

TTTTTTTTT1	PPPPPPPP	AAAAAA	MM	MM	AAAAAA	CCCCCCCC			
TTTTTTTTT	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	AA	AA	CC
TT	PP	PP	AA	AA	MM	MM	AA	AA	CC
TT	PPPPPPPP	AA	AA	MM	MM	AA	AA	AA	CC
TT	PPPPPPPP	AA	AA	MM	MM	AA	AA	AA	CC
TT	PP		AAAAAAA	MM	MM	AAAAAAA	CC		
TT	PP		AAAAAAA	MM	MM	AAAAAAA	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	CC
TT	PP		AA	AA	MM	MM	AA	AA	CC

....
....
....

RRRRRRRR	EEEEEEEEE	QQQQQ	
RRRRRRRR	EEEEEEEEE	QQQQQ	
RR RR	EE	QQ	QQ
RR RR	EE	QQ	QQ
RR RR	EE	QQ	QQ
RR RR	EE	QQ	QQ
RR RR	EE	QQ	QQ
RR RR	EE	QQ	QQ
RR RR	EE	QQ	QQ
RR RR	EE	QQ	QQ
RR RR	EEEEEEEEE	QQQQ	QQ
RR RR	EEEEEEEEE	QQQQ	QQ

Macros to generate TPARSE state tables

Version: 'V04-000'

* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
* ALL RIGHTS RESERVED.

* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
* TRANSFERRED.

* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
* CORPORATION.

* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.

++
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 PSELECT 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

```
TPASK_KEYNUMB = -1,
TPASK_KEYFLAG = 0,
TPASK_SUBEXPR = 0,
TPASK_TYPEVAL = 0,
TPASK_FINAL = 0;
LITERAL TPASK_MAXKEY = 220;
```

LITERAL

TPASM_CODEFLAG	= 256,	type is a keyword, special, etc
TPASM_EXTRAFLAG	= 512,	extra flags byte present
TPASM_LASTFLAG	= 1024,	last transition in state
TPASM_EXTFLAG	= 2048,	subexpression pointer present
TPASM_TRANFLAG	= 4096,	explicit target present
TPASM_MASKFLAG	= 8192,	mask longword present
TPASM_ADDRFLAG	= 16384,	data address present
TPASM_ACTFLAG	= 32768,	action routine present
TPASM_PARMFLAG	= 65536;	action routine parameter present

LITERAL

TPAS_KEYWORD	= 256,	keyword base type
TPAS_EXIT	= -1,	exit parser
TPAS_FAIL	= -2,	exit with failure
TPAS_FILESPEC	= 490,	filespec string
TPAS_UIC	= 491,	UIC string
TPAS_IDENT	= 492,	general identifier string
TPAS_ANY	= 493,	any single character
TPAS_ALPHA	= 494,	any alphabetic character
TPAS_DIGIT	= 495,	any numeric character
TPAS_STRING	= 496,	any alphanumeric string
TPAS_SYMBOL	= 497,	any symbol constituent set string
TPAS_BLANK	= 498,	any string of spaces and tabs
TPAS_DECIMAL	= 499,	decimal number
TPAS_OCTAL	= 500,	octal number
TPAS_HEX	= 501,	hexadecimal number
TPAS_LAMBDA	= 502,	empty string
TPAS_EOS	= 503,	end of string
TPAS_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 (TPASK_KEYNUMB, -1)
  %IF %DECLARED %QUOTE %QUOTE TPASPSECT STATE)
  %THEN UNDECLARE %QUOTE %QUOTE TPASPSECT STATE; %FI
  %IF %DECLARED (%QUOTE %QUOTE TPASPSECT KEY0)
  %THEN UNDECLARE %QUOTE %QUOTE TPASPSECT KEY0; %FI
  %IF %DECLARED (%QUOTE %QUOTE TPASPSECT KEY1)
```

```

%THEN UNDECLARE %QUOTE %QUOTE TPAS$PSECT_KEY1; %FI
MACRO  TPAS$PSECT STATE (OWN_GLOBAL) =
        PSECT  OWN_GLOBAL =
        %IF-%NULL (PSECT_ARG)
        %THEN  LIB$STATES
        %ELSE %NAME (PSECT_ARG, '_STATE')
        %FI
        (NOWRITE, SHARE, PIC, EXECUTE, ALIGN (1))

MACRO  %QUOTE %:
MACRO  TPAS$PSECT KEY0 (OWN_GLOBAL) =
        PSECT  OWN_GLOBAL =
        %IF-%NULL (PSECT_ARG)
        %THEN  LIB$KEY0S
        %ELSE %NAME (PSECT_ARG, '_KEY0')
        %FI
        (NOWRITE, SHARE, PIC, EXECUTE, ALIGN (1))

MACRO  %QUOTE %:
MACRO  TPAS$PSECT KEY1 (OWN_GLOBAL) =
        PSECT  OWN_GLOBAL =
        %IF-%NULL (PSECT_ARG)
        %THEN  LIB$KEY1S
        %ELSE %NAME (PSECT_ARG, '_KEY1')
        %FI
        (NOWRITE, SHARE, PIC, EXECUTE, ALIGN (1))

