TX-0 COMPUTER
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
CAMBRIDGE 39, MASSACHUSETTS

M-5001-35

July 24, 1961

PRELIMINARY OPERATING INSTRUCTIONS FOR MACRO III

MACRO III is an assembly program for the TX-0 Computer. It will assemble any program that MACRO IIA will assemble, and in addition has various new features of its own. It is assumed that the reader is familiar with the memos MACRO IIA (M-5001-5-1) and Relocatable Programming for the TX-0 (M-5001-34).

In addition to those recognized by MACRO IIA, MACRO III recognizes the following pseudo-instructions: repeat, variables, relocatable, entry, exit, frontloading, and readin. The new features can be described largely in terms of these pseudo-instructions.

repeat n,x causes the quantity x to be inserted in the program n times. n must have a definite value at its appearance on Pass 1, and cannot be an argument of a macro-instruction. x consists of everything between the comma and the next carriage return, and may consist of any words, parameter assignments, macro-instructions, etc. that are legal elsewhere except the pseudo-instructions repeat, text, and start. A tab may be substituted for the comma. If n is zero, x will be ignored. n may not be negative. Although legal, the pseudo-instructions define, terminate, constants, variables, readin, and frontloading are not meaningful within the range of a repeat.

Variables are symbolic syllables not otherwise defined which have at least one letter in upper case at their first appearance in the program. Succeeding appearances of a variable may, but need not, have a letter or letters in upper case. At the pseudo-instruction variables, a block of storage of length equal to the number of distinct variables preceding it is reserved. On pass 2, the variables will be assigned values of consecutive locations in the variables storage in the order of their appearance. Thus it is not necessary explicitly to reserve temporary storage registers in the program. Variables used in a macro-instruction definition cannot be defined in the definition, and therefore must be defined before it. If desired, this can be done by using the variable in the right-hand side of a parameter assignment before the macro-instruction definition. Note that at the end of Pass 1, all variables are defined as -0. The correct value is assigned during Pass 2.

<u>frontloading</u> if used must be the first thing on the English tape. It causes a front input routine using registers 1-37 to be prepared, and sets the location to 40. <u>CM select</u> must be on when the program is read in.

<u>readin</u> suppresses the input routine entirely. The entire program is punched in read-in mode.

<u>relocatable</u> sets the location to relocatable 0. The program will be assembled in relocatable format until a location assignment is encountered whose relocation count is 0. A location assignment with relocation count of +1 will cause MACRO to enter or remain in relocatable mode. The input routine is replaced with a one-word transfer instruction <u>trn 17000</u>, which transfers control to the Binary Relocatable Subroutine Loader. Address tags are defined relocatable or absolute as the current mode is relocatable or absolute. Symbols defined by parameter assignment will have a relocation count equal to that of the word on the right-hand side of the equals sign except that no symbol can have a relocation count exceeding 1 in magnitude. Storage words in relocatable mode may have relocation +1, -1, or 0; words in absolute mode may have relocation 0 only. Words in macro-instruction definitions may acquire any relocation so long as the final storage words assembled have relocation as specified above.

entry and exit are used to produce the program card and transfer vector, respectively. The usage is entry s1,s2,. . .,sn where the symbols s1, etc. are addresses in the program to which other programs may transfer control; and exit s1,s2, . . . ,sn where the symbols s1, etc. are the names of other programs or subroutines to which control is to be transferred. The arguments of entry must be defined elsewhere in the program, while the arguments of exit are defined as addresses in the transfer vector by this use and must not be defined elsewhere. Secondary entries to a program are defined by a second appearance of entry immediately after the first. To the extent that the pseudo-instructions relocatable, entry, and exit are used, they must be used in that order, and no storage words may intervene between them. A program with no entry is a main program, and the pseudo-instruction exit will cause the program card to be punched with a name of +0, as required by the loader. If neither entry nor exit is used, no program card will be provided. Since any program to be loaded by the BRS Loader must have a program card, it has been made possible to get a program card with a main program entry by using the pseudo-instruction entry with no arguments. The maximum number of arguments of entry is 37 (octal); there is no limit on the number of arguments of exit.

The treatment of the pseudo-instruction noinput, and the function of the noinput bit in the TAC is slightly different from that in MACRO IIA. In an absolute program, noinput replaces the normal input routine with an instruction trn 17744, which will transfer control to the normal input routine if it is in storage. If the input routine interprets this instruction, it will be read as an immediate start at location 17744, which, of course, means it is ignored since this is the entry to the input routine. This instruction is also supplied if the TAC is examined and bit 4 is off. In a relocatable program, the normal input routine is trn 17000, a transfer to the BRS Loader. Here, noinput, either on the tape or via the TAC, deletes this transfer. This is principally useful in the preparation of library or other tapes with more than one program.

MACRO III will accept constants within constants up to eight levels deep. Example:

add (llr (20) - add - (30

will cause three different constants to be stored: 20, 30, and llr (20) - add - (30. Missing right parentheses are supplied at the terminating character (tab, carriage return, or comma).

