Stanford Artificial Intelligence Laboratory
Memo AIM-263

Computer Science Department
Report No. STAN-CS-75-503

July 1975

12 FG

The Macro Processing System STAGE2:

Transfer of Comments to the Generated Text

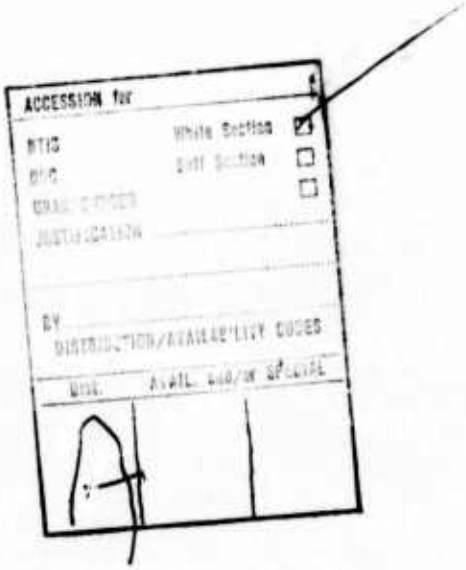
by

Odd Pettersen

Research sponsored by

Advanced Research Projects Agency
ARPA Order No. 2494

COMPUTER SCIENCE DEPARTMENT
Stanford University


ACCESSION for
NTIS
DDC
UNANNOUNCED
JUSTIFICATION
BY
DISTRIBUTION/AVAILABILITY CODES
DIST.
AVAIL. and/or SPECIAL

Stanford Artificial Intelligence Laboratory
Memo AIM-263

July 1975

Computer Science Department
Report No. STAN-CS-75-503

The Macro Processing System STAGE2:

Transfer of Comments to the Generated Text

by

Odd Pettersen

ABSTRACT

This paper is a short description of a small extension of STAGE2, providing possibilities to copy comments etc. from the source text to the generated text. The description presupposes familiarity with the STAGE2 system: its purpose, use and descriptions, like [1] to [9]. Only section 3 of this paper requires knowledge of the internal structures and working of the system, and that section is unnecessary for the plain use of the described feature. The extension, if not used, is completely invisible to the user: No rules, as described in the original litterature, are changed. A user, unaware of the extension, will see no difference from the original version.

This research was supported by the Advanced Research Projects Agency of the Department of Defense under Contract DAHC 15-73-C-0435 . The views and conclusions contained in this document are those of the author(s) and should not be interpreted as necessarily representing the official policies, either expressed or implied, of Stanford University, ARPA, or the U. S. Government.

Reproduced in the U.S.A. Available from the National Technical Information Service, Springfield, Virginia 22161.

The Macro processing system STAGE2: Transfer of comments to the generated text.

by

Odd Pettersen

SINTEF, Div. of Automatic Control,
The Technical University of Norway
(presently with
Stanford University
Artificial Intelligence Lab.
Computer Science Dept.)

December 1974

1. INTRODUCTION

The following is a short description of a small extension of STAGE2, providing possibilities to copy comments etc. from the source text to the generated text. The description presupposes familiarity with the STAGE2 system: its purpose, use and descriptions, like [1] to [9]. Only section 3 of this paper requires knowledge of the internal structures and working of the system, and that section is unnecessary for the plain use of the described feature.

The extension, if not used, is completely invisible to the user: No rules, as described in the original litterature, are changed. A user, unaware of the extension, will see no difference from the original version.

2. THE FLAG LINE

As described in [1], the input material for any translation by STAGE2 starts with a collection of macros, defining the correspondence between the source language of input (program) text following the macros, and the target language. In the very beginning of the input material, however, and preceeding the macros, is a so-called FLAG LINE. This line, terminated by carriage return, defines the character set used.

2.1. The original format of the Flag Line.

As also stated in [1], the flag line consists of the following characters, in the order shown:

Pos. in Flag line	Function of Character	Usual character
1	Source end-of-line flag	Ø
2	Source parameter flag	#
3	Target end-of-line flag	ß
4	Target parameter flag	#
5	Zero, defines all digits	0
6	Space. Also used as padding character	space
7	Left parenthesis	(
8	Addition operator	+
9	Subtraction operator	-

10	Multiplication operator	*
11	Division operator	/
12	Right parenthesis	)

If the character following immediately after), i.e. in position 13, is not a carriage return or space, the flag line will be considered "extended", effecting the copying of comments, as explained in section 2.2. Contrary, if the character immediately following) is carriage return or space (the latter relevant for punched cards), the flag line is considered "normal", and nothing is changed, with respect to use or the appearance of the generated text.

2.2. Extended Flag Line.

With a simple extension of the flag line, one can specify that comments in the source text shall be copied over to the generated text lines. By "comments" is here meant strings of characters, other than spaces, following the source end-of-line flag, before carriage return or end of line. As required by assembler or other systems program, to be used for later processing of the generated text, comments in the generated text should usually begin with a special delimiter, after which the remainder of the line will be considered as comments and neglected by the assembler (or post-processor). One often used symbol for this purpose is ; but any single character can be used, since this is to be specified on the extended flag-line. An example of an extended flag-line can look like:

.#$#0␣(+-*/)8 tab tab ␣␣␣;

where ␣ here symbolizes one space character, and tab symbolizes one "tab".

The first 12 characters are unchanged. Further along the line is typed the comment delimiter, which here is ; , in the position where it is to appear in the generated lines. The comments will be inserted in the generated first line, immediately following the delimiter, such that any tabulator or space characters in front of the comments in the source text will be suppressed, and substituted by tabulator and space characters as necessary to place the delimiter and comments in the proper place on the line. Also, if the source line comments begin with the delimiter character (following possible leading tabs and spaces), this delimiter is suppressed, to give only one delimiter character. Delimiter characters later on the line will be copied normally, however. By "first line" is here meant the first of the the group of lines generated by one macro, i.e. the group of lines corresponding to the source line.

The first character following) (i.e. in position 13) in the flag line should be an integer, in the range 1 to 9, specifying W = the number of spaces equivalent to one tabulator. The next character should be the tabulator character ("tab"). Following this, comes any number of more "tabs" (may be zero), followed by any number of spaces, which can also be zero. Tabs and spaces can not be mixed.

