RICE UNIVERSITY

AN INTERACTIVE MACROGENERATOR

BY

PETER WILLIAM MEIGS

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

THESIS DIRECTOR'S SIGNATURE:

Edward Alvin Feustel

HOUSTON, TEXAS

APRIL 26, 1972
ABSTRACT

AN INTERACTIVE MACROGENERATOR

AN INTERACTIVE MACROGENERATOR IS A MEANS FOR GENERATING A

BODY OF TEXT. THIS MEANS IS A COMPUTER PROGRAM WHOSE
IMPLEMENTATION PROVIDES A USER WITH A MEANS OF NAMING AND
PARAMETERIZING A STRING OF CHARACTERS. THIS NAME IS ASSOCIATED
WITH A LIST OF ARGUMENTS WHEN IT OCCURS AS A MACRO CALL (WHICH IS
A REQUEST TO EXPAND A NAMED STRING) AND THESE ARGUMENTS ARE USED
TO PRODUCE THE DESIRED TEXT. THE MACRO CALL CAN OCCUR IN ANY
CONTEXT, INCLUDING THE BODY OF THE NAMED STRING BEING EXPANDED.
ITERATION, RECURSION, AND CONDITIONAL EXPRESSIONS ARE HANDLED
SMOOTHLY AND EFFICIENTLY BY THE ALGORITHM DISCUSSED IN THIS
DOCUMENT.

THE DOCUMENT DESCRIBES THE IMPLEMENTATION OF A
MACROGENERATOR AND THE FEATURES IT INCLUDES. THIS DESCRIPTION
INCLUDES A USER'S MANUAL DESCRIBING THE SYNTAX OF THE MACRO CALL
AND THE SEQUENCE OF EVENTS THAT TAKE PLACE TO EVALUATE A MACRO
CALL. THE USER'S MANUAL ALSO DESCRIBES THE PRIMITIVE FUNCTIONS
CONTAINED IN THE MACROGENERATOR. AMONG THESE ARE THE FUNCTIONS
THAT PERFORM ARITHMETIC, DEFINE MACROS AND MANIPULATE MACRO
DEFINITIONS, PERFORM MASS STORAGE INPUT AND OUTPUT OPERATIONS,
PERFORM ITERATION, AND PROVIDE DIAGNOSTIC FACILITIES. EACH
FUNCTION IS THOROUGHLY DISCUSSED, GIVING DETAILED EXAMPLES WHERE HELPFUL.

THE IMPLEMENTATION IS DISCUSSED INTERNALLY. THE MACROGENERATOR IS VIEWED AS A STACK MANIPULATION ROUTINE, WHERE VARIOUS SYMBOLS AND MACRO CALLS ACHIEVE THEIR DESIRED RESULTS BY A SERIES OF STACK OPERATIONS. THESE STACK OPERATIONS ARE DETAILED BY DEFINING A PICTORIAL REPRESENTATION, AND THEN GIVING THE TRANSFORMATION THAT TAKES PLACE UPON ENCOUNTERING THE VARIOUS STACK OPERATORS. THE PRIMITIVE MACROS ARE DISCUSSED IN A SIMILAR FASHION. IN THIS WAY, ALL ASPECTS OF THE OPERATION OF THE ALGORITHM ARE DISCUSSED IN LEVELS OF CUMULATIVELY INCREASING DETAIL.

THE DOCUMENT IS CONCLUDED BY DISCUSSING POSSIBLE EXTENSIONS AND ARE... THAT CAUSED DIFFICULTY IN THEIR IMPLEMENTATION.
THE INTERACTIVE MACROGENERATOR IS INTENDED TO BE AN
EXTENSION OF GPM (GENERAL PURPOSE MACROGENERATOR—SEE
BIBLIOGRAPHY, C. STRACHEY). AS SUCH, EXCEPT FOR DIFFERENT WARNING
CHARACTERS, THE BASIC ALGORITHM IS EXACTLY AS DOCUMENTED BY
STRACHEY. THE EXTENSION PROVIDED HAS BEEN PRIMARILY IN TWO
ASPECTS: DISPOSITION OF INPUT AND OUTPUT, AND ADDITION OF MACHINE
MACROS, WHICH STRENGTHEN THE CAPABILITIES OF GPM. IDEAS FOR SOME
OF THE FEATURES WERE SUGGESTED IN PART BY MATERIAL DISCUSSED IN
THE OTHER REFERENCES INCLUDED IN THE BIBLIOGRAPHY. THESE,
HOWEVER, ARE INCLUDED PRIMARILY TO LEAD THE SERIOUS READER TO
OTHER SOURCES OF SIMILAR MATERIAL.
# Table of Contents

1. Introduction
   1.1 Motivation for Macro Generation  1-1
   1.2 Goals  1-2
   1.3 Uses  1-3
   1.4 Acknowledgements  1-5

   2.1 Syntax & Semantics  2-1
   2.1.1 Macro Call  2-1
   2.1.1.1 Representation in Input Stream  2-1
   2.1.2 Visualization of Macro Call  2-2
   2.1.2 Evaluation  2-2
   2.1.2.1 Nested Calls  2-2
   2.1.2.2 Arguments  2-3
   2.1.2.3 String Quotes  2-3
   2.1.3 Examples  2-4
   2.1.4 Definitions  2-6
   2.1.4.1 Effectuation  2-6
   2.1.4.2 Scope of Definition  2-6
   2.1.4.3 Examples  2-7
   2.2 Machine Macros  2-8
   2.2.1 Control Macros  2-9
   2.2.1.1 DLF  2-9
   2.2.1.2 DO  2-11
   2.2.1.3 BREAK  2-13
2.2.1.4  UPDATE  2-14
2.2.1.5  VAL  2-15
2.2.1.6  * (PARTIAL ARGUMENTS)  2-15
2.2.2  ARITHMETIC  2-18
2.2.2.1  BAR  2-18
2.2.2.2  BIN  2-20
2.2.2.3  DEC  2-21
2.2.2.4  OCT  2-21
2.2.2.5  EXAMPLES  2-21
2.2.3  I/O  2-22
2.2.3.1  FC  2-22
2.2.3.2  RS  2-24
2.2.3.3  WS  2-25
2.2.4  DIAGNOSTIC  2-26
2.2.4.1  DBUG  2-26
2.2.4.2  ECHAIN  2-26
2.3  TEXT HANDLING  2-27
2.3.1  INPUT IMAGES  2-27
2.3.2  CONTINUATION  2-28
2.3.3  OUTPUT IMAGES  2-28
2.4  PROGRAM INVOCATION  2-28
2.4.1  PROGRAM OPTIONS  2-29
2.4.2  INPUT STREAM CORRECTION  2-31
2.4.2.1  WHOLE IMAGE CORRECTIONS  2-31
2.4.2.2  PARTIAL IMAGE CORRECTIONS  2-31
2.4.3 EXAMPLES 2-33
2.4.4 RECURSION 2-33
2.4.5 ITERATION 2-34
2.4.6 EXTERNAL EVALUATION 2-35

3. STRUCTURE 3-1

3.1 NOTATION 3-1
3.2 STACK ENTRIES 3-2
3.2.1 E-CHAIN ENTRY 3-3
3.2.2 P-CHAIN ENTRY 3-3
3.2.3 F-CHAIN ENTRY 3-4
3.2.4 TEMPORARY DEFINITIONS 3-6
3.3 EVALUATION CONTROL 3-6
3.3.1 SCAN CYCLE 3-7
3.4 SCAN MODES 3-10
3.4.1 COPY-MODE 3-10
3.4.1.1 STRING QUOTES 3-10
3.4.1.2 UNWARN CHARACTER 3-10
3.4.2 WARNING CHARACTER MODE 3-11
3.4.2.1 ARGUMENT EXTRACTOR (6) 3-11
3.4.2.2 OPEN MACRO (C) 3-11
3.4.2.3 ARGUMENT SEPARATOR (,) 3-12
3.4.2.4 CLOSE MACRO (J) 3-12
3.4.2.5 MARKER (-U) 3-13
3.4.2.6 DO MARK 3-14
3.4.3 MACHINE MACROS 3-14
3.4.3.1. DIAGNOSTIC (DEBUG, ECHAIN)  3-15
3.4.3.2. I/O MACROS (RS, RS, FC, EUF)  3-15
3.4.3.3. ARITHMETIC HANDLING MACROS (DEC, UCT, BIN)  3-16
3.4.3.4. THE CONTROL MACROS  3-16

4. IMPLEMENTATION  4-1
4.1. NEW FEATURES  4-1
4.2. PROBLEMS ENCOUNTERED  4-3
4.3. POSSIBLE EXTENSIONS  4-4

5. BIBLIOGRAPHY  5-1
1. INTRODUCTION

1.1. MOTIVATION FOR MACRO GENERATION

The use of a computer usually involves problem analysis which has as its goal the discovery of some primitive set of building blocks which may be combined in such a manner in order to reduce a given problem to a small set of elements, which may be easily manipulated. This is what subroutines in Fortran, procedures in Algol, and macros in assembly language do. In Fortran and Algol, the subroutines defined constitute building blocks that are combined by temporarily altering the path of execution. In the assembly language macros, symbol manipulation allows the generation of machine instructions. Here, the building blocks are placed directly in the path of execution. Both of these methods are devices that are designed to extend the language beyond the original version provided. Since the method of solving a problem by solving each of its constituent parts is an integral part of the usage of a computer, it is reasonable to suggest that a generalized facility for providing this ability would be a useful addition to any computer installation. Furthermore, since all solutions are usually represented as symbolic text, the must
LOGICAL METHOD FOR PROVIDING THE ABOVE MENTIONED FACILITY WOULD BE TO PROVIDE A METHOD FOR PARAMETERIZING AND GENERATING SYMBOLIC TEXT. ANY METHOD THAT CAN MANIPULATE TEXT IN A DESIRABLE MANNER WILL ALSO BE A PROGRAMMING LANGUAGE. THIS PROGRAMMING LANGUAGE IS AN INTERPRETIVE ONE, AND AS SUCH, AFFORDS THE ABILITY TO PROVIDE SOME NUMBER OF UNIT RESPONSES FOR EACH UNIT INPUT. THIS ABILITY COMBINED WITH THE ABILITY TO GENERATE TEXT FROM PARAMETERIZED STRINGS IS PROVIDED BY A METHOD THAT IS DISCUSSED HEREIN AND IS TERMED AN "INTERACTIVE MACROGENERATOR".

1.2. GOALS

THE FUNDAMENTAL GOAL IN BUILDING THE INTERACTIVE MACROGENERATOR IS THAT IT BE USEFUL. SUPPORTING THIS MAIN GOAL ARE VARIOUS SUBGOALS, WHICH INCLUDE:

1. INTERACTIVENESS - IN ORDER TO EFFECTIVELY GENERATE TEXT ON A TIMESHARING SYSTEM, THOSE ADVANTAGES THAT EXIST BY RUNNING THE MACROGENERATOR ON-LINE SHOULD BE USED TO THEIR GREATEST EXTENT. THESE ADVANTAGES INCLUDE THE ABILITY TO BUILD UP A SET OF FUNCTIONS AND TEST THEM AS THEY ARE BEING BUILT. IN THIS WAY, EACH STEP CAN BE CONDITIONED ON THE SUCCESS OF THE PREVIOUS STEP. ONCE COMPLETED, IT IS THEN POSSIBLE TO SELECT THOSE INPUT IMAGES WHICH ARE TO BE
RETAINED AS THE INPUT STREAM ONCE THE TESTING FACILITIES ARE REMOVED.

2. COMPLETENESS - THE MACROGENERATOR SHOULD HAVE A SUFFICIENTLY LARGE SET OF PRIMITIVE FUNCTIONS SO THAT THE SET OF PROBLEMS THAT IT CANNOT SOLVE IS VERY SMALL. THIS IMPLIES THAT THE COMPUTATIONAL AND DECISION MAKING FACILITIES FOUND IN MOST COMMON PROGRAMMING LANGUAGES SHOULD BE PRESENT. THESE INCLUDE THE ABILITY TO PERFORM ARITHMETIC, DO INPUT AND OUTPUT OPERATIONS, PERFORM ITERATION, AND BRANCH WITHIN THE PROGRAM LOGIC.

