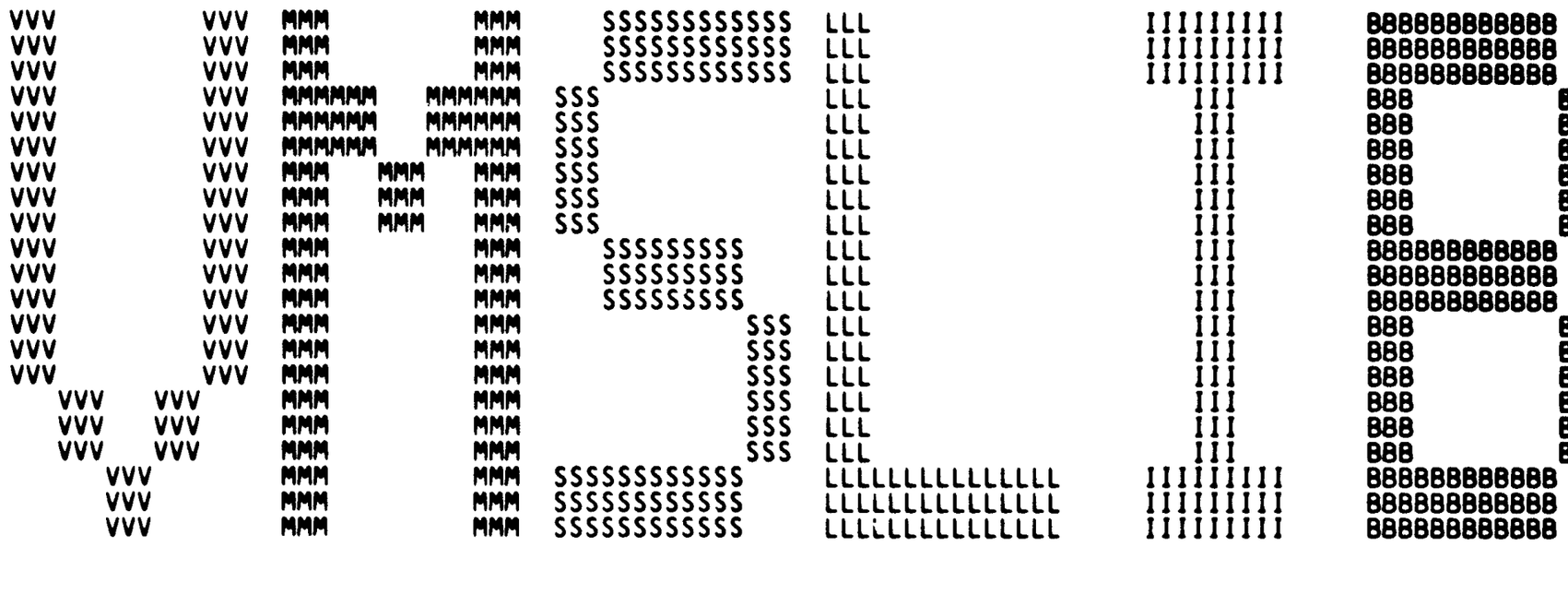
VVV VVV MMM MMM SSSSSSSSSSS LLL IIIIIIIII BBBBBBBBBBB
VVV VVV MMM MMM SSSSSSSSSSS LLL IIIIIIIII BBBBBBBBBBB
VVV VVV MMM MMM SSSSSSSSSSS LLL IIIIIIIII BBBBBBBBBBB
VVV VVV MMMMMM MMMMMM SSS LLL III BBB BBB
VVV VVV MMMMMM MMMMMM SSS LLL III BBB BBB
VVV VVV MMMMMM MMMMMM SSS LLL III BBB BBB
VVV VVV MMM MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM SSSSSSSSS LLL III BBBBBBBBBBB
VVV VVV MMM MMM SSSSSSSSS LLL III BBBBBBBBBBB
VVV VVV MMM MMM SSSSSSSSS LLL III BBBBBBBBBBB
VVV VVV MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM SSSSSSSSSSS LLLLLLLLLLLLLLL IIIIIIIII BBBBBBBBBBB
VVV VVV MMM MMM SSSSSSSSSSS LLLLLLLLLLLLLLL IIIIIIIII BBBBBBBBBBB
VVV MMM MMM SSSSSSSSSSS LLLLLLLLLLLLLLL IIIIIIIII BBBBBBBBBBB

```
TTTTTTTTTT  PPPPPPPP      AAAAAA    MM      MM    AAAAAA      CCCCCCCC
TTTTTTTTTT  PPPPPPPP      AAAAAA    MM      MM    AAAAAA      CCCCCCCC
    TT      PP      PP   AA    AA   MMMM  MMMM   AA    AA    CC
    TT      PP      PP   AA    AA   MMMM  MMMM   AA    AA    CC
    TT      PP      PP   AA    AA   MM  MM  MM   AA    AA    CC
    TT      PP      PP   AA    AA   MM  MM  MM   AA    AA    CC
    TT      PPPPPPPP     AA    AA   MM      MM   AA    AA    CC
    TT      PPPPPPPP     AA    AA   MM      MM   AA    AA    CC
    TT      PP           AAAAAAAAAA   MM      MM   AAAAAAAAAA    CC
    TT      PP           AAAAAAAAAA   MM      MM   AAAAAAAAAA    CC
    TT      PP           AA    AA   MM      MM   AA    AA    CC                ....
    TT      PP           AA    AA   MM      MM   AA    AA    CC                ....
    TT      PP           AA    AA   MM      MM   AA    AA      CCCCCCCC        ....
    TT      PP           AA    AA   MM      MM   AA    AA      CCCCCCCC        ....

RRRRRRRR    EEEEEEEEEE     QQQQQQ
RRRRRRRR    EEEEEEEEEE     QQQQQQ
RR      RR  EE           QQ      QQ
RR      RR  EE           QQ      QQ
RR      RR  EE           QQ      QQ
RR      RR  EE           QQ      QQ
RRRRRRRR    EEEEEEEE     QQ      QQ
RRRRRRRR    EEEEEEEE     QQ      QQ
RR  RR      EE           QQ  QQ  QQ
RR  RR      EE           QQ  QQ  QQ
RR    RR    EE           QQ    QQ
RR    RR    EE           QQ    QQ
RR      RR  EEEEEEEEEE     QQQQ  QQ
RR      RR  EEEEEEEEEE     QQQQ  QQ
```