        %QUOTE %:
TPAS$PSECT KEY0 (GLOBAL);
TPAS$PSECT KEY0 (OWN);
GLOBAL  KEY_TABLE      : VECTOR [0];
SWITCHES UNAMES;
%IF %DECLARED (TPASKEY0) %THEN UNDECLARE TPASKEY0; %FI
OWN    TPASKEY0       : VECTOR [0];
SWITCHES NOUNAMES;
TPAS$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
$STATE (STATE_LABEL) =
        TPAS$PSECT STATE (OWN);
        %IF NOT %NULL (STATE_LABEL)
        %THEN OWN STATE_LABEL : ALIGN (0) VECTOR [0];
        %FI
        %ASSIGN (TPASK_KEYFLAG, 0)
$STATE_ITEMS (%REMAINING)
        %IF TPASK_KEYFLAG
        %THEN

```

```

TPASPSECT KEY1 (OWN);
SWITCHES ONAMES;
OWN TPA$KEYFILL : VECTOR [1,BYTE] ALIGN (0) INITIAL (BYTE (255));
SWITCHES NOUNAMES;
UNDECLARE TPASKEYFILL;
XFI
PSECT OWN = SOWNS;
%;
```

The macro \$STATE_ITEMS is an iterative macro used to generate the transitions in a state.

MACRO

```

$STATE_ITEMS [ELEMENT] =
SWITCHES UNAMES;
%ASSIGN (TPASK_FINAL, %NULL (%REMAINING))
TPASMAKE_TRAN (TPASK_FINAL, %REMOVE (ELEMENT))
SWITCHES NOUNAMES;
%;
```

The macro TPASMAKE_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

```

TPASMAKE_TRAN (TPASK_FINAL, TYPE, TARGET, ACTION, MASK, ADDR, PARAM) =
%ASSIGN (TPASK_SUBEXPR, 0)
%IF TPASIFSUBEXPR (TYPE)
%THEN %ASSIGN (TPASK_TYPEVAL, TPAS_SUBEXPR)
%ASSIGN (TPASK_SUBEXPR, 1)

%ELSE
%IF TPASIFKEYWORD (TYPE)
%THEN
%ASSIGN (TPASK_KEYNUMB, TPASK_KEYNUMB+1)
%IF TPASK_KEYNUMB GEQU TPASK_MAXKEY
%THEN %ERROR ('Maximum number of keywords exceeded')
XFI
%IF %CHARCOUNT (TYPE) GTRU 65535
%THEN %ERROR ('Keyword longer than 65535 characters')
XFI
TPASPSECT KEY1 (OWN);
OWN TPASKEYST0 : VECTOR [0] ALIGN (0);
TPASPSECT KEY0 (OWN);
OWN TPASKEY : WORD INITIAL (TPASKEYST0 - TPASKEY0);
TPASPSECT KEY1 (OWN);
OWN TPASKEYST : VECTOR [%CHARCOUNT (TPASKEY_STRING (TYPE)) + 1, BYTE]
ALIGN (0) INITIAL (BYTE (TPASKEY_STRING (TYPE), 255));
UNDECLARE TPASKEY, TPASKEYST, TPASKEYST0;
TPASPSECT STATE (OWN);
%ASSIGN (TPASK_TYPEVAL, TPAS_KEYWORD + TPASK_KEYNUMB)
%ASSIGN (TPASK_KEYFLAG, 1)
%ELSE %ASSIGN (TPASK_TYPEVAL, TYPE)
XFI
XFI
```

```

    OWN      TPASTYPE : WORD ALIGN (0) INITIAL (TPASK_TYPEVAL
        + TPASK_SUBEXPR+TPASM_EXTFLAG
        %IF NOT %NULL (PARAM) %THEN +TPASM_EXTRAFLAG %FI
        %IF NOT %NULL (ACTION) %THEN +TPASM_ACTFLAG %FI
        %IF NOT %NULL (MASK) %THEN +TPASM_MASKFLAG
            %IF %NULL (ADDR)
                %THEN %ERROR ('Mask address missing') %FI %FI
            %IF NOT %NULL (ADDR) %THEN +TPASM_ADDRFLAG %FI
            %IF NOT %NULL (TARGET) %THEN +TPASM_TRANFLAG %FI
        + TPASK_FINAL+TPASM_LASTFLAG
    );
UNDECLARE      TPASTYPE;

%IF NOT %NULL (PARAM)
%THEN  OWN      TPASFLAGS2 : BYTE ALIGN (0) INITIAL (TPASM_PARMFLAG/65536);
UNDECLARE      TPASFLAGS2;
%FI

%IF TPASK_SUBEXPR
%THEN TPASMAKE_SUB (%REMOVE (TYPE))
%FI

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

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

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

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

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

```

XFI

%;

The following macro generates the offset for a subexpression call.

MACRO

```
TPASMAKE SUB (SUBNAME) =  
%IF NOT %DECLARED (SUBNAME)  
%THEN FORWARD SUBNAME : VECTOR [0];  
%FI  
OWN    TPASSUBEXP : WORD ALIGN(0) INITIAL (SUBNAME - TPASSUBEXP-2);  
UNDECLARE      TPASSUBEXP;  
%;
```

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$IFSUBLEXP (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  
%;
```

0434 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY