Chapter 12
Online commands

1 Online Command List

Online commands can be used to operate the controller via an RS-232C interface or via an Ethernet. This Chapter explains the online commands which can be used. For details regarding the RS-232C and Ethernet connection methods, refer to the "RCX340 Controller User's Manual".

About termination codes

During data transmission, the controller adds the following codes to the end of a line of transmission data.

- RS-232C
  - CR (0Dh) and LF (0Ah) are added to the end of the line when the "Termination code" parameter of communication parameters is set to "CRLF".
  - CR (0Dh) is added to the end of the line when the "Termination code" parameter of communication parameters is set to "CR".
- Ethernet
  - CR (0Dh) and LF (0Ah) are added to the end of the line.

When data is received, then the data up to CR (0Dh) is treated as one line regardless of the "Termination code" parameter setting, so LF (0Ah) is ignored.
The termination code is expressed as [cr/lf] in the detailed description of each online command stated in "2 Operation and setting commands" onwards in this Chapter.

1.1 Online command list: Operation-specific

Key operation

Operation type		Command	Option	Condition
Register program in the task		LOAD	*<program name>* | ,Tn , p PGm (m: 1-100, n: 1-16, p: 1-64)	2
Program	Reset program Execute program Stop program	RESET RUN STOP	Tn *<program name>* PGm (m: 1-100, n: 1-16)	2
Program	Execute one line Skip one line Execute to next line	STEP SKIP NEXT	Tn *<program name>* PGm (m: 1-100, n: 1-16)	2
Program	Execute before specified line Skip before specified line	RUNTO SKIPTO	Tn *<program name>* PGm (m: 1-100, n: 1-16, k: 1-9999)	2
Set break point		BREAK	*<program name>* | (n, n, n,...), k PGm | 0 0 (m: 1-100, n: 1-9999, k: 0/1)	2
Change manual movement speed		MSPEED	[*robot number*] k (robot number: 1-4, k: 1-100)	2
Move to absolute reset position		ABSADJ	[*robot number*] k, f (robot number: 1-4, k: 1-6, f: 0/1)	3
Absolute reset		MRKSET	[*robot number*] k (robot number: 1-4, k: 1-6)	3
Return-to-origin		ORGRTN	[*robot number*] k (robot number: 1-4, k: 1-6)	3
Change inching movement amount		IDIST	[*robot number*] k (robot number: 1-4, k: 1-10000)	2
Manual movement (inching)		INCH INCHXY INCHT	[*robot number*] km (robot number: 1-4, k: 1-6, m: +/-)	3
Manual movement (jog)		JOG JOGXY JOGT	[*robot number*] km (robot number: 1-4, k: 1-6, m: +/-)	3
Point data teaching		TEACH TCHXY	[*robot number*] m (robot number: 1-4, m: 0-29999)	2

Conditions: 1. Always executable.
2. Not executable during inputs from the programming box.
3. Not executable during inputs from the programming box, and while the program is running.
4. Not executable during inputs from the programming box, while the program is running, and when specific restrictions apply.

Utility

Operation type	Command	Option	Condition
Copy program	COPY	<*program name1*> TO <*program name2*> PGm (m: 1-100)	2
Copy points "m - n" to point "k"		Pm-Pn TO Pk (m: 0-29999, n: 0-29999, k: 0-29999)	
Copy point comments "m - n" to point comment "k"		PCm-PCn TO PCk (m: 0-29999, n: 0-29999, k: 0-29999)	
Delete program	ERA	<*program name*> PGm (m: 1-100)	2
Delete points "m - n"		Pm-Pn (m: 0-29999, n: 0-29999)	
Delete point comments "m - n"		PCm-PCn (m: 0-29999, n: 0-29999)	
Delete point names "m - n"		PNm-PNn (m: 0-29999, n: 0-29999)	
Delete pallet "m"		PLm (m: 0-39)	
Rename "program 1" to "program 2"	REN	<*program 1*> TO <*program 2*>	2
Check program syntax	SYNCHK	<*program name*> PGm , k (m: 1-100, k: 1-100)	2
Compile sequence program	SEQCMPL		2
Change program attribute	ATTR	<*program name*> PGm TO s (m: 1-100, s: RW/RO/H)	2
Setting main program	MAINPG	m (m: 1-100)	2
Initialize data Program	INIT	PGM	3
Initialize data Point		PNT	
Initialize data Point comment		PCM	
Initialize data Point name		PNM	
Initialize data Shift		SFT	
Initialize data Hand		HND	
Initialize data Pallet		PLT	
Initialize data General Ethernet Port		GEP	
Initialize data Input/output name		ION	
Initialize data Area check output		ACO	
Initialize data All data except parameters		MEM	
Initialize data Parameter		PRM	
Initialize data All data (MEM+PRM)		ALL	
Initialize data Communication parameter	INIT	CMU ETH	3
Initialize data Alarm history	INIT	LOG	3
Setting Input data	INPUT	SET d CAN CLR (d: input data)	2
Buffer clear Output message	MSGCLR		2
Change access level	ACCESS	k , ppppppp (k: 0/1, p: alphanumeric characters of 8 characters or less)	2
Setting password	SETPW		2
Setting Sequence execution flag	SEQUENCE	k (k: 0/1/3)	2
Reset alarm	ALMRST		2
Check or set date	DATE	yy/mm/dd (yy: 00-99, mm: 01-12, dd: 00-31)	2
Check or set time	TIME	hh: mm: ss (hh: 00-23, mm: 00-59. ss: 00-59)	2

Conditions: 1. Always executable.

2. Not executable during inputs from the programming box.
3. Not executable during inputs from the programming box, and while the program is running.
4. Not executable during inputs from the programming box, while the program is running, and when specific restrictions apply.

Data handling

Operation type		Command	Option	Condition
Acquiring status	Access level	?	ACCESS k [, ppppppppp] (k: 0/1, p: alphanumeric characters of 8 characters or less)	1
	Alarm status		ALM	
	Break point status		BREAK [*<program name>* | PGm] (m: 1-100)	
	Last (Current) point number reference		CURPNT	
	Emergency stop status		EMG	
	Selected hand status		HAND [*robot number*] (robot number: 1-4)	
	Inching movement amount status		IDIST [*robot number*] (robot number: 1-4)	
	Input data		INPUT	
	Online/offline status		LINEMODE [ETH | CMU]	
	Main program number		MAINPG	
	Remaining memory capacity		MEM	
	Mode status		MODE	
	Motor power status		MOTOR	
	Output message		MSG	
	Manual movement speed		MSPEED [*robot number*] (robot number: 1-4)	
	Return-to-origin status		ORIGIN [*robot number*] (robot number: 1-4)	
	Sequence program execution status		SEQUENCE	
	Servo status		SERVO [*robot number*] (robot number: 1-4)	
	Selected shift status		SHIFT [*robot number*] (robot number: 1-4)	
	Acquire task in RUN or SUSPEND status		TASKS	
	Task end condition		TSKECD Tk (k: 1-16)	
	Task operation status		TSKMON Tk (k: 1-16)	
	Version information		VER	
	Numerical data		*numerical expression*	
	Character string data		*character string expression*	
	Point data		*point expression*	
	Shift data		*shift expression*	
Read-out data		READ	*read-out file*	2
Write data		WRITE	*write file*	2

Conditions: 1. Always executable.
2. Not executable during inputs from the programming box.
3. Not executable during inputs from the programming box, and while the program is running.
4. Not executable during inputs from the programming box, while the program is running, and when specific restrictions apply.

Robot language independent execution

The Robot languages executable independently are the commands/functions with "✓" at "Online" column in Chapter 8 "robot language table".

Control code

Operation type	Command	Option	Condition
Execution language interruption	^C(=03H)		1

Conditions: 1. Always executable.
2. Not executable during inputs from the programming box.
3. Not executable during inputs from the programming box, and while the program is running.
4. Not executable during inputs from the programming box, while the program is running, and when specific restrictions apply.

