

KENT ON-LINE SYSTEM

Document: KUSE/ML1/2

The ML/1 sub-system

**P. J. Brown
University of Kent at Canterbury
September 1973**

Note

ML/I is a general purpose macro processor. It is specified in the ML/I User's Manual, which can be obtained from the author. This document is an Appendix to the User's Manual and describes the features peculiar to the KOS implementation of ML/I.

Another document available from the author is "The ML/I macro processor: a simple introductory guide". This is specially applicable to introducing the user to ML/I at a console session.

ML/I User's ManualAppendix G KOS version of ML/I for the ICL 4130G.1 Restrictions and additions

The implementation of ML/I that runs under the KOS on-line system for the ICL4130 contains all the features described in the ML/I User's Manual (4th Edition, August 1970), plus the New Features 1 to 4 (startlines, stop markers, controlled line numbers and optional warning markers) described in a supplement to the manual, except that, since the ML/I processor may be shared between several users, MCALTER is not available. (MCALTER is not even recognized as a macro name:) The MCONT macro is a feature peculiar to this implementation (see G.2.6.).

As in most recent versions of ML/I, newline, not semicolon, is the terminator for operation macro calls. Since MCALTER is not available the user must stick with this.

G.1.1. Compatibility with NICE version of ML/I

There is another implementation of ML/I for the ICL4130. This runs under the NICE executive. It is described in Appendix E of the ML/I User's Manual. The two implementations of ML/I differ in the way they are operated, but they are functionally identical to the following extent:

- (a) Any ML/I process run under the KOS implementation will also run under the NICE implementation.
- (b) Any ML/I process run under the NICE implementation will run under the KOS implementation provided that MCALTER is not used and provided the usage of S-variables conforms with the KOS implementation.

G.2 Operating instructions and I/O

The KOS implementation of ML/I is a sub-system of KOS. It is entered by means of the command:

ML1 DR-spec

where the DR-spec, which is omitted if the default devices are to be used, is as described in the KOS User's Manual. Any error in the entry command to ML/I will cause the command to be treated as illegal and hence ignored.

G.2.1 Use of user's workspace

ML/I needs some of the user's workspace for holding macro definitions, stacks, etc.. A reasonable algorithm for estimating the number of words of workspace needed is

$$\text{requirement} = 100 + 11/10 \times \text{number of characters needed to define macros.}$$

On entry ML/I finds out the largest piece of user's workspace that is free. Assume the size of this is N words. It outputs the message

N WORDS OF STORAGE AVAILABLE

and then uses an optional data question-and-answer to find out how much of this the user wants. The question is

STORAGE REQUIREMENT=

and the answer must be an integer between 50 and N (inclusive).

If ML/I is not running in conversational mode and the above question-and-answer is omitted (or supplied with an illegal answer), then all N words are allocated to ML/I. If the question-and-answer is supplied, it must immediately follow the entry command to ML/I.

Note that if nearly all the user's workspace is taken by ML/I, workspace may subsequently become exhausted if job files are created or extended while within ML/I. In this case it is usually best to exit from ML/I and then re-enter it, asking for rather less workspace.

G.2.2 Examples of entry commands

- (a) &ML1
- (b) &ML1 FROM JFILE 1 TO PRINTER
- (c) &ML1
STORAGE REQUIREMENT = 2040

might be used when ML/I was being run in non-conversational mode.

G.2.3 Input/output and commands

ML/I takes its input from the data device and sends its output to the results device. At the end of data ML/I returns to command status and permits, in addition to the global KOS commands, the supplementary commands CONT and SPACE. These are described in the Sections that follow.

G.2.3.1 Continue

<u>Form of Command</u>	CONT	<u>DR-spec</u>
------------------------	------	----------------

<u>Examples</u>	(a)	&CONT
	(b)	&CONT TO PRINTER
	(c)	&CONT FROM PTAPE TO DC (XXX,YYY)

Action Continue processing using given data and results devices. Note that, by using CONT, the user can split his data and his results into several separate pieces, possible going to different devices. However no call of a macro or any other construction may overlap between pieces of data and ML/I diagnoses an error if a construction remains unterminated at the end of a piece of data.

At each CONT command, all line counts are set back to zero, so that line counts in error messages always relate to the current piece of text.

G.2.3.2 Available space

<u>Form of Command</u>	SPACE
------------------------	-------

Action Print out, on the message device, the number of words of ML/I's allocation of user's workspace that are still free.

G.2.4. Breaks

User breaks are always allowed. If a break occurs while ML/I is processing it exits, but if a break occurs when ML/I is in command status then control remains within ML/I.

KNL6

G/4

KUSE/ML1

G.2.5 Examples of use of ML/I

The following is an example of how ML/I might be used at a console. Characters typed by the machine have been underlined.

*** KOS READY - VERSION KNL6 E

&JFILE IB
:TESTS FOR MACROS
:--PIG++PIG
:%67+5.
:MCSET P1=45
:%P1-8.
:.

&ML1

***662 WORDS OF STORAGE AVAILABLE STORAGE REQUIREMENT=600

***ML1 (VERSION KNL6D) PROCESSING

:MCINS :8 .
:MCSKIP MT, < >
:MCDEF PIG AS DOG
:.

***3 LINES, 3 CALLS

&CONT FROM JFILE
TESTS FOR MACROS
--DOG++DOG
72
37

***5 LINES, 3 CALLS

&CONT
:I AM A PIG
I AM A DOG

:.

