

.TITLE EODEF - EDITPC Pattern Operator Macros
.IDENT 'V04-000'
.NLIST

*
* 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 :
* EDITPC Pattern Operator Encodings

ABSTRACT :
The EDITPC instruction, Edit Packed to Character String, performs the
editing according to the pattern string which consists of one byte
pattern operators. Some pattern operators take no operands. Some
take a repeat count which is contained in the rightmost nibble of the
pattern operator itself. The rest take a one byte operand which
follows the pattern operator immediately. This operand is either an
unsigned integer length or byte character. Edit patterns must end
with the EOSEND pattern operator.

The EODEF macros permit easy construction of the edit pattern.

ENVIRONMENT:

AUTHOR: R. P. Grosso, Creation Date 9-Oct-1980

MODIFIED BY:

-

JP
PCI
JP
PCI
JP
PCI
JP
PCI
JP
PCI
JP
PCI
JP
PCI
JP
PCI
JP
PCI
JP
PC
JP
PC

```

: EOSEND EDIT PATTERN OPERATOR ENCODING
: CALL EOSEND
:
: .MACRO EOSEND
: .BYTE ^X00          : EOSEND encoding
: .ENDM EOSEND

: EOSEND_FLOAT EDIT PATTERN OPERATOR ENCODING
: CALL EOSEND_FLOAT
:
: .MACRO EOSEND_FLOAT
: .BYTE ^X01          : EOSEND_FLOAT encoding
: .ENDM EOSEND_FLOAT

: EOSCLEAR_SIGNIF EDIT PATTERN OPERATOR ENCODING
: CALL EOSCLEAR_SIGNIF
:
: .MACRO EOSCLEAR_SIGNIF
: .BYTE ^X02          : EOSCLEAR_SIGNIF encoding
: .ENDM EOSCLEAR_SIGNIF

: EOSSET_SIGNIF EDIT PATTERN OPERATOR ENCODING
: CALL EOSSET_SIGNIF
:
: .MACRO EOSSET_SIGNIF
: .BYTE ^X03          : EOSSET_SIGNIF encoding
: .ENDM EOSSET_SIGNIF

: EOSSTORE_SIGN EDIT PATTERN OPERATOR ENCODING
: CALL EOSSTORE_SIGN
:
: .MACRO EOSSTORE_SIGN
: .BYTE ^X04          : EOSSTORE_SIGN encoding
: .ENDM EOSSTORE_SIGN

: EOSLOAD_FILL EDIT PATTERN OPERATOR ENCODING
: CALL EOSLOAD_FILL <CH>
: WHERE CH IS THE FILL CHARACTER
:
: .MACRO EOSLOAD_FILL CH
: .BYTE ^X40          : EOSLOAD_FILL encoding
: .IF BLANK <CH>      : check to be sure CH isn't blank
: .WARN              :EOSLOAD_FILL - CHARACTER WAS BLANK OR NOT DELIMITED
: .ENDC
: .IF IDENTICAL CH,/  : avoid .ASCII /// if CH is /
: .BYTE ^X2F          : enter ASCII for "/"
: .IF FALSE           : if CH is not "/" then
: .ASCII /CH/        : fill char placed in fill register
: .ENDC
: .ENDM EOSLOAD_FILL

```

JP

PCI

JP

PCI

JP

PCI

JP

PCI

JP

PCI

JP

PCI

JP

PCI

JP

PCI

JP

PCI

JP

PCI

JP

PC

JP

PC

```

: EOSLOAD SIGN EDIT PATTERN OPERATOR ENCODING
: CALL EOSLOAD SIGN <CH>
: WHERE CH IS THE SIGN CHARACTER

```

```

: .MACRO EOSLOAD_SIGN CH
: .BYTE ^X41 ; EOSLOAD_SIGN encoding
: .IF BLANK <CH> ; check to be sure CH isn't blank
: .WARN ;EOSLOAD_SIGN - CHARACTER WAS BLANK OR NOT DELIMITED
: .ENDC
: .IF IDENTICAL CH,/ ; avoid .ASCII ///... if CH is /
: .BYTE ^X2F ; enter ASCII for "/"
: .IF FALSE ; if CH is not "/" then
: .ASCII /CH/ ; sign char placed in sign register
: .ENDC
: .ENDM EOSLOAD_SIGN

