDATA, LOCFN(BKEY, NODEP, NENTS)->@NODEP,
33956 5. 376 1 BKEY EQ VAL(NODEP) THEN< NPINC,
33966 5. 377 1 WHILE VAL(NODEP) NE 0 START
33975 5. 378 1 BDATA EQ VAL(1 FROM VAL(NODEP)) THEN< POPUP(NODEP)
33990 5. 379 1 ELSE VAL(NODEP)->@NODEP>
34000 5. 380 1 REPEAT>
34003 5. 381 1 EXIT
34004 5. 382 1 .
5. 383 1 . Utility function for tree delete.
5. 384 1 .
5. 385 1 VAR DELLST:0
5. 386 1 FN NDEL:ENTRY
34005 00158 5. 387 1 DUP(VAL)->@PARREC,->@NODEP,
34013 5. 388 1 NODE? THEN<MINUS 2 FROM NODEP->@NODEP, NENTS+1,
34029 5. 389 1 WHILE DUP(-1) GE 0 START NPINC, PUSH(VAL(NODEP), @DELLST),
34046 5. 390 1 VAL(NODEP)->@PARREC, NODEP, PARREC, NDEL(@PARREC),
34059 5. 391 1 ->@PARREC, ->@NODEP, NPINC,
34066 5. 392 1 REPEAT, LOSE,
34070 5. 393 1 ELSE
34073 5. 394 1 NENTS WHILE DUP(-1) GE 0 START NPINC, CLEARLST(NODEP),
34088 5. 395 1 NPINC REPEAT
34092 5. 396 1 LOSE>
34093 5. 397 1 EXIT
34094 5. 398 1 .
5. 399 1 . Delete B-tree function.
5. 400 1 . BTDEL( @tree)
5. 401 1 .
5. 402 1 BTDEL: ENTRY
34095 5. 403 1 ->@TREEP, NDEL(TREEP),
34101 5. 404 1 FOREACH(@DELLST, [->@NODEP, RELREC(MINUS 4 FROM NODEP, NDSIZ),
34118 5. 405 1 BCOUNT-1->@BCOUNT]), CLEARLST(@DELLST),
34133 5. 406 1 RELREC(MINUS 4 FROM VAL(TREEP), NDSIZ), BCOUNT-1->@BCOUNT,
34150 5. 407 1 0->TREEP,
34155 5. 408 1 EXIT
34156 5. 409 1 ENDBLOCK
5. 410 MAUTO PAGE

```

```

5. 411 MAUTO . String access and matching routines
5. 412 MAUTO . -----
5. 413 MAUTO .
5. 414 MAUTO BLOCK
5. 415 1 . Get character and put character routines.
5. 416 1 . GETCH(@CAP) PUTCH(@CAP, char)
5. 417 1 .
5. 418 1 VAR STRAD:0 VAR CHNO:0
00159 5. 419 1 VAR CMSK:&((255<--24)+(255<--16)+(255<--8)),
00161 5. 420 1 &((255<--24)+(255<--16)+255),
00162 5. 421 1 &((255<--24)+(255<--8)+255),
00163 5. 422 1 &((255<--16)+(255<--8)+255),
00164 5. 423 1 GETCH:ENTRY
34157 5. 424 1 DUP->@STRAD, VAL(1 FROM STRAD)->@CHNO, DUP(VAL)->@STRAD,
34175 5. 425 1 VAL() GT CHNO THEN<VAL((1+(CHNO-->2)) FROM STRAD),
34194 5. 426 1 -->((CHNO MASK 3)<--3), MASK 255
34203 5. 427 1 ELSE 0>
34211 5. 428 1 EXIT
34212 5. 429 1
5. 430 1 PUTCH:ENTRY
34213 5. 431 1 <=>, ->@STRAD, VAL(1 FROM STRAD)->@CHNO,
34226 5. 432 1 (1+(CHNO-->2)) FROM VAL(STRAD)->@STRAD,
34241 5. 433 1 VAL(STRAD) MASK VAL((CHNO MASK 3) FROM @CMSK),
34254 5. 434 1 <=>, <--((CHNO MASK 3)<--3), UNION, ->STRAD,
34268 5. 435 1 EXIT
34269 5. 436 1
5. 437 1 VAR FCHFN:@NULL, VAR CHP:0,
00166 5. 438 1 FN DOFCH:ENTRY
34270 5. 439 1 <=>, DUP GE 0 THEN<LOSE, LOSE, 0
34280 5. 440 1 ELSE <=>, MASK 255, DO FCHFN, ()+1, 1 >
34297 5. 441 1 EXIT
34298 5. 442 1
5. 443 1 FORCHS:ENTRY
34299 5. 444 1 ->@FCHFN, ->@CHP, NEG(VAL(CHP)),
34309 5. 445 1 WHILE 1 FROM CHP->@CHP,
34317 5. 446 1 VAL(CHP) , DOFCH, FALSE EXIT
34323 5. 447 1 VAL(CHP)--> 8, DOFCH, FALSE EXIT
34332 5. 448 1 VAL(CHP)-->16, DOFCH, FALSE EXIT
34341 5. 449 1 VAL(CHP)-->24, DOFCH, FALSE EXIT
34350 5. 450 1 GO BACK
34353 5. 451 1 .
5. 452 1 .SI MATCH4C
5. 453 1 ENDBLOCK
5. 454 MAUTO PAGE

```

```

5. 455 MAUTO      .      String input routines
5. 456 MAUTO      .      -----
5. 457 MAUTO      .
5. 458 MAUTO      .      IAUTO
5. 459 IAUTO      VAR ESCHAR:0      . escape character
5. 460 IAUTO      VAR ESCTAB:<>      . escape character table
00168 5. 461 IAUTO      <'S', # >
00172 5. 462 IAUTO      <'CR', 13>
00176 5. 463 IAUTO      <'FF', 12>
00180 5. 464 IAUTO      <';', #'>
00184 5. 465 IAUTO      VAR ESCEND:,ESCEND,      . end of table
00185 5. 466 IAUTO      MAUTO
5. 467 MAUTO      BLOCK
5. 468      1      ICON cr 13
5. 469      1      .
5. 470      1      .      Skip blanks routine
5. 471      1      .
5. 472      1      BLANKS:ENTRY WHILE CHAR EQ #      START ADVCH REPEAT EXIT
34366 5. 473      1      .
5. 474      1      .      Get string of input characters
5. 475      1      .      GETSTR(<unit number>, <max words>, <buffer address>)
5. 476      1      .
5. 477      1      VAR STRAD:0 VAR CHNO:0
00186 5. 478      1      VAR LIM:0      VAR CHAN:0 VAR SCHR:0
00189 5. 479      1      VAR BKSP:8      VAR BLANK:32
00191 5. 480      1      VAR EOFFLAG:0 . UGH!!!
00192 5. 481      1      FN TCHAR:ENTRY
34367 00193 5. 482      1      (CHAN EQ 0) THEN<
34375 5. 483      1      WHILE ((DUP(INCH(CHAN)) EQ BKSP) AND (SCHR GT 0)) START
34391 5. 484      1      LOSE, OPCH(SOUNIT, BLANK), OPCH(SOUNIT, BKSP),
34402 5. 485      1      (MASK(3, SCHR) EQ 0) THEN< (MINUS 1 FROM STRAD) -> @STRAD,
34421 5. 486      1      LOSE, VAL(STRAD) >
34425 5. 487      1      SCHR-1->@SCHR, MASK( COMPL(255<--(MASK(3, SCHR)*8))),
34446 5. 488      1      REPEAT, ELSE INCH(CHAN) >
34455 5. 489      1      (DUP EQ cr) THEN< 0, EXIT>
34465 5. 490      1      (DUP EQ 256) THEN< LOSE, DUP(0), EXIT>
34477 5. 491      1      1, EXIT
34480 5. 492      1      GETSTR:ENTRY
34481 5. 493      1      DUP->@CHNO, 0->@SCHR, <=>, -1, <=>, FROM, ->@LIM, ->@CHAN,
34502 5. 494      1      1 FROM CHNO ->@STRAD,
34510 5. 495      1      0 WHILE TCHAR AND (STRAD LE LIM) START,
34522 5. 496      1      UNION(<--(MASK(3, SCHR)*8)),
34532 5. 497      1      ((MASK(3, SCHR)*8) EQ 24) THEN< -> STRAD,
34549 5. 498      1      1 FROM STRAD -> @STRAD, 0>
34559 5. 499      1      SCHR+1->@SCHR,
34567 5. 500      1      REPEAT,
34570 5. 501      1      DUP->@EOFFLAG,
34574 5. 502      1      EQ 0 THEN< 0 ELSE cr> UNION( <--(MASK(3, SCHR)*8))->STRAD,
34600 5. 503      1      SCHR, EOFFLAG NE 0 THEN < +1> -> CHNO,
34616 5. 504      1      EXIT
34617 5. 505      1      PAGE

```

```

5. 506 1 . Read input image
5. 507 1 .
5. 508 1 READ:ENTRY
34618 5. 509 1 CURCHS EQ 0 THEN<NEXTFREE(@BUFFER)->@CURCHS>,
34632 5. 510 1 WHILE GETSTR(SIUNIT, 34, CURCHS) VAL(CURCHS) EQ 0 THEN<
34648 5. 511 1 INPST NE 0 THEN<CLOSEF(SIUNIT) POPUPREC(@SIUNIT, @INPST)
34663 5. 512 1 ELSE EXIT>
34668 5. 513 1 GO BACK> . read new string
34671 5. 514 1 0->@CURCOL, . initialize cap
34676 5. 515 1 EXIT
34677 5. 516 1 .
5. 517 1 . Advance character pointer
5. 518 1 .
5. 519 1 ADVCH:ENTRY ADV(@CURCOL) EXIT
34682 5. 520 1 .
5. 521 1 . Obtain current character
5. 522 1 .
5. 523 1 CHAR:ENTRY
34683 5. 524 1 WHILE DUP(GETCH(@CURCHS)) EQ 0 START
34693 5. 525 1 LOSE DO EOSTRACT
34694 5. 526 1 REPEAT
34700 5. 527 1 EXIT
34701 5. 528 1 .
5. 529 1 . Obtain next character with escape translation
5. 530 1 .
5. 531 1 VAR REPC:0
5. 532 1 NEXTCH:ENTRY
34702 00194 5. 533 1 DUP(CHAR), ADVCH NE ESCHAR THEN <EXIT> LOSE,
34713 5. 534 1 DUP(VAL(DICMATCH(@ESCTAB, ESCEND, @CURCHS)))
34721 5. 535 1 EQ ESCEND THEN <LOSE CHAR ADVCH EXIT>
34732 5. 536 1 ->@REPC, VAL((2+(VAL(1 FROM REPC)+3)-->2) FROM REPC),
34754 5. 537 1 EXIT
34755 5. 538 1 .
5. 539 1 . Build string from input (NEXTCH)
5. 540 1 . INSTRING(<addr of string>, [<term func>])
5. 541 1 .
5. 542 1 VAR CSADD:0 VAR CSCOL:0 . cap
00195 5. 543 1 VAR CSPNT:0 . address of string
00196 5. 544 1 INSTRING:ENTRY <=>, VAL(DUP), ->@CSADD, ->@CSPNT, 0->@CSCOL,
34770 00197 5. 545 1 WHILE DUP, DO START
34775 5. 546 1 PUTCH(@CSADD, NEXTCH) ADV(@CSCOL), CSCOL->@NOCHS,
34787 5. 547 1 REPEAT, LOSE, CSCOL MASK 1 THEN< PUTCH(@CSADD, 0)>
34804 5. 548 1 DUP(CSCOL)->CSADD, (+7)-->2 FROM CSADD->CSPNT,
34822 5. 549 1 EXIT
34823 5. 550 1 PAGE

```

```
5. 551 1 . Read string until space or CR
5. 552 1 .
5. 553 1 FNAME:ENTRY
34824 5. 554 1 DUMP(DUP(DLOC)) BLANKS . leave string address on stack
34828 5. 555 1 INSTRING(@TEMP, [CHAR NE # AND CHAR NE cr])
34847 5. 556 1 EXIT
34849 5. 557 1 .
5. 558 1 . Obtain a decimal number
5. 559 1 .
5. 560 1 ICON INCNST (((65536*32767+65535)/10)+1)
5. 561 1 ININT:ENTRY
34850 5. 562 1 0, WHILE DUP LE INCNST AND DIGIT START *10 + NEXTCH - #0, REPEAT
34872 5. 563 1 EXIT
34873 5. 564 1 .
5. 565 1 . Obtain a HEX number
5. 566 1 .
5. 567 1 FN HDIGIT:ENTRY DUP(CHAR) GE #0, <=>, AND LE 70 EXIT
34885 5. 568 1 INHEX:ENTRY
34886 5. 569 1 0, WHILE HDIGIT START <--4 + (NEXTCH, DUP GT #9 CHOOSE(55,#0), -)
34906 5. 570 1 REPEAT
34910 5. 571 1 EXIT
34911 5. 572 1 .
5. 573 1 . Letter test
5. 574 1 .
5. 575 1 LETTER:ENTRY
34912 5. 576 1 (DUP(CHAR) GE #A, DUP, <=>, AND LE #Z), <=>,
34924 5. 577 1 OR (DUP GE #a, <=>, AND LE #z)
34933 5. 578 1 EXIT
34935 5. 579 1 .
5. 580 1 . Digit test
5. 581 1 .
5. 582 1 DIGIT:ENTRY DUP(CHAR) GE #0, <=>, AND LE #9 EXIT
34947 5. 583 1 ENDBLOCK
5. 584 MAUTO PAGE
```

```

5. 585 MAUTO      .      Output routines
5. 586 MAUTO      .      -----
5. 587 MAUTO      .
5. 588 MAUTO      .      IAUTO
5. 589 IAUTO      VAR RLNO:0          . controls PAGE output
5. 590 IAUTO      VAR PGLNGTH:0      . page length
00198 5. 591 IAUTO      MAUTO
5. 592 MAUTO      DIR PAGE          . skip to new page
5. 593 MAUTO      BLOCK
5. 594      1      ICON cr 13
00199 5. 595      1      ICON ff 12
5. 596      1      .
5. 597      1      .      Output string routine
5. 598      1      .      OUTST(<string address>)
5. 599      1      .
34961 5. 600      1      OUTST:ENTRY FORCHS( , [OPCH(SOUNIT, <=>)]) EXIT
5. 601      1      .
5. 602      1      .      Output new line
5. 603      1      .
5. 604      1      OPNL:ENTRY OPCH(SOUNIT, cr), PGLNGTH EQ 0 THEN<EXIT>,
34976 5. 605      1      RLNO GE PGLNGTH THEN<REF PAGE>, ADV(@RLNO), EXIT
34989 5. 606      1      .
5. 607      1      .      Output form feed
5. 608      1      .
34996 5. 609      1      OPFF:ENTRY OPCH(SOUNIT, ff) EXIT
5. 610      1      .
5. 611      1      .      Output blank characters
5. 612      1      .
5. 613      1      OB: ENTRY OPCH(SOUNIT, # ) EXIT
35003 5. 614      1      OBW:ENTRY OUTST(@BLANK6), WIDTH-5,
35012 5. 615      1      WHILE DUP(-1) GE 0 START OB REPEAT LOSE
35026 5. 616      1      EXIT
35028 5. 617      1      .
5. 618      1      .      Output integer routine
5. 619      1      .      OPINT(<integer number>)
5. 620      1      .
5. 621      1      MACRO TIMES: ', WHILE -1, DUP GE 0 START,'
00207 5. 622      1      MACRO AGAIN: ', REPEAT, LOSE,'
00212 5. 623      1      FN OC: ENTRY OPCH(SOUNIT, <=>) EXIT
35034 5. 624      1      VAR BSHIFT:      0, VAR BM1:      0,
00214 5. 625      1      VAR DIGITCOUNT: 0, VAR CHARLOSE: 0,
00216 5. 626      1      VAR CHARTABLE: #0, #1, #2, #3, #4, #5, #6, #7,
00224 5. 627      1      #8, #9, #A, #B, #C, #D, #E, #F,
00232 5. 628      1      OPINT:ENTRY
35035 5. 629      1      DUP((DUP LT 0) AND (BASE EQ 10)),
35046 5. 630      1      CHOOSE(#-, # ), OC, TRUE NEG, OPINTD,
35055 5. 631      1      EXIT
35056 5. 632      1      PAGE

```

