ROBOBASIC Command Instruction Manual v2.10

Notice

This manual explains the commands used in roboBASIC. Hitec is not responsible for any misuse that may occur. This manual can be changed without notice to improve performance of the related product

- Index -

Chapter 1

Command Summary

for roboBASIC

Command summary

RoboBASIC is an exclusive programming language designed for controlling robots. With roboBASIC, commands that are needed to control a robot have been added to the general BASIC programming language.

➋ Symbol means that this command can be executed only in MR-C2000 series controller

➌ Symbol means that this command can be executed only in MR-C3000 series controller.

Commands related with declaration/ definition

DIM	Declare variable
AS	Assign variable when declaring variable
CONST	Declare constant
BYTE	Assign as byte type when declaring variable
INTEGER	Assign as integer type when declaring variable

Flow control commands

IF	Begin conditional statement
THEN	Execute the next statement when condition of conditional statement is true
ELSE	Execute the next statement when condition of conditional statement is false.
ELSEIF	Begin another conditional statement.
ENDIF	Finish conditional statement
FOR	Begin loop statement.
TO	Assign the repetitive range for a loop statement.

	NEXT	End loop statement
	GOTO	Split the flow of the program.
	GOSUB	Call up a sub routine
	RETURN	Return from a sub routine
	END	Finish the execution of the program.
	STOP	Stop the execution of program.
	RUN	Run program continuously.
	WAIT	Wait until program has completed.
	DELAY	Delay execution of program for set period of time.
➋	BREAK	Pause program execution and switch to debug mode.

Digital signal input and output command

	IN	Read signal from input port.
	OUT	Send signal to output port
	BYTEIN	Read byte signal from input port.
	BYTEOUT	Send byte signal to output port.
➋	INKEY	Incoming key from the input port.
	STATE	Status of the output port
	PULSE	Send pulse signal to the output port.
	TOGGLE	Reversing the status of output port.
➌	KEYIN	Receive analog keypad input.

Command for memory

PEEK	Read data from the controller RAM.
POKE	Write data to controller RAM.
ROMPEEK	Read data from controller's external EEPROM RAM.
ROMPOKE	Write data to controller EEPROM RAM.

Command for LCD

	LCDINIT	Initialize LCD module.
	CLS	Clear all characters in LCD module.
	LOCATE	Set letter placement in LCD module.
	PRINT	Display letter in LCD module.
	FORMAT	Set type format displayed on LCD module
	CSON	Make cursor appear on LCD module.
	CSOFF	Hide cursor on LCD module.
	CONT	Set letter contrast in LCD module.
	DEC	Output decimal numeral to LCD.
	HEX	Output hexadecimal numeral to LCD.
③	BIN	Output binary numeral to LCD.

Operand related operation

	AND	Use the logical AND conditional expression.
	OR	Use the logical OR conditional expression.
	MOD	Calculating module for arithmetic operation.
	XOR	Use the logical XOR conditional expression.
③	NOTReversing all bit.	

Command for motor control

	ZERO	Setup the 0-point(neutral angle) of servo.
	MOTOR	Turn on output port of servo.
	MOTOROFF	Turn off output port of servo.
	MOVE	Operate several servos at the same time.
	SPEED	Set the servo speed.
②	ACCEL	Set the servo acceleration.
	DIR	Set servo motor direction.

	PTP	Turn simultaneous control operation on/off.
	SERVO	Control the servo.
	PWM	Set the pulse width control for a DC motor.
②	FASTSERVO	Operate servo at maximum speed.
③	HIGHSPEED	Turn the servo fast mode on/off.
③	MOVEPOS	Move servo group declared by POS.
③	POS	Set a specific pose for the robot.
③	FPWM	Change pulse width and frequency
③	MOVE24	Operate all 24 servo at the same time.
③	INIT	Set the initial motion pose
③	MOTORIN	Read the present position value of servo.
③	AIMOTOR	Setup for using AI motor.
③	AIMOTOROFF	Cancel AI motor.
③	AIMOTORIN	Read the present position value of AI motor.
③	SETON	Setup for using the “setup function”
③	SETOFF	Cancel the “set up function”.
③	ALLON	Setup functions for all servos.
③	ALLOFF	Cancel the setup function for all servo.
③	GETMOTORSET	Read present value of servo and keep the current position.

Parameter assigning motor group

③	G6A	Assign #0~#5 servos to group A.
③	G6B	Assign #6~#11 servos to group B.
③	G6C	Assign #12~#17 servos to group C.
③	G6D	Assign #18~#23 servos to group D.
③	G6E	Assign #24~#29 servos to group E.
③	G8A	Assign #0~#7 servos to group A.
③	G8B	Assign #8~#15 servos to group B.
③	G8C	Assign #16~#23 servos to group C.
③	G8D	Assign #24~#31 servos to group D.

③	G12	Assign #0~#11 servos.
③	G16	Assign #0~#15 servos.
③	G24	Assign #0~#23 servos.
③	G32	Assign #0~#31 servos.

Command for sound controlling

②	BEEP	Make warning sound with PIEZO.
②	SOUND	Make frequency sound with PIEZO.
②	PLAY	Play a song with PIEZO.
③	MUSIC	Play music with PIEZO.
③	TEMPO	Setup a sounds rhythm.

Command for external communication

②	RX	Receive a RS-232 signal through RX port.
②	TX	Transmit a RS-232 signal through TX port.
②	MINIIN	Receive a minibus signal through the mini communication port.
②	MINIOUT	Transmit a minibus signal through the mini communication port.
③	ERX	Receive a RS-232 signal through the RX port.
③	ETX	Transmit a RS-232 signal through the TX port.

Commands for analog signal processing

③	AD	Read analog signal from the AD port.
③	REMOCON	Read a key value from an infrared remote control.
③	SONAR	Read distance from the ultrasonic wave port
③	RCIN	Read input value from a RC remote controller.
③	GYRODIR	Set the direction of a gyro.
③	GYROSET	Assign a gyro to a servo.

③	GYROSENSE	Set the sensitivity of a gyro.

Processing command

ON...GOTO	Skip according to the value of a variable.

Other commands

RND	Create a random number.
REMARK	Make an entry in text

Intention commands

	'$DEVICE	Setup the controller to be operated by the program that is currently running.
③	'$LIMIT	Confine the travel range of servo.

Chapter 2

General grammar for roboBASIC

Because the grammar of roboBASIC is based on the general BASIC programming language, most of roboBASIC is similar to or the same as BASIC. In this chapter, the general grammar of roboBASIC will be explained.

Character set

The character set of roboBASIC is composed of English letters (A-Z, a-z), numbers (0-9) and special symbols. The symbols listed in the following table have a special meaning in roboBASIC.

Symbol	Description
+	Addition symbol
-	Subtraction symbol
*	Multiplication symbol
/	Division symbol
%	Remnant symbol
.	Bit designation symbol
&	Numeral symbol
??	Text symbol
??	Character string symbol
:	label symbol
=	Equal sign or substitution symbol
<	Inequality symbol
>	Inequality symbol
<<	Bit left shift symbol
>>	Bit right shift symbol

Formula and operator

Formulas can be composed of a value that is calculated from integrating invariables, variables, and numerals with each other using operators. An operator executes arithmetic or logical operations for a given value. In roboBASIC, operators can be classified like the table below.

Classification	Function
arithmetic operator	Perform arithmetical computation.
relational operator	Compare numerical values.
logical operator	Compare combined condition or execute bit manipulation.
bit operator	Manipulate bit or execute operation for bit.

Arithmetic operators

An arithmetic operator is a symbol that executes a computation. Like the general BASIC language, addition (+), subtraction (-), multiplication (*), division (/), and modulus (% or MOD) can be used in roboBASIC. But there are some different points between roboBASIC and general BASIC.

Firstly, there is no precedence in the operator

In roboBASIC, a parentheses ( ) cannot be used.

Example: A = 1, B = 2, C = 3

General BASIC

A + B * C = 1 + 2 * 3 = 1 + 6 = 7 (in BASIC, a parentheses would be used if the addition equation took precedence before the multiplication)

RoboBASIC:

```
A + B * C = 1 + 2 * 3 = 3 * 3 = 9
```

Secondly, complicated mathematical computations can cause unexpected errors.

In this case, the mathematical computation must be divided into 2 or 3 computations.

Example:

```
D = A * B + C                 (Accepted)
F = A * B / C * D + E         (Avoid complicated arithmetic computations)
```

Thirdly, roboBASIC only supports byte types or integer types, so a decimal point in the outcome will be ignored.

Modulus Calculations use the “%” or “MOD” symbol and output will be a modulo.

Relational operators

A relational operator is used to compare two values. The output is either “TRUE” or “FALSE”. This output is used for controlling the flow of a program in an IF sentence.

Operator	Relation	expression
=	Equal to	X = Y
<>	Not equal	X <> Y
<	Less than	X < Y
>	Greater than	X > Y
<=	Equal or less than	X <= Y
>=	Equal or greater than	X >= Y

When an arithmetic and logical operator are combined in one formula, the arithmetic operator will be executed prior to the logical operator.

Logic operators

A logic operator is used for comparing combined conditions. The result of the calculation returns either “TRUE” or “FALSE”. This output is used for controlling the flow of a program in an IF sentence.

Operator	Meaning
AND	And
OR	Disjunction
XOR	Exclusive disjunction

Each operator has an output like the table below. In the table, "T" means true, "F" means false.

Value of X, Y		Output		
X	Y	X AND Y	X OR Y	X XOR Y
T	T	T	T	F
T	F	F	T	T
F	T	F	T	T
F	F	F	F	F

Bit operators

A bit operator executes calculations for each variable that is used in the robot controller making bit control through the input/output port easier.

There are bit sum (OR), bit product (AND) and exclusive bit sum for calculation of

the whole bit. In roboBASIC, the calculation symbols, left (<<), right (>>) and ".", are used to move a bit to a specific point.
If the value of A is 33 (binary number 00100001) and the value of B is 15 (binary number 00001111), the following results will occur when using the mentioned operators.

operator	output
A AND B	1 (00000001)
A OR B	47 (00101111)
A XOR B	46 (00101110)
A << 1	66 (01000010)
A >> 1	16 (00010000)
A.0	1 (0th bit of A)

When several operators are used in the same command, operation will be executed in the following order.
① arithmetic operator/ Bit operator
② relational operator
③ logical operator

Figure, Variable/Constant and other grammatical explanations

Because roboBASIC is developed to control hardware, roboBASIC does not support variables or constants that are related to strings used in general BASIC.

Figure type

There are byte type figures and integer type figures. The range according to the type of figure being used is shown below.

Figure type	size	Range
BYTE	1 byte (8bit)	0-255
INTEGER	2 byte (16bit)	0-65535

RoboBASIC does not support negative numbers. So when a "+" or "–" symbol is added in front of a number, the operation will result in an error.
Declarations must be in a suitable number type.

Antilogarithm

Because roboBASIC is designed to control hardware, using a hexadecimal number expression or another type of expression is more convenient than using decimal number type expressions. In roboBASIC, binary numbers (Bin), octonary numbers (Oct), Decimal numbers (Dec), and Hexadecimal numbers (Hex) can be used.

antilogarithm	Declaration	Usable Figure	Example
binary number	&B	0, 1	&B111101
octonary number	&O	0, ... , 7	&O75
Decimal number	N/A	0, ... , 9	61
Hexadecimal number	&H	0, ... , 9, A, ... F	&H3D

Constant and variable

A constant does not change while executing the program. RoboBASIC can define the constant as a byte type number or an integer type number. Constant type is defined automatically according to the range of the number. Once a constant is defined, it cannot be redefined. Defining a constant does not have an effect on the size of the program. Program modification can be more convenient when the number being frequently used is defined as a constant.

Example

```
CONST OFF = 0
CONST motor_1 = 3
CONST motor_1 speed = 200
```

A variable is the name of a memory location in data that is used within the program. In the minirobot controller, the number of variables is limited, so the variable declaration must be designed to minimize the size of the variables in accordance with the object.

```
DIM motor_1_delay AS INTEGER
DIM sensor_left AS BYTE
```

When declaring a constant or variable, follow the rules shown below.

First: English or Korean must be used in the first letter. In Korean (Chinese) or English, figures and "_" can be used for the variable or constant name.

Second: The variable or constant name cannot exceed 64 characters in length.

Third: The variable or constant name cannot be declared twice with the same name and there is no distinction between capital and lower case letters.

Fourth: When declaring a constant point greater than 65535, which is the limitation of the integer range, an error may occur.

