Be sure to read the following precautions for your safety.

This section describes safety precautions to avoid danger to you or someone else, to avoid damage of your property, and to use this product safely.

Precautions before using this product

This product is designed to be incorporated into general industrial machinery, and is NOT developed to be used in devices such as aerospace machines, security equipment, or other safety devices where a failure or malfunction of this product may directly threaten human lives or health.

Even if you use this product in a general device, make sure that you establish a sufficient level of safety in your device by incorporating a protection function into your machine and guarantee your products based on safety tests on the whole set.

If you will use this product in devices like the above, please contact us. It should be noted that RORZE will not be responsible for any damage caused by using a product in such a device without the consent of RORZE.

WARNING

Ignoring the following warnings may cause a death or a serious injury.

◇ Use this product at places where no explosive or flammable stuff exist nearby and no water is splashed on the product. Otherwise it may cause a fire and/or an injury.
◇ Turn off the power before moving or wiring the product. Otherwise you may suffer injuries or electric shocks.
◇ Do not forcibly bend, pull, or nip lead wires. Otherwise they may cause an electric shock, fire, and/or failure.
◇ Do not use lead wires with their sheath damaged. Otherwise they may cause an electric shock, fire, and/or failure.
◇ Make sure that wires are correctly and securely connected at electrical terminals. Otherwise they may cause an electric shock, fire, and/or failure.
◇ Do not touch the internal parts of this product.
◇ Do not disassemble or modify this product.
◇ Do not wire or operate a product with wet hands. Otherwise it may cause electric shocks.
◇ Assign a qualified person to transport, install, connect, operate, maintenance, or check this product. Otherwise it may cause an electrical shock, a fire and/or an injury.

CAUTION

Ignoring the following cautions may result in personal injuries and/or property damages.

◇ Make sure that the delivered product is the one you ordered. Installing the wrong product may cause a fire and/or a failure.

Check the following items before turning on the power.

◇ The output voltage of the power supply is as described in the specifications.
◇ The voltage/current of the input/output terminals conforms to the ratings in the specifications.
◇ Input/output terminals are not incorrectly wired or accidentally short-circuited.
◇ When connecting with terminal block, use a screwdriver whose tip fits an adjustment slot. Tighten the screw in the torque of less than 3.5kgf・cm(0.35N・m)(proper torque is 2.5kgf・cm(0.25N・m)).
◇ The connectors are securely cramped.
◇ When you run a product for the first time, make sure that the operation can be stopped immediately under an emergency situation.

Ignoring the above cautions may cause a fire and/or a failure.

◇ Immediately turn off the power, if you hear an unusual noise. Otherwise it may cause a fire and/or an injury.
◇ Do not touch this product when it is in operation, as a malfunction may occur.
◇ Do not carry this product by holding its connectors or lead wires. When the product is accidentally dropped, it may cause a personal injury.
◇ Do not place this product in unstable positions. When the product is accidentally dropped, it may cause a personal injury.

Under some circumstances, ignoring the precaution described in the CAUTION section may also result in a death or a severe injury.

Follow the above precautions described in both the WARNING and the CAUTION section.
# Table of Contents

## Hardware

- Description ............................................................................................................... A-2
- Features .................................................................................................................. A-2
- Differences between RC-233 and RC-234 .............................................................. A-2
  1. Dimensions ......................................................................................................... A-3
  2. Specifications ...................................................................................................... A-4
  3. RC-234 Limitations ............................................................................................ A-5
  4. Part Names and Functions ................................................................................ A-8
    - 4-1 Part names .................................................................................................. A-8
    - 4-2 Communication port ................................................................................ A-9
    - 4-3 Input/output ports ................................................................................... A-9
    - 4-4 Motor control ports ................................................................................. A-9
    - 4-5 Substitution of General-purpose Input and output ports ....................... A-11
    - 4-6 Suitable Socket name ............................................................................. A-12
  5. Wiring of Current Loop Signal Line ................................................................. A-13
  6. Mode and Driver Wiring method ...................................................................... A-14
    - 6-1 Simultaneous 2-axis control .................................................................. A-14
    - 6-2 Operation mode ...................................................................................... A-18
  7. Stepping Motor Speed Adjustment ................................................................. A-21
    - 7-1 Speed calculation .................................................................................. A-21
    - 7-2 Setting for command “OH”, “OL”, “OC”, “OS”, “OX” ............................ A-22
    - 7-3 Relationship between accel./decel. time and speed when total pulse is few ................................................................. A-23
    - 7-4 Position data ......................................................................................... A-23
  8. Communication Check .................................................................................... A-25
    (Be sure to read this section before using the RC-234)
    - 8-1 Communication (RC-234 ⇔ PC) .......................................................... A-25
    - 8-2 Motor Motion Check ............................................................................. A-28

## Software

- 9. Command Explanation .................................................................................. B-2
    - 9-1 Sending format ...................................................................................... B-2
    - 9-2 Replying format .................................................................................... B-4
- 10. Command Reference .................................................................................. B-5
- 11. User Program Explanation .......................................................................... B-162
    - 11-1 Command for user program ............................................................... B-162
    - 11-2 Relation between EEPROM and RAM ............................................. B-163
    - 11-3 Creating a User Program (Description of the Format) ....................... B-164
    - 11-4 Example of User Program ................................................................. B-170
- 12. User Program Command Reference ......................................................... B-172

## Appendix

- 13. Program ......................................................................................................... C-2
    - 13-1 Notes for programming ....................................................................... C-2
    - 13-2 Sample Program using Quick Basic .................................................. C-3
      ① Commands for each mode and Sample Program .................................. C-3
      ② Sample Program for Special Function ................................................ C-11
- 14. Link Master RC-002 .................................................................................. C-16
- 15. Control Command List ................................................................................ C-18
- 16. Command Default Setting .......................................................................... C-34

Appendix. Contents by Command Function ................................................................ C-36
<Hardware>

1. Dimensions
2. Specifications
3. RC-234 Limitations
4. Part Names and Functions
5. Wiring of Current Loop Signal Line
6. Mode and Driver Wiring method
7. Stepping Motor Speed Adjustment
8. Communication Check
INSTRUCTION MANUAL FOR RC-234

Description

The RC-234 is a controller which can control stepping motor and servo motor by pulse input. This can control 2 motors simultaneously, and circular and linear interpolation controls are available in the XY plane.

Features

●  S-curve acceleration provides smooth moves without a shock or damping.
●  Asymmetric S-curve accel./decel. control is available.
●  Closed loop control when used with an encoder for accurate positioning.
●  Stand-alone operation and control from PLC are available by downloading the user program.
●  Baud rate, acceleration rate and S-curve rate can be changed arbitrarily.
●  Pulse output frequency up to 1Mpps permits to be used with a high resolution microstep driver.

Differences between RC-233 and RC-234

We added simultaneous 2-axis control function to the RC-234 while keeping compatibility with the RC-233. When you replace the RC-233 with the RC-234, please note the following.

① Compatible mode
   To switch an alternate 2-axis control to a simultaneous 2-axis control, use the command “EM”.

② EEPROM
   It takes more time for the RC-234 to read the EEPROM compared with the RC-233. In case of storing the user program in the EEPROM, it takes about 2 seconds at max. to start after turning on the power.
   Also, it takes about 1 second at max. to read EEPROM by the command “IL” or “AL”.

③ Additional commands
   "9ALL", "9D", "AL*", "CY", "DCT", "DL*", "EECHECK", "EM", "GSSE", "M □",
   "QB", "QBR", "SL",

④ Modified commands
   "ES", "N", "GW", "GWT", "GR", "GRT", "OH", "OL", "OS",
   (There is no problem regarding compatibility because these commands are modified expansion functions.)
1. Dimensions
2. Specifications

<table>
<thead>
<tr>
<th>Specification</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply voltage</td>
<td>18 to 40VDC (Maximum rated voltage: 40VDC)</td>
</tr>
<tr>
<td>Supply current</td>
<td>Less than 100 mA at 24 VDC</td>
</tr>
<tr>
<td>Pulse output frequency and Encoder Response</td>
<td>0.1 to 1Mpps</td>
</tr>
<tr>
<td>Counter set value</td>
<td>0 to 16,777,215 or -8,388,608 to +8,388,607 (Data memory for the command “2”, “A”, “N”, etc.)</td>
</tr>
<tr>
<td>Data memory position</td>
<td>0 to 16,777,215 or -8,388,608 to +8,388,607 (Sets by the command “EP”)</td>
</tr>
<tr>
<td>Number of Profiles</td>
<td>Up to 1,000 points (each axis)</td>
</tr>
<tr>
<td>No. of control motors</td>
<td>2 (Simultaneous control is available)</td>
</tr>
<tr>
<td>Communication port</td>
<td>In: 3, Out: 3, RXD END: 1</td>
</tr>
<tr>
<td>Motor control ports</td>
<td>Motor 1, 2: 10 each (8 each for closed loop servo)</td>
</tr>
<tr>
<td>User Input/Output ports</td>
<td>Input Port: 8, Output Port: 8</td>
</tr>
<tr>
<td>Interface</td>
<td>Motor 1, 2: 10 each (8 each for closed loop servo)</td>
</tr>
<tr>
<td>Suitable Encoders</td>
<td>Open collector output type</td>
</tr>
<tr>
<td>Communication</td>
<td>Current loop method of RS-232C (Use Link Master RC-002)</td>
</tr>
<tr>
<td>Baud rate</td>
<td>38400, 19200, 9600, 2400, 1200, 300 (bps)</td>
</tr>
<tr>
<td>Data bit length</td>
<td>8 bits</td>
</tr>
<tr>
<td>Parity</td>
<td>none</td>
</tr>
<tr>
<td>Stop bit</td>
<td>1 bit</td>
</tr>
<tr>
<td>Memory capacity</td>
<td>8,000 bytes (approx. 1,500 lines of command level code)</td>
</tr>
<tr>
<td>Number of profiles</td>
<td>1,000 points (each axis)</td>
</tr>
<tr>
<td>Memory of command data in regard to mode setting</td>
<td>(Refer to command “DW”)</td>
</tr>
<tr>
<td>Number of time to rewrite EEPROM data</td>
<td>100,000 times Min. / 1 DATA</td>
</tr>
<tr>
<td>Number of year to maintain EEPROM data</td>
<td>More than 10 years</td>
</tr>
<tr>
<td>Weight</td>
<td>approx. 250g</td>
</tr>
<tr>
<td>Outside dimensions</td>
<td>27.5(H) x 105(W) x 56(D) mm</td>
</tr>
</tbody>
</table>

- Control of plural generate masters and I/O masters
  RC-002 can control up to 16 controllers (20 controllers by using option)
  (Generate master RC-234, I/O master RC-204A and RC-207A)
  Each I/O master and generate master is controlled by setting body No. (yellow rotary switch)

- Stepping Motor Drivers to be used
  RORZE Drivers (RD-0 * * series)
  RD-022(N), RD-023(N), RD-021M8, RD-023MS, RD-023MSG, RD-026MSA, RD-02CMSH, RD-A051, RD-053(N), RD-053MS
  Other Step Motor Drivers
  Other type translators with 2 line pulse inputs or pulse and direction input type
  Other Servo Motor Drivers
  Other type translators with 2 line pulse inputs or pulse and direction input type
### 3. RC-234 Limitations

#### I. Communication with Host or PC

- The RC-234, Generate Master is an intelligent controller with built-in pulse generator. It has original commands and the following controls are available by communicating with PC via RC-002 Link Master (RS-232 current adapter).
  - Encoder feed-back stepping motor driver control
  - Micro stepping motor driver control
  - Pulse input servo motor driver control

- To communicate with PC, it is necessary to connect RC-002, which converts RS-232C signal to current loop, in between PC and the controller.

- The baud rate of the RC-234 is programmable, but default setting is 9,600 bps so that please set baud rate of PC 9,600 bps first. It takes approx. 1msec to send 1 letter. (Refer to the Command “ES”)

- Communication distance
  Varies depending on the number of controllers, communication speed, etc.

(Ex.) Measured communication distance using shielded 3-pin copper wire with 0.3sq.

#### Table: Communication Distance

<table>
<thead>
<tr>
<th>Distance of current loop communication line (m)</th>
<th>300bps, 1200bps, 2400bps</th>
<th>9600bps</th>
<th>19200bps</th>
</tr>
</thead>
<tbody>
<tr>
<td>1m</td>
<td>RC-002 Number of RC-234s per loop</td>
<td>RC-002 Number of RC-234s per loop</td>
<td>RC-002 Number of RC-234s per loop</td>
</tr>
<tr>
<td>2m</td>
<td><img src="#" alt="Table Format" /></td>
<td><img src="#" alt="Table Format" /></td>
<td><img src="#" alt="Table Format" /></td>
</tr>
<tr>
<td>3m</td>
<td><img src="#" alt="Table Format" /></td>
<td><img src="#" alt="Table Format" /></td>
<td><img src="#" alt="Table Format" /></td>
</tr>
<tr>
<td>4m</td>
<td><img src="#" alt="Table Format" /></td>
<td><img src="#" alt="Table Format" /></td>
<td><img src="#" alt="Table Format" /></td>
</tr>
<tr>
<td>5m</td>
<td><img src="#" alt="Table Format" /></td>
<td><img src="#" alt="Table Format" /></td>
<td><img src="#" alt="Table Format" /></td>
</tr>
<tr>
<td>6m</td>
<td><img src="#" alt="Table Format" /></td>
<td><img src="#" alt="Table Format" /></td>
<td><img src="#" alt="Table Format" /></td>
</tr>
<tr>
<td>7m</td>
<td><img src="#" alt="Table Format" /></td>
<td><img src="#" alt="Table Format" /></td>
<td><img src="#" alt="Table Format" /></td>
</tr>
<tr>
<td>8m</td>
<td><img src="#" alt="Table Format" /></td>
<td><img src="#" alt="Table Format" /></td>
<td><img src="#" alt="Table Format" /></td>
</tr>
<tr>
<td>9m</td>
<td><img src="#" alt="Table Format" /></td>
<td><img src="#" alt="Table Format" /></td>
<td><img src="#" alt="Table Format" /></td>
</tr>
<tr>
<td>10m</td>
<td><img src="#" alt="Table Format" /></td>
<td><img src="#" alt="Table Format" /></td>
<td><img src="#" alt="Table Format" /></td>
</tr>
</tbody>
</table>
### 3. RC-234 Limitation

#### Table:  Number of RC-234s per loop

<table>
<thead>
<tr>
<th>Distance of current loop communication line</th>
<th>RC-002</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1m</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>△</td>
</tr>
<tr>
<td>200m</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
</tr>
<tr>
<td>400m</td>
<td>○</td>
<td>○</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
</tr>
<tr>
<td>600m</td>
<td>○</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
</tr>
<tr>
<td>800m</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
</tr>
<tr>
<td>1000m</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
<td>⌐</td>
</tr>
</tbody>
</table>

[Note] The communication distance will become shorter because of
- the large resistance of an electric wire.
- the effects of electric noise.
- the use of LED to shorten the end of current loop.

#### II. No. of Drivers and Stepping Motors

- Step Motor Drivers to be used
  - RORZE Drivers: RD-0**series
  - Other Step Motor Drivers: Other type translators with 2 line pulse inputs or pulse and direction input type
  - Other Servo Motor Drivers: Other type translators with 2 line pulse inputs or pulse and direction input type

- When using a servo driver (servo motor), the RC-234 is used in mode 2. (As for the wiring for servo driver and mode, refer to Chapter 6 “Mode and Driver Wiring Method”.)

- Wiring stepping motor with encoder can control stepping motor while controlling position by encoder. Use the RC-234 in mode 1. (As for the mode, refer to Chapter 6 “Mode and Driver Wiring Method”.)

#### III. No. of Controllers in the loop

- PC or Host can communicate with 15 Generate Master by setting the body number for each controller using the yellow rotary switch. Position ‘F’ cannot be used as it is used for the test - receiving characters from PC and sending it back to PC. Therefore, 15 controllers with body number 0 to E can be connected.

- The LinkMaster has a limitation of 20 controllers, virtually it is possible to connect 10 on one loop and 10 on the other loop. To connect the second 10 controllers remove the LED from RC-002.

- To control plural Generate masters and I/O masters, set the yellow rotary switch to the different number for each controller.
IV. Position Control limitation

- Position control in simultaneous 2-axis control (command “EM”=1) is fixed from –8,388,608 to +8,388,607 pulses and the command “EP” is invalid.
  (Refer to the command “EP”, “EM”)

- In case of mode 1, control of the stepping motor using encoder, encoder controls the position pulse data.
  As for the rotation speed and mechanical home (ORG) search, encoder’s pulse number is invalid.
  (Refer to Chapter 7 “Stepping motor Drive Method”.)

V. Limit Sensor

- The RC-234 can set the input logic for limits, ORG and EZ.  (Refer to the command “EA”)
  Please select active low (Normally open logic) or active high (Normally close logic) according to
  the sensor input logic.

- Please turn home (ORG) sensor ON 6 pulses or more at the home position.  Otherwise the
  RC-234 doesn't recognize the activation of the home (ORG) sensor.

- Avoid home (ORG) sensor and CW limit being ON at same time.  The home search operation will
  be erratic difficult if the home (ORG) sensor and CCW sensor are ON at the same time or close to
  each other.

VI. Stall Detection  (For more details, refer to the command “Q”.)

1) Please prepare stall sensor and stall slip by yourself.
2) Position of stall sensor
3) Period of stall sensor

VII. EEPROM Limitation

- The EEPROM has a limitation for the number to store the data.  Therefore, exceeding this
  limitation, there is a possibility not to store the data correctly.  The EEPROM will store data
  correctly as long as the data transfer has not exceeded 100,000 times.  The controller can be
  relied for data correctness up to 10 years.

- The EEPROM is blank when shipped.  If the commands “IW”, “AW”, “DW” to store the data to
  EEPROM are executed in wrong before understanding and stored data to EEPROM, there is a
  danger of malfunction.
  (To erase any existing data use the command “EE///”, refer to the command manual for more
  details.)

VIII. Alternate/simultaneous 2-axis control mode

- The default setting is an alternate 2-axis control mode.  To perform simultaneous 2-axis control,
  please use the command “EM”.
4. Part Names and Functions

4-1 Part names

<table>
<thead>
<tr>
<th>No.</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal name</td>
<td>TXD IN</td>
<td>COM OUT</td>
<td>RXD OUT</td>
<td>N.C</td>
<td>N.C.</td>
</tr>
<tr>
<td>No.</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>Signal name</td>
<td>COM IN</td>
<td>RXD IN</td>
<td>TXD OUT</td>
<td>RXD END</td>
<td>N.C.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Block Name</th>
<th>MOTOR 2</th>
<th>SERVO &amp; EMS</th>
<th>MOTOR 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>No.</td>
<td>25</td>
<td>23</td>
<td>21</td>
</tr>
<tr>
<td>Signal name</td>
<td>EB</td>
<td>CWLS</td>
<td>CCWLS</td>
</tr>
<tr>
<td>No.</td>
<td>26</td>
<td>24</td>
<td>22</td>
</tr>
<tr>
<td>Signal name</td>
<td>EZ</td>
<td>EA</td>
<td>ORG</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWR (Green)</td>
<td>This will light when power supply is turned on.</td>
</tr>
<tr>
<td>RUN (Green)</td>
<td>This will flash at a regular interval in normal operation.</td>
</tr>
<tr>
<td>TXD (Green)</td>
<td>This will flash in accordance with transmission data.</td>
</tr>
</tbody>
</table>
| ERR (Red)          | This will light when an error such as Limit error, Stall error, Emergency stop and Command error has occurred. This will turn off a light when querying error status by the Command " " (NULL) and "9ALL".
4-2 Communication port

Please connect Link Master RC-002 to RS-232C. As for the end RC-234, please connect RXD OUT to RXD END or COM. (Connectors for communication 1, 4, 7 pin are shorted inside the RC-234.) Please refer to wiring diagram. (Chapter 5 “Wiring of Current Loop Signal Line”)

4-3 Input/output ports

- Input port

When input port voltage keeps Low level more than 10 μsec, the RC-234 recognizes that the signal is ON.

Note) When you use D0 and D1 as an interrupt pin, please set High level to 3.5V or more.

- Output port

Open Collector (Darlington-Transistor)
Voltage: Less than 50V
Current: Less than 200mA (per contact)
Less than 800mA in total of 8 contacts
V_{ce(sat)}: Less than 1.1V (I_{c}: 200mA)

There is COM terminal to connect interval load, but this terminal is settled every 8 output terminals (Refer to the right wiring dimension.). Note that COM is a common terminal. (Right dimension shows 2 circuits out of 8 circuits.)

Note: Be careful not to apply minus voltage to the output terminal. There is danger of malfunction or failure.

4-4 Motor Control ports

<table>
<thead>
<tr>
<th>Output for motor control</th>
<th>Input for motor control</th>
</tr>
</thead>
</table>

\[ R_1 = 470 \Omega \ (EA, EB) \]
\[ R_2 = 2.2k \Omega \ (Others) \]
\[ R_{12} = 680k \Omega \ (Others) \]
\[ C_1 = \text{None} \ (EA, EB) \]
\[ C_1 = 470pF \ (Others) \]

\[ V_{ce} = 50V \]
\[ I_C = 100mA \]

Low Level: Less than 1.5V (EA, EB: Less than 1.0V)
High Level: More than 3.5V
### 4. Part Names and Functions

<table>
<thead>
<tr>
<th>Terminals</th>
<th>I/O</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| +5V       | O   | 5V logic supply  
|           |     | Load current 150mA MAX. (The current combined both terminals of motor 1 and 2.) |
| CW/P      | O   | 2 pulse method (2CK) : CW Pulse Output  
|           |     | Pulse and DIR method (1CK) : Pulse Output |
| CCW/D     | O   | 2 pulse method (2CK) : CCW Pulse Output  
|           |     | Pulse and DIR method (1CK) : DIR Output |
| CCWLS     | I   | CCW limit sensor input  
|           |     | Turning on CCWLS, pulse output will be stopped and becomes limit error. |
| ORG       | I   | ORG limit sensor input |
| CWLS      | I   | CW limit sensor input  
|           |     | Turning on CWLS, pulse output will be stopped and becomes limit error. |
| EA        | I   | Encoder A-phase signal input (in mode 1) |
| EB        | I   | Encoder B-phase signal input (in mode 1) |
| EZ        | I   | Encoder Z-phase signal input (in mode 1)  
|           |     | Stall detection sensor input in mode 0, 2  
|           |     | (Can act without connecting sensor) |
| EMS       | I   | Emergency stop input |
| INP       | I   | In position input when using servo motor driver in mode 2.  
|           |     | Can use as a general-purpose input terminal in mode 0, 1. |
| CLR       | O   | Deviation counter clear (reset) output for servo motor driver.  
|           |     | (in mode 2) Refer to the command “DS”.  
|           |     | Can use as a general-purpose output terminal in mode 0, 1. |
| GND       |     | Ground terminal  
|           |     | Common GND with GND of supply voltage terminal |

**Important Notice**

- To select each terminal’s input logic(either Normally open logic or Normally closed logic) for CCWLS, ORG, CWLS, EZ, EMS, INP, use the command “EA”.

- When you set the input logic of limit sensor incorrectly, even if you send the command to rotate the motor, the motor doesn’t rotate.

For more details of sensor input logic, please refer to the command “EA”. 
4-5 Substitution of General-purpose input and output ports

The RC-234 has general-purpose input and output ports, 8 input ports and 8 output ports. If they are not enough, you can use terminals for motor control as general-purpose I/O ports with the limitation below.

① If you use only one motor, you can use the terminals in the below table as general input ports.

<table>
<thead>
<tr>
<th>PORT No.</th>
<th>Signal name</th>
<th>Usage and its limitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>INPORT</td>
<td>CCWLS</td>
<td>In case that you use these sensor input terminals as general input ports, query the input status (ON, OFF) using the command “CLM2”.</td>
</tr>
<tr>
<td></td>
<td>ORG</td>
<td>During the motor 1 is rotating, you can't query the input status.</td>
</tr>
<tr>
<td></td>
<td>CWLS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EZ</td>
<td></td>
</tr>
</tbody>
</table>

② Only in case of mode 0 and 1, you can use the below terminals as input/output ports.

<table>
<thead>
<tr>
<th>PORT No.</th>
<th>Signal name</th>
<th>Usage and its limitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>INPORT</td>
<td>INP1</td>
<td>When you use INP (In position) terminal as a general input, can confirm input status (ON, OFF) using the command “CA” or the command “C8” (INP1), “C9” (INP2). At operation mode 2, because INP becomes a terminal for the in position input for the servo driver, you can’t use them as general-purpose input ports.</td>
</tr>
<tr>
<td></td>
<td>INP2</td>
<td></td>
</tr>
<tr>
<td>OUTPORT</td>
<td>CLR1</td>
<td>When you use CLR (deviation counter clear) as a general-purpose port, set the output (ON, OFF) using the command “DA” or the command “D8” (CLR1), “D9” (CLR2) and query output status using the command “COA” or the command “CO8” (CLR1), “CO9” (CLR2). In mode 2, because CLR terminals have deviation counter clear function, you can’t use them as general-purpose output ports.</td>
</tr>
<tr>
<td></td>
<td>CLR2</td>
<td></td>
</tr>
</tbody>
</table>

③ In case that you use one motor in mode 0 and 1, you can use both ports in ① and ② as general I/O ports.
4-6 Suitable Socket name

Sockets for I/O ports will not be provided. Flat cables with socket are available.

![Diagram](image)

A…(10 pin) Communication port
B…(26 pin) Motor control ports
C…(20 pin) I/O ports
D…Connector lock lever
   (Position stopper)
   Please use it according to your needs.

**For Flat cable**

<table>
<thead>
<tr>
<th>Sockets</th>
<th>Strain relieves</th>
</tr>
</thead>
<tbody>
<tr>
<td>A XG4M-1030 (for 10P)</td>
<td>A XG4T-1004 (for 10P)</td>
</tr>
<tr>
<td>B -2630 (for 26P)</td>
<td>B -2604 (for 26P)</td>
</tr>
<tr>
<td>C -2030 (for 20P)</td>
<td>C -2004 (for 20P)</td>
</tr>
</tbody>
</table>

**For Discrete wire**

<table>
<thead>
<tr>
<th>Sockets</th>
<th>Semi covers</th>
</tr>
</thead>
<tbody>
<tr>
<td>(for AWG#28 to 26)</td>
<td></td>
</tr>
<tr>
<td>A XG5M-1035 (for 10P)</td>
<td>A XG5S-0501 (for 10P)</td>
</tr>
<tr>
<td>B -2635 (for 26P)</td>
<td>B -1301 (for 26P)</td>
</tr>
<tr>
<td>C -2035 (for 20P)</td>
<td>C -1001 (for 20P)</td>
</tr>
</tbody>
</table>

| (for AWG#24)        |
| A XG5M-1032 (for 10P) |             |
| B -2632 (for 26P)   |             |
| C -2032 (for 20P)   |             |

**Connector lock lever**
D XG4Z-0002

**Tool**
XY2B-7006

* Manufactured by OMRON.

**Option (Flat cable with socket)**

<table>
<thead>
<tr>
<th>Communication port (10 pin)</th>
<th>I/O ports (20 pin)</th>
<th>Motor control ports (26 pin)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RCC-10P50L</td>
<td>RCC-20P50L</td>
<td>RCC-26P50L</td>
</tr>
<tr>
<td>RCC-10P100L</td>
<td>RCC-20P100L</td>
<td>RCC-26P100L</td>
</tr>
<tr>
<td>RCC-10P200L</td>
<td>RCC-20P200L</td>
<td>RCC-26P200L</td>
</tr>
<tr>
<td>RCC-10P300L</td>
<td>RCC-20P300L</td>
<td>RCC-26P300L</td>
</tr>
</tbody>
</table>

**Construction of Model**
RCC -10P 50L

- Length 50cm
- 10 pin
- Flat cable with socket

A-12
5. Wiring of Current Loop Signal Line

[Note]

I. When you connect plural Generate Masters and I/O Masters,
   a. The body number of each Generate Master and I/O Master should be identical.
   b. Please short 6th pin and 7th pin of end Generate Master or I/O Master that is located
      farthest from RC-002.

II. When you use only one Generate Master or I/O Master, please short 6th pin(RXD OUT) and 7th pin
    (RXD END) just the same way as I . b.

III. Generate master and I/O master are controlled using RS-232C method (signal current: 20mA) from
     PC. You should use Link Master RC-002 to convert usual RS-232C to current loop type.
     • Feature of Current Loop type
       1. Improves the noise-resistant.
       2. Can control plural generate masters and I/O masters by one PC.

IV. When the total number of Generate Masters and I/O Masters to connect with RC-002 exceeds over
    10, please remove the LED from RC-002, and use both of the two output terminals.
    You can connect up to 10 RC-234s or a combination of Generate Masters and I/O Masters to one
    of output terminals of RC-002.
6. Mode and Driver Wiring Method

The default setting of the RC-234 is an alternate 2-axis control mode. To perform simultaneous 2-axis control, please use the command “EM”.

1. The command “EM” = 0 (Alternate 2-axis control mode)
   The operation mode will become an alternate 2-axis control mode, that is to say, an alternate drive.

2. The command “EM” = 1 (Simultaneous 2-axis control mode)
   Operation mode (0,1,2) can set motor 1 and motor 2 separately.

The RC-234 has three modes.
- Mode 0 … Control of stepping motor driver.
- Mode 1 … Control of stepping motor using encoder.
- Mode 2 … Control of pulse input servo motor driver.

Use the command “E” for operation mode setting.
(Refer to Chapter 10 “Command Reference”.)

6-1 Simultaneous 2-axis control

Setting the command “EM” to 1 provides a simultaneous start of motor 1 and motor 2 in home search, positioning, etc. and a simultaneous control that one motor can move while the other motor is moving.

Commands
- When “EM” is 0, the following commands are common settings for motor 1 and motor 2, but when “EM” is 1, they are different settings for motor1 and motor2 separately.

The change of a motor
- When the command “EM” is 1, the change of a motor using the command “F” is available even under operation.

Position control
- When “EM” is 1, position control is limited to signed position control (−8,388,608 to +8,388,607 pulses) and the command “EP” is invalid.

(1) Independent axis operation

Performs operation commands after selecting the motor number by the command “F”.
Parameters of motor 1 and motor 2 are set separately.

Ex.)

\[
\begin{align*}
\text{S 1 F 1 D} & \quad \text{: Selects motor 1.} \\
\text{S 1 O X 1 0 0 D} & \quad \text{: Sets a frequency magnification to 100.} \\
\text{S 1 B 0 0 1 D} & \quad \text{: Moves to the position of position number 001 set by the command “A”.} \\
\text{S 1 F 2 D} & \quad \text{: Selects motor 2.} \\
\text{S 1 O X 2 0 0 D} & \quad \text{: Sets a frequency magnification to 200.} \\
\text{S 1 B 0 0 2 D} & \quad \text{: Moves to the position of position number 002 set by the command “A”.}
\end{align*}
\]
(2) Simultaneous start

Selecting motor no. 3 by motor control command provides you a simultaneous start of motor 1 and motor 2. You can select motor no. 3 in the following commands.

Ex.) Home search

\[\text{S1} \text{0 H} \text{3} \]  : Performs a high speed home search with a simultaneous start.

Ex.) Move to the position of the same position number

\[\text{S1} \text{A 0 1} \text{1 0 0} \]  : Sets the position to position number 001.
\[\text{S1} \text{B 0 3} \]  : Moves both axes to the absolute position of position number 001 with a simultaneous start.

To move motor 1 and motor 2 to a separate position at different speed, please use the command “MB”.

Ex.) Moves two motors to a separate position

\[\text{S1} \text{MA 1 0} \text{1 0 0} \text{2 0 0} \]  : Sets the position to position number 010.
\[\text{S1} \text{MB 1 0} \]  : Moves to the absolute position of position number 010 with a simultaneous start.

(3) Interpolation control

The RC-234 provides you a linear interpolation, circular interpolation and continuous interpolation operation besides an independent axis operation. Operation parameters are set for one axis. This axis called an interpolation control axis and motor1 is an interpolation control axis.

① Linear interpolation

The locus from the current position to the endpoint is performed by the linear interpolation. (See the diagram below.)
The accuracy of position in operation for specified linear curve in linear-interpolation is ±0.5 pulses within the range of whole interpolation.

Ex.) Linear interpolation

\[\text{S1} \text{F 1} \]  : Selects an interpolation control axis (motor 1).
\[\text{S1} \text{OX 3 0} \]  : Sets a parameter to the interpolation control axis.
\[\text{S1} \text{MA 2 2} \text{0 2 0 0} \]  : Sets the endpoint to position number 220.
\[\text{S1} \text{MS 2 2} \]  : Performs linear interpolation to the endpoint.

```plaintext
X

<table>
<thead>
<tr>
<th>0</th>
<th>1000</th>
<th>2000</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1000</td>
<td>2000</td>
</tr>
</tbody>
</table>

Y

<table>
<thead>
<tr>
<th>0</th>
<th>1000</th>
<th>2000</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1000</td>
<td>2000</td>
</tr>
</tbody>
</table>
```
② Circular interpolation

The RC-234 performs CW(CCW) circular interpolation between motor 1 and motor 2. The accuracy of position for the specified circular interpolation curve is ±0.5 pulses within the range of whole interpolation.

**Endpoint, Center specification**
Performs circular interpolation after specifying the endpoint and the center for the current position. (See the right diagram.)

**Ex.) CW circular interpolation**

\[
\begin{align*}
\text{S1 MA210 2000 1000} & : \text{Sets the endpoint to position number 210.} \\
\text{S1 MA110 1000 1000} & : \text{Sets the center to position number 110.} \\
\text{S1 MC1 220 110} & : \text{Performs CW circular interpolation move.}
\end{align*}
\]

**Endpoint, Intermediate point**
Performs circular interpolation after specifying the endpoint and the intermediate point for the current position. (See the right diagram.)

**Ex.) CW circular interpolation**

\[
\begin{align*}
\text{S1 MA210 2000 1000} & : \text{Sets the endpoint to position number 210.} \\
\text{S1 MA120 1000 2000} & : \text{Sets the intermediate point to position number 120.} \\
\text{S1 MC3 220 110} & : \text{Performs circular interpolation move from the current point to the endpoint via intermediate point.}
\end{align*}
\]

③ Continuous interpolation

The RC-234 performs linear interpolation and circular interpolation continuously.

**Ex.) Continuous interpolation**

**Position settings**

\[
\begin{align*}
\text{S1 MA110 1000 0} & : \text{Sets Position1} \\
\text{S1 MA100 1000} & : \text{Sets Position2} \\
\text{S1 MA310 3000 1000} & : \text{Sets Position3} \\
\text{S1 MA420 4000 2000} & : \text{Sets Position4} \\
\text{S1 MA330 3000 3000} & : \text{Sets Position5} \\
\text{S1 MA130 1000 3000} & : \text{Sets Position6}
\end{align*}
\]

**Interpolation path settings**

\[
\begin{align*}
\text{S1 MP0 1 C1 110 100} & : \text{Sets a path from the current position to 1 (C1: CW circular)} \\
\text{S1 MP0 2 S310} & : \text{Sets a path from 1 to 3 (S: linear interpolation)} \\
\text{S1 MP0 3 C3 330 420} & : \text{Sets a path from 3 to 5 (C3: Circular interpolation)} \\
\text{S1 MP0 4 S130} & : \text{Sets a path from 5 to 6 (S: linear interpolation)} \\
\text{S1 MG 0} & : \text{Performs continuous interpolation.}
\end{align*}
\]
6. Mode and Driver Wiring Method

[Note]

- **Endpoint lead-in move**

If the endpoint is not on the circumference, a circular interpolation move is completed when one axis reaches the endpoint in the endpoint quadrant. ("a" point in the diagram below)

The offset move from the endpoint of circular interpolation by linear interpolation to the endpoint is called "The endpoint lead-in move". The RC-234 performs the endpoint lead-in move automatically after completion of circular interpolation.

Even when the position on the circumference becomes decimal, a circular interpolation move is available by specifying the approximate integer position.

※ The right diagram shows an example that the endpoint is shifted from the circumference on purpose. Please note that you can't move the endpoint to any position arbitrarily with the automatic endpoint lead-in move after the circular interpolation move.

Also, when you set the endpoint of circular interpolation in the gray part, the motor will not stop. (Permanent rotation)

- **Ratio of pulse output**

The ratio of pulse outputs of motor 1 and motor 2 in linear/circular interpolation mode is fixed at 1:1. (The controller can’t change the ratio.)

Please use the motor which operates using the ratio X:Y=1:1 in the interpolation drive for the X-Y table, etc.

In mode 1, the position pulse number is set by not RC-234 but encoder. Please note that the ratio of pulse varies depending on a resolution or counting of encoder. (See the command "PA".)

- **Acceleration/deceleration in continuous interpolation move**

The acceleration in continuous interpolation move should be set to finish by the first intermediate point. (Point ① in the diagram "③ continuous interpolation" in the previous page)

Even if a motor is still accelerating at the first intermediate point, the point will be High speed.

Also, the starting point of deceleration is after the last intermediate point. (Point ⑤ in the diagram "③ continuous interpolation" in the previous page)

If the distance between the last intermediate point and the end point is short, a motor will stop without decelerating to the Low speed.

To avoid the above matter, please shorten the acceleration/deceleration time or extend the distance between the first and the last area. (Current position to point ①, point ⑤ to end point)
6-2 Operation mode (Command “E”)

(1) Mode 0 (Control of stepping motor driver)

- To select the type of sensor (active high or active low), please use the command “EA”.
- To connect two drivers to one RC-234, please connect the second driver to MOTOR 2 which is a terminal for motor control of RC-234.
(2) Mode 1 (Control of stepping motor using encoder)

- To select the type of sensor switch and encoder (active high or active low), please use the command "EA".
- To connect two drivers/encoders to one RC-234, please connect the second driver to MOTOR 2 which is a terminal for motor control of RC-234.
(Refer to Chapter 4 “Part Names and Functions”)

```plaintext
Motor1
Stepping Motor
Driver RD-0
Encoder Input
Encoder RD-0
Generate Master RC-234
LinkMaster RC-002
EMS
I/O Ports
Motor2
RC-234
```

- Encoder A phase → EA(8th pin)
- Encoder B phase → EB(9th pin)
- Encoder Z phase → EZ(10th pin)

- Encoder A phase → EA(24th pin)
- Encoder B phase → EB(25th pin)
- Encoder Z phase → EZ(26th pin)
(3) Mode 2 (Control of pulse input servo motor driver)

- Please set the EMS logic to the normally closed logic (active high). (Use the command “EA41B”)
- The alarm can be confirmed by querying bit 2 (Emergency Stop) by the command “9”.