```
5. 633 1 . Output integer magnitude routine
5. 634 1 . OPINTD(<integer magnitude>)
5. 635 1 .
5. 636 1 OPINTD:ENTRY
35057 5. 637 1 0->@CHARLOSE, 0->@DIGITCOUNT,
35067 5. 638 1 BASE EQ 10 THEN<
35075 5. 639 1 WHILE ()/BASE, DREM, <=>, ADV(@DIGITCOUNT), DUP EQ 0,
35088 5. 640 1 NO BACK, LOSE,
35092 5. 641 1 ELSE
35095 5. 642 1 WHILE () DUP MASK BM1, <=>, -->BSHIFT, ADV(@DIGITCOUNT),
35106 5. 643 1 DUP EQ 0, NO BACK, LOSE>
35114 5. 644 1 WIDTH EQ 0 CHOOSE(DIGITCOUNT, WIDTH) ->@CWIDTH,
35127 5. 645 1 CWIDTH-DIGITCOUNT,
35132 5. 646 1 DUP LT 0 THEN< NEG(-1) ->@CHARLOSE, OC(#?),
35149 5. 647 1 CWIDTH-1 ->@DIGITCOUNT
35154 5. 648 1 ELSE TIMES OC(PAD) AGAIN>
35177 5. 649 1 DIGITCOUNT TIMES <=>, OC(VAL( , FROM @CHARTABLE)), AGAIN
35199 5. 650 1 CHARLOSE TIMES <=>, LOSE AGAIN, #0->@PAD,
35222 5. 651 1 EXIT
35223 5. 652 1 .
5. 653 1 . Set OPINT parameters
5. 654 1 . SETOPP(<pad char>, <no of digits>)
5. 655 1 . SETBSE(<pad char>, <no. of digits>, <base>)
5. 656 1 .
5. 657 1 SETOPP:ENTRY ->@WIDTH, ->@PAD, EXIT
35231 5. 658 1 SETBSE:ENTRY
35232 5. 659 1 DUP ->@BASE, -1 ->@BM1,
35242 5. 660 1 BASE EQ 2 CHOOSE(1,3) ->@BSHIFT,
35255 5. 661 1 BASE EQ 16 THEN<4->@BSHIFT> SETOPP
35268 5. 662 1 EXIT
35270 5. 663 1 PAGE
```



```

5. 664 1 . Record pool handling routines
5. 665 1 . -----
5. 666 1 .
5. 667 1 . These routines manage a record pool which expands from the
5. 668 1 . initial value of MAXRLOC. Records of even length are allocated.
5. 669 1 . Released records are pooled and records are reallocated from
5. 670 1 . available space. The value of MAXRLOC is adjusted in case
5. 671 1 . space within the current pool is not available.
5. 672 1 .
5. 673 1 . These routines are used by the compiler dictionary
5. 674 1 . routines: NEWDREC, DELETE, FORGET, RENAME and INITDIC.
5. 675 1 .
5. 676 1 . GETREC(length)          acquire a record of length: length
5. 677 1 .                      address of record is returned on stack
5. 678 1 . RELREC(bufaddr, length) release a record at bufaddr, of
5. 679 1 .                      length: length
5. 680 1 .
5. 681 1 VAR RPLPTR:0, 0 VAR FRALOC:128
00234 5. 682 1 VAR REQS:0      VAR PTR:0
00236 5. 683 1 VAR EXBLK:0     VAR EXSIZ:0 VAR X:0
00239 5. 684 1 FN GETREC$
5. 685 1 FN RELREC$
5. 686 1 FN ERPAS2$
5. 687 1 .
5. 688 1 GETREC:ENTRY
35271 00240 5. 689 1 DUP+MASK 1->@REQS, GETREC$,
35280 5. 690 1 DUP EQ 0 THEN<LOSE, (REQS+FRALOC-1)/FRALOC*FRALOC->@EXSIZ,
35305 5. 691 1 DUP(NEG(EXSIZ) FROM MAXRLOC)->@MAXRLOC, ->@EXBLK, RELREC$,
35319 5. 692 1 GETREC$, DUP EQ 0 THEN<
35327 5. 693 1 OUTST('Unable to satisfy GETREC request for'),
35330 00250 5. 694 1 OPINT(REQS), OUTST(' words'), OPNL, ESTOP>>
35338 00253 5. 695 1 DUMP(DUP+REQS), WHILE DUP LT TEMP START
35350 5. 696 1 DUMP(MINUS 1 FROM TEMP), 0->TEMP
35358 5. 697 1 REPEAT
35364 5. 698 1 EXIT
35365 5. 699 1 GETREC$:ENTRY
35366 5. 700 1 MAXRLOC LE RLOC THEN<0, EXIT>
35377 5. 701 1 @RPLPTR->@PTR, WHILE DUP(VAL(PTR))->@X, NE 0 START
35395 5. 702 1 DUP(VAL(1 FROM X)-REQS)->@EXSIZ, GE 0 THEN<X, VAL(DUP)->PTR,
35421 5. 703 1 EXSIZ GT 0 THEN<DUP->@EXBLK, ERPAS2$(EXSIZ, <=>, FROM)>
35438 5. 704 1 EXIT>, X->@PTR
35441 5. 705 1 REPEAT 0,
35449 5. 706 1 EXIT
35450 5. 707 1 .
5. 708 1 RELREC:ENTRY
35451 5. 709 1 DUP+MASK 1->@EXSIZ, ->@EXBLK,
35462 5. 710 1 RELREC$ EXIT
35464 5. 711 1 PAGE

```

```

5. 712 1 RELREC$:ENTRY,
35465 5. 713 1 LAB ERLUP$:
5. 714 1 @RPLPTR->@PTR, WHILE DUP (VAL (PTR))->@X, NE 0 START
35483 5. 715 1 X+VAL (1 FROM X) EQ EXBLK THEN<VAL (X)->PTR, X->@EXBLK,
35509 5. 716 1 EXSIZ+VAL (1 FROM X)->@EXSIZ, GO ERLUP$>
35524 5. 717 1 X EQ (EXSIZ FROM EXBLK) THEN<VAL (X)->PTR,
35541 5. 718 1 EXSIZ+VAL (1 FROM X)->@EXSIZ, GO ERLUP$>
35556 5. 719 1 X->@PTR, REPEAT
35564 5. 720 1 GO <
35567 5. 721 1 ERPAS2$:ENTRY>
35568 5. 722 1 EXSIZ->1 FROM EXBLK, @RPLPTR->@PTR,
35581 5. 723 1 WHILE DUP (VAL (PTR))->@X, NE 0 START
35594 5. 724 1 EXSIZ LE VAL (1 FROM X) THEN<EXSIZ LT VAL (1 FROM X)
35613 5. 725 1 OR EXBLK GT X THEN<X->EXBLK, EXBLK->PTR, EXIT>>
35635 5. 726 1 X->@PTR, REPEAT 0->EXBLK, EXBLK->PTR,
35653 5. 727 1 EXIT
35654 5. 728 1 .
5. 729 1 REF POOL:ENTRY DUP (RPLPTR) EQ 0 THEN<OUTST ('RECPOOL is empty'),
35667 00258 5. 730 1 OPNL, LOSE, EXIT ELSE OUTST ('RECPOOL current/limit(K): '),
35676 00266 5. 731 1 OPINTD (MAXRLOC-->10), OUTST ('/'), OPINTD (RLOC-->10), OPNL,
35692 00268 5. 732 1 OUTST ('Free RECPOOL list'), OPNL>
35696 00274 5. 733 1 WIDTH, <=>, SETBSE (#0,8,10),
35706 5. 734 1 WHILE DUP NE 0 START
35713 5. 735 1 OPINTD (VAL (FROM (DUP<=>1))), OUTST (' words at '),
35723 00278 5. 736 1 OPINTD (DUP), OPNL, VAL,
35727 5. 737 1 REPEAT LOSE, ->@WIDTH, OUTST ('*** End of RECPOOL'), OPNL,
35738 00284 5. 738 1 EXIT
35739 5. 739 1 ENDBLOCK
5. 740 MAUTO PAGE

```

```
3. 12 MAUTO      COMP
5.  1 MAUTO      .
5.  2 MAUTO      .
5.  3 MAUTO      .
5.  4 MAUTO      .
5.  5 MAUTO      .
5.  6 MAUTO      .
5.  7 MAUTO      .
5.  8 MAUTO      .
5.  9 MAUTO      .
5. 10 MAUTO      .
5. 11 MAUTO      .
5. 12 MAUTO      .
5. 13 MAUTO      .
5. 14 MAUTO      .
5. 15 MAUTO      .
5. 16 MAUTO      .
5. 17 MAUTO      .
5. 18 MAUTO      .
5. 19 MAUTO      .
5. 20 MAUTO      .
5. 21 MAUTO      .
5. 22 MAUTO      .
5. 23 MAUTO      .
5. 24 MAUTO      .
5. 25 MAUTO      .
5. 26 MAUTO      .
5. 27 MAUTO      .
5. 28 IAUTO      VAR PBASE:0
5. 29 IAUTO      FN  PROGEND
5. 30 IAUTO      .
5. 31 IAUTO      .
5. 32 IAUTO      .
5. 33 IAUTO      FN  ONAUTO:ENTRY PBASE NE 0 EXIT
5. 34 IAUTO      PAGE

                                The MINT Compiler
                                -----

Based on implementation of SNIBBOL by D. F. Hendry
and R. K. Hessenberg, October 1974.

Enhancements and corrections by
H. Hermans and M. D. Godfrey, October 1979.

@Copyright D. F. Hendry.

The SNIBBOL system was renamed the
MINT (Machine-independent Organic Software Tools)
system, November 1979.

@Copyright under the name MINT D.F. Hendry 1979.

This is version 3.4X.

The purpose of this version is to implement a new
dictionary mechanism. The purpose of this mechanism is
to provide a framework for future language development.
Therefore, no special attempt is being made to maintain
compatibility with previous MINT systems.
                                M.D. Godfrey          1988

IAUTO
VAR PBASE:0                      . start of new compiler in PSPACE.
FN  PROGEND                      . end of compiler program
.
ONAUTO returns TRUE if Auto-compilation is active
.
FN  ONAUTO:ENTRY PBASE NE 0 EXIT
PAGE
```

35746 00285

```
5. 35 IAUTO . Introduce identifier class functions
5. 36 IAUTO .
5. 37 IAUTO FN IDINTRO . identifier intro action
5. 38 IAUTO FN OPINTRO . operator intro action
5. 39 IAUTO FN CNINTRO . integer constant intro action
5. 40 IAUTO FN SHUNT . shunt routine
5. 41 IAUTO FN SHUNTIM . force shunt
5. 42 IAUTO FN DIRECT . directive syntax action
5. 43 IAUTO FN MACACT . macro syntax action
5. 44 IAUTO FN DICACT . DICT syntax action
5. 45 IAUTO FN LABGEN . generative action for label
5. 46 IAUTO FN VARGEN . variable
5. 47 IAUTO FN FNGEN . function
5. 48 IAUTO FN PRIMGEN . primitive
5. 49 IAUTO FN ADRGEN . address const
5. 50 IAUTO FN STRGEN . string const
5. 51 IAUTO FN INTGEN . integer const
5. 52 IAUTO FN DICGEN . dictionary
5. 53 IAUTO FN GENCON . generate integer constant
5. 54 IAUTO FN SETPROG . implicit set prog state
5. 55 IAUTO FN SETDATA . implicit set data state
5. 56 IAUTO MAUTO
5. 57 MAUTO PAGE
```

```

5. 58 MAUTO .      Introduce CLASSEs.
5. 59 MAUTO .
5. 60 MAUTO .      syntax  generative  setting
5. 61 MAUTO .      action   action     action
5. 62 MAUTO .      -----
5. 63 MAUTO CLASS CLASS:  IDINTRO,  LABGEN,  SETDATA,
00288 5. 64 MAUTO      IAUTO
5. 65 IAUTO CLASS OPER:  OPINTRO,  LABGEN,  SETDATA,
00291 5. 66 IAUTO CLASS CCNST:  CNINTRO,  LABGEN,  SETDATA,
00294 5. 67 IAUTO .
5. 68 IAUTO CLASS INTCON:  0,      INTGEN,
00296 5. 69 IAUTO CLASS STRCON:  0,      STRGEN,
00298 5. 70 IAUTO CLASS ADRCON:  0,      ADRGEN,
00300 5. 71 IAUTO .
5. 72 IAUTO .      Define operator classes
5. 73 IAUTO .
5. 74 IAUTO OPER PRIMFN:  SHUNT,    PRIMGEN,
00302 5. 75 IAUTO OPER PRIMOP:  SHUNT,    PRIMGEN,
00304 5. 76 IAUTO OPER EXITT:  SHUNTIM,  PRIMGEN,
00306 5. 77 IAUTO .
5. 78 IAUTO      MAUTO
5. 79 MAUTO .
5. 80 MAUTO CLASS LAB:  SHUNT,    LABGEN,  NULL,
00309 5. 81 MAUTO CLASS VAR:  SHUNT,    VARGEN,  SETDATA,
00312 5. 82 MAUTO CLASS DIR:  DIRECT,  FNGEN,  SETPROG,
00315 5. 83 MAUTO CLASS FN:  SHUNT,    FNGEN,  SETPROG,
00318 5. 84 MAUTO CLASS MACRO:  MACACT,  LABGEN,  SETDATA,
00321 5. 85 MAUTO CLASS DICT:  DICTACT,  DICGEN,  SETDATA,
00324 5. 86 MAUTO .
5. 87 MAUTO .      Define non-relocatable constant
5. 88 MAUTO .
5. 89 MAUTO CCNST ICON:  SHUNT,    GENCON,  NULL,
00327 5. 90 MAUTO PAGE

```