More concisely: If the number of concecutive tabs in the flag line is k, followed by j spaces, the comment delimiter will be placed in position number:

$$D = [(13 : W) + k] * W + j$$

where : symbolizes integer division, discarding remainder.

The integer 13 originates from the thirteen leading characters on the line, before the first tab. As a matter of fact, one can simply forget the formula and just put the comment delimiter in the proper position along the line.

If the generated string, before comments, extends beyond the position specified for start of

comments, the comments will start immediately after the generated string.

To summarize, the character positions along an extended flag line have the following significance:

Pos. in Flag line	Function of Character	Usual character
1	Source end-of-line flag	Ø
2	Source parameter flag	#
3	Target end-of-line flag	$
4	Target parameter flag	#
5	Zero, defines all digits	0
6	Space. Also used as padding character	space
7	Left parenthesis	(
8	Addition operator	+
9	Subtraction operator	-
10	Multiplication operator	*
11	Division operator	/
12	Right parenthesis	)
13	=W, number of spaces between tabulator positions	8
14+i (i=0,1,...,k-1)	Tabulator character value	tab
next j characters (j=0 permitted)	Space, as padding character, used to fix the start of the comment section between two tab positions	space
14+k+j	Comment delimiter	;

Possible further characters, up to carriage return, are ignored.

3. DESCRIPTION OF MODIFICATIONS IN STAGE2

The modification of STAGE2, necessary to record extended flag-line and provide copying of comments, consists of 3 parts:

1. Extension of the first part of STAGE2, reading the flag line.
2. Modification of the line input routine.
3. Modification and extension of the routine that outputs characters to the line buffer.

Part 1 is close to the beginning of STAGE2, part 2 is around the label LOC 03, and part 3 is at LOC 57. The modifications are shown in the enclosed listings, on the following pages. The modifications are distinguished from the original parts of the program, by the use of small letters for comments. Also, the new or modified lines are not finished with the word STG2, which indicates original program text. The semicolon, introducing each comment, is superfluous here, it is merely included due to a habit of the author. The listings included here are only extracts of the program, showing the modifications and their surroundings. By comparison with a complete listing of the original text, it should be fairly simple to spot the places where the modifications are made.

The modified program-text partly explains itself, through the comments included. A couple of further details to note are, however:

The set of registers of the simulated FLUB machine is extended with 6 more triples: FLG, VAL, and

PTR, with suffix: AA, AB, AD, AE, AF, and ZC. This involves that the modified version of STAGE2 *no longer can be translated by SIMCMP*, since one more character is used in these variable names. For the bootstrapping implementation is therefore recommended, that the original version is used, until a primitive version of STAGE2 is running. Then, this one can be used to translate the modified version.

The variables mentioned are used in the following applications:

	VAL	PTR
AA	value of char	pointer, current character of comment text
AB	not used	pointer, end of comment text
AD	working variable	number of positions between each tab position
AE	tab character value	pos. no. corresponding to integer tabs before comments
AF	comment delimiter value	pos. no. of beginning comments
ZC	not used	pointer for current character during output

One more remark is important, concerning PTR ZC: This variable is also manipulated, and changed, by the internal mechanisms of the I/O-package:

PTR ZC is set to 0 at each call of READ NEXT *. and of WRITE NEXT *.. It is incremented by each statement CHAR = VAL *.. Here, * signifies any valid parameter, according to macro notations.

Only PTR ZC is affected by internal operations. All other FLUB registers are only modified by the STAGE2 program, as positively expressed by FLUB statements.

4. REFERENCES

[1] Waite, W. M.: A language independent macro processor. CACM, 10 (July 1967).

[2] Waite, W. M.: Building a Mobile Programming System. Tech. report 69-2, Graduate School Computing Center, Univ. of Colorado, 1969.

[3] Waite, W. M.: The STAGE2 macro processor. Tech. reports 69-3,69-3B, Graduate School Computing Center, Univ. of Colorado, 1969.

[4] Waite, W. M.: The implementation of STAGE2. Graduate School Computing Center, Univ. of Colorado.

[5] Waite, W. M.: A New Input/Output Package for the Mobile Programming System. Tech. report 71-10. Graduate School Computing Center, Univ. of Colorado, 1971.

[6] John M. Chambers: The STAGE2 Macro Processor. Academic Computing Center, The University of Wisconsin - Madison, July 1972.

[7] John M. Chambers: STAGE2 - FLUB. Unpublished note, Academic Computing Center, The University of Wisconsin - Madison, Jan. 1973.

[8] John M. Chambers: STAGE2 - INTRODUCTION. Unpublished note, Academic Computing Center, The University of Wisconsin - Madison, Febr. 1973.

[9] John M. Chambers: STAGE2 - DEBUGGING. Unpublished note, Academic Computing Center, The University of Wisconsin - Madison, Febr. 1973.

APPENDIX

The following pages comprise:

1. Extracts from listings of STAGE2, in FLUB.
2. Example: Macros for translation from FLUB to assembly for PDP-10
3. Extracts from PDP-10 assembly version of STAGE2, extracts corresponding to item 1 above, as translated with macros, item 2.

APPENDIX 1: Extracts from listings of STAGE2, in FLUB.

