UNIVERSITY OF MANCHESTER

DEPARTMENT OF COMPUTER SCIENCE

MU5/MX3 HIGH-LEVEL LANGUAGES MANUAL

Second Edition - JULY 1974

Contents

Chapter 1 - Introduction: Running a high-level language program.

A simple program consists of some 'job control' lines, the program proper, and possibly some data, in the following form:

```
***A  JOB <USER><PASSWORD><JOBNAME>
**<COMPILER NAME>  ( )
   <program>
**ENTER  ( )
   <data>
**STOP()
***Z
```

where:

<USER> is the user name which must be a previously created authorised user.

<PASSWORD> is his password

<JOBNAME> consists of up to 6 characters which uniquely identify the job.

***L (in place of ***A) causes the program to be listed, with page and line numbers on the lineprinter.

<COMPILER NAME> is either ALGOL, AA, FORTRAN or AUTO.

<program> is the program text following the particular language conventions.

<data> is optional data.

N.B. JOB implies an MU5 job. For a job to be run on the 1905E JOB is replaced by JOB5E.

Example:

```
***A JOB FRED DERF NODDY
**ALGOL  ( )
begin
 . . .
 . . .                    <ALGOL program (MU5)>
end;
*END OF PROGRAM;
**ENTER().
1  2  3          <Data>
**STOP()
***Z
```

This program has only input stream 0 (the data) and output stream 0 which is usually a lineprinter.

Full details of how to run more complicated jobs with, for example, several input streams, computing or priority requirements other than the default, or on-line jobs are found in the MU5/MX3 Operating System Manual.

Details of the procedures which may be called by 'program control sequence' statements starting ** are found in the MU5/MX3 Library Manual.

Statements starting with * are 'compiler directives', and are described in the relevant chapter for each language.

Accuracy of Number Representation - MU5

INTEGER: stored as fixed-pt signed (I32) number in the range $-2^{31} \leq I < 2^{31}$

REAL: stored as floating-pt signed (R64) number in the form $m.16^{-e}$, where

m = mantissa = sign bit, followed by binary point, followed by 52 binary digits

and e = exponent (11 bits) = an integer in the range $-1024 \leq e < 1024$

(i.e. largest magnitude = 10^{1233} approx, & accuracy = 15 decimal digits approx)

COMPLEX (Fortran & AA): stored as 2 REALs

DOUBLE PRECISION (Fortran): stored as for REAL, but m has 104 binary digits (R128)

(i.e. accuracy = 31 decimal digits approx)

Accuracy of Number Representation - MX3 (1905E)

INTEGER: stored as fixed-pt signed number in the range $-2^{23} \leq I < 2^{23}$

REAL: stored as floating-pt signed number in the form $m.2^{-e}$, where

m = mantissa = sign bit, followed by binary point, followed by 37 binary digits

and e = exponent (9 bits) = an integer in the range $-256 \leq e < 256$

(i.e. largest magnitude = 10^{76} approx, & accuracy = 11 decimal digits approx)

The remainder of this manual is devoted to details for each specific high-level language.

Chapter 2 - MU5 ALGOL 60

2.1 Introduction

The ALGOL compiler implements the language defined in the ALGOL-60 REPORT, subject to the restrictions given in section 2.2 below, but with the addition of the ICL I/O procedures as used on Atlas and by U.M.R.C.C (see section 2.9).

2.2 Restrictions on Algol Report

(a) Integer labels not allowed.

(b) Own arrays must have constant bounds, and all own variables are common to the different levels of a recursive procedure - i.e. as though they were declared in the outermost block.

(c) Every procedure parameter must have a specification part.

(d) The specifications of all procedures used as actual parameters in the calls for a particular procedure must be consistent.

(e) The type of an expression of the form A↑B will always be real unless A is an integer variable and B a positive integer constant, in which case the result will be integer.

(f) All uses of an array formal parameter within the body of a procedure must be consistent with regard to the number of subscripts, and must be consistent with any actual parameters used in a call for the procedure.

(g) See also section 2.4 below.

2.3 Compiler Directives

(a) Commence with * and terminate with ; .

(b) May appear anywhere in Algol program (as complete statement).

(c) Take immediate effect during the first pass of compilation (when the program text is read).

(d) Spaces and newlines are ignored (as elsewhere).