1.2 Online command list: In alphabetic order

Command	Option	Meaning	Condition
?	ACCESS k , ppppppp (k: 0/1, p: alphanumeric characters of 8 characters or less)	Acquire access level	1
	ALM	Acquire alarm status	
	BREAK | *<program name>* | PGm (m: 1-100)	Acquire break point status	
	CURPNT	Acquire Last (Current) point number reference	
	EMG	Acquire emergency stop status	
	HAND [*robot number*] (robot number: 1-4)	Acquire selected hand status	
	IDIST [*robot number*] (robot number: 1-4)	Acquire inching movement amount status	
	INPUT	Acquire input data status	
	LINEMODE |ETH | CMU |	Acquire online/offline status	
	MAINPG	Acquire main program number	
	MEM	Acquire remaining memory capacity	
	MODE	Acquire mode status	
	MOTOR	Acquire motor power status	
	MSG	Acquire output message	
	MSPEED [*robot number*] (robot number: 1-4)	Acquire manual movement speed	
	ORIGIN [*robot number*] (robot number: 1-4)	Acquire return-to-origin status	
	SEQUENCE	Acquire sequence program execution status	
	SERVO [*robot number*] (robot number: 1-4)	Acquire servo status	
	SHIFT [*robot number*] (robot number: 1-4)	Acquire selected shift status	
	TASKS	Acquire task in RUN or SUSPEND status	
	TSKECD Tk (k: 1-16)	Acquire task end condition	
	TSKMON Tk (k: 1-16)	Acquire task operation status	
	VER	Acquire version	
	numerical expression	Acquire numerical data	
	character string expression	Acquire character string data	
	point expression	Acquire point data	
	shift expression	Acquire shift data	
^C (=03H)		Execution language interruption	1
ABSADJ	[*robot number*] k, f (robot number: 1-4, k: 1-6, f: 0/1)	Move to absolute reset position	3
ACCESS	k , ppppppp (k: 0/1, p: alphanumeric characters of 8 characters or less)	Change access level	2
ARMRST		Reset alarm	
ATTR	| *<program name>* | PGm TO s (m: 1-100, s: RW/RO/H)	Change program attribute	2
BREAK	|*<program name>* | PGm | 0 | (n, n, n,...), k | 0 | (m: 1-100, n: 1-9999, k: 0/1)	Set break point	2

Command	Option	Meaning	Condition
COPY	<*program name1*> | TO <*program name2*> PGm (m: 1-100)	Copy program	2
	Pm-Pn TO Pk (m: 0-29999, n: 0-29999, k: 0-29999)	Copy points "m - n" to point "k"	
	PCm-PCn TO PCk (m: 0-29999, n: 0-29999, k: 0-29999)	Copy point comments "m - n" to point comment "k"	
DATE	yy/mm/dd (yy: 00-99, mm: 01-12, dd: 00-31)	Check or set the date	2
ERA	<*program name*> PGm (m: 1-100)	Delete program	2
	Pm-Pn (m: 0-29999, n: 0-29999)	Delete points "m - n"	
	PCm-PCn (m: 0-29999, n: 0-29999)	Delete point comments "m - n"	
	PNm-PNn (m: 0-29999, n: 0-29999)	Delete point names "m - n"	
	PLm (m: 0-39)	Delete pallet "m"	
IDIST	[*robot number*] k (robot number: 1-4, k: 1-10000)	Change inching movement amount	3
INCH INCHXY INCHT	[*robot number*] km (robot number: 1-4, k: 1-6, m: +/-)	Manual movement (inching)	3
INIT	ACO	Initialize area check output)	3
	ALL	Initialize all data (MEM+PRM)	
	CMU	Initialize communication parameter (RS-232C)	
	ETH	Initialize communication parameter (Ethernet)	
	GEP	Initialize General Ethernet Port	
	HND	Initialize hand data	
	ION	Initialize input/output name	
	LOG	Initialize alarm history	
	MEM	Initialize all data except parameters	
	PCM	Initialize point comment data	
	PGM	Initialize program data	
	PLT	Initialize pallet data	
	PNM	Initialize point name	
	PNT	Initialize point data	
	PRM	Initialize parameter data	
	SFT	Initialize shift data	
INPUT	SET d CAN CLR (d: input data)	Sets the input data to the data request by the INPUT statement	2
JOG JOGXY JOGT	[*robot number*] km (m: 1-4, k: 1-6, m: +/-)	Manual movement (jog)	3
LOAD	<*program name*> PGm , Tn , p (m: 1-100, n: 1-16, p: 1-64)	Register program in the task	2
MAINPG	m (m: 1-100)	Setting main program	2
MRKSET	[*robot number*] k (robot number: 1-4, k: 1-6)	Absolute reset	3
MSGCLR		Buffer clear Output message	1
MSPEED	[*robot number*] k (robot number: 1-4, k: 1-100)	Change manual movement speed	2

Command	Option	Meaning	Condition
NEXT	Tn *<program name>* PGm (m: 1-100, n: 1-16)	Execute program to next line	4
ORGRTN	[*robot number*] k (robot number: 1-4, k: 1-6)	Return-to-origin	3
READ	*read-out file*	Read-out data	2
REN	*<program 1>* TO *<program 2>*	Change program name from "1" to "2"	2
RESET	Tn *<program name>* PGm (m: 1-100, n: 1-16)	Reset program	2
RUN	Tn *<program name>* PGm (m: 1-100, n: 1-16)	Execute program	4
RUNTO	Tn *<program name>* PGm ,k (m: 1-100, n: 1-16, k: 1-9999)	Execute program before specified line	2
SEQCMPL		Compile sequence program	
SEQUENCE	k (k: 0/1/3)	Set sequence execution flag	2
SETPW		Setting password	
SKIP	Tn *<program name>* PGm (m: 1-100, n: 1-16)	Program: Skip one line	4
SKIPTO	Tn *<program name>* PGm ,k (m: 1-100, n: 1-16, k: 1-9999)	Program: Skip before specified line	2
STEP	Tn *<program name>* PGm (m: 1-100, n: 1-16)	Program: Execute one line	4
STOP	Tn *<program name>* PGm (m: 1-100, n: 1-16)	Stop program	2
SYNCHK	*<program name>* PGm , k (m: 1-100, k: 1-100)	Check program syntax	2
TEACH TCHXY	[*robot number*] m (robot number: 1-4, m: 0-29999)	Point data teaching	3
TIME	hh: mm: ss (hh: 00-23, mm: 00-59. ss: 00-59)	Check or set time	2
WRITE	*write file*	Write data	2
-		Robot language executable independently	4

Conditions: 1. Always executable.
2. Not executable during inputs from the programming box.
3. Not executable during inputs from the programming box, and while the program is running.
4. Not executable during inputs from the programming box, while the program is running, and when specific restrictions apply.

2.1 Program operations

1. Register task

Command format

@LOAD	<*program name*> PGm	,Tn, p [cr/lf]

Response format

OK[cr/lf]

Values
m ..... Program number: 1 to 100
n ..... Task number: 1 to 16
P ..... Task priority ranking: 1 to 64

Meaning Registers the specified program into "task n" with "priority p". The registered program enters the STOP status. When "task number n" is omitted, the task with the smallest number of those that have not been started is specified automatically. When "task priority p" is omitted, "32" is specified.
The smaller value, the higher priority. The larger value, the lower priority (high 1 to low 64).
When the task with a high task priority is in the RUNNING status, the task with a low task priority still remains in the READY status.

SAMPLE

```
Command:   @LOAD <PG_MAIN>, T1 [cr/lf]···Registers the
                                          program to task 1.
Response:  OK [cr/lf]
```

2. Reset program

Command format

1.@RESET	[cr/lf]	
2.@RESET	Tn <*program name*> PGm	[cr/lf]

Response format

OK[cr/lf]

Values
n ..... Task number: 1 to 16
m ..... Program number: 1 to 100

Meaning Executes the program reset.
Command format 1 resets all programs. When restarting the program, the main program or the program that has been executed last in task 1 is executed from its beginning.
Command format 2 resets only the specified program. When restarting the program that has been reset, this program is executed from its beginning.

SAMPLE

```
Command:   @RESET [cr/lf]·····Resets all programs.
Response:  OK [cr/lf]
Command:   @RESET T3 [cr/lf]····Resets only the program
                                that is executed by T3.
Response:  OK [cr/lf]
```

3. Program execution

Command format

```
1.@RUN  [cr/lf]
2.@RUN | Tn                | [cr/lf]
       | <program name>     |
       | PGm                |
```

Response format

```
OK[cr/lf]
```

Values n .....Task number: 1 to 16
m .....Program number: 1 to 100

Meaning Executes or stops the current program.
Command format 1 executes all programs in the STOP status.
Command format 2 executes only the specified program in the STOP status.

SAMPLE

```
Command:   @RUN [cr/lf]·····Executes all programs in
                            the STOP status.
Response:  OK [cr/lf]
Command:   @RUN T3 [cr/lf]·····Executes only the program
                            in the STOP status that is
                            registered in T3.
Response:  OK [cr/lf]
```

4. Stop program

Command format

```
1.@STOP [cr/lf]
2.@STOP | Tn              | [cr/lf]
        | <program name>  |
        | PGm             |
```

Response format

```
OK[cr/lf]
```

Values
n .....Task number: 1 to 16
m .....Program number: 1 to 100

Meaning Stops the program.
Command format 1 stops all programs.
Command format 2 stops only the specified program.