Macros to generate TPARSE state tables

Version: 'V04-000'


```
!++
```

FACILITY: System Library

ABSTRACT:

These macros are used to generate the state table used with TPARSE.
See the TPARSE module for a complete description.

ENVIRONMENT:

Native mode VAX processor; no operating system facilities are used.

```
!--
```

AUTHOR: Andrew C. Goldstein, CREATION DATE: 30-Aug-1977 16:33

MODIFIED BY:

V03-002 ACG0392 Andrew C. Goldstein, 19-Jan-1984 21:56
Add filespec token type

V03-001 ACG0345 Andrew C. Goldstein, 29-Jul-1983 16:00
Add UIC and identifier tokens; add PSECT argument

V0006 ACG0048 Andrew C. Goldstein, 20-Jun-1979 14:17
Change state table PSECTs to EXE

```
        V0005   ACG0043         Andrew C. Goldstein,  23-May-1979  21:20
        Change state table PSECTs to PIC

        V0004   ACG0024         Andrew C. Goldstein,  27-Feb-1979  16:42
        Fix PSECT names for new RTL standards

    Andrew C. Goldstein, 4-Oct-1977  16:35
    X0002 - Add action routine parameter; allow for keyword uniqueness testing.

    Andrew C. Goldstein, 22-Feb-1978  10:42
    X0003 - State table format changes (BL5)

;*
```

```
!
! Declare the various literals and compile time constants used to generate
! state tables.
!
COMPILETIME
        TPA$K_KEYNUMB   = -1,
        TPA$K_KEYFLAG   = 0,
        TPA$K_SUBEXPR   = 0,
        TPA$K_TYPEVAL   = 0,
        TPA$K_FINAL     = 0;
LITERAL TPA$K_MAXKEY    = 220;

LITERAL
        TPA$M_CODEFLAG  = 256,          ! type is a keyword, special, etc
        TPA$M_EXTRAFLAG = 512,          ! extra flags byte present
        TPA$M_LASTFLAG  = 1024,         ! last transition in state
        TPA$M_EXTFLAG   = 2048,         ! subexpression pointer present
        TPA$M_TRANFLAG  = 4096,         ! explicit target present
        TPA$M_MASKFLAG  = 8192,         ! mask longword present
        TPA$M_ADDRFLAG  = 16384,        ! data address present
        TPA$M_ACTFLAG   = 32768,        ! action routine present
        TPA$M_PARMFLAG  = 65536;        ! action routine parameter present

LITERAL
        TPA$_KEYWORD    = 256,          ! keyword base type
        TPA$_EXIT       = -1,           ! exit parser
        TPA$_FAIL       = -2,           ! exit with failure
        TPA$_FILESPEC   = 490,          ! filespec string
        TPA$_UIC        = 491,          ! UIC string
        TPA$_IDENT      = 492,          ! general identifier string
        TPA$_ANY        = 493,          ! any single character
        TPA$_ALPHA      = 494,          ! any alphabetic character
        TPA$_DIGIT      = 495,          ! any numeric character
        TPA$_STRING     = 496,          ! any alphanumeric string
        TPA$_SYMBOL     = 497,          ! any symbol constituent set string
        TPA$_BLANK      = 498,          ! any string of spaces and tabs
        TPA$_DECIMAL    = 499,          ! decimal number
        TPA$_OCTAL      = 500,          ! octal number
        TPA$_HEX        = 501,          ! hexadecimal number
        TPA$_LAMBDA     = 502,          ! empty string
        TPA$_EOS        = 503,          ! end of string