```

```

: EOSLOAD PLUS EDIT PATTERN OPERATOR ENCODING
: CALL EOSLOAD PLUS <CH>
: WHERE CH IS THE SIGN CHARACTER WHEN RESULT IS POSITIVE

```

```

: .MACRO EOSLOAD_PLUS CH
: .BYTE ^X42 ; EOSLOAD_PLUS encoding
: .IF BLANK <CH> ; check to be sure CH isn't blank
: .WARN ;EOSLOAD_PLUS - CHARACTER WAS BLANK OR NOT DELIMITED
: .ENDC
: .IF IDENTICAL CH,/ ; avoid .ASCII ///... if CH is /
: .BYTE ^X2F ; enter ASCII for "/"
: .IF FALSE ; if CH is not "/" then
: .ASCII /CH/ ; char to be placed in sign register
: .ENDC
: .ENDM EOSLOAD_PLUS

```

```

: EOSLOAD MINUS EDIT PATTERN OPERATOR ENCODING
: CALL EOSLOAD MINUS <CH>
: WHERE CH IS THE SIGN CHARACTER WHEN RESULT IS NEGATIVE

```

```

: .MACRO EOSLOAD_MINUS CH
: .BYTE ^X43 ; EOSLOAD_MINUS encoding
: .IF BLANK <CH> ; check to be sure CH isn't blank
: .WARN ;EOSLOAD_MINUS - CHARACTER WAS BLANK OR NOT DELIMITED
: .ENDC
: .IF IDENTICAL CH,/ ; avoid .ASCII ///... if CH is /
: .BYTE ^X2F ; enter ASCII for "/"
: .IF FALSE ; if CH is not "/" then
: .ASCII /CH/ ; char to be placed in sign register
: .ENDC
: .ENDM EOSLOAD_MINUS

```

```

: EOSINSERT EDIT PATTERN OPERATOR ENCODING
: CALL EOSINSERT <CH>
: WHERE CH IS INSERTED

```

JP

:

:

:

JP

PHI

:

JP

PHI

:

JP

PHI

:

JP

PHI

:

JP

PHI

:

JP

PHI

:

JP

PHI

:

JP

```

.MACRO EOSINSERT CH
.BYTE ^X44 ; EOSINSERT encoding
.IF BLANK <CH> ; check to be sure CH isn't blank
.WARN ;EOSINSERT - CHARACTER WAS BLANK OR NOT DELIMITED
.ENDC
.IF IDENTICAL CH,/ ; avoid .ASCII /// if CH is /
.BYTE ^X2F ; enter ASCII for "/"
.IF FALSE ; if CH is not "/" then
.ASCII /CH/ ; char to be inserted
.ENDC
.ENDM EOSINSERT

```

```

: EOSBLANK_ZERO EDIT PATTERN OPERATOR ENCODING
: CALL EOSBLANK_ZERO LEN
: WHERE LEN IS A POSITIVE INTEGER
:

```

```

.MACRO EOSBLANK_ZERO LEN
.BYTE ^X45 ; EOSBLANK_ZERO encoding
.IF EQUAL LEN
.WARN ;EOSBLANK_ZERO - LENGTH SHOULD NOT EQUAL ZERO
.ENDC
.BYTE LEN ; length to fill with contents of
; fill register if value of source
; string is zero.
.ENDM EOSBLANK_ZERO

```

```

: EOSREPLACE_SIGN EDIT PATTERN OPERATOR ENCODING
: CALL EOSREPLACE_SIGN LEN
: WHERE LEN IS A POSITIVE INTEGER
:

```

```

.MACRO EOSREPLACE_SIGN LEN
.BYTE ^X46 ; EOSREPLACE_SIGN encoding
.IF EQUAL LEN
.WARN ;EOSREPLACE_SIGN - LENGTH SHOULD NOT EQUAL ZERO
.ENDC
.BYTE LEN
.ENDM EOSREPLACE_SIGN

```

```

: EOSADJUST_INPUT EDIT PATTERN OPERATOR ENCODING
: CALL EOSADJUST_INPUT LEN
: WHERE LEN IS A POSITIVE INTEGER
:

```

```

.MACRO EOSADJUST_INPUT LEN
.BYTE ^X47 ; EOSADJUST_INPUT encoding
.IF LESS_EQUAL LEN
.WARN ;EOSADJUST_INPUT - LENGTH MUST BE GREATER THAN ZERO
.ENDC
.IF GREATER_EQUAL LEN - 32
.WARN ;EOSADJUST_INPUT - LENGTH MUST BE LESS THAN 32
.ENDC

```

JP

PH

:

JP

PH

:

JP

PH

:

JP

PH

:

JP

PH

:

JP

PH

:

JP

PH

:

JP

PH

:

JP

PH

:

JP

PH

:

JP

PH

:

JP

PH

:

JP

PH

:

JP

```

.BYTE  LEN                : length to fill with contents of
                        : fill register if value of source
                        : string is zero.
.ENDM  EOSADJUST_INPUT

```

```

: EOSFILL EDIT PATTERN OPERATOR ENCODING
:
:
:

```

```

CALL EOSFILL R
WHERE R, THE REPEAT COUNT IS BETWEEN 1 AND 15 INCLUSIVE

.MACRO EOSFILL R
:IF LESS_EQUAL R          : repeat cannot be zero
.WARN  :EOSFILL - REPEAT CANNOT BE ZERO
.ENDC
:IF GREATER_EQUAL R - 16  : repeat count must be contained in
                        : a nibble
.WARN  :EOSFILL - REPEAT IS GREATER THAN 15
.ENDC
.BYTE  <^X80 + R>        : EOSFILL operator encoding
                        : plus repeat count
.ENDM  EOSFILL

```

```

: EOSMOVE EDIT PATTERN OPERATOR ENCODING
:
:
:

```

```

CALL EOSMOVE R
WHERE R, THE REPEAT COUNT IS BETWEEN 1 AND 15 INCLUSIVE

.MACRO EOSMOVE R
:IF LESS_EQUAL R          : repeat cannot be zero
.WARN  :EOSMOVE - REPEAT CANNOT BE ZERO
.ENDC
:IF GREATER_EQUAL R - 16  : repeat count must be contained in
                        : a nibble
.WARN  :EOSMOVE - REPEAT IS GREATER THAN 15
.ENDC
.BYTE  <^X90 + R>        : EOSMOVE operator encoding
                        : plus repeat count
.ENDM  EOSMOVE

```

```

: EOSFLOAT EDIT PATTERN OPERATOR ENCODING
:
:
:

```

```

CALL EOSFLOAT R
WHERE R, THE REPEAT COUNT IS BETWEEN 1 AND 15 INCLUSIVE

.MACRO EOSFLOAT R
:IF LESS_EQUAL R          : repeat cannot be zero
.WARN  :EOSFLOAT - REPEAT CANNOT BE ZERO
.ENDC
:IF GREATER_EQUAL R - 16  : repeat count must be contained in
                        : a nibble
.WARN  :EOSFLOAT - REPEAT IS GREATER THAN 15
.ENDC
.BYTE  <^XA0 + R>        : EOSFLOAT operator encoding
                        : plus repeat count
.ENDM  EOSFLOAT

```

↑
↓
-

END OF EDIT PATTERN OPERATOR ENCODINGS