3. EASE IN USE - THE MACROGENERATOR SHOULD BE IMPLEMENTED IN SUCH A WAY THAT IT IS COMPATIBLE WITH THE ENVIRONMENT IN WHICH IT IS TO OPERATE. AS A RESULT, CONFORMITY TO THE STANDARDS APPLIED TO OTHER SYSTEM PROCESSORS IS REQUIRED, MAKING THE PROGRAM EASY TO USE. Thus, ITS UTILITY AS A SERVICE FUNCTION WITHIN THE LARGER SYSTEM CONTEXT WILL BE ENHANCED.

1.3. USES

SINCE ALL OF THE GOALS MENTIONED ABOVE HAVE BEEN ACHIEVED, A LARGE NUMBER OF POTENTIAL USES COME INTO EXISTENCE. THESE INCLUDE:
1. PROGRAM GENERATION - THE MACROGENERATOR CAN BE USED TO AID IN THE BUILDING OF PROGRAMS IN THE SAME WAY AS MACRO FACILITIES ON AN ASSEMBLER. RECURRING SECTIONS OF PROGRAM TEXT THAT WILL NOT CONVENIENTLY MAKE UP THE BODY OF A SUBROUTINE CAN BE PARAMETERIZED AND EXPANDED WHERE NEEDED. THIS SERVES TO INCREASE THE READABILITY OF THE TEXT BEING CREATED AND TO REDUCE THE FREQUENCY OF ERRORS THAT COULD OCCUR IN THE FAULTY GENERATION OF TEXT BY MANUAL MEANS.

2. CONTROL STREAM GENERATION - SINCE CONTROL CARDS ARE, IN EFFECT, THE HIGHEST LEVEL PROGRAMMING LANGUAGE WITHIN AN OPERATING SYSTEM, THE ABILITY TO GENERATE SEQUENCES OF THESE CARDS IS IN MANY RESPECTS SIMILAR TO PROGRAM GENERATION. THE MACROGENERATOR INTERFACES WITH THOSE SYSTEM FACILITIES WHICH ALLOW THE INTERPRETATION OF THE GENERATED CONTROL CARDS.

3. TEXT MANIPULATION - SINCE THE MACROGENERATOR IS AN INTERPRETER FOR A PROGRAMMING LANGUAGE, IN ADDITION TO ITS NORMAL POWERS OF TEXT GENERATION, IT ALSO HAS THE SAME ABILITY TO PERFORM THE TRANSFORMATION FROM DATA INPUT TO DATA OUTPUT AS OTHER PROGRAMMING LANGUAGES. IT IS SUBJECT ONLY TO THE LIMITATIONS OF TIME AND SPACE THAT ARE BUILT INTO ITS REALIZATION.

IT SHOULD BE NOTED THAT, WHILE ALL OF THE ABOVE ABILITIES
1.4 ACKNOWLEDGEMENTS

This work would never have been possible without the efforts of many people. I must first thank my wife, Shirley, for the time she has given up while this work was being done. Mr. J. W. Wyatt of the University Computer Center of the University of Houston must be thanked for providing the computer facilities on which to develop the program. Lockheed Electronics must be thanked for providing the keypunching of this document and for its understanding of the demands that such efforts place on one's time. Mr. J. H. Brassard of Lockheed Electronics must also be thanked for his help in preparing a document with a minimum of grammatical and spelling errors. Finally, Dr. E. A. Feustel must be thanked for his help and encouragement as a thesis advisor.
The Interactive Macrogenerator creates an output stream of characters. This output stream is the result of a direct copy of an input stream or the result of the evaluation of a macro call. The mechanism that is used in the course of calling the macros and their evaluation is described in this chapter.

2.1 Syntax & Semantics

2.1.1 Macro Call

2.1.1.1 Representation in Input Stream

The macro call has an extremely simple syntax: each character represents itself (with the exception of certain carriage control characters) in the course of a direct copy from input to output. Strings, however, that are enclosed by the open macro (\texttt{l}) and the close macro (\texttt{j}) characters are replaced with the result of a macro evaluation of the enclosed string. The string within the bracketing characters is divided into a list of actual arguments by the occurrence of commas (,). There can be any
NUMBER OF ACTUAL ARGUMENTS (INCLUDING NONE) AND EACH ACTUAL
ARGUMENT CAN BE OF ANY LENGTH (INCLUDING LENGTH ZERO). THESE ARE
EVALUATED FROM LEFT TO RIGHT. THE FIRST ACTUAL ARGUMENT (KNOWN AS
ARGUMENT ZERO) IS OF SPECIAL SIGNIFICANCE AND IT NAMES THE STRING
THAT IS TO BE NEAT SCANNED IN THE COURSE OF MACRO EVALUATION.
ARGUMENTS ARE CONSIDERED TO BE THE ACTUAL CHARACTER STRINGS TO BE
USED WHEN THE CALLED MACRO IS EXPANDED—(THAT IS, THEY WILL NOT BE
EVALUATED ANY FURTHER).

2.1.1.2. VISUALIZATION OF MACRO CALL

THE MACRO CALL 'LNAME,ARG1,ARG2,ARG3,...,ARGN' CAN BE
REGARDED AS A CALL ON THE MACRO 'NAME' WITH ARGUMENTS 'ARG1',
'ARG2', ..., 'ARGN'. IT CAN ALSO BE REGARDED AS A GENERALIZED
MACRO EVALUATION CALL WHERE 'NAME' IS MERELY AN ARGUMENT, ALBEIT
ONE OF SPECIAL SIGNIFICANCE. IT WILL BECOME CLEAR THAT THE LATTER
VISUALIZATION IS ONE THAT WILL SIMPLIFY THE UNIFICATION OF THE
CONCEPTS THAT WILL BE PRESENTED IN THIS DOCUMENT.

2.1.2. EVALUATION

2.1.2.1. NESTED CALLS

MACRO EVALUATION CONSISTS OF A SCAN OF THE NAMED STRING.
NESTED MACRO CALLS ARE HANDLED THE SAME WAY THAT THEY WOULD IF
THEY HAD COME IN FROM THE INPUT STREAM DIRECTLY: THAT IS, THEY
CAUSE A SWITCH OF THE EVALUATION SCAN TO ANOTHER NAMED STRING.

2.1.2.2. ARGUMENTS

THE ONE DIFFERENCE IN THE EVALUATION AT THE NAMED STRING LEVEL IS THAT THE ARGUMENTS PROVIDED IN THE MACRO CALL ARE AVAILABLE TO THE NAMED STRING AND CAN BE COPIED TO THE OUTPUT STREAM BY THE OCCURRENCE OF THE SYMBOLS '60', '61', ..., '69'. THESE SYMBOLS REFER TO THE RESPECTIVE ARGUMENTS PROVIDED, WHERE '60' IS THE MACRO NAME, '61' IS THE FIRST ARGUMENT FOLLOWING AND SO ON. ARGUMENTS NOT PROVIDED BUT REFERRED TO IN THE COURSE OF A SCAN CAUSE A NULL STRING (ONE OF ZERO CHARACTERS) TO BE COPIED TO THE OUTPUT STREAM. ARGUMENTS PROVIDED IN THE MACRO CALL BUT NOT REFERRED TO ARE EVALUATED AS PART OF THE ARGUMENT LIST BUT ARE OTHERWISE IGNORED.

IN CERTAIN CASES, IT IS CONVENIENT TO BE ABLE TO ACCESS ARGUMENTS TO THE MACRO WHOSE INVOCATION CAUSED A CALL ON THE MACRO UNDER EVALUATION. THIS IS DONE BY THE SYMBOLS '660', '661', ..., '669'. IN FACT, ANY NUMBER OF LEVELS OF MACRO NESTING CAN BE TRANSCENDED BY THIS METHOD, WHERE ONE ADDITIONAL '69' IS PREFIXED FOR EACH LEVEL THAT IS TO BE TRANSCENDED.

2.1.2.3. STRING QUOTES

IT IS CLEAR THAT IN THE COURSE OF EVALUATION CERTAIN CHARACTERS ARE GIVEN SPECIAL SIGNIFICANCE SUCH AS 'E', 'J', 'L',
AND SO ON. THESE ARE KNOWN AS WARNING CHARACTERS. THERE ARE TWO
METHODS FOR ALLOWING THESE WARNING CHARACTERS TO LOSE THEIR
SIGNIFICANCE TEMPORARILY. ONE MAY IS TO ENCLOSE THE STRING IN THE
STRING QUOTES `<>' AND `>`. A SCAN OF SUCH A STRING CAUSES THE
OUTEROST SET OF QUOTES TO BE STRIPPED OFF AND THE REST OF THE
STRING IS COPIED WITHOUT EVALUATION. ANY STRING QUOTE IN THIS
STRING MUST BE MATCHED BY ITS BRACKETING, CORRESPONDING, STRING
QUOTE, IN ORDER THAT THE PROPER QUOTE BE USED FOR CLOSING THE
QUOTED STRING. THIS REQUIREMENT CAN BE OVERKIDDEN BY THE USE OF
THE UNWARN (\) CHARACTER WHICH CAUSES THE NEXT CHARACTER FOLLOWING
IT TO BE DIRECTLY COPIED. SINCE THE UNWARN CHARACTER IS NOT
COPIED, IT IS INCLUDED IN THE OUTPUT STREAM; IT MUST APPEAR TWICE
IN SUCCESSION IN THE INPUT STREAM. THE UNWARN CHARACTER IS
INTERPRETED EVEN WHEN IT OCCURS BETWEEN STRING QUOTES IN ORDER TO
ALLOW THE USER TO INCLUDE AN UNMATCHED STRING QUOTE IN A QUOTED
STRING. THE UNWARN CHARACTER SHOULD BE USED WITH CAUTION SINCE IT
CAN CAUSE CHARACTERS WHOSE INTERPRETATION IS REQUIRED TO BE
TREATED AS IF THEY WERE NOT SPECIAL CHARACTERS. THIS IS
ESPECIALLY TRUE IF THE UNWARN CHARACTER IS USED IMPROPERLY IN
CONJUNCTION WITH STRING QUOTES.

2.1.3. EXAMPLES

THE PRECEDING SECTIONS CAN BE ILLUMINATED BY THE FOLLOWING
EXAMPLES. SUPPOSE THAT THE NAME 'ABA' IS DEFINED TO BE THE STRING 'XXX&1XXX' AND THAT THE NAME 'QQ' IS DEFINED TO BE THE STRING 'B'. THE FOLLOWING CALLS GIVE THE INDICATED RESULTS.

```
 CALL => RESULT
 1. [QQ] => B
 2. [ABA, ZJ] => XXXXXX
 3. [ABA, [QQ]J] => XXXBXXX
```

SUPPOSE 'ABA' AND 'QQ' ARE DEFINED AS ABOVE AND THE NAME 'C' IS DEFINED TO BE 'ABC[ABA, 6]J[CDJ]' AND 'CDJ' IS DEFINED TO BE 'QQ&1RR'. THEN THE CALL:

```
[EC, MJ] BCOMES THE CALL
[ABLABA, MJ[CDJ] WHICH BECOMES
[ABXXXMXXXQQMRK
```

THE CALL:

```
[EC, [[[C]]]J BCOMES THE CALL
[ABLABA, [[C]]J[CDJ] WHICH BECOMES
[ABAXXXEXXXQQ<CR]<RR
```

THE CALL:

```
[EC, \\[J BCOMES THE CALL
[ABLABA, >J[CDJ] WHICH BECOMES
```
2.14. DEFINITIONS

2.14.1. EFFECTUATION

The named strings exist in a chronologically ordered list called the environment chain (E-chain). Each element of this list consists of a link to the next older E-chain element, a string, and its associated name. This association is established by a call on the macro name 'DEF'. DEF is called with two and only two arguments, where the first is the name being defined and the second is the defining string. The defining string is customarily enclosed in string quotes to delay interpretation of the control characters contained therein until the macro is called.

2.14.2. SCOPE OF DEFINITION

Definitions are carried out as part of the ordinary process of a macro call and can therefore occur in any context (with certain exceptions) in which any other macro may occur. In particular, they may appear in the argument list of a macro call. Since argument lists are removed when a macro call is complete, this has the effect of a temporary definition.