However, in case of using two servo drivers, you can’t detect which servo driver turns ON Alarm.
7. Stepping Motor Speed Adjustment

When driving stepping motors, speed control method is trapezoidal (as the right diagrams) generally, but the RC-234 can select two kinds of driving method, trapezoidal and S-curve acceleration. (Refer to the below diagrams)

To control speed, you should decide the followings.

1. Total pulse
2. Acceleration rate (pps/sec) in acceleration/deceleration
3. Low speed (pps)
4. High speed (pps)
5. S-curve data (S-curve data=0: Trapezoidal accel./decel., except 0, set ts in the below diagram.)

The above settlement is set by the following RC-234 commands.

1. Total pulse Command “2”, “A”
2. Low speed Command “OL”
3. Acceleration rate Command “OS” or “OSA”(in acceleration), “OSB”(in deceleration)
4. High speed Command “OH”
5. S-curve data Command “OC”

In addition to the above, it is necessary to set the command “OX” for internal speed data of the RC-234.

7-1 Speed calculation

\[
\text{High speed } (f_H) = \frac{OH}{OX} \times \frac{300}{(pps)}
\]

\[
\text{Low speed } (f_L) = \frac{OL}{OX} \times \frac{300}{(pps)}
\]

\[
\text{Accel. rate } (\alpha) = \frac{1,474,560,000}{OX \times OS} \text{ (pps/sec)}
\]

\[
\text{Accel. time } (t_{acc}) = \frac{|OH-OL|}{24,576 \times (200-OC)} \times OS \text{ (sec)}
\]

\[
\text{Accel. time is the time to take to decrement speed from High to Low or to increment speed from Low to High.}
\]

\[
\text{S-Curve rate } (OC) = \frac{2 \times ts}{t_{acc}} \times 100 \%}
\]
7. Stepping Motor Speed Adjustment

{ Specification of S-Curve acceleration control }

S-curve acceleration/deceleration with small acceleration change rate allows you,

① Smooth acceleration/deceleration reduces vibration, eliminates the need of damping and makes your mechanical system last longer.

② Higher frequencies (pps) can be achieved because S-Curve needs less acceleration torque.

③ Registration time is reduced with servo motor control.

- Multiplication factor
  Relation between multiplication factor and the command “OX” is as follows:

\[
\text{Multiplication factor} = \frac{300}{\text{OX}}
\]

Low speed pps (fL) = Low speed data (OL) × Multiplication factor
High speed pps (fH) = High speed data (OH) × Multiplication factor

<Relation of OX data, Pulse output frequency and Multiplication factor>

<table>
<thead>
<tr>
<th>Setting value of OX data</th>
<th>Real OX value</th>
<th>Pulse output frequency</th>
<th>Multiplication factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>3,000</td>
<td>3,000</td>
<td>0.1 to 3,276</td>
<td>1/10</td>
</tr>
<tr>
<td>600 to 603</td>
<td>600</td>
<td>0.5 to 16,383</td>
<td>1/2</td>
</tr>
<tr>
<td>300 to 303</td>
<td>300</td>
<td>1 to 32,767</td>
<td>1</td>
</tr>
<tr>
<td>12 to 15</td>
<td>12</td>
<td>25 to 819,175</td>
<td>25</td>
</tr>
<tr>
<td>8 to 11</td>
<td>8</td>
<td>37.5 to 1,228,762</td>
<td>75/2</td>
</tr>
<tr>
<td>2 to 7</td>
<td>4</td>
<td>75 to 1,228,762</td>
<td>75</td>
</tr>
</tbody>
</table>

Note: An effective value of the OX is multiple of 4. Though OX data can be designated in the range of 2 to 16383, the data will be converted to a multiple of 4 and the converted value is used to determine a frequency multiplication. (For more details, please refer to the command “OX”.)

Note: When setting each data, please refer to each item in Chapter 10 “Command Reference”, command “OH”, “OL”, “OC”, “OS” and “OX”.

7-2 Setting for the command “OH”, “OL”, “OC”, “OS” and “OX”

Once you set these commands, setting value is valid until you put off the power or reset the value. Therefore, when you rotate a motor by the same speed pattern, there is no need to set speed data every time before using motor rotation command.
7-3 Relationship between accel./decel. time and speed when total pulse is few

If the time to rotate a motor by trapezoidal or S-curve accel/decel drive is less than acceleration data(tacc) + deceleration time, the time for the acceleration is not enough. Thus, the motor doesn't accelerate from Low (fL) to High (fH) as the diagrams below, and starts to decelerate at fH' on the way.

![Diagram showing trapezoidal and S-curve acceleration](image)

7-4 Position data

(1) Alternate 2-axis control mode  (Command "EM"=0)

The RC-234 can control position pulse number by the following methods.

- Absolute method (Moves a motor to the specified pulses from the home position.)
- Incremental method (Moves a motor by the specified pulses from the current position.)

The range of position pulse number is as follows:

<table>
<thead>
<tr>
<th>Mode</th>
<th>Position pulse number specified by the command “2” and “A”</th>
<th>Data memory position</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 to 16,777,215 or -8,388,608 to +8,388,607</td>
<td>0 to 16,777,215 or -8,388,608 to +8,388,607</td>
</tr>
<tr>
<td>1</td>
<td>-8,388,608 to +8,388,607</td>
<td>-8,388,608 to +8,388,607</td>
</tr>
<tr>
<td>2</td>
<td>0 to 16,777,215 or -8,388,608 to +8,388,607</td>
<td>0 to 16,777,215 or -8,388,608 to +8,388,607</td>
</tr>
</tbody>
</table>

Use the command “EP” to select signed or unsigned data (uni-directional or bi-directional counting) in mode 0 or 2.

(2) Simultaneous 2-axis control mode  (Command "EM"=1)

<table>
<thead>
<tr>
<th>Mode</th>
<th>Position pulse number specified by the command “2”, “A” and “MA”</th>
<th>Data memory position</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, 1, 2</td>
<td>-8,388,608 to +8,388,607</td>
<td>-8,388,608 to +8,388,607</td>
</tr>
</tbody>
</table>

The command “EP” is invalid.

[Note]
If a motor is rotated out of the range of data memory position, data memory position isn’t guaranteed after that.
7. Stepping Motor Speed Adjustment

◆Mode 1 limitation
In case of Mode 1, in addition to the above there are limitations as follows;

① Position pulse number set by commands such as “2” or “A” is managed by pulse number output from not the RC-234, but encoder.

② Position pulse number of commands such as “2” or “A” must meet the following inequality. (Refer to the command “PA” and “PB”.)

\[
\left\lfloor \frac{\text{Position pulse number} \times \text{PB data}}{\text{PA data}} \right\rfloor < 8,388,608
\]

Attempting to set the position pulse number beyond the following restriction is invalid and results in a command error.

・ Command “3”, “4”, “5”
  (Command to move the motor on the basis of pulse data of command “2”)
・ Command “B”, “B+(-)”
  (Command to move the motor on the basis of pulse data of command “A”)
  (Command to move the motor in accordance with the position pulse number set by the command “MA”)

③ In spite of the setting value of command “EP”, position pulse number is signed data (-8,388,608 to +8,388,607).

④ Speed(Command “OH”, “OL”, etc.) is set on the basis of actual pulse output frequency (pps) from the RC-234. Also, the pulse data used by home(ORG) search commands such as “0” is set according to the actual pulses put out from the RC-234. (Encoder’s pulse number is invalid.)
8. Communication Check

【Be sure to read this section before using the RC-234.】

8-1 Communication (RC-234 ⇔ PC)

This section takes the example of sample program using Quick Basic and describes communication method between PC and the RC-234 and the notice.

I ) Wiring diagram

- Set the body number (yellow rotary switch) to “1” for testing.
- Connect RXD OUT(6) and RXD END(7) on the controller’s 10 pin RS232C connector. (The above connection would mean there is only one controller in the chain.)
- Refer to the above diagram and Chapter 5 “Wiring of Current Loop Signal Line” and connect PC, RC-002 and RC-234 for testing communication. If you confirm only the communication status between PC and RC-204A, drivers are not needed

Note: Start the test by checking Power connection ensure that polarity are correct before turning the power ON.
II ) Sample communication program using Quick Basic

• Description
  The RC-234 has original commands.
  The sample program's main function is to send commands and display reply from the controller
  on the monitor screen.

• Program:

'RS.BAS
OPEN "com1;9600,n,8,1" FOR RANDOM AS #1 'open com port for communication
ON COM(1) GOSUB RECEIVE 'jump to receive subroutine
COM(1) ON
MAIN:
A$ = INKEY$ 'accept key input
IF A$ = "" THEN GOTO MAIN 'wait for key input
PRINT #1, A$; 'send to ascii character to the controller
PRINT A$; 'print the ascii character on the computer screen
GOTO MAIN 'repeat the key input
END

RECEIVE:
WHILE LOC(1) <> 0 'wait until PC receive reply from the controller
  N$ = INPUT$(LOC(1), #1) 'store the reply into N$ string
  PRINT N$ 'print the reply
WEND
RETURN

III ) Communication Check

A. Complete wiring
B. Set the computer to 9,600 baud rate setting.
C. Turn the power to the Controller and the Linkmaster (RC-002).
D. After loading Quick Basic and open “RS.bas”.
E. Run the program.

F. Send the $1 command by typing $1\r.
   (Carriage Return character) is entered by pressing enter key.
   (As for the command, please refer to Chapter 9 “Command Explanation”.
   Sending this command to the RC-234 results in the following three responses;
1. Display [>$10] on the screen

Communication is normal/successful.
Go ahead to Chapter 8-2 “Motor motion check”.

2. Display not [>$10] but any response on the screen

Communication is normal/successful
Send [$1] again.

3. No reply from the RC-234

If communication is normal/successful, the LED on RC-002 should flash when each character is sent and when the controller replies back.
While checking if LED is flashing or not, send [$1] again.

Communication Failure

① When LED is flashing (No reply from the RC-234):

- Check if the body number in the program is the same as set with digital switch at the RC-234.
- Check the wiring between LinkMaster and the RC-234.
- Is power supplied the RC-234?
- The sample program is terminated once. Then check the communication setting (RS-232C control, 9600 bps, 8 bit, no parity, one stop bit).

② When LED is not flashing:

- Check the connection between PC and LinkMaster.
- Check if LinkMaster is connected with DC power.
- Check if the last RC-234 in which terminals 6 RXD OUT and 7 RXD END are shorted.
- Check if the serial port is in modem specification.
  In DB-9, short 7 and 8 pins, and 4 and 6 pins.
  In DB-25, short 4 and 5 pins, and 6 and 20 pins.
- Check the program
  - Check if a signal is out to TXD at the PC every time a key pressed.
    Use multimeter to measure voltage.
  - Check if file numbers (#1, #2) is unified for the serial lines.
  - Check if the serial line is open.
- Check LED at the LinkMaster is broken or not connected properly.
  If LED is shorted and the communication becomes normal, LED is broken.
8-2 Motor Motion Check

<Wiring of driver and Generate master >

( I ) Before execute this chapter, execute Chapter 8-1 “Communication Check” (Ⅲ) and check if the communication is normal/successful.

( II ) Use any RORZE drivers RD-0** driver for the test.

Connect limit sensors as shown in the diagram. Use sensor with active low (normally open) output. If the sensor has active high (normally closed) output change the logic using the command “EA”.

Regarding to the wiring of motor and driver, please refer to the instruction manual for driver. [Note] Make sure that power is off when wiring.

( III ) If a motor will make the extraordinary noise or start to run when the power is turned on, put off the power immediately and confirm the wiring.

( IV ) Please send [S10] (Mechanical home search command) using the program in Chapter 8 “8-1 Communication”. If there is a reply [>] from the RC-234 and a motor stops at the point of home (ORG) sensor, the communication is normal/successful. (As for the home search, please refer to the command "0").
① When a reply is returned, but a motor doesn’t run
   · Check the status by sending [$1\text{ }\mid$]

   (1) Reply: >$11 \ldots$ The controller is generating pulses.
       Check the motor to driver wiring.
       Check if driver is set 2-clock logic.
       Check the driver to controller wiring.
       Check if driver has power.

   (2) Reply: >$12 \ldots$ Limit error
       Check CCW or CW limit sensor is ON.
       Check sensor logic (Active low/Active high). Change it using the command “EA”.

   (3) Reply: >$18 \ldots$ Command error
       Resend [$10\text{ }\mid$] after correcting the error.

② When a motor does not stop
   Sensor is not wired correctly or is broken.

③ When a motor malfunctions
   Check the motor to driver wiring.
   Check speed setting, use speed commands to change speed and acceleration.

   Resend the command [$10\text{ }\mid$] again and check the operation.
   If the motor runs and stops after searching home (ORG) sensor, the motor wiring is correct.
8. Communication Check
<Software>

9. Command Explanation
10. Command Reference
11. User Program Explanation
12. User Program Command Reference
9. Command Explanation

9-1 Sending format

When sending a command from PC to RC-234, follow the below format.

Format: \$ B\#  Command  CR

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>Dollar sign</td>
<td>Shows the beginning of a command. Dollar sign is always followed by B#. 24 hex in ASCII code.</td>
</tr>
<tr>
<td>B#</td>
<td>Body number</td>
<td>This is the number identified by the Rotary Switch. There are 15 kinds, number 0 through E in hexadecimals. One of the 15 can be used. F is for testing.</td>
</tr>
<tr>
<td>CR</td>
<td>Carriage return</td>
<td>Indicates the end of the data or command. (0DH in ASCII code)</td>
</tr>
</tbody>
</table>

Note:
- Normal RC-234 is \$ series using \$ (dollar sign) which indicates the beginning of a command, but we can supply [#] series using # (sharp sign) by placing a special order.
  Using only \$ series, rotary switch to set body no. of RC-234 has a limitation and can control only up to 15 pcs, but if add the [#] series, come to be able to control up to 20 pcs.

- If the command format is in wrong, for a command error a flag will be set and the command will be ignored. (See the command “(NULL)” and “9”.)

- If you send the command to run the motor while running the motor, for a command error a flag will be set and the command will be ignored. (See the command “(NULL)” and “9”.)

- Some commands to set the data have default. Because the default is not always “0”, see each command explanation or Chapter 16 “Command Default Setting”.
  Also, when write the data set by the command “AW”, “DW”, etc. in EEPROM, it is treated as the default and the value in Chapter 16 “Command Default Setting” is invalid.
  Execute the command “EE///” to initialize the default.
• Some commands set the data of motor 1, 2 in common and the others set the data separately. As for the commands to set the data separately, even if the format is the same, depending on the specified motor, data memory area is different. Therefore, before executing the command to set data separately, confirm the specified motor or specify motor by the command “F”. Even if setting value of motor 1 and 2 is the same, in case of the command to set the data separately set the data separately. (Before setting, specify the motor by the command “F”.) The default for motor 1 and 2 is the same.

• The commands to run the motor, “0”, “1”, “3”, “4”, “5”, “7”, “8”, “B”, etc. can specify motor 1 or 2 at the last of command.

Example $ 1 3 2 (Moves motor 2 to the specified position)

This example is equivalent to the action specifying motor 2 by the command “F” and sending command “3”. Therefore, if you send a command not to specify the motor (for example, command “4”), after executing the command to specify the motor (in the above example, command “32”) the command is valid for motor 2.

• In case of mode 1 using encoder, the position pulse number and data memory position range set by the command “A” etc. are based on input pulse from encoder. However, speed and home (ORG) search aren’t affected by encoder. (See Chapter 7 “Stepping Motor Speed Adjustment”.)

• Once you execute each setting command, the setting will be valid until turning off the power or resetting the data.
9-2 Reply format

When send the command from PC to the RC-234, the reply is returned from the RC-234. The reply format is as follows;

Reply : ( I ) > or ?

Reply format when sending a general command

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;</td>
<td>Communication</td>
<td>Normal This will be returned when communication is normal. (This indicates normality of communication, not normality of command).</td>
</tr>
<tr>
<td>?</td>
<td>Communication</td>
<td>abnormal This is returned when communication is abnormal, not when the command is abnormal.</td>
</tr>
</tbody>
</table>

Reply : ( II ) > $ B# DATA \n
Reply format when sending a query command (for example, command “6”, “9” etc.)

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;$</td>
<td>Prefix of the reply for the query command</td>
</tr>
<tr>
<td>B#</td>
<td>Indicates the body no. (0 to E) of the RC-234 returning reply.</td>
</tr>
<tr>
<td>DATA</td>
<td>Reply for the query command</td>
</tr>
<tr>
<td>\n</td>
<td>Carriage return (0DH) Indicates data end.</td>
</tr>
</tbody>
</table>

Reference

The following cause is conceivable for reasons that “?” is returned as a reply.

When the PC power is turned ON or OFF while the RC-234 power is ON, noise of power supply gets on the communication wire of RC-232C and the RC-234 sometimes receives meaningless noise. After this, when the RC-234 restores the communication with PC and a PC sends a valid command to the RC-234, “?” is sometimes returned.

This phenomenon will occur just after starting communication between PC and the RC-234. Therefore, if you add a routine to repeat sending the same command until receiving a normal reply once you receive “?” in the program, you can prevent the above malfunction.
10. Command Reference

This chapter describes commands for the RC-234. The commands arrange in numeral and alphabetical order.

Notice in context
- Example: PRINT #1, "$1; CHR$(&HD);
  CHR$(&HD) at the above command stands for carriage return at ASCII code.
  It is 13 at decimal and 0D(H) in HEX.

- Hexadecimal is abbreviated to HEX.

- As for the command mark [$], [B#] and [\], see Chapter 9 “Command explanation”.

<table>
<thead>
<tr>
<th>E0</th>
<th>E1</th>
<th>E2</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

E0, E1 and E2 represent mode 0,1,2 respectively.

○...Valid in this mode
×...Invalid in this mode
△...Invalid for some formats in this mode

During motion EEPROM
<table>
<thead>
<tr>
<th>○</th>
<th>×</th>
</tr>
</thead>
</table>

During motion... indicates whether the command can be used during motion.
EEPROM... indicates whether the command can be used in the user program.

○...Valid in this mode
×...Invalid in this mode
△...Invalid for some formats in this mode

As for the EEPROM and the user program, see Chapter 11 “User Program Explanation”

Important Notice

In this section, there are some sample programs to transmit commands sequentially and they are written by sequential form to save space of explanation. When you program practically, please be sure to describe the program “transmits a command, receives the reply for the command, and then transmits next command”.

In the example described in the command “3”,

1. PRINT #1, "$120200030"; CHR$(&HD);
2. PRINT #1, "$131"; CHR$(&HD);

1,2 are transmitted sequentially, but in fact, the routine to receive [>] which is the reply of Command “2” is required after 1 (between 1 and 2).
Similarly, the routine to receive [>] is required after 2.
As for the actual program, please see (Sample program in mode 0) in Chapter 13.2 “Commands for each mode and Sample Program”.

The RC-234 has three motion modes (Please see the command “E”) and the commands that can be used in any operation mode by way of explanation are called common command and the commands that can be used only in the specific mode are called exclusive command.
Command " " (NULL)

<table>
<thead>
<tr>
<th>E0</th>
<th>E1</th>
<th>E2</th>
<th>During motion</th>
<th>EEPROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
<td>○</td>
<td>×</td>
<td></td>
</tr>
</tbody>
</table>

Action: This command reads the status of operation – busy, idle, limit error, command error, etc.

Format: $ B#  

Reply: $ B# DT  

DT... Represents individual status bits in a hexadecimal notation. Each bit has a meaning as follows:

- Command error (bit 3)
- Emergency stop (bit 2)
- Limit error (bit 1)
- Motor motion (bit 0)

<Explanation of individual bits>

- Motor motion (bit 0)
  - bit 0 = 1 means that motor is running.
  - bit 0 = 0 means that motor is idle.

- Limit error (bit 1)
  - bit 1 = 1 means that the RC-234 has detected a stall error or the activation of the CW(CCW) limit sensors or the software limit during a motor's rotation. Additionally, bit 1 is set to 1 when the RC-234 has detected other error(s) in motor operation. See "Note-7" described in the command “9”, for more information.
  - bit 1 = 0 means that the RC-234 has detected no sensor error. Additionally, bit 1 is set to 1 when a RC-234 has detected other error(s) in motor operation.

- Emergency stop (bit 2)
  - bit 2 = 1 means that a RC-234 has detected the activation of the EMS input by the time it receives the command "9". Note that this bit does NOT provide the status at the time a command is executed.
  - bit 2 = 0 means that the RC-234 has detected no EMS input.

- Command error (bit 3)
  - bit 3 = 1 means that a RC-234 has received an ineffective command(s): (an) invalid command(s) with regard to format; or (a) motion command(s) whether the motor is in motion or not.
  - bit 3 = 0 means that all of the received commands were valid.
10. Command Reference

Command ""

Reply Sample: (I)  > $ 1 0 $
No error has been detected in the RC-234 with body number 1, and the motor is idle.
HEX data 0 → binary data: 0 0 0 0

(II) > $ 2 9 $
A command error has been detected in the RC-234 with body number 2, and the motor is running.
HEX data 9 → binary data: 1 0 0 1

(III) > $ 1 A $
A command and sensor error have been detected in the RC-234 with body number 1, and the motor is idle.
HEX data A → binary data: 1 0 1 0

Example
- PRINT #1,”$2”;CHR$(&HDL):
  Read the status of a motor motion and an error in the RC-234 with body number 2.

Note
- When command "(NULL)" is executed, a RC-234 sets each value of bit1,2,3 (except bit0) to 0 after querying the status of status flags and condition flags.

- The command "(NULL)" execution does not affect the status flags and the flags will remain intact the status. Note that the command immediately preceding the command "(NULL)" may have not caused an error, even if the reply to the "(NULL)" command shows an occurrence of an error.

- The command "9" can be also used to read error status such as a command error. The status flags for command "(NULL)" and "9" do not share a stored area. Therefore the execution of command "9" will clear the flags in its area but will not clear the flags for command "(NULL)".

- In simultaneous 2-axis motion, the status for motor 1 and motor 2 will be in logical OR. To query the status of each motor, please use the command "9D".

- The status indication LED ERR will light when either of status bit, such as the limit error (bit1), the emergency stop (bit2), the command error (bit3) is turned to 1 and turn off the light when the command " " (NULL), "9ALL" (error status query), a home search command, such as "0", "0A", "0H", "0Q", "0R", "0Z" or the user command "/EC" is performed.
10. Command Reference

**Command 0**

<table>
<thead>
<tr>
<th>E0</th>
<th>E1</th>
<th>E2</th>
<th>During motion</th>
<th>EEPROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>√</td>
<td></td>
<td></td>
<td>×</td>
<td></td>
</tr>
</tbody>
</table>

**Action**: This command searches for the home (ORG) sensor at the low speed determined by "OL" and "OX", and establishes the home position, where the pulse counter equals to 0. (Mechanical Home Search #1)

**Format** :

(I)  
$ B# 0$  
Performs Mechanical Home Search #1.

(II)  
$ B# 0 MT$  
Performs Mechanical Home Search #1 for a specified motor.  
MT: 1 or 2  
3: simultaneous start (When command “EM”=1)

**Reply**:  
> (For both I and II)

**Example**  
*PRINT #1,”$1 0”;CHR$(&HD)*;  
Sends a Mechanical Home Search command to the RC-234 with body number 1.

**Description**  
- There are three types of behavior, based on the motor position when a "0" command is executed:

**A. When the starting position is located at the CW side of the home (ORG) sensor,**

the motor moves in the CCW direction at the low speed until the home (ORG) sensor is activated, keeps rotating the "0 S" pulses farther, and stops. The position where the motor stops is defined as the home position and the pulse counter is set to 0.

![Diagram](image-url)
B. When the starting position is on the home (ORG) sensor (the sensor is ON),

the motor moves in the CW direction at the low speed until the home (ORG) sensor is
deactivated, keeps rotating \( S \times B \) pulses farther, and stops.
Then it reverses in the CCW direction until the home (ORG) sensor is activated again, keeps
rotating the \( S \) pulses farther, and stops. The position where the motor stops is defined
as the home position and the pulse counter is set to 0.

C. When the starting position is located at the CCW side of the home (ORG) sensor,

the motor moves in the CCW direction at the low speed until the CCW sensor is activated,
and stops. Then it rotates in the CW direction until the home (ORG) sensor is activated and
then deactivated, keeps rotating \( S \times B \) pulses farther, and stops.
Then, it rotates in the CCW direction until the home (ORG) sensor is activated again, keeps
rotating the \( S \) pulses farther, and stops.
The position where the motor stops is defined as the home position and the pulse counter is
set to 0.

The values of \( S \) and \( B \) default to 20 and 2, respectively. Those default values
will be used when a \( S \) command is executed immediately after the RC-234 is powered.
However, if the values of \( S \) and/or \( B \) have been stored in the EEPROM by "DW"
command, the stored values will be used as \( S \) and \( B \) parameters.
For more information on the default values, see Chapter 16 "Command Default Setting".
10. Command Reference

**Command \( \theta \)**

**Note**
- When the format (I) is executed, the target motor is the one that is currently specified. The command "F" can be used to query or specify the target motor.

- If a home (ORG) sensor is not connected to the system, the command "\( \theta \)" makes a motor keep running until the CW sensor is activated. If you need to stop the motor manually, use either "S" or "SS" command.

- Avoid placing sensors in such a manner that an ORG and CW or ORG and CCW sensors are activated simultaneously. With such layout, RC-234s can't correctly detect the home (ORG) sensor.

- Even in an operation mode 1, pulse value specified by the command "\( \theta \) S" is managed on an output-pulse basis rather than an encoder-pulse basis.

- In an operation mode 2, the CLR output, a signal to clear the counter of servo drives, will be activated at the time set by the command “DCT” when the RC-234 finishes to output the pulses specified by "\( \theta \) S" and completes the Home Search function.

- The status indication LED ERR will light when either the limit error, the emergency stop or the command error occurs and turn off the light when the command " " (NULL), "9ALL" (error status query), a home search command, such as "0", "0A", "0H", "0Q", "0R", "0Z" or the user command "/EC" is performed.
10. Command Reference

Command $\theta$ A

Action: This command searches for the home (ORG) sensor at the high speed determined by "OH" and "OX", and establishes the home position, where the pulse counter equals to 0.

(High Speed Home Search #2)

Format:

(I) $S$ B# $\theta$ A $\dagger$
Performs High Speed Home Search #2.

(II) $S$ B# $\theta$ A MT $\dagger$
Performs High Speed Home Search #2 for a specified motor.

MT.. 1 or 2
3: simultaneous start (When command “EM”=1)

Reply: $>$ (For both I and II)

Description

- There are two types of behavior based on the motor position when a "$\theta$ A" command is executed.

A. When the starting position is located at the CW side of the home (ORG) sensor or on the home (ORG) sensor,

  the motor rotates in the CCW direction at the high speed until the home (ORG) sensor is activated, and reverses in the CW direction at the low speed. When the home (ORG) sensor is activated again (in case the sensor is OFF) and then deactivated, the motor keep rotating "$\theta$ S" X "$\theta$ B" pulses farther and stops. Then, it rotates in the CCW direction until the home (ORG) sensor is activated again, keeps rotating the "$\theta$ S" pulses farther, and stops. The position where the motor stops is defined as the home position and the pulse counter is set to 0.

Note

As soon as the home (ORG) sensor is activated at the first time, the RC-234 changes the pulse speed to the low speed without deceleration. Therefore, a stall may occur and the motor may slip to some degree before stopping if it has been running at an extremely high speed.

Thus, use the command "$\theta$ A" keeping in mind the possibility that the work may bump against a mechanical CCW stopper if the distance is too small between the home (ORG) sensor and the stopper.

When a "$\theta$ A" function is initiated with the home (ORG) sensor activated, the motor rotates, from the beginning, in the CW direction at the low speed until the home (ORG) sensor is deactivated, keeps rotating "$\theta$ S" X "$\theta$ B" pulses farther, and stops.
10. Command Reference

Command \( \theta A \)

Then, it rotates in the CCW direction until the home (ORG) sensor is activated again, keeps rotating the \( \theta S \) pulses farther, and stops. The position where the motor stops is defined as the home position and the pulse counter is set to 0.

B. When the motor position is located at the CCW side of the home (ORG) sensor,

the motor rotates in the CCW direction at the high speed until the CCW sensor is activated, and reverses in the CW direction at the low speed. From the position where the home (ORG) sensor is activated and subsequently deactivated, the motor keeps rotating \( \theta S \times \theta B \) pulses farther and stops. Then, it rotates in the CCW direction until the home (ORG) sensor is activated again, keeps rotating the \( \theta S \) pulses farther, and stops. The position where the motor stops is defined as the home position and the pulse counter is set to 0.

Note
As soon as the CCW sensor is activated when the motor rotates in the CCW direction at the high speed, the RC-234 stops the pulse generation. Therefore, a stall may occur and the motor may slip to some degree before stopping if it has been running at an extremely high speed. Thus, use the command \( \theta A \) keeping in mind the possibility that the work may bump against a mechanical CCW stopper if the distance is too small between the CCW sensor and the stopper.

<Comparison between \( \theta A \) and \( \theta H \)>

There are two types of High Speed Home Search, \( \theta A \) and \( \theta H \). With a \( \theta A \) command, a motor immediately stops (without ramping deceleration) and continue the Home Search function at the low speed when the motor crosses the home (ORG) sensor at the High speed. On the other hand, \( \theta H \) decelerates from high to low speed and continue the function in the same situation.

Note
- In an operation mode 2, the CLR output, a signal to clear the counter of servo drives, will be activated at the time set by the command “DCT” when the RC-234 finishes to output the pulses specified by \( \theta S \) and completes the Home Search function.
- The status indication LED ERR will light when either the limit error, the emergency stop or the command error occurs and turn off the light when the command “” (NULL), “9ALL” (error status query), a home search command, such as “0”, “0A”, “0H”, “0Q”, “0R”, “0Z” or the user command “/EC” is performed.
**Command B**

<table>
<thead>
<tr>
<th>E0</th>
<th>E1</th>
<th>E2</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

**Action**: Sets a multiplication factor for the "S" parameter.

The multiplication value of "S" and "B" is used by a Home Search command such as "0", "R", "H", or "A".

For more information, see the description of the command "0".

The parameter should be separately specified for the motor 1 and 2.

**Format**: (I) $ B\# \ B \ DT$

Sets a multiplication factor for the "S" parameter.

DT.. Multiplication factor (0 to 5)

【DEFAULT=2】

(II) $ B\# \ B \ DT$

Queries the current value specified by the format (I).

The query is separately made for each motor.

**Reply**: In case of (I) >

In case of (II) > $ B\# \ DT$

DT.. Multiplication factor (0 to 5)

**Description**

- For more information, see the figure in the section of command "0".
10. Command Reference

**Command**  \( \emptyset H \)

<table>
<thead>
<tr>
<th>E0</th>
<th>E1</th>
<th>E2</th>
<th>During motion</th>
<th>EEPROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>×</td>
<td>☐</td>
</tr>
</tbody>
</table>

**Action**: This command searches for the home (ORG) sensor at the high speed determined by "OH" and "OX", and establishes the home position, where the pulse counter equals to 0.

(High Speed Home Search #1)

**Format**: (I) \( $ B# \emptyset H \)  
Performs High Speed Home Search #1.

(II) \( $ B# \emptyset H MT \)  
Performs High Speed Home Search #1 for a specified motor.

M T... 1 or 2  
3: simultaneous start (When command "EM"=1)

**Reply**: \( > \)  
(For both I and II)

**Description**

- There are two types of behavior based on the motor position when a "\( \emptyset H \)" command is executed,

  **A.** When the starting position is located at the CW side of the home (ORG) sensor or on the home (ORG) sensor,

  the motor rotates in the CCW direction at the high speed until the home (ORG) sensor is activated. Then it decelerates to the low speed, stops, and begins rotating in the CW direction at the low speed. When the home (ORG) sensor is activated again (in case the sensor is OFF) and then deactivated, the motor keep rotating "\( \emptyset S \times \emptyset B \)" pulses farther and stops. Then, it runs in the CCW direction until the home (ORG) sensor is activated again, keeps rotating the "\( \emptyset S \)" pulses farther, and stops. The position where the motor stops is defined as the home position and the pulse counter is set to 0.

**Diagram**

If the CCW sensor is activated during deceleration, the motor immediately stops and reverses the rotation in the CW direction to continue the Home Search.

Therefore, use the command "\( \emptyset H \)" keeping in mind the possibility that the work may bump against a mechanical CCW stopper if the distance is too small between the home (ORG) sensor and the stopper. When a "\( \emptyset A \)" function is initiated with the home (ORG) sensor activated, the motor rotates, from the beginning, in the CW direction at the low speed until the home (ORG) sensor is deactivated, keeps rotating "\( \emptyset S \times \emptyset B \)" pulses farther, and stops. Then, it runs in the CCW direction until the home (ORG) sensor is activated again, keeps rotating "\( \emptyset S \)" pulses farther, and stops. The position where the motor stops is defined as the home position and the pulse counter is set to 0.
When the motor position is located at the CCW side of the home (ORG) sensor, the motor rotates in the CCW direction at the high speed until the CCW sensor is activated, and reverses in the CW direction at the low speed. When the home (ORG) sensor is activated and then deactivated, the motor keeps rotating "θ S" X "θ B" pulses farther and stops. Then, it rotates in the CCW direction until the home (ORG) sensor is activated again, keeps rotating the "θ S" pulses farther, and stops. The position where the motor stops is defined as the home position and the pulse counter is set to 0.

Note
As soon as the CCW sensor is activated, the RC-234 stops the pulse generation. Therefore, a stall may occur and the motor may slip to some degree before stopping if it has been running at an extremely high speed. Thus, use the command "θ H" keeping in mind the possibility that the work may bump against a mechanical CCW stopper if the distance is too small between the CCW sensor and the stopper.

<Comparison between "θ A" and "θ H">
There are two types of High Speed Home Search, "θ A" and "θ H". With a "θ A" command, a motor immediately stops (without ramping deceleration) and continue the Home Search function when the motor crosses the home (ORG) sensor at the High speed. On the other hand, "θ H" decelerates from high to low speed and continue the function in the same situation.

Note
- In an operation mode 2, the CLR output, a signal to clear the counter of servo drives, will be activated for the time set by the command “DCT" when the RC-234 finishes to output the pulses specified by "θ S" and completes the Home Search function.
- The status indication LED ERR will light when either the limit error, the emergency stop or the command error occurs and turn off the light when the command “ " (NULL), “9ALL" (error status query), a home search command, such as "0", "0A", "0H", "0Q", "0R", "0Z" or the user command "/EC" is performed.
Action: This command locates the stall sensor at the middle of the slit for stall detection and establishes the Home position. See the description below.
This command is exclusively used in an operation mode 0.
The execution of "Ø Q" is not needed if stall detection is disabled by the command "QS".
The parameter should be separately specified for the motor 1 and 2.

Format:

(I) $ B# Ø Q 
Locates the stall sensor at the middle of the slit for stall detection.
The stall sensor must be activated when the command is issued.
If the sensor is deactivated at the time of issuing of the "Ø Q" command, it will result in a limit error.
Use the command "EA" to invert the input logic of the sensor before using the command "Ø Q".

(II) $ B# Ø Q pd 
Sets the pulses during which the stall input is assuredly activated, to avoid malfunctions during execution of the command format (I).
pd.. pulse data decimal number less than 5 digits (1 to 65,535) 
【DEFAULT=10】

(III) $ B# Ø Q D 
Queries the current setting specified by the format (II).

(IV) $ B# Ø Q W 
Queries the actual pulse number during which the stall input has been kept activated when the command "Ø Q" is executed in the format (I).

Reply: In case of (I, II) >

In case of (III, IV) > $ B# ◯◯◯◯◯ ◯
5-digit decimal number
The pulse number during which the stall input keeps activated.
Description

There are two ways to specify pulses during which the stall input keeps activated:

1. Use the format (II) with designed a pulse number during which the stall input assuredly keeps activated.
2. Use the format (II) with the pulse number that is obtained by the format (IV) after trial execution of a "Q" command.

The purpose of command "Q"

If the stall sensor is relatively located to the slit as shown in Fig. A after execution of the command "Q", the input signal may be ambiguous because the sensor is positioned at the edge of the slit. If a "Q" command is executed in such case, the relative position between the sensor and the slit is automatically adjusted as shown in Fig. B.

Note

- Please specify ON-OFF period of stall detection sensor by the command "Q" before execute the command "Q".
- If the value specified with the format (II) is greater than the actual pulse number during which the stall input keeps activated, the format (I) can not be correctly performed.
- Use the format (II) only if you can not correctly perform the format (I) because of the chatter of the stall input.
- If you can correctly perform the format (I) with the default value of 10 pulses (pd in the format (II)), you should not execute the format (II) to specify another value.
- When a "Q" command follows a "0" command, the Home position is automatically changed to the position determined by "Q" command.
- The status indication LED ERR will light when either the limit error, the emergency stop or the command error occurs and turn off the light when the command "NULL", "9ALL" (error status query), a home search command, such as "0", "0A", "0H", "0Q", "0R", "0Z" or the user command "/EC" is performed.
10. Command Reference

### Command \( \emptyset R \)

<table>
<thead>
<tr>
<th>Command</th>
<th>E0</th>
<th>E1</th>
<th>E2</th>
<th>During motion</th>
<th>EEPROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \emptyset R )</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>×</td>
<td>○</td>
</tr>
</tbody>
</table>

**Action**: This command searches for the home (ORG) sensor at the low speed determined by "OL" and "OX", and establishes the home position, where the pulse counter equals to 0. (Mechanical home search #2)

**Format**:

1. \( \$\ B# \emptyset \ R \)  
   Performs Mechanical Home Search #2.

