ML/I User's Manual — Appendix C

Implementation on the ICT 1900 Series

Lorne H Bouchard

30th August 1968

This implementation is based on version AGA of ML/I.

C.1 Restrictions and Additions

The 1900 implementation of ML/I contains all the features described in this manual with the following additions or amendments.

a. The number of P-variables is preset to 100.

b. `SPACES` is an allowable delimiter specification and stands for one or more spaces.

c. The S-variable feature has been implemented; these are used to control certain features of ML/I, mainly the input/output options (see Section C.3).

C.2 Using ML/I

ML/I is provided in the form of a binary dump on paper tape, and is loaded into store in the usual way, e.g.:

```
LO#ML/1  4
```

will load ML/I from `TR` (operator number) 4. ML/I is entered at location 20, i.e.,

```
GO#ML/1  20
```

On entry to ML/I the environment always consists of the operation macros, 100 P-variables (`P1-P100`) and 30 S-variables (`S1-S30`). Input is via the card reader and output will normally be on the line-printer, but may also be on cards (see Section C.3.1). All peripherals are allocated dynamically.

C.2.1 Store

The basic system occupies about 5K instructions and will request up to 16K of store on entry (the difference to be used as run-time stacks). One character is stored per 24 bit word as in the Titan implementation.

ML/I will only enter if at least 8K of store is available (i.e. with a minimum of 3K of stack space). If less store is available the program will halt:

```
0#ML/1; HALTED  :- NEEDS MORE CORE
```

at which point space should be made available by deleting one of the other programs in store and re-entering ML/I by the command:

```
GO#ML/1
```

C.2.2 Input

Input is punched on standard 80 column cards. The standard 1900 series 64 character code is used. Cards are assumed to be punched in free format and all 80 columns are treated as input data.

All 'non-visible' spaces are removed from a card on input. That is, each card is scanned backwards until a character other than a space is found and the next character position is set to the end-of-line character ('newline'). Thus a blank card would be represented internally as a single 'newline' character.

A '`****`' card is used to terminate the input stream. This will cause ML/I to print out monitoring information and to terminate the run by deleting itself.

C.2.3 Output

Output on the lineprinter is restricted to 80 characters per line (cards images). Any attempt to print (or punch) more than 80 characters on a line will cause a 'newline' (new card) to be output after the 80th character and the remaining characters will be printed (punched) on the new line (card). No information is lost, only the layout is disturbed. Any line of less than 80 characters is padded with spaces to form a standard 80 character record.

C.3 Use of S-variables

Thirty special global variables (`S1-S30`) are provided. These variables may be set, altered or tested by the user in the usual way (i.e. as the ordinary `P` or `T` variables).

These variables control various options of the input/output package and should be used with the utmost care.

Most users need only use `S6` and `S10-S25` as described in the next three sections. A complete list of the currently allocated S-variables is given in Section C.3.4.

C.3.1 Card output

Card output should be used sparingly as it is usually much slower than lineprinter output.

Slow peripherals output is controlled by the setting of bits in `S6`. On entry, `S6` is preset to 1 (bit 23 is 1) which means lineprinter output is requested. Setting bit 22 causes cards to be output. Thus

```
MCSET S6 = 3;
```

produces card and lineprinter output, and

```
MCSET S6 = 2;
```

produces only card output.

It is advisable not to use card output when debugging and to switch it off before terminating a program (otherwise monitoring information will also appear on cards).

This can be done simply by inserting the statement:

```
MCSET S6 = 1;
```

before the '`****`' card.

C.3.2 Formatted output

Special variable `S10` controls the tab reconstruction of the input line from cards. A 'tab' is defined as *one* or more spaces occurring before a tab position. It must be used with care or 'tabs' will be inserted all over the place!

A useful device is to define a macro, called `$` say, which will output a 'tab'. This is achieved by the following three statements:

```
MCSET S10=1;
MCDEF$AS<   >;
MCSET S10=0;
```

In general the rule is that 'tabs' are translated into a single space when they are encountered during the evaluation of an operation macro argument unless they are within a skip.

The following two statements would then produce identical output:

```
$LDX$3$X
      LDX   3  X
```

C.3.3 Tab Positions

The standard tab settings of PLAN are preset in 1900 ML/I at positions 7-13-16-36-80, but may be reset by the user as he wishes.

Up to 15 different tab positions are available and they can be set by setting S-variables `S11-S25` to the appropriate values. Note that the value of the tab setting is one less than the column number of the position.

Thus:

```
MCSET S11 = 5;
```

would set the first tab position at column 6.

C.3.4 Description of the 'tab' input/output mechanism

It might be of some help to describe the details of the 'tab' input/output sequences in 1900 ML/I.

On input, a card is read into store and 'non-visible' spaces are removed as described in C.2.2. If `S10` is non-zero the following sequence is obeyed:

1. Scan the 'tab' list backwards (`S25-S11`) until a 'tab' position not beyond the current end-of-line is found and if there is none, exit.
2. If a space precedes this 'tab' position then replace it and all previous spaces by a single 'tab' symbol, compressing the line in the process.
3. Consider the next lower 'tab' position and if there is none, exit; otherwise go back to 2.

On output the 'tab' symbol is dealt with in the following sequence of the output routine. The linepointer records the next available position (0-79) in the output line image which is cleared to spaces at the start of each line.

1. If the linepointer is beyond the last 'tab' position, treat the 'tab' as a 'newline' and exit.
2. Scan the 'tab' list backwards until a 'tab' position not greater than the linepointer is found and if there is none, treat the 'tab' as a 'newline' and exit.
3. Set the linepointer to the next 'tab' position but if it is above position 79, do a 'newline' and exit.