Bit pointing

In roboBASIC, variables can be handled as a bit unit. To handle variables as a bit unit, the pointing bit operator (".") is used. When using the pointing bit operator, bits 0~7 (byte variable) and bits 0~15 (integer variable) are possible. Only a figure or constant can be used with this operator.

Example

```
DIM A AS INTEGER
    CONST BIT_2 = 2

    A.1 = 1
    A.BIT_2 = 0
    A.3 = IN(1)             'Read value from #1 port and put this value in third bit of
integer variable A

   OUT 2, A.1               'Output value of 1st bit of integer variable A to #2 port
```

Explanation statement

Code explanations should be interspersed within the program for efficient management and creation. To insert an explanation statement, the symbol (') or the "REMARK" command is used. The placement of a statement within the program does not have an effect on the execution of the program.

Substitution statement (=)

A substitution statement is used to substitute a value with a variable. The symbol

"=" is used. The value is always to the left side of substitute symbol (=) and the variable, character string, computing formula or function is to right side of the substitute symbol (=).

Example

```
A = B                    'Substitute each variables
A.1 = 1                  'pointing bit substitution
A = ADIN(0)              'Function substitution.
A = 3 * 2 - 1            'Substitute numerical computing formula
A = C + B - A            'Substitute variable computing formula
A = "1"                  'Substitute ASCII code
```

Line label

A Line label is used for pointing to a location within the program. Characters and figures can be used for a line label. There are some rules when labeling.

First: A character label must not exceed 64 characters and the first letter must be in either English or Korean..

Second: The label symbol (:) must be attached after the character label.

Third: Numerals within the 0 65535 range can be used for label name. The label symbol is not required.

Fourth: The label name cannot be duplicated and there is no distinction between capital and lower case letters.

Usually, a label is used for flow control like the commands GOTO or GOSUB.

Example

```
DIM A AS INTEGER

START:
        A = IN(0)
        IF A = 0 THEN
                GOTO START
        ELSE
                GOSUB 10
        END
        GOTO START

10      OUT 1, 0
        DELAY 100
        OUT 1, 1
        RETURN
```

Chapter 3

Explanation of declaration Commands in roboBASIC

These commands are used for declaring variables or constants.

DIM ... AS Declare ...as

Declare variable

Sentence structure

① In the case of a single variable declaration :
- English sentence: DIM [variable name] AS [variable type]

② In the case of a multi-variable declaration:
- English sentence: DIM [variable name] AS [variable type], [variable type] AS [variable type]…

Explanation of command

A variable used in roboBASIC must be declared by the DIM command. The DIM command must use AS to declare the variable type. The variable name does not distinguish between capital and lower case letters. The variable name must not be duplicated.

A variable is used to process the value of sensor or the converted value of an analog signal. So by using the proper variable, program creation is more efficient. The number of usable variables is different between each robot controller.

The MR-C2000 series uses variables less than 30 bytes in size. The MR-C3000 series use variables less than 256 bytes in size. Byte type variables are 1 byte in size and integer type variables are 2 bytes in size so the declaration must be correct in order not to exceed the maximum number of variables.

Example of command

```
DIM I AS INTEGER        'Declare I as an integer type
DIM J AS BYTE           'Declare J as a byte type
```

CONST

Declare constant

Sentence structure

- English sentence: CONST [constant name] = figure

Explanation of the command

Specifying a constant name for a figure simplifies the programming process. Some of the advantages of using constants rather than figures or variables are:

① Once a constant is defined, it can be used throughout the whole program.
② Constants can not be changed by mistake.
③ Modification is easy.
④ A constant does not occupy a large amount of memory.

Example of command

```
CONST OFF = 0          'Declare OFF as 0 (constant)
CONST A = &HB1001      'Declare A as decimal number 9 (constant)
```

Chapter 4

Flow control

Command explanation

These commands are used to control or execute program flow.

IF ... THEN ...

Conditional statement

Sentence structure

① single condition:

```
- English sentence: IF [condition] THEN
                          [statement when condition is true]
```

② multiple conditions:

```
- English sentence: IF [condition1] THEN
                          [Statement when condition1 is true]
                    ELSEIF [condition 2] THEN
                          [Statement when condition2 is true]
                    ELSE
                          [Statement when condition 1 and condition2 are false]
                    ENDIF
```

Explanation of command

When an IF ... THEN condition is executed, the IF condition will be investigated. If the condition is TRUE, the THEN statement will be executed. When the condition is FALSE, each sequential ELSEIF condition will be investigated and executed otherwise the ELSE conditional statement will be executed. Here, the ELSEIF can be included or not as needed.

In roboBASIC, the (IF..THEN) sentence is essential in order to operate in accordance with an external input and save the external value as a variable. The conditional statement judges the value of variable and allows the robot to move according to the value.

Example of command

① Execution of the condition and statement is very simple. Both can be included in the same line.

```
IF A > 0 THEN B = 5
IF A < 5 THEN B = 0 ELSE B = 1
```

② The conditional formula of an IF sentence can use 2 kinds of conditions when using a relational operator.

```
IF A > 0 AND A < 5 THEN B = 3
IF A = 7 OR A = 9 THEN B = 1
```

③ Example of using a complicated IF sentence

```
IF A = 1 THEN
    B = 2
    C = 3
ELSEIF A = 3 AND A = 5 THEN
    B = 1
    C = 2
ELSEIF A = 8 THEN
    B = 6
    C = 0
ELSE
    B = 0
    C = 0
ENDIF
```

FOR ... NEXT

Loop a fixed number of times

Sentence structure

- English sentence: FOR *[Loop variable]* = *[Start]* TO *[End]*
  [Loop statement]
  NEXT *[Loop variable]*

Explanation of command

[*Loop variable*] counts the number of loops. [Start] is initial value of the loop variable and [end] is the last value of the loop variable. A figure, constant, or variable can be used for the [Start] and [End] steps.

In roboBASIC, the [End] step must be larger than the [start] step. RoboBASIC incrementally increases the loop variable. There are rules when using FOR...NEXT sentences.

① A FOR...NEXT sentence can be used within another FOR...NEXT sentence.

```
FOR I = 1 TO 10
    FOR J = 1 TO 5
        ..........
    NEXT J
NEXT I
```

② When using several FOR...NEXT sentences, the order of the NEXT [Loop variable] must not be changed.

Incorrect

```
FOR I = 1 TO 10
   FOR J = 1 TO 5
      ..........
   NEXT I
NEXT J
```

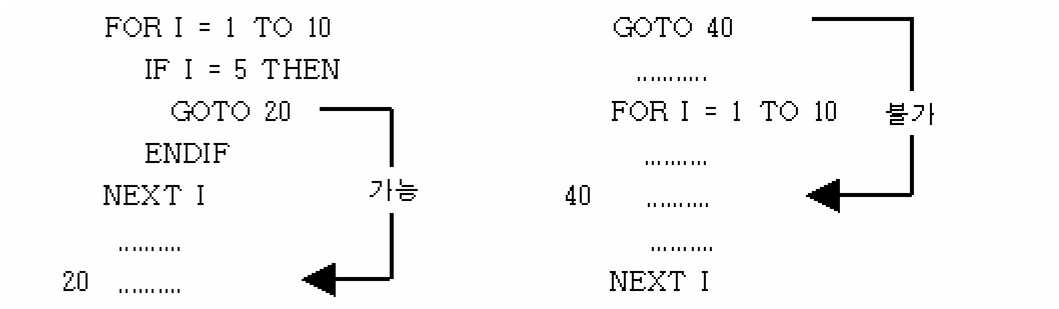
Correct

```
For I = 1 to 10
        For J = 1 to 5
            .........
        Next J
Next I
```

Even though the order of the loop variable has changed in the incorrect example, an error will not occur while creating objective code, but once uploaded to the minirobot controller unexpected results will occur.

③ It is possible to exit from inside a FOR···NEXT sentence but it is not possible to enter a FOR··· NEXT sentence from outside.

Correct　　　　　　　　　　　　Incorrect



```
   FOR I = 1 TO 10                   GOTO 40
     IF I = 5 THEN                     ..........
       GOTO 20                       FOR I = 1 TO 10
     ENDIF                             .........
   NEXT I                         40   .........
     .........                         .........
20   .........                       NEXT I
```

④ The value of a variable that is used in the [Loop variable], [Start] and [End] must not be changed arbitrarily during the execution of a FOR···NEXT sentence.

Example of command

Connect L.E.D. to port #0 of the controller and make it blink 5 times.

```
DIM A AS BYTE           ‘Declare variable to be used in repetition sentence

FOR A = 1 TO 5          ‘the number of repetition is 5 times
OUT 0, 0        ‘Turn on the L.E.D. connected in port #0 ON
        DELAY 100       ‘Delay for 100
        OUT 0, 1        ‘Turn off the L.E.D. connected in port #0 ON
        DELAY 100       ‘Delay for 100
NEXT A
```

GOTO

Move to a specific location.

Sentence structure

- English sentence: GOTO *[Line label]*

Explanation of command

The GOTO command changes the flow of the program by jumping to a specific line in the code. Using the GOTO command excessively will complicate the program so do not use it too frequently.

Example of command

```
        DIM I AS INTEGER
        DIM J AS BYTE

        I = 7
        IF I = 6 THEN GOTO L1
          ................
    L1:   J = 1
        OUT I, J
```

GOSUB ... RETURN

Call up a subroutine and return.

Sentence structure

```
- English sentence:        GOSUB [line label]
                        .............
           [line label]:    .............
                        ............
                    RETURN
```

Explanation of command

The GOSUB command calls up a frequently used Sub Routine and then returns. In this way GOSUB allows the program to smaller and more efficient.

It is possible to call a second sub routine from within the original sub routine. This is limited to 4 times with the *MR-C2000 series and 5 times with the MR-C3000 series.* More than this will cause errors.

Example of command

```
          DIM LED_PORT AS INTEGER

          LED_PORT = 1
START: .............
          .............
          GOSUB LED_TOGGLE
          ...............
          GOTO START
          END

LED_TOGGLE:
          TOGGLE LED_PORT
          RETURN
```

END

Finish the execution of the program.

Sentence structure

- English sentence : END

Explanation of command

After 2 seconds from the time of turning on the minirobot controller, a program saved in EEPROM will be executed. If the END command is not used at the end of a sub routine or execution sentence within the program, the program will run continuously. Always include the END command at the end of sub routines or execution sentences to prevent this.

Example of command

① Finish the execution of a program after executing.

```
        DIM A AS BYTE

START: A = IN(0)
        IF A = 1 THEN END
        ...........
        GOTO START
```

② A structural sub routine can be created.

```
          DIM A AS BYTE

          A = BYTEIN(0)
          IF A = 1 THEN
             GOSUB L1
          ELSEIF A = 3 THEN
             GOSUB L2
          ELSEIF A = 4 THEN
             GOSUB L3
          ELSE
             GOSUB L4
          ENDIF
          ...........
          END

L1: ..............
          RETURN

L2: .............
          RETURN

L3: ............
          RETURN

L4: ...........
          RETURN
```

STOP/RUN

Stop/Run program execution

Sentence structure

- English sentence: STOP/RUN

Explanation of command

This command will make the program either stop or run continuously. When the program is stopped, using the RUN command will initiate it again.

WAIT

Wait until program is finished.

Sentence structure

- English sentence: WAIT

Explanation of command

The control OS installed in the robot controller has the latest REAL-TIME program control.

When one program is executed, the next program will execute at the same time without stopping the previous program. If the next program is required to run only after the current program has finished, then the WAIT command is used.

Example of command

Ex 1: Output #7 and #8 ports after moving six motors.

```
MOVE 120, 100, 140, 90, 70, 150
WAIT
```

```
OUT 7, 1
OUT 8, 1
```

Ex 2: Output #8 after operating the six motors but output #7 at the same time.

```
MOVE 120, 100, 140, 90, 70, 150
OUT 7, 1
WAIT
OUT 8, 1
```

DELAY

Delay the execution of the program for a set time.

Sentence structure

- English sentence: DELAY *[Delay time]*

Explanation of command

This command will delay the program execution for a set time. Delay time for the MR-C2000 series controllers is 10ms and it is 1ms for the MR-C3000 series controllers.
A figure, constant, or variable can be used for [delay time]

Example of command

MR-C2000 series controller:

```
DELAY 10      'Delay for 100ms. (10ms * 10 = 100ms = 0.1sec)
```

MR-C3000 series controller:

```
DELAY 500     'Delay for 500ms. (1ms * 500 = 500ms = 0.5sec)
```

BREAK

Pause program execution and convert to debug mode.