(Section containing routines for input of Flag line and for input of normal lines)

```
PTR J = 0 + 0.                                                    STG2
FLG L = 1.          END-OF-LINE INDICATOR.                        STG2
VAL L = 0 - 1.      CARRIAGE RETURN IS -1.                        STG2
PTR L = 0 + 0.      LOCATION COUNTER.                             STG2
VAL M = CHAR.       LEFT PARENTHESIS.                             STG2
PTR M = 0 + 0.      RESET THE SKIP COUNT.                         STG2
FLG N = 0.          SET EXPRESSION SIGN POSITIVE.                 STG2
VAL N = CHAR.       ADDITION OPERATOR.                            STG2
FLG O = 0.                                                        STG2
VAL O = CHAR.       SUBTRACTION OPERATOR.                         STG2
VAL P = CHAR.       MULTIPLICATION OPERATOR.                      STG2
VAL Q = CHAR.       DIVISION OPERATOR.                            STG2
VAL R = CHAR.       RIGHT PARENTHESIS.                            STG2

VAL AF = 0 + 0.          ; read extension of FLAG LINE
VAL AE = 0 + 0.          ; initialize
PTR AE = 0.
PTR AF = 0.
PTR AD = 0.              ; remains=0 if no extension
VAL AD = CHAR.
TO 1E IF VAL AD LT 0.    ; no extension of FLAG LINE
TO 1E IF VAL AD = F.     ; no extension of FLAG LINE
VAL AD = AD - E.         ; corresp. no. positions for TAB
PTR AD = VAL AD.         ; into tab.-parameter
PTR AF = 5 + 3.          ; POSNO(no of pos.)←=13
PTR AE = AF / AD.        ; NMTAB ← POSNO/TABPOS
PTR AF = 0.              ; initialize POSNO
VAL AE = CHAR.           ; TAB-character value
LOC 1A.
PTR AE = AE + 1.         ; count no. of tabs
VAL AD = CHAR.
TO 1D IF VAL AD LT 0.    ; terminate extension
TO 1A IF VAL AD = AE.    ; read new if tab
LOC 1B.
TO 1C IF VAL AD NE F.    ; jump if not space
PTR AF = AF + 1.         ; count no. of spaces
VAL AD = CHAR.           ; read new if space
TO 1B.
LOC 1C.
TO 1D IF VAL AD LT 0.    ; terminate extension
TO 1A IF VAL AD = AE.    ; read new if tab
VAL AF = AD.             ; read comment-delimiter
LOC 1D.                  ; calculate extension-parameters
PTR AE = AE * AD.        ; pos.no. corresp. to integer tabs
PTR AF = AE + AF.        ; pos.no. of beginning comments

LOC 1E.                  ; continue original STAGE2
PTR R = 0 + 0.      SET NO REPETITION IN PROGRESS.                STG2
PTR 4 = 7 + 7.      LENGTH OF TWO DESCRIPTORS.                    STG2
PTR 8 = F + 7.      POINT TO THE FIRST AVAILABLE SPACE.           STG2
TO 01 BY D.         START WORKING IN EARNEST.                     STG2
LOC 01.             ROUTINE TO READ FROM THE INPUT.               STG2
GET I = A.          RECALL THE CHANNEL SPEC.                      STG2
READ NEXT I.        GRAB A LINE.                                  STG2
TO 38 IF FLG I NE 0.  GET OUT UNLESS ALL IS OK.                   STG2
PTR I = C + 0.                                                    STG2
VAL Y = 0 + 0.                                                    STG2
PTR Y = C + 0.                                                    STG2
```

```
TO 02 IF PTR M = 0.          SHOULD THIS LINE BE SKIPPED, NO.          STG2
PTR M = M - 1.               YES. DROP THE SKIP COUNT                  STG2
TO 01.                       TRY AGAIN.                                STG2
LOC 02.                      READING LOOP.                             STG2
PTR 9 = I + 0.               ADVANCE THE SPACE POINTER.                STG2
VAL I = CHAR.                READ THE NEXT CHARACTER.                  STG2
PTR I = 9 - 7.               POINT TO THE NEXT CHARACTER SPACE.        STG2
TO 97 IF PTR 8 GE I.         HAVE WE OVERRUN THE AREA, YES.            STG2
STO 9 = I.                   PUT AWAY THE CHARACTER.                   STG2
TO 04 IF VAL I = L.          WAS THIS A CARRIAGE RETURN, YES.          STG2
TO 03 IF VAL I = A.          HAVE WE COMPLETED THE READ, YES.          STG2
VAL Y = Y + 1.               BUMP THE INPUT STRING LENGTH.             STG2
TO 02 IF VAL I NE B.         NO, IS THIS A PARAMETER FLAG, NO.         STG2
PTR B = I + 0.               YES, SET THE PARAMETER POINTER AND        STG2
STO 9 = B.                   STORE IT WITH THE PHASE FLAG.             STG2
TO 02.                                                                 STG2
LOC 03.                      READ THE REMAINDER OF THE LINE.           STG2
PTR AA = 9.                ; remark limitmark
LOC 0A.                    ; loop to read comments etc.
TO 0B IF VAL I = L.        ; car.ret., i.e. no remainder
PTR 9 = I + 0.
LOC 0G.          ;
VAL I = CHAR.;                                          ;
TO 01 IF VAL I NE A.
VAL I = CHAR.              ; read new if first was "source EOL-flag"
LOC 01.
TO 0G IF VAL I = F.        ; eliminate leading spaces
TO 0G IF VAL I = AE.       ; eliminate leading tabs
TO 0J IF VAL I NE AF.      ; continue if no comment-delimiter
VAL I = CHAR.              ; eliminate comment-delimiter
TO 0H.
LOC 0J.
PTR AA = 9.                ; adjust startpointer
LOC 0H.                    ; normal read/store loop
PTR I = 9 - 7.
STO 9 = I.
TO 97 IF PTR 8 GE I.       ; error if full
TO 0B IF VAL I = L.        ; terminate when car.ret.
PTR 9 = I + 0.
VAL I = CHAR.              ; read next
TO 0H.                     ; and continue in loop
LOC 0B.                    ; remainder of line finished
PTR AB = 9.                ; set terminate-pointer
LOC 04.                        SCANNER.                                STG2
PTR U = 9 - 7.                 SET ALL PARAMETERS UNDEFINED.           STG2
STO U = 3.                                                             STG2
PTR U = U - 7.                                                         STG2
STO U = 3.                                                             STG2
PTR U = U - 7.                                                         STG2
STO U = 3.                                                             STG2
PTR U = U - 7.                                                         STG2
STO U = 3.                                                             STG2
PTR U = U - 7.                                                         STG2
STO U = 3.                                                             STG2
PTR U = U - 7.                                                         STG2
STO U = 3.                                                             STG2
PTR U = U - 7.                                                         STG2
STO U = 3.                                                             STG2
PTR U = U - 7.                                                         STG2
STO U = 3.                                                             STG2
PTR U = U - 7.                                                         STG2
STO U = 3.                                                             STG2
```