```
5. 91 MAUTO . Define system primitives
5. 92 MAUTO .
5. 93 MAUTO PRIMFN GET 1
5. 94 MAUTO PRIMFN GETV 2
5. 95 MAUTO PRIMFN VAL 3
5. 96 MAUTO PRIMFN DUP 5
5. 97 MAUTO PRIMFN LOSE 6
5. 98 MAUTO PRIMFN GLKP 7
5. 99 MAUTO PRIMFN SLKP 8
5. 100 MAUTO PRIMFN NEG 13
5. 101 MAUTO PRIMFN COMPL 18
5. 102 MAUTO .PRIMFN GETCH 36
5. 103 MAUTO .PRIMFN PUTCH 37
5. 104 MAUTO PRIMFN INCH 38
5. 105 MAUTO PRIMFN OPCH 39
5. 106 MAUTO PRIMFN SEGIO 39
5. 107 MAUTO PRIMFN MATCH 40
5. 108 MAUTO PRIMFN DICMATCH 41
5. 109 MAUTO PRIMFN STOP 42
5. 110 MAUTO PRIMFN TIME 44
5. 111 MAUTO PRIMFN OPENF 45
5. 112 MAUTO PRIMFN CLOSEF 47
5. 113 MAUTO PRIMFN TRAP 49
5. 114 MAUTO PRIMFN TRUE 51
5. 115 MAUTO PRIMFN FALSE 52
5. 116 MAUTO PRIMFN ADV 53
5. 117 MAUTO PRIMFN ESTOP 54
5. 118 MAUTO PRIMFN VMDEBUG 58
5. 119 MAUTO PRIMFN EMULATE 80
5. 120 MAUTO .
5. 121 MAUTO . PRIM
6. 1 MAUTO .
6. 2 MAUTO . System dependent primitives
6. 3 MAUTO .
6. 4 MAUTO PRIMFN PDUMP 43
6. 5 MAUTO PRIMFN EXR 48
6. 6 MAUTO .
6. 7 MAUTO . end of system dependent primitives
6. 8 MAUTO .
5. 122 MAUTO PRIORITY 4 PRIMOP MASK 15
5. 123 MAUTO PRIORITY 4 PRIMOP UNION 16
5. 124 MAUTO PRIORITY 4 PRIMOP DIFFER 17
5. 125 MAUTO PRIORITY 4 PRIMOP --> 56
5. 126 MAUTO PRIORITY 4 PRIMOP <-- 57
5. 127 MAUTO PAGE
```

5. 128 MAUTO	PRIORITY 8	PRIMOP	*	11
5. 129 MAUTO	PRIORITY 8	PRIMOP	/	12
5. 130 MAUTO	PRIORITY 10	PRIMOP	+	9
5. 131 MAUTO	PRIORITY 10	PRIMOP	-	10
5. 132 MAUTO	PRIORITY 10	PRIMOP	FROM	14
5. 133 MAUTO	PRIORITY 10	PRIMOP	ADIFF	55
5. 134 MAUTO	.			
5. 135 MAUTO	PRIORITY 12	PRIMOP	EQ	19
5. 136 MAUTO	PRIORITY 12	PRIMOP	NE	20
5. 137 MAUTO	PRIORITY 12	PRIMOP	LT	21
5. 138 MAUTO	PRIORITY 12	PRIMOP	GT	22
5. 139 MAUTO	PRIORITY 12	PRIMOP	LE	23
5. 140 MAUTO	PRIORITY 12	PRIMOP	GE	24
5. 141 MAUTO	.			
5. 142 MAUTO	PRIORITY 14	PRIMOP	NOT	25
5. 143 MAUTO	PRIORITY 16	PRIMOP	AND	26
5. 144 MAUTO	PRIORITY 18	PRIMOP	OR	27
5. 145 MAUTO	PRIORITY 18	PRIMOP	XOR	28
5. 146 MAUTO	PRIORITY 20	PRIMOP	CHOOSE	29
5. 147 MAUTO	.			
5. 148 MAUTO	PRIORITY 22	PRIMOP	->	4
5. 149 MAUTO	PRIORITY 22	PRIMOP	<=>	50
5. 150 MAUTO	.			
5. 151 MAUTO	PRIORITY 22	PRIMOP	YES	30
5. 152 MAUTO	PRIORITY 22	PRIMOP	NO	31
5. 153 MAUTO	PRIORITY 22	PRIMOP	GO	32
5. 154 MAUTO	PRIORITY 22	PRIMOP	DO	33
5. 155 MAUTO	.			
5. 156 MAUTO	PRIORITY 24	EXITT	ENTRY	34
5. 157 MAUTO	PRIORITY 24	EXITT	EXIT	35
5. 158 MAUTO	.			
5. 159 MAUTO	.	syntactic macros		
5. 160 MAUTO	.			
5. 161 MAUTO	MACRO <:	' LAB OUT\$ OUT\$, '		
00332 5. 162 MAUTO	MACRO >:	',OUT\$:FORGET OUT\$ '		
00338 5. 163 MAUTO	MACRO THEN:	',NO'		
00340 5. 164 MAUTO	MACRO ELSE:	'LAB Z ,GO Z,>RENAME Z OUT\$ '		
00348 5. 165 MAUTO	MACRO WHILE:	',LAB LOOP\$:'		
00352 5. 166 MAUTO	MACRO BACK:	'LOOP\$,FORGET LOOP\$'		
00358 5. 167 MAUTO	MACRO START:	'THEN<'		
00361 5. 168 MAUTO	MACRO REPEAT:	'GO BACK>'		
00364 5. 169 MAUTO	MACRO [:	'(GO<WHILE ENTRY, '		
00370 5. 170 MAUTO	MACRO]:	'EXIT>BACK)'		
00374 5. 171 MAUTO	MACRO HDICT:	'<HASHFUNC, DELIM, LAB HTBL HTBL, NMDREC, 1,		
5. 172 MAUTO		- ACTDIC, 0> 63, HTBL: RESERVE 64, FORGET HTBL, '		
00398 5. 173 MAUTO	PAGE			

```

5. 174 MAUTO . Compiler data structures
5. 175 MAUTO . -----
5. 176 MAUTO .
5. 177 MAUTO VAR PLOC . program location counter
5. 178 MAUTO VAR MAXPLOC:0, . max allowed value for PLOC
00399 5. 179 MAUTO MAXDLOC:0, . max allowed value for DLOC
00400 5. 180 MAUTO IAUTO
5. 181 IAUTO FN PASSEM . prog state assembler
5. 182 IAUTO FN DASSEM . data state assembler
5. 183 IAUTO FN CSACT . compile state element
5. 184 IAUTO VAR STATE . compiler state pointer
5. 185 IAUTO VAR PSREC . prog state record
5. 186 IAUTO VAR DSREC . data state record
5. 187 IAUTO VAR PBIAS: 0 . bias of program space in AUTO
5. 188 IAUTO VAR DBASE: 0 . start of data in AUTO
00401 5. 189 IAUTO VAR RLIST: 0 . list of addr for reloc in ENDAUTO
00402 5. 190 IAUTO .
5. 191 IAUTO VAR PSTATE: PSREC
00403 5. 192 IAUTO PSREC : PLOC, . prog state
00405 5. 193 IAUTO PASSEM, CSACT, MAXPLOC,
00408 5. 194 IAUTO .
5. 195 IAUTO VAR DSTATE: DSREC
5. 196 IAUTO DSREC : DLOC, . data state
00410 5. 197 IAUTO DASSEM, CSACT, MAXDLOC,
00413 5. 198 IAUTO .
5. 199 IAUTO . Location counter record
5. 200 IAUTO .
5. 201 IAUTO VAR LOCREC: DLOC:0, . location record
00414 5. 202 IAUTO PLOC:@PROGEND,
00415 5. 203 IAUTO STATE:PSTATE . current state
5. 204 IAUTO VAR LOCLIST:0,3 . list pointer
00417 5. 205 IAUTO .
5. 206 IAUTO . push-down lists
5. 207 IAUTO .
5. 208 IAUTO VAR STATEPDL:0 . state push down list
00418 5. 209 IAUTO VAR OPTLIST:0 . listing options list
00419 5. 210 IAUTO .
5. 211 IAUTO . Shunt stack
5. 212 IAUTO .
5. 213 IAUTO VAR SHUNTST: . shunt stack
00420 5. 214 IAUTO < . list pointer
00421 5. 215 IAUTO 1000, 0, <0 . stakbot's DSHUNT, DSTATUS, DCLASS, DADDR
00424 5. 216 IAUTO VAR STAKBOT: . dummy dictionary record
00425 5. 217 IAUTO 0, 0> . DLINK, DNAME
00427 5. 218 IAUTO 0, 0, 0 . class record
00429 5. 219 IAUTO >0, STAKBOT . finish ref to stakbot on shuntst
00431 5. 220 IAUTO MAUTO
5. 221 MAUTO PAGE

```



```

5. 222 MAUTO . Dictionary list structures
5. 223 MAUTO . -----
5. 224 MAUTO .
5. 225 MAUTO FN HASHFUNC . hashing function
5. 226 MAUTO FN DELIM . iden construction delimiter
5. 227 MAUTO FN ACTDIC . set dictionary active
5. 228 MAUTO VAR NMDREC . nonmatch record
5. 229 MAUTO DICT MAINDIC:HDICT . public compiler dictionary
00505 5. 230 MAUTO DICT INTDIC: HDICT . internal compiler dictionary
00578 5. 231 MAUTO IAUTO
5. 232 IAUTO FN NONMATCH . non-match action
5. 233 IAUTO 0, 0, . DSHUNT, DSTATUS
00580 5. 234 IAUTO <NONMATCH, . link to DCLASS, DADDR
00582 5. 235 IAUTO NMDREC: . non-match dictionary record
5. 236 IAUTO NMDREC, . link
00583 5. 237 IAUTO 0> . name, etc.
00584 5. 238 IAUTO NONMATCH, NONMATCH, 0, . class record
00587 5. 239 IAUTO VAR DIC:0, . addr of currently selected dict
00588 5. 240 IAUTO VAR DICP:0, . dictionary record list pointer
00589 5. 241 IAUTO ICON DICFLEN 9
5. 242 IAUTO ICON DICHsiz 63
5. 243 IAUTO FN @DICHASH: ENTRY 1 FROM DIC EXIT
35753 5. 244 IAUTO FN @DICDLM: ENTRY 2 FROM DIC EXIT
35760 5. 245 IAUTO FN @DICDLST: ENTRY 3 FROM DIC EXIT
35767 5. 246 IAUTO FN @DICNMR: ENTRY 4 FROM DIC EXIT
35774 5. 247 IAUTO FN @DICACCS: ENTRY 5 FROM DIC EXIT
35781 5. 248 IAUTO FN @DICTREE: ENTRY 6 FROM DIC EXIT
35788 5. 249 IAUTO FN @DICTREE: ENTRY 7 FROM DIC EXIT
35795 5. 250 IAUTO FN DICND: ENTRY (1 FROM DICHsiz) FROM DICDLST EXIT
35805 5. 251 IAUTO MACRO DICHASH: ' VAL(@DICHASH)'
00594 5. 252 IAUTO MACRO DICDLM: ' VAL(@DICDLM)'
00599 5. 253 IAUTO MACRO DICDLST: ' VAL(@DICDLST)'
00604 5. 254 IAUTO MACRO DICNMR: ' VAL(@DICNMR)'
00609 5. 255 IAUTO MACRO DICACCS: ' VAL(@DICACCS)'
00614 5. 256 IAUTO MACRO DICTACTR: ' VAL(@DICTACTR)'
00619 5. 257 IAUTO MACRO DICTREE: ' VAL(@DICTREE)'
00624 5. 258 IAUTO .
5. 259 IAUTO . Environment Control
5. 260 IAUTO . -----
5. 261 IAUTO .
5. 262 IAUTO VAR BASELIST . public compiler dict
5. 263 IAUTO VAR ENVSTRT : < . active dict for id insertion
00625 5. 264 IAUTO VAR BASESTRT: < . start of old dict list for AUTO
00626 5. 265 IAUTO VAR ENVLIST : <> > > . dictionary list
00627 5. 266 IAUTO BASELIST, @INTDIC,
00629 5. 267 IAUTO BASELIST: 0, @MAINDIC,
00631 5. 268 IAUTO VAR INSRTLST:<>0, ENVLIST, . list of active dics
00634 5. 269 IAUTO VAR LASTDICP: BASELIST . pointer to last dict for search
5. 270 IAUTO VAR CURRSTRT: 0 . for autocompilation
00635 5. 271 IAUTO FN ENVTOP:ENTRY, VAL(1 FROM ENVLIST)->@DIC, EXIT
35816 00636 5. 272 IAUTO PAGE

```

```

5. 273 IAUTO . Dictionary record access functions (based on CUR)
5. 274 IAUTO . -----
5. 275 IAUTO .
5. 276 IAUTO VAR CUR:0 . current dictionary record
5. 277 IAUTO VAR LAST:0 . last dictionary entry
00637 5. 278 IAUTO FN OLDDIC . re-lookup in old dictionary
5. 279 IAUTO
5. 280 IAUTO ICON DFLEN 5 . Length of fixed portion of dict rec.
00638 5. 281 IAUTO ICON DCURPOS 4 . position of CUR.
5. 282 IAUTO ICON MB (MINUS DCURPOS),
5. 283 IAUTO FN @DSTART: . start of record
5. 284 IAUTO FN @DSHUNT: ENTRY, MB FROM CUR EXIT . shunt factor (PRIORITY)
35823 5. 285 IAUTO FN @DSTATUS: ENTRY, &(MB+1) FROM CUR EXIT . identifier status
35830 5. 286 IAUTO FN @DCLASS: ENTRY, &(MB+2) FROM CUR EXIT . CLASS rec pointer
35837 5. 287 IAUTO FN @DADDR: ENTRY, &(MB+3) FROM CUR EXIT . assigned address
35844 5. 288 IAUTO FN @DLINK: ENTRY, CUR EXIT . next record addr.
35848 5. 289 IAUTO FN @DNAME: ENTRY, 1 FROM CUR EXIT . dictionary name
35855 5. 290 IAUTO FN DLENGTH: ENTRY, DUP(DFLEN+((DNAME+7)-->2)) +MASK 1 EXIT
35873 5. 291 IAUTO
5. 292 IAUTO MACRO DSTART: ' VAL(@DSTART)'
00643 5. 293 IAUTO MACRO DSHUNT: ' VAL(@DSHUNT)'
00648 5. 294 IAUTO MACRO DSTATUS: ' VAL(@DSTATUS)'
00653 5. 295 IAUTO MACRO DCLASS: ' VAL(@DCLASS)'
00658 5. 296 IAUTO MACRO DADDR: ' VAL(@DADDR)'
00662 5. 297 IAUTO MACRO DLINK: ' VAL(@DLINK)'
00666 5. 298 IAUTO MACRO DNAME: ' VAL(@DNAME)'
00670 5. 299 IAUTO FORGET MB
5. 300 IAUTO .
5. 301 IAUTO . Define class record structure
5. 302 IAUTO .
5. 303 IAUTO FN DSACT: ENTRY VAL(DCLASS) EXIT . syntax action
35878 5. 304 IAUTO FN DGEN: ENTRY VAL(1 FROM DCLASS) EXIT . generative action
35886 5. 305 IAUTO FN DSET: ENTRY VAL(2 FROM DCLASS) EXIT . setting action
35894 5. 306 IAUTO .
5. 307 IAUTO . Define state record structure
5. 308 IAUTO .
5. 309 IAUTO FN @LCTR: ENTRY VAL(VAL(STATE)) EXIT . address of current
35900 5. 310 IAUTO FN LCTR: ENTRY VAL(@LCTR) EXIT . location counter
35904 5. 311 IAUTO FN SDASSEM: ENTRY VAL(1 FROM VAL(STATE)) . pointer to state
35911 5. 312 IAUTO EXIT . dependent assembler
35913 5. 313 IAUTO FN STATEACT: ENTRY VAL(2 FROM VAL(STATE)) . state dependent
35920 5. 314 IAUTO EXIT . element processor
35922 5. 315 IAUTO .
5. 316 IAUTO PAGE

```