2000

Sentence structure

- English sentence: BREAK

Explanation of command

Pause the program execution and switch to debug mode. When the program is paused, the memory condition of minirobot controller is sent to the PC. Make sure that controller and the PC are connected to each other. Otherwise, the program will stop and stay in that condition. More detailed information is included in “Explanation of the roboBASIC program”.

The BREAK command will not work with the MR-C3000 series controllers. If using a MR-C3000 series controller, the progress of the program can be followed systematically with the debug mode.

Explanation of command

```
        ...............
    BREAK                       'Pause program execution
          ...............
```

ACTION no

Perform prescribed templet basic motions according to the number (No.) value

Sentence structure

- English sentence: ACTION *[No.]*

Explanation of command

Perform a prescribed motion from the template according to the motion number.
A maximum of 32 motions are available.

Example of command

```
ACTION 3            'Perform motion No.3.
ACTION 5            'Perform motion No.5.
ACTION 23           'Perform motion No.23.
```

Note: This command is for the MR-C3024 controller and Robonova-I Robot only.

goto AUTO

Move to the template program.

Sentence structure

- English sentence : goto AUTO

Explanation of command

Command to begin the included template program.

Example of command

```
Goto AUTO                'Move to the template Program .
```

Note: This command is for the MR-C3024 controller and Robonova-I Robot only.

Chapter 5
Explanation of digital signal input and output in roboBASIC

In the MR-C2000, there are 12 digital I/O ports. And in the MR-C3000, there are 40 digital I/O ports. These ports execute several different functions. Refer to the "controller explanation" for more information on the digital I/O ports.

IN()

Read digital signal value from port.

Sentence structure

- English command: IN(*[Port number]*)

Explanation of command

A signal value that is input through a port is saved as a variable. The values inputted are 0 or 1. Byte or integer type variables can be used. At this time, only the last 0^{th} bit value is available. The most efficient way is to use the byte type variable.

Example of command

```
DIM A AS BYTE
A = IN(0)       'Read signal (switch or sensor) from #0 and set as the variable A.
```

OUT

Send digital signal to port.

Sentence structure

- English sentence: OUT *[port number]*, *[output value]*

Explanation of command

Send a signal from the controller through a port. When sending a 0 (LOW) value, a 0V signal will be output. When sending a 1 (HIGH) value, a +5V signal will be output. Numerals (0 or 1), constants, and variables can be used for the [output value]. Bit settings can be used for the [output value] because only a 0 or 1 is available in the output port.

Example of command

This example was created to test the input/output ports. A push button is connected to port #0 and an LED is connected to port #3.

```
        DIM A AS INTEGER

        A = 0                           'Initialize variable A
START: A = IN(0)                        'Read the condition of the button
                                        'A.0 = IN(0) is available.
        IF A = 1 THEN                   'When button is not pushed,
           OUT 3, 0                     'turn off LED
        ELSE                            'Otherwise,
           OUT 3, 1                     'Turn on LED
```

```
ENDIF
GOTO START                    'check button again
```

BYTEIN()

Read signal from byte unit input port

Sentence structure

- English sentence: BYTEIN(*[byte port number]*)

Explanation of command

The robot controller port can input/output as a unit of one bit like the IN/OUT command. In some cases, the signal must be input/output as one unit (here, byte port of unit of port #8)

MR-C2000 series controller:

Byte port	port
0	#0~#7 port (#0 port is low order of byte port)
1	#8~#11 port (#8 port is low order of byte port)

MR-C3000 series controller:

Byte port	port
0	#0~#7 port (#0 port is low order of byte port)
1	#8~#15 port (#8 port is low order of byte port)
2	#16~#23 port (#16 port is low order of byte port)
3	#24~#31 port (#24 port is low order of byte port)
4	#32~#39 port (#8 port is low order of byte port)

Using the BYTEIN command, a signal value that is input through the byte input port is saved as a variable. The variable type must be declared as either a byte or integer.

Example of command

A = BYTEIN(0) *'All signals from #0~ #7 ports are Input as variable A*

BYTEOUT

Output a signal to a port as a byte unit.

Sentence structure

- English sentence: BYTEOUT *[Byte port number]*, *[output value]*

Explanation of command

Output signal values through the byte unit port. Numerals, constants, or variables can be used for the [byte port number]. *For the [output value],* numerals between 0~255, constants or byte variables can be used.

Example of command

```
BYTEOUT 0, 255       'Send 1 value(5v) to #0~ #7 ports
BYTEOUT 0, &h10101010
                     ''Send 1 value(5v) to #1,#3,#5,#7 ports
                     ‘'Send 0 value(0v) to #0,#2,#4,#6 ports
```

INKEY

Input the key value through the input port. 2000

Sentence structure

- English sentence: INKEY (*[port number]*)

Explanation of command

When a switch is pushed once, it is actually pushed hundreds or thousand of times electrically and mechanically. This phenomenon is called chattering. Chattering can cause errors so an added protection circuit is needed. In the robot controller, a chattering protection function is built into the software. To operate this software, use the INKEY command.

Example of command

Save pushed condition of a switch connected to port # 0 as variable A.

```
DIM A AS BYTE

A = INKEY(0)
```

STATE()

Read present value of output port.

Sentence structure

- English sentence: STATE (*[port number]*)

Explanation of command

If the condition value of an output port is required after sending a signal through the port, use the STATE function. Do not use the IN function.

Example of command

This is a sample program to test the output condition of port #1.

```
DIM A AS BYTE

OUT 1, 1
A = STATE(1)          'A = 1
OUT 1, 0
A = STATE(1)          'A = 0
```

PULSE

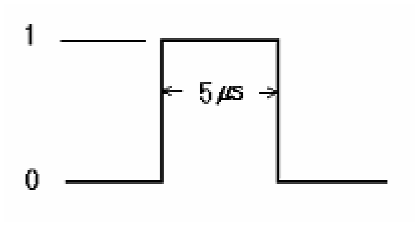
Send a pulse signal to an output port.

Sentence structure

- English sentence: PULSE *[port number]*

Explanation of command

Send pulse signal to an output port for 5 µs. Pulse signal is used to provide a signal to an external device.



Numerals, constants, or variables can be used for the [port number]

Example of command

```
PULSE 3                    'Send pulse signal to port #3.
```

TOGGLE

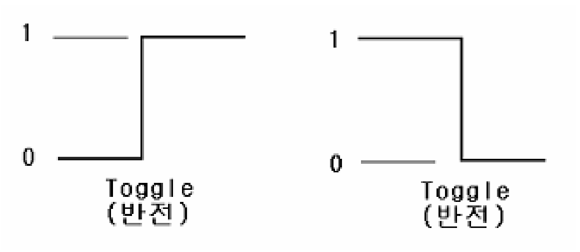
Reverse output port's signal.

Sentence structure

- English sentence: TOGGLE *[port number]*

Explanation of command

Reverse the output signal of an output port. If the signal of the port is 0 (low), the signal will be reversed as 1 (HIGH).



Numerals, constants, and variables can be used for the *[port number]*

Example of command

```
OUT 3, 1        'Send signal "1" to port #3.
TOGGLE 3        'Reverse the signal of port #3.
```

KEYIN()

Input several keys (analog key pad). 3000

Sentence structure

- English sentence: KEYIN(*[analog port number], [the number of keys]*)

Explanation of command

This command reads the values of 16 buttons through the AD converting ports (analog ports) in MR-C3000 series controller.

Numerals (0~7), constants, and byte variables can be used for the *[analog port number]*. Numerals (1~16), constants, and byte variables can also be used for the *[the number of key].*

The value range for a button is from 0 to 16. 0 means that key is not pushed. Numerals from 1 to 16 mean key is pushed.

Example of command

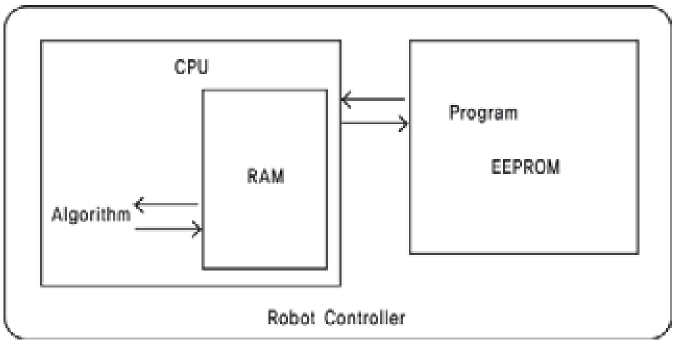
```
DIM K AS BYTE

                    K = KEYIN(0, 16)     'Input the values of 16 keys which are connected in AD port #0 as K
```

Chapter 6

Explanation of commands

Related to memory

The robot controller has a CPU and memory, so it can be described as a microcomputer. Memory execution is an important role in saving and calculating programs. External memory for the robot controller, in the form of the EEPROM, is used for saving user programs. The memory for executing calculations is located in the CPU.



Internal memory is called RAM but it is also called a register because of the special features of the robot controllers CPU. Internal memory is relative to the number of usable variables when a program is created. The MR-C2000 series controllers have 30bytes of variable space. The MR-C3000 series controllers have 256bytes of variable space. The other regions of memory are delegated to the internal use of the robot controller.

External memory is relative to the size of the created program. The MR-C2000 series has 4kbytes and the MR-C3000 series has 12k, 32k, 64k bytes of memory depending on the model.

PEEK()

Read contents of internal memory.

Sentence structure

- English sentence: PEEK (*[RAMregion]*)

Explanation of command

The PEEK function retrieves data from the internal memory. Do not use this function if the exact structure of the internal memory is not known.

MR-C2000series controller:
Numerals between 0~255, constants or byte variables can be used.

MR-C3000series controller:
Numerals between 0~65535, constants or byte variables can be used.

Example of command

```
DIM A AS BYTE

A = PEEK(43)          'Bring the value of RAM region address 43 to variable A.
```

POKE

Write data to internal memory.

Sentence structure

- English sentence: POKE *[RAMregion]*, *[data]*

Explanation of command

POKE command can be used to write data to internal memory. In the MR-C2000, numerals between 0~255, constants or byte variables can be used for the *[RAMregion]* .
In the MR-C3000, numerals between 0~65535, constants or byte variables can be used for the *[RAMregion].*

Numerals, constants, or variables (integer variables) can be used for *[data].*

Example of command

```
POKE &h40, 100                'Write 100 in RAM region address 40.
```

ROMPEEK()

Read data from external EEPROM.

Sentence structure

- English sentence: ROMPEEK (*[ROM region]*)

Explanation of command

The robot controller uses an EEPROM to save programs or other objects. The ROMPEEK or ROMPOKE functions that control external memory can be used to save data. If the region used already contains data, a fatal error may occur. Numerals, constants, or variables can be used for *[ROM region].*

ROMPOKE

Write data to external EEPROM.

Sentence structure

- English sentence: ROMPOKE *[ROM region], [data]*

Explanation of command

The robot controller uses an EEPROM to save programs or other objects. The ROMPEEK or ROMPOKE functions that control external memory can be used to save data. If a region used already contains data, a fatal error may occur. Numerals, constants, or variables can be used for *[ROM region].* Numerals between 0~255, constants or byte variables can be used for *[data].*

Chapter 7

Examination of the LCD module in roboBASIC

The LCD module designed for use with the robot controller is the MR-16202. Connect the LCD module to port #6 of the MR-C2000 series controller. The MR-C3000 has a specified LCD port.

The commands to control the LCD module and to display character strings will be explained here.

MR-16202 LCD module

LCDINIT

Initialize LCD module.

Sentence structure

- English sentence: LCDINIT

Explanation of command

LCD module must be initialized by using the LCDINIT command in order to prevent displaying unexpected characters. When the LCD module is initialized, all characters will be erased and the cursor will be sent to the top left hand corner.

Example of command

```
LCDINIT             'Initialize LCD module.
```

CLS

Erase characters in LCD module.

Sentence structure

- English sentence: CLS

Explanation of command

To erase all characters displayed in the LCD module, use the CLS command. When the CLS command is executed, all characters will be erased and the cursor will be sent to the top left hand corner. There are discrepancies between LCDINIT and CLS. With the CLS command, only characters will be erased, but with the LCDINIT command, all information, like internal variables, will be erased.

Example of command

```
CLS          'Erase what is displayed in the LCD module.
```

LOCATE

Point to the display position of a character in the LCD module.