(Section containing the Modified output routine)

```
STO 8 = 1.                          YES, SET THE TERMINATOR.                                STG2
PTR 8 = 8 + 7.                      ADVANCE THE SPACE POINTER.                              STG2
TO 97 IF PTR 8 GE 9.                HAVE WE OVERRUN THE AREA, YES.                          STG2
VAL I = CHAR.                       GET THE NEXT CHARACTER.                                 STG2
TO 55 IF VAL I NE C.                DID THAT CLOSE THE DEFINITION PHASE, NO.                STG2
FLG B = 0.                          YES, RESET THE PHASE FLAG.                              STG2
LOC 55.                             COMMON SYSTEM RETURN POINT.                             STG2
RETURN BY D.                        REGISTER D IS THE RETURN ADDRESS.                       STG2
LOC 56.                             PUNCH AN UNRECOGNIZED LINE.                             STG2
VAL W = 3 + 0.                      CHANNEL 3 USED WHEN A LINE IS NOT MATCHED.              STG2
PTR X = C + 0.                      ADDRESS THE FIRST CHARACTER.                            STG2
PTR AA = AB                     ; eliminate double output of comments
LOC 57.                         ; output characters
GET X = X.                      ; norm. loop, get character
TO 00 IF VAL X NE L.            ; continue in normal loop if not CR
TO 00 IF PTR AD = 0.            ; or if "main extension swith" off
TO 0D IF PTR AA = AB.           ; or if no remainder in this line
LOC 2A.                         ; output remainder of line:
TO 2B IF PTR ZC GE AE.          ; if pos.counter less integr. tab.pos
CHAR = VAL AE.                  ; then output tab and loop
TO 2A.
LOC 2B.                         ; loop for spaces
TO 2C IF PTR ZC GE AF.
CHAR = VAL F.                   ; put out space(s)
TO 2B.
LOC 2C.
TO 0C IF VAL AF = 0.
CHAR = VAL AF.                  ; put out comment-delimiter
LOC 0C.                         ; loop for output of remaining string
GET AA = AA.                    ; get first char.
TO 2E IF VAL AA NE A.           ; eliminate possible "source EOL-flag"
LOC 2D.                         ; loop for output remainder (comments)
GET AA = AA.                    ; get next character
LOC 2E.
CHAR = VAL AA.                  ; put out the character
TO 0F IF FLG AA = 1.            ; possible terminate
TO 20 IF PTR AA NE AB.          ; go loop if not exhausted
LOC 0D.                         ; normal output-loop
CHAR = VAL X.
TO 0E IF VAL X = L.             ; terminate if CR
TO 57 IF FLG X NE 1.                HAVE WE REACHED THE END, NO.                            STG2
LOC 0F.
PTR AA = AB.                    ; squeeze possible remaining of "remainder"
LOC 0E.                         ; end of line reached:
WRITE NEXT W.                       YES, PUT IT OUT ON THE DESIGNATED CHANNEL.          STG2
TO 98 IF FLG W NE 0.                TREAT ANY ERROR AS FATAL.                           STG2
TO 55 IF VAL X = L.                 ELSE IF THE LINE IS COMPLETE, RETURN.               STG2
CHAR = VAL X.                       ELSE REPRINT THE LAST CHARACTER                     STG2
TO 57.                              AND CONTINUE.                                       STG2
LOC 58.                             TRY FOR AN ALTERNATIVE MATCH.                       STG2
PTR Z = W + Z.                      GET THE POINTER TO THE ALTERNATIVE.                 STG2
TO 60 IF PTR W NE 0.                WAS THERE ONE AFTER ALL, YES.                       STG2
TO 71 IF FLG B = 2.                 NO, ARE WE DEFINING, YES.                           STG2
LOC 59.                             TRY EXTENDING THE PREVIOUS PARAMETER.               STG2
TO 70 IF PTR V GE 9.                IS THERE ONE TO EXTEND, NO.                         STG2
GET Z = V.                          RECALL THE MACRO POINTER.                           STG2
GET Y = Q.                          YES, RECALL THE INPUT POINTER                       STG2
GET X = Y.                          AND THE CURRENT CHARACTER                           STG2
TO 63 IF FLG Z = 2.                 IS THIS THE FIRST TIME FOR A PARAMETER, YESSTG2
TO 64 IF FLG Z = 3.                 NO, IS IT A PARAMETER EXTENSION, YES.               STG2
```