2. \( \$\ B# \emptyset \ R \ M T \)  
   Performs Mechanical Home Search #2 for a specified motor.  
   \( M T \ldots 1 \) or 2  
   3: simultaneous start (When command “EM”=1)

**Reply**: \( > \) (For both I and II)

**Example**

- \( \text{PRINT} \#1, \"\$1 \emptyset R1\"; \text{CHR}\$ (\&HD); \)
  Sends a Mechanical Home Search #2 command for the motor 1 to the RC-234 with body number 1.

**Description**

- There are two types of behavior based on the motor position when a "\( \emptyset R \)" command is executed,

  **A. When the starting position is located at the CW side of the home (ORG) sensor or on the home (ORG) sensor,**

  the motor rotates in the CCW direction at the low speed until the home (ORG) sensor is activated and then deactivated, and stops. Then it moves in the CW direction until the home (ORG) sensor is activated again, keeps rotating the "\( \emptyset S \)" pulses farther, and stops. The position where the motor stops is defined as the Home position and the pulse counter is set to 0.
B. When the motor position is located on the CCW side of the home (ORG) sensor.

The motor rotates in the CCW direction at the low speed until the CCW sensor is activated, and stops. Then it moves in the CW direction until the home (ORG) sensor is activated, keeps rotating the "0 S" pulses farther, and stops. The position where the motor stops is defined as the Home position and the pulse counter is set to 0. If a starting position is located, as shown below, at the CCW side of the home (ORG) sensor, a CCW sensor must be equipped.

The default value of "0 S" is 20 pulses. This default value will be used when a "0 R" command is executed immediately after the RC-234 is powered. However, if the value is stored in the EEPROM by "DW" command, the stored value will be used as the "0 S" parameter.

For more information on the default values, see Chapter 16 "Command Default Setting".

Note
- An error occurs and a motor immediately stops if the CW sensor is activated during a "0 R" execution.
- When the format (I) is executed, the target motor is the one that is currently specified. The command "F" can be used to query or specify the target motor.
- If a home (ORG) sensor is not connected to the system, the command "0 R" makes the motor keep running until the CW sensor is activated. If you need to stop the motor manually, use either "S" or "SS" command.
- The algorithm of the "0 R" command is similar to that of the Home Search function in RC-204As. However, if you need the same action as in RC-204As, set the "0 S" parameter to 6 pulses.
- In an operation mode 2, the CLR output, a signal to clear the counter of servo drives, will be activated at the time set by the command "DCT" when the RC-234 finishes to output the pulses specified by "0 S" and completes the Home Search function.
- The status indication LED ERR will light when either the limit error, the emergency stop or the command error occurs and turn off the light when the command " (NULL), "9ALL" (error status query), a home search command, such as "0", "0A", "0H", "0Q", "0R", "0Z" or the user command "/EC" is performed.
**Command Reference**

<table>
<thead>
<tr>
<th>Command</th>
<th>Ø S</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>E0</th>
<th>E1</th>
<th>E2</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

**During motion**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

**Action:**
This command sets a certain pulse number that a home-search command, such as "Ø", " Ø R", " Ø H", or " Ø A", uses during its execution. For more information, see the description of the command "Ø".

The parameter should be separately specified for the motor 1 and 2.

**Format**

(I)

\[
\$ B# \theta S \ p d \ "\]

Sets a pulse parameter that home-search commands use.

\( p d \) pulse data, decimal number less than 5 digits (1 to 65,535)

**DEFAULT=20**

(II)

\[
\$ B# \theta S \ "\]

Queries the current value specified by the format (I).

The query is separately made for each motor.

**Reply**

In case of (I) >

In case of (II) >

\[\$ B# \ "\]

5-digit decimal number

Current "Ø S" parameter

**Description**

- See the sections of "Ø", " Ø R", " Ø H", and " Ø A".

**Example**

1. `PRINT #1,"$1F1";CHR$(&HD);`
2. `PRINT #1,"$1S500";CHR$(&HD);`
   - Sets the "Ø S" parameter to 500 pulses for the motor 1 of the RC-234 with body number 1.

3. `PRINT #1,"$1F2";CHR$(&HD);`
4. `PRINT #1,"$1S30";CHR$(&HD);`
   - Sets the "Ø S" parameter to 30 pulses for the motor 2 of the RC-234 with body number 1.
Action: This command designates an algorithm with which home-search commands, such as "∅", "∅ R", "∅ H", and "∅ A", are performed. The command has two algorithms: the legacy one and the new one. The command is exclusively used in an operation mode 2 (for servo motors).

Format:

(I) $ B# ∅ X 1
Designates the new algorithm, with which home-search functions are performed. The new algorithm is implemented only if the operation mode is 2. In a mode 0 or 1, the legacy algorithm is implemented.

(II) $ B# ∅ X 0
Designates the legacy algorithm, with which home-search functions are performed. 【DEFAULT】

(III) $ B# ∅ X
Queries the current setting of "∅ X".

Reply: In case of (I, II) >

In case of (III) > $ B# DT
DT.. 0: The legacy algorithm will be implemented when a home-search function is performed.
1: The new algorithm will be implemented when a home-search function is performed.

Description

The process after a motor position is located at the CW side of the home (ORG) sensor during a Home Search execution

- Though a detailed description of motor motion is not given in the section of the command "∅", a motor actually follows the following algorithms when a motor position is moved at the CW side of the home (ORG) sensor after the Home Search is initiated:

5 pulses

The position where the RC-234 recognizes that the home sensor is activated

Sensor ON
One of the following algorithms will be implemented according to the setting of the command "0 X ".

A. The legacy algorithm ("0 X 0").

① The motor rotates in the CCW direction until the home (ORG) sensor is activated.

② As soon as the RC-234 detects the activation of the sensor, it stops the pulse output. Then it outputs five pulses more to rotate the motor in the CCW direction.

③ After completing the pulse output (5 pulses), the device checks whether the ORG is ON or OFF. If the sensor is ON, the program proceeds to the step ④. Otherwise, the device outputs five pulses again and the step ③ is processed again.

④ The RC-234 generates the pulses that have been specified by " S ", making the motor rotate in the CCW direction.

⑤ Finishing the pulse output, the device idles until the INP input is activated and completes the Home Search function after taking the following actions:
   - Activates the CLR output at the time set by the command "DCT".
   - Clears the stall error flag of command "9".
   - Sets the current position counter to 0.
   - Sets the encoder counter of the target motor to 0.
   - Clears the in-motion flag of command "(NULL)" and "9".

B. The new algorithm ("0 X 1")

① The motor rotates in the CCW direction until the home (ORG) sensor is activated.

② As soon as the RC-234 detects the activation of the sensor, it immediately stops the pulse output and activates the CLR output at the time set by the command "DCT".

③ After finishing the pulse output (5 pulses), the device idles until the INP input is activated and then checks whether the home (ORG) sensor is activated or not. If the sensor is activated, the program proceeds to the step ④. Otherwise, the device outputs five pulses again and the step ③ is processed again.

④ The RC-234 generates the pulses that have been specified by " S " command, making the motor rotate in the CCW direction.

⑤ Finishing the pulse output, the device idles until the INP input is activated and completes the Home Search function after taking the following actions:
   - Clears the stall error flag of command "9".
   - Sets the current position counter to 0.
   - Clears the in-motion flag of command "(NULL)" and "9".

(Note)

The maximum waiting time is 3 seconds in the Step ③ and ⑤, where the RC-234 idles until the INP input is activated after finishing the pulse output.

If the device receives no ON signal from the INP terminal during the time, it turns the CLR output ON at the time set by the command "DCT"; sets a limit error, and finishes the process.
Note

- This command is only used for designating the algorithm of Home Search function rather than for performing it.
- This command is valid only in an operation mode 2.
- The default setting is the "" (legacy algorithm), when RC-234s are powered. Once the command "" is executed, the setting remains valid until the "" is performed or the RC-234 is turned OFF.
- The setting of command "" can be stored in the EEPROM by the command "DW". If a setting of command "" is stored in the EEPROM, the value overrides the default value. Therefore, when a Home Search function is performed without executing a "" command, the stored value will be used (if any).
Action: This command performs a Home Search function, using the EZ (Z phase signal of an encoder) and home (ORG) sensor input.

Format:

(Ⅰ) \$ B# 0 Z 
Performs a Home Search function, using the EZ and home (ORG) sensor input.

(Ⅱ) \$ B# 0 Z DT 
Sets the number of EZ input to be sensed to establish the Home position after recognizing a home (ORG) sensor input.
D T.. 1 to 9

【DEFAULT=1】

(Ⅲ) \$ B# 0 Z D 
Queries the current number set by the format (Ⅱ).

Reply: In case of (Ⅰ, Ⅱ) >

In case of (Ⅲ) > $ B# DT 
D T.. The number of the Z phase input to be sensed (1 to 9).

Description:

- There are two types of actions based on the motor position when the format (Ⅰ) is sent.
  In the description below, the number specified by the format (Ⅱ) is assumed to be 4.

A. When the motor position is located at the CW side of the home (ORG) sensor,

the motor begins to rotate in the CCW direction at the low speed. When the RC-234 recognizes that the home (ORG) sensor keeps activated at least for 1 msec, it starts to check the Z phase input.

The check of the Z phase input is performed every 1 msec. If the RC-234 senses an activation of the input, it waits for the signal to turn OFF and then begins to detect another ON status of the input.

The above procedure is repeated. When the RC-234 detects the fourth ON status of the input, it reverses the motor in the CW direction. As the input turns OFF, the device stops the motor, rotates it five pulses in the CCW direction, and stops it. The position where the motor stops is defined as the Home position and the pulse counter is set to 0.
B. When the motor position is located at the CCW side of the home (ORG) sensor,

the motor rotates in the CCW direction at the low speed. When the RC-234 recognizes that
the CCW sensor is activated, it reverse the motor in the CW direction and starts to check the
home (ORG) sensor.

From the position where the home (ORG) sensor is activated and subsequently deactivated,
the motor keeps rotating "0 S" X "0 B" pulses farther and stops.

Then, the RC-234 reverses the motor in the CCW direction. When the device recognizes
that the home (ORG) sensor keeps activated at least for 1 msec, it starts to check the Z
phase input.

The check of the Z phase input is performed every 1 ms. If the RC-234 senses an
activation of the input, it waits for the signal to turn OFF and then begins to detect another
ON status of the input.

The above procedure is repeated. When the RC-234 detects the fourth ON status of the
input, it reverses the motor in the CCW direction. As the input turns OFF, the device stops
the motor, rotates it five pulses in the CCW direction, and stops it. The position where the
motor stops is defined as the Home position and the pulse counter is set to 0.

C. When the motor position is located on the home (ORG) sensor,

the motor rotates in the CW direction at the low speed. From the position where the home
(ORG) sensor is deactivated, the motor keeps rotating "0 S" X "0 B" pulses farther and stops.

Then, the RC-234 reverses the motor in the CCW direction. When the device recognizes
that the home (ORG) sensor keeps activated at least for 1 msec, it starts to check the Z
phase input.

The check of the Z phase input is performed every 1 msec. If the RC-234 senses an
activation of the input, it waits for the signal to turn OFF and then begins to detect another
ON status of the input.

The above procedure is repeated. When the RC-234 detects the fourth ON status of the
input, it reverses the motor in the CW direction. As the input turns OFF, the device stops
the motor, rotates it five pulses in the CCW direction, and stops it. The position where the
motor stops is defined as the Home position and the pulse counter is set to 0.
Command **Φ Z**

Note

- All figures in the previous description show such cases that the home (ORG) sensor and the Z phase (EZ) input are simultaneously activated for a certain period. However, the both inputs do not actually need to be synchronized.

- Sensing the activation of the home (ORG) sensor and the Z phase signal needs at least 2 msec. Therefore RC-234s may not correctly recognize the sensor if the sensor status keeps less than 2 msec.

- In a mode 2, a RC-234 activates the CLR output (Clear counter signal for servomotor drives) at the time set by the command “DCT” after the completion of outputting 5 pulses at the end of the process.

- Please execute the command "Φ Z." when one of motors is stopping.

- The status indication LED ERR will light when either the limit error, the emergency stop or the command error occurs and turn off the light when the command “” (NULL), “9ALL” (error status query), a home search command, such as ”0”, ”0A”, ”0H”, ”0Q”, ”0R”, ”0Z” or the user command “/EC” is performed.

- In some cases, RC-234s may not recognize the first ON input of the Z phase signal, which is caused by the ON timing of the home (ORG) sensor and the Z phase input. The following is the description of six scenarios of the timing:
Fig. A: After a RC-234 confirms that the home (ORG) sensor remains activated for 1 ms, it checks the Z phase input. If the input is activated, as in the case, the device checks the input again 1 msec later. If the input still remains activated, the device recognizes that the first Z phase signal exists.

Fig. B: After a RC-234 confirms that the home (ORG) sensor remains activated for 1 ms, it checks the Z phase input. If the input is deactivated, as in the case, the device checks for the signal every 1 msec from then. If the device finds the input pulse that keeps ON for 1 msec, it recognizes that the first Z phase signal exists.

Fig. C: A RC-234 recognizes that the first Z phase signal exists, as in Fig. B. As Fig. C shows, the ON period of the Z phase signal (EZ) should be 2 ms or more so that the RC-234 recognizes that the Z phase signal exists.

Fig. D: A RC-234 does not recognize the Z phase signal shown in Fig. D because it starts to search for the signal after confirming that the home (ORG) sensor remains ON for 1 msec.

Fig. E: Though the Z phase signal is ON when a RC-234 checks the input after confirming that the home (ORG) sensor has been remained ON for 1 msec, the device does not count the signal as a valid input. The reason is that the Z signal does not keep ON for 1 msec.

Fig. F: A RC-234 recognizes that the first Z phase signal exists as in Fig. B. As shown in Fig. F, the home (ORG) sensor and the Z phase signal do not need to be ON at the same time.
### Command 1

<table>
<thead>
<tr>
<th>E0</th>
<th>E1</th>
<th>E2</th>
<th>During motion</th>
<th>EEPROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
</tr>
</tbody>
</table>

**Action**: This command moves the motor, at the high speed determined by "OH" and "OL" with ramping acceleration/deceleration, to the electrical Home position (i.e. position counter = 0).

**Format**:  
(I) \( $ B# 1 \)  
Moves a motor to the Home position.

(II) \( $ B# 1 \) \( MT \)  
Moves a specified motor to the Home position.  
MT.. 1 or 2  
3: simultaneous start (When command “EM”=1)

**Reply**:  
> (For both I and II)

**Example**  
PRINT #1, “$ 2 1” : CHR $(& HD) ;  
This line instructs the RC-234 with body number 2 to move the motor to the Home position.
Command 2

<table>
<thead>
<tr>
<th>E0</th>
<th>E1</th>
<th>E2</th>
<th>During motion</th>
<th>EEPROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

**Action**: This command sets a certain position pulse number that the command "3", "4" or "5" uses during its execution. The specified value is shared between the motor 1 and 2.

**Format**

(I) \( \$ \ B\# \ 2 \ p \ p \ . \)  

- Position pulse number
- 0 to 16,777,215: unsigned number up to 8 digits  
- or \(-8,388,608 \) to \(+8,388,607\): signed number up to 7 digits

【DEFAULT=0】

**Note**: To change the data memory position range, use the command “EP”. When mode=1 and commend “EM”=1, the data is the number attaching the ± sign.

(II) \( \$ \ B\# \ 2 \)  
The current position will be stored as position pulse number that is used by the command “3”, “4” and “5” if the parameter pp is not specified. This format is meaningless if a motor is in motion. See the note below.

(III) \( \$ \ B\# \ 2 \ D \)  
Queries the current position pulse number specified by the command “2”.

**Reply**: In case of (I, II) >

In case of (III)

Under a uni-directional position management (i.e. EP=0 and 0 to 16,777,215)

> \( \$ \ B\# \)  
Decimal number with 8 digits  
The position pulse number currently stored

Under a bi-directional position management (i.e. EP=1, -8,388,608 to +8,388,607)

> \( \$ \ B\# \)  
Signed decimal number with 7 digits  
The position pulse number currently stored

**Example**

- `PRINT #1,"$123000";CHR$(&HD);`  
  Sets the position pulse number (pp) to 3,000 in the RC-234 with body number 1.

- `PRINT #1,"$12";CHR$(&HD);`  
  Sets the position pulse number (pp) to the current position pulse in the RC-234 with body number 1.
Action: This command moves the motor, at the high speed determined by "OH" and "OX" with ramping acceleration/deceleration, to the position specified by the command "2". (Absolute Move)

Format:

(I) \$ B# 3 MT
Moves a motor to the position specified by the command "2".

(II) \$ B# 3 MT
Moves a specified motor to the position specified by the command "2".
MT.. 1 or 2
3: simultaneous start (When command “EM”=1)

Reply: \>
(For both I and II)

Example

$122 0000: CHR$(&HD);
$131000: CHR$(&HD);

The first line sets the position pulse number to 2,000 in the RC-234 with body number 1.
The second line moves the motor 1 of the device to the specified position (2,000).

Example of absolute movement>

The direction of rotation varies based on the relation between a start and destination position. See the note below.

Note

- In execution of an absolute movement, if a start position pulse is smaller than a pulse number specified by the command "2", the motor will begin a CW rotation, otherwise CCW.
Command 4

Action: This command incrementally moves a motor a certain distance in the CW direction at a high speed with ramping acceleration/deceleration. The distance is specified by the command "2" and the speed and ramp are determined by "OH", "OL", "OX", and "OS". (Incremental Move)

Format: (I) $ B# 4 Rotates a motor in the CW direction by the distance set by the command "2".

(II) $ B# 4 MT Rotates a specified motor in the CW direction by the distance set by the command "2".
M T . . 1 or 2
3: simultaneous start (When command “EM”=1)

Reply: > (For both I and II)

Example
- PRINT #1,"$ 2 2 3 0 0 0"; CHR$(&HD);
- PRINT #1,"$ 2 4 1"; CHR$(&HD);

The first line sets the position pulse number to 3,000 in the RC-234 with body number 2. The second line rotates the motor 1 of the device by the specified pulses (i.e. 3,000) in the CW direction.

Example of incremental movement (CW direction)>

[Diagram showing incremental movement from Home position to CW direction with 3,000 pulses]
Action: This command incrementally moves a motor a certain distance in the CCW direction at a high speed with ramping acceleration/deceleration. The distance is specified by the command "2" and the speed and ramp are determined by "OH", "OL", "OX", and "OS". (Incremental Move)

Format: (I) $ B# 5  
Rotates a motor in the CCW direction by the distance set by the command "2".

(II) $ B# 5 MT  
Rotates a specified motor in the CCW direction by the distance set by the command "2".
MT: 1 or 2
3: simultaneous start (When command “EM”=1)

Reply: > (For both I and II)

Example:
PRINT #1, " $ 1 2 2 0 0 0" ; CHR$(&HD) ;  
PRINT #1, " $ 1 5" ; CHR$(&HD) ;

The first line sets the position pulse number to 2,000 in the RC-234 with body number 1. The second line rotates the motor 1 of the device by the specified pulses (i.e. 2,000) in the CCW direction.

<Example of incremental movement (CCW direction)>

Starting position

Home position

2,000 pulses
Command 6

<table>
<thead>
<tr>
<th>Command</th>
<th>E0</th>
<th>E1</th>
<th>E2</th>
<th>During</th>
<th>EEPROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>⬜</td>
<td>⬜</td>
<td>⬜</td>
<td>⬜</td>
<td>⬜</td>
</tr>
</tbody>
</table>

Action: This command queries the current position counter. The reply is an 8-digit decimal number.

Format:

(Ⅰ) $ B# 6
Queries the current position.

(Ⅱ) $ B# 6 MT
Queries the current position of a specified motor.
MT.. 1 or 2

Reply: (For both I and II)

Under a uni-directional position management (i.e. EP=0 and 0 to 16,777,215)

$ B# 00000000
Decimal number with 8 digits
Current position

Under a bi-directional position management (i.e. EP=1, -8,388,608 to +8,388,607)

$ B# +0000000
Signed decimal number with 7 digits
Current position

Note: To change the data memory position range, please perform the command “EP”.
When mode=1 and command “EM”=1, the data is the number attaching the ± sign.

Example

\[ \text{PRINT \#1, \"$16\"; CHR$(&HDA);} \]

Reply $10000350CR (0DH)

The first line queries the RC-234 with body number 1 to obtain the current position.
The second line is its response reporting that the current position is 3,500.
10. Command Reference

**Command 7**

<table>
<thead>
<tr>
<th>Action</th>
<th>This command rotates a motor in the CW direction at the low speed. The speed is determined by &quot;OL&quot; and &quot;OX&quot;. During the rotation, an &quot;H&quot; or &quot;L&quot; command allows the motor to change its speed to the high or low speed, respectively. The high speed is determined by &quot;OH&quot; and &quot;OX&quot;.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format</td>
<td>(I) $ B# 7 D Rotates a motor in the CW direction.</td>
</tr>
<tr>
<td></td>
<td>(II) $ B# 7 MT D Rotates a specified motor in the CW direction. MT = 1 or 2 3: simultaneous start (When command “EM”=1)</td>
</tr>
<tr>
<td>Reply</td>
<td>&gt; (For both I and II)</td>
</tr>
</tbody>
</table>

**Example**

```
PRINT #1, "$ 1 7"; CHR$(&HD):
```

This code instructs the RC-234 with body number 1 to rotate the motor in the CW direction.

**Note**

- Once a motor is started by this command, it keeps rotating until the CW sensor is activated. If you need to stop the motor manually, send either "S" or "SS" command.

**Command 7 ***

<table>
<thead>
<tr>
<th>Action</th>
<th>This command rotates the motor one pulse in the CW direction.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format</td>
<td>$ B# 7 * D</td>
</tr>
<tr>
<td>Reply</td>
<td>&gt;</td>
</tr>
</tbody>
</table>
10. Command Reference

**Command 8**

| Action | This command rotates a motor in the CCW direction at a low speed. The speed is determined by "OL" and "OX". During the rotation, an "H" or "L" command allows the motor to change its speed to the high or low speed, respectively. The high speed is determined by "OH" and "OX". |
| Format | (I) $ B# 8  ◯<br>Rotates a motor in the CCW direction. |
|        | (II) $ B# 8 MT  ◯<br>Rotates a specified motor in the CCW direction. MT.. 1 or 2 3: simultaneous start (When command “EM”=1) |
| Reply  | > (For both I and II) |

**Example**

- PRINT #1, "$ 18"; CHR$(&HD);$ 18"; CHR$(&HD);<br>This code instructs the RC-234 with body number 1 to rotate the motor in the CCW direction.

**Note**

- Once a motor is started by this command, it keeps rotating until the CCW sensor is activated. If you need to stop the motor manually, send either "S" or "SS" command.
**Command Reference**

<table>
<thead>
<tr>
<th>Command 9</th>
</tr>
</thead>
</table>

**E0** | **E1** | **E2** | **During motion** | **EEPROM** |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

**Action**: This command reads the condition flags in a RC-234.

**Format**:

(I)  

$ B# 9  

Reads all of the condition flags in a RC-234.

(II)  

$ B# 9 BT  

Reads a specified condition flag (bit0 to bit7) in a RC-234.

**Reply**: In case of (I)

$ B# DU DL  

DU and DL represent individual status bits in a hexadecimal notation. Each bit has a meaning as follows:

<table>
<thead>
<tr>
<th>Contents of DU</th>
<th>Binary data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target motor</td>
<td>bit7</td>
</tr>
<tr>
<td>Stall error</td>
<td>bit6</td>
</tr>
<tr>
<td></td>
<td>bit5</td>
</tr>
<tr>
<td></td>
<td>Mode</td>
</tr>
<tr>
<td></td>
<td>bit4</td>
</tr>
</tbody>
</table>

The most significant byte

<table>
<thead>
<tr>
<th>Contents of DL</th>
<th>Binary data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command error</td>
<td>bit3</td>
</tr>
<tr>
<td>Emergency stop</td>
<td>bit2</td>
</tr>
<tr>
<td></td>
<td>bit1</td>
</tr>
<tr>
<td></td>
<td>bit0</td>
</tr>
<tr>
<td></td>
<td>Communication error</td>
</tr>
<tr>
<td></td>
<td>Limit error</td>
</tr>
</tbody>
</table>

The least significant byte

- **Relation between HEX(Hexadecimal) and Binary**

<table>
<thead>
<tr>
<th>HEX</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>⬇️</td>
<td>⬇️</td>
<td>⬇️</td>
<td>⬇️</td>
<td>⬇️</td>
<td>⬇️</td>
<td>⬇️</td>
<td>⬇️</td>
<td>⬇️</td>
<td>⬇️</td>
<td>⬇️</td>
<td>⬇️</td>
<td>⬇️</td>
<td>⬇️</td>
<td>⬇️</td>
<td></td>
</tr>
<tr>
<td>bit3 (bit7)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>bit2 (bit6)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>bit1 (bit5)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>bit0 (bit4)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Command Reference

10. Command Reference

<Explanation of individual bits>

- Communication error (bit 0)
  bit0 = 1 means that a RC-234 has detected a communication error because of noise and/or data overflow in communication.
  bit0 = 0 means that the communication has been normally performed.

- Limit error (bit 1)
  bit1 = 1 means that a RC-234 has detected a stall error or the activation of the CW(CCW) limit sensors or the software limit during a motor's rotation.
  Additionally, bit1 is set to 1 when a RC-234 has detected other error(s) in motor operation.
  See "Note-7" described below, for more information.
  bit1 = 0 means that the RC-234 has detected no sensor error.

- Emergency stop (bit 2)
  bit2 = 1 means that a RC-234 has detected the activation of the EMS input by the time it receives the command "9". Note that this bit does NOT provide the status at the time a command is executed.
  bit2 = 0 means that the RC-234 has detected no EMS input.

- Command error (bit 3)
  bit3 = 1 means that a RC-234 has received an ineffective command(s): (an) invalid command(s) with regard to format; or (a) motion command(s) while a motor is running.
  bit3 = 0 means that all of the received commands were valid.

- Stall error (bit 6)
  bit6 = 1 means that a RC-234 have detected a stall error.
  bit6 = 0 means that the RC-234 has detected no stall error.

- Target motor (bit 7)
  bit7 = 1 means that the motor that is controlled by subsequent commands is motor 2.
  bit7 = 0 means that the target motor is motor 1.
  * To switch a target motor, use the command “F”.

Bit4 and bit5 represent the operation mode of a RC-234:

In case of (I)

<Explanation of bit4 and bit5>

<table>
<thead>
<tr>
<th>b5</th>
<th>b4</th>
<th>Current Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Mode 0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Mode 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Mode 2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Unused</td>
</tr>
</tbody>
</table>

In case of (II) $B\# DT D T..$ Binary data representing an individual flag in a RC-234.

- DT = 0 or 1 -- BT in the command format (II) is 0, 1, 2, 3, 6, or 7.
  For information on each flag, see "Explanation of individual bits" described above.
- DT = 0 to 2 -- BT is 4 or 5.
  The meaning of the value is described in the following table.

<Explanation of DT>

<table>
<thead>
<tr>
<th>DT</th>
<th>Current mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Mode 0</td>
</tr>
<tr>
<td>1</td>
<td>Mode 1</td>
</tr>
<tr>
<td>2</td>
<td>Mode 2</td>
</tr>
</tbody>
</table>

Note: To specify an operation mode, use the command "E".
Command 9

Example

\[ \text{PRINT } \#1, "\text{19}"; \text{CHR$}\ (&\text{HD}); \]

The command reads all of condition flags in the RC-234 with body number 1.

Reply Sample: (I)

> $1 0 0

No error has been detected.
The current operation mode is 0 and the target motor is motor 1.

HEX data 0 0 → Binary data 0 0 0 0, 0 0 0 0
b7 b6 b5 b4 b3 b2 b1 b0
0 0

Reply Sample: (II)

> $1 9 A

A command and sensor error has been detected.
The current operation mode is 1 and the target motor is motor 2.

HEX data 9 A → Binary data 1 0 0 0, 1 0 1 0
b7 b6 b5 b4 b3 b2 b1 b0
0 9

\[ \text{PRINT } \#1, "\text{194}"; \text{CHR$}\ (&\text{HD}); \]

Reads the current operation mode of a RC-234, using the format (II) with specified bit4.

Reply Sample:

> $1 2

The current operation mode is 2.

\[ \text{PRINT } \#1, "\text{296}"; \text{CHR$}\ (&\text{HD}); \]

Reads the status of a stall error in the RC-234 with body number 2. If the replied DT is 1, it means the RC-234 has detected a stall error.

Note

1. When the command "9" is executed in the format (I), a RC-234 sets each value of bit0, 1, 2, and 3 to 0 after querying the status of condition flags.

2. The command "9" execution in the format (II) does not affect the status flags and the flags will remain intact after querying the status of condition flags.

3. As mentioned in Note-1 and 2, the status flags remain intact until the format (I) is executed. Note that the command immediately preceding the command "9" may have not caused an error, even if the reply to the "9" command shows an occurrence of an error.

4. The command "(NULL)" can be also used to read error status such as a command error. The status flags for command "(NULL)" and "9" do not share a stored area. Therefore an execution of command "(NULL)" will clear the flags in its area but will not clear the flags for command "9".
⑤ Bit6 (stall error) is not cleared by the command "9" in any format. The flag will be automatically cleared upon an execution and completion of the command "0" (Mechanical home position search). When the RC-234 is in mode 1 (Encoder control), an execution of the command "RD" can force the bit6 to be set to 0.

⑥ Bit1 data is shared between a sensor and stall error. Therefore single data of bit1 is not sufficient to determine which is the cause of an error, a limit or stall sensor. To determine which error has detected, check bit6. For example, if bit1 is 1 and bit6 is 0, the cause of the error is a limit sensor. As the example shows, checking a correct limit sensor error requires to confirm that bit6 is 0 before performing command "9".

⑦ Each status flag for an sensor error of command "9" and "(NULL)" is set to 1 in the following cases as well as in the case of activation of a limit sensor:

- If a motion command is executed when bit6 of command "9" equals to 1, the status flag of an sensor error is set to 1. (except home-search-related commands such as command "0")
- If the absolute value of the difference between the pulse counter and the encoder pulse becomes 80 pulses or greater while a position-adjustment function is performed by the command "QJ" in mode 1 (encoder operation), the status flag of a sensor error is set to 1 upon completion of the move of 800 pulses.
- If a move beyond the following limitation is attempted in operation mode 1 by a absolute-movement command such as "3" or "B", the status flag of a sensor error is set to 1.
  Position pulse counter × "PB" < 8,388,608
- If a INP (in-position) signal is not transferred from a servo drive to the RC-234 within 3 seconds after a move command such as "3" and "B" completes outputting driving pulses, the status flag of a sensor error is set to 1.
- If the software limit (command "SL") is turned ON.

⑧ In simultaneous 2-axis motion, the status of the limit error, the emergency stop and the stall error for motor 1 and motor 2 will be in logical OR. To query the cause of stop of each motor, please use the command "9D".
Command Reference

**Command 9 ALL**

**Action**: This command queries the status of status flags, condition flags, general inputs/outputs and sensors all together.

**Format**: $ B# 9 A L L$

**Reply**: $ B# S T C D I N O U T S 1 S 2$

- **S T ..** The status of status flags
- **2-digit hex**
  - (Defaults of the upper 4 bits are 0.)
  - Refer to the command “” (NULL).
- **C D ..** The status of condition flags
- **2-digit hex**
  - Refer to the command “9”.
- **I N ..** The status of general inputs
- **2-digit hex**
  - Refer to the command “C”.
- **O U T ..** The status of general outputs
- **2-digit hex**
  - Refer to the command “CO”.
- **S 1 ..** The status of sensors (motor 1)
- **2-digit hex**
  - Refer to the command “CL”.
- **S 2 ..** The status of sensors (motor 2)
- **2-digit hex**
  - Refer to the command “CL”.

**Example**

- PRINT #1, “$19ALL”; CHR$(&HD);

  This code instructs the RC-234 with body number 1 to query the status all together.

**Note**

- Upon replying after executing this command, the status of command “” (NULL) and command “9”’s error bits (not including stall errors) are cleared. (See the command “” (NULL) and the command “9”.

- The status indication LED ERR will light when either the limit error, the emergency stop or the command error occurs and turn off the light when the command “” (NULL), “9ALL” (error status query), a home search command, such as “0”, “0A”, “0H”, “0Q”, “0R”, “0Z” or the user command “/EC” is performed.
Command 9 D

Action: This command queries the status and the cause of stop of each motor.

Format:
```
$ B# 9 D MT 
```
MT... 1 or 2
1: Queries the status and the cause of stop of motor 1.
2: Queries the status and the cause of stop of motor 2.
Not specified: Queries the status for the motor selected by the command “F”.

Reply:
```
> $ B# DU DL 
```
DU, DL... DU and DL represent individual status bits in a hexadecimal notation.

Description
Each bit has a meaning as follows:

<table>
<thead>
<tr>
<th>DU</th>
<th>DL</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>Status</td>
</tr>
<tr>
<td>bit 6</td>
<td>Default: 0</td>
</tr>
<tr>
<td>bit 5</td>
<td>Synchronous error (※ 1)</td>
</tr>
<tr>
<td>bit 4</td>
<td>Stop command</td>
</tr>
<tr>
<td>bit 3</td>
<td>EMS Error</td>
</tr>
<tr>
<td>bit 2</td>
<td>CW limit error</td>
</tr>
<tr>
<td>bit 1</td>
<td>CCW limit error</td>
</tr>
<tr>
<td>bit 0</td>
<td>Stall error</td>
</tr>
</tbody>
</table>

(※ 1: Errors caused by one axis stopping during synchronized motion)

- The error stop causing bits 0 to 5 are cleared when the move command is executed.

Note
- The status flag for the limit error is set to 1 when the software limit (command “SL”) is activated.
  To query the status of detection (soft ware or hard ware), use command “CL”.


Command A

Action: This command sets position pulse number on the position number (up to 1,000 points) for a command such as "B", "B+-(-)", etc.

If position pulse number is not specified and this command is executed, the current position is stored as the position pulse number.

The specified value is shared between the motor 1 and 2.

(Common data for motor 1 and 2)

Format:

(I) $ B# A p n pp D

Sets pp (position pulse data) on pn (position number).

pn: Position number (000 to 999) up to 1,000 points

Sets by 3-digit number assuredly.

pp: Position pulse number
0 to 16,777,215: unsigned number up to 8 digits
or
-8,388,608 to +8,388,607: signed number up to 7 digits

【DEFAULT=0】

Note: To change the data memory position range, please use the command “EP”. In mode 1, the data will be the signed number.

(II) $ B# A p n D

The current position will be stored as position pulse number of pn if a parameter pp is not specified.

pn: Position number (000 to 999) up to 1,000 points

Set by 3-digit number assuredly.

(III) $ B# A p n D

Reads the position pulse number of position number (pn) set by the command “A” and receives decimal number with 8 digits replay from RC-234.

pn: Position number (000 to 999) up to 1,000 points
Set by 3-digit number assuredly.

Reply:

In case of (I, II) 

In case of (III)

Under a uni-directional position management (i.e. EP=0 and 0 to 16,777,215)

$ B# 00000000 D

Decimal number with 8 digits

The position pulse number currently stored

Under a bi-directional position management (i.e. EP=1, -8,388,608 to +8,388,607)

$ B# 0000000 D

Signed decimal number with 7 digits

The position pulse number currently stored
Example

- PRINT #1, "$1A 00 2 3 0 0 0" ; CHR$(&HD);
  Sets the position pulse number of pn(position number)= 002 in the RC-234 with body number 1 to 3,000.

- PRINT #1, "$1A 02 2" ; CHR$(&H D);
  Sets the position pulse number of pn=022 to the current position pulse in the RC-234 with body number 1.

Note

- When the position pulse number is set by signed number, the position data memory is -8,388,608 to +8,388,607.

- The position data set by the command “A” is shared by the command “MA”.
Action
This command reads out the position pulse number specified by the command “A” and command “MA” and stored to EEPROM by the command “AW” to RAM. (The command “B” and the command “MB” refer the data for the command “A” and the command “MA” stored to RAM.)

Format
$ B# A L  

Reply
>

Description
① Automatic execution of the command “AL” accompanied by power-supply ON
When the data is stored in EEPROM by the command “AW”, the command “AL” is executed automatically as soon as RC-234 is powered. Therefore the default of command “A” will be ignored.

② Overwrite the data in the area of RAM
If Command “AL” is issued without executing Command “AW”, the default of RC-234 is copied in RAM.

Either executing Command “AL” or resetting the power supply will make the current data in RAM abandon and change it to the same setting as the data in EEPROM.

③ Erasing the data in EEPROM
The data written in EEPROM will not be erased until executing command “EE///”.
(Overwriting is possible.)
Note: After execution of Command “EE///”, all data stored in EEPROM by the command “AW”, “DW”, “IW” will be erased simultaneously.

Note
・ It takes longer to reply than other commands.

Action
This command initializes the current position data of command “A” and command “MA” stored in RAM. (The data in EEPROM isn’t rewritten.)

Format
$ B# A L *  

Reply
>
**10. Command Reference**

<table>
<thead>
<tr>
<th>Command</th>
<th>AW</th>
</tr>
</thead>
</table>

**Action**: This command stores the position data specified by the command “A” and the command “MA” in EEPROM. (Even if the power is turned OFF, this data is maintained.)

**Format**: $B\# A W$
Stores the data specified by the command “A” in EEPROM.

**Reply**: > $ B\# *$
When receiving the command “AW”, RC-234 replies [>], and upon the completion of storing the data in EEPROM, it replies [$ B\# * ].

