

softMC Training – Module 4

# Language (MC-Basic)



# Contents

- MC-Basic Syntax
- Instructions/Commands
- Constants
- Variables
- Data Types (Basic, Complex and Motion)
- Expressions
- Math
- Strings
- Virtual Entry Station

# Introduction

- softMC is programmed in MC-Basic (Motion Control BASIC)
- MC-Basic = Standard BASIC programming language enhanced for multi-tasking and motion-control

# Syntax

# Syntax

- Line oriented

- End of a line is considered the end of a command
- Line length limited to 80 characters

```
Move A1 100000 VCruise = 1000 Acceleration = 10000  
Deceleration = 10000 Absolute = 1 VelocityFinal = 500
```

**Invalid.** All the code for a command must be on one line and must not contain more than 80 characters

- Case insensitive

```
Dim MC_CLASS as String  
mC_cLaSS = "MC Class"
```

**Valid.** All MC-Basic commands, variable names, file names and task names are case insensitive.

- Exception: Strings are case sensitive

# Instructions / Commands

- Comments
- Declarations - Memory Allocation
- Assignment
- Printing

# Comments

- Comments are preceded by an apostrophe '
- Comments are preceded by the command **Rem**

```
    Sys.En = on           'Enable the system
    Al.En = on           'Enable Axis A1
    Al.StartType = Immediate
    Sleep 1000           'Sleep one second

    Sys.Motion = on     'Allow system motion
    Al.Motion = on     'Allow motion in axis
    For sample_int = 1 To 10 'Loop 10 times

rem  Move A1 to 10 position unit(s)
    Move A1 10 Absolute=0
    Sleep 2000           'Sleep two seconds

rem  Move A1 back to start
    Move A1 -10 Absolute=0
    Sleep 2000           'Sleep two seconds
Next sample_int
```

# Declarations

- Declarations allocate memory for variables and system elements
- Simple
  - Example: long integer
- Complex
  - Examples: cam table, group
- Examples

`Common Shared`

`Dim Shared`

`Dim`

# Assignments

- Assign a new value to a variable

- Syntax

*<value> = <expression>*

- Example

**x = y + 1**

# Printing

- Print messages from within a program or in terminal

```
Print "Hello World"
```

- Print to COM port/file (send ASCII characters)

```
Print#1 "Hello World!"
```

- Formatted print (PrintU or PrintUsing)

```
PrintU "The number is #, # ";i1,i2
```

When printed, the first hashtag will be replaced by the value of i1 and the second hashtag will be replaced by the value of i2. i1 and i2 will have one digit only.

```
PrintUsing "The Current Position is: #.##"; PFB
```

When printed, the value of PFB will be given as a number with 2 decimal places.

- Print to TCP/IP sockets

```
PrintU#1 "The number is #, # ";j1,j2
```

# Constants / Variables

# Constants and Variables

- The names of all constants, variables and system elements:
  - Must start with an alphabetical character (a-z, A-Z)
  - May contain up to 32 alphabetical characters, numbers (0-9) and underscores ( \_ )

# Constants

- Constants can be written in various formats:
  - Decimal  
131
  - Hex  
0x83
  - Binary  
0b10000011

# Literal Constants

- Literal constants are reserved words that have a fixed value
- **Blue** is the default color of recognized literal constants
- Examples

**PI**

= 3.141592653590001

**Off**

= 0

**On**

= 1

**False**

= 0

**True**

= 1

**Linear**

= 0 – Used in setting motion type

**Rotary**

= 1 – Used in setting motion type

# Variables

- Variables are declared
- Variable names are assigned by user
- System, task or local
- 32-bit signed integer
- Double precision floating point
- Arrays

# Variables

- **System** (global) variable – recognized by all tasks in the system
  - Syntax  
`Common Shared <variable> as <type>`
  - Example  
`Common Shared Sys_Var1 as Long`
- **Task** variable – recognized only by the task in which it is declared
  - Syntax  
`Dim Shared <variable> as <type>`
  - Example  
`Dim Shared Task_Var1 as Double`
- **Local** variable – recognized only within a program, subroutine or function
  - Syntax  
`Dim <variable> as <type>`
  - Example  
`Dim Y as Double`

# Variables – Assignments - Examples

- Assign a value to a variable

```
MyVariable = 27
```

- If a **Double** (floating point) value is assigned to a **Long** variable, the fractional part of the double value is truncated and the integer part is assigned to the long variable

```
-->Common Shared dposition as Double
-->Common shared lposition as long
-->dposition = 3.456789
-->?dposition
3.4567890000000000e+00
-->lposition = dposition
-->?lposition
3
```

# Data Types

# Basic Data Types

- **Numeric – Long:** 32-bit integer
- **Numeric – Double:** 64-bit floating point
- **String:** ASCII or UTF-8 string, unlimited length