APPENDIX 2: Example: Macros for translation from FLUB to assembly for PDP-10

```
.#$#0 (+-*/)8                    ;
GET # = #.
IF AC2 = 'PTR#20' SKIP 1$
        MOVE    2,PTR#20#F1$
SET AC2 TO 0$
        MOVEI 15,FLG#10#F1$
        JSR  UNPACK#F1$
$
STO # = #.
IF AC2 = 'PTR#10' SKIP 2$
        MOVE    2,PTR#10#F1$
SET AC2 TO 'PTR#10'$
        MOVEI 15,FLG#20#F1$
        JSR   PACK#F1$
$
FLG # = #.
IF #20 NE 0 SKP 2$
        SETZM    FLG#10#F1$
SKIP 4$
IF AC2 = 'FLG#20' SKIP 1$
        MOVE    2,FLG#20#F1$
SET AC2 TO 'FLG#10'$
        MOVEM  2,FLG#10#F1$
$
VAL # = PTR #.
IF AC2 = 'PTR#20' SKIP 1$
        MOVE    2,PTR#20#F1$
SET AC2 TO 'VAL#10'$
        MOVEM  2,VAL#10#F1$
$
PTR # = VAL #.
IF AC2 = 'VAL#20' SKIP 1$
        MOVE    2,VAL#20#F1$
SET AC2 TO 'PTR#10'$
        MOVEM  2,PTR#10#F1$
$
PTR # = 0.
        SETZM    PTR#10#F1$
$
VAL # = 0.
        SETZM    VAL#10#F1$
$
# # = 0 + 0.
        SETZM    #10#20#F1$
$
# # = # + 0.
IF AC2 = '#10#30' SKIP 1$
        MOVE    2,#10#30#F1$
        MOVEM  2,#10#20#F1$
SET AC2 TO '#10#20'$
$
```

```
# # = # + 1.
IF #20 NE #30 SKP 3$
IF AC2 = '#10#30' SKIP 5$
        AOS     #10#20#F1$
SKIP 5$
IF AC2 = '#10#30' SKIP 1$
        MOVE    2,#10#30#F1$
SET AC2 TO '#10#20'$
        AOJ     2,#F1$
        MOVEM   2,#10#20#F1$
$
# # = # - 1.
IF #20 NE #30 SKP 3$
IF AC2 = '#10#30' SKIP 5$
        SOS     #10#20#F1$
SKIP 5$
IF AC2 = '#10#30' SKIP 1$
        MOVE    2,#10#30#F1$
SET AC2 TO '#10#20'$
        SOJ     2,#F1$
        MOVEM   2,#10#20#F1$
$
# # = # + 7.
#10 #20 = #30 + 1$
$
# # = # - 7.
#10 #20 = #30 - 1$
$
# # = # + #.
IF AC2 = '#10#30' SKIP 3$
IF AC2 = '#10#40' SKIP 3$
        MOVE    2,#10#30#F1$
        ADD     2,#10#40#F1$
SKIP 1$
        ADD     2,#10#30#F1$
SET AC2 TO '#10#20'$
        MOVEM   2,#10#20#F1$
$
# # = # - #.
IF AC2 = '#10#30' SKIP 1$
        MOVE    2,#10#30#F1$
        SUB     2,#10#40#F1$
SET AC2 TO '#10#20'$
        MOVEM   2,#10#20#F1$
$
# # = #.
IF #20 = #30 SKP 4$
IF AC2 = '#10#30' SKIP 1$
        MOVE    2,#10#30#F1$
SET AC2 TO '#10#20'$
        MOVEM   2,#10#20#F1$
$
```

```
PTR # = # * #.
IF #30 NE 7 SKP 3$
PTR #10 = #20$
SKIP 8$
IF AC2 = 'PTR#20' SKIP 2$
IF AC2 = 'PTR#30' SKIP 3$
        MOVE    2,PTR#20#F1$
        IMUL    2,PTR#30#F1$
SKIP 1$
        IMUL    2,PTR#20#F1$
        MOVEM   2,PTR#10#F1$
SET AC2 TO 'PTR#10'$
$
PTR # = # / #.
IF #30 NE 7 SKP 3$
PTR #10 = #20$
SKIP 5$
IF AC2 = 'PTR#20' SKIP 1$
        MOVE    2,PTR#20#F1$
        IDIV    2,PTR#30#F1$
        MOVEM   2,PTR#10#F1$
SET AC2 TO 'PTR#10'$
$
TO # IF # # = #.
IF AC2 = '#20#30' SKIP 2$
        MOVE    2,#20#30#F1$
SET AC2 TO '#20#30'$
        CAMN    2,#20#40#F1$
        JRST  LOC#10#F1$
$
TO # IF # # NE 0.
IF AC2 = '#20#30' SKIP 2$
        MOVE    2,#20#30#F1$
SET AC2 TO '#20#30'$
        JUMPN   2,LOC#10#F1$
$
TO # IF # # GE 0.
IF AC2 = '#20#30' SKIP 2$
        MOVE    2,#20#30#F1$
SET AC2 TO '#20#30'$
        JUMPGE 2,LOC#10#F1$
$
TO # IF # # = 0.
IF AC2 = '#20#30' SKIP 2$
        MOVE    2,#20#30#F1$
SET AC2 TO '#20#30'$
        JUMPE   2,LOC#10#F1$
$
TO # IF # # NE #.
IF AC2 = '#20#30' SKIP 3$
IF AC2 = '#20#40' SKIP 4$
        MOVE    2,#20#30#F1$
SET AC2 TO '#20#30'$
        CAME    2,#20#40#F1$
SKIP 1$
        CAME    2,#20#30#F1$
        JRST  LOC#10#F1$
$
```

```
TO # IF # # GE #.
IF AC2 = '#20#30' SKIP 3$
IF AC2 = '#20#40' SKIP 4$
        MOVE    2,#20#30#F1$
SET AC2 TO '#20#30'$
        CAML    2,#20#40#F1$
SKIP 1$
        CAMG    2,#20#30#F1$
        JRST  LOC#10#F1$
$
TO # IF # # # #.
IF AC2 = '#20#30' SKIP 2$
        MOVE    2,#20#30#F1$
SET AC2 TO '#20#30'$
IF AC13 = 'BOL#40' SKIP 2$
SET AC13 TO 'BOL#40'$
        MOVE  13,BOL#40#F1$
        MOVE  15,#20#50#F1$
        JSR   BOOL#F1$
        JRST  LOC#10#F1$
$
TO #.
        JRST  LOC#10#F1$
$
TO # BY #.
        MOVEI 14,PTR#20#F1$
        MOVEI 15,LOC#10#F1$
        JSP   13,SUBRT#F1$
SET AC13 TO 0$
$
RETURN BY #.
        MOVE  13,PTR#10#F1$
        JRST  (13)#F1$
SET AC13 TO 0$
$
LOC #.
LOC#10:#F1$
SET AC2 TO 0$
SET AC13 TO 0$
$
STOP.
        SETA$
EXIT$
$
END PROGRAM.
LOWEND←.#F1$
BLOCK  MASSIZ#F1$
HIGEND←.#F1$
END  START#F1$
#F0$
$
```

```
READ NEXT #.
        MOVE  5,VAL#10#F1$
        JSR   READIN#F1$
        MOVEM 2,FLG#10#F1$
SET AC2 TO 'FLG#10'$
$
VAL # = CHAR.
        JSR   GET1C#F1$
        MOVEM 2,VAL#10#F1$
SET AC2 TO 'VAL#10'$
$
CHAR = VAL #.
SET AC2 TO 'VAL#10'$
        MOVE  2,VAL#10#F1$
        JSR   UTCHAR#F1$
        MOVEM 3,FLG#10#F1$
$
WRITE NEXT #.
        MOVE  5,VAL#10#F1$
        JSR   WRTLIN#F1$
        MOVEM 2,FLG#10#F1$
SET AC2 TO 'FLG#10'$
$
REWIND #.
        MOVE  5,VAL#10#F1$
        JSR   REWND#F1$
        MOVEM 3,FLG#10#F1$
$
MESSAGE # TO #.
        MOVEI 13,[ASCIZ/#10/]#F1$
        MOVE  15,VAL#20#F1$
        JSR   MSGOUT#F1$
        MOVEM  3,FLG#20#F1$
$
SET # TO #.
#F3$
$
IF # = # SKIP #.
IF #11 = #20 SKP #30$
$
IF # = # SKP #.
#F50$
$
IF # NE # SKP #.
#F51$
$
SKIP #.
#F4$
$$
```

Appendix 3: Extracts from PDP-10 assembly version of STAGE2, extracts corresponding to Appendix 1, as translated with macros, Appendix 2.