```

5. 317 IAUTO . Compiler syntax actions
5. 318 IAUTO . -----
5. 319 IAUTO .
5. 320 IAUTO . These routines are referenced when an identifier is
5. 321 IAUTO . matched in the input stream, with address of dic
5. 322 IAUTO . record in CUR.
5. 323 IAUTO MAUTO
5. 324 MAUTO DIR , . comma directive
5. 325 MAUTO FN COMPILE . compile function
5. 326 MAUTO IAUTO
5. 327 IAUTO FN SWINPUT . switch input streams
5. 328 IAUTO FN STEOS . end-of-string function
5. 329 IAUTO FN STEOP . end-of-process function
5. 330 IAUTO FN PROCESS . process one string
5. 331 IAUTO .
5. 332 IAUTO . Shunt routines
5. 333 IAUTO .
5. 334 IAUTO FN SHUNTSB:ENTRY
35923 5. 335 IAUTO <=>CUR, WHILE VAL(1 FROM SHUNTST)->@CUR, DUP GE DSHUNT START
35942 5. 336 IAUTO POPUP(@SHUNTST) DO DGEN
35945 5. 337 IAUTO REPEAT LOSE, ->@CUR
35951 5. 338 IAUTO EXIT
35955 5. 339 IAUTO .
5. 340 IAUTO SHUNT:ENTRY SHUNTSB(DSHUNT) PUSH(DCUR, @SHUNTST) EXIT
35965 5. 341 IAUTO .
5. 342 IAUTO SHUNTIM:ENTRY SHUNTSB(DSHUNT), DO DGEN, EXIT
35972 5. 343 IAUTO .
5. 344 IAUTO . Directive action
5. 345 IAUTO .
5. 346 IAUTO DIRACT:ENTRY OLDIC,
35974 5. 347 IAUTO (DSTATUS MASK 256) NE 0 THEN< OUTST('UNSET Directive: ')
35987 00676 5. 348 IAUTO OUTST(@DNAME) OPNL SHUNT EXIT>
35993 5. 349 IAUTO DO DADDR
35994 5. 350 IAUTO EXIT
35997 5. 351 IAUTO .
5. 352 IAUTO . Macro action
5. 353 IAUTO .
5. 354 IAUTO MACACT:ENTRY SWINPUT(DADDR, @STEOS) EXIT
36004 5. 355 IAUTO .
5. 356 IAUTO . Activate a dictionary
5. 357 IAUTO .
5. 358 IAUTO DICACT:ENTRY ACTDIC(DADDR) EXIT
36009 5. 359 IAUTO .
5. 360 IAUTO . Set prog state and set data state
5. 361 IAUTO .
5. 362 IAUTO SETPROG:ENTRY REF ,, @PSTATE->@STATE EXIT
36017 5. 363 IAUTO SETDATA:ENTRY REF ,, @DSTATE->@STATE EXIT
36025 5. 364 IAUTO PAGE

```

```

5. 365 IAUTO . Compiler generative actions
5. 366 IAUTO . -----
5. 367 IAUTO .
5. 368 IAUTO . The generative actions are invoked when a dictionary
5. 369 IAUTO . record is shunted off the shunt stack.
5. 370 IAUTO .
5. 371 IAUTO FN IDREF . gen forward or direct id ref
5. 372 IAUTO FN ASSEM . assemble generated code
5. 373 IAUTO FN STORE1 . store 1 word
5. 374 IAUTO FN DELDREC . delete dictionary rec
5. 375 IAUTO .
5. 376 IAUTO GENCON:
5. 377 IAUTO LABGEN: ENTRY ASSEM(@GET) EXIT
36030 5. 378 IAUTO VARGEN: ENTRY ASSEM(@GETV) EXIT
36035 5. 379 IAUTO FNGEN: ENTRY ASSEM(0) EXIT
36040 5. 380 IAUTO PRIMGEN: ENTRY STORE1(DADDR) EXIT
36045 5. 381 IAUTO STRGEN:
5. 382 IAUTO INTGEN: ENTRY LABGEN DELDREC EXIT
36049 5. 383 IAUTO ADRGEN: ENTRY DADDR DELDREC, ->@CUR, LABGEN EXIT
36058 5. 384 IAUTO DICGEN: ENTRY ASSEM(@GET), IDREF->@DIC, STORE1(DICACTR) EXIT
36070 5. 385 IAUTO .
5. 386 IAUTO . Generate forward or direct id reference
5. 387 IAUTO .
5. 388 IAUTO IDREF: ENTRY
36071 5. 389 IAUTO (ONAUTO AND ((DSTATUS MASK 512) EQ 0)) THEN<
36084 5. 390 IAUTO PUSH(D(LCTR, @RLIST)> DADDR,
36090 5. 391 IAUTO (DSTATUS MASK 256) NE 0 THEN <LCTR->@DADDR> . label not set
36104 5. 392 IAUTO EXIT
36105 5. 393 IAUTO .
5. 394 IAUTO . Assembly routines
5. 395 IAUTO .
5. 396 IAUTO ASSEM: ENTRY DO SDASSEM EXIT
36109 5. 397 IAUTO DASSEM: ENTRY LOSE STORE1(IDREF) EXIT
36114 5. 398 IAUTO PASSEM: ENTRY DUP NE 0 THEN <STORE1(DUP)> LOSE STORE1(IDREF) EXIT
36128 5. 399 IAUTO .
5. 400 IAUTO FN OUTMEMR: ENTRY
36129 5. 401 IAUTO OUTST('Next location ') OPINTD
36132 00681 5. 402 IAUTO OUTST(' exceeds max. avail. space.'), OPNL,
36137 00689 5. 403 IAUTO EXIT
36138 5. 404 IAUTO .
5. 405 IAUTO . Store an item at LCTR if space available
5. 406 IAUTO .
5. 407 IAUTO STORE1: ENTRY
36139 5. 408 IAUTO DUP(LCTR) GT VAL(VAL(3 FROM VAL(STATE)))
36148 5. 409 IAUTO THEN <OUTMEMR, ESTOP> ->, 1 FROM LCTR->@LCTR
36160 5. 410 IAUTO EXIT
36163 5. 411 IAUTO PAGE

```

```

5. 412 IAUTO . Compiler directives
5. 413 IAUTO . -----
5. 414 IAUTO .
5. 415 IAUTO . The set of directives following comprises 3 groups.
5. 416 IAUTO . These are: 1) syntactic directives defining language properties,
5. 417 IAUTO . 2) utility or general purpose directives,
5. 418 IAUTO . 3) dictionary maintenance directives.
5. 419 IAUTO .
5. 420 IAUTO FN IDLOAD . load id for setting
5. 421 IAUTO FN IDSET . identifier set routine
5. 422 IAUTO FN INTRO . general introduction action
5. 423 IAUTO FN REFCON . generate and reference const
5. 424 IAUTO FN NEWDREC . get new dictionary record
5. 425 IAUTO FN RELDIC . release dictionary records
5. 426 IAUTO FN SELDIC . select dictionary function
5. 427 IAUTO FN ADDTODIC . add record to dictionary
5. 428 IAUTO FN IDFIND . locate iden starting at dict ptr
5. 429 IAUTO FN IDFINDS . locate iden in specific dict
5. 430 IAUTO FN IDOFADDR . locate name from address value
5. 431 IAUTO FN ACTIVATE . syntax action on dic rec
5. 432 IAUTO FN NEXTELT . locate dic rec for next elt
5. 433 IAUTO FN RDIDEN . build identifier from input
5. 434 IAUTO FN CKDLOC . check for DLOC GE MAXDLOC
5. 435 IAUTO VAR BLOCKL:0 . current block level
5. 436 IAUTO VAR BLKDIC:0 . list of dicts created by BLOCK
00690 5. 437 IAUTO VAR CPRIOR:0 . operator precedence value for setting
00691 5. 438 IAUTO VAR COMPS:0,3 . compiler string input list
00693 5. 439 IAUTO VAR COLSAV:0 . CAP index save
00694 5. 440 IAUTO VAR STRLIST:0 . string list in auto compile
00695 5. 441 IAUTO MAUTO
5. 442 MAUTO DIR NOW . immediate compile and execute
5. 443 MAUTO DIR ! . close of NOW
5. 444 MAUTO DIR PRIORITY . set shunt factor for operators
5. 445 MAUTO FN IPAR . immediate param evaluation
5. 446 MAUTO FN READINP . main end of string action.
5. 447 MAUTO DIR BLOCK . push new dictionary
5. 448 MAUTO DIR ENDBLOCK . pop current dictionary
5. 449 MAUTO FN SAVBLOCK . as ENDBLOCK but save current dict
5. 450 MAUTO FN SETBLOCK . as BLOCK but restore named dict
5. 451 MAUTO FN INITDIC . initialize a dict
5. 452 MAUTO FN PUSHNDIC . create and push new dict
5. 453 MAUTO FN PUSHODIC . push saved dict
5. 454 MAUTO FN POPUPDIC . pop top dict
5. 455 MAUTO FN POPDIC . pop and release top dict
5. 456 MAUTO FN LASTDIC . set last dict for search
5. 457 MAUTO FN SETDIC . set dict access control
5. 458 MAUTO DIR \ . transient setting of active dict
5. 459 MAUTO DIR % . dict for next lookup
5. 460 MAUTO PAGE

```

```

36175 00696 5. 461 MAUTO CKDLOC:ENTRY GE MAXDLOC THEN<OUTMEMR(MAXDLOC), ESTOP> EXIT
5. 462 MAUTO FORGET OUTMEMR
5. 463 MAUTO .
5. 464 MAUTO . REF directive. 'de-activates' and permits reference to a
5. 465 MAUTO . directive without causing directive to be obeyed
5. 466 MAUTO .
36179 5. 467 MAUTO DIR REF:ENTRY NEXTELT SHUNT EXIT . locate iden and shunt it
5. 468 MAUTO .
5. 469 MAUTO . @ directive. generates an address constant
5. 470 MAUTO .
5. 471 MAUTO DIR @:ENTRY NEXTELT REFCON(0, @ADRCON, CUR) EXIT
36189 5. 472 MAUTO .
5. 473 MAUTO . # directive. generates literal constant from character
5. 474 MAUTO .
5. 475 MAUTO DIR #:ENTRY REFCON(512, @INTCON, NEXTCH) EXIT
36197 5. 476 MAUTO .
5. 477 MAUTO . MINUS directive. compiles negative integer constant
5. 478 MAUTO .
5. 479 MAUTO DIR MINUS:ENTRY
36198 5. 480 MAUTO BLANKS, 512, @INTCON, PUSH(D(@STAKBOT, @SHUNTST),
36208 5. 481 MAUTO NEG(IPAR), REF , , POPUP(@SHUNTST), REFCON(,,)
36214 5. 482 MAUTO EXIT
36216 5. 483 MAUTO .
5. 484 MAUTO . $ directive. Compiles HEX constant.
5. 485 MAUTO .
5. 486 MAUTO DIR $:ENTRY REFCON(512, @INTCON, INHEX) EXIT
36224 5. 487 MAUTO .
5. 488 MAUTO . ' directive. generates literal string
5. 489 MAUTO .
5. 490 MAUTO DIR ':ENTRY
36225 5. 491 MAUTO #,->@ESCHAR,
36230 5. 492 MAUTO STATE NE @DSTATE THEN <REFCON(0, @STRCON, DLOC)>
36245 5. 493 MAUTO ONAUTO THEN< PUSH(DLOC, @STRLIST)>
36254 5. 494 MAUTO INSTRING(@DLOC, [CHAR EQ 13 THEN <DO EOSTRACT, BLANKS,
36271 5. 495 MAUTO ADVCH> CHAR NE #'] ),
36280 5. 496 MAUTO CKDLOC(DLOC), ADVCH 0->@ESCHAR,
36289 5. 497 MAUTO EXIT
36290 5. 498 MAUTO .
5. 499 MAUTO . , directive. clears shunt stack up to (
5. 500 MAUTO .
5. 501 MAUTO REF ,:ENTRY SHUNTSB(99) EXIT
36295 5. 502 MAUTO PAGE

```

```

5. 503 MAUTO .   Open and close parenthesis directives
5. 504 MAUTO .
5. 505 MAUTO VAR LNKS:0,
00697 5. 506 MAUTO DIR (:ENTRY
36296 5. 507 MAUTO     PUSH(DGLKP, @LNKS), PUSH(D@STAKBOT, @SHUNTST)
36304 5. 508 MAUTO     WHILE ACTIVATE(NEXTLT) GO BACK,
36310 5. 509 MAUTO .
5. 510 MAUTO DIR ):ENTRY
36311 5. 511 MAUTO     VAL(DUP(@LNKS)) EQ 0 THEN<OUTST('Unmatched ) at column')
36323 00704 5. 512 MAUTO     OPINT(CURCOL) OPNL LOSE EXIT>
36330 5. 513 MAUTO     REF , , POPUP(@SHUNTST) SLKP(POPPEDUP())
36335 5. 514 MAUTO     EXIT
36337 5. 515 MAUTO FORGET LNKS
5. 516 MAUTO .
5. 517 MAUTO .   ;CR directive. carriage return does nothing
5. 518 MAUTO .   . directive forces end-of-string action
5. 519 MAUTO .
5. 520 MAUTO DIR ;CR:ENTRY EXIT
36339 5. 521 MAUTO DIR .:ENTRY DO EOSTRCT EXIT . end-of-string action
36344 5. 522 MAUTO .
5. 523 MAUTO .   : directive. sets an identifier
5. 524 MAUTO .
5. 525 MAUTO DIR ;::ENTRY
36345 5. 526 MAUTO     IDLOAD . pick up identifier dic rec
5. 527 MAUTO     DO DSET, . perform setting action
36348 5. 528 MAUTO     IDSET(LCTR) . set with location counter
36349 5. 529 MAUTO     EXIT
36351 5. 530 MAUTO .
5. 531 MAUTO .   PROG sets PROG state, DATA sets DATA state.
5. 532 MAUTO .
5. 533 MAUTO DIR PROG EQV @SETPROG
5. 534 MAUTO DIR DATA EQV @SETDATA
5. 535 MAUTO .
5. 536 MAUTO .   RESERVE directive. reserves data area
5. 537 MAUTO .
5. 538 MAUTO DIR RESERVE:ENTRY
36352 5. 539 MAUTO     REF, , IPAR WHILE DUP(-1) GE 0 START
36364 5. 540 MAUTO     0->DLOC, 1 FROM DLOC->@DLOC REPEAT LOSE, CKDLOC(DLOC)
36383 5. 541 MAUTO     EXIT
36385 5. 542 MAUTO .
5. 543 MAUTO .   EQV directive. inertly sets an identifier
5. 544 MAUTO .
5. 545 MAUTO DIR EQV:ENTRY IDLOAD IDSET(IPAR) EXIT
36390 5. 546 MAUTO PAGE

```