        TPA$_SUBEXPR    = 504;          ! subexpression

!
! The macro $INIT_STATE is used to initialize the table generator macros.
! It also defines labels for the state table entry point and the keyword table,
! and defines the symbols of all the special token types.
!
MACRO
        $INIT_STATE (START_STATE, KEY_TABLE, PSECT_ARG) =
                %ASSIGN (TPA$K_KEYNUMB, -1)
                %IF %DECLARED (%QUOTE %QUOTE TPA$PSECT_STATE)
                %THEN UNDECLARE %QUOTE %QUOTE TPA$PSECT_STATE; %FI
                %IF %DECLARED (%QUOTE %QUOTE TPA$PSECT_KEY0)
                %THEN UNDECLARE %QUOTE %QUOTE TPA$PSECT_KEY0; %FI
                %IF %DECLARED (%QUOTE %QUOTE TPA$PSECT_KEY1)
```

```
                %THEN UNDECLARE %QUOTE %QUOTE TPA$PSECT_KEY1; %FI
                MACRO   TPA$PSECT_STATE (OWN_GLOBAL) =
                        PSECT   OWN_GLOBAL =
                                %IF %NULL (PSECT_ARG)
                                %THEN _LIB$STATE$
                                %ELSE %NAME (PSECT_ARG, '_STATE')
                                %FI
                                (NOWRITE, SHARE, PIC, EXECUTE, ALIGN (1))
                        %QUOTE %;
                MACRO   TPA$PSECT_KEY0 (OWN_GLOBAL) =
                        PSECT   OWN_GLOBAL =
                                %IF %NULL (PSECT_ARG)
                                %THEN _LIB$KEY0$
                                %ELSE %NAME (PSECT_ARG, '_KEY0')
                                %FI
                                (NOWRITE, SHARE, PIC, EXECUTE, ALIGN (1))
                        %QUOTE %;
                MACRO   TPA$PSECT_KEY1 (OWN_GLOBAL) =
                        PSECT   OWN_GLOBAL =
                                %IF %NULL (PSECT_ARG)
                                %THEN _LIB$KEY1$
                                %ELSE %NAME (PSECT_ARG, '_KEY1')
                                %FI
                                (NOWRITE, SHARE, PIC, EXECUTE, ALIGN (1))
                        %QUOTE %;
                TPA$PSECT_KEY0 (GLOBAL);
                TPA$PSECT_KEY0 (OWN);
                GLOBAL  KEY_TABLE       : VECTOR [0];
                SWITCHES UNAMES;
                %IF %DECLARED (TPA$KEY0) %THEN UNDECLARE TPA$KEY0; %FI
                OWN     TPA$KEY0        : VECTOR [0];
                SWITCHES NOUNAMES;
                TPA$PSECT_STATE (GLOBAL);
                GLOBAL  START_STATE     : VECTOR [0];
                PSECT   GLOBAL = $GLOBAL$;
                PSECT   OWN = $OWN$;
                %;

!
! The $STATE macro is the main level macro. Each call to $STATE generates
! one state in the state table. The first argument, if not null, is a label
! to be applied to this state. Each of the remaining n arguments is a
! transition to another state, consisting of a list of transition elements:
! the token type, the target state, address of the user's action routine,
! a bitmask, and an address in which to store the mask. All of the transition
! elements except the token type are optional.
!
MACRO
        $STATE (STATE_LABEL) =
                TPA$PSECT_STATE (OWN);