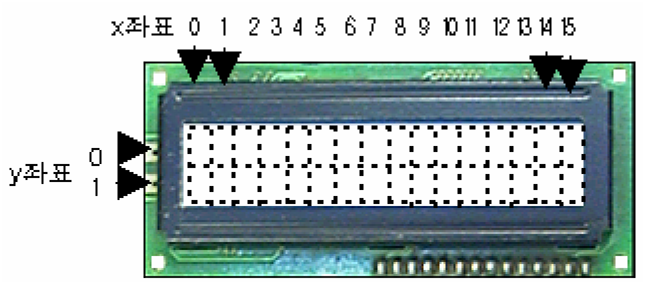
Sentence structure

- English sentence: LOCATE *[x coordinate], [y coordinate]*
- 한글구문 : 화면위치 *[x coordinate], [y coordinate]*

Explanation of command

Point to x and y coordinates within the LCD module with the LOCATE command. The coordinates of a 16x2 LCD module are set like the picture below. Numerals, constants, and variables can be used for the coordinates of x and y but it must start with 0

Example of command



LOCATE 0, 0 *'Send the cursor to the top left hand corner of the LCD module.*
LOCATE 4, 1 *''Send the cursor to the coordinates (4, 1) of the LCD module.*

PRINT

Output characters to the LCD module.

Sentence structure

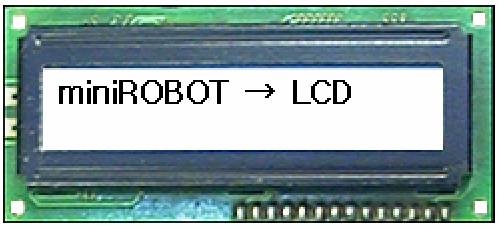
- English sentence: PRINT *"[character string]", [numeral]/ "[character string]]"*, ...

Explanation of command

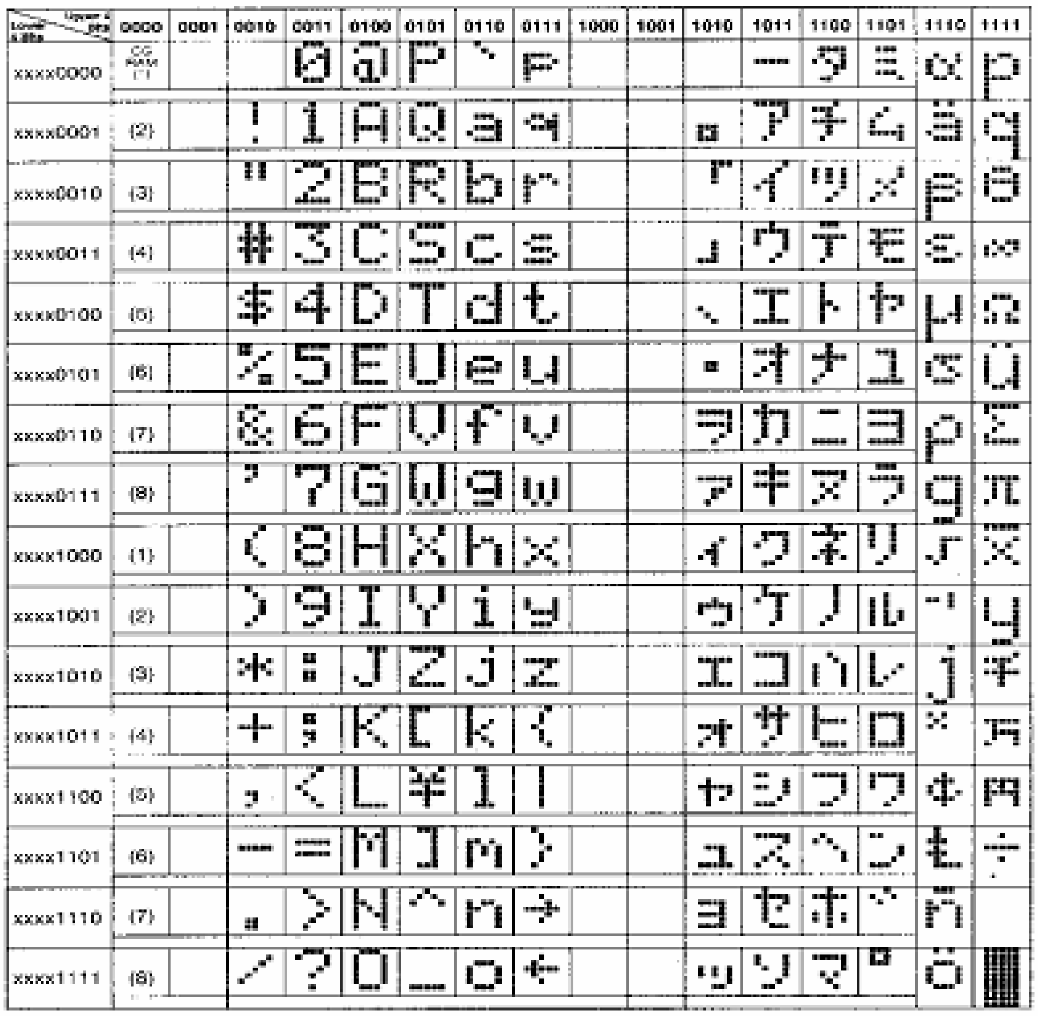
To output a character to the cursor's present location, use the PRINT command. The *[character string]* can be distinguished by a double quote (" "). The range of *[numeral] is* between 1 255 (0 cannot be used). In the LCD module, the character that is applicable to the numerals ASCII code will be displayed.

Example of command

```
CLS
PRINT "miniROBOT ", 126, "LCD"
```



The following are examples of ASCII code for a 16x2 line LCD module. Character code is dependent on the type of LCD module being used.



Upper 4 bits / Lower 4 bits	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
xxxx0000	CG RAM (1)			0	@	P	`	p				ー	タ	ミ	α	p
xxxx0001	(2)		!	1	A	Q	a	q			。	ア	チ	ム	ä	q
xxxx0010	(3)		"	2	B	R	b	r			「	イ	ツ	メ	β	θ
xxxx0011	(4)		#	3	C	S	c	s			」	ウ	テ	モ	ε	∞
xxxx0100	(5)		$	4	D	T	d	t			、	エ	ト	ヤ	μ	Ω
xxxx0101	(6)		%	5	E	U	e	u			・	オ	ナ	ユ	σ	ü
xxxx0110	(7)		&	6	F	V	f	v			ヲ	カ	ニ	ヨ	ρ	Σ
xxxx0111	(8)		'	7	G	W	g	w			ァ	キ	ヌ	ラ	g	π
xxxx1000	(1)		(	8	H	X	h	x			ィ	ク	ネ	リ	√	x̄
xxxx1001	(2)		)	9	I	Y	i	y			ゥ	ケ	ノ	ル	⁻¹	y
xxxx1010	(3)		*	:	J	Z	j	z			ェ	コ	ハ	レ	j	千
xxxx1011	(4)		+	;	K	[	k	{			ォ	サ	ヒ	ロ	ˣ	万
xxxx1100	(5)		,	<	L	¥	l	|			ャ	シ	フ	ワ	¢	円
xxxx1101	(6)		-	=	M	]	m	}			ュ	ス	ヘ	ン	£	÷
xxxx1110	(7)		.	>	N	^	n	→			ョ	セ	ホ	゛	ñ	
xxxx1111	(8)		/	?	O	_	o	←			ッ	ソ	マ	゜	ö	█

FORMAT()
DEC()
HEX()
BIN()

Specify the type of numeral to display in the LCD module.

Sentence structure

- Sentence of command: FORMAT (*[variable], [output type], [position of point]*)

Explanation of command

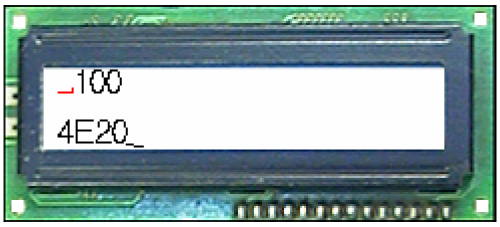
The LCD module follows a specific format when outputting a variable. The FORMAT command must be positioned after the PRINT command.

[variable type]/[output type]	Basic position of point	The number expression
Byte type /decimal	3	0 255
Byte type /hexadecimal	2	00 FF
Byte type /binary	8	00000000 11111111
Integer type /decimal	5	0 65535
Integer type /hexadecimal	4	0000 FFFF

Integer type /binary	16	0000000000000000	1111111111111111

Example of command

```
DIM A AS BYTE
DIM B AS INTEGER

LCDINIT
A = 100
B = 20000
LOCATE 0, 0
PRINT FORMAT(A, DEC, 4)
LOCATE 0, 1
PRINT FORMAT(B, HEX)
```



```
_100
4E20_
```

CSON()

CSOFF()

Show/hide the cursor in the LCD module.

Sentence structure

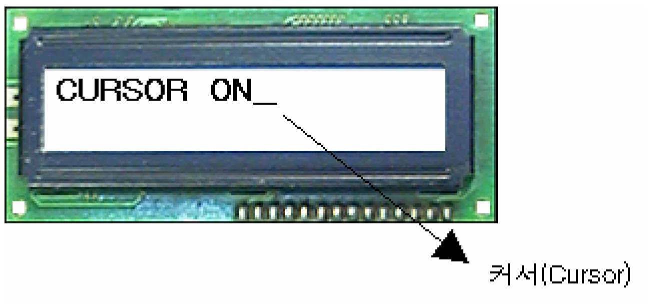
- English sentence: CSON / CSOFF

Explanation of command

The CSON/CSOFF command will show or hide the cursor in the LCD module. In general, the cursor is hidden when the LCD module is initialized.

Example of command

```
LCDINIT
CSON
PRINT "CURSOR ON"
```



CONT

Adjust the contrast of the LCD module.

Sentence structure

- English sentence: CONT *[contrast value]*

Explanation of command

The LCD module is backlit. Characters are displayed as a black color. With the CONT command, the thickness of the color can be adjusted. Numerals, constants, and variables can be used for the [contrast value]. As the [contrast value] increases, the character will thicken. The initial value is 7.

Example of command

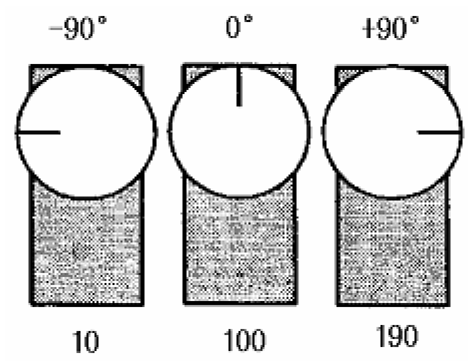
```
LCDINIT

CONT 10
PRINT "miniROBOT"
```

Chapter 8

Explanation of the motor Control commands in roboBASIC

The robot controller can control servos and DC motors. In the case of DC motors, the controller can control the speed, direction, and stop the motor using digital input and output commands.

The travel range of servos is from -90° to +90°. To operate servos in roboBASIC, degrees are expressed as numerals between 10 and 190, because the robot controller does not use negative number.



Various servos:

HS-311 HS-5645MG HSR-8498HB AIMOTOR

ZERO

Set the zero point of the servo.

Sentence structure

MR-C2000 series controller:

- English sentence: ZERO [motor 0 standard point], [motor 1 standard point], ..., [motor 5 standard point]

MR-C3000 series controller:

- English sentence : ZERO [group pointing], [motor n standard point], ...

Explanation of command

The zero point of the servos depends on each servo itself. This is due to product deviations. Some zero points can be 99 or 98, other zero points can be 101 or 102. These kinds of errors can be adjusted by the ZERO command. Once the zero point for each servo is set, it will be the standard point of the MOVE command.
The set zero point will be saved in the EEPROM to prevent it from being erased by power off.

Setting the zero point in MR-C2000 series:

1) Set the zero point for all servos as 100 to erase old data.
2) Set motor direction to the normal direction.

3) Turn off and turn on again.
4) Move motor to the zero (center) point using online function
5) Save this position as the zero point using the zero point command

EX1: eliminate old zero point value

```
ZERO 100,100,100,100,100,100
DIR    1,1,1,1,1,1                      'set motor direction to normal
```

EX2: Set zero point again. Save zero (center) point

```
DIR  1,1,1,1,1,1
ZERO 100, 101, 99                       'Set zero point of 3 servo motor
ZERO 102, 100, 100, 99, 101, 100        'Set zero point of 6 servo motor
```

In MR-C2000 series controller, the setting degree of zero point must be within 90~110.

Setting zero point in MR-C3000 series:

When setting zero points in the MR-C3000 controller, a group must be declared.

```
  ZERO G8B, 80, 120, 115, 80, 117, 88, 95, 120
                              'Set the zero point of Group8B(servos  #8~#15)
```

In the MR-C2000 series controller, setting the degree of zero point must be within 80~120.

MOTOR

Set the servo to be used.