Macro names are found as a result of a reverse chronological search in the E-chain. This fact, coupled with the
INTERACTIVE MACROGENERATOR       01 MAY 72       2-7

PRESENCE OF TEMPORARY DEFINITIONS, HAS THE EFFECT OF ALLOWING THE
ENVIRONMENT TO BE ALTERED FOR THE EXTENT OF A MACRO CALL, AND WHEN
THE CALL IS COMPLETE, THE ENVIRONMENT IS RESTORED TO ITS ORIGINAL
STATE.

2.1.4.3. EXAMPLES

THE MACROS USED IN THE ABOVE EXAMPLES COULD HAVE BEEN
DEFINED AS FOLLOWS:

1. LDEF,ABA,<XXXB1XXX>1
2. LDEF,WQ,BJ
3. LDEF,C,<ABLABA,61JLCDJ>3
4. LDEF,CU,<QQQ61RRR>3

AN EXAMPLE OF THE USE OF A TEMPORARY DEFINITION IS AS
FOLLOWS:

LADA,AA,LDEF,ABA,<**61**>11

AND THE RESULTANT STRING IS:

**XXX**

THE EFFECT OF A CONDITIONAL SELECTION OF CHARACTERS CAN BE
CAUSED BY THE CALL ON THE DEFINITION:

LDEF;EG,<CBI,LDEF,G1,G4,LDEF,G2,G3J>1

HERE, A CALL ON MACRO 'EG' WITH FOUR ARGUMENTS CAUSES THE THIRD TO
BE EVALUATED IF THE FIRST TWO ARE IDENTICAL. IF THEY ARE NOT, THE
FOURTH ARGUMENT IS EVALUATED. THIS MACRO PERFORMS ITS FUNCTION BY
TEMPORARILY DEFINING THE FIRST ARGUMENT TO BE THE FOURTH AND THE
SECOND TO BE THE THIRD. IF THE FIRST AND THE SECOND ARGUMENTS ARE
IDENTICAL, THEN THE EFFECT IS TO OVERRIDE THE FIRST DEFINITION.
THE MACRO BODY IS THEN 'LDI; LDEF, 61,64JLDEF, 61,63JL'. A CALL ON
THE NAME BEING TEMPORARILY DEFINED WILL TAKE THE LATEST DEFINITION
AND EXPAND IT.

2.2. MACHINE MACROS

THE VARIOUS MACHINE CODE MACROS PRESENTED BELOW PERFORM A
VARIETY OF FUNCTIONS THAT CANNOT BE PERFORMED BY COMBINATIONS OF
OTHER MACROS. IN THIS SENSE, THEY FORM A PRIMITIVE SET OF
ELEMENTARY FUNCTIONS WHICH IS BY NO MEANS COMPLETE. CONTAINED IN
A LATER SECTION OF THIS DOCUMENT IS A SET OF POSSIBLE ADDITIONS
THAT MIGHT BE INCORPORATED INTO LATER VERSIONS OF THE PROCESSOR
UNDER DISCUSSION. THE MACHINE CODE MACROS DESCRIBED BELOW
REPRESENT WHAT IS CONSIDERED TO BE SUFFICIENT TO ATTACK A LARGE
CLASS OF STRING GENERATIVE PROBLEMS.

THE MACHINE MACROS PRESENTED BELOW ARE DIVIDED INTO FOUR
CLASSES: CONTROL, ARITHMETIC, I/O, AND DIAGNOSTIC. EACH CLASS IS
DESCRIBED IN DETAIL AS FOLLOWS:
2.2.1. CONTROL MACROS

THE CONTROL MACROS ARE CHARACTERIZED BY PROVIDING THE ABILITY TO PERFORM MANIPULATIONS ON THE EVALUATION STACK IN ORDER TO ACHIEVE THE DESIRED RESULTS. CERTAIN OF THESE MACROS WILL REQUIRE NUMERIC ARGUMENTS THAT MUST BE PROVIDED IN THEIR BINARY REPRESENTATION RATHER THAN IN NUMERIC CHARACTERS. (SEE BIN AND DEC, 2.2.2.2, AND 2.2.2.3)

2.2.1.1. DEF

DEF HAS ALREADY BEEN SUBSTANTIALLY DESCRIBED IN SECTION 2.1.4 AS IT FITS INTO THE LARGER CONTEXT OF MACRO EVALUATION. IT IS AGAIN DESCRIBED HERE IN ANOTHER LIGHT AND FOR THE SAKE OF COMPLETENESS.

IT IS DEF'S FUNCTION TO ESTABLISH THE ASSOCIATION BETWEEN THE NAME OF A MACRO AND THE STRING THAT DEFINES IT. THIS ASSOCIATION IS MAINTAINED IN A CHRONOLOGICALLY ORDERED, LINEAR LIST CALLED THE E-CHAIN.

THE REQUIRED FUNCTION IS PERFORMED WHEN THE CALL

[DEF, NAME,<BODY>]

IS SCANNED IN THE INPUT STREAM. DEF CAUSES NO CHARACTERS TO BE INSERTED IN THE OUTPUT STREAM. INSTEAD, CHARACTERS ARE INSERTED UNTO THE EVALUATION STACK. SINCE THE ITEM PLACED ON THE
EVALUATION STACK CONSISTS OF INTERNAL CONTROL INFORMATION, THE USER MUST TAKE CARE TO PLACE THE DEFINITION WHERE NO ATTEMPT TO ACCESS ITS CHARACTERS WILL TAKE PLACE EXCEPT BY INVOCATION OF THE MACRO NAME. THIS IMPLIES THAT A DEFINITION MAY APPEAR IN THE SAME CONTEXT AS ANY OTHER MACRO CALL WITH THE FOLLOWING RESTRICTIONS:

1. IF A DEFINITION OCCURS IN THE ARGUMENT LIST OF A MACRO CALL, THAT ARGUMENT MUST NOT BE REFERRED TO AT ANY LEVEL OF MACRO EVALUATION. IN PARTICULAR, DEFINITIONS MAY NOT APPEAR IN ANY PART OF THE MACRO NAME (SEE 3.4.3.4).

2. IF A DEFINITION APPEARS IN THE DEFINING STRING OF A MACRO, THAT MACRO IS SUBJECT TO THE SAME RESTRICTIONS AS DEF UNLESS IT IS A TEMPORARY DEFINITION (SEE 3.2.4).

3. A CALL ON DEF MUST BE COMPOSED OF EXACTLY TWO ARGUMENTS, NEITHER OF WHICH MAY CONTAIN AN EVALUATED DEF (SINCE BOTH ARE REFERENCED, VIOLATING RESTRICTION 1).

THE DEFINITION ESTABLISHED BY DEF CAN EITHER BE TEMPORARY OR PERMANENT. IF THE INTERNAL CONTROL INFORMATION IS GENERATED AT A TIME WHEN OTHER MACRO CALLS WOULD HAVE THEIR RESULTS IMMEDIATELY INSERTED INTO THE OUTPUT STREAM, THEN THAT DEFINITION WILL SUPERCEDE ALL PREVIOUS DEFINITIONS OF THE SAME NAME AND WILL BE PERMANENT. IF THE CONTROL INFORMATION IS GENERATED AS THE RESULT
OF EVALUATING AN ARGUMENT LIST OF A MACRO CALL, THEN THE
DEFINITION IS TEMPORARY AND WILL SUPERCEDE ALL PREVIOUS
DEFINITIONS OF THE SAME NAME UNTIL THE MACRO CALL IN WHOSE LIST
THE DEF APPEARS IS COMPLETED. AT THIS TIME, ALL EMBEDDED
DEFINITIONS ARE UNLINKED AND THE ARGUMENT LIST IS REMOVED FROM THE
STACK. WHEN THIS IS DONE, PREVIOUSLY SUPERCEDED DEFINITIONS ARE
RESTORED TO THEIR ORIGINAL STATE.

2.2.1.2. DU

THE DU MACRO PROVIDES THE ABILITY TO REPEATITIVELY SCAN A
STRING OF CHARACTERS. SINCE A REPEAT COUNT OF ZERO WILL CAUSE NO
SCAN TO TAKE PLACE AND A REPEAT COUNT OF ONE WILL CAUSE EXACTLY
ONE SCAN OF THE STRING TO TAKE PLACE, AN EASY METHOD OF
CONDITIONALLY ScANNING A STRING IS PROVIDED.

THE DU MACRO IS CALLED AS FOLLOWS:

[DU:COUNT:<BODY>]

WHERE "COUNT" IS A BINARY VALUE INDICATING THE NUMBER OF TIMES A
SCAN OF "BODY" IS REQUIRED. IF "BODY" IS TO BE EVALUATED IN
THE COURSE OF THE EVALUATION OF THE DU MACRO, IT IS CUSTOMARILY
ENCLOSED IN STRING QUOTES TO DELAY EVALUATION UNTIL THEN. IF
REPLICATION OF THE RESULTS OF A STRING EVALUATION IS ALL THAT IS
REQUIRED, EXECUTION TIME CAN BE CONSERVED BY NOT ENCLOSING
"BODY" IN STRING QUOTES.
The body of the DO macro may contain requests to evaluate any macro, including a nested DO. In this case, proper use of string quotes can play an important role in helping to achieve desired results.

The DO macro provides the user a means of scanning a string that has been generated in the course of evaluating the input stream without first going through the process of associating that string with a name and then invoking the name in a macro call.

Even though the body is not scanned as a result of an ordinary macro call, the symbol '61' has its usual meaning: that is, it causes insertion of the first argument of the macro call that led to the scan of the string under evaluation. In this case, it provides the binary value of the index of the DO macro as it is incremented from one to 'COUNT'. The other symbols '62', '63', ..., '6n' are null in value, and 'DO' must not be used since it has been altered during the course of iteration in order to provide the necessary looping and indexing control.

As an example, it will be noted that the call:

`LUO, LBIN, 2J, <CLDO, LBIN, 3J, <EDO, 61, *JLCRJ>3>>`

where 'JLCRJ' evaluates to a carriage return and where 'LBIN, 11' calculates the binary value of '11' from its decimal integer representation (see 2.2.2.2), yields:

```
2.2.1.3. BREAK

The ability to exit a loop before the count is exhausted is desirable in many cases. This ability is afforded by use of the break macro. It will also cause termination of the scan of any string under evaluation except the input stream itself.

Break is called as follows:

LBREAK

where the break will be performed unconditionally. If a conditional break is desired, the call:

LBREAK, VALUE

will cause a break if 'VALUE' is not less than or equal to a binary zero (in the logical operators provided in 2.2.2.1, the value 1 is considered to represent a value of 'true' and 0 a value of 'false'. Hence, a break can be generated if some condition is determined to exist).
BREAK WILL CAUSE THE TERMINATION OF ONLY THE CURRENT LEVEL
OF SCAN. THE TERMINATION OF HIGHER LEVELS OF SCAN MUST BE
ACCOMPLISHED BY THE PROVISION OF A BREAK CALL AT THE HIGHER LEVEL.

FOR EXAMPLE, THE CALL:

LDU,[RIN,5],<LD0,LIBIN,6>,<ALBREAKJ>]

WILL RESULT IN THE FOLLOWING BEING GENERATED:

XXXXX

2.2.1.4. UPDATE

A CALL ON DEF PLACES A NEW MEMBER OF THE ENVIRONMENT CHAIN
ON THE EVALUATION STACK. THIS ACTION CAN CAUSE THE WASTEFUL
UTILIZATION OF STACK SPACE WHEN THE SOLE PURPOSE OF THE DEF IS TO
CAUSE A REPLACEMENT OF THE BODY OF A CURRENTLY DEFINED STRING.
ADDITIONALLY, RESTRICTIONS ARE PLACED ON WHERE THE DEF CAN APPEAR.
TO ALLEVIATE THESE RESTRICTIONS, THE DESIRED FUNCTION IS ONE THAT
IS ABLE TO DIRECTLY MODIFY THE DEFINING STRING OF AN E-CHAIN
ENTRY. THIS FUNCTION IS PROVIDED BY UPDATE.

THE CALL:

UPDATE,NAME,<NEW BODY>]

WILL CAUSE THE ENVIRONMENT CHAIN TO BE SEARCHED IN REVERSE
CHRONOLOGICAL ORDER FOR 'NAME'. WHEN IT IS FOUND, THE DEFINING
STRING THAT PREVIOUSLY EXISTED WILL BE REPLACED BY 'NEW BODY', PROVIDED THAT IT IS LESS THAN OR EQUAL IN LENGTH TO THE STRING ORIGINALLY ESTABLISHED AS THE DEFINING STRING BY A CALL ON Def.

UPDATE DOES NOT CAUSE GENERATION OF ANY OUTPUT CHARACTERS AND THEREFORE MAY BE CALLED IN ANY CONTEXT WHATSOEVER. OF COURSE, IF AN UPDATE IS MADE TO A TEMPORARILY DEFINED STRING, THE RESULTS OF THE UPDATE WILL BE REMOVED WHEN THE DEFINITION IS REMOVED. IF THE 'NEW BODY' IS TOO LONG, OR IF THE 'NAME' IS NOT FOUND, AN ERROR IS DIAGNOSED.

2.2.1.5. VAL

IN THE ENVIRONMENT CHAIN ARE MAINTAINED NAME-VALUE PAIRS FOR EACH MACRO DEFINITION. THE VALUE ASSOCIATED WITH A NAME IS SCANNED FOR EVALUATION WHEN THE NAME IS ENCLOSED IN THE OPEN MACRO (L) AND CLOSE MACRO (J) CHARACTERS. THE VALUE ASSOCIATED WITH A NAME IS MERELY COPIED TO THE OUTPUT STREAM WHEN IT APPEARS AS THE ARGUMENT TO VAL. THE VAL OF MACHINE CODE, MACROS IS NOT PERMITTED.

FOR EXAMPLE, IF 'X' IS DEFINED TO BE 'ABCD', THEN 'LVAL,X' IS 'ABCD'.

2.2.1.6. *(PARTIAL ARGUMENTS)*

THE ABILITY TO REFER TO ARGUMENTS IS PROVIDED BY THE SYMBOLS '60', '61', ..., '69'. THE ABILITY TO TRANSCEND LEVELS OF NESTING IS PROVIDED BY PREFIXING THE SYMBOLS BY AN ADDITIONAL '6'.
For each level that is to be transcended, each argument must be referred to in its entirety with this method and no more than nine arguments may be referenced by means of this mechanism. The partial arguments macro (**) overcomes each of these problems and provides additional capabilities.

The ** macro is called with at most four binary valued arguments giving different results in each case. The possibilities are best presented in the table below:

<table>
<thead>
<tr>
<th>Call</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ( \cdot ) J</td>
<td>The number of arguments provided to the macro under evaluation is returned as a binary value. The name is not included in the count.</td>
</tr>
<tr>
<td>2 ( \cdot ) ( \cdot ) ( \cdot ) ARGJ</td>
<td>The argument provided to the macro under evaluation, whose index is the binary value 'ARG', is located; and its character length is returned as a binary value. If the argument does not exist (as opposed to being null) an error is noted.</td>
</tr>
</tbody>
</table>
3. \( L^\ast, \text{ARG}, FC, NC \)  
   The argument provided to the macro under evaluation, whose index is the binary value 'ARG', is located and 'NC' of its characters are copied starting at 'FC'. If 'FC' is zero, the argument length is returned as in case 2. If 'NC' is zero, no characters are output. If 'FC' or 'FC+2*NC'-1 are greater than the total length of the argument, an error is indicated.

4. \( L^\ast, \ldots, LVL \)  
   \( L^\ast, \text{ARG}, \ldots, LVL \)  
   \( L^\ast, \text{ARG}, FC, NC, LVL \)  
   The three cases to the left perform exactly as above except that 'LVL' levels of macro call nesting are transcended before the rest of the call is evaluated. An 'LVL' equal to zero has the same effect as if 'LVL' had not been specified.

The following examples will clarify the use of the "*" macro. Suppose 'A' is defined to be 'L0,ONE,TWO,THREE', and the calls are made as indicated below ("\*" is the line continuation)
CHARACTER DESCRIBED IN 2.3.3; DEC AND BIN ARE DECIMAL TO BINARY AND BINARY TO DECIMAL CONVERSION FUNCTIONS DESCRIBED IN 2.2.2.2 AND 2.2.2.3:

DEFINITION OF 'B' CALL ON 'A' RESULT COMMENTS

10 L0J LDEC,[A]J 3 3 ARGS TO B
20 L0,L0,LBIN,JJ LDEC,[A,X]J 1 1 ARG TO A
30 L0,LBIN,JJ LDEC,[A]J 5 5 CHARS IN 'THREE'
40 L0,LBIN,JJ,
   LBIN,JJ LDEC,[A,XXX]J 3 3 A'S IN XXX
50 L0,LBIN,2JJ; LAJ Wo 2 CHARS OF 'TWO' FROM SECOND LBIN,2JJ,LBIN,2JJ
60 L0,LBIN,JJ; [A,ABCD] ABCD ENTIRE ARG TO LBIN,JJ,L0,LBIN,JJ,
   LBIN,JJ;

2.2.2. ARITHMETIC

THE ARITHMETIC HANDLING MACROS CONSIST OF ROUTINES FOR CONVERTING BETWEEN CHARACTER REPRESENTATIONS OF NUMBERS AND A BINARY REPRESENTATION, AND A MACRO FOR PERFORMING A VARIETY OF BINARY OPERATIONS.

2.2.2.1. BAR
BAR PERFORMS BINARY ARITHMETIC ON TWO BINARY VALUES AND PRODUCES A BINARY OUTPUT. IT IS CALLED AS FOLLOWS:

BAR, OP, ARG1, ARG2

WHERE 'ARG1' AND 'ARG2' ARE THE TWO BINARY INPUTS AND 'OP' IS THE NAME OF THE OPERATION TO BE PERFORMED. THESE ARE LISTED BELOW:

<table>
<thead>
<tr>
<th>OP</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>ARG1-ARG2 IS CALCULATED</td>
</tr>
<tr>
<td>*</td>
<td>ARG1*ARG2 IS CALCULATED</td>
</tr>
<tr>
<td>/</td>
<td>INTEGER DIVISION ARG1/ARG2 IS CALCULATED</td>
</tr>
<tr>
<td>//</td>
<td>ARG1 MODULO ARG2 IS CALCULATED</td>
</tr>
<tr>
<td>GT</td>
<td>=1 IF ARG1 GT ARG2 ELSE 0</td>
</tr>
<tr>
<td>GE</td>
<td>=1 IF ARG1 GE ARG2 ELSE 0</td>
</tr>
<tr>
<td>EQ</td>
<td>=1 IF ARG1 EQ ARG2 ELSE 0</td>
</tr>
<tr>
<td>LT</td>
<td>=1 IF ARG1 LT ARG2 ELSE 0</td>
</tr>
<tr>
<td>LE</td>
<td>=1 IF ARG1 LE ARG2 ELSE 0</td>
</tr>
<tr>
<td>NE</td>
<td>=1 IF ARG1 NE ARG2 ELSE 0</td>
</tr>
<tr>
<td>**</td>
<td>ARG1 IS <em>AND</em>ED WITH ARG2</td>
</tr>
<tr>
<td>++</td>
<td>ARG1 IS <em>OR</em>ED WITH ARG2</td>
</tr>
<tr>
<td>--</td>
<td>ARG1 IS <em>XOR</em>ED WITH ARG2</td>
</tr>
<tr>
<td>*/</td>
<td>ARG1*2**ARG2 IS CALCULATED</td>
</tr>
</tbody>
</table>

WHERE ARG1 IS CONSIDERED TO BE A POSITIVE INTEGER TO BE SHIFTED
The output of BAR is a binary value, which is the result of applying the operation to the two binary values given.

2.2.2.2. BIN

BIN accepts a string of up to 1000 characters and calculates a 36-bit, signed, binary value from the character representation of a signed decimal number. This number consists of an optional sign (+ or -) followed by any number of decimal digits. Any number of blanks may be embedded in the string without affecting the value generated. The calculated value is regarded as a positive value modulo 2^36, which is complemented if it is preceded by a '1'. Integer overflow will be undetected.

The binary value generated conforms to the form required of all binary values: that is, it consists of up to four 9-bit characters which, when concatenated, form the least significant bits of the value. If there are less than four characters in the value, the most significant bits are zero. The value must consist of at least one character, even if it is of value zero. All binary values generated by the arithmetic macros have leading characters of value zero omitted in order to shorten the string.