```

5. 547 MAUTO . NOW directive. opens body of immediately executable code.
5. 548 MAUTO .
5. 549 MAUTO BLOCK
5. 550 1 VAR RSAV:0,
00705 5. 551 1 REF NOW:ENTRY
36391 5. 552 1 PUSH(D(RLIST, @RSAV), REF , , PUSH(DREC(@LOCREC, @LOCLIST),
36402 5. 553 1 REF BLOCK, SETPROG, COMPILE('ENTRY, '))
36406 00708 5. 554 1 EXIT
36408 5. 555 1 .
5. 556 1 . ! directive. closes and executes the above
5. 557 1 .
5. 558 1 REF !:ENTRY
36409 5. 559 1 COMPILE('EXIT')
36411 00710 5. 560 1 DO VAL(1 FROM VAL(1 FROM LOCLIST)), . DO code at PLOC.
36423 5. 561 1 POPUPREC(@LOCREC, @LOCLIST),
36428 5. 562 1 POPPEDUP(@RSAV),
36431 5. 563 1 WHILE DUP NE RLIST START POPUP(@RLIST) REPEAT LOSE,
36445 5. 564 1 REF ENDBLOCK,
36446 5. 565 1 EXIT
36447 5. 566 1 .
5. 567 1 . Set current priority
5. 568 1 .
5. 569 1 REF PRIORITY:ENTRY
36448 5. 570 1 BLANKS DIGIT THEN< ININT ELSE NEXTELT, DSHUNT> ->@CPRIOR,
36463 5. 571 1 EXIT
36464 5. 572 1 .
5. 573 1 FN NOBLMSG:ENTRY OUTST('No BLOCK active. '), OPNL, EXIT
36470 00715 5. 574 1 VAR PREV:0, VAR DICADDR:0
00716 5. 575 1 FN UNLINKD:ENTRY
36471 00717 5. 576 1 ->@DICADDR, DUMP(ENVLIST), @ENVLIST->@PREV
36479 5. 577 1 WHILE (TEMP NE 0) AND (VAL(1 FROM TEMP) NE DICADDR)
36496 5. 578 1 START TEMP->@PREV, DUMP(VAL(TEMP)) REPEAT
36512 5. 579 1 TEMP NE 0 THEN<VAL(TEMP)->PREV>
36526 5. 580 1 EXIT
36527 5. 581 1 .
5. 582 1 . BLOCK directive. pushes new dictionary and makes it active.
5. 583 1 .
5. 584 1 REF BLOCK:ENTRY
36528 5. 585 1 ADV(@BLOCKL), PUSH(D(ENVSTRT, @INSRTLST),
36536 5. 586 1 PUSH(D(PUSHNDIC, @BLKDIC), ENVLIST->@ENVSTRT,
36545 5. 587 1 EXIT
36546 5. 588 1 PAGE

```



```

5. 589 1 . SETBLOCK function restores previously saved dictionary
5. 590 1 . and acts like another BLOCK invocation
5. 591 1 .
5. 592 1 SETBLOCK: ENTRY
36547 5. 593 1 ADV(@BLOCKL), PUSHD(ENVSTRT, @INSRTLST), VAL,
36556 5. 594 1 PUSHD(PUSHODIC(DUP), @BLKDIC), ENVLIST->@ENVSTRT,
36566 5. 595 1 EXIT
36567 5. 596 1 .
5. 597 1 . ENDBLOCK directive. pops current dictionary
5. 598 1 . SAVBLOCK function does same, but saves dictionary
5. 599 1 .
5. 600 1 REF ENDBLOCK:ENTRY
36568 5. 601 1 BLOCKL GT 0 THEN< BLOCKL-1->@BLOCKL, POPPEDUP(@BLKDIC)
36586 5. 602 1 UNLINKD(DUP) RELDIC() RELREC(DICDLST, DICHSIZ+1)
36597 5. 603 1 RELREC(DIC, DICFLEN) POPPEDUP(@INSRTLST)->@ENVSTRT
36606 5. 604 1 ELSE NOBLMSG>
36613 5. 605 1 EXIT
36614 5. 606 1 .
5. 607 1 SAVBLOCK:ENTRY
36615 5. 608 1 BLOCKL GT 0 THEN< BLOCKL-1->@BLOCKL, POPPEDUP(@BLKDIC)
36633 5. 609 1 UNLINKD(DUP) POPPEDUP(@INSRTLST)->@ENVSTRT,
36642 5. 610 1 <=>, -> ELSE NOBLMSG>
36648 5. 611 1 EXIT
36649 5. 612 1 .
5. 613 1 . IPAR function. evaluates expressions as immediate
5. 614 1 . parameters.
5. 615 1 .
5. 616 1 VAR SAVES:0
5. 617 1 IPAR:ENTRY
36650 00718 5. 618 1 PUSHD(DIC, @SAVES),
36655 5. 619 1 PUSHD(CUR, @SAVES), REF NOW, ACTIVATE(NEXTTEL), REF !,
36664 5. 620 1 POPPEDUP(@SAVES)->@CUR, POPPEDUP(@SAVES)->@DIC,
36676 5. 621 1 EXIT
36677 5. 622 1 ENDBLOCK
5. 623 MAUTO .
5. 624 MAUTO . & directive. evaluates and generates constant from
5. 625 MAUTO . expression.
5. 626 MAUTO .
5. 627 MAUTO DIR &:ENTRY
36678 5. 628 MAUTO PUSHD(@STAKBOT, @SHUNTST),
36683 5. 629 MAUTO REFCON(512, @INTCON, IPAR), REF , , POPUP(@SHUNTST),
36693 5. 630 MAUTO EXIT
36694 5. 631 MAUTO PAGE

```

```
5. 632 MAUTO .   FORGET directive. removes a dictionary record
5. 633 MAUTO .
5. 634 MAUTO DIR FORGET:ENTRY
36695 5. 635 MAUTO   NEXTELT CUR EQ @NMDREC THEN<NONMATCH EXIT>
36706 5. 636 MAUTO   BTREM(CUR, DADDR, @DICTREE),
36712 5. 637 MAUTO   VAL(CUR)->LAST,           . link across deleted record
36718 5. 638 MAUTO   DELDREC
5. 639 MAUTO   EXIT
36720 5. 640 MAUTO .
5. 641 MAUTO .   RENAME directive. renames an identifier
5. 642 MAUTO .
5. 643 MAUTO DIR RENAME:ENTRY
36721 5. 644 MAUTO   NEXTELT CUR EQ @NMDREC THEN<NONMATCH EXIT>
36732 5. 645 MAUTO   BTREM(CUR, DADDR, @DICTREE),
36738 5. 646 MAUTO   VAL(CUR)->LAST,           . link across deleted record
36744 5. 647 MAUTO   DSHUNT, DSTATUS, DCLASS, DADDR, DELDREC,
36753 5. 648 MAUTO   RDIDEN(DICDLM) NEWDREC(), ->@DCLASS, ->@DSTATUS, ->@DSHUNT
36761 5. 649 MAUTO   ADDTODIC, BTINSRT(CUR, DADDR, @DICTREE)
36769 5. 650 MAUTO   EXIT
36771 5. 651 MAUTO PAGE
```

```

5. 652 MAUTO . Dictionary manipulation
5. 653 MAUTO . -----
5. 654 MAUTO .
5. 655 MAUTO VAR LSTINTRO:0 . CUR of last intro
5. 656 MAUTO .
5. 657 MAUTO DIR ICL$:ENTRY SETDIC(1,@INTDIC) EXIT . Compatibility
36778 00719 5. 658 MAUTO DIR RCL$:ENTRY SETDIC(0,@INTDIC) EXIT . only.
36785 5. 659 MAUTO .
5. 660 MAUTO . LOCK, RD-ONLY, and UNLOCK
5. 661 MAUTO .
5. 662 MAUTO DIR LOCK: ENTRY, NEXTELT, SETDIC(0, DADDR) EXIT
36793 5. 663 MAUTO DIR RD-ONLY: ENTRY, NEXTELT, SETDIC(2, DADDR) EXIT
36801 5. 664 MAUTO DIR UNLOCK: ENTRY, NEXTELT, SETDIC(1, DADDR) EXIT
36809 5. 665 MAUTO BLOCK
5. 666 1 .
5. 667 1 . Allocate and initialize a new dictionary
5. 668 1 .
5. 669 1 FN INITDICR:ENTRY
36810 5. 670 1 GETREC(DICFLEN)->@DIC, GETREC(DICHSIZ+1)->@DICDLST,
36824 5. 671 1 @HASHFUNC->@DICHASH, @DELIM->@DICDLM, @NMDREC->@DICNMR,
36836 5. 672 1 1->@DICACCS, @ACTDIC->@DICTACTR,
36844 5. 673 1 ((DICFLEN-1) FROM DIC) ->DIC, DICHSIZ->(1 FROM @DICTACTR),
36862 5. 674 1 EXIT
36863 5. 675 1 .
5. 676 1 . Initialize dictionary: INITDIC(<dic rec>)
5. 677 1 .
5. 678 1 INITDIC:ENTRY
36864 5. 679 1 ->@DIC, DICDLST->@DICP, DICND,
36873 5. 680 1 WHILE DUP GT DICP START
36880 5. 681 1 @NMDREC->DICP, 1 FROM DICP->@DICP,
36893 5. 682 1 REPEAT LOSE, BTINIT(1, 65536, @BINLOC, @DICTREE),
36905 5. 683 1 EXIT
36906 5. 684 1 .
5. 685 1 . Release dictionary records
5. 686 1 .
5. 687 1 RELDIC:ENTRY
36907 5. 688 1 ->@DIC, DICDLST->@DICP, DICND,
36916 5. 689 1 WHILE DUP GT DICP START VAL(DICP)->@CUR,
36929 5. 690 1 WHILE CUR NE @NMDREC START DELDREC, VAL(CUR)->@CUR
36941 5. 691 1 REPEAT 1 FROM DICP->@DICP
36952 5. 692 1 REPEAT LOSE, BTDEL(@DICTREE),
36961 5. 693 1 EXIT
36962 5. 694 1 .
5. 695 1 . Allocate and push new dictionary - return dict address
5. 696 1 .
5. 697 1 PUSHNDIC:ENTRY
36963 5. 698 1 INITDICR, INITDIC(DIC) PUSHODIC(DUP(DIC))
36970 5. 699 1 EXIT
36972 5. 700 1 PAGE

```

```

5. 701 1 . Push dictionary whose address is on stack
5. 702 1 .
5. 703 1 PUSHODIC:ENTRY PUSHDI( , @ENVLIST) EXIT
36977 5. 704 1 .
5. 705 1 . Pop dictionary and release space
5. 706 1 .
5. 707 1 POPDIC:ENTRY
36978 5. 708 1 VAL(1 FROM ENVLIST) EQ @MAINDIC THEN<
36990 5. 709 1 OUTST('BASE dictionary not released. '), OPNL, EXIT>
36995 00728 5. 710 1 POPUPDIC RELDIC() RELREC(DICDLST, DICHSIZ+1)
37004 5. 711 1 RELREC(DIC, DICFLEN)
37009 5. 712 1 EXIT
37011 5. 713 1 .
5. 714 1 . Pop dictionary and return its address
5. 715 1 .
5. 716 1 POPUPDIC:ENTRY POPPEDUP(@ENVLIST) EXIT
37016 5. 717 1 .
5. 718 1 VAR IPNT:0,
00729 5. 719 1 FN DICFIND:ENTRY
37017 5. 720 1 ->@IPNT, DUMP(ENVLIST)
37022 5. 721 1 WHILE (TEMP NE 0) AND (IPNT NE VAL(1 FROM TEMP)) START
37041 5. 722 1 DUMP(VAL(TEMP)),
37045 5. 723 1 REPEAT TEMP
37048 5. 724 1 EXIT
37051 5. 725 1 .
5. 726 1 . ACTDIC(@DICT) sets current insertion dictionary.
5. 727 1 .
5. 728 1 ACTDIC:ENTRY
37052 5. 729 1 DUP, DICFLEN, <=>, FROM, VAL, EQ 0 THEN< INITDIC(DUP),
37066 5. 730 1 PUSHODIC(DUP)>
37068 5. 731 1 DUP(DICFIND()) NE 0 THEN< ->@ENVSTRT ELSE LOSE>
37083 5. 732 1 EXIT
37084 5. 733 1 .
5. 734 1 . LASTDIC(@DICT) sets pointer to last dict to be searched.
5. 735 1 .
5. 736 1 LASTDIC:ENTRY DUP(DICFIND()) NE 0 THEN< ->@LASTDICP ELSE LOSE> EXIT
37101 5. 737 1 .
5. 738 1 . SETDIC(<n>,@DICT) sets dictionary access control
5. 739 1 .
5. 740 1 SETDIC:ENTRY ->@DIC, ->@DICACCS EXIT
37108 5. 741 1 .
5. 742 1 . \dcl does ACTIVATE of next identifier looked up in dcl
5. 743 1 .
5. 744 1 REF \:ENTRY NEXTELT BLANKS, ACTIVATE(IDFINDS(DADDR)) EXIT
37116 5. 745 1 .
5. 746 1 . %dcl causes intro of next id into dcl
5. 747 1 .
5. 748 1 VAR ENVSAVE:0
5. 749 1 REF %:ENTRY ENVSTRT->@ENVSAVE, NEXTELT, DO DICACTR, EXIT
37127 00730 5. 750 1 PAGE

```