(e) Upper case letters only.

(f) Only the first word is obligatory - any others are ignored.

2.4 Algol Modes

*RESTRICTED ALGOL MODE;

Semi-colons not allowed in strings - string will be terminated (faulty) and syntax analysis continued as for normal program statements.

*FULL ALGOL MODE;

Semi-colons allowed in strings (as Algol Report) - if closing string quote is not recognised, then the rest of the program will be taken in as part of the string.

Default option :- RESTRICTED

2.5 Delimiter Mode

All delimiters must be in upper case letters and enclosed in primes, e.g., 'BEGIN'. This is the only mode available.

2.6 End of Program

Last 'END' must be followed by ;

This must then be followed by *END; or *END OF PROGRAM; if desired.

2.7 Monitoring

(a) General - see compiler writers manual.

(b) Comments :

(i) 'COMMENT' will be monitored (but not faulted) if it contains a delimiter

(ii) 'END'-type comment will be monitored even if O.K., and additionally if it contains a delimiter (not faulted)

(iii) 'COMMENT' appearing in an illegal position will be faulted.

2.8 Character Representations

ALGOL	ISO	ALTERNATIVES	
A-Z	A-Z		
a-z	a-z		
0-9	0-9		
+	+		
-	-		
×	*		
/	/		
÷		'DIV'	'/'
↑	↑	'EXP'	
<	<	'LT'	
≤		'LE'	
=	=	'EQ'	
≥		'GE'	
>	>	'GT'	
≠		'NE'	
≡		'EQUIV'	
⊃		'IMPL'	

ALGOL	ISO	ALTERNATIVES	
∨		'OR'	
∧		'AND'	
¬		'NOT'	
,	,		
.	.		
⏨	@		α '10' &
:	:		
;	;		£ π
:=		:=	
␣	$		$ % ²
(	(		
)	)		
[	[		
]	]		
⌜		"(	'(' ¢
⌝		)"	')' ¬
	"		½

Notes

1. The preferred characters are the ISO or first alternative - the second alternatives are only included to allow for existing programs.

2. In compound characters, / and | are equivalent, and the ISO characters $ and ; may be formed as compound characters.

2.8.1 Strings

(a) During the input of strings $ is converted to space (alternatives also), and the alternatives for string quotes are converted to special characters. No other conversions, but sub-strings will be further processed during perm procedures such as 'write text' and 'code' (see Section 2.9).

(b) Within strings, a 'basic symbol' (Algol Report 2.6.1) may be any ISO character subject to the conventions used above for ␣ , ⌜ and ⌝ . [Similarly in comments].

2.9 Permanent Procedures

2.9.1 Standard & Transfer Functions

These are implemented as described in the Algol Report 3.2.4 & 3.2.5 (with names as given), but note that trig functions always work in radians, and 'principal value' is in the range $-\pi/2$ to $+\pi/2$.

2.9.2 Input & Output

```
procedure select input (stream); value stream; integer stream;
integer procedure readch;
```

Reads next symbol from input and yields ISO code.

```
integer procedure nextch;
```

'Reads' next symbol without advancing input pointer.

```
real procedure read;
```

Reads next number from input, punched in the format defined in the Algol Report 2.5.1.

```
procedure select output (stream); value stream; integer stream;
procedure printch (symbol); value symbol; integer symbol;
```

Prints ISO symbol whose code is 'symbol'.

```
procedure print (x,m,n); value x,m,n; real x; integer m,n;
```

Prints value of x:

(a) $m \neq 0$, fixed point form with m places before, and n places after, the decimal point (rounded).
N.B. if $n = 0$, dec pt is suppressed.

(b) $m = 0$, floating point form in range $1.0 \leq x < 10.0$ with n dec places (rounded) and suitable exponent.