Sentence structure

MR-C2000 series controller:

- English sentence: MOTOR *[motor number]*

MR-C3000 series controller:

- English sentence: MOTOR *[motor number] / [Specifying group]*

Explanation of command

Setting servo in the MR-C2000 series:

In MR-C2000 series controllers, there are six ports (#0~#5) for the servos. Motor numbers that can be specified are 0~5. When you want to use all servos, set the motor number to 6.

If a number is not set for the servo, the servo will not operate at all. In *[motor number],* only numerals between 0~6 can be used.

```
Ex1) MOTOR 6          'All motor (#0~#5) will be used
Ex2) MOTOR 2          '#2 motor will be used
```

Setting a servo in the MR-C3000 series:

In the MR-C3000 series controller, there are 32 ports for 32 servos. Each motor can be assigned with *[motor number]*. Groups of motors can be assigned with *[group pointing]*.

Numerals, constants, byte variables can be used for [motor number].

Example of command

```
Ex 1) MOTOR 0                '#0 servo will be used
Ex 2) MOTOR G6A                'servo group 6A (#0~#5) will be used.
        MOTOR G6C            'servo group 6C (#12~#17) will be used.
Ex 3) MOTOR G8A              'servo group 8A (#0~#7) will be used
Ex 4) Setting the servo with a variable
      DIM I AS BYTE
      FOR I = 0 TO 31        'servos (#0~#31) will be used by using variable I'
          MOTOR I
      NEXT I
Ex 5) MOTOR G24              ''servo group 24 (#0~#23) will be used.
Ex 6) MOTOR ALLON            ''All servos will be used.
```

MOTOROFF

Turn off a servo.

Sentence Structure

MR-C2000 series controller:

- English sentence: MOTOROFF *[motor number]*

MR-C3000 series controller:

- English sentence: MOTOROFF *[motor number] / [group pointing]*

Explanation of command

The MOTOROFF command is the same as the MOTOR command.

MOVE

Operate several servos at the same time.

Sentence structure

MR-C2000 series controller:

-English sentence: MOVE *[motor0angle], [motor1angle], ..., [motor5angle]*

MR-C3000 series controller:

-English sentence: MOVE *[group pointing], [motor n angle], ...*

Explanation of command

Move command with the MR-C2000 series controller:

The MOVE command operates a servo to the specific angle desired. When used, the PWM function is disabled. The range of the *[motor angle] is* between 10 ~190. When you want to set servos #1,#3, and #4, the sentence will look like this:

```
MOVE   60,      , 100, 120
```

When you want set only the #2 motor, it will look like this example.

```
MOVE   , 140
```

This process can be very difficult, especially when 6 motors must be operated at

the same time. This process would be easier if *"servo real time controlling"* were used.
If a DC motor is used, 100 means 'stop', 190 means maximum speed with counter rotation and 10 means maximum speed with normal rotation.

If the motor is to be used after the previous operation, use the WAIT command.

Example of command

```
MOVE 100, 50, 140, 120, 80, 40
MOVE 120, , , 160
MOVE , 70, 100
MOVE , , , , , 100
```

Move command in MR-C3000 series controller:

In the MR-C3000 series controller, ports for servos and for PWM are different. So the move command and the PWM command can be used at the same time.

Example of command

EX 1)
```
MOVE   G6A, 85, 113, 72, 117, 115, 100
MOVE   G6C, 75,      , 96, 123,       , 122
MOVE   G8A, 85, 113, 72, 117, 115, 100, 95, 45
```

EX 2)
```
MOVE   G24, 85, 113, 72, 117, 115, 100
```

Is the same as;
```
MOVE24   85, 113, 72, 117, 115, 100
```

SPEED

Set the speed of a servo

Sentence structure

- English sentence: SPEED *[motor speed]*

Explanation of command

The SPEED command sets the speed of a servo operated by the MOVE command. In the *MR-C2000 series controller, numerals or constants between 1~15 can be used for [motor speed]. In the MR-C3000 series controller, byte variables can be used .* The normal setting is 3. Too fast a speed is dangerous for the robot as well as user.

Example of command

Ex 1)

```
SPEED 7                    'Set motor speed as 7.
```

Ex 2)

```
DIM STEP_SPEED AS BYTE             'Declare STEP_SPEED (Variable).
STEP_SPEED = 15            'Set the STEP_SPEED (Variable) as 15
SPEED STEP_SPEED           'Set the SPEED as STEP_SPEED
```

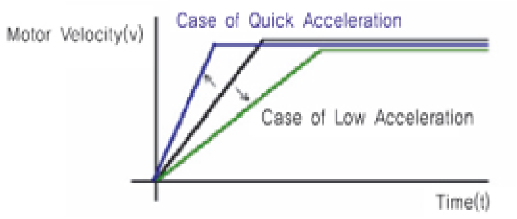
ACCEL

Set the acceleration of a servo

Sentence structure

- English sentence: ACCEL *[motor acceleration]*

Explanation of command



The ACCEL command sets the servo acceleration rate from 0 to the set speed.

[motor acceleration] uses numerals or constants between 0 and 15. The normal value is 3. A larger number increases servo acceleration.

When operating a servo for the first time, the servo rotates to the set point rapidly. To decrease this initial speed, it is better to use the ACCEL and SPEED command.

In the *MR-C3000 series controller, the ACCEL command cannot be used.*

Example of command

```
ACCEL 7      'Set the acceleration of servo as 7.
```

DIR

Set up the rotational direction of a servo.

Sentence structure

MR-C200 series controller:

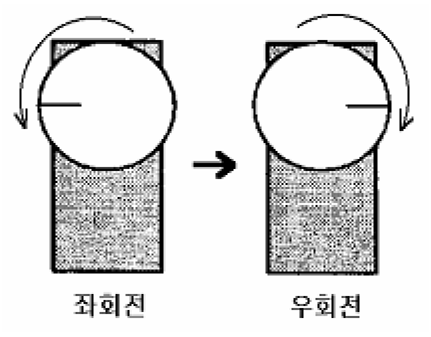
- English sentence: DIR [motor 0 direction], [motor 1 direction], ..., [motor 5 direction]

MR-C300 series controller:

- English sentence: DIR [pointing group], [motor n direction], ...

Explanation of command

A servo will turn left when the angle set is smaller than 100 (standard angle) and turn right when the angle set is larger than 100 (standard angle). In the picture bellow, servo rotation is to the left (normal direction) 10 degrees.



[motor direction] uses constants or numerals like 0 (counter rotation/left turn) or 1 (normal rotation/right turn). The default value is 0. For example, when 4 servos are used, if the # 3 servo is omitted, (like "DIR 0, 1, , 0"), it will turn in the standard direction.

Example of command

Example for the MR-C2000 series controller:

```
DIR 0, 1, 1, 0, 1, 0
DIR , , 0
```

Example for the MR-C3000 series controller

```
DIR G8A, 0, 1, 0, 0, 1, 0, 0, 0
DIR G8B, 1, 0, 1, 1, 0, 1, 1, 1
```

PTP Point to Point

Setup the On/OFF function for simultaneous control of servos

Sentence structure

MR-C2000 series controller:

- English sentence: PTP *[Set up Value]*

MR-C3000 series controller:

- English sentence: PTP *[SETON/SETOFF/ALLON/ALLOFF]*

Explanation of command

In the case of multiple movements and movements at different angles, the servos end times are different from each other. So, in the case of an Arm Robot or other servo-operated robot, the motions could be unstable.

In Robot Engineering, there is a theory called of Point-to-Point, which can calculate the timing endpoint of servos and ends all motions simultaneously allowing for smoother performance.

MR-C series controllers use this Point-to-Point control method by using the command PTP.

PTP control in the MR-C2000

The *[set up value]* uses 0 (cancel) or 1(set up) (numeral or constant) when two servos (No. 1 and No. 2) are used. See the examples below:

- Example of theatrical motion

```
PTP 0
MOVE 100, 100
MOVE 110, 120
```

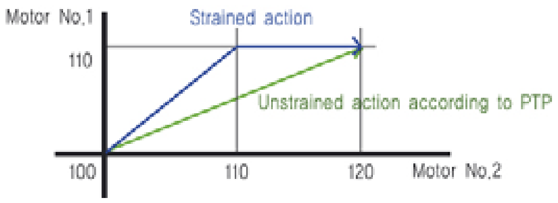
Description: No. 1 motor moves 10 degrees and No. 2 motor 20 degrees. Both traveling at the same speed will move 10 degrees together, and then No. 2 will move the remaining 10 degrees alone.

-Smooth motion by PTP function

```
PTP 1
MOVE 100, 100
MOVE 110, 120
```

Description: The No. 1 motor moves 10 degrees and the No. 2 motor 20 degrees but No. 1 moves at half the speed of No.2. Both servos move and stop at the same time.

See the graph bellow for a comparison between using the command "PTP" (green arrow) or not (blue arrow).



Notice that in normal motion, motor No.1 and No.2 move to the 110 angle simultaneously, but then only No.2 moves to 120 thereafter. However, using the command "PTP", by calculating between the expected movement angle 10 for No.1 and 20 for No.2, a smooth ending motion is accomplished.

PTP control in the MR-C3000 series controller)

Multiple servos can be used with the MR-C3000 series controllers. The function of the PTP can be adjusted to control all servos or individual groups.

- PTP SETON (PTP setup): Set up function of PTP as groups.
- PTP SETOFF (PTP cancel): Cancel function of PTP as groups.
- PTP ALLON (PTP all setup): Set up function of PTP for all servos.
- PTP ALLOFF (PTP all cancel): Cancel function of PTP for all servos.

In the MR-C3000 series controller, *using the command "WAIT"* at the end of a movement for each group, all servos in that group will end movement at the same time.

SERVO

One servo is operated

Sentence structure

- English sentence: SERVO *[motor No.],[motor Angle]*

Explanation of command

This command sets the desired motor angle. In case of the MR-C2000 series controller, the PWM function is canceled. The *[motor No.] is the motor port no. on the controller. The [motor Angle]* is between 10 and 190, using a numeral, constant or variable byte type is possible.

Example of command

Ex 1)

```
        SERVO 1,130                    ' operate No.1 motor at 130 position
```

Ex 2)

```
        DIM I AS BYTE

        FOR I = 10 TO 190
                SERVO 4, I
                DELAY 100
        NEXT I
```

PWM

Pulse Width Control

Sentence structure

- English sentence: PWM [motor No.],[Pulse width value]

Explanation of command

Control of PWM in the MR-C2000:

Servo controller ports and PWM ports are used in conjunction within the MR-C2000 series controllers. Hence, *[motor No.] is from 0 to 5.*

If the PWM Command is outputted with a pulse width value the servo function is canceled.

(Warning: In a program, the SERVO or MOVE command and the PWM command cannot be used together.)

Duty rate	Pulse width value	Duty rate	Pulse width value
0	0	60	153
10	25	70	178
20	51	80	204
30	77	90	229
40	102	100	254
50	127		

```
PWM 3, 127              ' PWM output of 50% duty rate at No.3 motor port.
```

PWM control in the MR-C3000 series controllers

The servo control and PWM do not use the same ports. There are 3 PWM ports (No.0~No.3) installed in the MR-C3000 series controllers (454Hz of PWM frequency in the MR-C3000 series).

```
PWM 0, 120              'Pulse output of 120 duty rate at PWM No.0 port.
```

FASTSERVO

Operate one servo at a faster speed 2000

Sentence structure

- English sentence: FASTSERVO *[motor No.],[motor Angle]*

Explanation of command

This command speeds a specific servo to the desired angle as fast as possible. Only the MR-C2000 series controllers use this command. The [motor No.] is the servo port and [motor angle] is the desired angle. Numerals and constants between 10 and 190 can be used for [motor angle].

Example of command

```
FASTSERVO 2,190         'Send No.2 motor to an angle of 190 as fast as possible.
```

HIGHSPEED

Set a servo to high speed mode

3000

Sentence structure

- English sentence: HIGHSPEED *[SETON/SETOFF]*

Explanation of command

This command sets or cancels the high-speed mode of a servo in the MR-C3000 series controllers. High speed is about 3 times faster than normal speed.

- HIGHSPEED SETON: Enable high-speed mode in the MR-C3000 series controllers.
- HIGHSPEED SETOFF: Cancel high-speed mode in the MR-C3000 series controllers. (Returns to the normal speed mode).