SAMPLE

```
Command:   @STOP [cr/lf] ·····  Stops all programs.
Response:  OK [cr/lf]
Command:   @STOP T3 [cr/lf]·····  Stops only the program
                                  that is executed by T3.
Response:  OK [cr/lf]
```

5. Execute one program line

Command format

```
@STEP | Tn              | [cr/lf]
      | <program name>  |
      | PGm             |
```

Command format

```
OK[cr/lf]
```

Values
n .....Task number: 1 to 16
m .....Program number: 1 to 100

Meaning Executes one line of the specified program. When executing one line of the GOSUB statement or CALL statement, the program operation enters the subroutine or sub-procedure.

SAMPLE

```
Command:   @STEP T3 [cr/lf]·····  Executes one line of the
                                  program that is executed
                                  by T3.
Response:  OK [cr/lf]
```

6. Skip one program line

Command format

@SKIP	Tn <*program name*> PGm	[cr/lf]

Response format

OK[cr/lf]

Values n .....Task number: 1 to 16
m .....Program number: 1 to 100

Meaning Skips one line of the specified program. When skipping one line of the GOSUB statement or CALL statement, all subroutines or sub-procedures are skipped.

SAMPLE

```
Command:  @SKIP T3 [cr/lf]·····Skips one line of the
                                program that is executed
                                by T3.
Response: OK [cr/lf]
```

7. Execute program to the next line

Command format

@NEXT	Tn <*program name*> PGm	[cr/lf]

Response format

OK[cr/lf]

Values n .....Task number: 1 to 16
m .....Program number: 1 to 100

Meaning Executes the specified program to the next line. Executing @NEXT on the line in the GOSUB or in the CALL statement make the program execute and return through the sub-procedure processing, then stop at the next line.

MEMO

- This is a same processing as setting the breakpoint on the next line in the program currently suspended and executing the program (@RUN).
  @STEP stops the program at the beginning line of the sub-procedure called by GOSUB or CALL statement.

SAMPLE

```
Command:  @NEXT T3 [cr/lf]·····Executes the program in
                                execution at T3 until the
                                next line.
Response: OK [cr/lf]
```

8. Execute program to line before specified line

Command format

```
@RUNTO | Tn                | , k [cr/lf]
       | <program name>    |
       | PGm               |
```

Command format

```
OK[cr/lf]
```

Values
n ..... Task number: 1 to 16
m ..... Program number: 1 to 100
k ..... Specified line number: 1 to 9999

Meaning Executes the specified program to the line before the specified line.

SAMPLE

```
Command:   @RUNTO T3, 15 [cr/lf]··· Executes the program that
                                    is executed by T3 to the
                                    14th line and stops at the
                                    15th line.
Response:  OK [cr/lf]
```

9. Skip program to line before specified line

Command format

```
@SKIPTO | Tn                | , k [cr/lf]
        | <program name>    |
        | PGm               |
```

Command format

```
OK[cr/lf]
```

Values
n ..... Task number: 1 to 16
m ..... Program number: 1 to 100
k ..... Specified line number: 1 to 9999

Meaning Skips the specified program to the line before the specified line.

SAMPLE

```
Command:   @SKIPTO T3, 15 [cr/lf]·· Skips the program that is
                                    executed by T3 to the 14th
                                    line and stops at the 15th
                                    line.
Response:  OK [cr/lf]
```

10. Set break point

Command format

```
1.@BREAK |<program name>|(n,n,n,...), k [cr/lf]
         | PGm          |
2.@BREAK |<program name>|0 [cr/lf]
         | PGm          |
3.@BREAK 0 [cr/lf]
```

Command format

```
OK[cr/lf]
```

Values

m .....Program number: 1 to 100
n .....Specified line number: 1 to 9999
k .....Set/Cancel: 0: Set, 1: Cancel

Meaning

Sets a break point to pause the program during program execution.
Command format 1 sets or cancels a break point in the specified line of the specified program. Multiple lines can also be specified.
Command format 2 cancels all break points set in the specified program.
Command format 3 cancels all break points.

SAMPLE

```
Command: @BREAK PG3 (1, 3), 1 [cr/lf] ··· Sets a break point in
                                           the first and third
                                           lines of PG3.
Response:  OK [cr/lf]
```

11. Check program syntax

Command format

```
@SYNCHK |<program name>| ,k [cr/lf]
        |PGm           |
```

Command format

```
RUN [cr/lf]
nnnn:gg.bbb [cr/lf]
nnnn:gg.bbb [cr/lf]
:
nnnn:gg.bbb [cr/lf]
nnnn:gg.bbb [cr/lf]
END [cr/lf]
```

Values

m .....Program number: 1 to 100
k .....Maximum number of error: 1 to 100
nnnn .....Line number where error occurred: 1 to 9999
gg.....Alarm group number
bbb .....Alarm classification number

Meaning Checks syntax of the program specified by *<program name>* or program number.
If there are syntax errors in the specified program, line number where error occurred, alarm group number and alarm classification number are output. For details regarding alarm group number and alarm classification number, refer to the "RCX340 Controller User's Manual" or "RCX340 Controller Operator's Manual".

SAMPLE

```
Command: @SYNCHK PG1, 100 [cr/lf]···· Sets a Maximum number of
                                      error at 100 and checks
                                      syntax of the program 1.
Response: RUN [cr/lf]
        1:5.239 [cr/lf]·········· Detects syntax errors
                                  "5.239: Illegal identifier"
                                  at 1th, 2nd, 3rd and 8th
                                  lines.
        2:5.239 [cr/lf]
        3:5.239 [cr/lf]
        8:5.239 [cr/lf]
        6:5.222 [cr/lf]·········· Detects syntax error
                                  "5.222:IF without ENDIF"
                                  at 6th line.
END [cr/lf]
```

12. Set main program

Command format

```
@MAINPG[cr/lf]
```

Response format

```
OK[cr/lf]
```

Values m: Program number .....1 to 100

Meaning Specifies the program which is always selected when all programs are reset. When "0" is specified at the main program number or program specified at the main program number doesn't exist, the program that has been executed last (current program) in the task 1 is selected after resetting all programs.

NOTE

- "Main program" corresponds conventional function "_SELECT" of RCX240, etc.

SAMPLE

```
Command:   @MAINPG 1[cr/lf] ···· Sets program number 1 at
                                  the main program.
Response:  OK[cr/lf]
```

13. Compile sequence program

Command format

```
@SEQCMPL[cr/lf]
```

Response format

```
RUN[cr/lf]
END[cr/lf]
```

Meaning Compiles the sequence program.
When the program named "SEQUENCE" doesn't exist or syntax errors exist in the program, an error message appears.
The execution program is created after successful termination of compiling and the letter "s" appears in Flag.
For details, refer to Chapter 7 "Sequence function".

SAMPLE

```
Command:   @SEQCMPL[cr/lf] ····· Compiles the sequence program.
Response:  RUN[cr/lf]
           END[cr/lf]
```

2.2 MANUAL mode operation

1. Change the MANUAL mode speed

Command format

```
@MSPEED [robot number] k[cr/lf]
```

Response format

```
OK[cr/lf]
```

Values *robot number* ..... 1 to 4 (If not input, robot 1 is specified.)
k ..... Manual movement speed: 1 to 100

Meaning Changes the manual mode movement speed of the robot specified by the *<robot number>*.

SAMPLE

```
Command:   @MSPEED 50[cr/lf]
Response:  OK[cr/lf]
```

2. Point data teaching

Command format

```
@TEACH [robot number] mmmmm[cr/lf]
@TCHXY [robot number] mmmmm[cr/lf]
```

Response format

```
OK[cr/lf]
```

Values *robot number* ..... 1 to 4 (If not input, robot 1 is specified.)
mmmmm ..... Point number for registering point data: 0 to 29999

Meaning Registers the current robot position as point data for the specified point number. If point data is already registered in the specified point number, then that point data will be overwritten.

The unit of the point data may vary depending on the command.
TEACH ..... "pulse" units
TCHXY ..... "mm" units

SAMPLE

```
Command:   @TEACH[2] 100[cr/lf]
Response:  OK[cr/lf]
```

3. Change inching movement amount

Command format

```
@IDIST [robot number] mmmmm [cr/lf]
```

Response format

```
OK[cr/lf]
```

Values *Robot number*.....1 to 4 (If not input, robot 1 is specified.)
mmmmm: inching movement amount....1 to 10000

Meaning Changes the inching movement amount of the robot specified by the <*robot number*>.