```

5. 751 1 . Identifier introduction and setting routines
5. 752 1 . -----
5. 753 1 .
5. 754 1 . IDINTRO: Identifier introduction for CLASS
5. 755 1 . OPINTRO: Identifier introduction for OPER
5. 756 1 . CNINTRO: Identifier introduction for constant (ICON).
5. 757 1 .
5. 758 1 IDINTRO:ENTRY
37128 5. 759 1 INTRO, CUR->@LSTINTRO, ACTIVATE(NEXTELT), 0->@LSTINTRO,
37141 5. 760 1 EXIT
37142 5. 761 1 .
5. 762 1 OPINTRO:ENTRY
37143 5. 763 1 INTRO ONAUTO THEN<ENVLIST, CURRSTRT->@ENVLIST,
37155 5. 764 1 IDSET(IPAR)->@ENVLIST ELSE IDSET(IPAR)>
37165 5. 765 1 DSTATUS UNION 512->@DSTATUS, . indicate constant
37172 5. 766 1 EXIT
37173 5. 767 1 .
5. 768 1 CNINTRO:ENTRY INTRO, IDSET(IPAR), DSTATUS UNION 512->@DSTATUS EXIT
37185 5. 769 1 .
5. 770 1 . Identifier introduction (CUR points to CLASS record for id)
5. 771 1 .
5. 772 1 INTRO:ENTRY
37186 5. 773 1 OLDIC, DADDR, . addr of CLASS rec
37189 5. 774 1 RDIDEN(DICDLM), NEWDREC(0), ->@DCLASS,
37197 5. 775 1 256->@DSTATUS, ADDTDIC, . indicate NOT set
37202 5. 776 1 CPRIOR ->@DSHUNT, 0 ->@CPRIOR,
37211 5. 777 1 ENVSAVE NE 0 THEN<ENVSAVE->@ENVSTRT, 0->@ENVSAVE>
37229 5. 778 1 EXIT
37230 5. 779 1 .
5. 780 1 . Load identifier for setting
5. 781 1 .
5. 782 1 IDLOAD:ENTRY
37231 5. 783 1 DUP(LSTINTRO) EQ 0 THEN< LOSE, POPPEDUP(@SHUNTST)>
37244 5. 784 1 ->@CUR, 0->@LSTINTRO
37249 5. 785 1 EXIT
37253 5. 786 1 .
5. 787 1 . IDSET(<addr or value to be set>) - identifier setting
5. 788 1 .
5. 789 1 VAR SETVAL:0, VAR DICPTR:0
00731 5. 790 1 IDSET:ENTRY
37254 00732 5. 791 1 DSTATUS MASK 256 EQ 0 THEN<BTREM(CUR, DADDR, @DICTREE),
37271 5. 792 1 ->@DADDR, OUTST(@DNAME), OUTST(' is now reset.') OPNL ELSE
37282 00737 5. 793 1 DSTATUS MASK 767 ->@DSTATUS, . clear not set bit (2**8)
37289 5. 794 1 ->@SETVAL, @DADDR, WHILE DUP NE 0 START . set addr in ref list
37300 5. 795 1 DUP->@DICPTR, VAL(DUP), <=>, SETVAL, <=>, ->,
37311 5. 796 1 REPEAT LOSE,
37315 5. 797 1 VAL(1 FROM DICPTR)->@DIC, 0->DICPTR, POPUP(@DICPTR)>
37332 5. 798 1 BTINSRT(CUR, DADDR, @DICTREE),
37338 5. 799 1 EXIT
37339 5. 800 1 PAGE

```

```

5. 801 1 . Identifier matching routines: IDFIND, IDFINDS
5. 802 1 .
5. 803 1 VAR CDIC :0, VAR CCUR :0, VAR SDIC:0
00739 5. 804 1 VAR CLAST:0, VAR CLENGTH:0
00741 5. 805 1 FN IDFINDD:ENTRY
37340 00742 5. 806 1 ->@DIC, DICACCS THEN<
37348 5. 807 1 SELDIC(@CURCHS), DICMATCH(DICP, @NMDREC, @CURCHS),
37358 5. 808 1 DUP->@LAST, VAL->@CUR,
37366 5. 809 1 CUR NE @NMDREC THEN<DUP(CURCOL-COLSAV) GT CLENGTH THEN<
37386 5. 810 1 ->@CLENGTH, CUR->@CCUR, DIC->@SDIC, LAST->@CLAST,
37404 5. 811 1 ELSE LOSE>>
37408 5. 812 1 COLSAV->@CURCOL>
37413 5. 813 1 EXIT
37414 5. 814 1 IDFIND:ENTRY
37415 5. 815 1 ->@CDIC, VAL(1 FROM ENVSTRT)->@SDIC, CURCOL->@COLSAV,
37432 5. 816 1 @NMDREC->@CCUR, 0->@CLENGTH,
37442 5. 817 1 WHILE CDIC NE VAL(LASTDICP) START
37451 5. 818 1 IDFINDD(VAL(1 FROM CDIC)), VAL(CDIC)->@CDIC,
37464 5. 819 1 REPEAT
37467 5. 820 1 CCUR->@CUR, SDIC->@DIC, CLAST->@LAST, COLSAV+CLENGTH->@CURCOL,
37490 5. 821 1 EXIT
37491 5. 822 1 IDFINDS:ENTRY
37492 5. 823 1 CURCOL->@COLSAV, 0->@CLENGTH, IDFINDD(),
37503 5. 824 1 COLSAV+CLENGTH->@CURCOL
37508 5. 825 1 EXIT
37512 5. 826 1 .
5. 827 1 . Relookup current identifier in compiler base dicts
5. 828 1 .
5. 829 1 OLDIC:ENTRY
37513 5. 830 1 ONAUTO THEN<COLSAV->@CURCOL, IDFIND(CURRSTRT)>
37525 5. 831 1 EXIT
37526 5. 832 1 .
5. 833 1 . Add record to the dictionary pointed to by ENVSTRT.
5. 834 1 . Record address is in CUR. Reverse length order.
5. 835 1 .
5. 836 1 VAR CAP:0,0
00743 5. 837 1 ADDTODIC:ENTRY
37527 00744 5. 838 1 VAL(1 FROM ENVSTRT)->@DIC, DADDR EQ 0 THEN< PUSH(DIC, @DADDR)>
37548 5. 839 1 @DNAME ->@CAP, SELDIC(@CAP), DICP->@LAST, DNAME
37561 5. 840 1 WHILE DUP LT VAL(1 FROM VAL(LAST)) . less than current
37569 5. 841 1 START VAL(LAST)->@LAST, . loop through dic
37580 5. 842 1 REPEAT LOSE,
37584 5. 843 1 VAL(LAST)->CUR, CUR->LAST . link record in
37592 5. 844 1 EXIT
37596 5. 845 1 PAGE

```

```

5. 846 1 . IDOFADDR(<address>, <dict>, [function executed on find])
5. 847 1 .
5. 848 1 VAR FACT:0, VAR CDIC:0, VAR SADDR:0, VAR DICNON:0,
00748 5. 849 1 IDOFADDR: ENTRY
37597 5. 850 1 ->@FACT, ->@CDIC, ->@SADDR, GLKP,
37607 5. 851 1 WHILE CDIC NE VAL(LASTDICP) START VAL(1 FROM CDIC)->@DIC,
37625 5. 852 1 FORBTVAL(SADDR, FACT, @DICTREE), VAL(CDIC)->@CDIC,
37637 5. 853 1 REPEAT LOSE,
37641 5. 854 1 EXIT
37642 5. 855 1 .
5. 856 1 . Literal constant reference routine.
5. 857 1 . REFCON (<mode>,<class record address>,<literal constant>)
5. 858 1 . - <mode>: 0 if relocatable, 512 if constant
5. 859 1 .
5. 860 1 REFCON:ENTRY
37643 5. 861 1 0->DLOC, NEWDREC(), ->@DCLASS, ->@DSTATUS, SHUNT,
37654 5. 862 1 EXIT
37655 5. 863 1 .
5. 864 1 . Release a dictionary record
5. 865 1 .
5. 866 1 DELDREC:ENTRY RELREC(@DSTART, DLENGTH) EXIT
37660 5. 867 1 .
5. 868 1 . Acquire new dictionary record
5. 869 1 .
5. 870 1 NEWDREC:ENTRY
37661 5. 871 1 DUP((VAL(DLOC)+7)-->2), +DFLEN, GETREC,
37675 5. 872 1 <=>DCURPOS, FROM->@CUR, MOVE( , DLOC, @DNAME)->@DADDR,
37688 5. 873 1 EXIT
37689 5. 874 1 .
5. 875 1 . Select dictionary based on character
5. 876 1 .
5. 877 1 SELDIC:ENTRY DO DICHASH, FROM DICDLST->@DICP EXIT
37700 5. 878 1 .
5. 879 1 . Hash on character
5. 880 1 .
5. 881 1 HASHFUNC:ENTRY GETCH() MASK DICHSIZ EXIT
37706 5. 882 1 .
5. 883 1 ENDBLOCK
5. 884 MAUTO PAGE

```

```

5. 885 MAUTO . Compiler driver routines
5. 886 MAUTO . -----
5. 887 MAUTO .
5. 888 MAUTO . IAUTO
5. 889 IAUTO FN SNBINIT . Initialization
5. 890 IAUTO FN SNBFCN . Compiler main loop
5. 891 IAUTO FN LISTSRC . List source text
5. 892 IAUTO .
5. 893 IAUTO SYSSTART: ENTRY
37707 5. 894 IAUTO DUP(0)->@SOUNIT, ->@SIUNIT, OUTST('MINT-3 System: Version ')
37718 00755 5. 895 IAUTO OUTST(SYSDAT$) OUTST('. Created on: ') OUTST(SYSDAT$)
37727 00760 5. 896 IAUTO OPNL, OUTST('Copyright D.F. Hendry, 1990, 2004'), OPNL,
37733 00770 5. 897 IAUTO SNBINIT, SNBFCN OUTST('End MINT.'), OPNL,
37739 00774 5. 898 IAUTO STOP
5. 899 IAUTO .
5. 900 IAUTO FN SETBTR: ENTRY
37741 5. 901 IAUTO ->@DIC, DICTREE NE 0 THEN<EXIT>
37753 5. 902 IAUTO DICDLST->@DICP, BTINIT(1, 65535, @BINLOC, @DICTREE),
37766 5. 903 IAUTO WHILE DICP LT DICND START VAL(DICP)->@CUR,
37779 5. 904 IAUTO WHILE CUR NE DICNMR START
37787 5. 905 IAUTO BTINSRT(CUR, DADDR, @DICTREE), VAL(CUR)->@CUR
37796 5. 906 IAUTO REPEAT 1 FROM DICP->@DICP
37807 5. 907 IAUTO REPEAT
37813 5. 908 IAUTO EXIT
37814 5. 909 IAUTO .
5. 910 IAUTO SNBINIT:ENTRY
37815 5. 911 IAUTO MAXDLOC EQ 0 . if new compiler
37817 5. 912 IAUTO THEN <MAXDS$<--7->@MAXDLOC,
37831 5. 913 IAUTO (MAXPS$<--7)+32767->@MAXPLOC,
37842 5. 914 IAUTO IDLOC->@DLOC> . Set initial DLOC value
37847 5. 915 IAUTO CLEARLST(@STATEPDL) . clear state push down list
37849 5. 916 IAUTO CLEARLST(@OPTLIST) . and list options
37852 5. 917 IAUTO 0->@LINENO, . init line no
37858 5. 918 IAUTO WHILE COMPST NE 0 START . clear previous buffer recs
37866 5. 919 IAUTO JOIN(CURCHS, @BUFFER) STEOS
37871 5. 920 IAUTO REPEAT
37875 5. 921 IAUTO @MAINDIC->@DIC, DICTREE EQ 0,
37885 5. 922 IAUTO SETBTR(@MAINDIC), SETBTR(@INTDIC), . Init btrees for compiler dicts
37891 5. 923 IAUTO THEN<COMPILE(' MAINDIC, DICT USERDIC:HDICT, RCL$, USERDIC ')>
37897 00786 5. 924 IAUTO WHILE BLOCKL GT 0 START REF ENDBLOCK REPEAT . in case BLOCK was active
37909 5. 925 IAUTO EXIT
37910 5. 926 IAUTO PAGE

```



```

5. 927 IAUTO . Main driver function
5. 928 IAUTO .
5. 929 IAUTO SNBFCN:ENTRY
37911 5. 930 IAUTO SETPROG SWINPUT(0, @READINP) PROCESS
37917 5. 931 IAUTO EXIT
37919 5. 932 IAUTO .
5. 933 IAUTO . PROCESS routine
5. 934 IAUTO .
5. 935 IAUTO VAR ENTRYPNT:0
5. 936 IAUTO PROCESS:ENTRY
37920 00787 5. 937 IAUTO PUSH(DGLKP, @ENTRYPNT),
37924 5. 938 IAUTO WHILE DO STATEACT GO BACK, . state dependent process
37929 5. 939 IAUTO .
5. 940 IAUTO . Compile state (PROG and DATA) element process
5. 941 IAUTO .
5. 942 IAUTO CSACT:ENTRY ACTIVATE(NEXTTEL) BLANKS EXIT
37934 5. 943 IAUTO .
5. 944 IAUTO . Lookup next element in input stream
5. 945 IAUTO .
5. 946 IAUTO NEXTTEL:ENTRY BLANKS IDFIND(ENVLIST) EXIT
37940 5. 947 IAUTO .
5. 948 IAUTO . Apply syntax action
5. 949 IAUTO .
5. 950 IAUTO ACTIVATE:ENTRY DO DSACT EXIT
37944 5. 951 IAUTO .
5. 952 IAUTO . Non-match syntax action
5. 953 IAUTO .
5. 954 IAUTO VAR OWNCDE:NULL
5. 955 IAUTO NONMATCH:ENTRY
37945 00788 5. 956 IAUTO GLKP, DO OWNCDE, LOSE,
37950 5. 957 IAUTO DIGIT THEN <REFCON(512, @INTCON, ININT), EXIT>
37961 5. 958 IAUTO DUMP(DLOC) INSTRING(@TEMP, @DELIM) . build identifier
37968 5. 959 IAUTO OPTLIST EQ 0 THEN <0->@CURCOL, LISTSRC>
37983 5. 960 IAUTO OUTST('Undefined identifier: ') OUTST(DLOC) OPNL REF .
37990 00795 5. 961 IAUTO EXIT
37992 5. 962 IAUTO .
5. 963 IAUTO . Compile a string
5. 964 IAUTO . COMPILE(<address of string>)
5. 965 IAUTO .
5. 966 IAUTO COMPILE:ENTRY
37993 5. 967 IAUTO PUSH(DSTATE, @STATEPDL) SETPROG SWINPUT(, @STEOP)
38001 5. 968 IAUTO PROCESS POPPEDUP(@STATEPDL)->@STATE
38006 5. 969 IAUTO EXIT
38010 5. 970 IAUTO PAGE

```

```
5. 971 IAUTO . Switch input routine
5. 972 IAUTO . SWINPUT(<string address>,<end-of-string action>)
5. 973 IAUTO .
5. 974 IAUTO SWINPUT:ENTRY
38011 5. 975 IAUTO PUSHDREC(@CURCHS, @COMPST) . push current string table
38015 5. 976 IAUTO ->@EOSTRACT ->@CURCHS, 0->@CURCOL, . new eos action
38027 5. 977 IAUTO EXIT
38028 5. 978 IAUTO PAGE
```