Example of command

```
HIGHSPEED SETON        'Set high speed mode.
```

MOVEPOS Move position
POS Motor position

Set movement position or motor position 3000

Sentence structure

```
- English sentence:   MOVEPOS [line label]
                          …………
               [line label]: POS [Group appoint], [motor n Angle], …
```

Explanation of command

When the position of the robot (consisting of the command 'move') is brought from other commands in the MR-C3000 series controller, it is handled with the command 'POS' (motor position) and the [line label] with the command 'MOVEPOS'. With the commands 'MOVEPOS' and 'POS', you can easily revise and write the roboBasic program.

Example of command

```
        …………………………
        MOVEPOS  POS01        ' Move command 'POS' part of 'POS01' label position
        …………………………
POS01:  POS G6A, 10, 32, 15, 120, 78, 93
POS02:  POS G6A, 67, 47, 32, 153, 23, 33
POS03:  POS G6A, 34, 37, 122, 162, 84, 28
```

FPWM Frequency pulse

Output PWM signal (Frequency can be variable) 3000

Sentence structure

- English sentence: FPWM *[port], [frequency], [duty rate]*

Explanation of command

The frequency of the PWM is changed and the pulse outputted in the MR-C3000 series controllers.
Port: From 0 to 2
Frequency: From 1(low frequency) ~5 (high frequency)
Duty rate: From 0 to 255

Example of command

FPWM 0, 1, 127	*'Output PWM signal (at 50% duty rate (127) and low frequency in the PWM No.0 port of MR-C3000 series controller)*

'

MOVE24

Move all 24 servo motors 3000

Sentence structure

- English sentence: MOVE24 *[Motor 0 Angle],………,[Motor 23 Angle]*

Explanation of command

You can operate groups of servos with the command “MOVE” in the MR-C3000 series controllers. The commands “MOVE24” and “MOVE G24” are used to operate 24 servos simultaneously. This command is good for operating a 16 to 24 servo robot.

Example of command

```
MOVE24   100, 45, 67, 44, 132, 122, , , , 76, 81, 90
```

INIT

Initial

Set initial position of the robot 3000

Sentence structure

- English sentence: INIT *[appoint Group], [motor n Angle],……*

Explanation of command

When the MR-C3000 series controller is installed in a robot, all of the servos are set to an initial position of "100" during the power on. As a result damage to the robot is possible.

To prevent damage, the initial power on position can be set to a position other than "100".

In the case of an analogue servo (HS-series), use the command "INIT".

In the case of a digital robot servo (HSR-series), use the command "GETMOTORSET".

Example of command

```
INIT G8A, 100, 45, 67, 44, 132, 122, 76, 81
```

MOTORIN() Motor input()

Read the present rate of the servo. 3000

Sentence structure

- English sentence: MOTORIN (*[motor No.]*)

Explanation of command

With this command, it is possible to read the present position value of any robot servo (HSR- series) connected to the MR-C3000 series controller and control them.

It is possible to use the numerals 0 to 31 as a constant for [motor No.].

Connected to controller: A motor angle of 10 to 190 can be read.

Not connected: 0 motor angle

Example of command

```
DIM   S0   AS   BYTE

MOTOR 0                  'Use No. 0 servo motor.
S0 = MOTORIN(0)          'Save value of No.0 servo motor to S0 variable.
```

AIMOTOR AI motor set up

Use AI motor 3000

Sentence structure

- English sentence: AIMOTOR *SETON/SETOFF/INIT/[motor No.] / [appoint group]*

Explanation of command

The AI motor is manufactured by Megarobotics. A Micro control chip is installed in the AI motor, which can communicate with the MR-C series controllers via RS232. The AI motor is controlled by the MR-C series controllers in the same way as a normal servo.

- Control angle of motor
- Present state of motor and control torque through PDI control (PGAIN, DGAIN)

AI motors can connect to ports No. 0 to No.30 (31 total ports).
[motor No.] : appoint each motor,
[appoint Group] : appoint motors as group

The command process is similar to the command "MOTOR". A numeral, constant, or a variable byte type is possible for the *[motor No.].*

When using an AI motor, declaring "use AI motor" is essential, but not required for other servos.

- AIMOTOR SETON: Set up using an AI motor.
- AIMOTOR SETOFF: Cancel using an AI motor.
- AIMOTOR INIT: Move AI motor to the initial position smoothly.

Example of command

```
AIMOTOR INIT        ‘Initialize AI motor
AIMOTOR SETON       ‘Declare use AI motor
AIMOTOR 0           'Use No. 0 AI motor
AIMOTOR G6B         'Use Group 6B motor(No.6-11)
```

AIMOTOROFF Cancel AI motor

Cancel AI motor 3000

Sentence structure

- English sentence: AIMOTOROFF *[motor No.] / [appoint Group]*

Explanation of command

Cancellation of the command "AIMOTOR".
The same command function as "MOTOROFF".

Example of command

```
AIMOTOROFF 0        'Cancel No.0 AIMOTOR.
AIMOTOROFF G6B      'Cancel all Group 6B motors(No.6-11).
AIMOTOR SETOFF      'Declare set off AI motor.
```

AIMOTORIN() AI motor input()

Read the present value of an AI motor 3000

Sentence structure

- English sentence: AIMOTORIN (*[motor No.]*)

Explanation of command

With this command it is possible to read the present position value and control an AI motor connected to the MR-C3000 series controllers.

It is possible to use numerals 0 to 30, as a constant for [motor No.].

Connected to the controller: A motor angle of 10 to 190 is read.

Not connected: 0 motor angle

Example of command

```
DIM  AI5  AS  BYTE

AIMOTOR INIT
AIMOTOR SETON
AIMOTOR 5                'Set using No.5 AI motor
AI5 = AIMOTORIN(5)       'Save value of No.5 AI motor to variable AI5
```

GETMOTORSET Set motor input

Read the present rate of a servo and maintain that state. 3000

Sentence structure

- English sentence: GETMOTORSET *[appoint Group], [appoint motor n input],…*

Explanation of command

With an interactive communication system, it is possible to read the present position value of a digital robot servo (HSR series) connected to the MR-C3000 series controller.

When the MR-C3000 series controller is installed in a robot, all of the servos are returned to the initial position of “100” during power on and as a result damage to the robot is possible.

To prevent damage, the initial power on position can be set to a position other than “100”.

Read the present position before initial power on and then start the appointed action smoothly according to the “move” command.

At [appoint motor n input], use “0” or “1”. In the case of “1”, read the present value of selected servo and maintain the present status of robot. In the case of “0”, move to initial value of the controller which is 100.

Example of command

```
GETMOTORSET G8A, 1, 1, 1, 1, 0, 0, 0, 0
‘No. 0번, 1, 2, 3 servo motors maintain the present value at power on.
‘No. 4, 5, 6, 7 servo motors move to initial value of 100 at power on.
```

Chapter 9

roboBASIC

Commands for Music Control

The MR-C series controllers have the ability to play a warning beep and music. To perform this function, a simply constructed piezo is needed separately. The pin out of the piezo consists of a (+) red terminal and white signal terminal.
If loud and clear sound is desired, a speaker can be connected, but a current driver circuit or AMP is required.

Connection to the MR-C2000 series controller:

The piezo is connected to the No.8 port of MR-C2000 series controller. The (+) terminal of the piezo is connected to the VCC of the No. 8 port and the (-) terminal connected to the SIG (signal) of the No.8 port.

Connection to the MR-C3000 series controller:

The piezo is connected to the No.28 port of the MR-C3000 series controller. The (+) terminal of the piezo is connected to the VCC of port No. 28 and the (-) terminal connected to the SIG (signal) of the No.28 port.

Note: The MR-C3024 controller has a built in piezo.

BEEP

Warning sound with piezo 2000

Sentence structure

- English sentence: BEEP

Explanation of command

To make a warning sound with the MR-C2000 series controllers, use the command "BEEP". Using a buzzer for a warning sound is only possible by connecting to the normal output ports using the command "OUT". *The MR-C3000 series controller uses the buzzer with the command "OUT".*

Example of command

```
BEEP                'Output warning sound
```

SOUND

Output sound with piezo 2000

Sentence structure

- English sentence: SOUND [pitch], [length], [pitch], [length], ...

Explanation of command

In the MR-C2000 series controllers, the signal frequency and delay time in the piezo port can be setup.
Set up value: From 1 to 254
Refer to the chart below:

Input	Frequency (Hz)	Input	Frequency (Hz)	Input	Frequency (Hz)
1	38.86k	70	800	160	389
2	23.81k	80	775	170	365
5	11.11k	90	689	180	344
10	5.88k	100	621	190	327
20	3.00k	110	565	200	311
30	2.00k	120	518	210	295
40	1.54k	130	478	220	283
50	1.23k	140	444	230	270
60	1.00k	150	413	240	260

Time	Length (11msec)
0.5 sec	45
1 sec	90
sec	180

Example of command

```
SOUND 60, 90, 60, 180, 30, 45
```

‘Generate a 1KHz frequency for 1sec, 2sec and generate 2KHZ frequency for 0.5sec.

PLAY

Play music with a piezo 2000

Sentence of command

- English sentence : PLAY "*[play music line]*"

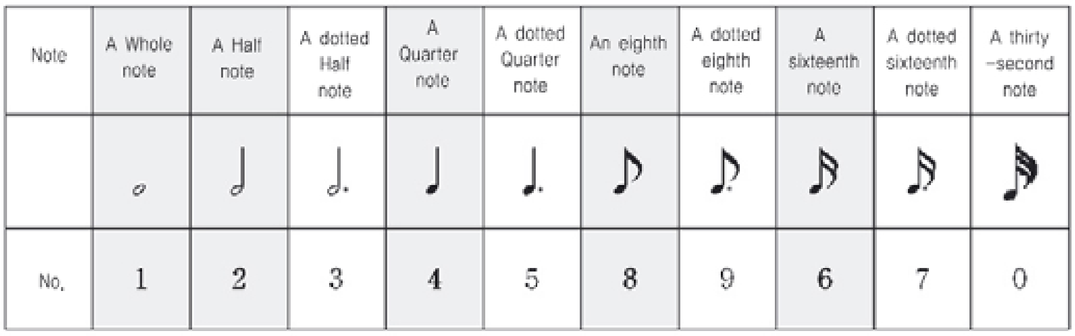
Explanation of command

The roboBASIC program and MR-C series controllers provide a function to play music.
To play the music, data must be added to the *[play music line].* Refer to the chart below:

play music line		Describe
Eng/Symbol	Korean	
C	도	"Do"
D	레	"Re"
E	미	"Mi"
F	파	"Fa"
G	솔	"Sol"
A	라	"La"
B	시	"Si"
T	템포, 박자, 속도	Control tempo, beat and speed
L	저	Choose low octave
M	중	Choose middle octave
H	고	Choose high octave

#, +	반음 올리기	Sharp a tone (#)
$, -	반음 내리기	Flat a tone (♭)
P, ,(a rest)	쉼표	A rest
<	한 옥타브 내린다	Drop an octave
>	한 옥타브 올린다	Raise an octave

T means tempo and the basic tempo value is 7.
You can adjust from 1 to 0 (0 means 10). 1 is the fastest tempo and 0 is the slowest.
3 octave stages are possible with the MR-C2000 series controller, they are named low, mid and high octave.
To adjust the length of a tone, use numerals 0 to 9 and refer to chart below:



Note	A Whole note	A Half note	A dotted Half note	A Quarter note	A dotted Quarter note	An eighth note	A dotted eighth note	A sixteenth note	A dotted sixteenth note	A thirty-second note
	𝅝	𝅗𝅥	𝅗𝅥.	♩	♩.	♪	♪.	𝅘𝅥𝅯	𝅘𝅥𝅯.	𝅘𝅥𝅰
No.	1	2	3	4	5	8	9	6	7	0

If the length of a tone is not chosen, the last length will be played.
"4CDEF8G" means playing Do, Re, Mi and Fa an a crotchet of Sol as the eighth note. The default value for the command "PLAY" is a middle octave. Crotchet, Do, and tempo 7 are the same octave and tempo value as the last adjustment.

① Tone length is attached [4Do 8Mi 6Fa].
② The # symbol or other marks are attached to the front of the tone [#Do $Mi +Fa -Sol 4#Do #4Do].