A 'newline' outputs the current contents of the output line image and clears the line image to spaces.

C.3.5 Allocation of S-variables

S-variable	Initial value	Function
1	0	ILPT: pointer to input line
2	0	OLPT: pointer to output line
3	0	FAST/SLOW input peripheral switch (SLOW)
4	0	CR/TR input switch (CR)
5	0	MT/ED input switch
6	1	LP/CP/TP output switch (LP)
7	0	MT/ED output switch
8	0	Spare
9	0	Spare
10	0	TAB reconstruct switch (OFF)
11	6	\
12	12	|
13	15	|
14	35	|
15	80	|
16	80	|
17	80	|
18	80	|-- TAB settings (PLAN tab positions)
19	80	|
20	80	|
21	80	|
22	80	|
23	80	|
24	80	|
25	80	/
26		Spare
27		Spare
28		Spare
29		Spare
30		Spare

C.4 Typical job

C.4.1 Listing of card input

```
MCSKIP!
;!DEFINE ENVIRONMENT
MCINS@.;!
MCSKIP MT,<>;!
MCSET S10=1;!TAB RECONSTRUCT INPUT LINE
MCDEF$AS<   >;!$ IS NOW A 'TAB'
MCSET S10=0;! NO MORE TAB RECONSTRUCT
MCSET S6 = 3;!CARD OUTPUT REQ'D
MCDEF VERSE WITH(,)AS<!
$OLD MCDONALD HAD A FARM E-I-E-I-O
$AND ON HIS FARM HE HAD SOME @A1. E-I-E-I-O.
$WITH A @A2. @A2. HERE AND A @A2. @A2. THERE,
$HERE A @A2. THERE A @A2. EVERYWHERE A @A2. @A2.,
$OLD MCDONALD HAD A FARM E-I-E-I-O.

>;
A CHILDREN'S SONG AS SIMPLE EXAMPLE OF MACRO EXPANSION

1.VERSE(CHICKS,CHEEP)
!DEFINE ALTERNATIVE SYNTAX FOR VERSE MACRO
MCDEF NOW WITHS TRY AND
AS<VERSE(@A1.,@A2.)>;!
2.NOW TRY DUCKS AND QUACK
MCSET S6=1;
```

C.4.2 Copy of Console log

```
   LO#ML/1 5  OK
 0#ML/1;
 0#ML/1; HALTED  :-
   GO#ML/1 20  OK
CR15  0-ML/1
LP12  0-ML/1
 0#ML/1;CORE   7616
 0#ML/1; HALTED  :- NEEDS MORE CORE
   DE#FIPB  OK
 0#FIPB; DELETED :-
   GO#ML/1  OK
 0#ML/1;CORE  16384
CP18  0-ML/1
 0#ML/1; DELETED :- EJ
```

C.4.3 Lineprinter output

```
A CHILDREN'S SONG AS SIMPLE EXAMPLE OF MACRO EXPANSION

1.    OLD MCDONALD HAD A FARM E-I-E-I-O
      AND ON HIS FARM HE HAD SOME CHICKS E-I-E-I-O.
      WITH A CHEEP CHEEP HERE AND A CHEEP CHEEP THERE,
      HERE A CHEEP THERE A CHEEP EVERYWHERE A CHEEP CHEEP,
      OLD MCDONALD HAD A FARM E-I-E-I-O.

2.    OLD MCDONALD HAD A FARM E-I-E-I-O
      AND ON HIS FARM HE HAD SOME DUCKS E-I-E-I-O.
      WITH A QUACK QUACK HERE AND A QUACK QUACK THERE,
      HERE A QUACK THERE A QUACK EVERYWHERE A QUACK QUACK,
      OLD MCDONALD HAD A FARM E-I-E-I-O.



VERSION AGA

MACROS ARE

NOW TRY
VERSE(
$
MCALTER
MCWARNG
MCWARN
MCNOWARN
MCGO
MCNODEF
MCSKIPG
MCDEF
MCSUB (
MCSET
MCLENG (
MCNOSKIP
MCDEFG
MCSKIP
MCINS
MCPVAR
MCNOINS
MCINSG
MCNOTE

WARNINGS ARE
```

```
INSERTS ARE

@

SKIPS ARE

<
!


AT END OF MACRO PROCESSING          18.21.55
25 LINES  23 CALLS
```

C.4.4 Listing of Card output

```
A CHILDREN'S SONG AS SIMPLE EXAMPLE OF MACRO EXPANSION

1.    OLD MCDONALD HAD A FARM E-I-E-I-O
      AND ON HIS FARM HE HAD SOME CHICKS E-I-E-I-O.
      WITH A CHEEP CHEEP HERE AND A CHEEP CHEEP THERE,
      HERE A CHEEP THERE A CHEEP EVERYWHERE A CHEEP CHEEP,
      OLD MCDONALD HAD A FARM E-I-E-I-O.

2.    OLD MCDONALD HAD A FARM E-I-E-I-O
      AND ON HIS FARM HE HAD SOME DUCKS E-I-E-I-O.
      WITH A QUACK QUACK HERE AND A QUACK QUACK THERE,
      HERE A QUACK THERE A QUACK EVERYWHERE A QUACK QUACK,
      OLD MCDONALD HAD A FARM E-I-E-I-O.
```

C.5 Queries about 1900 ML/I

All queries concerning 1900 ML/I should be addressed to:

LORNE H. BOUCHARD
COMPUTING CENTRE
UNIVERSITY OF ESSEX.

Copies of the binary dump paper tape of 1900 ML/I can also be obtained from the above.