The unit of the movement amount may vary depending on the command.
INCH ......"pulse" units: 1 to 10000 pulse
INCHXY ....."mm" units: 0.001 to 10.000mm
INCHXT ....."mm" units: 0.001 to 10.000mm

SAMPLE

```
Command:   @IDIST[2] 100[cr/lf]
Response:  OK[cr/lf]
```

2.3 Alarm reset

Command format

```
@ALMRST [cr/lf]
```

Response format

```
RUN[cr/lf]
END[cr/lf]
```

Meaning Resets the alarm.
However, this command cannot be used for the alarms which require the restart of system. In this case, turn off the controller and turn it on again.

SAMPLE

```
Command:   @ALMRST [cr/lf]
Response:  RUN[cr/lf]
           END[cr/lf]
```

2.4 Clearing output message buffer

Command format

```
@MSGCLR [cr/lf]
```

Response format

```
OK[cr/lf]
```

Values Clears the output message buffer of the controller. After the messages have been output by the PRINT statement, etc., the messages remaining in the buffer are cleared.

SAMPLE

```
Command:   @MSGCLR [cr/lf]
Response:  OK[cr/lf]
```

2.5 Setting input data

Command format

```
@INPUT | SET d | [cr/lf]
       | CAN
       | CLR
```

Response format

```
OK[cr/lf]
```

Values d: Input data ..... Value that is matched to the type of the variable specified by the INPUT statement.
(Character string is enclosed by " ")

Meaning Sets the input data for responding to a data request by INPUT statement of robot program.
The controller parameter "INPUT/PRINT using channel" should be set a current communication channel (CMU, ETH or iVY).

SET..... Sets the data which is input to the variable when INPUT statement is executed.
CAN..... Cancels the data request by INPUT statement.
CLR..... Clears the data specified @INPUT SET downward.

SAMPLE

```
<Online command>                    <Robot program>
@INPUT SET 10[cr/lf]
@INPUT SET 5[cr/lf]
OK[cr/lf]                           INPUT A%[cr/lf]
@?MSG[cr/lf]                        PRINT A%[cr/lf]
10[cr/lf]
OK[cr/lf]

<Online command>                    <Robot program>
@INPUT SET 10[cr/lf]
OK[cr/lf]
@INPUT CLR[cr/lf]
OK[cr/lf]
@INPUT SET 5[cr/lf]
OK[cr/lf]                           INPUT A%[cr/lf]
@?MSG[cr/lf]                        PRINT A%[cr/lf]
5[cr/lf]
OK[cr/lf]
```

2.6 Change access level

Command format

```
@ACCESS k ,pppppppp[cr/lf]
```

Response format

```
OK[cr/lf]
```

Values k: Access level .....0: Maintainer level, 1: Operator level
pppppppp: Password.....Alphanumeric characters of 8 characters or less

Meaning Changes access level. If password is omitted, sets without password.
When changes access level to the maintainer level and entered password is incorrect, "6.235: Password error" will occur.

REFERENCE

- For details regarding access level, refer to the RCX340 user's manual or operator's manual.

SAMPLE

```
Command: @ACCESS 0,password [cr/lf] ···· Sets "password" as password,
                                         and changes the level to
                                         "maintainer level".
Response: OK [cr/lf]
```

2.7 Setting input data

Command format

```
@SETPW [cr/lf]
```

Response format

```
READY[cr/lf]
pppppppp[cr/lf]
kkkkkkkk[cr/lf]
nnnnnnnn[cr/lf]
[cr/lf] ····· line-feed
OK[cr/lf]
```

Values

pppppppp: old password (current password)..... Alphanumeric characters of 8 characters or less
kkkkkkkk: new password ..... Alphanumeric characters of 8 characters or less
nnnnnnnn: new password (confirmation)..... Alphanumeric characters of 8 characters or less

Meaning

Changes the password for the access level changing to the maintainer level.
The current password is input for the old password, and the revised password is input for the new password and for the new password of confirmation. In the next line of the new password (confirmation), inserts line feeds only.
When input password as the old password is different from the current password or new password and new password (confirmation) are not same, "6.235: Password error" will occur.

REFERENCE
- For details regarding access level, refer to the RCX340 user's manual or operator's manual.

SAMPLE

```
Command:   @SETPW[cr/lf]
Response:  READY [cr/lf]
           oldpass [cr/lf] ····· Inputs "oldpass" as old
                                 password.
           newpass [cr/lf] ····· Inputs "newpass" as new
                                 password.
           newpass [cr/lf] ····· Inputs "newpass" as new
                                 password (confirmation).
           [cr/lf] ·············· line-feed
           OK [cr/lf]
```

3.1 Acquiring return-to-origin status

Command format 1

```
@?ORIGIN[cr/lf]
```

Response format 1

```
x [cr/lf]
OK [cr/lf]
```

Command format 2

```
@?ORIGIN robot number [cr/lf]
```

Response format 2

```
x y{,y{,{...}}} [cr/lf]
OK [cr/lf]
```

Values

Robot number ..... 1 to 4 (If not input, robot 1 is specified.)
x: Robot return-to-origin status ..... 0: Incomplete, 1: Complete
y: Axis return-to-origin status ..... Shows the status of the axis 1, axis 2, …, axis 6 from the left.
0: Incomplete, 1: Complete
(Omitted when the axis is not connected.)

Meaning Acquires return-to-origin status.
Command format 1 acquires the return-to-origin status of all robots while command format 2 acquires the status of the specified robot.

SAMPLE

```
Command:   @?ORIGIN 2 [cr/lf]
Response:  0 1,1,0,1 ·····Axis 3 of the robot 2 is
                          in the return-to-origin
                          incomplete status.
           OK [cr/lf]
```

3.2 Acquiring the servo status

Command format

```
@?SERVO [robot number] [cr/lf]
```

Response format

```
x y{,y{,{...}}} [cr/lf]
OK [cr/lf]
```

Values

Robot number ..... 1 to 4 (If not input, robot 1 is specified.)
x: Robot servo status..... 0: Servo off status
1: Servo on status
y: Axis servo status ..... Shows the status of the axis 1, axis 2, …, axis 6 from the left.
0: Mechanical brake on + dynamic brake on status
1: Servo on status
2: Mechanical brake off + dynamic brake off status
(Omitted when the axis is not connected.)

Meaning Acquires the servo status.

SAMPLE

```
Command:   @?SERVO[3] [cr/lf]
Response:  0 0,1,0,0 ·····Only the axis 2 of the
                          robot 3 is in the servo on
                          status.
           OK [cr/lf]
```

3.3 Acquire motor power status

Command format

```
@?MOTOR [cr/lf]
```

Response format

```
x [cr/lf]
OK [cr/lf]
```

Values

x: Motor power status.....0: Motor power off status
1: Motor power on status
2: Motor power on + all robot servo on status

Meaning Acquires the motor power status.

SAMPLE

```
Command:   @?MOTOR [cr/lf]
Response:  2
           OK [cr/lf]
```

3.4 Acquiring the access level

Command format

```
@?ACCESS[cr/lf]
```

Response format

```
k[cr/lf]
OK[cr/lf]
```

Values k: Access level .....0 to 1

Meaning Acquires the access level.

REFERENCE

- For details regarding access level, refer to the RCX340 user's manual or operator's manual.

SAMPLE

```
Command:   @?ACCESS[cr/lf]
Response:  1[cr/lf]
           OK[cr/lf]
```

3.5 Acquiring the break point status

Command format

```
@?BREAK | <program name> | [cr/lf]
        | PGm
```

Response format

```
n{,n{,{...}}} [cr/lf]
OK [cr/lf]
```

Values n: Line number on which break point "n" is set ..... 1 to 9999
Program name..... Program name intended to delete
m: Program number ..... 1 to 100

Meaning Acquires the break point status.

SAMPLE

```
Command:   @?BREAK <TEST>[cr/lf]
Response:  12,35[cr/lf]
           OK[cr/lf]
```

3.6 Acquiring the mode status

Command format

```
@?MODE[cr/lf]
```

Response format

```
k[cr/lf]
OK[cr/lf]
```

Values k: Mode status ..... 0: MANUAL mode
1: AUTO mode (Control source: Programming box)
2: AUTO mode (Control source release)
-1: Restricted mode

Meaning Acquires the controller mode status.