**Note**
- If the command “AW” is executed and the position data is reset by the command “A” and the command “MA” and then the command “AW” is executed again, the reset position data is overwritten in EEPROM.
- It takes a maximum of 3 seconds to write the data.
- Please see the description in Command “AL.”
**Command B**

**Action**: This command moves a motor to the position specified by the command “A”.
(Absolute movement)
As for the meaning of the absolute movement, see the command “3”.

**Format**

(I) $ B# B pn
Moves a motor to the position of pn specified by the command “A”:
p n .. Position number (000 to 999) up to 1,000 points
Sets by 3-digit number assuredly.

(II) $ B# B pn MT
Moves a specified motor to the position of pn specified by the command “A”:
p n .. Position number (000 to 999) up to 1,000 points
Sets by 3-digit number assuredly.
MT .. 1 or 2
3: simultaneous start (When command “EM”=1)

(III) $ B# B
Reads the pp of subsequent pn of the last move pn by the command “B” and
moves a motor to the position.

(IV) $ B# B MT
Reads the pp of subsequent pn of the last move pn by the command “B” and
moves a specified motor to the position.
MT .. 1 or 2
3: simultaneous start (When command “EM”=1)

(V) $ B# B N
Queries the position number of Command “B” to move subsequently.

**Reply**: In case of (I) to (IV) $ B# B pn

In case of (V) $ B# B# 0 0 O O O O
decimal 5 digits
【DEFAULT=00000】
Current position pulse number

**Note**: The position number is 3 digits long, but the reply will be returned with a 5-digit decimal number.
10. Command Reference

Command B

Description
- Format (Ⅲ) and (Ⅳ) are used when repeating movement to the pp position in order of position number of command “A”.

We’ll show you the examples repeating sequential movement from position number 100.

① First, move to the pp position of pn 100.
  $1B100$  Format (Ⅰ)

② Second, move to the pp position of pn 101.
  $1B101$  Format (Ⅲ)  (The same action as “$1B101”)

③ Move to the pp position of subsequent position number.
  $1B102$  Format (Ⅲ)  (The same action as “$1B102”)

④ Query the position number to move subsequently.
  $1B303$  Format (Ⅴ)

  $10010103$  Next position number is 103.

As the above, sequential movement is performed by executing format (Ⅲ) or (Ⅳ).

Example
- PRINT #1,”$1B005”;CHR$(&HD);
  This code instructs RC-234 with body number 1 to move to the pp position of position number 005 set by the command “A”.

Note
- If command “B” is executed by the position number which isn’t specified by the command “A”, it doesn’t move a motor to the home position because default of position pulse number of each position number is 0.

- The position number called out by the format (Ⅲ) or (Ⅳ) will be changed by the command “B+(-)” as well as command “B”.

- When format (Ⅲ) or (Ⅳ) is executed without executing format (Ⅰ) or command “B+(-)” after RC-234 is turned ON, the motor moves to the pp position of position number 001.

- In simultaneous 2-axis control, the two motors move to the same position. To move them to the different positions, use the command “MB”.

B-47
Action: This command incrementally moves a motor a certain distance in CW (in case of +) or CCW (in case of -) relative to current position. The distance is specified by the command “A”.

As for the meaning of incremental movement, please see the command “4” or “5”.

Format: (I)

$ B# B p n +  

This command incrementally moves a motor a certain distance in CW (in case of +) or CCW (in case of -) relative to current position.
The distance is specified by the command “A”.

p n . . Position number (000 to 999) up to 1,000 points
Sets by 3-digit number assuredly.

(II)

$ B# B p n MT +  

This command incrementally moves a specified motor a certain distance in CW (in case of +) or CCW (in case of -) relative to current position.
The distance is specified by the command “A”.

p n . . Position number (000 to 999) up to 1,000 points
Sets by 3-digit number assuredly.

MT . . 1 or 2
3: simultaneous start (When command “EM”=1)

(III)

$ B# B +  

Reads the position pulse number of subsequent pn of the last move pn by command “B+(-)” and moves a motor by the position data in CW (in case of +) or CCW (in case of -).

(IV)

$ B# B MT +  

Reads the position pulse number of subsequent pn of the last move pn by Command “B+(-)” and moves a specified motor by the position data in CW (in case of +) or CCW (in case of -).

MT . . 1 or 2
3: simultaneous start (When command “EM”=1)

Reply: > (for all, I, II, III and IV)
Description

- Format (Ⅲ) and (Ⅳ) are used when repeating incremental movement in order of position number of command “A”.

We’ll show you the examples repeating sequential movement from position number 020 in CW(in case of +) and CCW(in case of -) by turns.

① First, move in CW direction by the position data of the position number 020.

$1B020+$  Format (Ⅰ)

② Second, move in CCW direction by the position data of the position number 021.

$1B-$  Format  (Ⅲ)

③ Move in CW direction by the position data of the next position number.

$1B+$  Format  (Ⅲ)

Example

- PRINT #1, ”$1B0112+”; CHR$(&HD): This code instructs a motor 2 of RC-234 with body number 1 to move by position data of position number 011 set by the command “A” in the CW direction.

Note

- If the command “B+(−)” is executed by position number which isn’t set by the command “A”, the motor doesn’t move because the default of position pulse number of each position number is 0.

- The position number called out by the format (Ⅲ) and (Ⅳ) will be changed by the command “B” as well as the command “B+(−)”. Please query the current position number by the command “BN”.

- If executing format (Ⅲ) and (Ⅳ) without executing format (Ⅰ) or the command “B” after RC-234 is turned ON, RC-234 refers to the position pulse number of number 001 and moves a motor by the data.

- Perform the incremental movement by the pulse number of “absolute value” of position pulse number set by the command “A”.

  For example, whichever the position pulse number is set by the command “A001” to +5,000 or −5,000, if the command “B001+” is executed, the motor moves to the CW direction by 5,000 pulses and if the command “B001−” is executed the motor moves to the CCW direction by 5,000 pulses.

- In simultaneous 2-axis control, the two motors moves by the same distance. To move them by the different distance, use the command “MB+”
Action: This command reports the signal status of general inputs and the status of INP1,2 input bits.

Format:

(I) \$ B# C
Queries the status of general input ports (Signal input is ON or OFF) all together.

(II) $ B# C BT
Queries the status of inputs bits. Also, can inquire the status of INP1,2.
B... bit number (0 to 9)
T... bit number (0 to 9)
0 to 7...general inputs
8, 9...INP1, 2

(III) $ B# C A
Can query the status of general inputs, INP1,2 all together
(Note: This format can't be used when either motor 1 or motor 2 is in mode 2.)

Reply:

In case of (I) $ B# D U D L
DU, DL... Hexadecimal values (0 to F) representing the upper and the lower 4 bits of a specified port
(As for the relation between hex and binary, please see the next page.)

In case of (II) $ B# DT
DT... A binary value representing the data of a specified bit
DT=1: bit ON, DT=0: bit OFF

In case of (III) $ B# IP DU DL
IP... A 2-bit hexadecimal value representing INP1 and INP2 status.
(See the below table.)
DU, DL... Hexadecimal values (0 to F) representing the upper and the lower 4 bits of the general-purpose port.

<table>
<thead>
<tr>
<th>IP(HEX)</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>INP1</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>INP2</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
</tr>
</tbody>
</table>
10. Command Reference

B-51

Command C

- Relation between HEX (Hexadecimal) and Binary

<table>
<thead>
<tr>
<th>HEX</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
</tr>
</tbody>
</table>

- bit3 (bit7) 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1
- bit2 (bit6) 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
- bit1 (bit5) 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
- bit0 (bit4) 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1

Note: bit0 to bit7 is equivalent to 13 to 20 pin respectively.

Example

- PRINT #1,"$1C";CHR$(&HDF;

This code queries the data of the genera-purpose input port to RC-234 with body number 1.

<table>
<thead>
<tr>
<th>Reply Sample :</th>
<th>&gt;</th>
<th>$</th>
<th>1</th>
<th>2</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Note: between bit0 to bit7, bit0 to bit3 and bit5 are activated.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HEX data 2 F</th>
<th>→</th>
<th>Binary data</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 F</td>
<td></td>
<td>0 1 0 1 1 0 1</td>
</tr>
</tbody>
</table>

Note

- INP1 and INP2 are configured as general inputs in operation mode E0 and E1. In an operation mode 2, please use the command “CI” to query the status of INP1, 2 inputs bits.
Action: This command queries the status of INP1 and INP2 inputs bits. The command is exclusively used in an operation mode 2.

Format:

(I) $ B# C I ≡
Queries the status of INP1 and INP2 inputs bits.

(II) $ B# C I MT ≡
Queries the status of INP1 and INP2 inputs bits of a specified motor. MT.. 1 or 2

Reply:

> $ B# DT ≡ (For both I and II)
D T.. 0 INP signal is ON. (Motor is running.)
1 INP signal is OFF.

Note) If DT were 0, the actual signal level (High or Low) depends on the input logic set by the command “EA”.

Description:

- If the data queried by the command “Cl” is 1 when a servo motor completes the rotation in mode 2, it means a normal end.
  If the data remains 0 more than 3 seconds, a limit error will occur.

Note:

- INP(in position) terminal is connected to position complete outputs of a driver for servo motor.
- We presume that the reason the action will be completed without inputting INP signal to INP terminal is that the input logic of INP becomes high active and INP remains Low level,
  (To set input logic use the command “EA”.)
- Execute the command “C” to confirm the status of INP input when INP terminals are used as general inputs in mode 0 or 1.
Command Reference

Action:
This command reports the input status of each sensor such as home (ORG) sensor, limit sensor, etc. for the motor specified by the command “F” to PC.

Format:
(Ⅰ) $B#CL
Queries the status of each sensor such as home (ORG) sensor, Limit sensor. (Whether signal input is ON or OFF.)

(Ⅱ) $B#CLBT
Queries the status of a specified bit of each sensor such as home (ORG) sensor, Limit sensor.
B T . . Bit number of each sensor (0 to 7)
As for the bit number of sensor, see the list in the description.

(Ⅲ) $B#CLM\ TM
Queries the status of each sensor such as home (ORG) sensor, Limit sensor of a specified motor.
M T . . 1 or 2

Reply:
In case of (Ⅰ, Ⅲ) $B#DU DL
DU, DL... Hexadecimal values (0 to F) representing the upper and the lower 4 bits of each sensor

In case of (Ⅱ) $B#DT
DT... A binary value representing the data of a specified sensor

Description:
- Each bit represents the status of sensors as the following list.

(Bit Assignment>

<table>
<thead>
<tr>
<th>DU</th>
<th>DL</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit4</td>
<td>Home (ORG) sensor</td>
</tr>
<tr>
<td>bit5</td>
<td>Emergency stop signal</td>
</tr>
<tr>
<td>bit6</td>
<td>Stall Detection sensor</td>
</tr>
<tr>
<td>bit7</td>
<td>Not used (Default: 0)</td>
</tr>
</tbody>
</table>
Command CL

<Sensor Status>

<table>
<thead>
<tr>
<th>D U (in Hex data)</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home (ORG) sensor</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>Emergency stop signal</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>Stall Detection sensor</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D L (in Hex data)</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>CW LS</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>CCW LS</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
</tr>
</tbody>
</table>

Example

\* PRINT #1, "$1 CL" ; CHR$(&HD) ;
This code queries the status of the sensor in RC-234 with Body number 1.

Reply Sample : ▶ $ 1 2 1 u
This reply means that CW LS (bit0) and emergency stop signal (bit5) is turned ON.

HE X data 2 1 → Binary data

DU

DL

b7 b6 b5 b4 b3 b2 b1 b0
2 1

\* PRINT #1, "$1 CL 4" ; CHR$(&HD) ;
This code queries the status of home (ORG) sensor in RC-234 with body number 1.

Reply Sample : ▶ $ 1 0 u
DT is 0 means that home (ORG) sensor is turned OFF.
DT is 1 means that home (ORG) sensor is turned ON.
Action: This command reports the signal status of general outputs and the status of CLR1, CLR2.

Format: (I) $B# C O
Queries the status of general output bits (whether signal output is ON or OFF).

(II) $B# C O A
Queries the status of general output bits and CLR1, CLR2.
(Note: This format can't be used when either motor 1 or motor 2 is in mode 2.)

Reply: In case of (I) $B# DU DL
DU, DL... Hexadecimal values (0 to F) representing the upper and the lower 4 bits of general outputs.
(As for the relation between hex and binary, see "Relation between HEX(Hexadecimal) and Binary" in the command "D".)

In case of (II) $B# DC DU DL
DC... Hexadecimal value (0 to 3) representing the status of CLR1, CLR2.
(See the following list.)
DU, DL... Hexadecimal values (0 to F) representing the upper and the lower 4 bits of general outputs

<table>
<thead>
<tr>
<th>D C (HEX)</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR1</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>CLR2</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
</tr>
</tbody>
</table>

Note:
- Format (II) is effective to query the status of CLR terminal when CLR(counter clear) terminals are used as general outputs.
- The format (II) can't be used in mode 2 (servo motor control). CLR terminals can't be used as general outputs, because in mode 2 CLR terminals are used as counter clear outputs of servo motor. Use the command “DS” to confirm the status of CLR terminals in mode 2.
- See the note in the command “D”.
Action: This command enables the function to use the general input D0 and D1 as interruption inputs. One interruption input is available per general input.

Format: (I) $ B\# C Y IBT MD . P MT p n$

This command enables the interruption input and stores the position pulse to the specified position number of command “A”/command “MA” (only when EM=1) when the interruption of a specified bit has occurred. 【Default: Disable】

IBT . interruption input bit (0: D0, 1: D1)
M D . interruption mode 1: Enables
2: Disables once the interruption is executed.
M T . motor number (1, 2, 3 (only when EM=1))
M T = 1, 2 : stored as the position of command “A”
M T = 3 : stored as the position of command “MA”
p n . position number (000 to 999) up to 1,000 points
Sets by 3-digit number assuredly.

(II) $ B\# C Y IBT MD . O OBT DT$

This command enables the interruption input and turns the specified general output ON/OFF when the interruption of a specified bit has occurred.
【Default: Disable】

IBT . interruption input bit (0: D0, 1: D1)
M D . interruption mode 1: Enables
2: Disables once the interruption is executed.
O B T . general output bit (0 to 7)
D T . output setting (0: OFF, 1: ON)

(III) $ B\# C Y IBT MD . S ST MT$

Enables the interruption input and stops the designated motor when the interruption of a specified bit has occurred. 【Default: Disable】

IBT . interruption input bit (0: D0, 1: D1)
M D . Interruption mode 1: Enables
2: Disables once the interruption is executed.
S T . stop method (0: Immediate stop, 1: Stop with ramping deceleration)
M T . motor number 1, 2
3: Stops both axes (only when EM=1)

(IV) $ B\# C Y IBT \emptyset$

Disables the interruption input.

IBT . interruption input bit (0: D0, 1: D1)

(V) $ B\# C Y L IBT DT$

Sets the interruption edge.

IBT . interruption input bit (0: D0, 1: D1)
D T . interruption input logic
0: falling edge 【Default】
(The status of the data acquired by the command “C” is changed 1 to 0.)
1: rising edge
(The status of the data acquired by the command “C” is changed 0 to 1.)
(Ⅵ) \$ B# C Y P IBT MT CCW IA CW IB

Sets the data memory position range that enables the interruption input.

IBT. interruption input bit (0, 1)
MT. motor number 1, 2
CCW. data memory position range in the CCW direction
  -8,388,608 to 16,777,215
CW. data memory position range in the CW direction
  -8,388,608 to 16,777,215

When both CW and CCW is 0, the interruption is enabled regardless the position.
【Default】

(Ⅶ) \$ B# C Y N IBT

Sets the format (Ⅰ) to (Ⅲ) and queries the number of times of interruption.
When the format (Ⅰ) to (Ⅲ) is set, the number of times is cleared.

IBT. interruption input bit (0, 1)

(Ⅷ) \$ B# C Y IBT

Queries the status of “interruption input”.

IBT. interruption input bit (0, 1)

(Ⅸ) \$ B# C Y L IBT

Queries the interruption edge. (Rising edge or falling edge)

IBT. interruption input bit (0, 1)

(Ⅹ) \$ B# C Y P IBT

Queries the data memory position range that enables the interruption input.

IBT. interruption input bit (0, 1)
Reply : In case of (I to VI) 

In case of (VII)

$ \text{B# CT CT..}$

- Number of times of interruption
  - 3-digit decimal number (000 to 255)
  - When it exceeds 255, it returns to 000.

In case of (VIII)

- In case of format (I)
  $ > \text{B# MD P MT pn}$

- In case of format (II)
  $ > \text{B# MD O OBT DT}$

- In case of format (III)
  $ > \text{B# MD S ST MT}$

$ \text{B# 0}$

Reply when the interruption is disabled.

MD.. Interruption mode
- 0: Disables
- 1: Enables
- 2: Disables once the interruption is executed.

MT.. Motor number 1, 2, 3

OBT.. General output bit (0 to 7)

ST.. Stop method
- 0: Immediate stop,
- 1: Stop with ramping deceleration

pn.. position number (000 to 999)

DT.. Output setting (0: OFF, 1:ON)

In case of (IX)

$ > \text{DT DT..}$

DT.. Interruption input logic
- 0: Interruption at a falling edge
- 1: Interruption at a rising edge

In case of (X)

$ > \text{B# MT CCW CW CW}$

MT.. Motor number 1, 2

CCW.. Data memory position range in the CCW direction
- Signed decimal number with 8 digits
  - (-08,388,608 to +16,777,215)

CW.. Data memory position range in the CW direction
- Signed decimal number with 8 digits
  - (-08,388,608 to +16,777,215)
Action: Sets the individual bits or whole output ports.

Format: (I)

$ B\# D DU DL $Δ$

Sets whole output ports ON or OFF simultaneously.

D U.. Sets the upper 4 bits (bit7 to bit4) in hexadecimal notation.
D L.. Sets the lower 4 bits (bit3 to bit0) in hexadecimal notation.

【DEFAULT=00】
bit data 1----Open collector output transistor is turned ON.
bit data 0----Open collector output transistor is turned OFF.

Format: (II)

$ B\# D BT DT B $Δ$

Sets specified bits individually.
Also, sets CLR1 and CLR2 output bits

B T.. bit no.(0 to 9)
0 to 7... general output ports
8, 9... CLR1, CLR2
(Note: When either motor 1 or motor 2 is in mode 2, bit8, 9 can’t be turned ON/OFF.)

D T.. bit data (0, 1)
DT=1...Open collector output transistor is turned ON.
DT=0...Open collector output transistor is turned OFF.

- Relation between HEX(Hexadecimal) and Binary

<table>
<thead>
<tr>
<th>HEX</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
</tr>
<tr>
<td>bit3 (bit7)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>bit2 (bit6)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>bit1 (bit5)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>bit0 (bit4)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Note: bit0 to bit7 is equivalent to 3 to 10 pin respectively.
Command D

D C . . Sets the status of CLR1, CLR2 output bits in Hexadecimal notation.
(See the below table.)
D U . . Sets the upper 4 bits (bit7 to bit4) in Hexadecimal notation.
D L . . Sets the lower 4 bits (bit3 to bit0) in Hexadecimal notation.

【DEFAULT=000】

<table>
<thead>
<tr>
<th>D C (HEX)</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR1</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>CLR2</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
</tr>
</tbody>
</table>

Reply : > (For I, II, III)

Example
- PRINT #1, "$1DE1"; CHR$(&HD();
This code sets the data for the general output port in RC-234 with body no. 1 to “E1”.
Then bit0 and bit 5 to 7 are set to “ON” and the remaining bits are set to “OFF”.
(As for the conversion of HEX data and binary data, see the table “Relation between HEX and binary”.)

<table>
<thead>
<tr>
<th>HEX data E1</th>
<th>Binary data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>D U</td>
</tr>
<tr>
<td></td>
<td>D L</td>
</tr>
<tr>
<td></td>
<td>b7 b6 b5 b4 b3 b2 b1 b0</td>
</tr>
<tr>
<td></td>
<td>E 1</td>
</tr>
</tbody>
</table>

Note
- The defaults for whole output ports are set to OFF (0).
- As for the general output ports, see Chapter 4 “Part Names and Functions”.
- Bit 8 and 9 in format (II) and format (III) are used to turn CLR outputs ON/OFF when CLR (counter clear) terminals are used as the general outputs.
- In mode 2 (Servo motor control), CLR can't be turned ON/OFF by the command “D”.
  Because CLR terminals are used as the counter clear output of servo motor in mode 2, they can't be used as the general outputs. Execute the command “DS” to set CLR outputs ON/OFF forcibly in mode 2.
Action : This command sets the time to output CLR signal (deviation counter clear), This command is used only in mode 2 (servo motor). The parameter should be separately specified for the motor 1 and 2.

<table>
<thead>
<tr>
<th>Format</th>
<th>$</th>
<th>$</th>
<th>B#</th>
<th>D</th>
<th>C</th>
<th>T</th>
<th>TM</th>
<th>□</th>
</tr>
</thead>
<tbody>
<tr>
<td>(I)</td>
<td>Sets the time to output CLR signal.</td>
<td>T M .</td>
<td>0</td>
<td>130 μ sec [DEFAULT]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>15msec</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>130msec</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>500msec</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>no output signal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(II) Queries the time it takes to output the current CLR signal

Reply : In case of (I) >

In case of (II) > $ $ B# TM □

T M . | 0 | 130 μ sec |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15msec</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>130msec</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>500msec</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>no output signal</td>
<td></td>
</tr>
</tbody>
</table>

Description
- Unless CLR is connected with the deviation counter clear of servo motor driver and turned ON, OFF forcibly using the command “DS”, CLR output is turned ON at the time set by the command “DCT”. in the case of the below. (Only in mode 2)

1. When the home search has been completed by home search command ("∅", "∅ R", etc.)
   - In case of Command "∅", "∅ A", "∅ H"
     When the motor runs in the CCW direction until the home (ORG) sensor is activated, keeps rotating the “∅ S” pulses farther and stops.
   - In case of Command "∅ R"
     When the motor runs in the CW direction until the home (ORG) sensor is activated, keeps rotating the “∅ S” pulses farther and stops.
   - In case of Command "∅ Z"
     When the motor runs in the CCW direction until the home (ORG) sensor is activated, RC-234 detects ON status for the number of the Z-phase input to be sensed, rotates the motor five pulses in the CCW direction and stops it.

2. When the limit sensor or the emergency stop signal is turned ON by the motion command during motion.
   (Except the home search command ("∅", "∅ R" etc.))
③. When the device received no ON signal from INP terminal for about 3 seconds after finishing the pulse output during command execution. (At the same time, each status flag of a limit error (bit1) of command "(NULL)" and "9" is set to 1.)

④. When the command “S” is executed during motion.

⑤. When the command “SS” is executed while the motor is running at Low speed.

⑥. When stall error has occurred during the stall detection.
### Command Reference

#### D L

**Action**: This command copies the data stored in EEPROM by the command “DW” to RAM. (The actions of RC-234 are executed based on the data in RAM.)

**Format**: $\text{B# D L } \star$

**Reply**: >

**Description**

1. **Auto-start of Command “DL” upon the activation of the power**
   - When the data is stored in EEPROM by the command “DW”, the command “DL” is executed automatically as soon as RC-234 is powered.
   - Therefore, the default of target commands will be ignored. (See the table in the command “DW”.)

2. **Overwriting the data in RAM area**
   - Before the command “DW” is executed to overwrite the data in RAM area, if the command “DL” is executed, the default value of RC-234 is copied to RAM.
   - If the command “DL” is executed or RC-234 is turned OFF and turned ON again, the current RAM data is abandoned and changed to the same setting as the data of EEPROM.

3. **Erasing the contents of the EEPROM**
   - The data written in EEPROM will not be erased until executing the command “EE///”. (Overwriting is possible.)
   - Note: When executing the command “EE///”, whole data written in EEPROM by the command “AW”, “DW” and “IW” will be erased at the same time.

**Note**

- It takes longer to reply than other commands.

#### D L *

**Action**: This command initializes the setting values of speed and functions written in RAM. The data of the specified command written into EEPROM with the command “DW” is used. (The contents in EEPROM aren’t changed.)

**Format**: $\text{B# D L } \star$

**Reply**: >
**Command Reference**

**Action**: This command sets CLR(counter clear) output terminal for the servo motor driver forcibly. This command is exclusively used in an operation mode 2.

**Format**:

(I) \( \$ \ B\# \ D \ S \ C2 \ C1 \)  
Sets the individual status of CLR1 and CLR2 (ON, OFF).
- \( C1 = 0 \) --- OFF  
- \( 1 \) --- ON

- \( C2 = 0 \) --- OFF  
- \( 1 \) --- ON

(II) \( \$ \ B\# \ D \ S \ R \)  
Turns OFF the status of CLR1 and CLR2.

(III) \( \$ \ B\# \ D \ S \)  
Queries the status of current CLR output ports (ON or OFF).

**Reply**: In case of (I, III) >

In case of (II) > \( \)  
\( C1 \) --- The status of CLR1  
- \( C1 = 0 \) --- OFF  
- \( 1 \) --- ON

- \( C2 = 0 \) --- OFF  
- \( 1 \) --- ON
**Action:** This command stores the command setting value to set speed, function, etc. in EEPROM.

**Format:** $ \mathrm{B\#} \ D \ W \ \#$

**Reply:** $ \mathrm{B\#} \ * \ \#$

When RC-234 received the command “DW”, the reply [>] is returned and when storing in EEPROM is completed, the reply [ $ \mathrm{B\#} \ * \ \#$] is returned.

**Description**

- If the command “DW” is executed, the following setting value is stored in EEPROM.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Ø B &quot;</td>
<td>Multiples of &quot;Ø S&quot; command data</td>
</tr>
<tr>
<td>&quot;Ø Q &quot;</td>
<td>Pulse during stall detection sensor ON</td>
</tr>
<tr>
<td>&quot;Ø S &quot;</td>
<td>Distance moved in steps after sensing the trailing edge of the home (ORG) sensor</td>
</tr>
<tr>
<td>&quot;Ø X &quot;</td>
<td>Home search Algorithm in mode 2</td>
</tr>
<tr>
<td>&quot;Ø Z &quot;</td>
<td>Number of EZ pulse after sensing home (ORG) sensor</td>
</tr>
<tr>
<td>&quot;2 &quot;</td>
<td>Motion counter</td>
</tr>
<tr>
<td>&quot;D C T &quot;</td>
<td>CLR output signal setting</td>
</tr>
<tr>
<td>&quot;E &quot;</td>
<td>Mode setting</td>
</tr>
<tr>
<td>&quot;E A &quot;</td>
<td>Sensor logic</td>
</tr>
<tr>
<td>&quot;E D &quot;</td>
<td>Clock logic 2P/P&amp;D</td>
</tr>
<tr>
<td>&quot;E E &quot;</td>
<td>Echo back</td>
</tr>
<tr>
<td>&quot;E L &quot;</td>
<td>Carriage return</td>
</tr>
<tr>
<td>&quot;E M &quot;</td>
<td>Alternate/simultaneous 2-axis control setting</td>
</tr>
<tr>
<td>&quot;E P &quot;</td>
<td>Pulse counting</td>
</tr>
<tr>
<td>&quot;E R &quot;</td>
<td>Error display</td>
</tr>
<tr>
<td>&quot;K &quot;</td>
<td>Error output</td>
</tr>
<tr>
<td>&quot;O C &quot;</td>
<td>S-curve data</td>
</tr>
<tr>
<td>&quot;O H &quot;</td>
<td>High speed data</td>
</tr>
<tr>
<td>&quot;O L &quot;</td>
<td>Low speed data</td>
</tr>
<tr>
<td>&quot;O S &quot;</td>
<td>Accel. Data</td>
</tr>
<tr>
<td>&quot;O X &quot;</td>
<td>Multiplication factor to determine the speed of a motor</td>
</tr>
<tr>
<td>&quot;P A &quot;</td>
<td>Encoder counting mode</td>
</tr>
<tr>
<td>&quot;P B &quot;</td>
<td>Encoder ratio</td>
</tr>
<tr>
<td>&quot;Q &quot;</td>
<td>Stall detection</td>
</tr>
<tr>
<td>&quot;Q E &quot;</td>
<td>Stall detection Maximum limit of the difference</td>
</tr>
<tr>
<td>&quot;Q J &quot;</td>
<td>Stall adjust</td>
</tr>
<tr>
<td>&quot;Q J A &quot;</td>
<td>Stall adjust pulse number (Max.)</td>
</tr>
<tr>
<td>&quot;Q J O &quot;</td>
<td>Stall adjust speed</td>
</tr>
<tr>
<td>&quot;Q J T &quot;</td>
<td>Stall adjust time</td>
</tr>
<tr>
<td>&quot;Q S &quot;</td>
<td>Stall detection checking</td>
</tr>
<tr>
<td>&quot;Q S S &quot;</td>
<td>Specify the target axis (detected axis/both axes) when a stall is detected.</td>
</tr>
<tr>
<td>&quot;S L &quot;</td>
<td>Software limit</td>
</tr>
<tr>
<td>&quot;S U M &quot;</td>
<td>Sum checking</td>
</tr>
</tbody>
</table>

**Note:** Common data for motor 1 and 2 is stored in the different area of EEPROM.
Command DW

Note

- If the command “DW” is executed, current setting values of the above target commands are written in EEPROM.

- If the data of target commands will be changed and the command “DW” is executed again after the command “DW” executed once, the reset data are overwritten in EEPROM.

- It’s possible to overwrite in EEPROM. If the command “DW” is executed, the current setting values of the target commands in RAM will be stored in EEPROM and the old data in EEPROM are abandoned.

- It takes a maximum of 3 seconds to write the data.

- See the description in the command “DL”.
Action: This command sets the mode of operating.
When command “EM”=0 (alternate 2-axis control), the setting is shared between the
motor 1 and 2. (Common data for motor 1 and 2).
When command “EM”=1 (simultaneous 2-axis control), this setting should be separately
designated for each motor.

Format: (I)  \[\$ B# E MN \]
Sets the mode no. for “Stall detection of stepping motor using a stall sensor,
Control using encoder, Using servo motor” etc.
MN.. Number representing the mode (0 to 2)
【DEFAULT=0】

(II)  \[\$ B# E \]
Queries the current mode number.

Reply: In case of (I) >

In case of (II)  \[\$ B# MN \]
MN.. Number representing the mode. (0 to 2)

Description
・ Mode has three patterns.
  For further information, see Chapter 6 “Mode and Driver Wiring method” or 13-2 ①“Commands for
each Mode and Sample Program”.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Can control stepping motor.</td>
</tr>
<tr>
<td>1</td>
<td>Can control stepping motor using encoder.</td>
</tr>
<tr>
<td>2</td>
<td>Can control servo motor.</td>
</tr>
</tbody>
</table>

Example
・ PRINT #1,”$ 1 E 1” ; CHR$(&H D) ;
  This code sets RC-234 with body no. 1 to the mode to control stepping motor using encoder.

Note
・ In mode 1, in spite of the setting of the command “EP”, the data is the number attaching the ± sign.
  Also, in mode 1, the position is managed by the encoder pulse.
  However, speed (command “OH”, “OL”, etc.) is set by the actual pulse speed output to the driver.
  (See Chapter 7 “Stepping Motor Speed Adjustment”.)
Action: This command sets the input logic (active high or active low) for limits, origin, EMS, and encoder inputs. This setting should be separately designated for each motor except some formats.

Format: (I) $ B# E A DU DL ⬤
Sets the input logic for all sensors of the target motor.
DU... Sets the upper 4 bits (bit7 to bit4) in hexadecimal notation.
DL... Sets the lower 4 bits (bit3 to bit0) in hexadecimal notation.
As for the bit number of the sensor, see the following page “Bit number list”.
[DEFAULT=00]

(II) $ B# E A BT DT B ⬤
Sets the input logic for each sensor of the target motor individually.
BT... Bit number of each sensor (0 to 7)
As for the bit number of the sensor, see the following page “Bit number list”.
DT... Input logic
DT = 0 --- active low (Normally open logic)
DT = 1 --- active high (Normally closed logic)

(III) $ B# E A H ⬤
Sets the input logic for all sensors of the motor 1, 2 to the same input logic as Rorze controller RC-231HA.
(See the following page “Input logic of RC-231 and RC-231HA”.)

(IV) $ B# E A L ⬤
Sets the input logic for all sensors of motor 1, 2 to the same input logic as Rorze controller RC-231.

(V) $ B# E A ⬤
Queries the setting of “EA”.
The query should be made separately for each motor (1 or 2).

Reply: In case of (I) to (IV) >

In case of (V) $ B# DU DL ⬤
DU, DL... Hexadecimal values (0 to F) representing the upper and the lower 4 bits of each sensor.
Corresponding to “Bit number list”.

B-68
10. Command Reference

<bit number list>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>bit7 not used (Always 0)</td>
</tr>
<tr>
<td>U</td>
<td>bit6 not used (Always 0)</td>
</tr>
<tr>
<td></td>
<td>bit5 INP1 and INP2</td>
</tr>
<tr>
<td></td>
<td>bit4 EMS</td>
</tr>
<tr>
<td>D</td>
<td>bit3 ORG</td>
</tr>
<tr>
<td>L</td>
<td>bit2 CW and CCW Limit sensor inputs</td>
</tr>
<tr>
<td></td>
<td>bit1 EZ</td>
</tr>
<tr>
<td></td>
<td>bit0 not used (Always 0)</td>
</tr>
</tbody>
</table>

Bit data 0 … Normally open logic
Bit data 1 … Normally closed logic

<Input logic of RC-231 and RC-231HA>

<table>
<thead>
<tr>
<th>Sensor</th>
<th>INP</th>
<th>EMS</th>
<th>ORG</th>
<th>LIMIT</th>
<th>EZ</th>
<th>HEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC-231</td>
<td>active low</td>
<td>active low</td>
<td>active low</td>
<td>active low</td>
<td>active low</td>
<td>00</td>
</tr>
<tr>
<td>RC-231HA</td>
<td>active low</td>
<td>active low</td>
<td>active high</td>
<td>active high</td>
<td>active high</td>
<td>0E</td>
</tr>
</tbody>
</table>

(Hex data is the value queried by the format (Ⅴ).)

Description

- The difference will be occurred depending on the sensor input logic as follows;

  When the input logic of the sensor is set to active low (Normally open):
  - Limits sensor and home (ORG) sensor use active low switch.
    The limit sensor terminal is usually the High level signal and when sensor is turned ON, it becomes the Low level signal.

  - Encoder signal is the Low level when Z-phase is OFF and when Z-phase is turned ON, it becomes the High level.

  - Emergency stop signal is the High level when EMS is OFF and when EMS is turned ON, it becomes Low level.

  - In mode 2 (servo motor operation), INP(in-position) terminals can be connected with a low active servo diver.
Command **EA**

When the input logic of the sensor is set to active high (Normally closed):

- Limits sensor and home (ORG) sensor use active high switch.
  - The sensor terminal is usually the Low level signal and when sensor is turned ON, it becomes the High level signal.

![Active Low Switch Diagram](image)

- Encoder signal is the Low level when Z-phase is OFF and when Z-phase is turned ON, it becomes the High level.
- Emergency stop signal is the Low level when EMS is OFF and when EMS is turned ON, it becomes High level.
- In mode 2 (servo motor operation), INP (in-position) can be connected with a high active servo driver.

**Important caution**

Default value (at the time of shipment) of input logic for each sensor is set to active low (Normally open).

**Note**

- Execute the command “EA” after changing the motor using the command “F” to designate the data for each motor separately.
- Set the input logic for the terminal which isn’t connect to a sensor to active low (Normally open) assuredly. If the input logic is set to active high (normally closed) when a sensor isn’t connected, RC-234 recognizes that the sensor is ON at all time.
- Setting emergency stop input logic is the common for each motor because there is only one EMS terminal. Whichever motor you select, the value for each motor is the same.
- As for the change of INP input logic, see the command “CI” exclusively used in an operation mode 2.
**Command Reference**

**E0 E1 E2**

<table>
<thead>
<tr>
<th>Command</th>
<th>E0</th>
<th>E1</th>
<th>E2</th>
<th>During motion</th>
<th>EEPROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>ED</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>×</td>
<td>○</td>
</tr>
</tbody>
</table>

**Action**

This command sets the clock logic to CW & CCW 2P (2CK) logic or PULSE & DIR (1CK) logic.

- When command “EM”=0 (alternate 2-axis control), the setting is shared between the motor 1 and 2. (Common data for motor 1 and 2).
- When command “EM”=1 (simultaneous 2-axis control), this setting should be separately designated for each motor.

**Format**

(I) $ B# E D 1

Sets the clock logic to CW & CCW 2P (2CK) logic.  
【DEFAULT】

(II) $ B# E D 0

Sets the clock logic to PULSE & DIR (1CK) logic.

(III) $ B# E D

Queries the current pulse output method, 2P (2CK) logic or PULSE & DIR (1CK) logic.

**Reply**

In case of (I, II) >

In case of (III) $ B# DT ...

0  Pulse output is pulse signal & rotation direction signal method.
1  Pulse output is CW clock pulse & CCW clock pulse method.

**Description**

- 2P (2CK) is the method to output 2 types of clock pulse signal, CW clock pulse and CCW clock pulse.
- PULSE & DIR (1CK) is the method to output clock pulse signal and direction signal, CW or CCW.

- In case that the driver has 1CK/2CK dip switches. (RD-023MS, etc.)

<table>
<thead>
<tr>
<th>Clock logic</th>
<th>RC-234 Command</th>
<th>Driver’s DIP SW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pulse signal &amp; rotation direction signal method</td>
<td>ED0</td>
<td>1CK</td>
</tr>
<tr>
<td>CW clock pulse &amp; CCW clock pulse method</td>
<td>ED1</td>
<td>2CK</td>
</tr>
</tbody>
</table>

In case that the driver has no 1ck/2ck dip switches, Rorze drivers are 2P(2CK) logic except exceptions. Please set RC-234 with 2P(2CK) logic.