```

5. 979 IAUTO . Standard end-of-string action
5. 980 IAUTO .
5. 981 IAUTO STEOS:ENTRY POPUPREC(@CURCHS,@COMPST) EXIT
38035 5. 982 IAUTO .
5. 983 IAUTO . Standard end-of-process action
5. 984 IAUTO .
5. 985 IAUTO STEOP:ENTRY STEOS SLKP(POPPEDUP(@ENTRYPNT)) EXIT
38042 5. 986 IAUTO .
5. 987 IAUTO . Compiler string input routines
5. 988 IAUTO . -----
5. 989 IAUTO .
5. 990 IAUTO . Read input image
5. 991 IAUTO .
5. 992 IAUTO READINP:ENTRY
38043 5. 993 IAUTO READ, ADV(@LINENO)
38046 5. 994 IAUTO VAL(CURCHS) EQ 0 THEN<STEOP, EXIT>
38058 5. 995 IAUTO FOREACH(@OPTLIST, [DO]) . process listing options
38068 5. 996 IAUTO EXIT
38070 5. 997 IAUTO BLOCK
5. 998 1 .
5. 999 1 . General identifier read routine for dictionary
5.1000 1 . introduction.
5.1001 1 .
5.1002 1 VAR DELFUNC:0
5.1003 1 RDIDEN:ENTRY
38071 00796 5.1004 1 ->@DELFUNC, DUMP(DLOC), #;->@ESCHAR, BLANKS
38082 5.1005 1 INSTRING(@TEMP, DELFUNC), 0->@ESCHAR,
38093 5.1006 1 EXIT
38094 5.1007 1 .
5.1008 1 . Identifier delimiter routine
5.1009 1 .
5.1010 1 DELIM:ENTRY
38095 5.1011 1 DUP(CHAR),DUP NE # , AND (<=>, NE #:), AND (<=>, NE 13),
38111 5.1012 1 EXIT
38112 5.1013 1 .
5.1014 1 ENDBLOCK
5.1015 IAUTO PAGE

```

```

5.1016 IAUTO . Compiler listing output routines
5.1017 IAUTO . -----
5.1018 IAUTO .
5.1019 IAUTO VAR LASTPC:0 . controls PLOC listing
5.1020 IAUTO VAR PGNUM:0 . current page number
00797 5.1021 IAUTO VAR PGTEXT:0 . address of page heading text
00798 5.1022 IAUTO MAUTO
5.1023 MAUTO DIR TITLE . page heading directive
5.1024 MAUTO DIR D$ . decompile directive
5.1025 MAUTO DIR LIST .
5.1026 MAUTO DIR NOLIST .
5.1027 MAUTO DIR LOCS .
5.1028 MAUTO DIR LCODE .
5.1029 MAUTO FN DECOMP . print object text
5.1030 MAUTO FN FTITLE . function called by TITLE
5.1031 MAUTO FN LOOKDF . looks up and displays first id
5.1032 MAUTO FN LOOKD . looks up and displays all ids
5.1033 MAUTO BLOCK
5.1034 1 FN LISTLOC . list location counters
5.1035 1 FN PRNTCODE . list VM code
5.1036 1 FN EJECT . form feed and list header
5.1037 1 FN PRINTLOC . conditionally list loc counters
5.1038 1 .
5.1039 1 . LIST source option
5.1040 1 .
5.1041 1 REF LIST:ENTRY
38113 00799 5.1042 1 OPTLIST EQ 0 THEN<PUSHD(@LISTSRC, @OPTLIST)>
38126 5.1043 1 EXIT
38127 5.1044 1 .
5.1045 1 . List location counters option
5.1046 1 .
5.1047 1 REF LOCS:ENTRY
38128 5.1048 1 OPTLIST NE 0 THEN<PUSHD(@LISTLOC, @OPTLIST)>
38141 5.1049 1 EXIT
38142 5.1050 1 .
5.1051 1 . List VM object code.
5.1052 1 .
5.1053 1 REF LCODE:ENTRY
38143 5.1054 1 OPTLIST NE 0 THEN<PUSHD(@PRNTCODE, @OPTLIST)>
38156 5.1055 1 EXIT
38157 5.1056 1 PAGE

```

```

5.1057 1 . Listing off option
5.1058 1 .
5.1059 1 REF NOLIST:ENTRY CLEARLST(@OPTLIST) 0->@LASTPC EXIT
38167 5.1060 1 .
5.1061 1 . TITLE directive enables page header printing
5.1062 1 . TITLE(<lines per page>,<1st page nr>,<addr of header>)
5.1063 1 .
5.1064 1 REF TITLE:ENTRY FTITLE(IPAR) EXIT
38171 5.1065 1 .
5.1066 1 . Function to record page title
5.1067 1 .
5.1068 1 FTITLE:ENTRY
38172 5.1069 1 PGTEXT NE 0 THEN<RELREC(PGTEXT, ((VAL(PGTEXT)+7)-->2)), 0->@PGTEXT>,
38197 5.1070 1 DUP NE 0 THEN<GETREC(DUP((VAL(DUP)+7)-->2))->@PGTEXT, MOVE(<=>,PGTEXT)>
38221 5.1071 1 DUP EQ 0 THEN<LOSE, PGNUM->@PGNUM, REF NOLIST,
38235 5.1072 1 DUP EQ 0, CHOOSE(@NOLIST, @LIST), DO, ->@PGLNGTH, REF PAGE, ADV(@RLNO),
38252 5.1073 1 EXIT
38253 5.1074 1 .
5.1075 1 . PAGE directive. ejects a page
5.1076 1 .
5.1077 1 REF PAGE:ENTRY
38254 5.1078 1 OPTLIST NE 0 THEN< EJECT>, 0->@RLNO,
38268 5.1079 1 EXIT
38269 5.1080 1 .
5.1081 1 . check for page eject and print page header
5.1082 1 .
5.1083 1 EJECT:ENTRY
38270 5.1084 1 PGLNGTH EQ 0 THEN< EXIT>, RLNO NE 0 THEN< OPFF>,
38288 5.1085 1 PGTEXT NE 0 THEN<WIDTH, OUTST(PGTEXT),
38301 5.1086 1 OUTST('Page'), SETOPP(# , 4), OPINT(DUP(PGNUM)+1->@PGNUM),
38319 00801 5.1087 1 SETOPP(#0,5), OUTST(' Date: '), OUTST(DATE),
38330 00804 5.1088 1 OUTST(' Compiler Level: '), OUTST(SYSID$), OPCH(SOUNIT,13),
38341 00810 5.1089 1 OPCH(SOUNIT,13), OPCH(SOUNIT,13), . OPNL, OPNL, OPNL,
38351 5.1090 1 SETOPP(#0<=>)>
38355 5.1091 1 0->@RLNO,
38360 5.1092 1 EXIT
38361 5.1093 1 PAGE

```

```

5.1094 1 . list source text routine
5.1095 1 .
5.1096 1 LISTSRC:ENTRY
38362 5.1097 1 WIDTH, OB, OB, SETOPP(# ,2), OPINTD(SIUNIT),
38374 5.1098 1 OUTST(' '), SETOPP(# ,4), OPINTD(LINENO), OB,
38386 00812 5.1099 1 BLOCKL GT 0 THEN< SETOPP(# ,4), OPINTD(BLOCKL),
38402 5.1100 1 OUTST(' ') ELSE
38408 00815 5.1101 1 LOOKD(VAL(1 FROM ENVSTRT))>
38415 5.1102 1 OB, SETOPP(#0<=>), OUTST(CURCHS),
38423 5.1103 1 DUP(RLNO+1)->@RLNO, GE PGLNGTH THEN <EJECT>
38439 5.1104 1 EXIT
38440 5.1105 1 .
5.1106 1 . List object code.
5.1107 1 .
5.1108 1 VAR LTMP:0
5.1109 1 FN LISTCODE:ENTRY
38441 00816 5.1110 1 VAL(DUP) EQ 0 THEN < ->, EXIT>, ->@LTMP,
38454 5.1111 1 DUP NE VAL(LTMP) THEN< DUP, -1,
38466 5.1112 1 VAL(LTMP), <=>, DECOMP, ->LTMP,
38474 5.1113 1 ELSE LOSE>
38478 5.1114 1 EXIT
38479 5.1115 1 .
5.1116 1 . List location counters
5.1117 1 .
5.1118 1 VAR LASTP:0 . previous
5.1119 1 VAR LASTD:0 . values
00817 5.1120 1 LISTLOC:ENTRY
38480 00818 5.1121 1 LASTPC NE 0 THEN <OBW, ELSE
38492 5.1122 1 PRINTLOC(VAL(VAL(PSTATE))-ONAUTO*PBIAS, @LASTP)>
38504 5.1123 1 PRINTLOC(VAL(VAL(DSTATE))-ONAUTO*DBASE, @LASTD),
38516 5.1124 1 EXIT
38517 5.1125 1 .
5.1126 1 PRNTCODE:ENTRY
38518 5.1127 1 LASTPC NE 0 THEN <DUP(RLNO+(PLOC-LASTPC))->@RLNO, GE PGLNGTH
38538 5.1128 1 THEN <EJECT, (PLOC-LASTPC)->@RLNO>>
38553 5.1129 1 LISTCODE(PLOC, @LASTPC)
38557 5.1130 1 EXIT
38559 5.1131 1 .
5.1132 1 . Conditionally print location counter
5.1133 1 .
5.1134 1 PRINTLOC:ENTRY
38560 5.1135 1 DUMP DUP EQ VAL(TEMP)
38564 5.1136 1 THEN <LOSE OBW ELSE OB, OPINTD(DUP) ->TEMP>
38580 5.1137 1 EXIT
38581 5.1138 1 PAGE

```

```
5.1139 1 . Decompile functions
5.1140 1 .
5.1141 1 . These functions display identifier names, given an address:
5.1142 1 .
5.1143 1 . LOOKD(<address>) - displays first name whose address matches.
5.1144 1 . LOOKDF(<address>) - displays each name whose address matches.
5.1145 1 .
5.1146 1 LOOKD:ENTRY
38582 5.1147 1 DICP, <=>, CUR, <=>, DIC, <=>, @NMDREC -> @CUR,
38596 5.1148 1 IDOFADDR( , ENVLIST, [ ->@CUR, LOSE SLKP, EXIT]),
38612 5.1149 1 OUTST(DUP(@DNAME)), VAL-9
38616 5.1150 1 WHILE DUP LT 0 START OB, +1, REPEAT LOSE,
38634 5.1151 1 ->@DIC, ->@CUR, ->@DICP,
38643 5.1152 1 EXIT
38644 5.1153 1 .
5.1154 1 LOOKDF:ENTRY
38645 5.1155 1 DICP, <=>, CUR, <=>, DIC, <=>,
38654 5.1156 1 IDOFADDR( , ENVLIST, [ ->@CUR, OUTST(@DNAME) OUTST(' ', ' ') EXIT]),
38673 00820 5.1157 1 ->@DIC, ->@CUR, ->@DICP,
38682 5.1158 1 EXIT
38683 5.1159 1 .
5.1160 1 . D$ directive - decompile
5.1161 1 . D$(<start>,<end>)
5.1162 1 .
5.1163 1 REF D$:ENTRY IPAR DECMP EXIT
38687 5.1164 1 PAGE
```

```

5.1165 1 . function DECMP prints lines of object code.
5.1166 1 .
5.1167 1 VAR END:0
5.1168 1 DECMP:ENTRY
38688 00821 5.1169 1 ->@END, WIDTH, DUP LT 10 THEN< 10->@WIDTH>, <=>, WHILE DUP LE END START
38713 5.1170 1 OB OPINTD(DUP) OB LOOKD(DUP) OB OB
38720 5.1171 1 OPINT(DUP(VAL(DUP))) OB OB LOOKDF( ) OPNL,
38729 5.1172 1 1, <=>, FROM
38732 5.1173 1 REPEAT LOSE ->@WIDTH,
38740 5.1174 1 EXIT
38741 5.1175 1 ENDBLOCK
3. 13 MAUTO ODIR
5. 1 MAUTO .
5. 2 MAUTO . *** Optional compiler directives
5. 3 MAUTO .
5. 4 MAUTO . ERROR - error interrupt address
5. 5 MAUTO .
5. 6 MAUTO ERROR: ESTOP
5. 7 MAUTO .
5. 8 MAUTO . STOP! directive: normal exit.
5. 9 MAUTO .
5. 10 MAUTO DIR STOP!:ENTRY STOP EXIT
38745 5. 11 MAUTO .
5. 12 MAUTO . SI directive - insert input from named file.
5. 13 MAUTO .
5. 14 MAUTO DIR SI:ENTRY
38746 5. 15 MAUTO PUSHREC(@SIUNIT,@INPST), 0->@LINENO,
38756 5. 16 MAUTO OPENF(1,FNAME)->@SIUNIT,
38763 5. 17 MAUTO EXIT
38764 5. 18 MAUTO .
5. 19 MAUTO . SO directive - send output to named file.
5. 20 MAUTO .
5. 21 MAUTO DIR SO:ENTRY OPENF(2,FNAME)->@SOUNIT, EXIT
38773 5. 22 MAUTO .
5. 23 MAUTO . OBREAK directive - reset output to unit 0.
5. 24 MAUTO .
5. 25 MAUTO DIR OBREAK:ENTRY CLOSEF(SOUNIT), 0->@SOUNIT, EXIT
38783 5. 26 MAUTO .
5. 27 MAUTO . CDUMP directive - PDUMP the current system.
5. 28 MAUTO .
5. 29 MAUTO DIR CDUMP:ENTRY
38784 5. 30 MAUTO PDUMP(IPAR), DUP(IPAR) NE 0 THEN<GO ELSE LOSE>
38799 5. 31 MAUTO EXIT
38800 5. 32 MAUTO PAGE

```



```

5. 33 MAUTO      UNLOCK INTDIC
5. 34 MAUTO      DIR ?
5. 35 MAUTO      DIR OBJ      DIR DPRMPT      DIR NOPRMPT
5. 36 MAUTO      DIR CURDIC DIR PREVDIC      DIR LISTDICS DIR LV$
5. 37 MAUTO      DIR T$
5. 38 MAUTO
5. 39 MAUTO      FN PRTOBJ
5. 40 MAUTO      VAR FULDIC:1 VAR CURDLST:0
00822 5. 41 MAUTO BLOCK
5. 42 1          . List dictionary entries
5. 43 1          .
5. 44 1          FN LVLIST
5. 45 1          FN LVSLIST
5. 46 1          FN DICHEAD: ENTRY
38801 00823 5. 47 1          OUTST('Dictionary: ')
38803 00827 5. 48 1          LOOKD(DIC), OUTST('State: '), DICACCS EQ 1 THEN< OUTST('Open. ')
38820 00833 5. 49 1          ELSE OUTST('Locked. ')>
38827 00836 5. 50 1          EXIT
38828 5. 51 1          VAR OPW:0
5. 52 1          FN OPCNT: ENTRY,
38829 00837 5. 53 1          WIDTH->@OPW, SETOPP(#0, 2), OPINTD(), SETOPP(#0, OPW) EXIT
38846 5. 54 1          VAR CNTR:0, VAR IDCNT:0,
00839 5. 55 1          VAR CDIC:0, VAR DICP:0,
00841 5. 56 1          VAR LSTPRNT:0
5. 57 1          FN SHRTLST: ENTRY
38847 00842 5. 58 1          DUP(CNTR) GE 7 THEN <LOSE, 0, OPNL, ELSE +1, OUTST(' ') >
38869 00844 5. 59 1          EXIT
38870 5. 60 1          FN FULLST: ENTRY
38871 5. 61 1          OPINTD(DADDR), OB, CUR, LOOKD(DCLASS), ->@CUR, WIDTH ->@OPW,
38888 5. 62 1          SETOPP(# , 2), OPINTD(DSHUNT), SETOPP(#0, OPW),
38901 5. 63 1          DUP(CNTR) GE 2 THEN <LOSE, 0, OPNL, ELSE +1, OUTST(' ') >
38923 00846 5. 64 1          EXIT
38924 5. 65 1          FN LVCNTR: ENTRY
38925 5. 66 1          DICDLST->@DICP, DUP(0)->@CNTR->@IDCNT, DICHEAD, OPNL
38940 5. 67 1          WHILE DICP LT DICND START VAL(DICP)->@CUR,
38954 5. 68 1          WHILE CUR NE DICNMR START ADV(@IDCNT)
38964 5. 69 1          DUP(MASK(VAL(1 FROM @DNAME),255)) LT 33 THEN<
38980 5. 70 1          OUTST(';;'), OPCNT, MINUS 6, ELSE LOSE, DUP(@DNAME),
38992 00848 5. 71 1          DUP(VAL-9) GT 0 THEN< CNTR NE 0 THEN<OPNL>,
39012 5. 72 1          2->@CNTR, -8> <=>, OUTST>
39022 5. 73 1          WHILE DUP LT 0 START OB, +1, REPEAT
39036 5. 74 1          LOSE, DO LSTPRNT, ->@CNTR, VAL(CUR)->@CUR,
39049 5. 75 1          REPEAT 1 FROM DICP->@DICP,
39060 5. 76 1          REPEAT CNTR NE 0 THEN<OPNL>
39072 5. 77 1          EXIT
39073 5. 78 1          PAGE

```