SAMPLE

```
Command:   @?MODE[cr/lf]
Response:  1[cr/lf]
           OK[cr/lf]
```

3.7 Acquiring the communication port status

Command format

```
@?LINEMODE | ETH | [cr/lf]
           | CMU |
```

Response format

```
k[cr/lf]
OK[cr/lf]
```

Values k ..... 0: OFFLINE, 1: ONLINE

Meaning Acquires the specified communication port status.
ONLINE / OFFLINE commands allow to change a specified communication port to the "online" / "offline" mode, respectively.

SAMPLE

```
Command:   @?LINEMODE ETH [cr/lf]
Response:  1[cr/lf]
           OK[cr/lf]
```

3.8 Acquiring the main program number

Command format

```
@?MAINPG[cr/lf]
```

Response format

```
m[cr/lf]
OK[cr/lf]
```

Values m: Program number..... 0 to 100
(If not registered in the main program, acquires 0.)

Meaning Acquires the program number which is registered in the main program.

SAMPLE

```
Command:   @?MAINPG[cr/lf]
Response:  1[cr/lf]
           OK[cr/lf]
```

3.9 Acquiring the sequence program execution status

Command format

```
@?SEQUENCE[cr/lf]
```

Response format

```
1. 1,s[cr/lf]
   OK[cr/lf]
2. 3,s[cr/lf]
   OK[cr/lf]
3. 0[cr/lf]
   OK[cr/lf]
```

Values s .....The sequence program's execution status is indicated as 1 or 0.
(1: Program execution is in progress. 0: Program execution is stopped.)

Meaning Acquires the sequence program execution status.
Response output means as follows:
1 .....Enabled
3 .....Enabled and output is cleared at emergency stop
0 .....Disabled

SAMPLE

```
Command:   @?SEQUENCE[cr/lf]
Response:  0[cr/lf]
           OK[cr/lf]
```

3.10 Acquiring the version information

Command format

```
@?VER[cr/lf]
```

Response format

```
cv,cr-mv-dv1,dr1/dv2,dr2[cr/lf]
```

Values
- cv ..... Host version number
- cr ..... Host revision number (Rxxxx)
- mv ..... PLO version number (Vx.xx)
- dv? (?: 1, 2) ..... Driver version number (Vx.xx)
- dr? (?: 1, 2) ..... Driver revision number (Rxxxx)

Meaning Acquires the version information.

SAMPLE

```
Command:   @?VER[cr/lf]
Response:  V8.02,R1021-V5.10-V1.01,R0001/V1.01,R0001[cr/lf]
           OK[cr/lf]
```

3.11 Acquiring the tasks in RUN or SUSPEND status

Command format

```
@?TASKS[cr/lf]
```

Response format

```
n{,n{,{...}}}[cr/lf]
OK[cr/lf]
```

Values n: Task number ..... 1 to 16 (Task currently run or suspended)

Meaning Acquires the tasks in RUN or SUSPEND status.

SAMPLE

```
Command:   @?TASKS[cr/lf]
Response:  1,3,4,6[cr/lf]
           OK[cr/lf]
```

3.12 Acquiring the tasks operation status

Command format

```
@?TSKMON Tk[cr/lf]
```

Response format

```
m,n,f,p[cr/lf]
OK [cr/lf]
```

Values

k : Task number ..... 1 to 16
m : Execution program number ..... 1 to 100
n : Task execution line number..... 1 to 9999
f : Each task status..... R: RUN
U: SUSPEND
S: STOP
W: WAIT
p : Priority level of each task ..... 17 to 47

Meaning Acquires the status of specified task.

SAMPLE

```
Command:   @?TSKMON T3[cr/lf]
Response:  5,11,R,32[cr/lf]
           OK[cr/lf]
```

3.13 Acquiring the task end condition

Command format

```
@?TSKECD Tk[cr/lf]
```

Response format

```
gg.bbb[cr/lf]
OK[cr/lf]
```

Values

k : Task number .....1 to 16
gg : Alarm group number of the task end condition
bbb : Alarm classification number of the task end condition

Meaning Acquires the specified task end condition.
For details about alarm group number and classification number of the task end condition, refer to RCX340 user's or operator's manual.

MEMO

- When the specified task ends by error, acquires this alarm number.

SAMPLE

```
Command: @?TSKECD T1[cr/lf]····Acquires the end condition of task 1.
Response:1.5[cr/lf]·············The end condition of task 1:
                                1.5: Program ended by "HALT".
         OK[cr/lf]
```

3.14 Acquiring the shift status

Command format

```
@?SHIFT[robot number][cr/lf]
```

Response format

```
m[cr/lf]
OK[cr/lf]
```

Values *Robot number* .....1 to 4 (If not input, robot 1 is specified.)
m: .....Shift number selected for the specified robot: 0 to 39
Shift not selected: -1

Meaning Acquires the shift status of the robot specified by the <*robot number*>.

SAMPLE

```
Command:   @?SHIFT[cr/lf]
Response:  1[cr/lf]
           OK[cr/lf]
```

3.15 Acquiring the hand status

Command format

```
@?HAND[robot number][cr/lf]
```

Response format

```
m[cr/lf]
```

Values *Robot number* .....1 to 4 (If not input, robot 1 is specified.)
m.....Hand number selected for the specified robot: 0 to 31
Hand not selected: -1

Meaning Acquires the hand status of the robot specified by the <*robot number*>.

SAMPLE

```
Command:   @?HAND[cr/lf]
Response:  1[cr/lf]
           OK[cr/lf]
```

3.16 Acquiring the remaining memory capacity

Command format

```
@?MEM[cr/lf]
```

Response format

```
k/m[cr/lf]
```

Values k .....Remaining source area (unit: bytes)
m .....Remaining global identifier area (unit: bytes)

Meaning Acquires the remaining memory capacity.

SAMPLE

```
Command:   @?MEM[cr/lf]
Response:  102543/1342[cr/lf]
           OK[cr/lf]
```

3.17 Acquiring the alarm status

Command format

```
@?ALM[cr/lf]
```

Response format

```
gg.bbb[cr/lf]
OK[cr/lf]
```

Values gg.....Alarm group number
bbb .....Alarm classification number

Meaning Acquires the alarm which occurs in the controller.
For details regarding the alarm group number and alarm classification number, refer to the RCX340 user's or operator's manual.

MEMO

- The requirable alarms are number 400 or more of alarm classification number. If multiple alarms occur, the alarm with larger alarm classification number (more serious alarm) is acquired.

SAMPLE

```
Command:   @?ALM[cr/lf]
Response:  12.600[cr/lf]
           OK[cr/lf]
```

3.18 Acquiring the emergency stop status

Command format

```
@?EMG[cr/lf]
```

Response format

```
k[cr/lf]
OK[cr/lf]
```

Values k: Emergency stop status .....0: normal operation, 1: emergency stop

Meaning Acquires the emergency stop status by checking the internal emergency stop flag.

SAMPLE

```
Command:   @?EMG[cr/lf]
Response:  1[cr/lf]
           OK[cr/lf]
```

3.19 Acquiring the manual movement speed

Command format

```
@?MSPEED [robot number] [cr/lf]
```

Response format

```
k[cr/lf]
OK[cr/lf]
```

Values *Robot number*.....1 to 4 (If not input, robot 1 is specified.)
k: manual movement speed ...1 to 100 (unit: %)

Meaning Acquires the value of the manual movement speed specified by <*Robot number*>.

SAMPLE

```
Command:   @?MSPEED[cr/lf]
Response:  50[cr/lf]
           OK[cr/lf]
```

3.20 Acquiring the inching movement amount

Command format

```
@?IDIST [robot number] [cr/lf]
```

Response format

```
mmmmm[cr/lf]
OK[cr/lf]
```

Values *Robot number*.....1 to 4 (If not input, robot 1 is specified.)
mmmmm: Inching movement amount .....1 to 10000

Meaning Acquires the inching movement amount specified by <*Robot number*>.

SAMPLE

```
Command:   @?IDIST[2][cr/lf]
Response:  100[c/lf]
           OK[cr/lf]
```

3.21 Acquiring the last reference point number (current point number)

Command format

```
@?CURPNT[cr/lf]
```

Response format

```
k[cr/lf]
OK[cr/lf]
```

Values k: Current point number.....0 to 29999

Meaning Acquires the point number which is referred last. The current point number (the point number of last reference) is renewed by operations which uses the point data (point edit, for example).

MEMO

- The current point number is renewed by following operations: the point reference and the point setting movement by remote commands, the trace movement or teaching by programming box or RCX-Studio Pro, etc.

SAMPLE

```
Command:   @?CURPNT[cr/lf]
Response:  100[cr/lf]
           OK[cr/lf]
```

3.22 Acquiring the output message

Command format

```
@?MSG[cr/lf]
```

Response format

```
ssssss···· ssssss[cr/lf]
OK[cr/lf]
```

Values s: Message character string

Meaning Acquires one line of message which is input from the output message buffer of the controller by the PRINT statement, etc.

SAMPLE

```
Command:   @?MSG[cr/lf]
Response:  MESSAGE[cr/lf]·····PRINT  "MESSAGE"  is
                              executed in a program.
           OK[cr/lf]
```

MEMO

- For executing this command, it is required that the "INPUT/PRINT using channel" parameter is set at the port to execute command.
- When the output message buffer is empty, only "OK" is output as the response.

3.23 Acquiring the input data

Command format

```
@?INPUT[cr/lf]
```

Response format

```
d[cr/lf]
OK[cr/lf]
```

Values d: Input data

Meaning Acquires the input data by the INPUT statement.

SAMPLE

```
Command:   @?INPUT[cr/lf]
Response:  INPUT_SAMPLE[cr/lf]
           OK[cr/lf]
```

3.24 Acquiring various values

1. Acquiring the value of a numerical expression

Command format

```
@?numerical expression[cr/lf]
OK[cr/lf]
```

Response format

```
numerical value[cr/lf]
```

Meaning Acquires the value of the specified numerical expression.
The numerical expression's value format is "decimal" or "real number".

SAMPLE 1

```
Command:   @?SQR(100*5)[cr/lf]
Response:  2.236067E01[cr/lf]
           OK[cr/lf]
```

SAMPLE 2

```
Command:   @?LOC1(WHERE)[cr/lf]
Response:  102054[cr/lf]
           OK[cr/lf]
```

2. Acquiring the value of a character string expression

Command format

```
@?character string expression[cr/lf]
```

Response format

```
character string[cr/lf]
OK[cr/lf]
```

Meaning Acquires the value (character string) of the specified character string expression.

SAMPLE

```
The case of A$="ABC" and B$="DEF".
Command:   @?A$+B$+"123"[cr/lf]
Response:  ABCDEF123[cr/lf]
           OK[cr/lf]
```

3. Acquiring the value of a point expression

Command format

```
@?point expression[cr/lf]
```

Response format

```
point data[cr/lf]
OK[cr/lf]
```

Meaning Acquires the value (point data) of the specified point expression.

SAMPLE

```
Command:   @?P1+WHRXY[cr/lf]
Response:  10.410 -1.600 52.150 3.000 0.000 0.000 0 0 0[cr/lf]
           OK[cr/lf]
```

4. Acquiring the value of a shift expression

Command format

```
@?shift expression[cr/lf]
OK[cr/lf]
```

Response format

```
shift data[cr/lf]
```

Meaning Acquires the value (shift data) of the specified shift expression.

SAMPLE

```
Command:   @?s1[cr/lf]
Response:  25.000  12.600  10.000  0.000[cr/lf]
           OK[cr/lf]
```

4.1 Absolute reset

Command format

```
@ABSADJ[robot number] k,f[cr/lf]
@MRKSET[robot number] k[cr/lf]
```

Response format

```
RUN[cr/lf] ·····At movement start
END[cr/lf] ·····At movement end
```

Values

Robot number ..... 1 to 4 (If not input, robot 1 is specified.)
k ..... Axis number: 1 to 6
f ..... Movement direction / 0: + direction, 1: - direction

Meaning

Performs the absolute reset operation of the specified axis of the robot specified by the *<robot number>*.
This command is available only to axes whose return-to-origin method is set as "Mark".
ABSADJ ..... Moves the specified robot axis to an absolute reset position.
MRKSET ..... Performs absolute reset on the specified robot axis.

SAMPLE

```
Command:   @ABSADJ 1,0[cr/lf]
Response:  RUN[cr/lf] ···········Movement start
           END[cr/lf] ···········Movement end
```

4.2 Return-to-origin operation

Command format

```
@ORGRTN[robot number] k[cr/lf]
```

Response format

```
RUN[cr/lf] ·····At movement start
END[cr/lf] ·····At movement end
```

Values *Robot number*.....1 to 4 (If not input, robot 1 is specified.)
k .....Axis number: 1 to 6

Meaning Performs the return-to-origin operation of the specified axis of the robot specified by the <*robot number*>.
For the axis with the semi-absolute specifications, when the return-to-origin is executed, the absolute search operation is performed.

SAMPLE

```
Command:   @ORGRTN 1[cr/lf]
Response:  RUN[cr/lf] ····· Movement start
           END[cr/lf] ····· Movement end
```

4.3 Manual movement: inching

Command format

```
@INCH[robot number] km [cr/lf]
@INCHXY[robot number] km [cr/lf]
@INCHT[robot number] km [cr/lf]
```

Response format

```
RUN[cr/lf] ·····At movement start
END[cr/lf] ·····At movement end
```

Values

Robot number.....1 to 4 (If not input, robot 1 is specified.)
k .....Axis number: 1 to 6
m .....Movement direction / +, -

Meaning

Manually moves (inching motion) the specified axis of the robot specified by the <*robot number*>.
The robot performs the same motion as when moved manually in inching motion with the programming box's jog keys (moves a fixed distance each time a jog key is pressed).

The unit of the movement amount and operation type by command are shown below.
INCH .....“pulse” units. Only the specified axis moves.
INCHXY .....“mm” units. According to the robot configuration, the arm tip of the robot moves in the direction of the Cartesian coordinate system.
INCHT .....“mm” units. According to the robot configuration, the hand attached to the arm tip of the robot moves.

SAMPLE

```
Command:   @INCH 1+[cr/lf]
Response:  RUN[cr/lf] ···········Movement start
           END[cr/lf] ···········Movement end
```

4.4 Manual movement: jog

Command format

```
@JOG[robot number] km [cr/lf]
@JOGXY[robot number] km [cr/lf]
@JOGT[robot number] km [cr/lf]
```

Response format

```
RUN[cr/lf] ·····At movement start
END[cr/lf] ·····At movement end
```

Values

Robot number.....1 to 4 (If not input, robot 1 is specified.)
k .....Axis number: 1 to 6
m .....Movement direction / +, -

Meaning

Manually moves (jog motion) the specified axis of the robot specified by the <*robot number*>.
The robot performs the same motion as when holding down the programming box's jog keys in manual mode.

To continue the operation, it is necessary for the JOG command to input the execution continue process (^V(=16H)) by the online command at intervals of 200ms. If not input, the error stop occurs.

Additionally, after the movement has started, the robot stops when any of the statues shown below arises.

- When software limit was reached.
- When stop signal was turned off.
- When STOP key on the programming box was pressed.
- When an online command (^C (=03H)) to interrupt execution was input.

The unit of the movement amount and operation type by command are shown below.

JOG .....”pulse” units. Only the specified axis moves.
JOGXY.....”mm” units. According to the robot configuration, the arm tip of the robot moves in the direction of the Cartesian coordinate system.
JOGT .....”mm” units. According to the robot configuration, the hand attached to the arm tip of the robot moves.

SAMPLE

```
Command:   @JOG 1+[cr/lf]
Response:  RUN[cr/lf] ····· Movement start
           END[cr/lf] ····· Movement end
```

5 Data file operation commands

5.1 Copy operations

1. Copying a program

Command format

@COPY	*<program name 1>* PGn	TO *<program name 2>* [cr/lf]

Response format

```
RUN[cr/lf] ·····At prosess start
END[cr/lf] ·····At prosess end
```

Values

Program name 1 ..... Program name in copy source (32 characters or less consisting of alphanumeric characters and underscore)

Program name 2 ..... Program name in copy destination (32 characters or less consisting of alphanumeric characters and underscore)

n: Program number ..... 1 to 100

Meaning Copies the program specified by *<program name 1>* or program number to *<program name 2>*.

SAMPLE

```
Command:   @COPY <TEST1> TO <TEST2> [cr/lf]
Response:  RUN [cr/lf] ····· Process start
           END [cr/lf] ····· Process end
```

2. Copying point data

Command format

```
@COPY   Pmmmmm-Pnnnnn   TO   Pkkkkk[cr/lf]
```

Response format

```
RUN[cr/lf] ·····At prosess start
END[cr/lf] ·····At prosess end
```

Values

mmmmm ..... Top point number in copy source: 0 to 29999

nnnnn ..... Last point number in copy source: 0 to 29999

kkkkk ..... Top point number in copy destination: 0 to 29999

Meaning Copies the point data between Pmmmmm and Pnnnnn to Pkkkkk.

SAMPLE

```
Command:   @COPY P101-P200 TO P1101[cr/lf]
Response:  RUN [cr/lf] ····· Process start
           END [cr/lf] ····· Process end
```

3. Copying point comments

Command format

```
@COPY   PCmmmmm-PCnnnnn   TO   PCkkkkk[cr/lf]
```

Response format

```
RUN[cr/lf] ·····At prosess start
END[cr/lf] ·····At prosess end
```

Values
mmmmm .....Top point comment number in copy source: 0 to 29999
nnnnn .....Last point comment number in copy source: 0 to 29999
kkkkk .....Top point comment number in copy destination: 0 to 29999

Meaning Copies the point comments between PCmmmmm and PCnnnnn to PCkkkkk.

SAMPLE

```
Command:   @COPY PC101-PC200 TO PC1101[cr/lf]
Response:  RUN [cr/lf] ····· Process start
           END [cr/lf] ····· Process end
```

5.2 Erase

1. Erasing a program

Command format

@ERA	<*program name*> PGn	[cr/lf]

Response format

```
RUN[cr/lf] ·····At prosess start
END[cr/lf] ·····At prosess end
```

Values
Program name .....Program name to be erased (32 characters or less consisting of alphanumeric characters and underscore)
n: Program number .....1 to 100

Meaning Erases the designated program.

SAMPLE

```
Command:   @ERA <TEST1> [cr/lf]
Response:  RUN [cr/lf] ····· Process start
           END [cr/lf] ····· Process end
```

2. Erasing point data

Command format

```
@ERA   Pmmmmm-Pnnnnn[cr/lf]
```

Response format

```
RUN[cr/lf] ·····At prosess start
END[cr/lf] ·····At prosess end
```

Values mmmmm ..... Top point number to be erased: 0 to 29999
nnnnn ..... Last point number to be erased: 0 to 29999

Meaning Erases the point data between Pmmmmm and Pnnnnn.

SAMPLE

```
Command:   @ERA P101-P200[cr/lf]
Response:  RUN [cr/lf] ····· Process start
           END [cr/lf] ····· Process end
```

3. Erasing point comments

Command format

```
@ERA   PCmmmmm-PCnnnnn[cr/lf]
```

Response format

```
RUN[cr/lf] ·····At prosess start
END[cr/lf] ·····At prosess end
```

Values mmmmm ..... Top point comment number to be erased: 0 to 29999
nnnnn ..... Last point comment number to be erased: 0 to 29999

Meaning Erases the point comments between PCmmmmm and PCnnnnn.

SAMPLE

```
Command:   @ERA PC101-PC200[cr/lf]
Response:  RUN [cr/lf] ····· Process start
           END [cr/lf] ····· Process end
```

4. Erasing point name

Command format

```
@ERA PNmmmmm-PNnnnnn [cr/lf]
```

Response format

```
RUN[cr/lf] ·····At prosess start
END[cr/lf] ·····At prosess end
```

Values mmmmm .....Top point name number to be erased: 0 to 29999
nnnnn .....Last point name number to be erased: 0 to 29999

Meaning Erases the point names between PNmmmmm and PNnnnnn.

SAMPLE

```
Command:   @ERA PC101-PC200[cr/lf]
Response:  RUN [cr/lf] ····· Process start
           END [cr/lf] ····· Process end
```

5. Erasing pallet data

Command format

```
@ERA PLm[cr/lf]
```

Response format

```
RUN[cr/lf] ·····At prosess start
END[cr/lf] ·····At prosess end
```

Values m .....Pallet number to be erased: 0 to 39

Meaning Erases the PLm pallet data.

SAMPLE

```
Command:   @ERA PL1[cr/lf]
Response:  RUN [cr/lf] ····· Process start
           END [cr/lf] ····· Process end
```

6. Erasing hand

Command format

```
@ERA Hm [cr/lf]
```

Response format

```
RUN[cr/lf] ·····At prosess start
END[cr/lf] ·····At prosess end
```

Values m .....Hand number to be erased: 0 to 31

Meaning Erases the hand definition data of "Hm".

SAMPLE

```
Command:   @ERA H2 [cr/lf]
Response:  RUN [cr/lf] ····· Process start
           END [cr/lf] ····· Process end
```

7. Erasing shift

Command format

```
@ERA Sm [cr/lf]
```

Response format

```
RUN[cr/lf] ·····At prosess start
END[cr/lf] ·····At prosess end
```

Values m .....Shift number to be erased: 0 to 39

Meaning Erases the shift data of "Sm".

SAMPLE

```
Command:   @ERA S1 [cr/lf]
Response:  RUN [cr/lf] ····· Process start
           END [cr/lf] ····· Process end
```

8. Erasing area check output setting

Command format

```
@ERA ACm [cr/lf]
```

Response format

```
RUN[cr/lf] ·····At prosess start
END[cr/lf] ·····At prosess end
```

Values m .....Area check output setting number to be erased: 0 to 31

Meaning Erases the area check output setting of "ACm".

SAMPLE

```
Command:   @ERA AC3 [cr/lf]
Response:  RUN [cr/lf] ····· Process start
           END [cr/lf] ····· Process end
```

9. Erasing general-purpose Ethernet port

Command format

```
@ERA GPm [cr/lf]
```

Response format

```
RUN[cr/lf] ·····At prosess start
END[cr/lf] ·····At prosess end
```

Values m .....General-purpose Ethernet port number to be erased: 0 to 15

Meaning Erases the general-purpose Ethernet port of "GPm".

SAMPLE

```
Command:   @ERA GP5 [cr/lf]
Response:  RUN [cr/lf] ····· Process start
           END [cr/lf] ····· Process end
```

5.3 Rename program

Command format

@REN	*<program name 1>* PGn	TO *<program name 2>* [cr/lf]

Response format

```
RUN[cr/lf] ·····At prosess start
END[cr/lf] ·····At prosess end
```

Values

Program name 1 ..... Program name before renaming: shown with 32 characters or less consisting of alphanumeric characters and _ (underscore)

Program name 2 ..... Program name after renaming: shown with 32 characters or less consisting of alphanumeric characters and _ (underscore)

n: Program number.... 1 to 100

Meaning Changes the name of the specified program.

SAMPLE

```
Command:   @REN <TEST1> TO <TEST2>[cr/lf]
Response:  RUN [cr/lf] ····· Process start
           END [cr/lf] ····· Process end
```

5.4 Changing the program attribute

Command format

@ATTR	*<program name>* PGn	TO s [cr/lf]

Response format

```
OK[cr/lf]
```

Values

Program name ..... Program name to change the attribute: shown with 32 characters or less consisting of alphanumeric characters and _ (underscore)

s: Attribute..... RW: Readable/writable
RO: Not writable (read only)
H: Hidden

n: Program number.... 1 to 100

Meaning Changes the attribute of the program specified by the *<program name>* or program number.

SAMPLE

```
Command:   @ATTR <TEST1> TO RO[cr/lf]
Response:  OK[cr/lf]
```

5.5 Initialization process

1. Initializing the memory area

Command format

```
@INIT memory area[cr/lf]
```

Response format

```
RUN[cr/lf] ·····At prosess start
END[cr/lf] ·····At prosess end
```

Values

Memory area.....Memory area to be initialized.
One of the following memory areas is specified.
PGM .....Initializes the program area.
PNT .....Initializes the point data area.
SFT.....Initializes the shift data area.
HND.....Initializes the hand data area.
PLT .....Initializes the pallet data area.
PCM.....Initializes the point comment area.
PNM .....Initializes the point name area.
ION .....Initializes the input/output name area.
ACO .....Initializes the area check output setting area.
GEP.....Initializes the general-purpose Ethernet port setting area.
MEM.....Initializes the above areas (PGM ... all data up to GEP).
PRM.....Initializes the parameter area.
ALL .....Initializes all areas (MEM+PRM).

Meaning Initializes the memory area.

SAMPLE

```
Command:   @INIT PGM[cr/lf]
Response:  RUN [cr/lf] ····· Process start
           END [cr/lf] ····· Process end
```

2. Initializing the communication port

Command format

```
@INIT communication port [cr/lf]
```

Response format

```
RUN[cr/lf] ·····At prosess start
END[cr/lf] ·····At prosess end
```

Values *Communication port* .....Communication port to be initialized
Specify any of the ports shown below for the communication port.
CMU .....Initializes the RS-232C port.
ETH.....Initializes the Ethernet port.

Meaning Initializes the communication port.
For information about the communication port initial settings, refer to the RCX340 user's or operator's manual.

SAMPLE

```
Command:   @INIT CMU [cr/lf]
Response:  RUN [cr/lf] ····· Process start
           END [cr/lf] ····· Process end
```

3. Initializing the alarm history

Command format

```
@ INIT LOG[cr/lf]
```

Response format

```
RUN[cr/lf] ·····At prosess start
END[cr/lf] ·····At prosess end
```

Meaning Initializes the alarm history.

SAMPLE

```
Command:   @INIT LOG[cr/lf]
Response:  RUN [cr/lf] ····· Process start
           END [cr/lf] ····· Process end
```

5.6 Data readout processing

Command format

```
@READ read-out file[cr/lf]
```

Response format

```
BEGIN [cr/lf] ····· At process start
(Data output: The contents may vary depending on the read-out file.)
END [cr/lf] ····· At process end
```

Values *Read-out file*..... Designates a read-out file name.

Meaning Reads out the data from the designated file.

Online commands that are input through the RS-232C port have the same meaning as the following command.

- SEND <*read-out file*> TO CMU

Commands via Ethernet have the same meaning as the following command.

- SEND <*read-out file*> TO ETH

NOTE

- For more information about files, refer to the earlier Chapter 10 "Data file description".

Type	Read-out file name	Definition format	
		All	Individual file
User memory	All file	ALL	————
	Program	PGM	<bb…b>>
	Point data	PNT	Pn
	Point comment	PCM	PCn
	Point name	PNM	PNn
	Parameter	PRM	/cccccccc/
	Shift definition	SFT	Sn
	Hand definition	HND	Hn
	Pallet definition	PLT	PLn
	General Ethernet port	GEP	GPn
	Input/output name	ION	iNMn(n)
	Area check output	ACO	ACn
Variable, constant	Variable	VAR	ab...by
	Array variable	ARY	ab...by(x)
	Constant	————	"cc...c"
Status	Program directory	DIR	<<bb…b>>
	Parameter directory	DPM	————
	Machine reference (sensor or stroke-end)	MRF	————
	Machine reference (mark)	ARP	————
	System configuration information	CFG	————
	Controller	CNT	————
	Robot	RBT	————
	Driver	DRV	————
	Option board	OPT	————
	Self check	SCK	————
	Alarm history	LOG	————
	Remaining memory size	MEM	————
Device	DI port	DI()	DIn()
	DO port	DO()	DOn()
	MO port	MO()	MOn()
	TO port	TO()	TOn()
	LO port	LO()	LOn()
	SI port	SI()	SIn()
	SO port	SO()	SOn()
	SIW port	SIW()	SIWn()
	SOW port	SOW()	SOWn()
Others	File end code	EOF	————

a: Alphabetic character b: Alphanumeric character or underscore (_) c: Alphanumeric character or symbol
i: I/O type n: Number x: Expression (Array argument) y: variable type

SAMPLE

```
Command: @READ PGM [cr/lf] ····· Reads out all programs.
         @READ P100 [cr/lf] ····· Reads out the point 100.
         @READ DINM2(0) [cr/lf] ··· Reads out the input/output
                                    name of DI2(0).
```

5.7 Data write processing

Command format

```
@WRITE write file[cr/lf]
```

Response format

```
READY[cr/lf] ····Input request display
OK [cr/lf] ·····After input is completed
```

NOTE

• For more information about files, refer to the earlier Chapter 10 "Data file description".

Values *Write file* ..... Designates a write file name.

Meaning Writes the data in the designated file.
Online commands that are input through the RS-232C port have the same meaning as the following command.
• SEND CMU TO <*write file*>
Commands via Ethernet have the same meaning as the following command.
• SEND ETH TO <*write file*>

MEMO

- At the DO, MO, TO, LO, SO, SOW ports, an entire port (DO(), MO(), etc.) cannot be designated as a WRITE file.
- Some separate files (DOn(), MOn(), etc.) cannot be designated as a WRITE file. For details, refer to Chapter 10 "Data file description".

Type	Write file name	Definition format	
		All	Separate file
User memory	All file	ALL	———
	Program	PGM	<bb...b>>
	Point data	PNT	Pn
	Point comment	PCM	PCn
	Point name	PNM	PNn
	Parameter	PRM	/cccccccc/
	Shift definition	SFT	Sn
	Hand definition	HND	Hn
	Pallet definition	PLT	PLn
	General Ethernet port	GEP	GPn
	Input/output name	ION	iNMn(n)
	Area check output	ACO	ACn
Variable, constant	Variable	VAR	ab...by
	Array variable	ARY	ab...by(x)
Device	DO port	———	DOn()
	MO port	———	MOn()
	TO port	———	TOn()
	LO port	———	LOn()
	SO port	———	SOn()
	SOW port	———	SOWn()

a: Alphabetic character b: Alphanumeric character or underscore (_) c: Alphanumeric character or symbol
i: I/O type n: Number x: Expression (Array argument) y: variable type

SAMPLE

```
Command: @WRITE PRM [cr/lf]·····Writes the label specified
                                 parameter.
         @WRITE P100 [cr/lf]·····Writes the point 100.
         @WRITE DINM2(0)[cr/lf]····Writes the input/output
                                 name of DI2(0).
```

6 Utility commands

6.1 Setting the sequence program execution flag

Command format

```
@SEQUENCE k[cr/lf]
```

Response format

```
OK[cr/lf]
```

Values k .....Execution flag / 0: disable, 1: enable, 3: enable (DO reset)

Meaning Sets the sequence program execution flag.

SAMPLE

```
Command:   @SEQUENCE 1[cr/lf]
Response:  OK[cr/lf]
```

6.2 Setting the date

Command format

```
@DATE yy/mm/dd[cr/lf]
```

Response format

```
OK[cr/lf]
```

Values yy/mm/dd.....Date to be set. (year, month, day)
yy.....Lower 2 digits of the year (00 to 99)
mm .....Month (01 to 12)
dd .....Day (01 to 31)

Meaning Sets a date in the controller.

NOTE

- To change only the year or month, the slash (/) following it can be omitted.

  Example:

  To set the year to 2016, enter 16(cr/lf).

  To set the month to June, enter /06(cr/lf).

MEMO

- The currently set values are used for the omitted items.
- If only [cr/lf] is transmitted, then the date remains unchanged.
- If an improbable date is entered, then "5.202: Data error" occurs.

SAMPLE 1

```
To change only the day,
 //15[cr/lf]·····Day is set to 15th.
```

SAMPLE 2

```
Command:   @DATE 16/01/14[cr/lf]
Response:  OK[cr/lf]
```

6.3 Setting the time

Command format

```
@TIME hh:mm:ss[cr/lf]
```

Response format

```
OK[cr/lf]
```

Values

hh:mm:ss ..... Current time
hh ..... hour (00 to 23)
mm ..... minute (00 to 59)
ss ..... second (00 to 59)

Meaning Sets the time of the controller.

MEMO

- The currently set values are used for the omitted items.
- If only [cr/lf] is transmitted, then the time remains unchanged.
- If an improbable time is entered, then "5.202: Data error" occurs.

SAMPLE 1

```
To change only the minute,
 :20:[cr/lf]·····Minute is set to 20.
```

SAMPLE 2

```
Command:  @TIME 10:21:35[cr/lf]
Response: OK[cr/lf]
```

7 Individual execution of robot language

Command format

```
@robot language[cr/lf]
```

Response format 1

```
OK[cr/lf] or NG=gg.bbb [cr/lf]
```

Response format 2

```
RUN[cr/lf] or NG=gg.bbb[cr/lf] ·····At process start
END[cr/lf] or NG=gg.bbb[cr/lf] ·····At process end
```

Values

OK, END..... Command ended correctly.
NG..... An error occurred.
RUN ..... Command starts correctly.
gg: Alarm group number ..... 0 to 99
bbb: Alarm classification number..... 0 to 999

Meaning Robot language commands can be executed.

- Only independently executable commands are executed.
- Command format depends on each command to be executed.

SAMPLE 1

```
Command:   @SET DO(20) [cr/lf]
Response:  OK[cr/lf]
```

SAMPLE 2

```
Command:   @MOVE P,P100,S=20[cr/lf]
Response:  RUN [cr/lf] ····· Process start
           END [cr/lf] ····· Process end
```

Command format

```
^C (=03H)
```

Response format

```
NG=1.8
```

Meaning Interrupts execution of the current command.

SAMPLE

```
Command:   @MOVE P,P100,S=20[cr/lf]
           ^C
Response:  NG=1.8[cr/lf]
```