ML/I User's Manual — Appendix F

Implementation on the CDC 6000 series

L.T. Shafe and K.J. McCallum
London University Computer Service

September 1971

F.1 Overview

The LUCS version of ML/I is written in BCPL for the CDC 6000 series machines. The processor as implemented is identical to that described in *ML/I User's Manual*, 4th Edition, August 1970, except for a number of extensions described here.

F.2 The ML/I Job Control Card Format

At the present time ML/I is on permanent file and before it can be used must be attached to the user's job, for details of this see the author.

The ML/I job card takes the form:

```
ML1.
```

or

```
ML1 (P1,P2,...,Pn)
```

where each of the parameters of the optional parameter list take the form `c = fn`, where `c` is any of the options listed below and `fn` is either the filename to be used or zero indicating no file is to be produced. If any parameter is omitted the standard file name associated with each parameter is used.

Code Letter		**Standard File name**
I	Source Input	INPUT
I1 I2 I3 I4	Alternative input streams that can be switched to by setting `S11`.	None
O	Text output	TEXT
M	Monitor output, error messages and headings.	OUTPUT
L	List source input	OUTPUT

F.3 Details of Implementation

Ten P-variables are assigned at the start of each run, if more are required `MCPVAR` must be used. None of the P-variables are initialised.

The implementation makes optimum use of the field length available, if the process aborts because of lack of storage it can be re-run with a larger field length. However, such an error usually indicates an error in the program. A field length request of 20000 decimal words is sufficient for most applications.

At the start of each run a heading is printed at the head of a page. An attempt is made to read from the input file, if it is empty it is rewound and a warning message issued, if the

file is still empty the job is aborted. If the end of any of the input streams 1–4 is reached input is automatically switched back to the main input file.

At the end of each run is printed a message indicating whether macro processing was completed or was terminated, followed by a summary of the number of macro calls performed, the size of the free space area and an indication of the number of words in use when the process finished. Finally a list of all the names known to the macro processor is printed.

F.4 S-variables

All the S-variables are initialised to zero, the following list indicates the action performed when they have this value and when the value is altered as shown. The action performed when an S-variable is set to a value not indicated below, or when an S-variable not listed is altered, is undefined.

F.4.1 Input switch - `S11`

`S11 = 0` Input from standard input file (`I = fn`)

`S11 = 1` Input from input file 1 (`I1 = fn`)

`S11 = 2` Input from input file 2 (`I2 = fn`)

`S11 = 3` Input from input file 3 (`I3 = fn`)

`S11 = 4` Input from input file 4 (`I4 = fn`)

F.4.2 Rewind switch - `S12`

`S12 = 0` when the end of a record is read input is switched to the standard input file, if reading standard input file process finishes.

`S12 = 1` when the end of a record is read skip back one record and switch to standard input file, if reading standard input file the process finishes.

F.4.3 Recall switch - `S13`

`S13 = 0` when the end of the standard input file is reached the process ends.

`S13 = 1` when the end of the standard input file is reached recall the macroprocessor.

F.4.4 Textoff switch - `S14`

`S14 = 0` write text to text file.

`S14 = 1` do not write to the text file.

F.4.5 Listoff switch - S15

`S15 = 0` list the input file (unless `L = 0`)

`S15 = 1` switch off the listing

F.4.6 Ignore spaces - S16

`S16 = 0` do not ignore spaces on input file

`s16 = 1` ignore all spaces on input file

F.4.7 Ignore newlines - S17

`S17 = 0` do not ignore the newline character from input

`S17 = 1` ignore all newline characters from input

F.4.8 Multiple characters - S18

`S18 = 0` process input normally

`S18 = n` preprocess input as follows:
if character has a display character code value n then read the next character, if the next character = `N` then character is set to Newline, or if the next character = `T` then character is set to Tab, otherwise the character is set to the next character.

This facility enables newlines and tabs to be punched on cards.

F.4.9 Line Numbers - S19

`S19 = 0` each line has a line number on the listing. The line numbers start at 1 and are incremented by 1 at each newline character. Lines read from input by `MCATOM` are numbered starting from 1 again.

`S19 = 1` no line numbers are written on listing.

F.4.10 Rewind Control - S20

`S20 = 0` when the process finishes all output files, not called `OUTPUT`, are rewound.

`S20 = 1` no output files are rewound.

F.4.11 CP Time Output - S21

`S21 = 0` No action

`S21 <> 0` The value of `S21` is printed out, followed by the number of CP seconds taken by the job so far. `S21` is then set to zero. If 90% or more of the time available has been used then the process terminates immediately.