(Section containing routines for input of Flag line and for input of normal lines)

```
        SETZM    PTRJ      ;STG2
        MOVE   2,FLG1      ;END-OF-LINE INDICATOR.                    STG2
        MOVEM  2,FLGL
        MOVE   2,VAL0      ;CARRIAGE RETURN IS -1.                    STG2
        SOJ    2,
        MOVEM  2,VALL
        SETZM    PTRL      ;LOCATION COUNTER.                         STG2
        JSR   GET1C        ;LEFT PARENTHESIS.                         STG2
        MOVEM 2,VALM
        SETZM    PTRM      ;RESET THE SKIP COUNT.                     STG2
        SETZM    FLGN      ;SET EXPRESSION SIGN POSITIVE.             STG2
        JSR   GET1C        ;ADDITION OPERATOR.                        STG2
        MOVEM 2,VALN
        SETZM    FLGO      ;STG2
        JSR   GET1C        ;SUBTRACTION OPERATOR.                     STG2
        MOVEM 2,VALO
        JSR   GET1C        ;MULTIPLICATION OPERATOR.                  STG2
        MOVEM 2,VALP
        JSR   GET1C        ;DIVISION OPERATOR.                        STG2
        MOVEM 2,VALQ
        JSR   GET1C        ;RIGHT PARENTHESIS.                        STG2
        MOVEM 2,VALR

        SETZM    VALAF     ; read extension of FLAG LINE
        SETZM    VALAE     ; initialize
        SETZM    PTRAE
        SETZM    PTRAF
        SETZM    PTRAD     ; remains=0 if no extension
        JSR   GET1C
        MOVEM 2,VALAD
        MOVE  13,BOLLT     ; no extension of FLAG LINE
        MOVE  15,VAL0
        JSR   BOOL
        JRST  LOC1E
        CAMN   2,VALF      ; no extension of FLAG LINE
        JRST  LOC1E
        SUB    2,VALE      ; corresp. no. positions for TAB
        MOVEM  2,VALAD
        MOVEM  2,PTRAD     ; into tab.-parameter
        MOVE   2,PTR5      ; POSNO(no of pos.)<-=13
        ADD    2,PTR3
        MOVEM  2,PTRAF
        MOVE   2,PTRAF     ; NMTAB <- POSNO/TABPOS
        IDIV   2,PTRAD
        MOVEM  2,PTRAE
        SETZM    PTRAF     ; initialize POSNO
        JSR   GET1C        ; TAB-character value
        MOVEM 2,VALAE
LOC1A:
        AOS      PTRAE     ; count no. of tabs
        JSR   GET1C
        MOVEM 2,VALAD
        MOVE  13,BOLLT     ; terminate extension
        MOVE  15,VAL0
        JSR   BOOL
        JRST  LOC1D
        CAMN   2,VALAE     ; read new if tab
        JRST  LOC1A
```

```
LOC1B:
        MOVE    2,VALAO     ; jump if not space
        CAME    2,VALF
        JRST    LOC1C
        AOS       PTRAF     ; count no. of spaces
        JSR     GET1C       ; read new if space
        MOVEM   2,VALAO
        JRST    LOC1B
LOC1C:
        MOVE    2,VALAO     ; terminate extension
        MOVE   13,BOLLT
        MOVE   15,VAL0
        JSR     BOOL
        JRST    LOC10
        CAMN    2,VALAE     ; read new if tab
        JRST    LOC1A
        MOVEM   2,VALAF     ; read comment-delimiter
                            ; calculate extension-parameters
LOC10:
        MOVE    2,PTRAE     ; pos.no. corresp. to integer tabs
        IMUL    2,PTRAO
        MOVEM   2,PTRAE
        ADD     2,PTRAF     ; pos.no. of beginning comments
        MOVEM   2,PTRAF

                            ; continue original STAGE2
LOC1E:
        SETZM     PTRR      ;SET NO REPETITION IN PROGRESS.          STG2
        MOVE    2,PTR7      ;LENGTH OF TWO DESCRIPTORS.              STG2
        AOJ     2,
        MOVEM   2,PTR4
        MOVE    2,PTRF      ;POINT TO THE FIRST AVAILABLE SPACE.     STG2
        AOJ     2,
        MOVEM   2,PTR8
        MOVEI  14,PTR0      ;START WORKING IN EARNEST.               STG2
        MOVEI  15,LOC01
        JSP    13,SUBRT
                            ;ROUTINE TO READ FROM THE INPUT.         STG2
LOC01:
        MOVE    2,PTRA      ;RECALL THE CHANNEL SPEC.                STG2
        MOVEI  15,FLGI
        JSR   UNPACK
        MOVE    5,VALI      ;GRAB A LINE.                            STG2
        JSR     READIN
        MOVEM   2,FLGI
        JUMPN   2,LOC98     ;GET OUT UNLESS ALL IS OK.               STG2
        MOVE    2,PTRC      ;STG2
        MOVEM   2,PTRI
        SETZM     VALY      ;STG2
        MOVE    2,PTRC      ;STG2
        MOVEM   2,PTRY

        MOVE    2,PTRM      ;SHOULD THIS LINE BE SKIPPED, NO.        STG2
        JUMPE   2,LOC02
        SOJ     2,          ;YES, DROP THE SKIP COUNT                STG2
        MOVEM   2,PTRM
        JRST  LOC01         ;TRY AGAIN.                              STG2
                            ;READING LOOP,                           STG2
LOC02:                      ;ADVANCE THE SPACE POINTER.              STG2
        MOVE    2,PTRI
        MOVEM   2,PTR9
        JSR     GET1C       ;READ THE NEXT CHARACTER.                STG2
        MOVEM   2,VALI
        MOVE    2,PTR9      ;POINT TO THE NEXT CHARACTER SPACE.      STG2
        SOJ     2,
        MOVEM   2,PTRI
```

```
        CAMG    2,PTR8      ;HAVE WE OVERRUN THE AREA, YES.          STG2
        JRST    LOC97
        MOVE    2,PTR9      ;PUT AWAY THE CHARACTER.                 STG2
        MOVEI   15,FLGI
        JSR     PACK
        MOVE    2,VALI      ;WAS THIS A CARRIAGE RETURN, YES.        STG2
        CAMN    2,VALL
        JRST    LOC04
        CAMN    2,VALA      ;HAVE WE COMPLETED THE READ, YES.        STG2
        JRST    LOC03
        AOS     VALY        ;BUMP THE INPUT STRING LENGTH.           STG2
        CAME    2,VALB      ;NO. IS THIS A PARAMETER FLAG, NO.       STG2
        JRST    LOC02
        MOVE    2,PTRI      ;YES, SET THE PARAMETER POINTER AND      STG2
        MOVEM   2,PTRB
        MOVE    2,PTR9      ;STORE IT WITH THE PHASE FLAG.           STG2
        MOVEI   15,FLGB
        JSR     PACK
        JRST    LOC02       ;STG2
                            ;READ THE REMAINDER OF THE LINE.         STG2
LOC03:
        MOVE    2,PTR9      ; remark limitmark
        MOVEM   2,PTRAA
                            ; loop to read comments etc.
LOC0A:                      ; car.ret., i.e. no remainder
        MOVE    2,VALI
        CAMN    2,VALL
        JRST    LOC0B
        MOVE    2,PTRI
        MOVEM   2,PTR9
                            ;
LOC0G:
        JSR     GETIC       ;
        MOVEM   2,VALI
        CAME    2,VALA      ;
        JRST    LOC0I
        JSR     GETIC       ; read new if first was "source EOL-flag"
        MOVEM   2,VALI

LOC0I:
        MOVE    2,VALI      ; eliminate leading spaces
        CAMN    2,VALF
        JRST    LOC0G
        CAMN    2,VALAE     ; eliminate leading tabs
        JRST    LOC0G
        CAME    2,VALAF     ; continue if no comment-delimiter
        JRST    LOC0J
        JSR     GETIC       ; eliminate comment-delimiter
        MOVEM   2,VALI
        JRST    LOC0H

LOC0J:
        MOVE    2,PTR9      ; adjust startpointer
        MOVEM   2,PTRAA
                            ; normal read/store loop
LOC0H:
        MOVE    2,PTR9
        SOJ     2,
        MOVEM   2,PTRI
        MOVE    2,PTR9
        MOVEI   15,FLGI
        JSR     PACK
        MOVE    2,PTR8      ; error if full
        CAML    2,PTRI
        JRST    LOC97
        MOVE    2,VALI      ; terminate when car.ret.
        CAMN    2,VALL
        JRST    LOC0B
```

```
        MOVE    2,PTRI
        MOVEM   2,PTR9
        JSR     GETIC           ; read next
        MOVEM 2,VALI
        JRST    LOC0H           ; and continue in loop
LOC0B:                          ; remainder of line finished
        MOVE    2,PTR9          ; set terminate-pointer
        MOVEM   2,PTRAB
LOCA4:                          ;SCANNER.                                STG2
        MOVE    2,PTR9          ;SET ALL PARAMETERS UNDEFINED.           STG2
        SOJ     2,
        MOVEM   2,PTRU
        MOVEI 15,FLG3           ;STG2
        JSR     PACK
        SOJ     2,              ;STG2
        MOVEM   2,PTRU
        MOVEI 15,FLG3           ;STG2
        JSR     PACK
        SOJ     2,              ;STG2
        MOVEM   2,PTRU
        MOVEI 15,FLG3           ;STG2
        JSR     PACK
        SOJ     2,              ;STG2
        MOVEM   2,PTRU
        MOVEI 15,FLG3           ;STG2
        JSR     PACK
        SOJ     2,              ;STG2
        MOVEM   2,PTRU
        MOVEI 15,FLG3           ;STG2
        JSR     PACK
        SOJ     2,              ;STG2
        MOVEM   2,PTRU
        MOVEI 15,FLG3           ;STG2
        JSR     PACK
        SOJ     2,              ;STG2
        MOVEM   2,PTRU
        MOVEI 15,FLG3           ;STG2
        JSR     PACK
        SOJ     2,              ;STG2
        MOVEM   2,PTRU
        MOVEI 15,FLG3           ;STG2
        JSR     PACK
```

(Section containing the Modified output routine)

```
        MOVE   2,PTR8      ;YES, SET THE TERMINATOR.                    STG2
        MOVEI  15,FLG1
        JSR    PACK
        AOJ    2,          ;ADVANCE THE SPACE POINTER.                  STG2
        MOVEM  2,PTR8
        CAML   2,PTR9      ;HAVE WE OVERRUN THE AREA, YES.              STG2
        JRST   LOC97
        JSR    GET1C       ;GET THE NEXT CHARACTER.                     STG2
        MOVEM  2,VAL1
        CAME   2,VALC      ;DID THAT CLOSE THE DEFINITION PHASE, NO.    STG2
        JRST   LOC55
        SETZM    FLGB      ;YES, RESET THE PHASE FLAG.                  STG2
                           ;COMMON SYSTEM RETURN POINT.                 STG2
LOC55:                     ;REGISTER D IS THE RETURN ADDRESS.           STG2
        MOVE   13,PTR0
        JRST   (13)
                           ;PUNCH AN UNRECOGNIZED LINE.                 STG2
LOC56:                     ;CHANNEL 3 USED WHEN A LINE IS NOT MATCHED.  STG2
        MOVE   2,VAL3
        MOVEM  2,VALW
        MOVE   2,PTRC      ;ADDRESS THE FIRST CHARACTER.                STG2
        MOVEM  2,PTRX
        MOVE   2,PTRAB     ; eliminate double output of comments
        MOVEM  2,PTRAA
                           ; output characters
LOC57:                     ; norm. loop, get character
        MOVE   2,PTRX
        MOVEI  15,FLGX
        JSR  UNPACK
        MOVE   2,VALX      ; continue in normal loop if not CR
        CAME   2,VALL
        JRST   LOC0D
        MOVE   2,PTRA0     ; or if "main extension swith" off
        JUMPE  2,LOC0D
        MOVE   2,PTRAA     ; or if no remainder in this line
        CAMN   2,PTRAB
        JRST   LOC0O
                           ; output remainder of line:
LOC2A:                     ; if pos.counter less integr. tab.pos
        MOVE   2,PTRZC
        CAML   2,PTRAE
        JRST   LOC2B
        MOVE   2,VALAE     ; then output tab and loop
        JSR    UTCHAR
        MOVEM  3,FLGAE
        JRST   LOC2A
                           ; loop for spaces
LOC2B:
        MOVE   2,PTRZC
        CAML   2,PTRAF
        JRST   LOC2C
        MOVE   2,VALF      ; put out space(s)
        JSR    UTCHAR
        MOVEM  3,FLGF
        JRST   LOC2B
LOC2C:
        MOVE   2,VALAF
        JUMPE  2,LOC0C
        MOVE   2,VALAF     ; put out comment-delimiter
        JSR    UTCHAR
        MOVEM  3,FLGAF
                           ; loop for output of remaining string
LOC0C:
        MOVE   2,PTRAA     ; get first char.
        MOVEI  15,FLGAA
        JSR  UNPACK
```

```
        MOVE    2,VALAA     ; eliminate possible "source EOL-flag"
        CAME    2,VALA
        JRST    LOC2E
LOC2D:                      ; loop for output remainder (comments)
        MOVE    2,PTRAA     ; get next character
        MOVEI 15,FLGAA
        JSR   UNPACK
LOC2E:
        MOVE    2,VALAA     ; put out the character
        JSR     UTCHAR
        MOVEM 3,FLGAA
        MOVE    2,FLGAA     ; possible terminate
        CAMN    2,FLG1
        JRST    LOC0F
        MOVE    2,PTRAA     ; go loop if not exhausted
        CAME    2,PTRAB
        JRST    LOC2D
LOC0D:                      ; normal output-loop
        MOVE    2,VALX
        JSR     UTCHAR
        MOVEM 3,FLGX
        CAMN    2,VALL      ; terminate if CR
        JRST    LOC0E
        MOVE    2,FLGX      ;HAVE WE REACHED THE END, NO.          STG2
        CAME    2,FLG1
        JRST    LOC57
LOC0F:
        MOVE    2,PTRAB     ; squeeze possible remaining of "remainder"
        MOVEM   2,PTRAA
LOC0E:                      ; end of line reached:
        MOVE    5,VALW      ;YES, PUT IT OUT ON THE DESIGNATED CHANNEL. STG2
        JSR     WRTLIN
        MOVEM 2,FLGW
        JUMPN   2,LOC98     ;TREAT ANY ERROR AS FATAL.             STG2
        MOVE    2,VALX      ;ELSE IF THE LINE IS COMPLETE, RETURN. STG2
        CAMN    2,VALL
        JRST    LOC55
        MOVE    2,VALX      ;ELSE REPRINT THE LAST CHARACTER       STG2
        JSR     UTCHAR
        MOVEM 3,FLGX
        JRST    LOC57       ;AND CONTINUE.                         STG2
LOC58:                      ;TRY FOR AN ALTERNATIVE MATCH.         STG2
        MOVE    2,PTRW      ;GET THE POINTER TO THE ALTERNATIVE.   STG2
        ADD     2,PTRZ
        MOVEM   2,PTRZ
        MOVE    2,PTRW      ;WAS THERE ONE AFTER ALL, YES.         STG2
        JUMPN   2,LOC60
        MOVE    2,FLGB      ;NO, ARE WE DEFINING, YES.             STG2
        CAMN    2,FLG2
        JRST    LOC71
LOC59:                      ;TRY EXTENDING THE PREVIOUS PARAMETER. STG2
        MOVE    2,PTRV      ;IS THERE ONE TO EXTEND, NO.           STG2
        CAML    2,PTR9
        JRST    LOC70
        MOVEI 15,FLGZ       ;RECALL THE MACRO POINTER.             STG2
        JSR   UNPACK
        MOVE    2,PTRQ      ;YES, RECALL THE INPUT POINTER         STG2
        MOVEI 15,FLGY
        JSR   UNPACK
        MOVE    2,PTRY      ;AND THE CURRENT CHARACTER             STG2
        MOVEI 15,FLGX
        JSR   UNPACK
```

```
MOVE   2,FLGZ      ;IS THIS THE FIRST TIME FOR A PARAMETER, YESSTG2
CAMN   2,FLG2
JRST  LOC63
CAMN   2,FLG3      ;NO, IS IT A PARAMETER EXTENSION, YES.       STG2
JRST  LOC64
```

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: STAN-CS-75-503	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE *(and Subtitle)*: THE MACRO PROCESSING SYSTEM STAGE2: TRANSFER OF COMMENTS TO THE GENERATED TEXT.	5. TYPE OF REPORT & PERIOD COVERED: Technical (Rept.), July 1975
	6. PERFORMING ORG. REPORT NUMBER: STAN-CS-75-503, [illegible] AIM-263
7. AUTHOR(s): D. C. Pettersen	8. CONTRACT OR GRANT NUMBER(s): DAHC 15-73-C-0435
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Computer Science Department, Stanford University, Stanford, California 94305	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: ARPA Order [illegible]-2494
11. CONTROLLING OFFICE NAME AND ADDRESS: Col. Dave Russell, Deputy Director ARPA/IPT, ARPA Headquarters, 1400 Wilson Blvd., Arlington, Va. 22209	12. REPORT DATE: July [illegible]975
	13. NUMBER OF PAGES: 20 (12) 23p.
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*: ONR Representative: Philipp Surra, Durand Aeronautics Bldg., Rm. 165, Stanford University, Stanford, Ca. 94305	15. SECURITY CLASS. *(of this Report)*: UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT *(of this Report)*: Releasable without limitations on dissemination	DISTRIBUTION STATEMENT A Approved for public release; Distribution Unlimited
17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*	
18. SUPPLEMENTARY NOTES	
19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*	
20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*: see reverse.	

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

094 120

#20

This paper is a short description of a small extension of STAGE2, providing possibilities to copy comments etc. from the source text to the generated text. The description presupposes familiarity with the STAGE2 system: its purpose, use and descriptions, like [1] to [9]. Only section 3 of this paper requires knowledge of the internal structures and working of the system, and that section is unnecessary for the plain use of the described feature. The extension, if not used, is completely invisible to the user: No rules, as described in the original literature, are changed. A user, unaware of the extension, will see no difference from the original version.