Due to the kind of binary value that is generated by BIN, it can be used to generate an ASCII character, given its decimal character code value.
2.2.2.3. DEC

DEC performs the conversion of a binary value to a decimal character representation, prefixed by a ' - ' if the value is negative. The string output consists of at least one character (even if it is the character '0') but otherwise consists of no embedded blanks and no leading zeroes.

As with BIN, the form of the binary value can be used in calculating character codes. In this case, an argument of a single ASCII character will yield its decimal character code.

2.2.2.4. OCT

OCT functions exactly the same way as DEC except that a full twelve character, unsigned, octal number is produced. This function is useful in obtaining status bits returned by the FC macro discussed in the next section.

2.2.2.5. Examples

<table>
<thead>
<tr>
<th>CALL</th>
<th>YIELD</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEC,BIN,1000JJ</td>
<td>1000</td>
</tr>
<tr>
<td>DEC,BIN,000000000001JJ</td>
<td>1</td>
</tr>
<tr>
<td>DEC,LABAR++,BIN,1J,LIBIN,2JJJ</td>
<td>3</td>
</tr>
<tr>
<td>LUCT,LABAR++,BIN,1J,BIN,35JJJ</td>
<td>400000000000</td>
</tr>
<tr>
<td>BIN,DEC,XXXXJJ</td>
<td>XXXX</td>
</tr>
</tbody>
</table>
2.2.3. I/O

THE I/O MACROS ALLOW THE USER TO DIRECTLY ACCESS INPUT FROM EITHER THE MAIN SYSTEM INPUT STREAM (I.E., THE TELETYPewriter IN DEMAND MODE OR THE CARD READER IN BATCH MODE) OR FROM ANY PROPERLY DEFINED AND STRUCTURED FILE. THE USER IS ALSO GIVEN THE ABILITY TO DEFINE SUCH FILES AND TRANSFER INFORMATION TO THEM IN A PROPERLY STRUCTURED MANNER. THE MAIN SYSTEM OUTPUT DEVICE (I.E. THE TELETYPewriter IN DEMAND MODE OR THE LINE PRINTER IN BATCH MODE) CAN ALSO BE USED AS THE DESTINATION OF IMAGE TRANSFERS. ALL OF THE FEATURES ARE EXPLAINED BELOW. ALL OF THESE FEATURES ARE HIGHLY DEPENDENT ON THE COMPUTER OPERATING SYSTEM. WHILE THE DISCUSSION BELOW WILL ATTEMPT TO REMAIN SYSTEM INDEPENDENT, IT IS HELPFUL TO HAVE A BASIC FAMILIARITY WITH THE COMPONENTS OF THE EXECUTIVE SYSTEM THAT PERTAIN TO THE SUBJECT UNDER DISCUSSION.

2.2.3.1. FC

FC IS THE FILE CONTROL MACRO. ITS FUNCTION IS TO ALLOW THE USER TO PASS ONE OF THE EXECUTIVE SYSTEM FILE CONTROL IMAGES TO THE SYSTEM FOR EXECUTION. THE OUTPUT OF THIS MACRO IS A BINARY VALUE REPRESENTING THE DIAGNOSTIC BIT SETTINGS RETURNED BY THE OPERATING SYSTEM. THESE FACILITIES ARE QUITE EXTENSIVE AND ARE THOROUGHLY DOCUMENTED IN THE EXEC 8 OPERATING SYSTEM MANUALS.

FC IS CALLED AS FOLLOWS:
INTERACTIVE MACROGENERATOR 01 MAY 72 2-23

[CFC,<EXEC 8 CONTROL IMAGE >]

Care should be taken to ensure that the control image is valid, since the processor's execution will be immediately terminated if it is not. If the control image is valid, the results of its execution can be discovered via the binary value that is returned by FC. It is customary to enclose the control statement in string quotes to avoid conflicts in the usage of certain characters. It is also customary to call FC as an argument to UCL in order that the diagnostic bit settings returned can be easily interpreted.

Three defs involving FC are provided below: These will facilitate the file control by naming functions that are more familiar to the user. These are:

1. DEF:OPEN,<LNS,,COCT,CFC,<WASG,X >61 . JJJ>
2. DEF:CLOSE,<LWS,,COCT,CFC,<WRKPT >61 . JJJ>
3. DEF:REWIND,<LNS,,COCT,CFC,<WRKPT >61 . JJJ>

These functions are called as follows:

1. OPEN:FILENAME
2. CLOSE:FILENAME
3. REWIND:FILENAME

Where 'FILENAME' is a maximum of 12 characters chosen from the set A,...,Z,i,u,...,9,S,:. A file must be opened to be used by the I/O macros. Closing a file releases it and requires that it be
RE-OPENED FOR LATER USE BY THE I/O MACROS. WHEN A FILE IS CLOSED, THE INFORMATION IT CONTAINS WILL BE LOST UNLESS IT IS A CATALOGUED FILE. WHEN A FILE IS REMOVED THE NEXT IMAGE TRANSFER TO OR FROM THE FILE TAKES PLACE FROM ITS BEGINNING.

CALLS ON THE 3 MACROS DEFINED ABOVE CAUSE THE DIRECT OUTPUT OF THE DIAGNOSTIC BIT SETTINGS IN OCTAL TO THE MAIN SYSTEM OUTPUT DEVICE BUT OTHERWISE CAUSE NO CHARACTERS TO BE GENERATED, AND THUS THEY CAN BE CALLED IN ANY CONTEXT.

NS

NS PROVIDES THE ABILITY TO INPUT A STRING, EITHER FROM THE MAIN SYSTEM INPUT DEVICE OR FROM A DESIGNATED FILE. THE CALL ON NS IS AS FOLLOWS;

1. LRSJ
2. LRSJ
3. LRSJ,FILENAME

WHERE 'FILENAME' IS A VALID FILE NAME THAT HAS SUCCESSFULLY BEEN OPENED PRIOR TO THIS CALL. IF FORMS 1 AND 2 ARE USED, INPUT IS FROM THE MAIN SYSTEM INPUT DEVICE. IF FORM 3 IS USED INSTEAD, THE NEXT IMAGE IN THE FILE NAMED IS READ. THE IMAGES READ ARE VARIABLE IN LENGTH AND CONSIST OF INTEGER MULTIPLES OF FOUR CHARACTERS WITH NO TRAILING BLANKS, EXCEPT THOSE NECESSARY TO LENGTHEN THE IMAGE TO CONTAIN THE PROPER MULTIPLICITY OF
CHARACTERS. AN IMAGE MAY CONSIST OF ZERO CHARACTERS. WHEN THE LAST IMAGE OF A FILE IS READ, SUBSEQUENT READS ON THE SAME FILE CAUSES NULL IMAGES TO BE TRANSFERRED. WHENEVER AN END OF FILE CONDITION IS DETECTED, A SWITCH IS SET. THE SETTING OF THIS SWITCH IS RETRIEVED BY A CALL ON 'EUF'. THE SETTING WILL BE:

1) A BINARY ZERO - IF THE LAST RS ON ANY FILE WAS SUCCESSFUL; OR

2) A BINARY ONE - IF THE LAST RS ON ANY FILE WAS NOT SUCCESSFUL. THIS SWITCH IS ONLY CHANGED WHEN RS IS CALLED.

WHEN AN IMAGE HAS BEEN SUCCESSFULLY READ BY RS, IT IS INSERTED INTO THE OUTPUT STREAM OR ONTO THE STACK, WHICHEVER IS APPLICABLE.

2.2.3.3. nS

THE MACRO nS CAUSES THE TRANSFER OF ONE OR MORE IMAGES TO THE DESIGNATED DESTINATION. IT IS CALLED AS FOLLOWS:

1. LWS, IMAGE
2. LWS, FILENAME, IMAGE

WHERE 'FILENAME' IS A VALID FILE NAME WHICH HAS BEEN SUCCESSFULLY OPENED. IF FORM 1 IS USED, THE DESTINATION IS THE PRIMARY SYSTEM OUTPUT DEVICE. IF FORM 2 IS USED, THE IMAGE IS TRANSFERRED TO THE FILE NAMED.
"S causes no characters to be generated and hence can be called in any context.

"S interprets the ']' as the end-of-image character and will split images containing this character unto separate lines. Images will also be split among multiple lines if they will not fit on one line. Line length is 72 characters in demand mode and 132 in batch mode.

2.2.4. Diagnostic

The diagnostic macros are for the purpose of giving the user the ability to dump parts of the stack in order to help discover the source of error diagnostics. They are called by the error detection routines whenever an error occurs. Since none of the diagnostic macros transmit any characters to the output stream, they may be called in any context.

2.2.4.1. DBUG

DBUG will cause the levels of macro calls to be traced and their argument lists dumped. The trace will proceed from the most deeply nested call to the outermost call.

2.2.4.2. LCHAIN
ECHAIN WILL CAUSE A DUMP OF THE ENVIRONMENT CHAIN, STARTING WITH THE YOUNGEST MEMBER AND PROCEEDING TO THE OLDEST. MACHINE LANGUAGE MACROS ARE INDICATED AS SUCH BY THE STRING ***ASH***.

2.3. TEXT HANDLING

IN ORDER TO OPERATE ON CHARACTERS IN AN IMAGE-ORIENTED OPERATING SYSTEM, IT IS NECESSARY TO SIMULATE THE ATTRIBUTES OF CERTAIN CARRIAGE CONTROL CHARACTERS. THE EFFECTS OF THIS SIMULATION IS DESCRIBED IN THE PARAGRAPHS THAT FOLLOW.

2.3.1. INPUT IMAGES

IMAGES ENTER THE MACROGENERATOR IN VARYING LENGTHS. THESE IMAGES ARE SHORTENED TO CONTAIN NO MORE THAN THREE TRAILING BLANKS. IN ORDER TO PRESERVE THE END-OF-IMAGE PROPERTY OF THE LAST NON-BLANK CHARACTER OF AN IMAGE, IT IS FOLLOWED BY THE END-OF-LINE CHARACTER ("?"). THIS CHARACTER MAY ALSO BE EMBODIED IN THE IMAGE, GIVING THE EFFECT OF MULTIPLE IMAGES WHEN THERE WAS ONLY ONE TRANSFERRED. THESE COMBINED CONDITIONS IMPLY THAT ALL TRAILING BLANKS IN AN IMAGE ARE LOST, AND WILL NOT BE INCLUDED IN THE INPUT STREAM.
2.3.2. CONTINUATION

IMAGES THAT MUST SPAN MULTIPLE INPUT IMAGES CAN BE
GENERATED BY USE OF THE CONTINUATION CHARACTER (*!*). THIS
CHARACTER CAUSES THE REMAINDER OF AN INPUT IMAGE TO BE IGNORED,
INCLUDING THE END-OF-LINE CHARACTER.

2.3.3. OUTPUT IMAGES

CHARACTERS ARE TRANSFERRED TO THE OUTPUT STREAM AS THEY ARE
GENERATED. THEY ARE COLLECTED FROM THE OUTPUT STREAM AND
CONVERTED INTO AN IMAGE WHENEVER THE END-OF-LINE CHARACTER APPEARS