<u>Use the command "MUSIC" instead of "PLAY" when using the MR-C3000 series controllers</u>

Example of command

```
PLAY "M4GGAA GGE GGEED"
PLAY "M4GGAA GGE GEDEC"
```

MUSIC

Play music with a piezo. 3000

Sentence of command

- English sentence: MUSIC "*[play music line]*"

Explanation of command

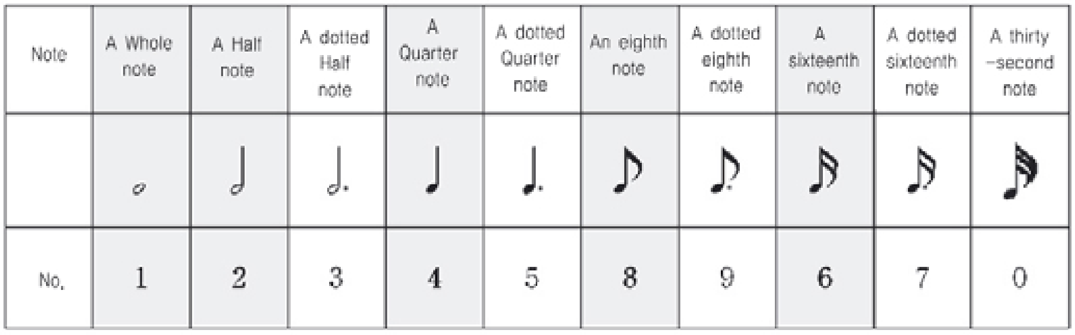
The roboBASIC program and MR-C series controllers provide a function to play music.
To play the music, data must be added to *[play music line].* Refer to the chart below:

play music line		Describe
Eng/Symbol	Korean	
C	도	"Do"
D	레	"Re"
E	미	"Mi"
F	파	"Fa"
G	솔	"Sol"
A	라	"La"
B	시	"Si"
[		Shorten the tone 1.5 times for a group of notes
]		Lengthen the tone 1.5 times for a group of notes

O	옥	Choose octave
M	중	Choose 3 octave
.		Lengthen the tone 1.5 times
#, +	반음 올린다.(#)	Sharp a tone (#)
$, -	반음 내린다.(♭)	Flat a tone (♭)
P, ,(rest)	쉼	A rest
<, L		Drop an octave
>, H		Raise an octave

7 octave stages are possible with the MR-C3000 series controller. To adjust the length of a tone, use numerals 0 to 9. Refer to the chart below:

In the case of dotted notes, write as a numeral or "." in the *[play music line]*.



Note	A Whole note	A Half note	A dotted Half note	A Quarter note	A dotted Quarter note	An eighth note	A dotted eighth note	A sixteenth note	A dotted sixteenth note	A thirty-second note
	𝅝	𝅗𝅥	𝅗𝅥.	♩	♩.	♪	♪.	𝅘𝅥𝅯	𝅘𝅥𝅯.	𝅘𝅥𝅰
No.	1	2	3	4	5	8	9	6	7	0

You have to use the command "PLAY" instead of "MUSIC" with the MR-C2000 series controllers.

The command "TEMPO" is available separately with the MR-C3000 series controllers.

Example of command

```
MUSIC "O34GGAA GGE GGEED"
MUSIC "O3GGA4.A GGE GEDEC"
```

TEMPO

Set the tempo of the music 3000

Sentence structure

- English sentence: TEMPO *[Set up Value]*

Explanation of command

Setup the music tempo using the command “MUSIC” when using the MR-C3000 series controllers.

Chapter 10

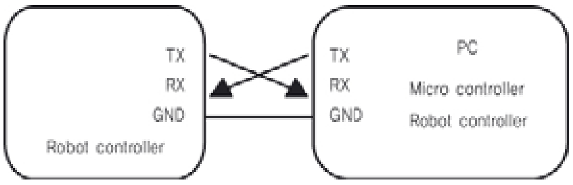
roboBASIC

Commands for outside communication

Outside communication of MR-C2000 series controller)

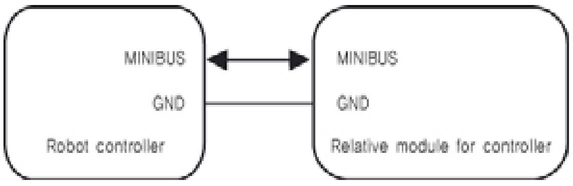
Two types of outside communication can be used with the MR-C2000 series controllers. One is the RS232 serial communication and the other is the miniBUS.

In serial communications, interactive communication with a personnel computer (RS232 compatible) or other MR-C controller is possible. In addition, it can communicate by either cable or wirelessly using an RF module. Three cables are needed for RS232 communication (connecting transmit (TX), receive (RX) and ground (GND)).



When connecting to a computer, a voltage converting circuit (MAX232) should be used.

With the miniBUS, the BUS signal and ground (GND) are used for interactive communication.



The miniBUS uses only one cable for interactive communication. Specific rules should be followed. The LCD module is one example of data transmission using the minibus. Here the controller built into the LCD is set to receive only.

Outside communication with the MR-C3000 series controller:

High-speed RS232 communications with outside equipment is possible with the MR-C3000 series controllers. However, miniBUS communication is not possible with the MR-C3000 series. Both controller types have a max RS232 connection speed of 115,200bps.

RX

RX Port receives a RS232 signal. 2000

Sentence structure

RX *[Port Speed], [Receipt Variables], [Receipt Error Process Label]*

Explanation of commend

By using the No. 9 Port of the MR-C2000 series controllers, data via RS232 can be received.
The [Port Speed] is represented by the numbers 1 to 4 where the numbers correspond to specific speed and port settings, which are explained below.

Number	Port Setting
1	1200bps, 8Bit data, No parity, 1 Stop bit
2	2400bps, 8Bit data, No parity, 1 Stop bit
3	2400bps, 8Bit data, No parity, 1 Stop bit
4	4800bps, 8Bit data, No parity, 1 Stop bit

[Receipt Variables] is the received data variable. Only declared byte form variables are allowed.
[Receipt Error Process Label] is the label for errors that occur during reception. As in the case of the communication buffer being empty. All programs awaiting data through RS232 Port can use the sentence structure below.

```
Retry:
          RX 4, A, Retry
```

To receive RS232 signals with the MR-C3000 series controllers, ERX (Data receipt) must be used.

Example of commend

In this example the ASCII code, &h80 (16 in analog), is received via a RS232 connection from an external terminal, the LED in the Zero Port is turned on, all others are turned off.

```
DIM   A   AS   BYTE

 Retry:  RX 4, A, Retry
         IF A = &h80 THEN
                   OUT 0, 0
         ELSE
                   OUT 0, 1
         ENDIF
         GOTO Retry
```

TX

TX Port transmits RS232 signal 2000

Sentence structure

TX *[Port Speed], [Data]*

Explanation of commend

By using the No. 10 Port of the MR-C2000 series controller, data via RS232 can be transmitted.
The [Port Speed] is represented by the numbers 1 to 4 where the numbers correspond to a specific speed and port settings, which are explained below.

Number	Port Setting
1	1200bps, 8Bit data, No parity, 1 Stop bit
2	2400bps, 8Bit data, No parity, 1 Stop bit
3	2400bps, 8Bit data, No parity, 1 Stop bit
4	4800bps, 8Bit data, No parity, 1 Stop bit

[Data] is a data value transmitted through the TX Port. Numerals, constants, and variables can be used. In the case of transmitting the letter “A”, the ASCII code related with the letter “A” should be sent. Refer to the example below.

```
DIM I AS BYTE

        I = "A"
        TX 4, I
```

If letter queues are put into variables, the ASCII value of the letter in the queue is put into variables.

To transmit a RS232 signal with the MR-C3000 series controllers, the command "ETX"(Data transmission) should be used instead of the command "TX" .

Example of commend

The following is an example where the key value of byte port "0" is continuously transmitting through the RS232 to an external terminal.

```
DIM   A   AS   BYTE

 Main:
        A = BYTEIN(0)
        TX   4, A
        GOTO Main
```

MINIIN

Receive Minibus signals through the minibus port 2000

Sentence structure

MINIIN

Explanation of commend

The miniBUS data is received using one of the six miniBUS Ports in the MR-C2000 series controller as long as “0” data is not received. The program structure can be seen below.

```
DIM A AS BYTE

Retry:
        A = MINIIN
        IF A = 0 THEN GOTO Retry
```

MINIOUT

MiniBUS Port transmits miniBUS signals 2000

Sentence structure

MINIOUT *[Data], [Data]…*

Explanation of commend

The miniBUS data is transmitted using the miniBUS Port No. 6 of the MR-C2000 series controllers. Minibus communication protocols are similar to those of the RS232. Numerals, constants, and variables are used for [Data]. An unlimited amount of data can be transmitted, but the *numeral "0" cannot be transmitted.*

Example of commend

```
MINIOUT 100, 20, 76, 65
```

ERX

ERX Port receives RS232 signals 3000

Sentence of commend

ERX *[Port Speed], [Receipt Variables], [Receipt Error Process Label]*

Explanation of commend

Data is received using ERX Port of the MR-C3000 series controllers. The *[Port Speed]* constants are listed below.

Number	Port Setting
2400	2400bps, 8Bit data, No parity, 1 Stop bit
4800	4800bps, 8Bit data, No parity, 1 Stop bit
9600	9600bps, 8Bit data, No parity, 1 Stop bit
14400	14400bps, 8Bit data, No parity, 1 Stop bit
19200	19200bps, 8Bit data, No parity, 1 Stop bit
28800	28800bps, 8Bit data, No parity, 1 Stop bit
38400	38400bps, 8Bit data, No parity, 1 Stop bit
57600	57600bps, 8Bit data, No parity, 1 Stop bit
76800	76800bps, 8Bit data, No parity, 1 Stop bit
115200	115200bps, 8Bit data, No parity, 1 Stop bit
230400	230400bps, 8Bit data, No parity, 1 Stop bit

[Receipt Variables] are the variables that store received data, only declared byte variables can be used.

[Receipt Error Process Label] is the label location for data that has not been received yet.

```
Retry:
        ERX  9600,  A,  Retry
```

To receive RS232 signals from the MR-C2000 series controller, the command “RX” should be used instead of the “ERX” command (Data receipt).

ETX

ETX Port transmits RS232 signals. 3000

Sentence structure

English: ETX *[Port Speed], [Data]*

Explanation of commend

Data is transmitted through the ETX Port of MR-C3000 series controller. The *[Port Speed]* constants are listed below.

Number	Port Setting
2400	2400bps, 8Bit data, No parity, 1 Stop bit
4800	4800bps, 8Bit data, No parity, 1 Stop bit
9600	9600bps, 8Bit data, No parity, 1 Stop bit
14400	14400bps, 8Bit data, No parity, 1 Stop bit
19200	19200bps, 8Bit data, No parity, 1 Stop bit
28800	28800bps, 8Bit data, No parity, 1 Stop bit
38400	38400bps, 8Bit data, No parity, 1 Stop bit
57600	57600bps, 8Bit data, No parity, 1 Stop bit
76800	76800bps, 8Bit data, No parity, 1 Stop bit
115200	115200bps, 8Bit data, No parity, 1 Stop bit
230400	230400bps, 8Bit data, No parity, 1 Stop bit

[Data] is the value to be transmitted through the ETX Port. Numerals, Variables and Constants can be used for *[Data]*. See the example below.

```
DIM I AS BYTE

I = "A"
ETX   9600, I
```

If the letter “A” is being transmitted, the ASCII code for the letter “A” should be sent. A letter being placed into a variable still must be sent in ASCII code first.
To transmit RS232 signals from the MR-C2000 series controllers, the command “TX” should be used instead of the command “ETX” (Data Transmission).

Chapter 11

roboBASIC

Analogue signal process

Command description

AD()

Analogue signal from the AD Port is converted into a Digital signal. 3000

Sentence structure

AD (*[AD Port]*)

Explanation of commend

There are eight AD Ports, numbered from zero to seven, in the MR-C3000 series controllers (Digital In-Out Ports 32 to 39), that convert an analogue signal from external sensors or devices into a digital signal. Constants and byte variables are used for *[AD Port].*

Example of commend

In the following example, a value is outputted to a LCD module after receiving an analogue signal from AD port No. 1.

DIM a AS BYTE	*Declare byte variable "a".*
LCDINIT	*Use of LCD module is initialized.*
CLS	*All data on LCD screen is cleared.*
CSOFF	*The cursor disappears.*
MAIN:	*A label named MAIN is declared.*
a = AD (1)	*The value inputted in AD Port #1 is saved as variable "a".*
LOCATE 5,0	*The cursor is located at 5.0 on the LCD*
PRINT FORMAT(a,DEC,2)	*The inputted value, a, is outputted to the LCD module as two digits using the decimal system.*
GOTO MAIN	*Go to MAIN.*

REMOCON()

Read infrared remote controller values from AD Transformation Port #7 3000

Sentence structure

REMOCON (*[Remocon(#)]*)

Explanation of commend

Port No. 7 is used for an infrared remote controller. *[REMOCON(#)]* is assigned the number 1, but other numbers can be used depending on which version of the MR-C3000 (3024) controller being used. In Regards to more specific details, refer to the examples in the following chapters.