# Complex Data Types

- Structure
- Semaphore
- Array
  
- User Error/Note: ASCII string and a unique integer exception number
  
- Generic axis
- Generic group

# Structures

- A structure is a data type used for storing a list of variables of different types within one variable.
  - Can be defined in Config.prg and in libraries
  - Precedes the Program block

```
Type <variable>  
<variable> as <type>  
<variable> as <type> {<of> <robot_type>}  
...  
End Type
```

- A structure element is addressed through the structure's name and the arrow sign (->)

```
<structure>{ []... }-><element>{ [] }
```

# Motion Data Types

- CAM Table
- PLS (programmable limit switch)
- Compensation Table
  
- Point – Robot Joint
- Point – Robot Location
  
- Moving Frame

# Expressions

# Expressions

- Expression: a combination of constants and variables with operators to produce a single value.

= is used for assignment

**X = Y + Z \* A / B**

- Condition: a logical expression that (when evaluated) is :

True if the result is not zero and

False if the result is zero.

= is used for comparison (logical operator)

**If X = Y + Z \* A / B Then ...**

- binary-operator operand

**x + y**

- unary operator operand

**+ y**

# Math

# Operators – Arithmetic

In order of precedence:

Parentheses ( )

<b>Operation</b>	<b>Symbol</b>	<b>Relative Precedence</b>
Exponentiation	^	1
Negation/N/A minus	-	2
Multiplication	*	3
Division	/	3
Modulus	MOD	4
Addition	+	5
Subtraction	-	5

# Math Functions

- Abs(x)
- Atn (x/y) in radians, between  $\pm\pi/2$
- Atan2(x, y) in radians
- Cos(x), Sin(x), Tan(x) in radians
- Exp(x), Log(x) natural log
- Sgn(x) Sign
- Sqrt(x) Square root
- Round(x) Round to nearest even

# Binary Operators

- Relational

Operation	Symbol	Relative Precedence
Equality	=	1
Inequality	≠	1
Less than	<	1
Greater than	>	1
Less than or equal to	<=	1
Greater than or equal to	>=	1

- Logical

Operation	Symbol	Relative Precedence
Complement	NOT	1
And	AND	2
Or	OR	3
Exclusive Or	XOR	4

- Bit-wise

Operation	Symbol	Relative Precedence
Bitwise complement	BNOT	1
Bitwise And	BAND	2
Bitwise Or	BOR	3
Bitwise Exclusive Or	BXOR	4

# Strings

# String Operations

- Concatenation
- Compare
- Find in string
- Extract
- Convert to/from number
- Convert to/from upper/lower case

# String Functions

ASC(S,I)	Returns an ASCII character value from within a string, S, at position, I.
BIN\$(X)	Returns the string representation of a number (X) in binary format (without the Ob prefix).
CHR\$(X)	Returns a one-character string corresponding to a given ASCII value, X.
HEX\$(X)	Returns the string representation of a number (X) in hexadecimal format (without the 0x prefix).
INSTR(I,SS,S)	Returns the position, I, of the starting character of a substring, SS, in a string, S.
LCASE\$(S)	Returns a copy of the string, S, passed to it with all the uppercase letters converted to lowercase.
LEFT\$(S,X)	Returns the specified number, X, of characters from the left-hand side of the string, S.
LEN(S)	Returns the length of the string, S.
LTRIM\$(S)	Returns the right-hand part of a string, S, after removing any blank spaces at the beginning.
MID\$(S,I,X)	Returns the specified number of characters, X, from the string, S, starting at the character at position, I.
RIGHT\$(S,X)	Returns the specified number of characters, X, from the right-hand side of the string, S.
RTRIM\$(S)	Returns the left-hand part of a string, S, after removing any blank spaces at the end.
SPACE\$(X)	Generates a string consisting of the specified number, X, of blank spaces.
STR\$(X)	Returns the string representation of a number, X.
STRING\$(X,{S},{Y})	Creates a new string with the specified number, X, of characters, each character of which is the first character of the specified string argument, S, or the specified ASCII code, Y.
UCASE\$(S)	Returns a copy of the string, S, passed to it with all the lowercase letters converted to uppercase.
VAL(S)	Returns the real value represented by the characters in the input string, S.

# System

# System Parameters

- Provide information about the system
- Some can be used in expression processing
- Examples

`System.Time`      ``set/get the time`

`System.Date`      ``set/get the date`

`System.Clock`      ``get the system clock (in milliseconds)`

# Virtual Entry Station (VES)

# Virtual Entry Station

## VESExecute

- Translates strings into commands
- Creates a virtual terminal
- Interprets commands when user is not using API

**END**