



CMS Modula-2

Programming Language for
68000 Based Controllers

Features

- ❑ Very easy to use
- ❑ High speed of execution
- ❑ Simple but powerful debugger
- ❑ Fully Multi-tasking
- ❑ Fully Recursive
- ❑ Extendible
- ❑ Interactive
- ❑ No run time License fee
- ❑ Rich set of control statements
- ❑ Several variable types
- ❑ Local and global variables
- ❑ Extensive error checking by the compiler
- ❑ Can use 'user generated' assembler routines
- ❑ Deals with signals and interrupts

Description

CMS Modula-2 is an interactive programming language which is as easy to use as Basic, but has the structure of Pascal or 'C'. Modula-2 is a simple language to learn with many of its features descending directly from Pascal. Extra library words have been added to interface directly with the hardware, making the reading of ADCs or the control and testing of digital lines a simple matter. The whole essence of the language is to produce a programming tool that can be used by non computer skilled personnel. To this effect the language is very interactive. Statements can be typed directly from the keyboard and they will execute immediately. The programs under development can be stopped anywhere and debugging can take place, again directly from the keyboard.

The language is completely multi-tasking so that several programs can be run concurrently. The procedures and functions are entirely recursive with both local and global variables.

The language is extendible and can use program libraries to add new word definitions. If a particular application requires a special feature, this can be linked to the language's dictionary making it appear as a standard procedure or function.

Development can take place using a standard PC, by using a word processor, to pre-

pare the program text file. Once ready, a pop up terminal is invoked and the text file is down loaded into RAM on the processor card where it can be tested. The final program can easily be made 'turnkey' and blown into EPROM.

Statements

Declaration

```
MODULE <name>;
PROCEDURE <name> [ ( <param>
{,<param>} ) ] [ <:type> ];
RETURN [ <exp> ];
IMPORT <library module>;
FROM <library module> IMPORT
<procedure> {,<procedure> };
CONST <id> : <type> | <ARRAY>
[ <sub-range> ] OF <type>;
VAR <id> {,<id>} : <type> | <AR-
RAY> [ <sub-range> ] OF <type>;
```

```
BEGIN
END <name>.
```

Editing

```
Load <file_name>
Save <file_name>
C <file_name>
M2 <file_name>
```

```
New
Debug
List [ <name> ]
```

```
Mdir
Lock
UnLock
TurnKey
```

Structure

```
FOR <id> := <exp> TO <exp> [ BY
<exp> ] DO <statement list> END;
REPEAT < statement list > UNTIL
<exp>;
WHILE <exp> DO <statement list>
END;
LOOP <statement list> [ EXIT ]
END;
IF <exp> THEN <statement list>
ELSIF <exp> THEN <statement
list>
```

Example Program

```
MODULE Factory;(* Control and Monitor a heater
*)
FROM Terminal IMPORT Write, WriteLn, WriteInt,
ReadChar;
FROM ADIO IMPORT Adc, InCh;
CONSTsize = 1000;(* Global Variables *)
VARreply : CHAR;
heat : INTEGER;
flag : BOOLEAN;
(* Control and monitor a heater *)
BEGIN
Write("Machine Monitoring");
WriteLn;
StartProcess( Control, size );
StartProcess( Panic, size );
LOOP
Write("Input Instruction ");
```

```
(* These routines will run concurrently *)
PROCEDURE Control;
CONST oven = 1;(* analog i/p 1 *)
heater= 2; (* digital line 2 *)
highLimit = 70;(* 70 degC max*)
lowLimit = 65; (* 65 degC min *)
max = 100;(* overheat point *)
tenseconds = 1000; (* execution rate *)
BEGIN
LOOP
heat := Adc( oven );(* read temp. *)
IF heat > highLimit THEN (* test limits *)
TurnOff( heater );
IF heat > max THEN (* overheating *)
Colour( red );
Write("Overheating");
WriteLn;
ShutDown
END
ELSIF heat < lowLimit THEN
TurnOn( heater )
END;
IF flag THEN RETURN END;
```

```
PROCEDURE PrintOutTemp;
VARx : INTEGER;
BEGIN
x := heat;
Colour( green );
Write("The temperature is ");
WriteInt( x,0 );
Write("degC");
WriteLn
END PrintOutTemp;
```



ELSE <statement list>
END;

CASE <exp> OF

<label> : <statement list>

{ | <label> : <statement list> }

[ELSE <statement list>]

END;

Library

Write(<exp> [,<exp>]);

WriteLn;

ReadLn(<string>);

ReadChar(<char>);

<id> := <exp>

INC(<id>);

DEC(<id>);

Numeric Operators

*multiply

DIVdivide

MODmodulus

>>shift right

<<shift left

ANDlogical AND

ORlogical OR

EorLogical exclusive OR

=relational equals

<>relational not equals

<=relational less than
or equals

>=relational greater
than or equals

<relational less than

>relational greater than

String Operators

+concatenate

<>relational not equals

=relational equals

>relational dictionary
greater than

<=relational dictionary
less than or equals

"<string constant>"

'<string constant>'

'<char>'

Conversion Functions

CHR(<numeric>)

ORD(<char>)

FLOAT(<numeric>)

TRUNC(<real>)

Len(<string>)

Val(<string>)

Str(<string>)

Hex(<string>)

Indirection

ADR(<id>)returns storage
address

!(<adr>)integer

?(<adr>)byte

\$(<adr>)converts data at
address to a string

Multi-tasking

StartProcess(<name>,<space>);

Init(<id>);

WAIT(<id>);

SEND(<id>);

Sleep(<time>);

Variable types

INTEGER

CARDINAL

CHAR

BOOLEAN

REAL

BITSET

POINTER

ADDRESS

BYTE

WORD

SIGNAL

ARRAY [sub-range] {,[sub-range] }
OF <type>

Constants

TRUEBoolean

FALSEBoolean

1234Decimal

0d0ahHexadecimal

'string'string

'c'character

Functions

(<exp>)

+

NOT

ABS()

ODD()

Even()

Sgn()

Rnd()



Cambridge Microprocessor Systems Limited,

Unit 17 - 18 Zone 'D',
Chelmsford Road Ind. Est.,
Great Dunmow,
Essex, U.K. CM6 1XG.

Telephone 01 371 875644
FAX 01 371 876077

E-Mail sales@cms.uk.com

Web Site http://www.cms.uk.com

Order Codes

J-307M	307-Module Development Pack+ Multi Minos License
J-030	Micro-Module Modula-2 Development Pack+ Multi Minos License
J-030S	Micro-Module Modula-2 Starter Pack+ Single Minos License
J-038	Micro-Midget Modula-2 Development Pack+ Multi Minos License
J-038S	Micro-Midget Modula-2 Starter Pack+ Single Minos License
DUP-130	Upgrade from single to unlimited Minos copy License for Micro-Module or Midget

This same package will also be available for future controller products
manufactured by Cambridge Microprocessor Systems Ltd.

MODULA2 970224

01 371 875644



Cambridge Microprocessor Systems Ltd.