Stage 2 Pocket Guide

```
.$.$0 (+-*/) Example flag line
```

Col	Char	Use
1	.	Source end of line flag
2	$	Source Parameter flag
3	.	Code body end of line flag
4	$	Code body escape flag
5	0	Character zero
6		Blank character
7	(	Left parenthesis
8	+	Add operator
9	-	Subtract operator
10	*	Multiply operator
11	/	Divide operator
12	)	Right parenthesis

First Language Under Bootstrap

```
FLG $ = $.          PTR $ = $ + $.
VAL $ = PTR $.      PTR $ = $ - $.
PTR $ = VAL $.      PTR $ = $ * $.
                    PTR $ = $ / $.
GET $ = $.
STO $ = $.          STOP.
                    BEGIN $.
VAL $ = $ + $.      END PROGRAM.
VAL $ = $ - $.
VAL $ = CHAR.       READ NEXT $.
CHAR = VAL $.       WRITE NEXT $.
MESSAGE $ TO $.     REWIND $.

LOC $.              TO $ BY $.
TO $.               RETURN BY $.

TO $ IF FLG $ = $.  TO $ IF PTR $ = $.
TO $ IF FLG $ NE $. TO $ IF PTR $ NE $.
TO $ IF VAL $ = $.  TO $ IF PTR $ GE $.
TO $ IF VAL $ NE $. TO $ IF PTR $ LT $.
```

Parameter Conversions

`$D0` Copy parameter to constructed line

`$D1` Copy value of parameter to CL, if null or undefined copy null

`$D2` Copy value of parameter to CL, if null copy null, if undefined, define from symbol generator

`$D3` Copy break character to CL, if input param at EOL copy null (see `$D7`)

`$D4` Evaluate param as arithmetic expression, copy value to CL, error if non-numeric value

`$D5` Copy length of param to CL

`$D6` Copy CL to parameter

`$D7s` Start iteration, break CL on break characters in string s, repeat iteration at `$F8`

`$D8` Copy integer character code to CL

`$0j` Request jth new symbol from the symbol generator

Where `D` is parameter 1 through 9

Processor Functions

`$F0` Terminate processing, write EOF on output channel.

`$F1m` Output constructed line immediately. If m omitted default is chan 3. If m is 0 (null channel) output is discarded.

`$F2n` Copy channel 1 to channel n.
`m$F2n` Copy channel m to channel n.
`mR$F2nR` Rewind and copy channel m to n.

`$F3` `SET $ = $.`
Set parameter 1 to parameter 2 in symbol table.

`$F4` `SKIP $.`
Skip parameter 1 code body lines.

`$F50` `IF $ = $ SKIP $.`
`$F51` `IF $ -= $ SKIP $.`
Compare parameters 1 and 2, if equal (`F50`) or not equal (`F51`), skip parameter 3 lines.

`$F6-` `IF $ LT $ SKIP $.`
`$F60` `IF $ EQ $ SKIP $.`
`$F61` `IF $ NE $ SKIP $.`
`$F6+` `IF $ GT $ SKIP $.`
Compare as arithmetic expressions.

`$F7` Constructed line sets count controlled loop.

`$F8` Advance an iteration.

`$F9` Escape from code body.

`$FE` Force error traceback.

Stage 2 Template Matching Algorithm

1. If one of the branches of the current node matches the current input character, and if the match is EOL, then the input is recognized, begin code body processing.
2. If the match is not EOL then apply rule 1 to the next input character and the next node of the current branch.
3. Otherwise (no match) if one of the branches of the current node is a parameter flag then apply rule 1 to the current input and the node following the parameter flag.
4. If the current node is the root the match fails, output the line.
5. If the branch entering this node is not the parameter flag, apply rule 3 to the previous node and input.
6. The substring matched by the branch entering the current node is lengthened by appending the shortest balanced substring of the input line beginning at the current input character. Then apply rule 1 to the input character and the new substring.
7. If no such substring can be found apply rule 4 to the previous node and the first input character matched by the parameter flag. If the parameter flag matched the null string , the current input character is used.

Initial Base Register Contents

Reg	FLG	VAL	PTR
0	0	0	0
1	1	1	1
2	2	2	2
3	3	3	3
4		4	
5		5	10
6		6	
7		7	Number of address units per word
8		8	Address 1st free mem
9		9	Address last free mem

Stage 2 Error Messages

`CONV`	Parameter conversion error
`EXPR`	Expression evaluation error
`IOCH`	I/O channel error
`FULL`	Memory full error