                %IF NOT %NULL (STATE_LABEL)
                %THEN OWN STATE_LABEL : ALIGN (0) VECTOR [0];
                %FI
                %ASSIGN (TPA$K_KEYFLAG, 0)
                $STATE_ITEMS (%REMAINING)
                %IF TPA$K_KEYFLAG
                %THEN
```

```
                    TPA$PSECT_KEY1 (OWN);
                    SWITCHES ONAMES;
                    OWN TPA$KEYFILL : VECTOR [1,BYTE] ALIGN (0) INITIAL (BYTE (255));
                    SWITCHES NOUNAMES;
                    UNDECLARE TPA$KEYFILL;
                %FI
                PSECT   OWN = $OWN$;
                %;
!
! The macro $STATE_ITEMS is an iterative macro used to generate the transitions
! in a state.
!
MACRO
        $STATE_ITEMS [ELEMENT] =
                SWITCHES UNAMES;
                %ASSIGN (TPA$K_FINAL, %NULL (%REMAINING))
                TPA$MAKE_TRAN (TPA$K_FINAL, %REMOVE (ELEMENT))
                SWITCHES NOUNAMES;
                %;
!
! The macro TPA$MAKE_TRAN is called to generate each transition entry in a state.
! Its arguments include the final flag (set to 1 for the last transition in
! a state) followed by the elements  of the transition.
!
MACRO
        TPA$MAKE_TRAN (TPA$K_FINAL, TYPE, TARGET, ACTION, MASK, ADDR, PARAM) =
                %ASSIGN (TPA$K_SUBEXPR, 0)
                %IF  TPA$IFSUBEXPR (TYPE)
                %THEN %ASSIGN (TPA$K_TYPEVAL, TPA$_SUBEXPR)
                      %ASSIGN (TPA$K_SUBEXPR, 1)

                %ELSE
                    %IF TPA$IFKEYWORD (TYPE)
                    %THEN
                        %ASSIGN (TPA$K_KEYNUMB, TPA$K_KEYNUMB+1)
                        %IF  TPA$K_KEYNUMB GEQU TPA$K_MAXKEY
                        %THEN %ERROR ('Maximum number of keywords exceeded')
                        %FI
                        %IF  %CHARCOUNT (TYPE) GTRU 65535
                        %THEN %ERROR ('Keyword longer than 65535 characters')
                        %FI
                        TPA$PSECT_KEY1 (OWN);
                        OWN     TPA$KEYSTO : VECTOR [0] ALIGN (0);
                        TPA$PSECT_KEY0 (OWN);
                        OWN     TPA$KEY : WORD INITIAL (TPA$KEYSTO - TPA$KEY0);
                        TPA$PSECT_KEY1 (OWN);
                        OWN     TPA$KEYST : VECTOR [%CHARCOUNT (TPA$KEY_STRING (TYPE)) + 1, BYTE]
                                        ALIGN (0) INITIAL (BYTE (TPA$KEY_STRING (TYPE), 255));
                        UNDECLARE       TPA$KEY, TPA$KEYST, TPA$KEYSTO;
                        TPA$PSECT_STATE (OWN);
                        %ASSIGN (TPA$K_TYPEVAL, TPA$_KEYWORD + TPA$K_KEYNUMB)
                        %ASSIGN (TPA$K_KEYFLAG, 1)
                    %ELSE %ASSIGN (TPA$K_TYPEVAL, TYPE)
                    %FI
                %FI
```

```
OWN     TPA$TYPE        : WORD ALIGN (0) INITIAL (TPA$K_TYPEVAL
                + TPA$K_SUBEXPR*TPA$M_EXTFLAG
                  %IF NOT %NULL (PARAM) %THEN +TPA$M_EXTRAFLAG %FI
                  %IF NOT %NULL (ACTION) %THEN +TPA$M_ACTFLAG %FI
                  %IF NOT %NULL (MASK) %THEN +TPA$M_MASKFLAG
                        %IF %NULL (ADDR)