**Note**

- The default (at the time of shipment) of clock logic is set to CW & CCW 2P(2CK) logic
### Command EE

| Action | This command sets echo back function. You can detect an error on a communication line. |
| Format | (I) $ B# E E 1 $ Sets echo back function ON. |
| | (II) $ B# E E 0 $ Sets echo back function OFF. 【DEFAULT】 |
| | (III) $ B# E E $ Queries the current setting of “EE”. |

**Reply:** In case of (I) >

In case of (II) $ B# E E 0 $ When RC-234 is set with echo back function off, the reply is the same as format (I).

In case of (III) in case that the controller is set with echo back function on $ B# E E $ $ B# 1 $ in case that the controller is set with echo back function off $ > $ $ B# 0 $
Description

- When echo back function is set by the format (I), the reply is returned as the following format:

  <Format>

<table>
<thead>
<tr>
<th>Echo back part</th>
<th>Reply part</th>
</tr>
</thead>
<tbody>
<tr>
<td>Echo back part</td>
<td>Command itself received by RC-234</td>
</tr>
<tr>
<td>Reply part</td>
<td>The format is left out [&gt;] from the usual response format.</td>
</tr>
</tbody>
</table>

  When the usual reply format is only [>], the reply part isn't returned to PC.

  The following table shows the difference of reply when the echo back function is ON and OFF.

<table>
<thead>
<tr>
<th>Command</th>
<th>Examples of response from RC-234 to Host</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Echo back function ON</td>
</tr>
<tr>
<td>$1\downarrow$</td>
<td>$1\downarrow$ $1\ 0\downarrow$</td>
</tr>
<tr>
<td>$1\ 0\downarrow$</td>
<td>$1\ 0\downarrow$</td>
</tr>
<tr>
<td>$1\ 6\downarrow$</td>
<td>$1\ 6\downarrow$ $1\ 0\ 0\ 0\ 0\ 0\ 3\ 0\ 0\downarrow$</td>
</tr>
<tr>
<td>$1\ B\ 0\ 1\downarrow$</td>
<td>$1\ B\ 0\ 1\downarrow$</td>
</tr>
</tbody>
</table>

Note

- The command “SUM” and the command “EE” can’t be set at the same time. Please use the command “EE” after resetting the command “SUM”.

- When echo back function is set ON, normal response isn’t received by the BASIC sample program given as an example in Chapter 8 “Communication Check”. Because carriage return is included between command itself and reply part of returned reply, the reply part is overwritten on command itself and displayed. Because carriage return is replaced to “】” mark in Chapter 13-2 ② “Sample Program for Special Function” program example 2, the reply can be confirmed.
Action: This command checks whether the data stored in EEPROM by the command "DW", "AW", and "IW" is correct.

Format: \( \$ B# E E C H E C K \)

Reply: \( \geq B# D T \) DATA_OK! The data in EEPROM is normal.

- DW_ERROR The data in EEPROM by the command "DW" is abnormal.
- AW_ERROR The data in EEPROM by the command "AW" is abnormal.
- IW_ERROR The data in EEPROM by the command "IW" is abnormal.

When RC-234 received the command "EECHECK", the reply \( \geq \) is returned and then the reply \( \$ B# D T \) is returned after completion of checking the data in EEPROM.

Note
- Usually, the abnormal data reply isn't returned. In case that the abnormal reply is returned, please contact our company.
**Action**: This command erases all data stored in EEPROM.

**Format**: $B#E E/

**Reply**: $B#* When RC-234 received the command “EE///”, the reply [>] is returned and then the reply [$ B# * ] is returned after completion of erasing the data in EEPROM.

**Description**
- All data stored by following commands are erased by the command “EE///”.

<table>
<thead>
<tr>
<th>Store Command</th>
<th>Contents stored in EEPROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;I W&quot;</td>
<td>User program</td>
</tr>
<tr>
<td>&quot;A W&quot;</td>
<td>Position data of the command “A”</td>
</tr>
<tr>
<td>&quot;D W&quot;</td>
<td>Data for the speed, function setting etc.</td>
</tr>
<tr>
<td>&quot;G A&quot;</td>
<td>Auto-start function</td>
</tr>
<tr>
<td>&quot;E S&quot;</td>
<td>Baud rate setting data</td>
</tr>
</tbody>
</table>

**Note**
- To erase data stored in EEPROM, it will take about 5 seconds.
  - If a command is sent during erasing, RC-234 ignores it and returns no reply for the command.
10. Command Reference

**Command** E L

<table>
<thead>
<tr>
<th>E0</th>
<th>E1</th>
<th>E2</th>
<th>During motion</th>
<th>EEPROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>◯</td>
<td>◯</td>
<td>◯</td>
<td>×</td>
<td>◯</td>
</tr>
</tbody>
</table>

**Action**: This command adds carriage return at the end of reply.

**Format**: (I)

```
$ B# E L 1 $D
```

Enables carriage return to the reply of the general command [>].

After setting, the reply of general command becomes [> $D].

The reply of the query command (command “6”, “9”, etc.) isn’t changed.

(II)

```
$ B# E L 0 $D
```

Disables carriage return to the reply of the general command [>].

After resetting, the reply of general command returns only [>].

【DEFAULT】

(III)

```
$ B# E L $D
```

Queries the current setting of “EL”.

**Reply**: In case of (I) > $D

In case of (II) >

In case of (III) > $ B# DC $D

<table>
<thead>
<tr>
<th>D C...</th>
<th>1</th>
<th>carriage return enabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>carriage return disabled</td>
</tr>
</tbody>
</table>

**Description**

- The following table shows the examples of the difference of the reply when the command “EL” is ON(enable carriage return) and OFF(disable carriage return).

<table>
<thead>
<tr>
<th>Command</th>
<th>Reply examples</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Function: ON</td>
</tr>
<tr>
<td>$ 1 0 $D</td>
<td>&gt; $D</td>
</tr>
<tr>
<td>$ 1 $D</td>
<td>&gt; $ 1 0 $D</td>
</tr>
<tr>
<td>$ 1 B 0 1 $D</td>
<td>&gt; $D</td>
</tr>
<tr>
<td>$ 1 6 $D</td>
<td>&gt; 1 0 0 0 0 0 3 0 0 $D</td>
</tr>
</tbody>
</table>

**Note**

- To check the current setting, see the sample program in Chapter 13-2 ② “Sample Program for Special Function” which replaces carriage return to “]” mark and display on the monitor screen.
10. Command Reference

<table>
<thead>
<tr>
<th>Command</th>
<th>E M</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0</td>
<td>E1</td>
</tr>
<tr>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>During motion</td>
<td>EEPROM</td>
</tr>
<tr>
<td>×</td>
<td>○</td>
</tr>
</tbody>
</table>

Action: This command selects control mode, alternate 2-axis control/simultaneous 2 axis control.

Format: (I) $B# EM 1
Selects simultaneous 2-axis control.

(II) $B# EM 0
Selects alternate 2-axis control (compatible with RC-233) 【DEFAULT】

(III) $B# EM
Queries the current setting of control mode.

Reply: In case of (I, II) ▶

In case of (III) ▶ $B# md.. md
md 0: alternate 2-axis control
1: simultaneous 2-axis control

Note
- See Chapter 6 "6-1 simultaneous control".
Action: This command sets uni-directional (0 to 16,777,215) or bi-directional counting (-8,388,608 to +8,388,607). The specified value is shared between the motor 1 and 2. (Common data for motor 1 and 2)

Format:

(I) \$ B# E P 1
Sets bi-directional counting (-8,388,608 to +8,388,607)

(II) \$ B# E P 0
Sets uni-directional counting 0 to 16,777,215 【DEFAULT】

(III) \$ B# E P
Queries the current counting mode 0 to 16,777,215 or -8,388,608 to +8,388,607.

Reply: In case of (I, II) >

In case of (III) > \$ B# DT
DT.. 0 --- data memory position range: 0 to 16,777,215
1 --- data memory position range: -8,388,608 to +8,388,607
Description
• The reply of the position query command “2D”, “6”, “A□□□□□□”, etc. will be changed depending on the change of counting mode (0 to 16,777,215 or -8,388,608 to +8,388,607)

< Sample reply when querying by the command “2D” or “A□□□□□□” >

<table>
<thead>
<tr>
<th>uni-directional counting (EP=0)</th>
<th>bi-directional counting (EP=1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ B# 0 0 0 0 0 0 0 $</td>
<td>$ B#+ 0 0 0 0 0 0 0 $</td>
</tr>
<tr>
<td>$ B# 0 3 8 8 6 0 7 $</td>
<td>$ B#+ 8 3 8 6 0 7 $</td>
</tr>
<tr>
<td>$ B# 0 3 8 8 6 0 8 $</td>
<td>$ B#− 8 3 8 6 0 8 $</td>
</tr>
<tr>
<td>$ B# 1 6 7 7 2 1 5 $</td>
<td>$ B#− 0 0 0 0 0 0 1 $</td>
</tr>
</tbody>
</table>

uni-directional counting 0 to 16,777,215
bi-directional counting -8,388,608 to +8,388,607

< Sample reply when querying by the command “6” >

<table>
<thead>
<tr>
<th>uni-directional counting (EP=0)</th>
<th>bi-directional counting (EP=1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ B# 0 0 0 0 0 0 0 $</td>
<td>$ B#+ 0 0 0 0 0 0 0 $</td>
</tr>
<tr>
<td>$ B# 0 3 8 8 6 0 7 $</td>
<td>$ B#+ 8 3 8 6 0 7 $</td>
</tr>
<tr>
<td>$ B# 0 3 8 8 6 0 8 $</td>
<td>$ B#− 8 3 8 6 0 8 $</td>
</tr>
<tr>
<td>$ B# 1 6 7 7 2 1 5 $</td>
<td>$ B#− 0 0 0 0 0 0 1 $</td>
</tr>
</tbody>
</table>

uni-directional counting 0 to 16,777,215
bi-directional counting -8,388,608 to +8,388,607

The position data used by the command “6” is a 8-digit number even when “EP”=1. RC-234 acknowledges the accurate current position even if it is out of range (-8,388,608 to +8,388,607). However, when the position data is a 8-digit number and the command “EP” =1, the reply will be the last 7 digits.

Note
• When mode=1 and the command “EM=1 (simultaneous 2-axis control), the value set by the command “EP” is ignored and bi-directional counting is automatically selected.
Action: This command enables “Error display function” which returns the reply [>@] in case of command error.

Format: 
(I) \$
B\#
E
R
1
Enables “error display function”, which adds [@] after the reply of command [>] and display it when there is a mistake of format in the executed command.

(II) \$
B\#
E
R
0
Disables “error display function”. After resetting, the reply returns to the usual one. 【DEFAULT】

(III) \$
B\#
E
R
Queries the current setting of “ER”.

Reply: In case of (I, II) [>]

In case of (III) [>] $ B# DC
D C . 1 “error display function” ON
0 “error display function” OFF

Note

- “Error display function” is valid for the user command of the executing user program. If “Error display function” is enabled, when a command error has occurred in the executing user program, RC-234 returns one word ‘[@] to the host and the user program will be terminated.

- “Error display function” returns [>@] only for the command occurred the error. This function itself doesn’t change the command error flag of command “(NULL)” and “9”.

- The reply [>@] when the command error has occurred after enabling this function doesn’t contain [\[. To add [\[, enable the command “EL”. 
### Command Reference

<table>
<thead>
<tr>
<th>Command</th>
<th>E S</th>
</tr>
</thead>
</table>

| Action | This command sets the baud rate of communication. At the same time, the changed baud rate data is stored in EEPROM. |

<table>
<thead>
<tr>
<th>Format</th>
<th>(I)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>B#</td>
</tr>
</tbody>
</table>

Sets the baud rate.
At the same time, the data is stored in EEPROM.

- B P . . 0 9,600 bps [DEFAULT] 3 38,400bps
- 1 1,200 bps 4 19,200bps
- 2 300 bps 5 2,400bps

(II) |

Queries the current baud rate.

<table>
<thead>
<tr>
<th>Reply</th>
<th>In case of (I)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;</td>
<td></td>
</tr>
</tbody>
</table>

In case of (II) |

| B P . . | $ | B# | BP | [ ] |

- 0 9,600 bps 3 38,400bps
- 1 1,200 bps 4 19,200bps
- 2 300 bps 5 2,400bps

**Example**

- PRINT #1,"$1ES1";CHR$(&HDF);
  This code sets the baud rate of RC-234 with body number 1 to 1,200 bps.
  After execution of this command, it’s impossible to communicate by the baud rate except 1,200 bps.

**Note**

- Once baud rate is changed, if the power of RC-234 is turned OFF, the data written in EEPROM is called out every time the power is turned ON and communicates with the changed baud rate.

- From just after executing format (I), the changed baud rate is effective.
  Therefore, to communicate continuously, change the baud rate of PC.

- To restore the default, send the command “ES 0”.
  Also, the command “EE///” can restore the default, but if executing “EE///”, all data written in EEPROM by the command “AW”, “DW”, “IW” is erased.

- When the baud rate which is not in a PC side is set by the format (I), the communication with PC is impossible.
  In such a case, set the baud rate of the PC to 9,600 bps and set the body number (yellow rotary switch) to F and send "$F*" and then send only “B” (Don’t attach “$F”) to RC-234.
  If the body number is set to the previous number (except F) again and the power is reset, all data written in EEPROM are erased and the default value(9,600bps) is restored.
Command: F

<table>
<thead>
<tr>
<th>E0</th>
<th>E1</th>
<th>E2</th>
<th>During motion</th>
<th>EEPROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

Action: This command selects motor.

Format:

(1) \$ $ B# F MT $ Selects motor.
M T ... 1 or 2

[DEFAULT=1]

(2) \$ $ B# F $ Queries the target motor.

Reply: In case of (1) 

\[
\text{In case of (2): } \]$ $ B# MT $ \]

Description:

- After selecting a motor by the command “F”, it will become a target motor for the command executed after that.
- Some commands designate a setting for each motor separately and even if the format is the same, the data memory area is different at the time of selecting the motor 1 and the time of selecting the motor 2.

<Commands which can designate the data for motor 1 and 2 separately>


- Commands which can designate the data in simultaneous 2-axis control (Command “EM”=1”)

The move commands except some commands can select motor 1 or 2 and run it.

Example) \$ 1 4 2 $ (Rotates motor 2)

This example carries out operation equivalent to the case where the command “4” is send, after selecting a motor 2 by the command “F”. Therefore, if motor 1 is selected by the command “F”, a target motor is changed to motor 2 after executing this example.

<Commands executed after selecting a motor>

- Command “$0”, “$ 0 R”, “$ 0 H”, “$ 0 A”, “$ 1”, “$ 3”, “$ 4”, “$ 5”, “$ 6”, “$ 7”, “$ 8”, “$ B”, “$ B + ( - )”

Without selecting a motor before executing the above commands, last selected motor will become a target motor.
10. Command Reference

<Commands to query the data after selecting a motor>

Command “6”, “Q D”, “C L (M 1, M 2)”, “C I”

These commands by itself can select a motor to query the data but the current target motor will not be changed.
If the above commands are executed without selecting a motor, the data of the current target motor is queried.

< Others, the commands that is effective for the command “F” >

Example

```
PRINT #1, "$1F1"; CHR$(&HD);
PRINT #1, "$10"; CHR$(&HD);
PRINT #1, "$1F2"; CHR$(&HD);
PRINT #1, "$10"; CHR$(&HD);
```

The program in the 1st line, selects the motor 1 connected to RC-234 with body no. 1. Then executes a home search of the motor 1 in 2nd line. In 3rd line, selects motor 2 and execute the home search of motor 2 in 4th line.
However, because RC-234 can’t move 2 motors simultaneously and also execute the command “F” during motion, the following routine is necessary before executing the second line of example to confirm the status of motor 1 by the command “(NULL)” and repeat the confirmation using command “(NULL)” if motor 1 is during motion.

```
CHECK:
   PRINT #1, "$1"; CHR$(&HD);
WAIT:
   IF LOC(1) < 4 THEN GOTO WAIT
   X$=MID$(INPUT$(LOC(1), #1), 3, 1)
   IF X$ <> " " THEN GOTO CHECK
```

NOTE

- In alternate 2-axis control (command “EM”=0), RC-234 can’t change the target motor during motion.
- The data set by the command “E”, “ED”, “OX”, “QE”, “QJA”, “QJT” when the command “EM”=0 is shared by the data of motor 1 when the command “EM”=1.
10. Command Reference

**Command G**

<table>
<thead>
<tr>
<th>E0</th>
<th>E1</th>
<th>E2</th>
<th>During motion</th>
<th>EEPROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☒</td>
</tr>
</tbody>
</table>

**Action**: This command executes a user program stored by the command “I” in RAM.

**Format**: (Ⅰ)  
\[ \$ \text{B#} \text{G} \text{)} \]
Executes a user program from the beginning.

(Ⅱ)  
\[ \$ \text{B#} \text{G} \text{LB} \text{)} \]
Executes the program from the label command “/G LB” in the user program.  
(See Chapter 12 “User Program Command Reference”.)

LB... Label name of the user program  
0 to 9, @, ¥, etc. (1 byte character except A to Z, a to z, $, #, /)

Note: a to z can be used in the user program as a label but they can't be used in the command “G”.

**Reply**:  
\[ > \] (For both I and II)

**Note**  
- See Chapter 11 “User Program Explanation” as for the user program.
- The command “G” executes not the user program written in EEPROM but the user program written in RAM.
- If the format (Ⅰ) is executed without any user program in RAM the controller will set the error flag ON.  
  Also, when you select the label which doesn't exist in the user program and the format (Ⅱ) is executed error will occur.  
  To query the contents of the user program in RAM use the command “IR”.
- The user program will be stopped upon the occurrence of command error in the user program.  
  The command “G” itself can be used when motor is rotating, but if there is a command that can't be used during motion in the user program, upon the execution of the command, the error will occur and the user program will be terminated.  
  However, if the cause of the error is not a command in the user program, the user program does not stop and continues to run.
- During executing the user program, the command “G” can't be executed again.  
  Also, the command “I”, “IW”, “IL” can't be used too.
- To query the status of the user program use the command “GSS”.  
  Also, to terminate the user program during motion, please use the command “GE” or “GES”.
### Command Reference

#### Command: GAS

**Action:** This command enables autostart of the user program. Autostart data is stored in EEPROM and the user program is executed automatically every time when RC-234 is turned ON.

**Format:**

1. $B# G A S$
   - Enables autostart.

2. $B# G A R$
   - Disables autostart [DEFAULT]

3. $B# G A$
   - Queries the current setting of "GA".

**Reply:**
- In case of (Ⅰ, Ⅱ)
  - >

- In case of (Ⅲ)
  - > $B# DT$
    - 0 Autostart is enabled
    - 1 Autostart is disabled

**Note**

- If a RC-234 is turned off before the command "IW" is executed to save a User Program on the RAM into the EEPROM, the User Program is cleared. Make sure that the command "IW" is sent, otherwise the command "GAS" is meaningless.

- When only user program is executed using autostart, there is no need to connect PC(RS-232C) and Link master RC-002 to RC-234.

- Changes in a user program of an EEPROM do not affect the setting of the autostart function. To disable autostart use the format (Ⅱ). Also, the command "EE///" can also reset the function but if the command "EE///" is executed, the whole data stored into EEPROM using the command "AW", "DW", "IW" are erased simultaneously.
### Command Reference

<table>
<thead>
<tr>
<th>Command</th>
<th>G C</th>
</tr>
</thead>
</table>

**Action**: This command resumes the user program after temporary stoppage.

**Format**: $ B\# \ G \ C \ [\text{Enter}]$

**Reply**: $>

**Note**
- See the note in the command “GS”.
### Command G E

<table>
<thead>
<tr>
<th>E0</th>
<th>E1</th>
<th>E2</th>
<th>During motion</th>
<th>EEPROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td>×</td>
<td>×</td>
<td>○</td>
<td>×</td>
</tr>
</tbody>
</table>

**Action**: This command ends the user program.

**Format**:
$$\text{G E}$$

**Reply**: $>

**Note**
- If the motor is running, it will not stop until it reaches a designated position, or it will continue to run when it has been activated by the command “7” or “8”. To stop the motor, use the command “S”.

---

### Command G E S

<table>
<thead>
<tr>
<th>E0</th>
<th>E1</th>
<th>E2</th>
<th>During motion</th>
<th>EEPROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td>×</td>
<td>×</td>
<td>○</td>
<td>×</td>
</tr>
</tbody>
</table>

**Action**: This command ends the user program immediately. If either of the motors were running it will be stopped immediately.

**Format**:
$$\text{G E S}$$

**Reply**: $>
**Command Reference**

<table>
<thead>
<tr>
<th>Command</th>
<th>GN</th>
</tr>
</thead>
</table>

**Action**: This command queries the next step to be executed in a user program. Also, the command is useful to spot the program line where execution has terminated in an error state.

**Format**:  
```
$ B# G N △
```

**Reply**:  
```
> $ B# △△△△△ △
```

- 5-digit decimal number
- Number of [/]

**Description**
- Each command is preceded by [/].
- The command “GN” queries the number of [/] from [/] of the first command in the user program to [/] of the command executed next. (As for the format of the user program, see Chapter 11 “User program explanation”.)

**Example**:  
```
/7/T1 0 0/S-END
```

When the command “GN” is executed while executing the user program like the above, if the reply indicates the number of [/] is 3, it indicates that command “S” is executed next. However, please be careful that “during executing this command the user program is running.

The command is useful to spot the program line where execution has terminated in an error state or where execution has terminated by the command “GE” or “GES”. If the user program ended with the command “/END” normally, the program line number of “/END” is returned as a reply because there is no command executed next.
Command G R

Action: This command queries the status of individual bits of any buffer and current number in loop counter.

Format:

(I) $ B# G R BF
Queries the current status in the data buffer A to F used as Variable in the user program.
B F . . Data buffer (A to F)

(II) $ B# G R W BF
Queries the current status in the 4-byte data buffer WA to WF used as Variable in the user program.
W(BF) . 4-byte Data buffer (WA to WF)

(III) $ B# G R CT
Queries the current counter number in loop counter used in the user program.
C T . . Counter 1 or 2

(IV) $ B# G R X CT
Queries the current counter number in loop counter for the position number used in the user program.
X(CT) . Counter X1 to X5

Reply:

Incase of (I) > $ B# DU DL
DU, DL . . Hexadecimal values (0 to F) representing the upper and the lower 4 bits of the queried data buffer

Incase of (II) > $ B# d t
-999,999,999 to +999,999,999
Signed 9-digit decimal number

Incase of (III, IV) > $ B# OOOOO
5-digit decimal number

Note:
- As for the user program, see Chapter 11 “User Program Explanation”. 
Action: This command queries the current time left.

Format:
(I) $B\# GR T

(II) $B\# GR T NO.

NO.. Number of timer (1 to 5)

Reply: $B\# \ldots \ldots \ldots$

5-digits decimal number
in 100 msec time base (5 digits $\times$ 100 msec)

Description
- The data of the timer is used for the following commands.

  Commands "T", "P"
  Commands for user program "/T", "/TB", "/TS"

  Executing these command decrements the setting value in 100 msec time base. To query the progress use the command "GRT". If the value queried by the command "GRT" is 0, it means that timer has timed out.
10. Command Reference

#### Command GS

<table>
<thead>
<tr>
<th>Action</th>
<th>This command pauses a user program. To resume the user program use the command “GC”.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format</td>
<td>$ B# G S</td>
</tr>
<tr>
<td>Reply</td>
<td>&gt;</td>
</tr>
</tbody>
</table>

**Note**
- When the command “GE” or “GES” to terminate the user program has been executed under the condition that the user program is stopped temporarily using the command “GS”, it’s impossible to resume the user program using the command “GC”.
- To query the status of user program use the command “GSS”.

#### Command GSS

<table>
<thead>
<tr>
<th>Action</th>
<th>This command queries the status of user program.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format</td>
<td>$ B# G S S</td>
</tr>
<tr>
<td>Reply</td>
<td>&gt; $ B# DT</td>
</tr>
</tbody>
</table>

**DT** Indicates the status of user program.
- DT = 1 user program running
- 0 no user program running
- P user program paused
- E user program ended because of stop

**Note**
- If the status of user program is queried again after confirming that the user program is ended because of stop using the command “GSS”, it replies that DT is 0 and the user program is stopped.
- During running the user program, if the command error has occurred in the user program, the user program is stopped immediately and DT data of command “GSS” becomes E which indicates user program is ended because of stop.
- To query the status of error, please use the command “QSSE”.
**Action**: This command reports the cause of stop in the user program.

**Format**: $B\# G S S E <br>

**Reply**: $B\# DT...<br>DT..<br>0 user program running<br>1 normal end<br>2 incorrect command<br>3 no jump-label<br>4 move command execution during motion<br>5 termination of user program by “?” error<br>6 termination of user program by command
Command   $GW

**Action**: This command changes the buffer data of buffer and current number in loop counter.

**Format**:

(I) $B\# GW BF DU DL S

Sets the bits in the data buffer A to F used as Variable in the user program and changes the buffer data.

- BF.. data buffer (A to F)
- DU.. Sets the upper 4 bits (bit7 to bit4) of the specified BF in Hexadecimal notation.
- DL.. Sets the lower 4 bits (bit3 to bit0) of the specified BF in Hexadecimal notation.

(II) $B\# GW W BF DU DL S
dt

Sets the bits in the 4-byte data buffer WA to WF used as Variable in the user program and changes the 4-byte data buffer.

- W(B F).. 4-byte data buffer (WA to WF)
- dt.. -999,999,999 to +999,999,999

(III) $B\# GW CT DU DL S

Changes the counter number in loop counter used in the user program.

- CT.. counter 1 or 2
- dt.. counter data (0 to 65,535)

(IV) $B\# GW X CT DU DL S

Changes the counter number in loop counter for the position number used in the user program.

- X(CT).. counter X1 to X5
- dt.. counter data (0 to 999)

**Reply**: $ (for all, I, II, III and IV)

**Note**

- As for the user program, see Chapter 11 “User Program Explanation”.

---

10. Command Reference
## Command Reference

### Command: GWT

<table>
<thead>
<tr>
<th>Command</th>
<th>G W T</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0</td>
<td>E1</td>
</tr>
<tr>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td></td>
<td>During motion</td>
</tr>
<tr>
<td>○</td>
<td>×</td>
</tr>
</tbody>
</table>

**Action**: This command changes timer data during motion.

**Format**:

1. $(\text{Ⅰ})$
   - $\$ \text{B# G W T , } \text{TIME}$
   - Timer data (0 to 32,767 (Approx. 54 minutes)).
   - The `time` data is specified in terms of 100 milliseconds, hence the actual wait time is `time` multiplied by 100 msec.

2. $(\text{Ⅱ})$
   - $\$ \text{B# G W T NO , } \text{TIME}$
   - Number of timer (1 to 5)
   - Timer data (0 to 32,767 (Approx. 54 minutes)).
   - The `time` data is specified in terms of 100 milliseconds, hence the actual wait time is `time` multiplied by 100 msec.

**Reply**: $>$ (For both Ⅰ and Ⅱ)

**Description**

- If the command “GWT” is executed during counting time by the command “T”, “P” in PC based control or by the command for the user program such as “/T”, “/TB”, “/TS”, counting timer data is rewritten to the timer data set by the command “GWT” and restart to count time from the rewritten timer data.
- If this command is executed under the condition that timer has timed out (when timer isn’t used), a command error will occur.

**Note**

- The numbered timer in the format (Ⅱ) is independent by number of timer 1 to 5 separately, but the timer in format (Ⅰ) is common for the command “T”, “P” and user program command “/T”, “/TB” and “/TS”.

---

B-94
### Command H

<table>
<thead>
<tr>
<th>Command</th>
<th>H</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>E0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>0</td>
</tr>
<tr>
<td>E2</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>During motion</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EEPROM</td>
<td>0</td>
</tr>
</tbody>
</table>

**Action**: If this command is sent during rotating motor using the command “7” or the command “8”, motor speed will become high speed.

**Format**: `$ B# H

**Reply**: $
**10. Command Reference**

### Command I

<table>
<thead>
<tr>
<th>Command</th>
<th>I</th>
</tr>
</thead>
</table>

**Action:** This command transmits the user program to RAM in RC-234.

**Format:**

1. \$ B# I  

2. **Body of User Program**  

   Upon receiving the reply [>] from RC-234 after executing format ①, the user program will be transmitted to RAM using format ②.

   The user program memory is 8,000 bytes.  (Approx. 1,500 User commands)

**Reply:** >

**Description**

- The body of user program transmitted by the format ② can be divided into several times.

  - **User program division A**  
  - **User program division B**

  For example, if the above division A is transmitted and after a while division B is transmitted, RC-234 recognizes division A and division B as the body of user program.

  However, if “$” is transmitted during transmitting the user program, the user program is finished to transmit the user program, and the user program transmitted up to the middle is recognized as the body of user program.

  In case like the above, if “$” is transmitted after transmitting division A, RC-234 recognizes only division A as the body of user program and finishes to transmit.

**Note**

- As for the user program, see Chapter 11 “User Program Explanation”.

- The user program transmitted to RAM by the command “I” can be executed by the command “G”.

- To save the user program after turning OFF the power, store it into EEPROM using the command “IW”.

  The use program stored in EEPROM is automatically moved to RAM upon power up.

- Use the command “IR” to query the contents of the user program in RAM.

- During executing the user program, the command “I”, “IW”, “IL” can’t be used. Use the command “GSS” to query whether the user program is executing.

- “$” can’t be used for a label of the user program, message return the command “/CO”, etc.

  Because “$” is treated as a control code to indicate the start of the command in RC-234, when “$” is included during sending the user program, the user program after “$” (including “$”) can’t be sent correctly.

- When the first letter is not “/” on sending the user program RC-234 returns “>” and the controller will set the error flag ON.
**Command **\( IL \)  

<table>
<thead>
<tr>
<th>E0</th>
<th>E1</th>
<th>E2</th>
<th>During motion</th>
<th>EEPROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
<td>○</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

**Action**: This command reads out the user program stored in EEPROM using the command “IW” to RAM.

**Format**: \( \$B#\ \ I\ L\ \ )

**Reply**: >

**Description**

①Auto-start of the command “IL” upon the activation of the power

When the data is stored in EEPROM by the command “IW”, command “IL” is executed automatically as soon as RC-234 is powered.

②Overwriting the data in RAM area

If the command “IL” is executed or RC-234 is turned OFF and turned ON again, the current RAM data is abandoned and the user program in EEPROM is read out to RAM.

③Erasing the contents of the EEPROM

The data written in EEPROM will not be erased until executing the command “EE///”.

(Overwriting is possible.)

Note: When executing the command “EE///”, whole data written in EEPROM by the command “AW”, “DW” and “IW” will be erased at the same time.

**Command **\( IR \)  

<table>
<thead>
<tr>
<th>E0</th>
<th>E1</th>
<th>E2</th>
<th>During motion</th>
<th>EEPROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>×</td>
</tr>
</tbody>
</table>

**Action**: This command queries the contents of the user program.

**Format**: \( \$B#\ \ I\ R\ )

**Reply**: > **User Program**

- User program memory is 8,000 byte. (Approx. 1,500 user commands)
Command IW

<table>
<thead>
<tr>
<th>E0</th>
<th>E1</th>
<th>E2</th>
<th>During motion</th>
<th>EEPROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td></td>
<td></td>
<td>×</td>
<td></td>
</tr>
</tbody>
</table>

Action: This command stores the user program in RAM into EEPROM.

Format: \$ B# I W

Reply: > $ B# * When RC-234 received the command “IW”, the reply [>] is returned and then the reply [ $ B# * ] is returned after completion of storing the data into EEPROM.

Note:
- If the command “IW” is executed, the user program is rewritten and the command “IW” is executed again, rewritten user program is overwritten into EEPROM.
- It takes about 3 seconds at max. to store the user program.
- See the description in the command “IL”.

Action: This command stores the user program in RAM into EEPROM.

Format: \$ B# I W

Reply: > $ B# * When RC-234 received the command “IW”, the reply [>] is returned and then the reply [ $ B# * ] is returned after completion of storing the data into EEPROM.

Note:
- If the command “IW” is executed, the user program is rewritten and the command “IW” is executed again, rewritten user program is overwritten into EEPROM.
- It takes about 3 seconds at max. to store the user program.
- See the description in the command “IL”.

Action: This command stores the user program in RAM into EEPROM.

Format: \$ B# I W

Reply: > $ B# * When RC-234 received the command “IW”, the reply [>] is returned and then the reply [ $ B# * ] is returned after completion of storing the data into EEPROM.

Note:
- If the command “IW” is executed, the user program is rewritten and the command “IW” is executed again, rewritten user program is overwritten into EEPROM.
- It takes about 3 seconds at max. to store the user program.
- See the description in the command “IL”.

Action: This command stores the user program in RAM into EEPROM.

Format: \$ B# I W

Reply: > $ B# * When RC-234 received the command “IW”, the reply [>] is returned and then the reply [ $ B# * ] is returned after completion of storing the data into EEPROM.

Note:
- If the command “IW” is executed, the user program is rewritten and the command “IW” is executed again, rewritten user program is overwritten into EEPROM.
- It takes about 3 seconds at max. to store the user program.
- See the description in the command “IL”.
**Command K**

**Action**: This command enables/disables an "error output" function that activates a general-purpose output bit upon an occurrence of either a limit error, an emergency stop, or a stall error. With the "error output" function enabled, RC-234 checks the status flags in "(NULL)" command and turns on a specified output bit if it detects a specified error. (Once the function is enabled, it remains effective until disabled by this command in format (IV).)

**Format**: (I) $ B# K DT . DU DL [J]

Specifies, in hexadecimal, one or more errors to be checked and the bit(s) of the general-purpose port to be activated upon the occurrence of the error(s). The format initiates the "error output" function at the same time. (The bits of the sensor error and the emergency stop in the status flag are cleared upon an execution of this format.)

**D T . .** A 3-bit hexadecimal value representing (an) error(s) to be checked

<table>
<thead>
<tr>
<th>DT (HEX)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limit error</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Emergency stop</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Stall error</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

(O = errors to be checked)

**D U . .** A hexadecimal value representing the upper four bits (bit7-bit4) of the general-purpose port to be activated if a specified error occurs.

**D L . .** A hexadecimal value representing the lower four bits (bit3-bit0) of the general-purpose port to be activated when a specified error occurs.

- **Bit Data 1**: Means the bit will be activated upon the occurrence of an error.
- **Bit Data 0**: Means the bit will remain intact upon the occurrence of an error.

**· Relation between HEX(HEXadecimal) and Binary**

<table>
<thead>
<tr>
<th>H E X</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
</tr>
<tr>
<td>bit3 (bit7)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>bit2 (bit6)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>bit1 (bit5)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>bit0 (bit4)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Command Reference

### Command K

#### Format (Ⅱ)

```
$ B# K DT BT
```

Specifies one or more errors to be checked on a bit basis and a general-purpose port to be activated upon an occurrence of the error(s).

The format starts the "error output" function at the same time.

(The bits of the sensor error and the emergency stop in the status flags are cleared upon an execution of this format.)

- **DT**.. A 3-bit hexadecimal value representing (an) error(s) to be checked
  - DT = 1 to 7
  - (See the table of the format (Ⅰ).)
- **BT**.. A bit number to be specified (0 to 7).

#### Format (Ⅲ)

```
$ B# K 0
```

Disables the "error output" function. 【DEFAULT】

#### Format (Ⅳ)

```
$ B# K C
```

Deactivates the bit(s) of general-purpose port assigned by the format (Ⅰ) or (Ⅱ).

(The bits of a sensor error and an emergency stop in the status flags are cleared upon an execution of this format.)

When an "error output" function is disabled by the format (Ⅲ), an execution of this command is invalid and results in a command error.

#### Format (Ⅴ)

```
$ B# K
```

Queries the errors to be detected and output bits to be activated upon errors.

Reply: In case of (Ⅰ) to (Ⅳ)

```
>
```

In case of (Ⅴ)

```
> $ B# DT DU DL
```

- **DT**.. A 3-bit hexadecimal value representing (an) error(s) being checked.
  - (See the table of the format (Ⅰ).)
  - DT = 0 means that the function is disabled.
- **DU, DL**.. Hexadecimal values (0-F) representing the upper and lower four bits of the general-purpose port to be activated upon error(s).
Description
Each error is checked in the following manner:

- **Limit error**
  If the sensor error bit of the status flags (command "(NULL)") changes from 0 to 1 and the stall error bit of the condition flags (command "9") is 0, (a) specified bit(s) of the general-purpose port is(are) activated.

- **Emergency stop**
  If the emergency stop bit of the status flags (command "(NULL)") changes from 0 to 1, (a) specified bit(s) of the general-purpose port is(are) activated.

- **Stall error**
  If the stall error bit of the condition flags (command "9") changes from 0 to 1, (a) specified bit(s) of the general-purpose port is(are) activated.

After the output port is activated with the "error output" function, the following procedure is needed to resume the function.
- Send the command "K" in the format (IV) to deactivate the output port and to clear the error bit(s) of the status flag.
- Send the command "0" or "RD" to clear the condition flags when the function of a stall error detection is enabled.

**Example**
- `PRINT #1, "$1K2,FF"; CHR$(&HD(;
  If the RC-234 with body number 1 detects an emergency stop (EMS terminal) input, it activates all output bits of the general-purpose port.

- `PRINT #1, "$1K5,3"; CHR$(&HD(;
  If the RC-234 with body number 1 detects a limit or a stall error, it activates the bit3 of the general-purpose port.

**Note**
- In simultaneous 2-axis control, this command enables/disables an "error output" function upon an occurrence of an error in either motor.
- For more information on the status flags and condition flags, see the section of command "(NULL)" and "9", respectively.
- Each error such as a limit, emergency stop, or stall can NOT be separately allocated to the particular bit of the output port.
- An execution of the format (III) does not deactivate the output port the "error output" function has turned on. To turn off the port, use the format (IV).
- Neither disabling an "error out" function by the format (III) nor changing an output port setting by the format (I) or (II) deactivates output bits that have been already activated by the previous performance of the function. In this case, the sensor error and emergency stop bits of the "status flags" are cleared.
  To turn off the port, use the command "D".
- A command execution in the format (I) or (II) will be overridden by succeeding commands in the format (I) or (II).
### Command Reference

#### Command: L

<table>
<thead>
<tr>
<th>E0</th>
<th>E1</th>
<th>E2</th>
<th>During motion</th>
<th>EEPROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>◯</td>
<td>◯</td>
<td>◯</td>
<td>◯</td>
<td>◯</td>
</tr>
</tbody>
</table>

**Action**: This command decelerates the motor speed with ramp to the low speed while a motor is running by the command "7" or "8".