IN THE OUTPUT STREAM OR WHENEVER THE IMAGE REACHES THE MAXIMUM
ALLOWABLE LENGTH (72 CHARACTER IN DEMAND MODE OR 132 IN BATCH
MODE).

2.4. PROGRAM INVOCATION

THE INTERACTIVE MACROGENERATOR CONFORMS TO THE STANDARD
UNIVAC 1100 PROCESSOR CONVENTIONS. IT IS EXECUTED WHEN THE
FOLLOWING CONTROL CARD OCCURS IN A RUNSTREAM:
WIMAGE OPTIONS SPEC1, SPEC2, SPEC3

WHERE 'SPEC1' NAMES THE INPUT STREAM, 'SPEC2' NAMES THE DESTINATION OF THE OUTPUT STREAM, AND 'SPEC3' NAMES THE UPDATED VERSION OF THE INPUT STREAM. ALL THREE SPECIFICATIONS DESIGNATE PROGRAM FILE ELEMENTS.

2.4.1 PROGRAM OPTIONS

THE OPTIONS ARE SPECIFIED BY A KEY LETTER FOLLOWING THE COMMA AFTER THE PROGRAM NAME. MULTIPLE OPTIONS MAY BE SPECIFIED IN ANY ORDER. THEY ARE INTERPRETED AS FOLLOWS:

<table>
<thead>
<tr>
<th>OPTION</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>EXECUTE THE PROCESSOR AS IF IN BATCH MODE.</td>
</tr>
<tr>
<td>D</td>
<td>THE STACK IS NOT DUMPED WHEN A DIAGNOSTIC MESSAGE IS GIVEN FOR AN ERROR. FURTHERMORE, THE ERROR LIMIT OF 5 IS RESET TO BE UNLIMITED.</td>
</tr>
<tr>
<td>E</td>
<td>THE ENVIRONMENT CHAIN IS TO BE DUMPED WHENEVER THE EVALUATION CHAIN IS DUMPED.</td>
</tr>
</tbody>
</table>
THE INPUT STREAM IMAGES FOLLOW
THE 'WIMAGE' CARD AND ARE TO BE
RETAINED IN THE DESTINATION
SPECIFIED IN 'SPECI'.

THE MOST COMPREHENSIVE LISTING
POSSIBLE IS PRODUCED.

THE INPUT STREAM IS NOT TO BE
PRINTED.

THE OUTPUT STREAM IS TO BE
PRINTED.

THE INPUT STREAM SPECIFIED IN
'SPECI' IS TO BE UPDATED.

CORRECTION IMAGES ARE TO BE
LISTED.

ALL ERRORS CAUSE PROGRAM
TERMINATION, NO DIAGNOSTIC
STACK DUMPS WILL BE PERFORMED.
THE INPUT AND OUTPUT STREAMS
WILL BE LOST.
2.4.2. INPUT STREAM CORRECTION

When the input stream exists in the program file element named in 'SPECS', correction images may be merged with the input stream to produce an updated input stream. The images are of two kinds: whole image corrections, and partial image corrections.

2.4.2.1. WHOLE IMAGE CORRECTIONS

Two forms of whole line correction exist: insertions and replacements. An insertion is performed when the image 'N' occurs among the correction images. The images that follow the 'N' card are inserted following the N-th image of the source element. This process of insertion continues until an image with an 'M' as the first character is encountered. The new image count designator is then re-evaluated. When the image 'N,M' occurs among the correction images, images numbered 'N' through 'M' are deleted. Insertions then proceed as in the 'N' case.

2.4.2.2. PARTIAL IMAGE CORRECTIONS

Partial corrections allow the changing of selected portions of an image in the source element. A partial correction is made when the image count designator 'N,M' appears in the correction stream. A partial correction specification card is then expected for each image in the specified range in the source element.
THESE PARTIAL CORRECTION SPECIFICATION CARDS ARE ONE OF FOUR FORMS:

1. /OLD STRING/NEW STRING/  
The first occurrence of 'OLD STRING' found in the source element image is replaced with 'NEW STRING'. The rest of the line is shifted according to the new length. Any character may be used in place of '/'.

2. /OLD STRING/NEW STRING/  
The first occurrence of 'OLD STRING' is replaced with 'NEW STRING'. The remainder of the image is blanked. Any character may be used in place of '/'.

3. FC, LC/NEW STRING/  
The characters occupying positions 'FC' through 'LC' are replaced by 'NEW STRING'. If the lengths are not equal, the rest of the line is shifted accordingly. Any character except a digit
4. FC/NEW STRING

'NEW STRING' IS INSERTED
STARTING AT POSITION 'FC'.
THE REST OF THE IMAGE IS
BLANKED.

2.4.3. EXAMPLES

EXAMPLES OF SOURCE ELEMENTS THAT ILLUSTRATE VARIOUS
FUNCTIONS OF THE INTERACTIVE MACROGENERATOR ARE ILLUSTRATED BELOW.
THESE INCLUDE FACILITIES AND CONCEPTS USED IN CONTROLLING
RECURSION, ITERATION, AND THE EXTERNAL EVALUATION OF IMAGE FILES.

2.4.4. RECURSION

THE TRADITIONAL EXAMPLE OF RECURSION IS THE FACTORIAL
FUNCTION. THIS IS DEFINED WITH THE FOLLOWING DEFS:

1. LUEF,-,<LDEC,LBAR,-,LBIN,61J,EBIN,62JJJ>]
2. LUEF,*,<LDEC,LBAR,*,EBIN,61J,EBIN,62JJJ>]
3. LUEF,EQ,<LBAR,LQ,EBIN,61J,EBIN,62JJJ>]
4. LUEF,NE,<LBAR,NE,EBIN,61J,EBIN,62JJJ>]

OR A COMMA MAY BE USED IN
PLACE OF ‘/’.
These macros function as follows: \texttt{NE} and \texttt{EQ} are defined to facilitate control of recursion. \texttt{FACT} then evaluates the factorial by first testing the argument against 1. If the argument is 1, the count for the \texttt{DO} will be zero and the body will not be performed. For the same reason, \texttt{BREAK} will not stop the evaluation. The output will then be the following 1. If the argument is not 1, the body of the \texttt{DO} is evaluated to compute the product of the argument and the factorial of the argument after being decremented by one. The \texttt{BREAK} is performed and the 1 following it is not output as a result of the evaluation.

2.4.5. Iteration

The same function, \texttt{FACT}, can also be evaluated by iteration. With the same \texttt{DEFS} 1 through 4 as above, \texttt{FACT} can be defined as follows:

\begin{verbatim}
LDEF,FACT,<LDO,\texttt{LABIN},\texttt{IJ},LDO,\texttt{LABIN},\texttt{IJ},; \texttt{LABC}>,\texttt{61},J,\texttt{LABD}>,\texttt{81},J,; \texttt{LABC}>,\texttt{61},J,\texttt{LABD}>,\texttt{81},J,; \texttt{LABC}>,\texttt{61},J,\texttt{LABD}>,\texttt{81},J,;
\end{verbatim}

This macro functions by generating a string of characters
THAT WILL REPRESENT THE PRODUCT OF ALL OF THE TERMS AND WILL THEN EVALUATE IT.

2.4.3. EXTERNAL EVALUATION

EXTERNAL EVALUATION CONSISTS OF THE INPUT OF IMAGES FROM A FILE AND THEN EVALUATING THEM. THIS IS USEFUL IF IT IS DESIRED TO KEEP A FILE OF MACRO DEFINITIONS THAT ANY EXECUTION OF THE INTERACTIVE MACROGENERATOR MAY REFER TO AT ANY TIME. BY BUILDING LIBRARIES OF FUNCTIONS, THE USER CAN EASILY BUILD AN EXTENSION OF THIS PROCESSOR FROM ITS BASIC COMPONENTS. THE EXTERNAL EVALUATION IS PERFORMED AS FOLLOWS:

1. LUPEN,EXTERNAL;

2. LUEF, EVAL, <LOU, LBU, Y99, <LCRKEK, LEUF, LDU, LB14, 11;

3. LEVAL, EXTERNAL;

WHERE OPEN IS DEFINED AS IN 2.2.3.1. EVALUATION OF ALL IMAGES IN THE SPECIFIED FILE IS PERFORMED WHEN CALL 3* IS ENCOUNTERED.

CALLS 2* AND 3* CAN BE USED TO BOOTSTRAP A SET OF DEFINITIONS, IF THE FILE IS ALREADY OPENED. IN GENERAL, THERE ARE NO RESTRICTIONS ON THE RELATIONSHIP BETWEEN INPUT STRAIN IMAGES AND MACRO CALLS.

THIS IS ALSO TRUE FOR EVAL AS DEFINED ABOVE.
3. STRUCTURE


3.1. NOTATION

IN ORDER TO DISCUSS STACK OPERATION, A NOTATION IS REQUIRED TO ENABLE THE STACK OPERATIONS TO BE CLEARLY DEFINED. THE FOLLOWING NOTATION SHOULD SUFFICE:

1. THE STACK IS COMPOSED OF ENTRIES WHICH ARE SEPARATED BY SEMICOLONS. AN ENTRY IS VARIABLE IN LENGTH AND MAY BE OF ANY MAGNITUDE PROVIDED THAT THERE EXISTS SPACE ON THE STACK TO HOLD IT.

2. AN ENTRY IS COMPOSED OF CONTROL CELLS AND DATA CELLS. THESE ARE ORGANIZED TO FORM ENTRY ITEMS WITHIN THE ENTRY. ENTRY ITEMS ARE SEPARATED BY COMMAS.
3. A DATA CELL REPRESENTS THE SMALLEST UNIT OF INFORMATION AND IS OF SUFFICIENT SIZE TO CONTAIN ONE CHARACTER.

4. A CONTROL CELL IS USED TO CONTAIN POINTERS TO OTHER CELLS OR TO CONTAIN THE LENGTH OF A STRING OF CELLS. IT IS AN INTEGER MULTIPLE OF THE DATA CELL IN SIZE (IN THE IMPLEMENTATION UNDER DISCUSSION THE RATIO IS 2 TO 1). IT IS ENCLOSED IN PARENTHESES.

5. AN ENTRY ITEM IS INDICATED BY THE STRING "(LI) TEXT". THE CELL "(LI)" IS THE CONTROL CELL FOR THE I-TH ENTRY ITEM IN THE ENTRY. IT CONTAINS THE TOTAL LENGTH IN CELLS THAT THE ENTRY ITEM OCCUPIES ON THE STACK, INCLUDING THE LENGTH OF "(LI)". "TEXT" IS THE TEXT THAT THAT ENTRY ITEM IS TO CONTAIN. THE ENTRY ITEM "(NK)" IS OF SPECIAL SIGNIFICANCE; IT IS KNOWN AS "MARKER" AND SIGNIFIES THE END OF AN ENTRY. WHEN ENCOUNTERED IN A SCAN, IT SIGNIFIES THE END OF A MACRO DEFINING STRING.

3.2. STACK ENTRIES

THREE TYPES OF ENTRIES CAN EXIST ON STACK. THESE ARE THE L-CHAIN, P-CHAIN, AND F-CHAIN ENTRIES. THEIR FORMATS ARE DEFINED BELOW.
3.2.1. E-CHAIN ENTRY

The E-CHAIN entries contain the definitions that are made by calls on DEF and that were predefined to the processor. These appear on the stack as follows:

```
<------EL OF LATER E-CHAIN ENTRIES
\v
\(EL),(LU)NAME,(LI)BODY,(MK)\;
\v
<MORE E-CHAIN----->
```

In this definition 'EL' represents the link to the next older E-CHAIN entry and 'EL' is pointed to by the 'EL' of the next younger E-CHAIN entry. 'NAME' is the name being defined and 'BODY' defines it. The 'EL' of the youngest E-CHAIN entry is pointed to by 'ED', the list entry point indicator for the E-CHAIN.

3.2.2. P-CHAIN ENTRY

The P-CHAIN entries hold the argument list for a macro call for the duration of the evaluation of the definition that it invoked. It appears on the stack as follows:
PL OF LATER P-CHAIN ENTRIES

\( \text{PL} \), \( \text{CP} \), \( \text{NAME} \), \( \text{ARG1} \), \ldots, \( \text{ARGN} \), \( \text{MK} \);

\( \text{NEXT} \ \text{PL} \)

'PL' IS THE P-CHAIN LINK. IT POINTS TO THE CALL WHOSE
EVALUATION LED TO THE CALL REPRESENTED BY THIS ENTRY AND IS
POINTED TO BY CALLS MADE BY THE EVALUATION OF THE CALL REPRESENTED
BY THIS ENTRY. THIS CHAIN ALLOWS ACCESS TO THE VARIOUS LEVELS OF
NESTING. THE 'PL' OF THE YOUNGEST P-CHAIN ENTRY IS POINTED TO BY
'PU', THE LIST ENTRY POINT INDICATOR FOR THE P-CHAIN.

'CP' IS A POINTER TO THE STRING BEING EVALUATED WHEN THIS
CALL WAS MADE. WHEN THIS CALL IS COMPLETE, THIS POINTER WILL
DETERMINE WHERE THE SCAN IS TO BE RESUMED.

'PHI' CONTAINS THE LENGTH OF THE TOTAL P-CHAIN ENTRY,
INCLUDING ITS OWN LENGTH. WHEN THE CALL IS COMPLETE, THIS VALUE
DETERMINES THE NUMBER OF CELLS THAT ARE TO BE COPIED OVER WHEN THE
P-CHAIN ENTRY IS REMOVED AND STACK COMPACTIFICATION TAKES PLACE.

THE VARIOUS ENTRY ITEMS SHOWN ARE PRECISELY THE ARGUMENTS
PROVIDED IN THE CALL:

\( \text{LNAME}, \text{ARG1}, \ldots, \text{ARGN} \)

3.2.3. P-CHAIN ENTRY
The F-chain entries contain the partially completed argument lists for macros whose arguments are under evaluation. When the close macro (J) character is encountered, the last argument is in place and the F-chain entry becomes a P-chain entry. It appears on the stack as follows:

\[
\begin{array}{c}
\text{PREV} \downarrow \text{FL} \downarrow \text{LATER F-CHAIN ENTRY} \downarrow \text{CH} \downarrow \text{S} \\
\text{V} \downarrow \text{V} \\
\text{(MP)} \downarrow \text{(FL)} \downarrow \text{(U)} \downarrow \text{(L)} \downarrow \text{ARGU} \downarrow \text{(L1)} \downarrow \text{ARGU} \downarrow \cdots \downarrow \text{(LN) ARGU} \downarrow \\
\text{NEXT FL} \downarrow \text{V}
\end{array}
\]

In this entry, 'ARGU' is under construction. 'H' points to the cell that is to contain its length when complete. 'S' points to the next stack cell to receive text.

'FL' is the F-chain link. It points to earlier F-chain entries. It is pointed to by subsequent F-chain entries. If it is a cell in the latest F-chain entry, it is pointed to by 'FU', the list entry point indicator for the F-chain.

Since another F-chain entry under construction may have been interrupted by the current entry, the earlier value of 'H' is saved as 'IMPH' and is restored when conversion of the entry to a P-chain entry takes place. It points to the length cell of the previous entry item being built.

Since the F-chain entry is to eventually become a P-chain entry, an additional control cell is required. This is allocated.
3.2.4 TEMPORARY DEFINITIONS

When a DEF appears in the argument list of the macro call, characters are inserted in the entry item being formed in the F-CHAIN ENTRY. These characters form the data cells and control cells for an E-CHAIN ENTRY. When the F-CHAIN ENTRY is converted into a P-CHAIN ENTRY, this E-CHAIN ENTRY is still present. When the macro call is complete, the P-CHAIN ENTRY is removed and the space that it occupied is covered over by information that followed it on the stack. When this happens, the E-CHAIN ENTRY must first be unlinked, and in so doing, it ceases to exist. It is then destroyed with the rest of the P-CHAIN ENTRY. The restriction disallowing reference to arguments containing DEFS is necessary to prevent the control cells from being scanned.

3.3 EVALUATION CONTROL

It is the purpose of this section to discuss the sequence of operations executed as the MACROGENERATOR evaluates text. Using this information as an overview, the previous material which defined the stack entries will give a much greater meaning and
UNDERSTANDING TO THE DETAILED STACK MANIPULATION PROCEDURES THAT WILL FOLLOW.

3.3.1. SCAN CYCLE

CHARACTERS ENTER THE SCANNING MECHANISM FROM THE INPUT STREAM AND ARE COPIED DIRECTLY TO THE OUTPUT STREAM UNTIL A WARNING CHARACTER IS ENCOUNTERED. THIS INPUT STREAM IS THE CHARACTER STREAM THAT IS PROVIDED AS DATA TO THE MACRO GENERATOR. THE INPUT STREAM CAN ALSO COME FROM STRINGS THAT ARE AVAILABLE FOR SCANNING ON THE STACK, SUCH AS THE BODY OF A MACRO. SIMILARLY, THE OUTPUT STREAM IS USUALLY DIRECTED TO THE OUTPUT DESTINATION ELEMENT NAMED ON THE PROCESSOR CONTROL CARD. THIS STREAM CAN ALSO BE DIRECTED TO THE TOP OF THE STACK, WHICH IS THE CASE WHEN THE ARGUMENTS TO A MACRO CALL ARE BEING EVALUATED. IT IS THIS SWITCHING OF INPUT AND OUTPUT STREAMS THAT IS THE GIST OF THE EVALUATION OF A TEST STREAM.

THE SEQUENCE OF OPERATIONS FOR CONTROLLING THE SWITCHING OF CHARACTERS BEGINS WHEN THE OPEN MACRO CHARACTER (}) IS ENCOUNTERED IN THE INPUT STREAM. THIS INITIALIZES AN F-CHAIN ENTRY AND THE OUTPUT STREAM IS SWITCHED FROM ITS PREVIOUS DESTINATION TO THE TOP OF THE STACK. THE PREVIOUS DESTINATION IS SAVED IN A CONTROL CELL IN THE ENTRY SO THAT IT CAN BE RESTORED WHEN THE ARGUMENT LIST IS
As characters enter from the input stream, other open macro characters may be encountered, causing a nesting of partially created argument lists. As argument separators (,) are encountered, entry items are created from the partial item under construction on the top of the stack. When the close macro character (}) is encountered, the partial item under construction is completed. The transformation from F-chain entry to P-chain entry is ready to take place.

The transformation from F-chain entry to P-chain entry consists of first removing the entry from the F-chain, and placing it as the newest member of the P-chain. The control cell which held the pointer to the previous output stream destination is used to restore this value as the indicator for the current output stream destination pointer (*M*). Now that the F-chain entry belongs to the F-chain no longer, it becomes a viable P-chain entry when the two remaining control cells are set to their proper values. One of these is *PHI*, which will be set to contain the length of the total P-chain entry; the other is *CP*. Since the current input stream source is about to be switched, this indicator is preserved in *CP*. This being done, the P-chain entry is complete.

At this time, the first entry item of the new P-chain entry contains the name of the macro body which is to become the new current input stream. This name is used as the object of a search.
In the E-chain, when the matching name is found, scanning will continue with the first character in the body of the macro definition unless the macro is a machine macro. If this is the case, a transfer is made to the proper routine; otherwise, scanning resumes at a new level. The scan at the new level continues as before.

Two items remain undiscovered; one is the argument extractor (6). This character sets a pointer to the top of the P-chain. If another '6' follows, the pointer is moved to the next level P-chain entry. This process continues until no more argument extractors are present. The digit that follows the series of argument extractors indicates the index of the P-chain entry item of the P-chain entry last pointed to which is to be copied without further evaluation to the current output destination.

The last item to be discussed is the end-of-body marker. This internal character causes the macro call termination sequence to be initiated, which begins by unlinking all E-chain entries that lie within the argument list of the P-chain entry. The current input source is set to its value prior to this call; that is, to 'CP'. The space occupied by the argument list is closed by sliding those characters created after the P-chain entry CP to the first character originally occupied by the first cell of the P-chain entry. The number of cells removed is determined by
"PHI" which holds the length of the P-chain entry. At this point, the characters generated as a result of the macro call are in place exactly as if they had occurred explicitly in the input stream.

3.4. SCAN MODES

3.4.1. COPY-MODE

COPY-MODE occurs when non-warning characters are being processed. Characters are simply passed from the input stream to the output stream. This is affected by the string quotes "<" and ">", and the unwarn character "\".

3.4.1.1. STRING QUOTES

The first string quote encountered in a scan initiates copy mode for all but the unwarn character until a matching, bracketing string quote is found. When this happens, copy-mode terminates. Neither the first quote encountered nor its bracketing quote are copied.

3.4.1.2. UNWARN CHARACTER
WHEN THE UNNAMED CHARACTER IS SCANNED IN ANY CONTEXT, IT IS NEVER COPIED. INSTEAD, IT CAUSES THE CHARACTER FOLLOWING IT TO BE COPIED. THIS OCCURS EVEN IF THE NEXT CHARACTER IS ANOTHER UNNAMED CHARACTER.

3.4.2.0 WARNING CHARACTER MODE

3.4.2.1 ARGUMENT EXTRACTOR (b)

THE ARGUMENT EXTRACTOR CAUSES THE INDICATED ARGUMENT TO BE COPIED TO THE OUTPUT DESTINATION AS NOTED UNDER SECTION 3.3.1.

3.4.2.2 OPEN MACRO (c)

THE FOLLOWING TRANSFORMATION TAKES PLACE TO CREATE AN F-CHAIN ENTRY:

INITIAL STATES

\[ <\text{HP} > <\text{FL} > <\text{S} > \]

\[ V \quad V \quad V \]

\[ \ldots ? \ldots \ldots ? \ldots \ldots \ldots ? \]

FINAL STATE:
3.4.2.4. ARGUMENT SEPARATOR (,)

The following transformation takes place to complete an entry item:

INITIAL STATE:

\[
\begin{align*}
\text{\textless-NEW FL \textless-S} \\
\text{V} \\
\text{\textless-} \text{(HP)} \text{\textless-FL} \text{\textless-(U)} \text{\textless-(U)} \text{\textless-}\text{V} \\
\text{\textless-}\text{V} \\
\end{align*}
\]

FINAL STATE:

\[
\begin{align*}
\text{\textless-HP} \\
\text{\textless-} \text{(U)} \text{\textless-}\text{V} \\
\text{\textless-}\text{V} \\
\end{align*}
\]

3.4.2.4. CLOSE HACKU (J)

The following transformation takes place to convert an F-CHAIN ENTRY INTO A P-CHAIN ENTRY:
INITIAL STATE:

:<=CP  :<=PL  :<=FL  :<=HP  :<=S
V     V     V     V     V
**??**??**??;(HP),(FL),(U),(L0)ARGU,(L1)ARG1,...,(U)??
\[\Delta\]
:<=-------V
EARLIER FL ENTRY<=V

FINAL STATE:

:<=------------------------\[\Delta\]
V
**??**??**??;(PH1),(PL),(CP),(LU)ARGU,...,(LM)ARGU,(MK)??
\[\Delta\]
:<=--------------V
<=-HP

3.4.2.5 MARKER (-U)

THE FOLLOWING TRANSFORMATION TAKES PLACE TO TERMINATE A
MACRO CALL:

INITIAL STATE:

:<=S
V
**??ILP-CHAIN ENTRY?ILRESULT?J?
FINAL STATE:

<-

V

***RESULT??

3.4.2.6. DU MARK

The DU MARK is a character that is placed as the last character in the body of the DU macro call. It serves the same function as Marker: that is, to stop the scan of a string, instead of closing the macro call; however, the iteration count is checked to see if another scan of the DU macro body is to be made.

3.4.3. MACHINE MACROS

This sub-section will discuss the implementation of the machine macros. These are placed in the environment chain in the same way as macros defined by DEF. The one difference is that the control cell that would normally hold the length of the defining string holds instead the index of an entry in a transfer vector. This index is distinguished from the length cell by having its high order bit set. When a jump into the transfer vector occurs, the P-chain entry that caused the call is at the top of the stack and is the youngest entry on the P-chain. In this way, arguments
Provided are available to the machine macro for its use in evaluating the call.