***1 LINES, 1 CALLS

&JOB

*** EXIT FROM ML1

G.2.6. Another use of ML/I - Conversational macros

If a macro tries to insert argument zero (e.g. %AO., %WAO. or %BO.) then this is taken as a request for a fresh line of source text. This source text, excluding the newline at the end, is then taken as the argument and evaluated in the normal way. The facility is best used at a conversational device, but can also be used in non-conversational mode.

The following is an example. Assume a macro is defined:

```
MCDEF ASKME
AS<WHAT IS YOUR NAME?
MCDEFG N AS %AO.
WHAT IS YOUR AGE?
MCSET P1 = %AO.
WHAT IS YOUR WAGE?
YOU SHOULD BE EARNING MORE THAN %AO.,N
>
```

Then its use at a console might proceed thus.

```
:ASKME
WHAT IS YOUR NAME?
:JOHN
WHAT IS YOUR AGE
:29
WHAT IS YOUR WAGE?
:£25,000
YOU SHOULD BE EARNING MORE THAN £25,000, JOHN
```

To completely master the use of argument zero it is necessary to understand ML/I's input routine fully. When ML/I is processing an identifier it needs to read ahead to the next punctuation character after it. Sometimes ML/I reads even further ahead. Thus if there was a macro ML WITHS PIG, ML/I would need to read the next atom after each newline character to see if it was PIG. We will say that any text that ML/I has input but has not yet processed is in limbo. Thus the newline following ASKME in the above example is in limbo. Argument zero causes new input to be read, ignoring any text in limbo. Characters are taken in sequence until a newline is read. Note that if the current source line has only been partially input argument zero will absorb the remainder. Thus in ASKME JACK, JACK would be taken as the first argument zero. Any text in limbo is processed after scanning returns to the source text. Thus the newline following ASKME would be processed after ASKME had been called. Scanning would then resume with a fresh line of input.

ML/I uses the ordinary KOS output mechanisms, which means output is in units of lines. Thus it is not possible to have a question on the same line as the answer, and this is why all the questions in the ASKME macro are followed by newlines.

Note that argument zero is evaluated. Thus if a macro contains

```
MCDEF YES AS 1
%AO.
```

and the user types YES, then this will be taken as 1. It is not permissible to insert argument zero during the evaluation of another argument zero. A violation of this rule produces the message

... IS ILLEGAL MACRO ELEMENT

An attempt to insert argument zero into the source text also leads to this error.

If there is no data left when argument zero tries to read some, then the process is aborted. (This is not, however, treated as an error, and no message is given).

MCCONT macro

Purpose Similar to &CONT command.

General form MCCONT argA NL

Examples (a) MCCONT FROM DC (XYZ)

(b) MCCONT FROM DC (%A1.) TO DC (T%SL0.)

System action The current data and results devices are closed. If ARG A is not a correct DR-spec the usual KOS error message is output (e.g. SYNTAX ERROR IN COMMAND or DC ERROR...) and the processes are aborted. If ARG A is correct, the data and results devices are re-set accordingly and the line count set back to one. (Note that the ML/I input routine sometimes needs to read ahead a few characters, especially when multi-atom names are in the environment. (See definition of "in limbo" in the previous Section). Hence it is conceivable that a few characters taken from the previous data device may be processed after MCCONT has changed the device. If MCCONT is called from the source text this will only happen if there is a construction of form NL WITHS... in the environment.

G.3 Character set

ML/I accepts any character that the appropriate device routine accepts. Spaces, tabs and newlines are treated as characters in the ordinary way. There is a method for inputting characters not provided by the physical device being used (see Section G.7).

G.4 Error messages

Error messages are output on the message device as they are detected. With reference to Chapter 6 (of the ML/I User's Manual), the number 2N (i.e. the maximum number of characters in a piece of text inserted into an error message without being truncated) is 64. In conversational mode, some of the redundant newlines are removed from error messages so that the output is less bulky.

Illegal input characters are dealt with by the KOS system, not by ML/I.

There are a number of special error messages and informative messages peculiar to this implementation. All of them should be self-explanatory. It can be seen from the above example that, when it has finished processing a piece of text, ML/I prints a message giving the number of source lines scanned (as given by S2) and the number of macro calls performed during the processing of that text.

G.5 Integer calculations

All macro variables and constants used in macro expressions must fit into a 24-bit word. Overflow is not detected and its effect is undefined.

The initial environment contains ten permanent variables. They are not initialized to any particular values.

G.6 Layout keywords

The five following layout keywords exist

SPACE	meaning "space".
TAB	meaning "tab".
NL	meaning "newline".
SL	meaning "startline".
SPACES	meaning a sequence of one or more spaces.

G.7 S-variables

The KOS implementation of ML/I has twenty-one S-variables. However, since I/O is controlled largely by the KOS system rather than by S-variables, their usage is rather limited and is a small subset of the facilities provided by the NICE implementation. The full list is as follows

<u>Variable</u>	<u>Meaning</u>	<u>Initial value</u>
S1 - S4	see New Features	0
S10	KOS device number of default input device (i.e. console number if console, 508 if card reader).	
S16	code to be translated on input	-1
S17	code to replace S16	0
S21	(0 means suppress output) (1 means normal output)	1

S16 and S17 are used to input a character not provided by the input medium. If the input routine finds a character whose internal code on the 4130 corresponds to the value of S16 then the character whose internal code is given by S17 is substituted in its place before the character is fed to ML/I. For example if it is desired to use a dollar sign (internal code 4) for a tab (internal code 1) then this can be achieved by

```
MCSET S16 = 4
MCSET S17 = 1
$X$Y.....
```

If it is required to switch this option off at any time (e.g. to process some text on disc containing genuine dollar signs), S16 can be reset to an impossible internal code such as minus one.