F.4.12 Stop Control - S22

`S22 = 0` no action

`S22 = 1` macro processing is terminated immediately with the message:

```
PROCESS STOPPED
```

`S22 = 2` macro processing is terminated immediately abnormally with the message:

```
PROCESS ABORTED
```

F.5 New Features

F.5.1 Macro Class

This facility has only been introduced into the LUCS implementation of ML/I and so users are warned of possible incompatibilities with other versions of ML/I.

The facility was introduced to speed up the recognition of text containing macro calls, inserts or skips. The standard method, as suggested in the *ML/I User's Manual* is to compare the evaluated and unevaluated forms. The method is based on the fact that the evaluated form is assumed to be different from the unevaluated.

To enable a direct test of a string to see if it contains any macro calls, skips or inserts a new class was introduced in the `BC` form of the conditional `MCGO` statement, classified by the letter `M`.

The following forms an amendment to the definition of the `BC` variant of the conditional form of `MCGO` in the *ML/I User's Manual*:

> "If `{`*arg C*`}` is the letter `M`, then a true value results only if `{`*arg B*`}` contains one or more macro calls, skips or inserts. The test is made after `{`*arg B*`}` has been evaluated."

For example, in the following simple expression macro it may be required to test if each argument contains any further nested calls:

```
MCDEF (+)
AS [MCGO L1 IF £A1. BC M
LOAD £A1.
MCGO L2 IF £A2. BC M
LOAD £A2.
. . .]
```

As with the example in the manual it is possible to test the argument only for macro calls, or skips or inserts. For example, to test if the argument involved any macro calls, the test might be written:

```
MCGO L1 IF MCNOSKIP MCNOINS £A1. BC M
```

F.5.2 MCATOM

This system function has only been introduced into the LUCS implementation of ML/I, and so users should beware of incompatibilities with other implementations.

Purpose

Function to access a string of atoms.

General form

`MCATOM ( [{arg A}, ? ] {arg B}, {arg C} )`

The left parenthesis is part of the macro name. It may optionally be preceded by spaces.

Examples

a. `MCATOM (ABC/XYZ,2,2)`
This function has value `/`

b. `MCATOM (123.456,3,2)`
This function has value `.456` as the second and third arguments are evaluated and the smaller signifies the first atom of the string required and the larger the last atom.

c. `MCATOM (THIS IS A TEST,5,10)`
This function has value `A TEST` as spaces (and newlines) are taken as atoms and if the larger argument is higher than the number of atoms in the text the end of the text terminates the search.

d. `MCATOM (1,1)`
This function has as value the next item in the source text. If the end of the source text is reached null atoms are returned.

e. `MCATOM (£A1.,P1,T3)`
The arguments are evaluated first.

f. `MCATOM (2,3)`
This function has as value the two atoms after the next atom in the source text. Note that once an atom has been read from the source text it cannot be reread later.

System Action

Let `L` be the number of atoms in `{`*arg A*`}`, or if `{`*arg A*`}` is absent the number of atoms left in the source text. Let `RB` be the result of evaluating `{`*arg B*`}` and `RC` be the result of evaluating `{`*arg C*`}`.

If `RB` is greater than than `RC` then `S` becomes `RC` and `D` becomes `RB-RC`, otherwise `S` becomes `RB` and `D` becomes `RC-RB`.

If `S` is less than one then `S` becomes one.

If `L` is greater than or equal to `(S+D)` the result of the function call is a string of atoms from the `S`th atom to the `(S+D)`th atom, otherwise it is a string of atoms from the `S`th atom to the `L`th atom.

Notes

a. the definition of an atom is given in the *ML/I User's Manual*.

b. The *ML/I User's Manual* states that it is not possible to pick up the atom immediately following a macro name unless it is followed by some predefined delimiter. If the macro is called from the source text this is now possible.

F.5.3 MCNUM

This system function has only been introduced into the LUCS implementation of ML/I, and so users should beware of incompatibilities with other implementations.

Purpose

Function to find the number of atoms of a string.

General form

`MCNUM ( {arg A} )`
The left parenthesis is part of the macro name, it may optionally be preceded by spaces.

Examples

a. `MCNUM (THIS IS A TEST)`
The result of this function call is 7.

b. `MCNUM (£A1.)`

System Action

This function was introduced to complement `MCATOM` as `MCLENG` does `MCSUB`. The value of the function is the number of atoms in `{`*arg A*`}`.