N.B. Total number of characters =

m+n+4	if	$m \neq 0$ & $n \neq 0$
m+3	if	$m \neq 0$ & $n = 0$
n+11	if	$m = 0$

```
procedure space (n); value n; integer n;
        Prints n spaces.
procedure newline (n); value n; integer n;
        Prints n newlines
procedure paperthrow;

procedure write text (string); string string;
        Prints the string of characters as given between the string
        quotes except for sub-strings 'one level down' which obey
        the following convention:
        <substring>  :=''( <units> )''
        <units>      :=  <unit> | <unit> <units>
        <unit>       :=  P | <N> C | <N> S
        <N>          :=  <digit> | <null>
        and P indicates 'paperthrow'
            C indicates 'newline'
            S indicates 'space'
        e.g. write text ( ''(HELLO  ''(2C5S)'' GOODBYE)'' );
             is equivalent to
             write text ( ''(HELLO)'' );    newline(2);
             space(5);     write text ( ''(GOODBYE)'' );

integer procedure code (symbol); string symbol;
        Yields the ISO code for the single ISO symbol given
        in string quotes.
        Also :   code ( ''($)'' ); gives ISO 'space' code
                 code ( ''(EL)'' ); gives ISO 'newline' code
```

2.10 Library Procedures

CTL Library procedures may be called as for 'permanent' Algol procedures, but full stops in the names should be omitted (to give valid Algol names).

The correspondence between Library parameters and corresponding Algol formal parameter types is as follows:-

Library	Algol	
I32	integer	called by value
R64	real	called by value
S	integer, real, Boolean, array, string	called by name
LABEL	label	called by name or value
U64/I64	string	of 8 chars only
PROC	procedure	(possibly with type)

In the case of function procedures the result types have the following correspondence:-

Library	Algol
I32	integer
R64	real

Library procedures having other parameters or results (e.g. S-type result) cannot be called from an Algol program.

Chapter 3 - MU5 FORTRAN

3.1 Introduction

The FORTRAN compiler implements the language described in USAS X3.9-1966 USA STANDARD FORTRAN. In so far as it is possible to do so, the compiler will fault any aberration from the definition, and in such cases as it is not possible, the action of the prohibition will be defined below.

3.2 Compiler Directives

(a) Commence with *

(b) Occupies cols 7 to 72 with continuations

(c) May appear anywhere in Fortran program, but certain directives (e.g. I/O) should be at the start if possible.
[N.B. Program Control Sequence statements must start with ** in cols 1 & 2]

3.3 Program Layout

The program unit(s) making up the program may be in any order, but must be followed by the compiler directive *END.

3.4 Monitoring

See compiler writers manual.

3.5 Implementation

3.51 Accuracy (ANSI 1.2.2)

See Introduction to this Manual

3.52 Double Precision Evaluate (ANSI TABLE 1)

Expression evaluated in Double Precision

3.53 STOP Statement

The message

STOPPED AT LINE -

n (octal string)

is output on the monitor output and the program is terminated by calling the library procedure

STOP (I32)

P1 = octal string as integer or 0

3.54 PAUSE Statement

The message

PAUSED AT LINE -

n (octal string)

is output on the monitor output and then a message is sent to the supervisor by

SEND.MESSAGE.TO.SUP (S, S, I32, I32, I32)

P1 = descriptor of short message PAUSE (or n)

P2 = vector descriptor (V32)

word 0 -

1 -

2 1

P3 = 16 await FREE.PROCESS command from supervisor

P4 = -1 short message

P5 = irrelevent

The user may take action accordingly.

(See MU5/MX3 Operating Systems Manual 3.3.3 for details of BATCH supervisor)

3.55 DATA Initialisation

A DATA statement within a procedure causes the variables to be initialised each time the procedure is entered (permitted by ANSI 10.2.6).

3.56 FORMAT

A group of field descriptors has the same basic syntactic format as the format specification except that it may only contain field descriptors and basic groups of descriptors. Basic groups similarly except that they may only contain field descriptors.

3.57 REWIND Statement

This sets the stream position to the start of the specified stream. Note that in some cases this may not be the same as the start of the input or output.

3.58 BACKSPACE Statement

This moves back the stream position to its previous position i.e. just after previous control character.

3.59 ENDFILE Statement

An end of text character %04 is output on the specified stream.

3.6 Calling Library Procedures

This is a facility which is NOT in standard FORTRAN. Any program making use of it is thereby not standard and will not run on another system unless it also provides the same facility.

LIB CALL <LIB.NAME> (<PARAMETER.LIST>)[,<NAME> | <NIL>]

The names of the library procedures are given in the appropriate manual. If the procedure is a function then the result is assigned to the program variable specified. The type must correspond i.e.

proc result	variable type
I32	integer variable reference
R64	real variable reference
R128	double precision variable or complex variable reference
U64	real variable reference or array name
U32	integer variable reference

The assignment of a U64 to an array name redefines the array base to this value during the remainder of the procedure.

The parameter lists are defined for each procedure and the interface of types is given below.

parameter type	actual parameter
I32	integer expression
R64	real expression
R128	double precision expression or complex expression
U32	integer variable reference or integer constant
U64	integer variable reference, real variable reference, integer or real constant or array name
S	real variable reference, array name, Hollerith string

Although there are no 32 bit vectors since all FORTRAN vectors use the 64 bit storage unit, integer vectors can be used.

There is no protection by the FORTRAN system against a user who interferes with the program or surrounding system (especially input/output) by use of library procedures.

3.7 Input/Output System Interface

The input/output system used in MU5 is based on 8 input and 8 output streams. Streams 0 are always defined and output 0 is used as the monitor output for the compiler and system. The user may also use this output stream for the program output. The other streams, if required, must be defined by the program control sequence statements.

The FORTRAN input/output system is not based on streams but on unit numbers. The correspondence between streams and unit numbers is done by the Fortran system, as directed by the user.

```
*UNIT (I, I, I, I)

P1 = 0 :        input
   = 1 :        output
P2 = stream number
P3 = unit number
P4 = logical record length:-
     0 default setting, 80 input, 120 output
```

by default the stream number unit number correspondence is:-

```
Input streams 0 - 7
        unit  1 - 8     logical record length 80

Output streams 0 - 7
        unit   9 - 16   logical record length 120
```

If a unit number is defined which was a default setting of this unit number then the default stream does not correspond to any unit number.

```
e.g.    *UNIT (0, 1, 3, 0)
```

Input stream is unit 3
and input stream 2 has no longer a default unit number, and therefore must be explicitly specified if required.

The unit number may be any positive integer constant.

Chapter 4 - MU5 ATLAS AUTOCODE

4.1 Introduction

The ATLAS AUTOCODE compiler implements the language described in 'A Definition of Atlas Autocode' (J. S. Rohl, Jan 1970), subject to the restrictions given in section 4.2 below.

4.2 Restrictions on Definition Manual

(a) No machine code (Section 4.10 of manual)

(b) Maximum of 16 textual levels (Section 5.5.3.1)

(c) Maximum depth of cycles = 40 (Section 5.6.3)

(d) array bound check is ignored, since this is performed by hardware (Section 5.8.1)

(e) A call for a routine for which no spec has been given is assumed to be a call for a 'perm' OR 'library' routine (Section 7 and 4.8 below)

4.3 Compiler Directives

Since many of these are already included in the definition as 'delimiters' the prefix * is not used, and all Directives appear as delimiters.

4.4 Delimiter Modes

'NORMAL DELIMITERS'	e.g.	begin
'UPPER CASE DELIMITERS'	e.g.	BEGIN
'SINGLE CASE DELIMITERS'	e.g.	'BEGIN'

Default option :- SINGLE CASE

4.5 Production Runs

The directive 'PRODUCTION RUN' causes the compiler to omit some of the standard checks (e.g. when compiling cycle statements). It should normally only be used with a thoroughly tested program.

4.6 Monitoring

See Compiler Writers' Manual.

4.7 Character Representation

AA	ALTERNATIVE	AA	ALTERNATIVE
A-Z		,	
a-z		.	
0-9		α	∂
+		:	
-		;	
*		?	
/		)	
↑	↑	(	
<	'LT'	π	∉
≤	'LE'	'	
=	'EQ'		
≥	'GE'	₳	←
>	'GT'	$	$
≠	'NE'	$	
|	:	¦	

Notes

1. In compound characters, / and | are equivalent.
2. Delimiters such as 'LT' may also appear as lt or LT according to the current delimiter mode.

4.8 Library Procedures

CTL Library procedures may be called as for 'permanent' AA routines, but full stops in the names should be omitted (to give valid AA names), and parameterless procedures must be called without the empty ( ).

The correspondence between Library parameters and corresponding AA formal parameter types is as follows:-

Library	AA
I32	integer
R64	real
R128	complex
S	integer name real name complex name array name
PROC	routine/ type fn

In the case of functions the result types have the following correspondence:-

Library	AA
I32	integer fn
R64	real fn
R128	complex fn

Library procedures having other parameters or results (e.g. S-type result) cannot be called from an AA program.

Chapter 5 - MU5 AUTOCODE

The MU5 AUTOCODE compiler implements the language defined in the 'MU5 AUTOCODE MANUAL'. This includes:-

(a) Punching conventions

(b) Delimiter conventions

(c) Compiler directives

Chapter 6 - MX/3 ALGOL 60

6.1 Introduction

The MX/3 ALGOL Compiler implements most of the language defined in the ALGOL - 60 REPORT, subject to the restrictions given in section 6.2 below. Input and Output is performed by using the MX/3 library (see the MU5/MX3 library manual and section 6.10), which includes procedures compatible with those used by I.C.L, as used on ATLAS, and by U.M.R.C.C.

6.2 Restriction on the ALGOL REPORT

The restrictions on the use of the MX/3 ALGOL compiler are in general more severe than those of the MU5 ALGOL Compiler:-

a) Integer labels not allowed.

b) No 'OWN' variables.

c) Every procedure parameter must have a specification part.

d) Some Procedure parameter types are not yet implemented. These are:-

   1) Procedure parameters.

   2) Label parameters.

   3) Switch parameters.

e) an expression of the form A↑B will always be real unless A is an integer variable and B is a positive integer constant, in which case the result will be integer.

f) The switch facility is not implemented as in the ALGOL REPORT. If a switch element forms part of a switch list, and when referred to its subscript is out of range, a fault condition is generated. In the case of a simple jump via a switch list, the instruction will be treated as a dummy instruction if the subscript is out of range, i.e. control will be passed to the following instruction.

6.3 Compiler Directives

a) Commence with 'DIRECTIVE' and terminate with ;.

b) May appear anywhere in the ALGOL program.

c) Take effect from the point of their appearance (at compile time).

d) Spaces and newlines are ignored as elsewhere.

The majority of directives are for compiler debugging, but the following is of interest to users:-

'DIRECTIVE' ARRAY BOUND CHECK ON/OFF;

This may be switched on or off. The default option being on.

6.4 Compilation Mode

Semicolons are accepted within strings, but will be monitored.

6.5 Delimiter Mode

Only card delimiter mode, as shown below, exists on this compiler, e.g.,

'BEGIN'

6.6 End of Program

The last 'END' must be followed by a ; .

N.B. In the MX/3 Compiler, the recognition of any P.C.S (i.e., **) command will cause compilation to terminate and a return to the P.C.S to be effected.

6.7 Monitoring

Delimiters occuring within a 'COMMENT' sequence will be monitored.

6.8 Character Reprsentations

These are as for MU5 ALGOL (see section 2.8) except that the following are not allowed:-

'_' '<' '>' ↓ ↑

6.8.1 Strings

Within strings a 'basic symbol' (ALGOL REPORT 2.6.1) may be any I.S.O character subject to the following conventions:-

a) During input % and $ are converted to 'SPACE'.

b) Within inner string quotes (substrings) only the following formats are allowed and will be converted during input of the string:-

1) <N> C = <N> newlines

<N> S = <N> spaces

P = newpage

where <N> is an optional unsigned integer.

6.9 Permenant Procedures

The MX/3 ALGOL Compiler has no permenant procedures of its own, all I/O and mathamatical functions being performed by user calls for procedures within the MX/3 library (see section 6.10).

6.10 Library Procedures

These appear as normal ALGOL procedure calls within an ALGOL program. However the full stops should be omitted from the names in the MU5/MX3 library manual to give valid ALGOL names.

The correspondence between library parameters and ALGOL formal parameter type is as follows:-

Library	ALGOL	
I32	'INTEGER'	called by value
R64	'REAL'	called by value
S	'INTEGER' / 'REAL' / 'BOOLEAN' / 'ARRAY' / 'STRING'	called by name
LABEL	see section 6.2	
U64/I64	'STRING' of $\leq$ 6 characters	
PROC	see section 6.2	

In the case of function procedures the result types have the following correspondence :-

Library	ALGOL
I32	'INTEGER'
R64	'REAL'

Library procedures having other parameters or results (e.g. S-type result) cannot be called from an Algol program.