**Format**: $B#L

**Reply**: >
Action: This command sets two dimensional position pulse number that the command “MB”, “MC1”, “MP”, “MS”, etc. use during its execution. If position pulse number is not specified and this command is executed, the current position is stored as the position pulse number.

Format: (I) $ B# M A p n pp1 . pp2
Sets pp1 and pp2 (position pulse data) of pn (position number).
   pn . Position number (000 to 999) up to 1,000 points
   Sets by 3-digit number assuredly.
   pp1. pp2. Position pulse number (-8,388,608 to +8,388,607)
      signed number up to 7 digits [DEFAULT=0]
      (pp1: motor 1, pp2: motor 2)

(II) $ B# M A p n
The current position will be stored as position pulse number of pn if position pulse number is not specified.
   pn . Position number (000 to 999) up to 1,000 points
   Sets by 3-digit number assuredly.

(III) $ B# M A p n D
Reads the position pulse number of position number (pn) set by the command “MA” and receives decimal number with 8 digits replay from RC-234.
   pn . Position number (000 to 999) up to 1,000 points
   Set by 3-digit number assuredly.

Reply: In case of (I, II) $ B# pp1 , pp2
In case of (III) $ B# pp1 , pp2

Example
- PRINT #1, " $ 1 MA 0 0 2 3 0 0 0 , - 2 0 0 0 " ; CHR$ (&H D);
  Sets the position pulse number of pn= 002 in the RC-234 with body number 1 to 3,000 (motor 1) and -2,000 (motor 2).

- PRINT #1, " $ 1 MA 0 2 2 " ; CHR$ (&H D);
  Sets the position pulse number of pn=022 to the current position pulse in the RC-234 with body number 1.

Note
- This command is enabled when the command “EM”=1.

- Position pulse of pp1 is shared by the command “A”. Therefore, the above example sets the position pulse number of pn=002 of command “A” to 3,000.
**Command MB**

<table>
<thead>
<tr>
<th>Action</th>
<th>Format</th>
<th>Reply</th>
</tr>
</thead>
<tbody>
<tr>
<td>This command moves motor 1 and motor 2 to the position specified by the command “MA” with a simultaneous start. (Absolute move) The speed parameter should be separately specified for motor 1 and 2.</td>
<td>(I) $ B# M B p n [\rangle] Moves a motor to the position of pn specified by the command “MA”. pn... Position number (000 to 999) up to 1,000 points Set by 3-digit number assuredly. (II) $ B# M B [\rangle] Reads the pp of subsequent pn of the last move pn by the command “MB” and moves a motor to the position. (III) $ B# M B N [\rangle] Queries the position number of command “MB” to move subsequently.</td>
<td>In case of (I, II) &gt; In case of (III) &gt; $ B# 00000 [\rangle] decimal 5 digits [DEFAULT=00000] Current position pulse number</td>
</tr>
</tbody>
</table>

Note: The position number is 3 digits long, but the reply will be returned with a 5-digit decimal number.

Example

- PRINT #1,"$1MB 2";CHR$(&H0D);
  This code instructs motor 1 and motor 2 of RC-234 with body number 1 to move to the pp position of position number 002 set by the command “MA” with a simultaneous start.

Note
- This command is enabled when the command “EM”=1.
Command MB+

**Action**: This command incrementally moves motor 1 and motor 2 certain distance set by the command “MA” relative to current position with a simultaneous start. (Incremental Move)

The speed parameter should be separately specified for motor 1 and 2.

**Format**:

(Ⅰ) $B# M B + pn$

This command incrementally moves a certain distance set by the command “MA”.

* pn... Position number (000 to 999) up to 1,000 points
  Sets by 3-digit number assuredly.

(Ⅱ) $B# M B +$

Reads the position pulse data of subsequent pn of the last move pn by the command “MB+” and moves by the position data.

**Reply**: $>(For both Ⅰ and Ⅱ )

**Example**

- PRINT #1,"$1MB+ 2" ; CHR$(&HD);
  This code instructs motor 1 and motor 2 of the RC-234 with body number 1 to move by the position data of position number 002 set by the command “MA” with a simultaneous start.

**Note**

- This command is enabled when the command “EM”=1.
Command MC1

<table>
<thead>
<tr>
<th>Command</th>
<th>MC1</th>
</tr>
</thead>
</table>

**Action**: This command performs CW circular interpolation to the position specified by the command "MA". (Absolute move)

The speed parameter should be specified for motor 1.

**Format**: $B#MC1<pn1>,<pn2>$

- $B#$: Command code
- $M$: Motor number
- $C1$: Command number
- $pn1$, $pn2$: Position number (000 to 999) up to 1,000 points

Sets by 3-digit number assuredly.

(pn1: end point, pn2: center of a circle)

**Reply**: >

**Example**

- PRINT #1, "$1MC1 005, 006" ; CHR$(&HD);

This code instructs the RC-234 with body number 1 to perform CW circular interpolation to the position of position number 005 set by the command "MA" (center of a circle: position number 006). (Absolute move)

<table>
<thead>
<tr>
<th>Current position</th>
<th>Absolute coordinates</th>
<th>End point</th>
<th>Absolute coordinates</th>
<th>Center of a circle</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,100</td>
<td>0,100</td>
<td>100,100</td>
<td>100,100</td>
<td>Perfect circle</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100,200</td>
<td></td>
<td></td>
<td>90° circle</td>
<td></td>
</tr>
<tr>
<td></td>
<td>200,100</td>
<td></td>
<td></td>
<td>180° circle</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100, 0</td>
<td></td>
<td></td>
<td>270° circle</td>
<td></td>
</tr>
</tbody>
</table>

**Note**

- This command is enabled when the command “EM”=1.

- Do not change the speed parameter during interpolation.

- In mode 1, divide the execution of a circular interpolation move in a perfect circle 2 times or more.
  (Ex. Performs circular interpolation in 180° circle twice.)
**Command Reference**

**Command** **MC 1+**

<table>
<thead>
<tr>
<th>E0</th>
<th>E1</th>
<th>E2</th>
<th>During motion</th>
<th>EEPROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>◯</td>
<td>◯</td>
<td>◯</td>
<td>✗</td>
<td>○</td>
</tr>
</tbody>
</table>

**Action**: This command performs CW circular interpolation by certain distance set by the command “MA” relative to current position. (Incremental Move)

The speed parameter should be specified for motor 1.

**Format**: $B#M C1+ pn1, pn2$

- $B#$: Position number (000 to 999) up to 1,000 points
- $M$: Sets by 3-digit number assuredly.
- $pn1, pn2$: (pn1: end point, pn2: center of a circle)

**Reply**: >

**Example**

- PRINT #1,"$1MC1+ 005, 006" CHRS (&HD);  
  This code instructs the RC-234 with body number 1 to perform CW circular interpolation by position data of position number 005 set by the command “MA”. (Incremental move)  
  center of a circle: position number 006

<table>
<thead>
<tr>
<th>Current position Incremental coordinates</th>
<th>End point Incremental coordinates</th>
<th>Center of a circle Incremental coordinates</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,0</td>
<td>0, 0</td>
<td>100,0</td>
<td>Perfect circle</td>
</tr>
<tr>
<td>100, 100</td>
<td></td>
<td>90° circle</td>
<td></td>
</tr>
<tr>
<td>200, 0</td>
<td></td>
<td>180° circle</td>
<td></td>
</tr>
<tr>
<td>100,-100</td>
<td></td>
<td>270° circle</td>
<td></td>
</tr>
</tbody>
</table>

**Note**

- This command is enabled when the command “EM”=1.
- Do not change the speed parameter during interpolation.
Command Reference

**Command MC2**

<table>
<thead>
<tr>
<th>E0</th>
<th>E1</th>
<th>E2</th>
<th>During motion</th>
<th>EEPROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
</tbody>
</table>

**Action**: This command performs CCW circular interpolation to the position specified by the command “MA”. (Absolute move)

The speed parameter should be specified for motor 1.

**Format**:

```
$ B# M C 2 pn1 , pn2
```

- **pn1**: Position number (000 to 999) up to 1,000 points
- **pn2**: Sets by 3-digit number assuredly.
  - (pn1: end point, pn2: center of a circle)

**Reply**: >

**Example**

- PRINT #1, "$1MC2 5, 6" ; CHR$(&HD):

  This code instructs RC-234 with body number 1 to perform CCW circular interpolation to the position of position number 005 set by the command “MA”. (Absolute move)

  Center of a circle: position pulse number 006

---

<table>
<thead>
<tr>
<th>Current position</th>
<th>End point</th>
<th>Center of a circle</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absolute coordinates</td>
<td>Absolute coordinates</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0,100</td>
<td>0,100</td>
<td>100,100</td>
<td>Perfect circle</td>
</tr>
<tr>
<td></td>
<td>100,200</td>
<td></td>
<td>90° circle</td>
</tr>
<tr>
<td></td>
<td>200,100</td>
<td></td>
<td>180° circle</td>
</tr>
<tr>
<td></td>
<td>100,0</td>
<td></td>
<td>270° circle</td>
</tr>
</tbody>
</table>

**Note**

- This command is enabled when the command “EM”=1.
- Do not change the speed parameter during interpolation.
- In mode 1, divide the execution of a circular interpolation move in a perfect circle 2 times or more. (Ex. Performs circular interpolation in 180° circle twice.)
**Command Reference**

**Command MC 2+**

**Action:** This command performs CCW circular interpolation by certain distance set by the command “MA” relative to current position. (Incremental Move)

The speed parameter should be specified for motor 1.

**Format:**

```
$ B# M C 2 + pn1 , pn2
```

- `$`: Command character
- `B#`: Parameter character
- `M`: Motor number
- `C`: Mode number
- `2`: Subcommand number
- `+`: Speed parameter
- `pn1, pn2`: Position number (000 to 999) up to 1,000 points

Sets by 3-digit number assuredly.

(pn1: end point, pn2: center of a circle)

**Reply:**

```
>
```

**Example**

```
PRINT #1,"$1MC2+ 5, 6"; CHR$(&HD);
```

This code instructs RC-234 with body number 1 to perform CCW circular interpolation by the position data of position number 005 set by the command “MA”. (Incremental move)

Center of a circle: position number 006

<table>
<thead>
<tr>
<th>Current position Incremental coordinates</th>
<th>End point Incremental coordinates</th>
<th>Center of a circle Incremental coordinates</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,0</td>
<td>0, 0</td>
<td>100,0</td>
<td>Perfect circle</td>
</tr>
<tr>
<td>100, 100</td>
<td></td>
<td></td>
<td>90° circle</td>
</tr>
<tr>
<td>200, 0</td>
<td></td>
<td></td>
<td>180° circle</td>
</tr>
<tr>
<td>100, -100</td>
<td></td>
<td></td>
<td>270° circle</td>
</tr>
</tbody>
</table>

**Note**

- This command is enabled when the command “EM”=1.

- Do not change the speed parameter during interpolation.
**Command MC3**

**Action**: This command performs circular interpolation to the position specified by the command "MA" via intermediate point. (Absolute move)

- The speed parameter should be specified for motor 1.

**Format**: $B\#MC3pn1,pn2$

- $pn1$: Position number (000 to 999) up to 1,000 points
- $pn2$: Sets by 3-digit number assuredly.

- (pn1: end point, pn2: intermediate point)

**Reply**: >

**Example**

- `PRINT #1, "$1MC3005,006\"; CHR$(&HD)`;

  This code instructs the RC-234 with body number 1 to perform a circular interpolation to the position of pn005 via intermediate point (position number 006). (Absolute move)

**Note**

- This command is enabled when the command "EM"=1.

- This command performs circular interpolation with different three points (current position, intermediate point, endpoint). The motion to return to the current position is not available.

- Do not change the speed parameter during interpolation.
**Command MC3+**

**Action**: This command performs a circular interpolation by certain distance set by the command "MA" via intermediate point relative to current position. The speed parameter should be specified for motor 1.

**Format**: 

```
$ B# M C 3 + pn1 , pn2
```

- $: Position number (000 to 999) up to 1,000 points
- B#: Sets by 3-digit number assuredly.
- M: (pn1: end point, pn2: intermediate point)

**Reply**: >

**Example**

```
PRINT #1,"$1MC3+ 5, 6" ; CHR$(&HD);
```

This code instructs RC-234 with body number 1 to perform circular interpolation to end point by the position data of position number 005 set by the command "MA" via intermediate point (position number 006) relative to current position. (Incremental move).

**Note**

- This command is enabled when the command “EM”=1.
- This command performs a circular interpolation with different three points (current point, intermediate point, endpoint). Return to the current position motion is not available.
- Do not change the speed parameter during interpolation.

---

**Command MG**

**Action**: This command performs an interpolation pattern set by the command “MP”. (Continuous interpolation) The speed parameter should be specified for motor 1.

**Format**: 

```
$ B# M G mn
```

- $: Interpolation pattern Hexadecimal values (0 to F)

**Reply**: >

**Example**

```
PRINT #1,"$1MG3" ; CHR$(&HD);
```

This code performs interpolation pattern 3.

**Note**

- This command is enabled when the command “EM”=1.
- Do not change the speed parameter during interpolation.
Action: This command sets the interpolation pattern that the command “MG” uses. The combination of linear/circular interpolation is set by the position pulse set by the command “MA”. You can set a maximum of 16 patterns and each pattern has up to 15 intermediate points.

Format:

(Ⅰ) $B# M P mn , tn mv pn1 ⊗
Sets to move to the next intermediate point with a linear interpolation.

- mn.. interpolation pattern (0 to F): 16 patterns
- tn.. intermediate point number (1 to F): 15 points
- mv.. interpolation method
  - S : linear interpolation absolute move
  - S+ : linear interpolation incremental move
- pn1.. position number of intermediate point: 000 to 999

(Ⅱ) $B# M P mn , tn mv pn1 , pn2 ⊗
Sets to move to the next intermediate point by circular interpolation.

- mn.. interpolation pattern (0 to F): 16 patterns
- tn.. intermediate point number (1 to F): 15 points
- mv.. interpolation method
  - C1 : CW circular interpolation absolute move
  - C1+ : CW circular interpolation incremental move
  - C2 : CCW circular interpolation absolute move
  - C2+ : CCW circular interpolation incremental move
  - C3 : intermediate position specification circular interpolation absolute move
  - C3+ : intermediate position specification circular interpolation incremental move
- pn1.. position number of intermediate point: 000 to 999
- pn2.. position number of center of a circle ※Note1: 000 to 999

※Note1: pn2 will be intermediate position if the intermediate position specification circular interpolation is specified.

(Ⅲ) $B# M P C mn ⊗
Erases specified interpolation pattern mn.

- mn.. interpolation pattern (0 to F): 16 patterns

(Ⅳ) $B# M P mn , tn ⊗
Queries tnth position of specified interpolation pattern mn.

- mn.. interpolation pattern (0 to F): 16 patterns
- tn.. intermediate point number (1 to F): 15 points
Reply :

In case of (Ⅰ, Ⅱ, Ⅲ)

> 

In case of (Ⅳ)  > $ B# m v pn1 ⊆ 

Reply in case of linear interpolation

m v . . interpolation method
S : linear interpolation absolute move
S+ : linear interpolation incremental move

> $ B# m v pn1 . pn2 ⊆ 

Reply in case of circular interpolation

m v . . interpolation method
C1 : CW circular interpolation absolute move
C1+ : CW circular interpolation incremental move
C2 : CCW circular interpolation absolute move
C2+ : CCW circular interpolation incremental move
C3 : intermediate position specification circular interpolation absolute move
C3+ : intermediate position specification circular interpolation incremental move
pn 1: position number of intermediate point 000 to 999 3 digits
pn 2: position number of center of a circle※Note2 000 to 999 3 digits
(※Note2 pn2 will be intermediate position if intermediate position specification circular interpolation is specified.)

> 

Reply when interpolation method is not specified.

Example
(1) PRINT #1, "$1MP3, 1S 33"; CHR$(&HD);
This code enables linear interpolation absolute move to intermediate point 1 of interpolation pattern 3.

(2) PRINT #1, "$1MP3, 2C1 34, 35"; CHR$(&HD);
This code enables CW circular interpolation absolute move to intermediate point 2 of interpolation pattern 3.

(3) PRINT #1, "$1MP3, 3C3 36, 37"; CHR$(&HD);
This code enables sequential intermediate position specification circular interpolation absolute movement to intermediate point 3 of interpolation pattern 3.
The interpolation pattern 3 of command "MG" performs (1) through (3) sequentially.

Note
・ This command is enabled when the command "EM"=1.
Action: This command performs linear interpolation to the position specified by the command “MA”.
(Absolute move)
The speed parameter should be specified for motor 1.

Format:

(Ⅰ) \$ B\# M S p n
Performs linear interpolation to the pp position of pn.
\[p n\ldots\] Position number (000 to 999) up to 1,000 points
Sets by 3-digit number assuredly.

(Ⅱ) \$ B\# M S
Reads the pp of subsequent pn of the last move pn by the command “MS” and moves a motor to the position.

(Ⅲ) \$ B\# M S N
Queries the position number of command “MS” to move subsequently.

Reply:

In case of (Ⅰ, Ⅱ)
> 
In case of (Ⅲ)
> \$ B\# 0 0 0 0 0 
decimal 5 digits
【DEFAULT=00000】
Current position pulse number

Example

• PRINT #1,” $1 MS 0 0 2”: CHR$(&hD);
This code instructs RC-234 with body number 1 to perform linear interpolation to the absolute position of pn002 set by the command “MA”.

Note
• This command is enabled when the command “EM”=1.
• Do not change the speed parameter during interpolation.
# Command Reference

## MS+ Command

**Action**
This command incrementally moves motor 1 and motor 2 by certain distance set by the command “MA” relative to current position with a linear interpolation. (Incremental Move)
The speed parameter should be specified for motor 1.

**Format**

### (I)

$B# M S + pn$

This command incrementally moves by the position data of pn with a linear interpolation.

- **pn**... Position number (000 to 999) up to 1,000 points
- Sets by 3-digit number assuredly.

### (II)

$B# M S +$

Reads the position number of subsequent pn of the last move pn by the command “MS+” and moves by the position data.

**Reply**

> (For both I and II)

**Example**

```
PRINT #1, "$1 MS+ 2" ; CHR$(&HD(;
```

This code instructs the RC-234 with body number 1 to incrementally move by the position data set by the command “MA” with a linear interpolation. (Incremental move)

**Note**

- This command is enabled when the command “EM”=1.
- Do not change the speed parameter during interpolation.
Action: This command enables/disables a function that changes a motor speed while the motor is in motion. The speed parameter should be separately specified for motor 1 and 2.

Format:

(I) $ B# N PN sp pj

Specifies speed and position data at each speed-change point (PN).

- **PN** . . Point number (1 to 9, A) up to 10 points
- **sp** . . Speed data (decimal up to 5-digit) to which the speed is changed at each point (0 to 32,767)
- **pj** . . Remainder pulses (decimal up to 8-digit) to the destination (0 to 16,777,215)

   【sp, pj : DEFAULT=0, 0】

(II) $ B# N S

Enables the "speed-change" function according to the data specified by the format (I).

The function of this format is only applied to the next command that starts a motor rotation. For example, if you have executed a command such as "B" with this function and want to perform the function again for another move command, you need to send this format again to enable the function. For more information on the function, see the description below.

(III) $ B# N C

Clears all data on this function specified by the format (I).

(IV) $ B# N R

Disables the function.

Note: Once a move command, such as "B" is executed, the function is automatically disabled.

(V) $ B# N

Queries whether the function is enabled or disabled.

(VI) $ B# N PN

Queries the speed data and remainder pulses at a specified speed-change point.

   **PN** . . Point number (1 to 9, A) up to 10 points
Reply: In case of (Ⅰ) to (Ⅳ)

In case of (Ⅴ)

$  
B#  
DC  
1: The function is enabled.
0: The function is disabled.

In case of (Ⅵ)

$s  
B#  
s p  
.  
p j  
$  
sp.. Speed data (5-digit decimal)
$p j$. Remainder pulses (8-digit decimal)

Description

- The function enabled by the format (Ⅱ) will be applied to the following move commands when they are executed:

  Command "1", "3", "4", "5", "B", "B+ (−)"

To use the speed-change function, follow the procedure below:

1. Use the format (Ⅰ) to specify the speed data and the remainder pulses to the destination at each speed-change point.
   (If you want to override a previous settings, use the format (Ⅲ) to clear data before sending the format (Ⅰ).)
   Speed-change points can be registered up to 10 points. However, all of the points are not necessarily specified.
   An example is the below settings in which three points are specified.

<table>
<thead>
<tr>
<th>PN</th>
<th>sp</th>
<th>pj</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8,000</td>
<td>20,000</td>
</tr>
<tr>
<td>2</td>
<td>4,000</td>
<td>10,000</td>
</tr>
<tr>
<td>3</td>
<td>2,500</td>
<td>5,000</td>
</tr>
</tbody>
</table>

2. When the function is enabled by the format (Ⅱ) and one of the move commands mentioned above is issued, the motor starts rotating and changes its speed from low to high with acceleration ramp. When the remainder pulses to the destination becomes equal to or less than pj specified for speed-change point 1 (PN = 1) while the motor is rotating at the high speed, the motor changes its speed to the value determined by sp for the point 1 and keeps rotating.
   The same process will be repeated for the rest of the speed-change point 2, 3, and so on.
   When all of the points are not specified, after the speed is changed to the defined value for the last speed-change point, it keeps the same speed until the motor reaches the destination.
   Upon the completion of the move, the speed-change function is disabled. During the process, the function changes the speed temporarily and does not affect the settings of "OH", "OL", or other speed-related commands.
**Command N**

If you issue an "A00125000" to designate the position data of command "A" as 25,000, followed by sending the command "N" in the format (I) with parameters in the previous table; performing the format (II) to enable the function; and executing the command "B001", the motor will move and change its speed as in the following figure, where the "OH" parameter is assumed to 5,000 (default value):

```
The acceleration/deceleration time (S) at each speed-change point is calculated by the following equation:

\[ S = \frac{|A-B|}{4,915,200} \times OS \text{ (sec)} \]

A: Speed data (SP) before the speed is changed.
B: Speed data (SP') after the speed is changed.
OS: Setting of "OS".

Example

```
PRINT #1, "$1A00125000"; CHR$(&HD;
PRINT #1, "$1NC"; CHR$(&HD;
PRINT #1, "$1N18000,20000"; CHR$(&HD;
PRINT #1, "$1N24000,10000"; CHR$(&HD;
PRINT #1, "$1N320032500,5000"; CHR$(&HD;
PRINT #1, "$1NS"; CHR$(&HD;
PRINT #1, "$1B001+"; CHR$(&HD;
PRINT #1, "$1NS"; CHR$(&HD;
PRINT #1, "$1B0012"; CHR$(&HD;
```

The series of the commands instruct the RC-234 with body number 1 to make a back-and-forth motion for the attached motor with changing the speed as shown in the above figure.
10. Command Reference

Command \textit{N}

**Important notice 1**

Set the remainder pulses at each speed-change point, 1 to 9 and A in such a manner that a remainder pulse with the bigger pulse count is assigned to a point with the smaller number.

In this function, the pulse set for the change point, 1 to 9 and A with the smallest number is firstly compared with the actual remainder pulses to a destination, secondly with the next smallest number, and so on.

When a motor begins to move, the pulse set for speed-change point 1 is compared with the actual remainder pulse to a destination. If the actual pulse is equal to or smaller than the setting, the RC-234 changes motor speed and start a comparison with the speed-change point 2.

If the remainder pulse for the point 1 is 0 (default value), the setting for the point 2 is never used and the motor will reach the destination without changing its speed. The reason is that the actual remainder pulse never becomes smaller than the setting for the point 1 where the value is 0.

Additionally, if the total pulse of a move from a starting point to a destination is smaller than the remainder pulse set for the point 1, the motor moves at the speed determined by the point 1 settings from the beginning.

**Important notice 2**

When changing speed using the command "N" please set the remainder pulses not to start speed-change during accel/deceleration. (not to overlap accel./decel. period.)

If speed-change is executed during accel/deceleration, the command "N" may lead to an unexpected move.

Also, the pulse number to accelerate/decelerate (\(P_g\)) is calculated by the following equation:

\[
P_g = \frac{|(A)^2 - (B)^2|}{32768 \times O X} \times O S \text{ (pulses)}
\]

A: Speed data (SP) before the speed is changed.
B: Speed data (SP') after the speed is changed.
OS: Setting of "OS".
Action: This command specifies an S-curve rate during acceleration/deceleration periods. This setting should be separately designated for each motor. For the details of S-curve rate and information on other speed parameters, see Chapter 7 “Stepping Motor Speed Adjustment”.

Format: (Ⅰ) $B#O C d t \ldots$  
S-curve rate during acceleration/deceleration periods (%)  
(decimal up to 3-digit: 1 to 100)  
【DEFAULT=0】  

(Ⅱ) $B#O C$  
Queries the setting of the S-curve rate. The ratio should be separately queried for each motor.

Reply: In case of (Ⅰ) >

In case of (Ⅱ) > $B#O O O$  
3-digit decimal  
Setting of the S-curve rate

Description

- The below figure shows how a speed pattern varies with S-curve rates or "OC" parameters.

S-CURVE = 0% (Trapezoidal)  
S-CURVE = 50%  
S-CURVE = 100%

S-curve rate = 100 x T x 2Ts (%)  

When an S-curve rate is 0 (default value), the speed pattern is a legacy trapezoidal acceleration/deceleration model.

When an S-curve rate is other than 0, a speed pattern is made S-shaped on acceleration/deceleration. The S-curve rate determines the ratio between a total acceleration/deceleration time (T) and S-shaped time (Ts). When a S-curve rate is 100%, Ts is half of T. Ts period can not exceed T x 0.5.
Note

- If a high speed setting (command "OH") is smaller than a low speed ("OL"), an unexpected move may occur on an S-curve acceleration/deceleration.

- In interpolation control, the speed parameter should be specified for motor 1.
### Command Reference

#### Action
This command specifies a high speed. This setting should be separately designated for each motor. For the details of a high speed and information on other speed parameters, see Chapter 7 "Stepping Motor Speed Adjustment".

#### Format

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(I)</td>
<td>$ B# O H h s</td>
</tr>
<tr>
<td>(II)</td>
<td>$ B# O H</td>
</tr>
<tr>
<td>(III)</td>
<td>$ B# O H I i d</td>
</tr>
<tr>
<td>(IV)</td>
<td>$ B# O H D d d</td>
</tr>
</tbody>
</table>

#### Reply
In case of (I, III, IV) >

In case of (II) > $ B# O O O O O 5-digit decimal Setting of a high speed
**Description**

- Changing high speed data while a motor is in motion.

As shown in the above figure, if the high speed data is changed to $OH'$ at the time of $t_0$ while a motor is running at a high speed ($f_H$) determined by $OH$, the motor speed alters from $f_H$ to $f_H'$. This means that the speed pattern is changed from the dotted line (original curve) to the solid line at $t_0$.

where $T_1$ and $T_2$ are calculated by the following equation:

$$T_1 = \frac{|OH' - OH|}{4,915,200} \times OS \text{ (second)}$$

$$T_2 = \frac{|OH' - OL|}{4,915,200} \times OS \text{ (second)}$$

$f_H$ is derived from the following equation:

$$f_H = \frac{OH}{OX} \times 300 \text{ (pps)}$$

**Note**

- To specify different data for motor 1 and 2, use the command "F" to designate a desired motor before issuing the command "OH".

- High speed data can be changed during motion. If the motor is running at high speed, an execution of command "OH" changes the motor speed to the new speed with acceleration/deceleration ramp.

- In interpolation control, the speed parameter should be specified for motor 1.
**Command Reference**

**OL**

<table>
<thead>
<tr>
<th>Command</th>
<th>O L</th>
</tr>
</thead>
</table>

**Action**: This command specifies a low speed at which a motor starts to move. Home-search functions are also performed at this speed. (Low speed)

This setting should be separately designated for each motor.

For the details of the low speed and information on other speed parameters, see Chapter 7 "Stepping Motor Speed Adjustment."

**Format**:

(Ⅰ) $ B# O L I s$

Specifies a speed when a motor runs at a "low speed".

- **ls**... Low speed data (decimal up to 5-digit; 1 to 32,767)

  【DEFAULT=500】

(Ⅱ) $ B# O L$

Queries the setting of "OL".

- The query should be made separately for each motor (1 or 2).

(Ⅲ) $ B# O L I i d$

Increments the low speed parameter by **id**.

- **id**... Increment of quantity (decimal up to 5-digit: 1 to 32,766)

(Ⅳ) $ B# O L D d d$

Decrements the low speed parameter by **dd**.

- **dd**... Decrement of quantity (decimal up to 5-digit: 1 to 32,766)

**Reply**: In case of

(Ⅰ, Ⅲ, Ⅳ) >

In case of (Ⅱ) > $ B# O O O O O$

5-digit decimal

Setting of the low speed
Description
- Changing low speed data while a motor is in motion.

As shown in the above figure, if the low speed data is changed to OL' at the time of t0 while a motor is running at the low speed (fL) determined by OL, the motor speed alters from fL to fL'. This means that the speed pattern is changed from the dotted line (original curve) to the solid line at t0.

where T is calculated as follows:

\[ T = \frac{|O L' - O L|}{4,915} \times O S \text{ (second)} \]

fL is derived from the following equation:

\[ f_L = \frac{O L}{O X} \times 300 \text{ (pps)} \]

Note
- To specify different data for motor 1 and 2, use the command "F" to designate a desired motor before issuing an "OL" command.

- Low speed data can be changed during motion.
  If the motor is running at low speed, the execution of command "OL" changes the motor speed with acceleration/deceleration ramp.

- In interpolation control, the speed parameter is specified for motor 1.
Command Reference

Action: This command sets acceleration data used in acceleration/deceleration motion. This setting should be separately designated for each motor. For the details of the acceleration data and information on other speed parameters, see Chapter 7 "Stepping Motor Speed Adjustment".

Format:

(I) \$ B# O S s s

Sets acceleration data.

ss... Acceleration data (decimal up to 5-digit; 1 to 32,767)

【DEFAULT=300】

(II) \$ B# O S A s s

Sets acceleration data when accelerating.

ss... Acceleration data (decimal up to 5-digit; 1 to 32,767)

【DEFAULT=300】

(III) \$ B# O S B s s

Sets acceleration data when decelerating.

ss... Acceleration data (decimal up to 5-digit; 1 to 32,767)

【DEFAULT=300】

(IV) \$ B# O S

Queries the setting of acceleration data.
The query should be made separately for each motor (1 or 2).

(V) \$ B# O S A

Queries acceleration data when accelerating.
The query should be made separately for each motor (1 or 2)

(VI) \$ B# O S B

Queries acceleration data when decelerating.
The query should be made separately for each motor (1 or 2)

(VII) \$ B# O S I i d

Increments acceleration data by id.
i d... Increment of quantity (decimal up to 5-digit: 1 to 32,766)

(VIII) \$ B# O S D d d

Decrements acceleration data by dd.
d d... Decrement of quantity (decimal up to 5-digit: 1 to 32,766)
The acceleration data determines the slant of the heavy line in the figure below, which is an acceleration/deceleration ramp.

**Description**

- The acceleration data determines the slant of the heavy line in the figure below, which is an acceleration/deceleration ramp.

\[
\text{Acceleration rate} = \frac{1,474,560,000}{\text{O} \times \text{O} \times \text{S}} \quad (\text{pps/sec})
\]

T (acceleration/deceleration time) is calculated as follows:

\[
T = \frac{|A - B|}{24,576 \times (200 - OC)} \times \text{O} \times \text{S} \quad (\text{second})
\]

where

- A: The speed data after acceleration/deceleration is completed.
- B: The speed data before acceleration/deceleration is begun.

See the Note in the "OH", "OL", and "OC" section.
Command  OS

Note

- To specify the several data for motor 1 and 2, use the command "F" to designate a desired motor before issuing the command "OS".

- The setting data of command “OS” overwrites the data set by the commands “OSA” and “OSB”. Also, if acceleration/deceleration time is specified separately, the data set by the format (Ⅱ) is returned when acceleration/deceleration data is queried by the format (Ⅳ).

- To specify acceleration/deceleration time with command “OSA” and “OSB” separately, please set the deceleration time to twice of acceleration time or less.

- An acceleration/deceleration time is determined not only by OS, but also by OH, OL, and OC. If you want to change one of the latter three parameters and keep an acceleration/deceleration time at a constant value, modify an OS parameter.

- Changing acceleration data can be performed even if a motor is rotating. However, modifying the data while a motor is in motion is not recommended because an improper setting may lead to the stall of the motor.

- Attempting to increment/decrement the acceleration data by the format (Ⅶ) or (Ⅷ) with parameters that set data beyond a limit range (1 to 32,767) results in a command error, and the data will not be changed.

- In interpolation control, the speed parameter is specified for motor 1.
**Command O X**

**Action**: This command sets a multiplication factor to determine the speed of a motor.
- When command “EM”=0 (alternate 2-axis control), the setting is shared between the motor 1 and 2. (Common data for motor 1 and 2).
- When command “EM”=1 (simultaneous 2-axis control), this setting should be separately designated for each motor.

For the details of the multiplication factor and information on other speed parameters, see Chapter 7 "Stepping Motor Speed Adjustment".

**Format**:

(I) $ B# O X x s

Sets a multiplication factor.

\[ x \times s \] Multiplication factor (decimal up to 5-digit; 2 to 16,383)

【DEFAULT=300】

(II) $ B# O X

Queries the data set by the format (I).

**Reply**: In case of (I) >

In case of (II) > $ B# OOOOO 5-digit decimal

Setting of the multiplication factor

**Description**

- Frequency multiplication is determined by the following equation:

\[
\text{Frequency multiplication} = \frac{300}{O \times X}
\]

**Important notice**

An effective value of the OX is multiple of 4, no matter what value is given as the OX data.

Though OX data can be designated in the range of 2 to 16383, the data will be converted to a multiple of 4 and the converted value is used to determine a frequency multiplication.

If you designate 30, for example, as OX data to obtain a frequency multiplication of 10 (= 300/30), the 30 is converted into 28 and the effective frequency multiplication will be 10.7 (= 300/28).

This specification was designed to keep a backward compatibility with our RC-231.
10. Command Reference

**Command** O X

<Difference between OX setting data and Effective OX data>

<table>
<thead>
<tr>
<th>Given OX data</th>
<th>Effective OX data</th>
<th>Multiplication factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 to 7</td>
<td>4</td>
<td>7 5</td>
</tr>
<tr>
<td>8 to 1 1</td>
<td>8</td>
<td>3 7.5</td>
</tr>
<tr>
<td>1 2 to 1 5</td>
<td>1 2</td>
<td>2 5</td>
</tr>
<tr>
<td>3 0 0 to 3 0 3</td>
<td>3 0 0</td>
<td>1</td>
</tr>
</tbody>
</table>

Note

- Changing OX data can be performed even if a motor is in motion. However, issuing the command OX with a different value from the current data immediately changes the motor speed without ramping acceleration/deceleration. This may cause the motor to stall. Therefore change an OX data when a motor is not in rotation.

- OX data set to extremely big number, which leads to a small frequency multiplication, may cause a motor to run at excessively low speed and the motor may appear to stop when a move command is executed.

- In interpolation control, the speed parameter is specified for motor 1.
Action : This command activates a specified bit of the output port for a specified period of time.

Format : $B#P BT TIME

- BT.. Bit number to be specified (0-7).
- TIME.. Period of time for which an output bit is activated (0 to 32,767 (Approx. 54 minutes)).
  The wait time is specified in terms of 100 milliseconds, hence the actual period is \textit{time} multiplied by 100 msec.

Reply : >

Example
- \texttt{PRINT #1, "$1P2300" ; CHR$(&HDA)}
  Activates the bit2 of the output port, and deactivates it after 30 seconds (300 x 0.1sec).

Note
- If this command is applied to a bit that has been already turned on, the bit will be turned off after the period specified by the command.
- Other timer-related commands, such as "T", "GWT", \(\gamma/T\), or \(\gamma/TB\), can not be used when the command "P" is in operation. The reason is that those commands share a single timer resource in RC-234.
10. Command Reference

**Command Reference**

Action: This command sets the mode of counting when an encoder is used. This setting should be separately designated for each motor. The command is exclusively used in an operation mode 1.

Format: (I)  

$ B# P A DT \[  
Sets the mode of counting.  
D T .. 1: Uni-counting.  
2: Bi-counting.  
4: Quadrature-counting.  
【DEFAULT=1】

(II)  

$ B# P A \[  
Queries the mode of counting. The query should be made separately for each motor (1 or 2).

Reply: In case of (I)  

In case of (II)  

Description:
- The A and B phase signals from an encoder should be connected to a RC-234 at EA and EB terminal, respectively.

- What is the "mode of counting?"
In the following description, the resolution of an encoder is assumed to be 1,000 pulse/rev. The encoder generates 1,000 pulses per one rotation. A RC-234 can recognize the signal as 1,000, 2,000, or 4,000 pulses/rev because the encoder output is comprised of two phases: A and B. The mode of counting determines in which way a RC-234 treats the encoder output, 1,000, 2,000, or 4,000 pulses/rev. When PA data is set to 2, the position of a motor is controlled with double the precision compared to PA = 1. (The actual precision depends on the accuracy of the encoder.) Similarly, PA = 4 achieves quadruple the precision of PA=1.
Command P A

Note

・Position data of command "2" or "A" should be given in terms of encoder pulse rather than of actual output pulse from RC-234.
For example, if you use an encoder with the resolution of 1,000 pulse/rev with PA=2 and want to rotate a motor one revolution, you set 2,000 (1,000 x 2) as the position data of "2" or "A", etc. (The number to be specified is NOT of actual pulses that are transmitted to a drive.)

・When using the command "PA", use the command "PB" as well to specify the ratio between an encoder pulse and a motor pulse.

・To set different data for the motor 1 and 2, use the command "F" to designate a desired motor before issuing the command "PA".

・Position data of commands such as "2" or "A" must meet the following inequality.

\[
\left| \frac{\text{Position data} \times \frac{\text{PB data}}{\text{PA data}}}{\text{PB data}} \right| < 8,388,608
\]

・See the section "Mode 1 limitation" in Chapter 7 "Stepping Motor Speed Adjustment".
Action: This command sets the ratio between motor resolution and encoder resolution when an encoder is used. This setting should be separately designated for each motor. The command is exclusively used in an operation mode 1.

Format: (I) $B#PBra$
Sets the ratio between motor and encoder.
ra... Resolution ratio (1-255)
【DEFAULT=10】

(II) $B#PB$
Queries the setting of the resolution ratio between motor and encoder. The query should be made separately for each motor (1 or 2).

Reply: In case of (I) >

In case of (II) > $B#OOO$
3-digit decimal
Resolution ratio between motor and encoder

Description
- The command “PB” specifies the resolution ratio between motor and encoder.

\[
\text{Resolution ratio (PB)} = \frac{\text{Motor resolution}}{\text{Encoder resolution}}
\]

Example) Resolution of motor: 200 steps/rev. 200
Resolution of motor drive: Microstep size of 50 50
Resolution of encoder: 1,000 pulses/rev. 1,000

In this case, the setting should be as follows:

\[
P B = \frac{200 \times 50}{1,000} = 10
\]

As the above equation shows, the PB setting is independent from the mode of counting ("PA") and determined only by the resolution of an encoder and by pulse number that makes one revolution of a motor.
Command  PB

Note
- Use the command "PA" when you set the mode of counting.

- To set different data for motor 1 and 2, use the command "F" to designate a desired motor before issuing the command "PB".

- Position data of commands such as "2" or "A" must meet the following inequality.

\[
\left| \frac{\text{Position data} \times \text{PB data}}{\text{PA data}} \right| < 8,388,608
\]

- See the section "Mode 1 limitation" in Chapter 7 "Stepping Motor Speed Adjustment".
Command Q

Action: This command queries whether a stall of a motor has been detected.

Format: $ B# Q

Reply: $ B# DT.. 0: No stall has been detected. 1: A stall has been detected.

Description
- Once a stall is detected, RC-234 does NOT accept or execute move-related commands other than home-search commands. To reset the condition, use one of the following procedures.

1. In operation mode 0, 2, execute one of home-search commands.
2. In operation mode 1, execute the command "RD" or a home-search command.
**Command Q**

**Action**: This command sets the on-off period of a stall sensor. This setting should be separately designated for each motor. This command is exclusively used in an operation mode 0.

**Format**

1. $(\text{Ⅰ})$ 
   \[
   \$ \ B# \ Q \ s \ l \ \&
   \]
   Sets the on-off period of a stall sensor in units of pulse.
   
   $s\ l...$ On-off period (100 to 65,535)
   
   【DEFAULT=400】

2. $(\text{Ⅱ})$ 
   \[
   \$ \ B# \ Q \ D \ \&
   \]
   Queries the data set by the format $(\text{Ⅰ})$.

3. $(\text{Ⅲ})$ 
   \[
   \$ \ B# \ Q \ D \ MT \ \&
   \]
   Queries the data set by the format $(\text{Ⅰ})$ for a specified motor.
   
   $M\ T...1\ or\ 2$

**Reply**: In case of $(\text{Ⅰ})$  

In case of $(\text{Ⅱ}, \text{Ⅲ})$  

\[
> \ $ \ B# \ \& \ OOOOO \ \&
\]

5-digit decimal

Current on-off period.

**Description**

- Stall sensor devices may be linear or rotary in shape as shown below.

**<Stall detection with a rotary device>**

In the figure, a rotary disk for stall detection is attached to a motor axis.
Command Q

< Stall detection with a linear device >

Stall Sensor
(Please set to move according to the motor rotation.)

To EZ terminal

On-Off Period

In this figure, the stall sensor moves linearly along the comb-shaped device in synchronization with the rotation of a motor.

The output signal from the sensor must be connected to a RC-234 at the EZ terminal.

(Remarks)
On-Off period: You set this data with the format (I) according to the design of the stall detection device.

ON,OFF: Both on and off state must be kept for at least 5 msec. Otherwise stall detection may fail.

Important notice

Use the command "EA" to specify the sensor logic of the EZ input so that the sensor is ON at the home position (where the position counter = 0).

< Sample timing chart with On-Off period set to 500 >

Note
- To perform stall detection, the command "QS" should be performed as well.
  See the command " Ø Q " as well.
Command Reference

**Action**
This command enables a “vibration suppression” function to reduce vibration when stopping by adding 1 pulse of reverse rotation and 1 pulse of forward rotation just after completion of command. This setting should be separately designated for each motor.

**Format**

(I) \( \$ \ B# \ Q \ B \ RT \ FT \)  
Sets the output timing of additional pulses.

- RT.. Timing for reverse rotation (0 to 65,535)
- FT.. Timing for forward rotation (0 to 65,535)
- Range: 0 to approx. 0.14 seconds  
  (setting value 1 = approx. 2.17 \( \mu \) sec)  
- **DEFAULT=0**

(II) \( \$ \ B# \ Q \ B \)  
Queries the output timing of additional pulses.

**Reply**

In case of (I) \( > \)

In case of (II) \( > \ $ \ RT \ FT \)  
- RT.. timing for reverse rotation (0 to 65,535)  
- FT.. timing for forward rotation (0 to 65,535)  
- Range: 0 to approx. 0.14 seconds  
  (setting value 1 = approx. 2.17 \( \mu \) sec)

![Diagram](Example of movement in the CW direction)

The dotted lines are added pulses by the vibration suppression function.

**Note**

- When the time for both reverse and forward rotation are anything except 0, the vibration suppression function is turned ON.

- In interpolation operation, this function is not available.
Action: This command outputs pulses by the designated offset pulse number just before the execution of the command.
The backlash offset is performed every time the direction of motion is changed and the slip offset is performed regardless of the direction of motion just before the execution of the command.
This setting should be separately designated for each motor.

Format:

(I) $B# QB BR MD , DATA

Enables/disables the backlash/slip offset

MD: 0: Disables the offset output. 【DEFAULT】
1: Enables the backlash offset.
2: Enables the slip offset.

DATA: Offset pulse number (0 to 4,095) 【DEFAULT=0】

(II) $B# QB BR 0

Disables backlash/slip offset.

(III) $B# QB BR O L cs

Sets the speed data when the offset is enabled.

cs: Speed data decimal number up to 5 digits (1 to 32,767)

When the value of the command “OX” is 300, the unit is 1pps.
【DEFAULT=500】

(IV) $B# QB BR

Queries the status of the backlash/slip offset setting.

(V) $B# QB BR OL

Queries the speed when the offset is enabled.

Reply:

In case of (I, II, III) >>

In case of (IV) >> $B# MD , DATA

MD: 0: Disables the offset output.
1: Enables the backlash offset.
2: Enables the slip offset.

DATA: Offset pulse number (5-digit decimal number)

In case of (V) >> $B# CS

cs: Speed data (5-digit decimal number)
10. Command Reference

Command QBR

Descriptions
・The action after moving in the CCW direction is as follows:

<table>
<thead>
<tr>
<th>Backlash offset</th>
<th>→CW move</th>
<th>→CW move</th>
<th>→CCW move</th>
<th>Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>Disabled</td>
<td>Enabled</td>
<td></td>
<td>When the direction is changed</td>
</tr>
<tr>
<td>Slip offset</td>
<td>Enabled</td>
<td>Enabled</td>
<td>Enabled</td>
<td>Regardless of the direction</td>
</tr>
</tbody>
</table>

Note
・Please set the number of offset pulses to be more than the pulse data set by the command “Θ S”.

・This command can't be used in circular interpolation and linear interpolation. (Please disable the backlash/slip offset.)

・This command can't be used together with the command “QJ” (Adjust function) for mode 1 at the same time.

・The slip offset is a function to adjust the pulse deficiency by slip in case of the cost effective positioning for when fine positioning is not necessary. (when performing a table drive with rollers, etc.)
  It's an adjustment only for deficiency not for excess.
**Command Reference**

<table>
<thead>
<tr>
<th>Command</th>
<th>Q E</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0</td>
<td>E1</td>
</tr>
<tr>
<td>☑️</td>
<td>☑️</td>
</tr>
<tr>
<td>During motion</td>
<td>EEPROM</td>
</tr>
<tr>
<td>✗</td>
<td>☑️</td>
</tr>
</tbody>
</table>

**Action:** This command sets the maximum limit of the difference between a pulse count and an encoder count.

This command exclusively used in an operation mode 0 (when the command "QSE" is used) and a mode 1.

When command "EM"=0 (alternate 2-axis control), the setting is shared between the motor 1 and 2. (Common data for motor 1 and 2).

When command "EM"=1 (simultaneous 2-axis control), this setting should be separately designated for each motor.

**Format:**

(Ⅰ)