                        %THEN %ERROR ('Mask address missing') %FI %FI
                  %IF NOT %NULL (ADDR) %THEN +TPA$M_ADDRFLAG %FI
                  %IF NOT %NULL (TARGET) %THEN +TPA$M_TRANFLAG %FI
                + TPA$K_FINAL*TPA$M_LASTFLAG
                );
UNDECLARE       TPA$TYPE;

%IF NOT %NULL (PARAM)
%THEN   OWN     TPA$FLAGS2 : BYTE ALIGN (0) INITIAL (TPA$M_PARMFLAG/65536);
UNDECLARE       TPA$FLAGS2;
%FI

%IF TPA$K_SUBEXPR
%THEN TPA$MAKE_SUB (%REMOVE (TYPE))

%FI

%IF NOT %NULL (PARAM)
%THEN   OWN     TPA$PARAM       : LONG ALIGN (0) INITIAL (PARAM);
UNDECLARE       TPA$PARAM;
%FI

%IF NOT %NULL (ACTION)
%THEN   OWN     TPA$ACTION      : LONG ALIGN (0) INITIAL (ACTION-TPA$ACTION-4);
UNDECLARE       TPA$ACTION;
%FI

%IF NOT %NULL (ADDR)
%THEN   OWN     TPA$ADDR        : LONG ALIGN (0) INITIAL (ADDR-TPA$ADDR-4);
UNDECLARE       TPA$ADDR;
%FI

%IF NOT %NULL (MASK)
%THEN   OWN     TPA$MASK        : LONG ALIGN (0) INITIAL (MASK);
UNDECLARE       TPA$MASK;
%FI

%IF NOT %NULL (TARGET)
%THEN
    %IF NOT %DECLARED (TARGET)
    %THEN FORWARD TARGET : VECTOR [0];
    %FI
    OWN TPA$TARGET      : WORD ALIGN (0) INITIAL
                (%IF %IDENTICAL (TARGET, TPA$_EXIT)
                 OR  %IDENTICAL (TARGET, TPA$_FAIL)
                 %THEN TARGET
                 %ELSE TARGET - TPA$TARGET - 2
                 %FI
                );
    UNDECLARE TPA$TARGET;
```

```
                %FI

                %;
!
! The following macro generates the offset for a subexpression call.
!
MACRO
        TPA$MAKE_SUB (SUBNAME) =
                %IF NOT %DECLARED (SUBNAME)
                %THEN FORWARD SUBNAME : VECTOR [0];
                %FI
                OWN     TPA$SUBEXP : WORD ALIGN(0) INITIAL (SUBNAME - TPA$SUBEXP-2);
                UNDECLARE       TPA$SUBEXP;
                %;
!
! The following macro returns 1 if the argument is a keyword, where a keyword
! is defined as an alphanumeric string of two or more characters.
!
MACRO
        TPA$IFKEYWORD (TYPE) =
                %IDENTICAL (TYPE, %STRING (TYPE))
                AND %CHARCOUNT (%STRING (TYPE)) GTRU 1
                %;
!
! The following macro returns 1 if the argument is a subexpression call,
! identified by being enclosed in parentheses.
!
MACRO
        TPA$IFSUBEXPR (TYPE) =
                NOT %IDENTICAL (TYPE, %REMOVE (TYPE))
                %;
!
! Macros to generate keyword strings. Handles the special string of the form
! 'A*', used to signify a single character keyword.
!
MACRO
        TPA$KEY_STRING (TYPE) =
                %IF %CHARCOUNT (TYPE) EQL 2
                %THEN TPA$ONE_STRING (%EXPLODE (TYPE))
                %ELSE TYPE
                %FI
                %;
MACRO
        TPA$ONE_STRING (A, B) =
                %IF B EQL '*'
                %THEN A
                %ELSE %STRING (A, B)
                %FI
                %.
```

CLIMAC
REQ

TPAMAC
REQ

UTLDEFB
B32

STARMISC
MAR

EQDEFM
MAR

TPAMAR
MAR

CALBYNAME
LIS

SCRPROLOG
REQ

STARLET
SDL

SCRTCB
REQ

OFFSET
MAR

SYITABLE
MAR

EQDEFB
B32

JPITABLE
MAR

DVITABLE
MAR

UTLDEFM
MAR

B32MSG
LIS

SCRTERM
REQ

RSXLBLDF
MAR

C74MSG
LIS