```

5. 79 1 LVLIST: ENTRY @FULLST->@LSTPRNT, LVCNTR, EXIT
39081 5. 80 1 LVSLIST: ENTRY @SHRTLST->@LSTPRNT, LVCNTR, EXIT
39089 5. 81 1 .
5. 82 1 REF LV$: ENTRY
39090 5. 83 1 OPNL, OUTST('**Dictionary Entries:'), OPNL
39094 00855 5. 84 1 OUTST('(Identifier - address where set - CLASS - PRIORITY)'),
39098 00869 5. 85 1 OPNL, ENVLIST->@CDIC,
39104 5. 86 1 WHILE CDIC NE VAL(LASTDICP) START VAL(1 FROM CDIC)->@DIC,
39122 5. 87 1 OPNL, FULDIC EQ 1 THEN<LVLIST ELSE LVSLIST>
39136 5. 88 1 OUTST('Item count: ') OPINT(IDCNT) OPNL, VAL(CDIC)->@CDIC,
39149 00873 5. 89 1 REPEAT OPNL
39152 5. 90 1 EXIT
39154 5. 91 1
5. 92 1 REF ?:
5. 93 1 REF CURDIC: ENTRY 0->@CURDLST, VAL(1 FROM ENVLIST)->@DIC,
39169 5. 94 1 FULDIC EQ 1 THEN<LVLIST ELSE LVSLIST>
39182 5. 95 1 EXIT
39183 5. 96 1
5. 97 1 REF PREVDIC: ENTRY CURDLST EQ 0 THEN<ENVLIST->@CURDLST>
39197 5. 98 1 VAL(CURDLST), DUP EQ VAL(LASTDICP) THEN<LOSE, ENVLIST->->@CURDLST,
39214 5. 99 1 VAL(1 FROM CURDLST)->@DIC, LVSLIST,
39224 5. 100 1 EXIT
39225 5. 101 1
5. 102 1 REF LISTDICS: ENTRY
39226 5. 103 1 OPNL, ENVLIST->@CDIC,
39232 5. 104 1 WHILE CDIC NE VAL(LASTDICP) START VAL(1 FROM CDIC)->@DIC,
39250 5. 105 1 0->@IDCNT, DICHEAD, DICDLST->@DICP,
39261 5. 106 1 WHILE DICP LT DICND START VAL(DICP)->@CUR,
39274 5. 107 1 WHILE CUR NE DICNMR START
39282 5. 108 1 ADV(@IDCNT), VAL(CUR)->@CUR,
39291 5. 109 1 REPEAT 1 FROM DICP->@DICP,
39302 5. 110 1 REPEAT OUTST(' Item count: ') OPINT(IDCNT)
39310 00878 5. 111 1 CDIC EQ ENVSTRT THEN<OUTST(' <- Introductions')> OPNL,
39323 00884 5. 112 1 VAL(CDIC)->@CDIC,
39329 5. 113 1 REPEAT
39332 5. 114 1 EXIT
39333 5. 115 1
5. 116 1 FN PRMPT:ENTRY SOUNIT EQ 0 THEN<LOOKD(VAL(1 FROM ENVSTRT)) EXIT
39350 5. 117 1 REF DPRMPT :ENTRY @PRMPT->@GETSTR EXIT
39357 5. 118 1 REF NOPRMPT:ENTRY @ENTRY->@GETSTR EXIT
39364 5. 119 1 VAR CURSAVE:0 VAR CURDAD:0, VAR CURCLS:0, VAR DSTAT:0,
00888 5. 120 1 LAB OEXIT
5. 121 1 REF PRTOBJ:ENTRY SWINPUT( ,@STEOS), REF OBJ, DO EOSTRACT EXIT
39373 5. 122 1 FN ADDROF:ENTRY
39374 5. 123 1 SWINPUT( ,@STEOS), NEXTELT, DADDR, DO EOSTRACT
39380 5. 124 1 EXIT
39384 5. 125 1 PAGE

```

```

5. 126 1 .
5. 127 1 .   T$ directive to provide a basic trace facility
5. 128 1 .
5. 129 1 FN TRACE
5. 130 1 VAR TADDR:0, . traced instruction address
00889 5. 131 1 LAB TSUB
5. 132 1 .
5. 133 1 VAR SLVL: 0, RESERVE 29,
00919 5. 134 1 LAB ELVL  EQV (@SLVL+26)
5. 135 1 VAR FIRST:0, VAR LAST:0,
00921 5. 136 1 VAR CNT:  0, VAR LVL:SLVL,
00923 5. 137 1 VAR CNTT: 0,
00924 5. 138 1 VAR SV1:  0, VAR SV2:0,
00926 5. 139 1 VAR FCN:  0,
00927 5. 140 1 VAR OPCDE:0,
00928 5. 141 1 .
5. 142 1 VAR T1    EQV (1 FROM @WIDTH) . Save CSADD
5. 143 1 VAR T2    EQV (2 FROM @WIDTH) . Save CSCOL
5. 144 1 .
5. 145 1 FN S:ENTRY SETOPP(# ,7),EXIT
39391 5. 146 1 .
5. 147 1 .   TRACE Function for general tracing
5. 148 1 .
5. 149 1 .   call: TRACE (<start>,<end>,<fcn address>)
5. 150 1 .
5. 151 1 .   When <end> is zero, trace is ended.
5. 152 1 .   <start> and <end> are trace range;
5. 153 1 .   <fcn address> is function called to determine if instruction
5. 154 1 .   is to be traced.
5. 155 1 .   Location TADDR contains address of next storage location.
5. 156 1 .   Function must return a "true" or "false" object.
5. 157 1 .
5. 158 1 TRACE: ENTRY,
39392 5. 159 1   ->@FCN,->@LAST,->@FIRST,
39401 5. 160 1   LAST NE 0 THEN <OUTST('Trace mode active'), OPNL,
39413 00934 5. 161 1   OUTST('LVL LABEL'), OUTST(@BLANK6), OB, OUTST('V-ADDR      '),
39423 00942 5. 162 1   OUTST('VAL OP.CODE      STACK:0')
39425 00949 5. 163 1   OPCDE NE 0 THEN<OUTST(' INCR INSTR COUNT')>
39437 00955 5. 164 1   OPNL,
39438 5. 165 1   TRAP (@TSUB),
39441 5. 166 1   ELSE
39444 5. 167 1   TRAP(DUP(0)),OPINT(CNTT),
39451 5. 168 1   OUTST(' Instructions traced. - Trace mode ended'), OPNL,
39455 00966 5. 169 1   DUP->@CNTT, DUP->@CNT, ->@SLVL, @SLVL->@LVL>
39471 5. 170 1   EXIT
39472 5. 171 1 .
5. 172 1 .   Internal trace routine called by interpreter
5. 173 1 .
5. 174 1 PAGE

```

```

5. 175 1 TSUB:
5. 176 1 ->@TADDR, DUP, ->@SV1,
39479 5. 177 1 ADV(@CNT),ADV(@CNTT),
39485 5. 178 1 DUP(VAL(TADDR)) EQ @ENTRY THEN <VAL(LVL)+1->LVL>
39504 5. 179 1 DUP EQ @GLKP THEN <LVL LE @ELVL THEN < @SLVL,
39521 5. 180 1 WHILE DUP NE LVL START
39528 5. 181 1 DUP->@SV2, GLKP EQ VAL(1 FROM SV2) THEN <
39543 5. 182 1 VAL(LVL)->SV2, ELSE +2, REPEAT>
39558 5. 183 1 ->@LVL, GLKP->1 FROM LVL,
39568 5. 184 1 DUP(LVL)+2->@LVL, VAL->LVL,
39581 5. 185 1 ELSE OUTST('GLKP level exceeded'), OPNL>>
39588 00972 5. 186 1 EQ @SLKP THEN<WHILE LVL NE @SLVL START
39602 5. 187 1 LVL-2->@LVL,DUP NE VAL(1 FROM LVL) THEN<REPEAT>>
39624 5. 188 1 ((TADDR ADIFF FIRST) GE 0) AND ((LAST ADIFF TADDR) GE 0)
39640 5. 189 1 THEN <DO FCN,
39647 5. 190 1 THEN < WIDTH,T1,T2, DREM, SUNIT, 0->@SUNIT, 0->@WIDTH,
39670 5. 191 1 SETOPP(# ,2), OPINT(VAL(LVL)), OB, CUR, LOOKD(DUP(TADDR)),
39686 5. 192 1 S, OB, OPINTD, OB, S, OPINTD(DUP(VAL(TADDR))), OB, LOOKD,
39698 5. 193 1 S, OPINT(SV1),
39702 5. 194 1 OPCDE NE 0 THEN<S, OUTST(' #'), SETOPP(# ,6),
39719 00974 5. 195 1 OPINTD(CNT),0->@CNT>
39727 5. 196 1 OPNL,->@CUR,->@SUNIT, ->@DREM, ->@T2,->@T1,->@WIDTH >>
39746 5. 197 1 VAL(TADDR) EQ @EXIT THEN <VAL(LVL)-1->LVL>
39764 5. 198 1 TRAP(TADDR),
39767 5. 199 1 EXIT
39768 5. 200 1 .
5. 201 1 . Standard TRACE directive T$
5. 202 1 .
5. 203 1 REF T$: ENTRY,
39769 5. 204 1 IPAR->@OPCDE,
39773 5. 205 1 TRACE(,, [OPCDE EQ 0 OR VAL(TADDR) EQ OPCDE]),
39793 5. 206 1 EXIT
39794 5. 207 1 PAGE

```

```

5. 208 1 . Directive to display identifier properties.
5. 209 1 .
5. 210 1 REF OBJ:ENTRY NEXTELT, CUR EQ @NMDREC THEN<
39804 5. 211 1 OUTST('Identifier name not found. '), OPNL, REF . , EXIT>
39810 00982 5. 212 1 CUR->@CURSAVE, WIDTH, SETBSE(# ,0,10),
39824 5. 213 1 OUTST('In Dictionary: '), LOOKD(DIC), OPNL,
39831 00987 5. 214 1 OUTST('Name : '), OUTST(@DNAME), OUTST(', CLASS: '), LOOKD(DCLASS),
39842 00994 5. 215 1 OUTST(', Priority: '),
39845 00998 5. 216 1 OPINT(DSHUNT), OUTST(', Status: '), OPINT(DSTATUS), OPNL,
39855 01002 5. 217 1 MASK(DSTATUS, 256) NE 0 THEN<OUTST('Identifier not set. '), OPNL,
39870 01008 5. 218 1 ->@WIDTH, EXIT>
39874 5. 219 1 SETBSE(#0,6,10), OUTST('Value: '),
39884 01011 5. 220 1 DCLASS->@CURCLS, @INTDIC->@DIC, DICACCS->@DSTAT, SETDIC(1,@INTDIC),
39904 5. 221 1 (CURCLS EQ ADDROF('FN')) OR (CURCLS EQ ADDROF('DIR')),
39917 01015 5. 222 1 CURSAVE->@CUR THEN<
39925 5. 223 1 OPNL, DADDR->@CURDAD WHILE VAL(CURDAD) NE 35 START
39940 5. 224 1 DECMP(CURDAD, CURDAD), ADV(@CURDAD) REPEAT
39951 5. 225 1 DECMP(CURDAD, CURDAD), GO OEXIT>
39959 5. 226 1 CURCLS EQ ADDROF('MACRO'), CURSAVE->@CUR THEN< OUTST('; '),
39976 01020 5. 227 1 OUTST(DADDR), OUTST('; '), OPNL, GO OEXIT>
39986 01022 5. 228 1 (CURCLS EQ ADDROF('PRIMOP')) OR (CURCLS EQ ADDROF('PRIMFN'))
39998 01028 5. 229 1 OR (CURCLS EQ ADDROF('ICON')),
40006 01030 5. 230 1 CURSAVE->@CUR THEN< SETBSE(# ,0,10), OPINT(DADDR), OPNL, GO OEXIT>
40028 5. 231 1 (CURCLS EQ ADDROF('VAR')), CURSAVE->@CUR THEN<
40042 01032 5. 232 1 SETBSE(# ,8,10), OPINT(VAL(DADDR)), OPNL, GO OEXIT>
40057 5. 233 1 CURCLS EQ ADDROF('CLASS'), CURSAVE->@CUR THEN< OPNL,
40072 01035 5. 234 1 OUTST('Syntax: '), DECMP(DUP(DADDR)),
40079 01038 5. 235 1 OUTST('Gen : '), DECMP(DUP(1 FROM DADDR))
40088 01041 5. 236 1 OUTST('Set : '), DECMP(DUP(2 FROM DADDR)), GO OEXIT>
40102 01044 5. 237 1 CURCLS EQ ADDROF('DICT'), THEN< CURSAVE->@CUR, ADDROF(@DNAME)->@DIC,
40121 01046 5. 238 1 OPNL, FULDIC EQ 1 THEN<LVLIST ELSE LVSLIST> GO OEXIT>
40138 5. 239 1 OUTST('CLASS record format unknown. DADDR = '), OPINT(DADDR), OPNL,
40145 01057 5. 240 1 OEXIT:
5. 241 1 ->@WIDTH, SETDIC(DSTAT, @INTDIC)
40152 5. 242 1 EXIT
40154 5. 243 1 ENDBLOCK
5. 244 MAUTO LOCK INTDIC
5. 245 MAUTO
3. 14 MAUTO .
3. 15 MAUTO PROGEND: , FORGET PROGEND,
3. 16 MAUTO .
3. 17 MAUTO GENSY
New system generated. Space used (PROG/DATA): 07386 04105
41190 04612 3. 18 USERDIC .
1. 4 USERDIC OBREAK

```