```
$ B# Q E DT [ ]
```

Sets the maximum limit of the difference in terms of actual output pulses.

D T . . 0 to 9

【DEFAULT=4】

<table>
<thead>
<tr>
<th>DT</th>
<th>Maximum limit of the difference</th>
<th>DT</th>
<th>Maximum limit of the difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>±4 pulses</td>
<td>5</td>
<td>±128 pulses</td>
</tr>
<tr>
<td>1</td>
<td>±8 pulses</td>
<td>6</td>
<td>±256 pulses</td>
</tr>
<tr>
<td>2</td>
<td>±16 pulses</td>
<td>7</td>
<td>±512 pulses</td>
</tr>
<tr>
<td>3</td>
<td>±32 pulses</td>
<td>8</td>
<td>±1,024 pulses</td>
</tr>
<tr>
<td>4</td>
<td>±64 pulses</td>
<td>9</td>
<td>±2,048 pulses</td>
</tr>
</tbody>
</table>

(Ⅱ)

```
$ B# Q E [ ]
```

Queries the data set by the format (Ⅰ).

**Reply:**

In case of (Ⅰ)

```
>
```

In case of (Ⅱ)

```
> $ B# DT [ ]
```

D T . . See the table above.

**Description**

- RC-234 monitors an actual output pulse and an input pulse of an encoder. If the difference between them exceeds the QE setting, the controller sets the stall error flag.

- What is the "limit of the difference between a pulse count and an encoder count?"

  The following explanation is in the case where we control a motor with an encoder, assuming PA = 2 and PB = 10.

  When a command that rotates a motor 1,000 pulses has actually moved it 990 pulses, the difference between a pulse count and an encoder count is 10 (= 1,000 - 990) pulses. (In this calculation, all the figures are counted in terms of an encoder pulse.)

  Conversion the unit from the encoder pulse to an actual output pulse makes the 1,000 pulses into 200 (= 10 x 2 (PA) x 10 (PB)) pulses.

  The 200 pulses are called the "difference between a pulse count and an encoder count". If the 200 pulses exceed the data specified by the command "QE", RC-234 recognizes that a stall error has occurred.
(Remarks)
Though the above explanation describes that the stall detection is performed after the motor has been stopped, the controller also checks for a stall error every 50 msec while a motor is running.

Recommended settings
The recommended setting for a "QE" data is pulses equivalent to 7.2 degrees of rotation, whether the motor is 2-phase or 5-phase. The reason is that a deviation due to a stall should be equal to or greater than 7.2 degrees in both 2-phase and 5-phase motor.

Example) When you use a standard 1.8° stepper motor (2-phase) with 50 of microstep size, the pulses equivalent to 7.2° rotation is 200 (= 360°/1.8° x 50 x 7.2°/360°). So the recommended "QE" data is 5 (±128 pulses).

However, RC-234 may mistakenly detect a stall error with the recommended setting if the accuracy of an encoder, motor, and/or drive is not appropriate. In such a case, change the QE data to a bigger number.
**Action**: This command enables an “adjustment function”, which corrects a motor position after the motor completes a move.

This command is exclusively used in an operation mode 1.

**Format**:

(Ⅰ) $ \text{B# Q J M2 M1}  \quad \text{Enables/disables the adjustment function for each motor.}

M 1... Setting of the adjustment function for motor 1.

M1 = 0: Disabled  【DEFAULT】

1: Enabled.

M 2... Setting of the adjustment function for motor 2.

M2 = 0: Disabled  【DEFAULT】

1: Enabled.

(Ⅱ) $ \text{B# Q J}  \quad \text{Queries the current settings for motor 1 and 2.}

(Ⅲ) $ \text{B# Q J R}  \quad \text{Disables the adjustment function for both motor 1 and 2.}

(Ⅳ) $ \text{B# Q J T TIME}  \quad \text{Sets a delay time when the adjustment function starts after a motor move is completed.}

When command “EM”=0 (alternate 2-axis control), the setting is shared between the motor 1 and 2. (Common data for motor 1 and 2).

When command “EM”=1 (simultaneous 2-axis control), this setting should be separately designated for each motor.

TIME... Delay time (1 to 999)

Time base 10 msec  (TIME × 10 msec)

【DEFAULT=3】

(Ⅴ) $ \text{B# Q J T}  \quad \text{Queries the delay time set by the format (IV).}

(Ⅵ) $ \text{B# Q J A pp}  \quad \text{Sets the limit of adjusting pulse. If RC-234 can not complete the correction of a motor position after it outputs the adjusting pulses specified by this format, it sets a limit error flag. (See the Description below for detail.)}

When command “EM”=0 (alternate 2-axis control), the setting is shared between the motor 1 and 2. (Common data for motor 1 and 2).

When command “EM”=1 (simultaneous 2-axis control), this setting should be separately designated for each motor.

pp... Limit of adjusting pulse (1 to 65,535)

【DEFAULT=800】

(Ⅶ) $ \text{B# Q J A}  \quad \text{Queries the data set by the format (VI).}
Command \texttt{Q J}

(\text{Ⅶ}) \begin{array}{cccccc}
$ & B# & Q & J & O & p s \\
\end{array}

Specifies the speed while an adjustment function is carried out. If the setting is 0, the speed determined by OL and OX is used. When the command “EM”=0 (in the alternate 2-axis control), the setting is shared between motor 1 and 2. (Common data for the motor 1 and 2) When the command “EM”=1 (in simultaneous 2-axis control), the setting should be separately designated for each motor.

\texttt{ps}\ldots Speed while the adjustment function is performed (0 to 16,382).

\[
\text{Actual motor speed} = \text{ps} \times \frac{300}{\text{OX}} \quad \text{(pps)}
\]

【DEFAULT=0】

(\text{Ⅸ}) \begin{array}{cccccc}
$ & B# & Q & J & O & p s \\
\end{array}

Queries the data set by the format (Ⅷ).

Reply: In case of (I, III, IV, VI, VII)

In case of (Ⅱ) \begin{array}{cccccc}
> & $ & B# & M2 & M1 \\
\end{array}

\text{M1}\ldots Setting of adjustment function for motor 1.
\quad \text{M1} = 0: \text{Disabled}
\quad \text{M1} = 1: \text{Enabled}

\text{M2}\ldots Setting of adjustment function for motor 2.
\quad \text{M2} = 0: \text{Disabled}
\quad \text{M2} = 1: \text{Enabled}

In case of (Ⅴ) \begin{array}{cccccc}
> & $ & B# & \text{TIME} \\
\end{array}

\text{TIME}\ldots Delay time (3-digit decimal).
\quad \text{Time base 10 msec} \quad \text{(TIME} \times 10 \text{msec)}

In case of (Ⅶ) \begin{array}{cccccc}
> & $ & B# & \text{p p} \\
\end{array}

\text{p p}\ldots Limit of adjusting pulse (5-digit decimal)

In case of (Ⅸ) \begin{array}{cccccc}
> & $ & B# & \text{ps} \\
\end{array}

\text{ps}\ldots Speed while the adjustment function is performed (5-digit decimal).
Command  Q J

Description
- An “adjustment function” is the function that automatically corrects a motor position after the motor completes the move. If an actual position measured by an encoder differs from a target position specified by the move-command, the correction is made at a low speed.

<The process of the adjustment function>
①. After a motor move initiated by one of move-commands is completed, RC-234 waits for the delay time specified by the format (IV).

②. If an actual position measured by an encoder differs from a target position specified by the move-command, the controller attempts to correct the deviation by rotating the motor at the speed specified by the format (IV) or the speed determined by OL and OX when the data of the format (Ⅷ) is 0. If there is no difference between the target position and the actual position, the adjustment function is finished.

③. When the motor moves beyond the target position, the controller stops it and jumps to Step 4.
If the limit sensor in the direction of the motor move is activated before the motor reaches the target position, the controller sets a limit error flag and finishes the adjustment function. However, if the current position is beyond the target position when the limit sensor is turned on, the controller continues to perform the function in Step 4 rather than sets the error flag.
If the motor does not reach the target position when the controller completes outputting the pulses defined by the format (VI), the controller sets a limit error flag and finishes the function.

④. The controller rotates the motor every 1 pulse in the opposite direction to Step ③ at the speed determined by the format (Ⅷ). When the motor position is equal to or beyond the target position, the function is finished.

Note
- The factory default of the “adjustment function” is Disabled.
- The adjustment function will not be initiated if the difference between a target position and an actual position is bigger than the setting specified by the format (VI).
- The reply for the command “(NULL)” should be “in motion” while an adjustment function is being performed.
- Even if a stall detection function is enabled, the function will not be executed while the adjustment function is running.
- This command can’t be used together with the command “QBR” (the backlash offset, the slip offset) at the same time.
Command: QS

Action: This command enables/disables a stall check function while the motor is in motion.

Format: (I)  
S  B#  Q  S  M2  M1  
Disables/enables a stall check function for each motor.
M1: Stall check for motor 1.
   M1 = 0: Disabled. 【DEFAULT】
          1: Enabled while the motor is in motion.
          2: Enabled whether the motor is in motion or not.※
M2: Stall check for motor 2.
   M2 = 0: Disabled. 【DEFAULT】
          1: Enabled while the motor is in motion.
          2: Enabled whether the motor is in motion or not.※

(II)  
S  B#  Q  S  
Queries whether the stall check is enabled for each motor.

(III)  
S  B#  Q  S  R  
Disables the stall check function for both motor 1 and 2.

(IV)  
S  B#  Q  S  S  DT  
Selects the target axis when a stall is detected by the command “QS”. (When command “EM”=1)
D T: 0: Stops the axis which detected a stall. 【DEFAULT】
       1: Stops the both axes when a stall is detected.

(V)  
S  B#  Q  S  S  
Queries the target axis when a stall is detected.

※The setting of 2 is valid only in an operation mode 1. With the parameter set to 2, the stall check will be performed only for the motor that is selected by the command “F”. So even if the parameter is set to 2 for both motor 1 and 2, the stall check is not executed simultaneously for both motors when they are both still. In other operation modes, the parameter set to 2 is equivalent to the parameter set to 1.
Command Q S

Reply: In case of (Ⅰ, Ⅲ, Ⅳ)

In case of (Ⅱ) M 1 .. M 2 .. M 1, M 2 .. The current status of the stall check. (See the above description.)

In case of (Ⅴ) DT .. DT.. 0: Stops the axis on which a stall was detected. 1: Stops the both axes when a stall was detected.

Description
- Commands related to a stall check vary among operation modes.

<table>
<thead>
<tr>
<th>Operation mode</th>
<th>Commands to configure the stall check function</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode 0</td>
<td>Command “Q&quot;, &quot;Θ Q&quot;, &quot;Q S E&quot;</td>
</tr>
<tr>
<td>mode 1</td>
<td>Command “Q J&quot;, &quot;Q J T&quot;, &quot;Q E&quot;</td>
</tr>
<tr>
<td>mode 2</td>
<td>Command “Q&quot;, &quot;Θ Q&quot;</td>
</tr>
</tbody>
</table>

Example
- PRINT #1, "$1QS1"; CHR$(&HD); Enables a stall check for motor 1, during its rotation, of the RC-234 with body number 1.

Note
- With a servomotor (mode = 2), this command can be utilized to check its motion. (Servomotors never stall.)
- In this case, however, a stall check device should be designed in consideration of the effect of accumulated pulses inherent to servo drives.
- The command "QSS" is enabled when the command “EM”=1 and the cause of stop is queried by the command “9D”.

Enables a stall check for motor 1, during its rotation, of the RC-234 with body number 1.
Action:
This command allows an operation mode 0 to work with an encoder for a stall check. This command is exclusively used in an operation mode 0.

Format:
(I) $ B# Q S E M2 M1
Disables/enables a stall check function for each motor.
M1.. Stall check for motor 1.
   M1 = 0: Disabled. 【DEFAULT】
   1: Enabled.
M2.. Stall check for motor 2.
   M2 = 0: Disabled. 【DEFAULT】
   1: Enabled.

(II) $ B# Q S E
Queries whether the stall check is enabled for each motor.

(III) $ B# Q S E R
Disables the stall check function for both motor 1 and 2.

Reply:
In case of (I, III)
> $ B# M2 M1

In case of (II)
> $ B# M2 M1
M1.. Stall check for motor 1.
   M1 = 0: Disabled
   1: Enabled.
M2.. Stall check for motor 2.
   M2 = 0: Disabled
   1: Enabled.

Description
This command allows motor positions to be managed in an operation mode 0 and stalls to be checked with an encoder.

Specify the settings shown in the table below before using the command "QSE".

<table>
<thead>
<tr>
<th>Command</th>
<th>To specify</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;P A&quot;</td>
<td>The mode of counting when an encoder is used.</td>
</tr>
<tr>
<td>&quot;P B&quot;</td>
<td>The ratio between a motor resolution and encoder resolution.</td>
</tr>
<tr>
<td>&quot;Q S&quot;</td>
<td>A stall check setting.</td>
</tr>
</tbody>
</table>
Action: (I) This command with no parameter accompanied defines the current position as the home position, where the position counter is 0.

(II) This command with a position parameter changes the current position counter to the specified number.

Format: (I) $ B# R Defines the current position as the home position, where the position counter is 0.

(II) $ B# R pp Changes the current position counter to pp.

pp: Position data
- 0 to 16,777,215: unsigned decimal number up to 8 digits
- -8,388,608 to +8,388,607: signed decimal number up to 7 digits

Note: To change position data memory, perform the command “EP”. When mode=1 and the command “EM”=1, position data memory is signed number(±).

Reply: > (For both I and II)

Example
- PRINT #1, "$1R8000"; CHR$(&HD); Sets the current position counter to 8,000. After execution of the line, the home position (position counter = 0) is located at 8,000 pulses in the CCW direction from the current position.

Note
- Home-search commands such as "θ", "θ A", "θ H", or "θ R" ignore the setting of the command "R" because they mechanically search for the home (ORG) sensor, while the command "R" electrically sets the position counter.

- If a stall detection function is enabled (by the command "QS") and the command "R" is executed, the function may not work properly from then on.
10. Command Reference

<table>
<thead>
<tr>
<th>Command</th>
<th>R D</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0</td>
<td>E1</td>
</tr>
<tr>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

**Action**: This command clears a stall error.

**Format**: $B#R D$

**Reply**: >

**Description**
- The "RD" command clears a stall error and let RC-234 accept and execute subsequent commands including move-commands. Once RC-234 detects a stall, it does not perform any move-related commands except "Θ" (home-search command) from then. In an operation mode 2, RC-234 can resume a normal operation after receiving the command "RD" because it holds an absolute position by using an encoder.

---

<table>
<thead>
<tr>
<th>Command</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0</td>
<td>E1</td>
</tr>
<tr>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

**Action**: This command immediately stops a motor in motion without ramping deceleration.

**Format**: (Ⅰ) $B#S$

(Ⅱ) $B#S MT$

Selects a motor and stops it immediately.

M T... 1, 2

3 (both 1 and 2)

**Reply**: > (For both I and II)

**Note**
- The format (Ⅱ) can't be performed when command “EM”=0 (in alternate 2-axis control mode).
- This command immediately stops a motor without ramping deceleration, which may result in a stall if the motor is running at an extremely high speed. Once a stall occurs, the management of the motor position can not be kept effective. To avoid a motor stall, use the command “SS” instead of “S”. 

---
Action: This command enables the software limit function to stop a motor forcibly when it moves exceeding the designated data memory position range. This setting should be separately designated for each motor.

Format: (I) 

Enables/disables the "software limit" function for each motor.

M2.. Enables the function for motor 2
0: Disables the software limit 【DEFAULT】
1: Stops a motor in motion immediately when the software limit is activated.
2: Stops a motor in motion with ramping deceleration when the software limit is activated.

M1.. Enables the function for motor 1
0: Disables the software limit. 【DEFAULT】
1: Stops a motor in motion immediately when the software limit is activated.
2: Stops a motor in motion with ramping deceleration when the software limit is activated.

(II) 

Sets the data memory position range of the software limit in the CW direction and CCW direction.

CCW.. data memory position rage of the software limit in the CCW direction
-8,388,608 to 16,777,215 【DEFAULT=0】

CW.. data memory position rage of the software limit in the CW direction
-8,388,608 to 16,777,215 【DEFAULT=0】

(III) 

Queries the status of software limit function for each motor.

(IV) 

Queries the data memory position range of the software limit in the CW direction and the CCW direction.

Reply: In case of (I, II)

In case of (III)

In case of (III)
In case of (Ⅳ)  

<table>
<thead>
<tr>
<th>Command</th>
<th>S</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCW</td>
<td>$</td>
<td>B#</td>
</tr>
<tr>
<td>..</td>
<td>CW</td>
<td></td>
</tr>
</tbody>
</table>

The position data of the software in the CCW direction  
Signed decimal number with 8 digits

The position data of the software in the CW direction  
Signed decimal number with 8 digits

Note

- A command error will occur if the CW position data exceeds the CCW data memory position range when format (Ⅱ) is enabled.

- The format (Ⅱ) doesn’t depend on the data set by the command “EP” (Data memory position range).
  Even when the command “EP”=0 (Position memory data range is uni-directional) the position data of the software limit can be set to a negative value.

- The software limit is invalid while a home search command (Ø, Ø Q, etc.) is performed.

- The default of format (Ⅱ) is (0, 0). When the software limit function is turned on by format (Ⅰ) and the CW data position and CCW data position are the same, the motor doesn’t move.

- If “deceleration stop” is specified when the software limit is activated, the deceleration will start from the designated position.

- In continuous interpolation (command “MG”), a motor can’t stop with ramping deceleration even if “deceleration stop” is specified. A motor stops immediately.
Command Reference

**Command**  

<table>
<thead>
<tr>
<th>Command</th>
<th>S S</th>
</tr>
</thead>
</table>

**E0**  

**E1**  

**E2**  

**During motion**  

**EEPROM**  

**Action**  

This command stops a motor in motion with ramping deceleration.

**Format**  

(I)  

\$ B# S S  

Stops a motor selected by the command “F” with ramping deceleration.

(II)  

\$ B# S S MT  

Selects a motor and stops it with ramping deceleration.  
MT... 1, 2  
3 (both 1 and 2)

**Reply**  

> (For both I and II)

**Note**  

- The format (II) can't be performed when the command “EM”=0 (in alternate 2-axis control mode).

- This command allows a motor to decelerate to the low speed determined by OX and OL before stopping it to avoid a motor stall. Use the command “S” if a motor should be immediately stopped in an emergency.


**Command Reference**

**SUM**

**Action**: This command enables/disables a sum check function to detect garbled characters in communication.

**Format**:

(I)  

$ B# S U M 1  

Enables the sum check function.

(II)  

$ B# S U M 0  

Disables the sum check function. [DEFAULT]

(III)  

$ B# S U M  

Queries whether the sum check function is activated or not.

**Reply**: In case of (I, II)  

$  

In case of (III)

When the sum check function is enabled, the following reply will be returned.

$ B# 1 ASCII  

ASCII: Two ASCII characters. (See the Description below for detail.)  
For example, if $ B# is "1", ASCII is "C4".

When the sum check function is disabled, the following reply will be returned.

$ B# 0  


Command: **SUM**  
**Description**  
*Once a sum check function is enabled (by the format (I)), the subsequent commands must be sent in the following format.*

### Sending Format

<table>
<thead>
<tr>
<th>Format</th>
<th>Command Body</th>
<th>ASCII Code</th>
</tr>
</thead>
</table>

- **Command Body**: A normal command string without its last character,  
- **ASCII Code**: The low-order 2 digits of the sum of each character that comprises a Command Body, represented in hexadecimal.

For example, if the sum of a Command Body is 10F (hex), the command should be sent in the following format:  
```
Command Body: [0 F]  
```
As shown in the format above, send "0 F" instead of "F" as ASCII Code. The format is also applied to replies from RC-234.

### Examples of commands for sending

<table>
<thead>
<tr>
<th>Sum Check disabled</th>
<th>Sum Check enabled</th>
<th>Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1endment</td>
<td>$1 5 5</td>
<td></td>
</tr>
</tbody>
</table>
| $16 | $1 6 8 B  | $1 6 = 3 6 + 4 9 + 5 6  
|  |  | = 1 3 9 = 8 B (H) |
| $1 A 0 1 3 0 0  | $1 A 0 1 3 0 0 8 A  | $1 A 0 1 3 0 0 = 3 6 + 4 9 + 6 5  
|  |  | + 4 8 + 4 9 + 5 1 + 4 8 + 4 8  
|  |  | = 3 9 4 = 1 8 A (H) |
Command Reference

<Format of replies from RC-234>

The reply format includes two types: one is for replies that contain a [ ] at the end of their lines; the other is for replies that are not accompanied by [ ]. Once the command "EL1" is issued, however, the format for replies without a [ ] is applied to replies with a [ ].

- The format for replies without a [ ]

```
＞　・ ・ ・ This reply means that a command sent to RC-234 had an incorrect sum check.

Probable causes: A problem on the communication line or an ASCII Code mistakenly set.
```

```
＞　・ ・ ・ This reply means that RC-234 normally* received a command sent to it.

* The reply DOES show that the communication line had no problem but does NOT imply that the received command was valid.
```

- The format for replies with a [ ] (replies for query commands)

```
Reply Body  | ASCII Code  | [ ]
Reply Body : A normal reply string without its last character, [ ].
ASCII Code : The low-order 2 digits of the sum of each character that comprises Reply Body, represented in hexadecimal.
```

<Examples of replies with a check sum function>

<table>
<thead>
<tr>
<th>Reply String</th>
<th>Reply Body</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>＞$ 1 0 C 3[ ]</td>
<td>＞$ 1 0</td>
<td>No error on the communication line.</td>
</tr>
<tr>
<td>＞$ 1 0 C 5[ ]</td>
<td>＞$ 1 0</td>
<td>The wrong ASCII Code implies an error on the communication line. The &quot;＞$ 1 0 &quot; is not reliable.</td>
</tr>
<tr>
<td>＞$ 1 0 8 F B[ ]</td>
<td>＞$ 1 0 8</td>
<td>No error on the communication line.</td>
</tr>
</tbody>
</table>

As the above table shows, your program can detect an error on a communication line by checking ASCII Code in a reply string. Additionally, if a format ( II ) is sent to disable the sum check function, a format for sending commands and receiving replies returns back to the usual format. Send commands in accordance with an appropriate format.

Note

- The functions of command "SUM" and "EE" can not be used simultaneously.
  Disable the command "EE" function before using the command "SUM".

- A sample program of command "SUM" is provided in Chapter 13-2 ② "Sample Program for Special Function", which automatically adds a check sum to commands and verifies the check sum in reply strings.
Command T

<table>
<thead>
<tr>
<th>Command</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0</td>
<td>○</td>
</tr>
<tr>
<td>E1</td>
<td>○</td>
</tr>
<tr>
<td>E2</td>
<td>○</td>
</tr>
<tr>
<td>During</td>
<td>○</td>
</tr>
<tr>
<td>EEPROM</td>
<td>×</td>
</tr>
</tbody>
</table>

Action: This command exposes the timer resource of RC-234. The command can be used to set the timer and to query the timer status.

Format:

(I) $ B# T TIME

Sets timer data and starts the timer to count. When the time is up, RC-234 automatically sends the following string to the host.

$ B# T

TIME: Timer data (0 to 32,767 (Approx. 54 minutes)).

The time data is specified in terms of 100 milliseconds, hence the actual wait time is time multiplied by 100 msec.

(II) $ B# T *

This format has the same function as the format (I), except that it does NOT send the following string when the time is up. $ B# T

To check whether the time is up or not, use the format (III).

(III) $ B# T

Checks whether the time is up or not.

Reply: In case of (I, II) >

In case of (III) > $ B# DT

DT: 0: The time is up.

1: The timer is still counting.

Example

PRINT #1, " $1T*10"; CHR$(&HD);
REPEAT:
PRINT #1, "$1T"; CHR$(&HD);
WAIT:
IF LOC(1) < 5 THEN GOTO WAIT
Q$=MID$(INPUT$(LOC(1), #1), 4, 1)
IF Q$<>"0" THEN GOTO REPEAT

The program, in the 1st line, sets the timer data to 1 second under the condition that it does not expect a "$1T[\text{ack}]" as an automatic reply when the time is up. Then it waits for a reply that consists of 5 characters ("$10[\text{ack}]") in the 2nd through 5th line. In the 6th and 7th line, it checks whether the time is up by testing Q$. If Q$ is not "0", the program jumps to the 2nd line and repeat the procedure again to wait for the timer is up.
Note

- The function of command "T" is just to set the timer in RC-234 and start it. For example, if the command "T50", which sets 5 seconds and starts the timer, is sent and the command "1", which moves a motor to the home position, is subsequently sent, the expected action from RC-234 is NOT that "1" is executed 5 seconds later BUT that the counting of 5-second timer and the execution of the "1" occurs simultaneously. To start the command "1" 5 seconds later, create a program similar to the sample above.

- Other timer-related commands, such as "GWT", "P", "/T", or "/TB", can not be used when the command "T" is in operation. The reason is that those commands share a single timer resource in RC-234.

- Note that the command "T" is not identical to the command "/T" which is used in user programs.
Action: Reads the version of the ROM externally attached to RC-234.

Format: $ B# V

Reply: $ B# RC-234 Ver×.×× by RORZE (××-××-××)
11. User Program Explanation

A simple command sent via RS-232C can perform a sequence of commands by creating a user program. The program is generated using all the valid "stand alone" program commands. Once the program is transferred to EEPROM and autostart is enabled, PC is not needed to run the stored program. To run the stored program, turn the power to the system OFF and then ON again (wait for 2 secs before turning the power ON again). The controller will start the program automatically.

11-1. Command for user program

The table below shows the commands necessary for loading, limiting, and storing user program.

<table>
<thead>
<tr>
<th>Command</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;I &quot;</td>
<td>Transfers the user program to RAM.</td>
</tr>
<tr>
<td>&quot;I R&quot;</td>
<td>Queries the contents of the user program stored in RAM.</td>
</tr>
<tr>
<td>&quot;G&quot;</td>
<td>Executes a user program stored by the command &quot;I &quot;.</td>
</tr>
<tr>
<td>&quot;G LB&quot;</td>
<td>Executes a user program from the label &quot;GLB&quot; in the user program.</td>
</tr>
<tr>
<td>&quot;GAS&quot;</td>
<td>Enable auto-start.</td>
</tr>
<tr>
<td>&quot;GAR&quot;</td>
<td>Disable auto-start.</td>
</tr>
<tr>
<td>&quot;GS&quot;</td>
<td>Pauses a user program.</td>
</tr>
<tr>
<td>&quot;GC&quot;</td>
<td>Resumes the user program after temporary stoppage.</td>
</tr>
<tr>
<td>&quot;GE&quot;</td>
<td>Ends the user program.</td>
</tr>
<tr>
<td>&quot;GES&quot;</td>
<td>Ends the user program immediately.</td>
</tr>
<tr>
<td>&quot;GW BF&quot;</td>
<td>Rewrites the buffer data used in the user program.</td>
</tr>
<tr>
<td>&quot;GWW BF&quot;</td>
<td>Changes timer data during motion.</td>
</tr>
<tr>
<td>&quot;GRT&quot;</td>
<td>Queries the current status in the data buffer used in the user program.</td>
</tr>
<tr>
<td>&quot;GRT NO&quot;</td>
<td>Queries the current time left.</td>
</tr>
<tr>
<td>&quot;GN&quot;</td>
<td>Queries the next stop to be executed in a user program.</td>
</tr>
<tr>
<td>&quot;GSSE&quot;</td>
<td>Queries the cause of stop in a user program</td>
</tr>
<tr>
<td>&quot;GSS&quot;</td>
<td>Queries the status of user program.</td>
</tr>
<tr>
<td>&quot;I W&quot;</td>
<td>Stores the user program in RAM into EEPROM</td>
</tr>
<tr>
<td>&quot;I L&quot;</td>
<td>Reads out the user program stored in EEPROM to RAM</td>
</tr>
</tbody>
</table>

For more details, see Chapter 10 "Command Reference".

To load user program using BASIC, see Chapter 13 "13-2 ②Sample Program for Special Function" Example 3.

*Stand alone
Action to control motor etc. by the program stored in RC-234 without PC.
### 11-2 Relation between EEPROM and RAM

<table>
<thead>
<tr>
<th>Personal Computer</th>
<th>User Program</th>
<th>Position Data</th>
<th>Speed Data</th>
<th>Baud rate Setting</th>
<th>Autostart Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;I&quot; &quot;I R&quot;</td>
<td>&quot;A&quot; &quot;M A&quot;</td>
<td>&quot;O H&quot; &quot;O L&quot;</td>
<td>&quot;E S&quot;</td>
<td>&quot;G A S&quot; &quot;G A R&quot;</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RC-234</th>
<th>RAM</th>
<th>User Program</th>
<th>Position Data</th>
<th>Speed Data</th>
<th>Baud rate Setting</th>
<th>Autostart Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;I W&quot; &quot;I L&quot;</td>
<td>&quot;A W&quot; &quot;A L&quot;</td>
<td>&quot;D W&quot; &quot;D L&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EEPROM</th>
<th>User Program</th>
<th>Position Data</th>
<th>Speed Data</th>
<th>Baud rate Setting</th>
<th>Autostart Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;I&quot; &quot;I R&quot;</td>
<td>&quot;A&quot; &quot;M A&quot;</td>
<td>&quot;O H&quot; &quot;O L&quot;</td>
<td>&quot;E S&quot;</td>
<td>&quot;G A S&quot; &quot;G A R&quot;</td>
<td></td>
</tr>
</tbody>
</table>

"I" "I R" etc.
11-3 Creating a User Program (Description of the Format)

<What is User Program for?>
A control program saved in an EEPROM of RC-234 provides the following functions:
1) RC-234 can be controlled with a sequencer (PLC).
2) A simple command sent via RS-232C can perform a sequence of commands. (A macro function).
The program, which you create and store in the EEPROM to use the functionality, is called User Program.

<The format of User Program>

Format: /
command [parameter(s)]

- [/] symbol
  Each command has to be preceded by "/" and also used as a separator between commands.

- Commands used in User Programs are classified in two types:
  1) Commands exclusive to User Program
  2) Commands applicable to both in communication and in User Program

For more information on the type 1, see Chapter 12 "User Program Command Reference". RC-234 returns no reply to commands exclusive to User Program, except those whose function is to send some data to a PC.

In regard to the type 2, see Chapter 15 "Control Command List", where commands having "O" mark in their EP boxes are available in User Programs. These commands are applicable to User Programs with the format of "command". The type 2 commands in a User Program transmit no reply to a PC, though their original reply in a communication mode is a character ">".
Query commands of the type 2 return a string in which a character ">" is removed from their original reply form.
1. Creating a User Program on a RAM in RC-234

Format: $B#I

Reply: >

Format: /
Command
/
Command
......
/
E
N
D

Reply: >

Send the command "I". The RC-234 will respond with ">". Then a User Program can be transmitted. After transferring the contents of the program, a "I" must be sent as a terminator as shown above.

NOTE:"/END"
The string "/END" is needed at the end of a program. The string can be also placed wherever the program should be terminated by, for example, conditional jumps in the program.

Example) "/0//JON1+1/END/3//END"

2. Executing a User Program with a PC

Format: $B#G

The command starts the User Program from the first line in the program.
"/GLB" commands in a program allow you to start the program from a specified line.

Format: $B#G LB

The command starts a program from the next step of the label command "/GLB"

Available characters for LB: Figures (0 - 9) or symbol characters except alphabets, $, #, and /.

Note: a to z can be used in the user program as a label but can't be performed in the command "G".

A label command "/GLB" serves as a jump-label or the name of a subroutine program. The "/GLB" command does not take any action by itself.
3. Saving a User Program in EEPROM of RC-234

A User Program, which is explained in the previous sections, is initially saved in a RAM area. If a RC-234 is turned off, the program will disappear. The program must be transferred to an EEPROM area to be retained.

**Format:**

\[
\text{\$ B# I W }\]

The command copies the User Program from the RAM to the EEPROM. The RC-234 responds with the following string:

\[
> \text{\$ B# * }\]

When the controller receives the command " I W", it sends back a " > " as a reply to the command. The rest of the string, "\$ B# * " is transferred after writing of the data into the EEPROM is completed.

Now, the program is protected if the power is turned off. When the RC-234 is powered again, the program stored in the EEPROM will automatically load to the RAM.

4. Auto-start function

**Format:**

\[
\text{\$ B# G A S }\]

The command enables the auto-start function. At the same time, the setting of the auto-start is stored in the EEPROM. With the auto-start set, the RC-234 automatically runs the User Program when it is turned on. In this way, RC-234 runs in a standalone mode, without a PC (RS-232C) or a Link-Master (RC-002).

**Format:**

\[
\text{\$ B# G A R }\]

The command disables the auto-start function.
5. Supplemental information on User Program & EEPROM

Relation between RAM and EEPROM>
The actual location of a User Program, which is stored by the command "I" and started by the command "G", is in a RAM rather than in an EEPROM.
To view a User program in a RAM, use the command "IR".
To copy a User Program in a RAM to an EEPROM, use the command "IW".
Turning off a RC-234 before execution of command "IW" results in loosing the program in the RAM, which has been sent by the command "I".
The command "IL" loads a RAM with a program stored in an EEPROM.
When the command is executed, a previous program in the RAM is overwritten. That should be noted.

Power-on behavior of User Program>
When a RC-234 is turned on, a program, which has been saved in the EEPROM by the command "IW", is automatically copied to the RAM.

Erasing all the data in EEPROM>
To erase the entire data in an EEPROM, which includes position data, and various settings except a User Program, use the command "EE///". The command deletes the User Program and set position data and other settings at the default values.

Limitations in communication commands while a User Program is running>
The following communication commands are invalid while a User Program is running.
Inapplicable commands: "G", "I", "IL", "IW", "AW", "DW", "AL", and "DL".
To use the commands, terminate the User Program first.
Use the command "GES" or "GE" to terminate the User Program.
To query the running status of a User Program, use the command "GSS".
To query the cause of stop in a User Program, use the command "GSSE"

Auto-start function>
If a RC-234 is turned off before the command "IW" is executed to save a User Program on the RAM into the EEPROM, the User Program is cleared. Make sure that the command "IW" is sent, otherwise the command "GAS" is meaningless.
Changes in a User Program of an EEPROM do not affect the setting of the auto-start function. To disable the auto-start function, use the command "GAR".

Command errors in User Program>
Any command error during execution of a User Program terminates the program.
However, if the cause of the error is not a command in the User Program but a command that the controller has received from a PC, the User Program does not stop and continues to run.
Termination of User Program

The command "/END" terminates a User Program. If you want to externally terminate the program, send either the command "GES" or "GE" to the RC-234 via communication line.

The command "GE" does terminate a User program but does not stop a motor immediately. If the motor is running, it will not stop until it reaches a designated position, or it will continue to run when it has been activated by the command "7" or "8". To stop the motor, use the command "S".

To pause a User Program, use the command "GS". To resume the User program, send the command "GC".

Command "GSSE"

This command can query the running status (the cause of stop) of a User Program.

Example: Assuming that a command error occurred in a User Program:
Send a command "/ $ 1 G S S E [ ]", the following reply will be returned.

This indicates that a User Program has been terminated by an incorrect command.

Debugging a User Program

Additional commands useful for debugging are as follows:

Command "G N"
Queries the next step to be executed in a User Program.
The command is useful to spot the program line where execution has terminated in an error state.

Command "/ C O"
Sends a specified string to a host. The command "/ C O O K 0 0 1 ", for example, transfers the string "O K 0 0 1 " to a host. With the message from the controller, identification of a step that is being executed can be easily obtained.
<Setting the values of configuring commands>

There are two ways to store in an EEPROM the values of configuring commands such as speed-setting commands:

① Add setting values such as speed data to a User Program and send the command "I W" to save them in the EEPROM.

In the example below, high-speed data is stored in the EEPROM.