MACRO III symbol punches are not compatible with MACRO IIA or with FLIT I. They can be read by FLIT II, which will define the symbols according to the location of the program in memory.

Various new error stops have been added to MACRO III. These are:

use	Undefined symbol in an <u>entry</u>. -0 is substituted for the entry and the BBS Loader will ignore it.
usr	Undefined symbol in the count of a <u>repeat</u>. The symbol is taken as 0. This is the only undefined symbol alarm which can occur on Pass 1.
ir_	In general, illegal relocation. The third letter identifies type as in case of undefined symbol. The relocation is taken as 0.
ilr	Illegal repeat. Negative count, or repeat in the range of a repeat. The repeat is ignored.

mdx	Multiply defined exit. A symbol in the arguments of exit is defined with a conflicting value elsewhere in the program. The attempt at redefinition is ignored.
mdv	Multiply defined variable. An attempt to define a symbol as a variable failed because the symbol was previously defined as other than a variable. The attempt is ignored. The error stop occurs on Pass 1 only.
cld	Constants location disagreement. The location of the pesudo-instruction constants differs on Pass 2 from that observed on Pass 1. This can happen if tapes are processed in the wrong order, if the location of constants was indefinite on Pass 1, or if the PETR missed a word. Other causes may obtain also. The effect is that all preceding constants syllables have been assigned the wrong value. Assembly cannot be continued.
vld	Variables location disagreement. See cld.

Thanks are due to Robert Wagner, who assisted in writing and debugging MACRO III.

Author R. A. Saunders
R. A. Saunders

Approved J B Dennis
J. B. Dennis

Macro III Program Examples

```
Demonstration Main Program      | title

relocatable
exit tst, opt                   | defines tst=0+reloc, opt=1+reloc
                                | space before opt is optional

sta,      llr (add tbl          | execution starts here
          slr get
          llr (add (101001
          slr sw

lp,       cla
get,      xx
          sto T                 | defines variable T
          llr .                 | . is relocated here
          tra opt               | call subroutine
sw,       xx
          pno
          add t                 | t may be in upper case
          tra tst               | call subroutine
          add get
          add (1
          sto get
          add (-add-tbl-2       | if tbl were relocatable, constant
          trn lp                | would have relocation count of -1
          hlt
          tra sta

variables                       | in relocatable programs, use variables
constants                       | before constants lest next program be
                                | stored on top of variables

16000|

tbl,      1         2         3

xx=hlt

start                           | address not significant
```

Do not forget the carriage return after start. Note that this program has a bug in it--it needs llr . before tra tst.

Binary Output Tape for Demonstration Main Program

The symbol r used here indicated the relocation constant which is determined by the loader at execution time.

```
Address   Word      Relocation (if any)

          417000                    | transfer control to BRS Loader

          600000                    | program card identifier
          -2                        | word count
          100000                    | relocation word
          0                         | designates main program entry
          2          +r             | enters at relocated 2
          2                         | number names in transfer vector
          77775                     | checksum

          100000                    | relocatable block, origin relocatable 0
          -25                       | last address
          25202                     | relocation word
 0+r|     502020                    | flex tst) transfer vector, altered
 1+r|     712200                    | flex opt) by loader during patching
 2+r|     300027     +r             | sta,
 3+r|     100007     +r
 4+r|     300031     +r
 5+r|     100013     +r
 6+r|     700000                    | lp,
 7+r|     630000                    | get,
10+r|     26         +r             | t=26+r
          501252                    | relocation word
11+r|     300011     +r
12+r|     500001     +r             | call subroutine opt
13+r|     630000                    | sw,
14+r|     664020                    | pno
15+r|     200026     +r
16+r|     500000     +r             | call subroutine tst
17+r|     200007     +r
20+r|     200032     +r
21+r|     7          +r
          504000                    | relocation word
22+r|     200033     +r
23+r|     400006     +r
24+r|     630000
25+r|     500002     +r
          614573                    | checksum

          100027                    | new relocatable block, origin 27+r
          -33                       | last address
          20000                     | relocation word
27+r|     216000                    | beginning of constants
30+r|     101001
31+r|     200030     +r             | add (101001
32+r|     1
33+r|     561775
          356753                    | checksum
```

```
        16000           | absolute block, origin 16000
        -16002          | last address
16000|  1               | tbl,
16001|  2
16002|  3
        777773          | checksum

        200000          | start block denotes end of tape
```

Nothing is stored in register 26+r, which is reserved for T.

Demonstration Intermediate Level Subroutine

```
reloc
entry tst
exit opt,dpt

ts1,      cla
          add (-2000
ts2,      ial
          dis
          ial
          add (1
          trn ts2

ts3,      lac
          llr .
          tra dpt
          add (101001
          prt
          cal
tx,       xx                  | beware of multiply defined tags

tst,      ial                 | entry point
          add (tra-llr+2
          sto tx
          tac
          trn ts4
          cyl
          trn ts1
          cyl
          trn ts3
          tra tx-1

ts4,      slr T1
          lac
          llr .
          tra opt
          add (100101
          pno
          llr T1
          tra ts1

          xx=hlt

vari
const
start
```