Remocon(1) IR remocon (ID type)

Example of commend

DIM a AS BYTE *Declare the variable received values.*

```
MAIN:                        MAIN label to constantly receive the value of remocon.
a = REMOCON(0)               The value of remocon is put into the variable ”a”.
ON a GOTO MAIN,KEY1,KEY2,KEY3,KEY4  Go to MAIN unless a receiving value exists.
 GOTO MAIN                   Go to MAIN.
 END
 KEY1:                       Process it when a receiving value is one.
           ……………….
 GOTO MAIN                   Go to MAIN
 KEY2:                       Process it when a receiving value is two.
           ……………….
 GOTO MAIN                   Go to MAIN
 KEY3:                       Process it when a receiving value is three.
           ……………….
 GOTO MAIN                   Go to MAIN
 KEY4:                       Process it when a receiving value is four.
           ……………….
 GOTO MAIN                   Go to MAIN
```

SONAR()

Reads calculated distances from an ultrasound sensor connected to the Ultrasound Port 3000

Sentence structure

SONAR (*[Ultrasound Port]*)

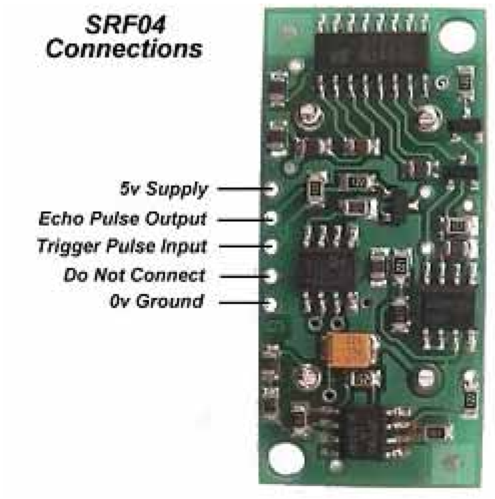
Explanation of commend

Digital In and Out Ports 0 to 23 of the MR-C3000 series controller can be used as Ultrasound ports 0 to 11. Refer to the following description.

Digital In and Out Port of MR-C3000 series controller	Ultrasound Port
Port #0	#0 Ultra-sonic Port output
Port #1	#0 Ultra-sonic Port input
Port #2	#1 Ultra-sonic Port output
Port #3	#1 Ultra-sonic Port input
Port #4	#2 Ultra-sonic Port output
Port #5	#2 Ultra-sonic Port input
Port #6	#3 Ultra-sonic Port output
Port #7	#3 Ultra-sonic Port input
Port #8	#4 Ultra-sonic Port output
Port #9	#4 Ultra-sonic Port input
Port #10	#5 Ultra-sonic Port output
Port #11	#5 Ultra-sonic Port input
Port #12	#6 Ultra-sonic Port output
Port #13	#6 Ultra-sonic Port input
Port #14	#7 Ultra-sonic Port output
Port #15	#7 Ultra-sonic Port input
Port #16	#8 Ultra-sonic Port output
Port #17	#8 Ultra-sonic Port input
Port #18	#9 Ultra-sonic Port output
Port #19	#9 Ultra-sonic Port input
Port #20	#10 Ultra-sonic Port output
Port #21	#10 Ultra-sonic Port input
Port #22	#11 Ultra-sonic Port output
Port #23	#11 Ultra-sonic Port input

The value of the [Ultrasound Port] must be a fixed number. The returning value from

SONAR must be within the range of 0 to 3000. If the value returned is 0, then the distance is not sensed, otherwise an “XX” value is returned.
An available ultrasound sensor for the MR-C3000 series controllers is the SRF04 model from ROBOT ELECTRONICS Inc.



Example of commend

DIM A AS INTEGER	*Declare fixed number variable A.*
A = SONAR(3)	*The received value (distance) is saved in variable A using Ultrasound Port #3 (Digital In and Out Port #6, 7).*

RCIN()

Input the pulse values from RC transmitters and receivers 3000

Sentence structure

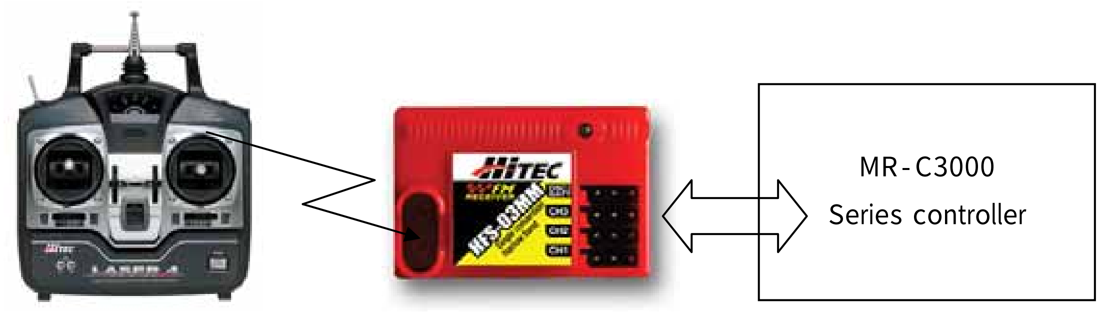
RCIN (*[RC Receipt Port]*)

Explanation of commend

RC receivers are attached to the AD Input Port of the MR-C3000 series controller. From there the receiving value of the transmitter can be read. The table below shows the configuration of *[RC Receipt Port]*.

AD Port Number of MR-C3000 series controller (Digital In and Out Port Number)	RC Receipt Port
Port #0 (Port #32)	RC receipt Port #0
Port #0 (Port #32)	RC receipt Port #1
Port #0 (Port #32)	RC receipt Port #2
Port #0 (Port #32)	RC receipt Port #3
Port #0 (Port #32)	RC receipt Port #4
Port #0 (Port #32)	RC receipt Port #5
Port #0 (Port #32)	RC receipt Port #6
Port #0 (Port #32)	RC receipt Port #7

To prevent malfunctions caused by electronic interference, FM receivers should be used rather than AM. If the transmitter has high-end functions, more varied actions or movements can be realized.



Example of commend

```
DIM A AS BYTE

A = RCIN(0)
```

A *Received signal from the RC Receipt Port is saved as variable A.*

GYRODIR

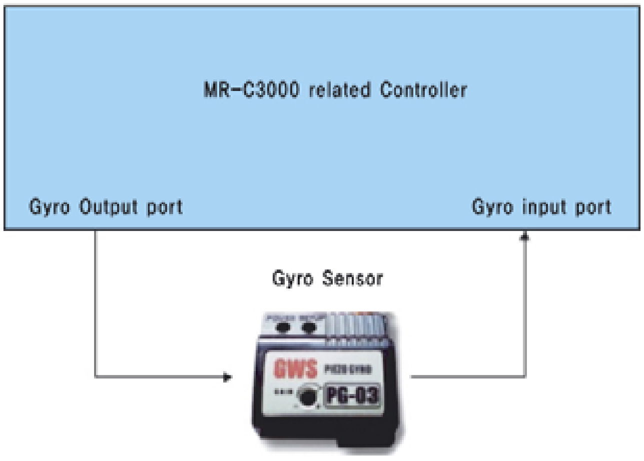
Sets direction of servos when supported by a Gyro. 3000

Sentence structure

GYRODIR *[Group], [Motor Direction]* …

Explanation of commend

This process controls the direction of a servo group when a gyro is connected to an AD port of the MR-C3000 series controllers. The number of gyros that can be used is four. Refer to the following chart.



AD Port Number (Digital In and Out Port Number) of MR-C3000 series controller.	Gyro Port
Port #0 (Port #32)	Gyro #1 channel output Port
Port #1 (Port #33)	Gyro #2 channel output Port
Port #2 (Port #34)	Gyro #3 channel output Port
Port #3 (Port #35)	Gyro #4 channel output Port
Port #4 (Port #36)	Gyro #1 channel input Port
Port #5 (Port #37)	Gyro #2 channel input Port
Port #6 (Port #38)	Gyro #3 channel input Port
Port #7 (Port #39)	Gyro #4 channel input Port

Since a gyro is reversible, the specific direction should be determined from the present value of the servo. [Motor Direction] is either a "0" or "1". A value of 1 will increase the servo position and a 0 will decrease it.

Example of commend

```
GYRODIR G6A, 1, 1, 0, 0, 1, 0
```

GYROSET

Determines which gyro will control a particular servo group. 3000

Sentence structure

GYROSET *[Group], [Motor N Gyro] …*

Explanation of commend

Gyroset determines which servo in a servo group [Group] is controlled by a specific gyro. [Motor N Gyro] is the specific gyro port being used for each servo in the group. Refer to the example below.

01,02,03,04 : GWS PG03	11,12,13,14 : KRG-1
21,22,23,24 : reserved	31,32,33,34 : reserved

Example of commend

```
GYROSET G6B, 1, 1, 2, 2, 0, 0
```

#6 servo receives Gyro sensor #1 and processes it.
#7 servo receives Gyro sensor #1 and processes it.
#8 servo receives Gyro sensor #2 and processes it.
#9 servo receives Gyro sensor #2 and processes it.
#10 and #11 servos do not use Gyro sensor.

GYROSENSE

Sets servo sensitivity to a Gyro. 3000

Sentence of commend

GYROSENSE *[Group], [Motor N Gyro Sensitivity]* …

Explanation of commend

Four gyros can be connected to the MR-C3000 series controller. GYROSENSE sets a single servos sensitivity to a gyro.

[Motor N Gyro Sensitivity] uses numerals from 0 to 255 or constants to control sensitivity of each servo in a group. A setting of “0” will not change a servos sensitivity. As the value increases so does the servos response to the gyro.

Example of commend

GYROSENSE G6A, 100, 100, 255, 255, 50, 50

#0 and #1 servo set to 100 of Gyro sensitivity.

#2 and #3 servo set to maximum (255) of Gyro sensitivity.

#4 and #5 servo set to 50 of Gyro sensitivity.

Chapter 12

roboBasic

Process Commands and Others

ON…GOTO

Conditional divergence according to the value of variables.

Sentence structure

ON *[Variable]* GOTO *[Line Label], [Line Label]…*

Example of commend

"ON… GOTO" is the command used when the program to process diverges according to the value of a *[Variable].* When using the IF command, the On…GOTO command can still be used. In fact, when comparing the two commands, ON…GOTO assists in the creation of smaller code.
The following is a comparison of IF to ON… GOTO.

```
A = BYTEIN(0)

IF A = 0 THEN
   GOTO 10
ELSEIF A = 1 THEN
   GOTO 20
ELSEIF A = 2 THEN
   GOTO 30
ELSE
   GOTO 40
ENDIF
```

```
A = BYTEIN(0)

ON A GOTO 10, 20, 30
IF A>2 THEN GOTO 40
```

Following the GOTO command is [Line Label]. [Line Label] increases numerically when the [variable] is true. Numerals and constants can be used for [variable]. A maximum number of 255 can be used for [Line Label].

RND

Random.

Sentence Structure

```
RND
```

Explanation of commend

To create a random program within the MR-C controllers, use the RND command. This command will create a random number between 0 and 255.

Example of commend

```
DIM A AS BYTE

A = RND
BYTEOUT 0, A                Random value is outputted at Byte Port 0.
```

REMARK

Place a statement within the code

Sentence structure

REMARK *[Description]*

Explanation of commend

It is good programming practice to enter statements within the code to explain certain procedures. This is done with either a (') or the REMARK command. Following the command, a description is entered on the same line. REMARK has no effect on the operation of a program.

Example of commend

```
REMARK 8 LEDs are turned on.
BYTEOUT 0, 0
```

Chapter 13
roboBasic
Command Description

‘$DEVICE

Sets the controller applicable to the program.

Sentence structure

‘$DEVICE *[Controller]*

Explanation of commend

This command sets the controller that the program is uploaded to after compiling. If a controller has already been assigned, it will be changed to the controller specified by [controller].

Example of commend

'$DEVICE MRC2000 *Present program to use the MR-C2000 controller.*

‘$LIMIT

Limits the range of movement for each servo. 3000

Sentence structure

'$LIMIT *[Motor Number], [Minimum Value], [Maximum Value]*

Explanation of commend

The command “$LIMIT” is used to set a servos total rotational angle in order to prevent damage by over driving a servo.
[Motor Number] is the specific servo being used. The range is from 0 to 31. *[Minimum Value]* and *[Maximum Value]* is the desired minimum and maximum angle between 10 and 190. The default servo angle is 10 to 190.

Example of commend

'$LIMIT 0, 50, 100 *The servo angle is limited to between 50 and 100.*