FRED 9/2/71 Binary Mode 3.

(*)

FRED 9/2/71 Binary Mode 3"

"FRED is a symbol stream processor. It takes as its input a stream of characters and produces as its output another stream of characters which is produced from the input by direct copying except in the case of macro calls in the input stream, which are evaluated before they are put into the output stream.

A macro call consists of a macro name and a list of parameters, each separated by a comma. The name is preceded by * and the last parameter followed by a semicolon.

e.g. *MAC, 2, 5;

Before the macro call can be evaluated the macro must have been defined by associating its name with a symbol string. This string may contain special symbols ~1, ~2, ... which stand for the first, second, etc, formal parameters; the symbol ~0 stands for the ~~mac~~ name of the macro being evaluated.

or "1, "2, depending on the Telecode being used; in these notes ~ is used for clarity

e.g. if name ABC defines string AB~1C~2AB
the call *ABC, XY, PQ;
will produce ABXYCPQAB

The system is completely general and it is possible to use a macro call in place of or in conjunction with a symbol string anywhere. In particular, macro calls are allowed in the actual parameters of other macro calls (including the name) and also in the defining string. The following examples demonstrate this point

e.g.

Name	Associated String
A	A~1A
APA	P~1~1P

Macro-call	Result
*A, C;	ACA
*A, *A, C;;	AACAA
**A, P;, Y;	PYYP

Enclosing any string in the string~~x~~ quotes <......> has the effect of preventing evaluation of any macro calls inside; in place of an evaluation, however, one layer of string quotes is removed.

e.g.

Input string	Result
Q<*A,C;>R	Q*A,C;R
Q<*>R<;>	Q*R;
Q<<*A,C;>>R	Q<*A,C;>R

The use of string quotes makes it possible to include any symbol in the output stream except an unmatched opening or closing string quote.

(*) what about SKIPS & TRIGGERS ?

(*) built into FRED

A macro is defined by a special macro DEF which has been ~~written in machine code and included in the system~~. DEF takes two arguments: the name of the macro to be defined, and the defining symbol string. It is usual to enclose the symbol string in string quotes in order to prevent any macro calls or uses of formal parameters from being effective during the process of definition. These quotes will be removed by the normal process of evaluating the arguments of DEF.

```
*DEF, A, <A~/A>;
*DEF, B, <B*A, X~/X; B>;
*DEF, APA, <P~/~/P>;
```

As definition is performed by an ordinary macro call the system insures that it is possible to carry out a definition anywhere it is possible to use a macro call. In particular a definition can be included in an actual parameter for a macro call and hence in the symbol string defining a macro.

In general the actual parameter list of a macro call is lost when the call has been completed and this applies also to definitions that are part of the list. Definitions of this sort are therefore temporary and their scope is confined to this particular macro call. If a macro name which has already been defined is defined again by a call of DEF, the latest definition supersedes the earlier one, though without destroying it.

The basic macro UPDATE which takes two arguments has the same sort of effect as DEF except that instead of establishing a new definition it alters the value associated with its first argument to be its second argument. There is a limitation on the use of UPDATE as the space available for the value is fixed by the first definition, the new string may be of equal length or shorter.

Integer arithmetic is provided with the aid of three machine code macros: BIN converts a digit string, possibly preceded by a sign, into a signed binary integer.

DEC is the inverse operation, converting a signed binary intege into a decimal digit string of characters.

BAR takes three arguments, the first being the character +, −, ., /, or R, the other two being binary numbers. It performs the indicated operation on these. BAR, R, x, y; gives the remainder when x is divided by y.

Punching Rules.

(*)
Utility prog users may not know about SIR subroutines.

~~Input is via CHIP subroutine and hence all CHIP rules for 920-903 code equivalents apply.~~

Tapes may be punched in 920 code or 903 code and the first non-blank character on the tape must be a "Newline".

All tapes must end with an unmatched ">" character followed by ~~a~~ "Newline". Stopcode is not recognised.

(*)
Suggest:-

Tapes may be punched in 900-Series Telecode (or 903, ISO, or ASCII codes with even-parity), using the character " to introduce formal parameters, or in 920 Telecode, using the character ~.

Blank, Erase, and Carriage return will be ignored everywhere. All tapes must start with Newline, or Carriage Return + Linefeed

All tapes must end with .... by Newline, or Carriage Return + Linefeed, and an optional Haltcode.

(*) ? Haltcode alone is not recognised & may cause corruption of FRED.

Method of use.

1) Load FRED under initial instructions - Mode 3.

2) Put first tape in reader.

(*) Swap.

(For output in 920 code trigger at 8)

For output in ~~903~~ 900-Series code trigger at 10.

3) For subsequent tapes trigger at 14.

Error Indications.

If the input stream contains contextual? errors an error number will be output in legible tape, preceded by 18" of blanks.

(*) and followed by – if not only not? is ERROR used?

Continuation after errors is not allowed.

Other tape errors such as parity error, impermissible character, etc will cause a legible tape message of the form " CHI/O ERROR N " to be output.

(*) must list these

Contextual? error indications are:

1 - Unmatched ; in definition string.
2 Unquoted ~ in argument list.
3
4 Not enough arguments supplied in call
5 Probably a missing ;
6
7 Undefined macro-name.
8
9 Update string too long.

Also FRED shoots through reader if.....