3.4.3.1. Diagnostic (DEBUG, ECHAIN)

The two diagnostic macros are very simple. The DEBUG macro causes each entry on the p-chain to be formatted and printed. The ECHAIN macro causes a similar function to take place on the environment chain. No characters are generated as a result of the call, so no stack manipulation is necessary.

3.4.3.2. I/O Macros (RS, WS, FC, EOF)

FC is primarily an interface with the executive function CSFs. The interface is established by copying the control image provided beyond the top of the stack, left-justified on a word boundary. A character translation is made from 8-bit characters to 6-bit characters and the executive function is performed. A binary value is returned, and this is output as the result stream of the macro call.

WS is accomplished in the same way as FC except that instead of requesting a file control analysis by the system, an image transfer is requested. The first argument is used to specify the image destination as part of the image transfer request. No characters are output as part of the result stream of the macro call.
KS functions by requesting an image from the operating system from the source named in the first argument. The image is then transformed into the characters that are to be output as the result stream of the macro call. If a valid image transfer takes place, the EOF flag is set to zero. If not, it is set to a binary one. When EOF is called, this value becomes its output result stream.

3.4.3.3. Arithmetic Handling Macros (DEC, ULC, BIN, BAR)

The arithmetic handling macros depend on facilities for inputting and outputting binary values. DEC and ULC function by retrieving the binary value from the argument list, converting it to the proper character representation, and copying the representation to the output stream. BIN performs a character scan of the argument and calculates the binary value it represents, which is then output. Finally, BAR performs its function by first retrieving the two binary arguments, the operation indicated is compared with those defined internally, and a transfer takes place to the appropriate routine. Once calculated, the binary value is then output.

3.4.3.4. The Control Macros

The control macros do not function as simply as the other classes. Their operation is more closely tied to the structure of the macro generator. These usually operate by providing a method
TO SLIGHTLY ALTER THE COURSE OF EVALUATION IN ORDER TO ACHIEVE THE DESIRED RESULTS, THESE PERFORM THEIR FUNCTIONS AS FOLLOWS:

1. DEF SERVES TO LINK AN ADDITIONAL ENTRY INTO THE E-CHAIN. WHEN DEF IS CALLED THE TOP OF THE P-CHAIN IS AS FOLLOWS:

;(PHI),(PL),(CP),(LU)'DEF',(LI)'NAME','(LZ)'BODY','(MF);

THE DESIRED FORM IS:

;(EL),(LI)'NAME','(LZ)'BODY','(MA);

THIS IS ACHIEVED BY OVERWRITING PART OF THE ARGUMENT CONTAINING THE CALL NAME 'DEF' WITH THE ENVIRONMENT CHAIN HEADER 'EL'. THAT NEW CONTROL CELL IS THEN SET TO BE THE TOP OF THE E-CHAIN. AT THIS POINT THE CONFIGURATION IS AS FOLLOWS:

;(PHI),(PL),(CP),(LU)??,(EL),(LI)'NAME','(LZ)'BODY','(MA);

'PHI' IS THEN SHORTENED TO CONTAIN THE LENGTH OF ONLY THAT AREA OCCUPIED FROM 'PHI', UP TO, BUT NOT INCLUDING, 'EL'. A TRANSFER IS THEN MADE TO THE FUNCTION THAT HANDLES MARKER AND THE AREA INDICATED BY 'PHI' IS REMOVED. SINCE PART OF THIS FUNCTION IS TO PROPERLY ADJUST THE ENVIRONMENT CHAIN POINTERS, 'EL' WILL BE PROPERLY SET WHEN THE MARKER FUNCTION IS COMPLETE. THE RESULTS ARE THEN AS REQUIRED.

2. UPDATE WILL MODIFY AN EXISTING E-CHAIN ENTRY. THE MACRO
NAME IS SEARCHED FOR IN THE ENVIRONMENT CHAIN. WHEN IT IS FOUND, THE DEFINING BODY IS REPLACED WITH THE BODY GIVEN IN THE CALL ON UPDATE IF THE LENGTH IS LESS THAN OR EQUAL TO THE LENGTH OF THE ORIGINAL DEFINING STRING. MARKER IS MADE THE LAST CHARACTER IN THE STRING SO THAT SUBSEQUENT REFERENCES TO THE NAME WILL CAUSE THE PROPER RESULTS; EVEN IF THE NEW BODY IS SHORTER IN LENGTH, THE CONTROL CELL CONTAINING THE ORIGINAL BODY LENGTH REMAINS UNAFFECTED, ALLOWING THE BODY TO LATER BE RESTORED TO ITS ORIGINAL LENGTH. A TRANSFER IS THEN MADE TO THE MARKER HANDLING FUNCTION CAUSING THE ARGUMENT LIST TO BE REMOVED AND THE CALL CONCLUDED.

3. VAL FUNCTIONS BY LOCATING THE NAME IN THE ENVIRONMENT STACK AND COPYING THE DEFINING BODY TO THE OUTPUT STREAM. THE COPY TERMINATES WHEN MARKER IS FOUND. THE CALL IS CONCLUDED BY A TRANSFER TO THE MARKER HANDLING FUNCTION.

4. "\*: THE PARTIAL ARGUMENTS FUNCTION, HAS A VARIETY OF FORMS BUT ALL FUNCTION IN A UNIFORM FASHION. THE EVALUATION BEGINS BY RETRIEVING THE ENTIRE SET OF ARGUMENTS. THESE ARE STORED IN THEIR BINARY REPRESENTATION (IF PROVIDED), OR AS A NEGATIVE ZERO (IF NOT PROVIDED). THE PRESENCE OR ABSENCE OF EACH OF THE VARIOUS ARGUMENTS DETERMINES WHICH FORM IS BEING USED. ONE ARGUMENT DETERMINES THE NUMBER OF CALL LEVELS TO TRANSCEND. THIS IS DONE BY MOVING UP THE
P-CHAIN LINKS. IF AN ARGUMENT LENGTH IS TO BE RETRIEVED, THE ARGUMENT IS LOCATED; AND ITS LENGTH CELL, AFTER ADJUSTMENT, IS OUTPUT. IF THE NUMBER OF ARGUMENTS IS REQUIRED, THESE ARE COUNTED; AND THE VALUE IS OUTPUT. IF PART OF AN ARGUMENT STRING IS SPECIFIED, THESE CHARACTERS ARE LOCATED AND OUTPUT; THE CALL IS CONCLUDED BY A TRANSFER TO THE ARGUMENT HANDLING FUNCTION, WHICH REMOVES THE ARGUMENT LIST.

5. THE FIRST STEP EXECUTED BY THE BREAK FUNCTION IS TO TEST AN ARGUMENT WHICH MAY BE PROVIDED. IF IT IS ZERO, THE MACRO CALL IS TERMINATED, HAVING NO EFFECT ON THE STACK. IF THERE ARE NO ARGUMENTS, OR THE ARGUMENT PROVIDED IS GREATER IN BINARY VALUE THAN ZERO, THE BREAK TAKES EFFECT. WHEN BREAK IS CALLED, ITS P-CHAIN ENTRY APPEARS ON THE STACK AS FOLLOWS:


WITH 'CP' POINTING TO THE DATA CELL INDICATING WHERE SCANNING IS TO BE RESUMED WHEN THIS CALL IS COMPLETE. 'CP' IS THEN CHANGED TO POINT TO A MARKER STORED ON THE STACK. WHEN THE P-CHAIN ENTRY FOR BREAK IS REMOVED, THE SCAN FOR THE CALLING STRING IS RESUMED ON A MARKER, WHICH TERMINATES THE CALL, CAUSING THE BREAK IN THE SCAN.

6. IT IS A VERY CONCISE FUNCTION IN ITS IMPLEMENTATION. THE
STRUCTURE OF THE ENTIRE MACRO GENERATOR LENDS ITSELF EASILY
TO ENABLE AN OTHERWISE POTENTIALLY DIFFICULT MACHINE MACRO
TO BE CONSTRUCTED SIMPLY. THE DU MACRO, WHEN CALLED,
APPEARS ON THE STACK, AT THE TOP OF THE P-CHAIN AS FOLLOWS:

;(PHI),(PL),(CP),(LU),'DU', (LI),'COUNT',(LZ),'BODY',(MK)

WHEN EVALUATION BEGINS, THE P-CHAIN ENTRY IS MODIFIED TO
THE FOLLOWING:

;(PHI'),(PL),(CP),(LU'), 'CONTROL', (LI),'LOOP INDEX', (MK);

WHERE PHI' NOW CONTAINS THE LENGTH OF THE MODIFIED P-CHAIN
ENTRY, 'LU' CONTAINS THE LENGTH OF 'CONTROL', AND 'LI'
CONTAINS THE CURRENT LOOPING INDEX AS A FULL-BINDY BINARY
VALUE. 'CONTROL' CONTAINS THE MAXIMUM INDEX VALUE AND THE
BODY TO BE SCANNED IN THE FOLLOWING FORM:

;(LU'),'DU', (LI),'COUNT', (LZ),'BODY', (DU-MARK),

IT IS NOT DIFFICULT TO NOTICE THAT THIS IS ARGUMENTS ZERO,
ONE AND TWO, WITH 'LU' MODIFIED TO ENCOMPASS THE REST OF
THE ENTRY IN THE LENGTH IT DEFINES. MARKER IS CHANGED TO
BE 'DU-MARK'. THE ENTIRE P-CHAIN ENTRY IS LENGTHENED TO
INCLUDE 'LOOP INDEX' AND 'MK' AS SHOWN ABOVE. 'COUNT',
BEING IN FACT THE MAXIMUM LOOPING INDEX VALUE IS TESTED
AGAINST 'LOOP INDEX'. IF 'LOOP INDEX' IS LESS THAN THE
MAXIMUM VALUE, IT IS INCREMENTED. 'BODY' BECOMES THE INPUT
STREAM SOURCE, AND SCANNING RESUMES. WHEN 'DU-MARK' IS
ENCOUNTERED, THE P-CHAIN ENTRY AT THE HEAD OF THE LIST IS A
DU MACRO CALL. THE LOOP INDEX IS TESTED AGAINST THE
MAXIMUM VALUE AS ABOVE. IN BOTH THE INITIAL AND SUBSEQUENT
CASES, IF THE MAXIMUM VALUE IS REACHED, THE 'DU-MARK' IS
TREATED AS THE END OF A DEFINING STRING; AND A TRANSFER IS
MADE TO THE MARKER HANDLING FUNCTION IN ORDER TO REMOVE THE
P-CHAIN ENTRY FOR THE COMPLETED DU. SINCE 'CONTROL'
BEHAVES LIKE ARGUMENT ZERO OF THE P-CHAIN ENTRY, AND
CONSEQUENTLY 'LOOP INDEX' LIKE ARGUMENT ONE, THE SYMBOL
'GI' EVALUATED IN 'BODY' WILL REFER TO THE 'LOOP INDEX'
VALUE.
4. IMPLEMENTATION

The implementation at hand is the result of many months effort. It was programmed by generating the 1108 assembly code to match the CPL program provided in a paper entitled 'A General Purpose Macrogenerator' by C. Strachey. This effort brought a thorough understanding of the algorithm used by Strachey. It was then possible to apply the knowledge gained in other areas to enhance GPM. The result is a UNIVAC 1108 processor that is useful in a variety of ways within the context of the operating system.

4.1 NEW FEATURES

In adding new features, the central criteria has been their utility. Features that were not consistent with the processor philosophy as a whole were not included either. Those features that were incorporated satisfy the above criteria. These features are enumerated as follows:

1. The processor is designed to interface easily with the other system processors. The interactive macrogenerator can be used in conjunction with FORTRAN, ALGOL, COBOL, and
ANY NUMBER OF OTHER COMPILERS.

2. THE I/O MACROS WERE INCLUDED. THESE ALLOW THE READING AND WRITING OF FILES WRITTEN IN STANDARD DATA FILE FORMAT (SOFT). FILE MANIPULATION IS POSSIBLE.

3. THE DIAGNOSTIC MACROS WERE INCLUDED. A SIMILAR FORM EXISTED AS PART OF THE GPM ERROR MONITOR SYSTEM BUT THESE WERE NOT DIRECTLY REQUESTABLE VIA A MACRO CALL.

4. CERTAIN ERRORS OF IMPLEMENTATION IN GPM WERE CORRECTED. CERTAIN CASES COULD CAUSE IMPROPER ACTION. THESE CASES ARE DETECTED AND PREVENTED.

5. ARGUMENT RETRIEVAL IS IMPROVED. THE FULL CAPABILITY OF THE *M* MACRO ALLOWS FOR A GENERALIZED FORM OF CHARACTER PROCESSING THAT GPM DID NOT ALLOW. CONCATENATED "S" ALLOW ACCESS TO ARGUMENTS AT DIFFERENT LEVELS OF MACRO CALLS.

6. ITERATION IS MADE EFFICIENT WITH THE DU MACRO. IN GPM, LOOPSING IS ONLY POSSIBLE BY RECURSION. A TIME SAVINGS FACTOR OF TEN HAS REALIZED WITH THE DU MACRO. OTHER FEATURES (BREAK, FOR INSTANCE) ENHANCE THE EASE WITH WHICH THE DU MACRO CAN BE CONTROLLED.

7. BAR HAS BEEN EXPANDED TO INCLUDE MANY MORE OPERATIONS, INCLUDING ARITHMETIC VALUE COMPARISON, WHICH WAS DIFFICULT
IF NOT IMPOSSIBLE IN 'GPM'.

8. EDITING FEATURES ARE PRESENT FOR THE INPUT STREAM.

9. THE PROGRAM IS INTERACTIVE. ERRORS DO NOT USUALLY CAUSE
PROGRAM TERMINATION, AND EACH LINE INPUT CAUSES AN
IMMEDIATE RESPONSE. FEATURES ARE PRESENT TO PRESERVE THE
INPUT STREAM IF AN ABNORMAL TERMINATION IS REQUIRED.

4.2. PROBLEMS ENCOUNTERED.

THE CONSIDERABLE OBSCURITY OF ADAPTING ANOTHER'S WORK TO
FIT ONE'S OWN NEEDS WAS INDEED A PROBLEM TO BE RECKONED WITH.
ONCE THIS UNDERSTANDING WAS ATTAINED, THE ONLY MAJOR PROBLEMS WERE
THE ELUSIVE DETAILS WHICH PLAUGE A PROGRAM AND KEEP IT FROM BEING
'JUST RIGHT'. THE REQUIREMENTS AND ENHANCEMENTS PRESENTED HERE
SEEMED TO SUGGEST THEMSELVES. THIS IS NOT MEANT TO DETRACT FROM
THEIR WORTH, BUT INSTEAD MEANT TO SUGGEST THE GOOD FORTUNE OF
HAVING GOOD IDEAS COME AT FIRST INSTEAD OF AT LAST.
4.3. POSSIBLE EXTENSIONS:

AS SUGGESTED EARLIER IN THIS DOCUMENT, THE LIST OF PRIMITIVE FUNCTIONS PROVIDED IS BY NO MEANS A COMPLETE SET OF PRIMITIVE FUNCTIONS. FEATURES THAT MIGHT BE ADDED LATER MIGHT INCLUDE SUCH FUNCTIONS AS:

1. OUTPUT FORMATTING FUNCTIONS THAT PROVIDE THE ABILITY TO SET TAB STOPS IN AN IMAGE ARE DESIRABLE.

2. THE ABILITY TO LINK INTO THE 'CP' DATA CELLS IN THE P-CHAIN WOULD ALLOW ALTERATION OF MACRO CALL PATHS DURING THE COURSE OF EVALUATION. BREAK DOES THIS ON A LIMITED BASIS.

3. THE ABILITY TO DEFINE AND RETRIEVE DATA STRUCTURES WOULD ENHANCE THE PROCESSOR'S UTILITY.

4. THE ABILITY TO TRANSFER TO DIFFERENT POINTS IN THE INPUT STREAM FOR EVALUATION SCAN WOULD DETRACT FROM THE UTILITY AS AN INTERACTIVE PROCESSOR BUT WOULD PROBABLY INCREASE THE OVERALL UTILITY OF THE PROGRAM.
5. BIBLIOGRAPHY


3. STRACHEY, C., A GENERAL PURPOSE MACROGENERATOR,, COMPUTER JOURNAL 8,3(1965), PP. 225-241

4. UNIVAC, UNIVAC 1100 SERIES EXEC II AND EXEC II ASSEMBLER, UP-4049 REV 3, 1971

5. UNIVAC, UNIVAC 1100 SERIES OPERATING SYSTEM, UP-4144 REV 2, 1971