```
$ B# I
/ O H 2 0 0 0 / ... user commands ... / E N D
$ B# I W
```

② Use the command "D W" to copy setting values such as speed data from the RAM to the EEPROM.
Note that the command "D L" must be executed before a User Program is run.
However, when RC-234 is powered, the command "D L" is automatically executed by itself.
So there is no need to perform the command explicitly as long as the data in the EEPROM is not changed.

In the example below, high-speed data and a User Program are stored in the EEPROM.

```
$ B# O H 2 0 0 0
$ B# I
/ ... user commands ... / E N D
$ B# I W
$ B# D W
```

The second procedure is recommended to save the memory area of the EEPROM.

※ In a User Program, the speed parameter (OH, OL, OX, OS, and OC) and the position data (A, MA), etc. can be set using 4-byte data buffer W (BF).
Besides, the position number can be set using a loop counter for the position number.

Ex.)
/ LDWA, 1000 Sets the value of WA to 1000
/OHWA Sets the value of OH to WA
/LDWAF, 2000 Sets the value of WF to 2000
/A100WF Sets the data of the position number 100 to WF
/CSX1,100 Sets 100 to the counter X1
/BCX1 Moves to the position number CX1(=100)
11-4 Example of User Program

1) \( /D \ 0 \ 0 \ /0 \ /_/ \ D \ 0 \ 1 \ B \ / \ J \ O \ N \ 0 + 1 \ / J - 2 \ /D \ 0 \ 0 \ / 2 \ 5 \ 0 \ 0 \ 0 \)

\( /3 \ /_/ \ E N D \)

① Turns all of the output port bits OFF (0).
② Performs a mechanical home search (command "0").
③ Proceeds to the next step after the home search is completed or a motor stops.
④ Turns the output port bit0 ON (1). In this case, the activated bit0 indicates that the home search is finished.
⑤ Jumps forward two steps (i.e. the command "/D 0 0") if the input port bit0 is ON (1).
⑥ Jumps backward one step (i.e. the command "/JON +1").
⑦ Performs the same function as in ①.
⑧ Sets a pulse position at 5,000 pulses.
⑨ The motor moves to the position specified in the previous procedure.
⑩ Proceeds to the next step after the motor stops or the absolute move is completed.
⑪ Exits the User Program.

Turns all bits of output port OFF

Mechanical home search

Waits for completion of move

Turns output port bit0 ON

Is input port bit0 ON?

YES

Turns all bits of output port OFF

Sets 5,000 pulses

Moves to the point

Waits for completion of move

END

2) \( /G \ 1 \ / 2 \ 5 \ 0 \ 0 \ 0 \ /3 \ /_/ \ E N D \ /G \ 2 \ / 2 \ 6 \ 0 \ 0 \ 0 \ /3 \ /_/ \ E N D \)

\( /G \ 3 \ / 2 \ 7 \ 0 \ 0 \ 0 \ /3 \ /_/ \ E N D \)

This User Program contains three different programs as shown by the underlines. They are independent from one another. These programs move a motor to the position of 5,000, 6,000, and 7,000 pulses, respectively.

Sending the command "G" or "G 1 " starts the program from the "/G 1 " to the first "/E N D".
Similarly, the program from the "/G 2 " to the second "/E N D" is activated by sending the command "G 2 ", and from the "/G 3 " to the third "/E N D" by "G 3 ".

A "macro" registered in RC-234 allows a simple command to perform multiple instructions, as shown in the above example.
3) \(/EC\,2\,0\,0\,0\,3/\,?/T\,1\,0/\,4/\,?/END\)

This program drives a motor to the position of 2,000 pulses, waits for the completion of the move, idles for one second, and then moves the motor in the CW direction by 2,000 pulses.

The "/EC" clears errors.

The "/?" checks for a limit-sensor error, an input signal of emergency-stop, and a stall error. If any of them has occurred, the motor is halted immediately and the User Program is terminated.

What if "/?" is not added to the User Program:
Though the motor stops upon occurrence of any of the errors, the succeeding steps in the User Program continues to run.

For example, without the "/?" in the User Program, a move command that is executed after the emergency-stop signal halts the rotation would start the motor to run again if the signal is off at the time of the command execution.
(As long as the emergency-stop signal remains ON, move commands are skipped but the other commands are executed).

The following figures show the action when the emergency-stop signal is turned ON during a move to the 2,000-pulse position in the above program, assuming that the signal keeps ON for 0.5 second and then turns OFF.

<Program with a "/ ?">

/2\,0\,0\,0\,/3/\,?/T\,1\,0/\,4/\,?/END

As the emergency-stop signal is turned ON after the command "/3" attempts to move the motor to the 2,000-pulse position, the User Program is terminated at the "/?".

<Program without a "/ ?">

/2\,0\,0\,0\,/3/T\,1\,0/\,4/END

As the emergency-stop signal is turned ON after the command "/3" attempts to move the motor to the 2,000-pulse position, the motor is stopped, but the next command, "/T\,1\,0" (idle for one second), will be executed. After the signal is turned OFF as in the assumption and the waiting time is over, the command "/4" that moves the motor in the CW direction by 2,000 pulses is executed.
12. User Program Command Reference

The User Program can be applied to all the operation modes (0 to 2).

Command / (NULL)  | Wait for the motor to finish the move

Format  

Use this command to confirm the status of motor and if motor is during motion, wait for the motor to finish the move. If the motion ends, go to the following command.

Command / ADD  | Addition of data in the data buffer

Format 1  

Adds the data in the data buffer (BF) and the data (DU, DL) and writes the result to the data buffer (BF).

BF... Data buffer (A to F)  
Each data buffer has 1-byte data.  
DU, DL... One-byte data for addition.  
DU represents the upper four bits of the data, and DL, the lower. Each constant must range from "0"(zero) to "F" in hexadecimal.

Format 2  

Adds the data in the data buffer (BF) and the data in the data buffer (B2) and writes the result to the data buffer (BF).

BF... Data buffer (A to F)  
B2... Data buffer (A to F)  
Each data buffer has 1-byte data.

Format 3  

Adds the data of the 4-byte buffer W(BF) and the data (DT) and writes the result to the 4-byte data buffer W(BF).

W(BF)... 4-byte data buffer (WA to WF)  
The data of each data buffer is −999,999,999 to +999,999,999.  
DT... Data for addition (-999,999,999 to +999,999,999)  
Signed number up to 9 digits
Command \( /\ A\ D\ D \)

Format 4

\[
/ A D D W BF . W B2
\]

Adds the data in the 4-byte data buffer \(W(BF)\) and the data in the 4-byte data buffer \(W(B2)\) and writes the result to 4-byte data buffer \(W(BF)\).

\[
W(BF)\ldots 4\text{-}byte\ data\ buffer\ (WA\ to\ WF)\nW(B2)\ldots 4\text{-}byte\ data\ buffer\ (WA\ to\ WF)\n\]

Each data in the 4-byte data buffer is \(-999,999,999\) to \(+999,999,999\).

[Note]

- In the case of overflow, a command error will occur.

---

Command \( /\ A\ N\ D \)

Data buffer AND operation

Format 1

\[
/ A N D BF . DU DL
\]

Implements Logical AND in the data buffer \((BF)\), \(DU\), and \(DL\) and store the result to the specified data buffer \((BF)\).

\[
BF\ldots Data\ buffer\ (A\ to\ F)\nEach\ data\ buffer\ has\ one\ byte\ data.\nDU, DL\ldots 1\text{-}byte\ data\ to\ implement\ AND\nThey\ represent\ upper\ 4\ bit(DU)\ and\ lower\ 4\ bit\ (DL)\ and\nset\ by\ HEX\ data(0\ to\ F)\n\]

Format 2

\[
/ A N D BF . B2
\]

Implements Logical AND in the data buffer \((BF)\) and data buffer \((B2)\) and store the result to data buffer \((BF)\).

\[
BF\ldots DATA\ buffer\ (A\ to\ F)\nB2\ldots DATA\ buffer\ (A\ to\ F)\nEach\ data\ buffer\ has\ 1\text{-}byte\ data.\n\]
**Command /B S**

**Data buffer**  Set or reset a bit

**Format**
/ B S BF BT DT

Sets or resets the data buffer (BF) on a bit basis.

- **BF**... DATA buffer (A to F)
  - Each data buffer has 1-byte data.
- **BT**... Bit number (0 to 7)
- **DT**... 0: Resets the specified bit of the buffer to 0 (zero).
  - 1: Sets the specified bit of the buffer to 1 (one).

**Command /C C**

**Counter**  Transmission of value to PC

**Format 1**
/ C C CT

Transmits the current value of a loop counter to a PC(RS-232C).

- **CT**... Counter (1 or 2)

**Transfer Format 1**
$ B# CT OOOOO [dec]

Counter value
(5-digit decimal number)

**Format 2**
/ C C X CT

Transmits the current value of a loop counter for the position number to a PC(RS-232C)

- **X(CT)**... Counter (X1 to X5)

**Transfer Format 2**
$ B# X CT OOOOO [dec]

Counter value
(5-digit decimal number)
Command /CD

Format 1 / C D CT

Decrements the loop counter 1 or 2 (CT) by 1. (The command does not change a counter with its value of 0.)

CT... Counter (1 or 2)

Format 2 / C D X CT

Decrements the value of a loop counter for the position number X1 to X5 (X(CT)) by 1. (The command does not change a counter with its value of 0.)

X(CT)... Counter (X1 to X5)

Command /CO

Transmission of character string

Format / C O DT

Transmits a character string (DT) to a PC via RS-232C.

DT... Arbitrary character sequence with a length of 20 or less.
### Command /C S

<table>
<thead>
<tr>
<th>Command</th>
<th>Format 1</th>
<th>Format 2</th>
<th>Format 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>/C S</td>
<td>/ C S CT . CN</td>
<td>/ C S X CT . CN</td>
<td>/ C S X CT . C X C2</td>
</tr>
</tbody>
</table>

Sets the value of the loop counter 1 or 2 (CT).

CT… Counter (1 or 2)
CN… Counter value, which must be less than 6 digit with its range of 0 to 65,535.

Sets the value of a loop counter for the position number X1 to X5 (X(CT)).

X(CT)... Counter (X1 to X5)
CN… Counter value, which must be less than 4 digit with its range of 0 to 999

Copies the value of a loop counter for the position number X(C2) to the loop counter for the position number X(CT).

X(CT)... Counter (X1 to X5)
X(C2)... Counter (X1 to X5)

**Ex.**

- `/ C S X 1, 1` Sets 1 to the counter X1.
- `/ A C X 1, 1 0 0 0 0` Sets 10,000 to the position number CX1.
- `/ B C X 1` Moves to the position of the position number CX1 (=001)

### Command /C U

<table>
<thead>
<tr>
<th>Command</th>
<th>Format 1</th>
<th>Format 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>/C U</td>
<td>/ C U CT</td>
<td>/ C U X CT</td>
</tr>
</tbody>
</table>

Increments the loop counter 1 or 2 (CT) by 1.
(The command does not change a counter with its value of 65,535.)

CT… Counter (1 or 2)

Increments the value of the loop counter for the position number CT (1 to 5) by 1.
(The command does not change a counter with its value of 999.)

X(CT)... Counter (X1 to X5)
Command \text{/D I V}  Division of data in the data buffer

Format 1  \[ / \] D I V \[ \] BF . DU DL

Divides the data in the data buffer (BF) by the data (DU, DL) and writes the result to the data buffer (BF).

BF... Data buffer (A to F)
Each data buffer has 1-byte data.
DU, DL... One-byte data for division.
DU represents the upper four bits of the data, and DL, the lower. Each constant must range from "0"(zero) to "F" in hexadecimal.

Format 2  \[ / \] D I V \[ \] BF . B2

Divides the data in the data buffer (FB) by the data in the data buffer (B2) and writes the result to the data buffer (BF).

BF... Data buffer (A to F)
B2... Data buffer (A to F)
Each data buffer has 1-byte data.

Format 3  \[ / \] D I V \[ \] W BF . DT

Divides the data in the 4-byte data buffer W(BF) by the data (DT) and writes the result to the 4-byte data buffer W(BF).

W(BF)... 4-byte data buffer (WA to WF)
The data for each data buffer is \(-999,999,999\) to \(+999,999,999\).
DT... Data for division \((-999,999,999\) to \(+999,999,999\))
Signed number with 9 digits

Format 4  \[ / \] D I V \[ \] W BF . W B2

Divides the data in the 4-byte data buffer W(BF) by the data in the 4-byte data buffer W(B2) and writes the result to the 4-byte data buffer W(BF).

W(BF)... 4-byte data buffer (WA to WF)
W(B2)... 4-byte data buffer (WA to WF)
The data of each data buffer is \(-999,999,999\) to \(+999,999,999\).

[Note]
- Division by 0 is not possible. It causes a command error. The decimal places are rounded down to the nearest whole number.
Command /EC

Format 1

Sets bit0, 1, 2, 6 of the condition flag (see the command 9) to 0.

Format 2

Select one of bit 0, 1, 2, 6 of condition flag (see the command 9) and set the selected bit to 0.

BT... bit (0,1,2,6)

<Condition Flag>

B7 B6 B5 B4 B3 B2 B1 B0

Ignored Stall Error Limit Error Communication Error Emergency stop

[Note]

- The status indication LED ERR will light when either the limit error, the emergency stop or the command error occurs and turn off the light when the command “ ” (NULL), “9ALL” (error status query), a home search command, such as "0", "0A", "0H", "0Q", "0R", "0Z" or the user command "/EC" is performed.

Command /END

Termination of a program

Format

Terminates the user program.

Note:

- The command may be placed anywhere according to need and there is no limit to the number of times of use. However, User Programs must end with the command.
### Command \(/G\)

**Format** \(/ \ G \ LB\)

The command is used as a label in User Programs. The labels are referred by jump commands such as "/J" or "/SC".

- **LB**... Label symbol in a User Program:
  - Alphabet (a to z), figure (0 to 9), and other symbols except A to Z, $, #, and /.

**Note:** RC-234 can use alphabetic characters (a to z), but has limitations. A label that has a lower case letter, for example "/JGa1", can be used only in User Programs. Lower case labels cannot be applied to a communication command such as "/Gz" to start a User Program from the specified label (i.e., "/Gz").

### Command \(/I1\)

**Format** \(/ \ I \ 1 \ BF\)

Reads the input port status and stores it in the data buffer (BF). The command stores 1-byte hexadecimal data in the data buffer.

- **BF**... Data buffer (A to F)

### Command \(/IL\)

**Format** \(/ \ I \ L \ BF\)

A sensor status, which is the same data as obtained by a communication command "C L", is stored in the data buffer (BF). The command stores 1-byte hexadecimal data in the data buffer.

- **BF**... Data buffer (A to F)
Command / J

Unconditional jump

Format  /  J  JUMP

The program jumps to the address specified by JUMP.

JUMP... See the table below.

<Description of JUMP>

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
</table>
| ++[DT]  | The program jumps forward DT+1 steps.  
 DT ... Step number (0 to 999). |
| −[DT]   | The program jumps backward DT-1 steps.  
 DT ... Step number (0 to 999). |
| A[DT]   | The program jumps to the DTth command step from the top of the program.  
 A[DT] value of the first command in the program is 1 (one).  
 DT ... Absolute address (1 to 999). |
| G[LB] [DT] | The program jumps to the DTth command step from the label command "/G LB".  
 The DT value to specify the "/G LB" itself is 0 (zero).  
 If no label of "/G LB" is found, a command error occurs.  
 LB ... Label symbol: alphabet (a to z), figure (0 to 9), and other symbols  
 except A to Z, $, #, and /.  
 DT ... Step number (1 to 999). |

Examples:

/ J + 1  The next command is skipped. The next command but one will be performed in the following execution step.

/ J − 2  The command in the previous step will be executed in the following execution step.

/ J A 1  The program jumps to the first command in the User Program.

/ J G 0 1  The subsequent command to the label "/ J G 0 1 " will be performed in the following execution step.
Command /JB

Format 1

The program jumps to the address (JUMP) if the bit (BT) in the data buffer (BF) is 1.

BF... Data buffer (A to F)
    Each data buffer has 1-byte data.
BT... Bit number (0 to 7)
JUMP... See the table below.

Format 2

The program jumps to the address (JUMP) if the bit (BT) in the data buffer (BF) is 0.

BF... Data buffer (A to F)
    Each data buffer has 1-byte data.
BT... Bit number (0 to 7)
JUMP... See the table below.

<Description of JUMP>

<table>
<thead>
<tr>
<th>JUMP</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+[DT]</td>
<td>The program jumps forward DT+1 steps.</td>
</tr>
<tr>
<td></td>
<td>DT ... Step number (0 to 999).</td>
</tr>
<tr>
<td>—[DT]</td>
<td>The program jumps backward DT-1 steps.</td>
</tr>
<tr>
<td></td>
<td>DT ... Step number (0 to 999).</td>
</tr>
<tr>
<td>A[DT]</td>
<td>The program jumps to the DTth command step from the top of the program.</td>
</tr>
<tr>
<td></td>
<td>A[DT] value of the first command in the program is 1 (one).</td>
</tr>
<tr>
<td></td>
<td>DT ... Absolute address (1 to 999).</td>
</tr>
<tr>
<td>G[LB] [DT]</td>
<td>The program jumps to the DTth command step from the label command “/GLB”.</td>
</tr>
<tr>
<td></td>
<td>The DT value to specify the “/GLB” itself is 0 (zero).</td>
</tr>
<tr>
<td></td>
<td>If no label of “/GLB” is found, a command error occurs.</td>
</tr>
<tr>
<td></td>
<td>LB ... Label symbol: alphabet (a to z), figure (0 to 9), and other symbols</td>
</tr>
<tr>
<td></td>
<td>except A to Z, $, #, and /.</td>
</tr>
<tr>
<td></td>
<td>DT ... Step number (1 to 999).</td>
</tr>
</tbody>
</table>

Examples:

/ J B O N B 2 G 0 1
If the bit2 in the data buffer B is 1, the subsequent command to the label “/G 0” will be performed in the following execution step.
Otherwise the subsequent command to the “/ J B O N B 2 G 0 1” will be performed.

/ J B O F A 7 – 3
If the bit7 in the data buffer A is 0, the command 2 steps previous to the “/ J B O F A 7 – 3” will be performed in the following execution step.
Otherwise the subsequent command to the “/ J B O F A 7 – 3” will be performed.
Command / J C

Format 1

The program jumps to the address (JUMP) if the value of the loop counter (CT) is 0.

CT... Counter number (1 or 2)
JUMP... See the table in the next section.

Format 2

The program jumps to the address (JUMP) if the logic derived from the value of the loop counter (CT), the comparison symbol (SN), and the comparison value (CN) are true.

CT... Counter (1 or 2)
SN... Comparison condition (<, =, or >)
CN... Value to be compared with the counter.
   Decimal number (0 to 65,535)
JUMP... See the table in the next section.

Format 3

The program jumps to the address (JUMP) if the counter for the position number X(CT) is 0.

X(CT)... Counter (X1 to X5)
JUMP... See the table in the next section.

Format 4

The program jumps to the address (JUMP) if the logic derived from the value of the loop counter for the position number X(CT), the comparison symbol (SN), and the comparison value (CN) are true.

X(CT)... Counter (X1 to X5)
SN... Comparison condition (<, =, or >)
CN... Value to be compared with the counter.
   Decimal number (0 to 999)
JUMP... See the table in the next section
Format 5

The program jumps to the address (JUMP) if the logic derived from the value of the loop counter for the position number X(CT), the comparison symbol (SN), and the value of the loop counter for the position number X(C2) are true.

X(CT)... Counter (X1 to X5)
SN... Comparison condition (<, =, or >)
X(C2)... Counter (X1 to X5)
JUMP... See the table below.

Format 6

The program jumps to the address (JUMP) if the logic derived from the value of the loop counter for the position number X(CT), the comparison symbol (SN), and the 4-byte data buffer W(BF) are true.

X(CT)... Counter (X1 to X5)
SN... Comparison condition (<, =, or >)
W(BF)... 4-byte data buffer (WA to WF)
JUMP... See the table below.

<Description of JUMP>

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+[DT]</td>
<td>The program jumps forward DT+1 steps. DT ... Step number (0 to 999).</td>
</tr>
<tr>
<td>−[DT]</td>
<td>The program jumps backward DT-1 steps. DT ... Step number (0 to 999).</td>
</tr>
<tr>
<td>A[DT]</td>
<td>The program jumps to the DTth command step from the top of the program. A[DT] value of the first command in the program is 1 (one). DT ... Absolute address (1 to 999).</td>
</tr>
<tr>
<td>G[LB][DT]</td>
<td>The program jumps to the DTth command step from the label command “/G LB”. The DT value to specify the “/G LB” itself is 0 (zero). If no label of “/G LB” is found, a command error occurs. LB ... Label symbol: alphabet (a to z), figure (0 to 9), and other symbols except A to Z, $, #, and /. DT ... Step number (1 to 999).</td>
</tr>
</tbody>
</table>

Examples:

/ JC 1 A 5 0
If the value of the counter1 is 0, the 50th command from the top of the program will be performed in the following execution step.
Otherwise the subsequent command to the “/ JC 1 A 5 0” will be performed.

/ JC 1 > 3 0 0 G 2 1
If the value of the counter1 is greater than 3000, the subsequent command to the label “/ G 2” will be performed in the following execution step.
Otherwise the subsequent command to the “/ JC 1 > 3 0 0 G 2 1” will be performed.
Command /JE

Format 1

/J J E JUMP

The program jumps to the address (JUMP) if any error flag has been set in a condition flag. The command checks the bit 0, 1, 2, and 6 in the condition flag.

JUMP... See the table in the previous section.

Format 2

/J J E BT JUMP

The program jumps to the address (JUMP) if the specified bit (BT) in the condition flag is 1.

BT... Bit in the condition flag (0, 1, 2, or 6).
JUMP... See the table in the previous section.

<Condition Flag>

B7 B6 B5 B4 B3 B2 B1 B0

Ignored Stall Error Ignored

Ignored Ignored

Communication Error Limit Error Emergency stop Command Error

Note

- Commands in a wrong format cannot be detected with the "/JE" command.
  If any of commands in a User Program contains a format error, the program is terminated at the time the command is fetched.

- In simultaneous 2-axis control, the command "/JE" jumps to the address (JUMP) by a logical OR of the motor 1 and the motor 2. To jump to the address (JUMP) by condition data for each motor, use the command "/JEM".
Command / J E M

**Format 1**

```
/ J E M MT JUMP
```

The program jumps to the address (JUMP) if any error flag has been set in a cause of stop flag.

- **MT**... Target motor (1 or 2)
- **JUMP**... See the table below.

**Format 2**

```
/ J E M MT BT JUMP
```

The program jumps to the address (JUMP) if the specified bit (BT) in the cause of stop flag is 1.

- **MT**... Target motor (1 or 2)
- **BT**... Bit in the condition flag (0 to 5).
- **JUMP**... See the table below.

### <Cause of stop Flag>

<table>
<thead>
<tr>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ignored</td>
<td>Stall Error</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0(fixed)</td>
<td>CW Limit Error</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Synchronous error (*1)</td>
<td>CCW Limit Error</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop command</td>
<td>Emergency stop</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(*1: Errors caused by one axis stopping during synchronized motion)

### <Description of JUMP>

<table>
<thead>
<tr>
<th>+[DT]</th>
<th>The program jumps forward DT+1 steps. DT ... Step number (0 to 999).</th>
</tr>
</thead>
<tbody>
<tr>
<td>—[DT]</td>
<td>The program jumps backward DT-1 steps. DT ... Step number (0 to 999).</td>
</tr>
<tr>
<td>A[DT]</td>
<td>The program jumps to the DTth command step from the top of the program. DT ... Absolute address (1 to 999). A[DT] value of the first command in the program is 1 (one).</td>
</tr>
<tr>
<td>G[LB] [DT]</td>
<td>The program jumps to the DTth command step from the label command &quot;/GLB&quot;. The DT value to specify the &quot;/GLB&quot; itself is 0 (zero). If no label of &quot;/GLB&quot; is found, a command error occurs. LB ... Label symbol: alphabet (a to z), figure (0 to 9), and other symbols except A to Z, $, #, and /. DT ... Step number (1 to 999).</td>
</tr>
</tbody>
</table>

**Note**

- The cause of stop bits 0 to 5 are cleared when a move command is performed.
The program jumps to the address (JUMP) if either a motor 1 or a motor 2 is running.

JUMP... See the table in the previous section.

The program jumps to the address (JUMP) if the specified motor is running.

MT... Target motor (1 or 2)
JUMP... See the table in the previous section.

Example: / J M - 2
If a motor is running, the command in the previous step will be executed in the following execution step. Otherwise, the next command will be executed.
12. User Program Command Reference

Command /JO

Format 1

/ J O N BT JUMP

The program jumps to the address (JUMP) if the bit (BT) of the general-purpose input port is ON. Additionally, the status of “INP1”, “INP2”, or sensors such as ORG, CW limit, or CCW limit can be chosen as a criterion for the command. In these cases, if a specified sensor is ON, the program jumps to JUMP.

BT... Bit (0 to 9)
0 to 7: General-purpose input port.
8, 9: INP1 and INP2, respectively

Sensor input (A to C)
A: Home (ORG) sensor.
B: CW limit sensor.
C: CCW limit sensor.

JUMP... See the table below.

Format 2

/ J O F BT JUMP

The program jumps to the address (JUMP) if the bit (BT) of the general-purpose input port is OFF. Additionally, the status of “INP1”, “INP2”, or sensors such as ORG, CW limit, or CCW limit can be chosen as a criterion for the command. In these cases, if a specified sensor is OFF, the program jumps to JUMP.

BT... See the description in the Format 1.
JUMP... See the table below.

<Description of JUMP>

<table>
<thead>
<tr>
<th>Description</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>+[DT]</td>
<td>The program jumps forward DT+1 steps. DT ... Step number (0 to 999).</td>
</tr>
<tr>
<td>―[DT]</td>
<td>The program jumps backward DT-1 steps. DT ... Step number (0 to 999).</td>
</tr>
<tr>
<td>A[DT]</td>
<td>The program jumps to the DTth command step from the top of the program. A[DT] value of the first command in the program is 1 (one). DT ... Absolute address (1 to 999).</td>
</tr>
<tr>
<td>G[LB][DT]</td>
<td>The program jumps to the DTth command step from the label command “/GLB”. The DT value to specify the “/GLB” itself is 0 (zero). If no label of “/GLB” is found, a command error occurs. LB ... Label symbol: alphabet (a to z), figure (0 to 9), and other symbols except A to Z, $, #, and /. DT ... Step number (1 to 999).</td>
</tr>
</tbody>
</table>
Command /JO

Examples:

/JOIN2G11
If the bit2 of the input port is ON, the program jumps to the subsequent command to the
label "/G 1".
If the bit2 is OFF, the next command will be performed in the following execution step.

/JOF1-1
If the bit1 of the input port is OFF, the command is repeated until the bit1 turns ON.

/JON8+3
If the INP1 is ON, the next three steps will be skipped and the fourth command will be
performed in the following execution step.

/JONCG21
If the CCW limit sensor is ON, the program jumps to the subsequent command to the label
"/G 2".

Note:
When conditional jumps based on the status of a sensor such as ORG, CW limit, or CCW limit
are used, the following should be considered:

・To use commands that performs conditional jumps based on the limit sensors does not affect
the inherent function of the sensors.
Accordingly if the limit sensor turns ON while a motor is running, the motor immediately
stops.

・The commands that perform conditional jumps based on a sensor status can not be used
when any of the home-search-related commands (i.e., "@O", "@Z", "@R", "@H", and "@A")
is being executed.
Avoid using the command in such cases.

・When the command "/JO", based on the status of sensors, is executed, it checks the status
of the sensor attached to a motor that has been specified by the command "F".

・The status of a sensor is checked when the "/JO" command is executed.
This means that a User Program may act against a programmer's intention. For example, a
"/JONA-1" is written to be expected that the command will be repeated until the home
(ORG) sensor turns ON. However, if the motor is running at a high speed, the command
may not be performed at the time the motor passes the home (ORG) sensor and the sensor
is ON.
The program jumps to the address (JUMP) if the logic derived from the current position of the motor (MT), the comparison symbol (SN), and the comparison value (PJ) are true.

MT... Motor (1 or 2)
SN... Comparison condition (<, =, or >)
PJ... Data to be compared with the current position (unit: pulse).
   0 to 16,777,215 (unsigned number) or -8,388,608 to +8,388,607 (signed number).
JUMP... See the table below.

<table>
<thead>
<tr>
<th>Description of JUMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>+[DT]</td>
</tr>
<tr>
<td>−[DT]</td>
</tr>
<tr>
<td>A[DT]</td>
</tr>
<tr>
<td>G[LB] [DT]</td>
</tr>
</tbody>
</table>

Example: /JP 1>6 0 0 0 0 G 2 1
If the position of the motor 1 is bigger than 60,000, the program jumps to the subsequent command to the label "/G 2". If the position is 60,000 or less, the next command will be performed in the following execution step.

Note:
• A User Program may act against a programmer’s intention. For example, a "/JP 2 = 5 0 0 0 + 1 /J - 2" is written to be expected that these two commands will be repeated until the motor 2 reaches the position of 5,000 pulse. However, when the motor is running at a high speed, the command may not be performed at the time the motor passes the position.
Command /JR

Format 1

/J R BF SN DU DL JUMP

The program jumps to the address (JUMP) if the logic derived from the data buffer (BF), the comparison symbol (SN), and the comparison value (DU and DL) are true.

BF... Data buffer (A to F)
Each data buffer has 1-byte data.
SN... Comparison condition (<, =, or >)
DU, DL... One-byte data to be compared with the data buffer.
DU represents the upper four bits of the data, and DL, the lower.
Each constant must range from "0" (zero) to "F" in hexadecimal.
JUMP... See the table in the previous section.

Format 2

/J R BF SN R B2 JUMP

The program jumps to the address (JUMP) if the logic derived from the data buffer BF, the comparison symbol (SN) and the other data buffer (B2) are true.

BF... Data buffer (A to F)
SN... Comparison condition (<, =, or >)
B2... Data buffer (A to F)
JUMP... See the table in the previous section.

Examples:
/JRB<EEA3
If the value in the data buffer B is less than EE (HEX value), the program jumps to the third command from the top of the User Program.

/JRR=R+C+5
If the value in the data buffer A is the same as the C's, the program jumps forward to the sixth (5+1) command from the current step.

Note:
• All data are treated as unsigned values.

Examples:

<table>
<thead>
<tr>
<th>Comparison as unsigned</th>
<th>33(H)&gt;FF(H)</th>
<th>33(H)&gt;11(H)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comparison as signed</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>Comparison as singed</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>
Command / JT

Format 1

The program jumps to the address (JUMP) if the timer (NO) that has been set by the command "/TB" is functioning (i.e., the timer is not expired).

NO... Number of timer (1 to 5)
JUMP... See the table below.

Format 2

The program jumps to the address (JUMP) if the timer that has been set by the command "/TB" is functioning (i.e., the timer is not expired).

JUMP... See the table below.

<Description of JUMP>

<table>
<thead>
<tr>
<th>Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT</td>
<td>The program jumps forward DT+1 steps.</td>
</tr>
<tr>
<td></td>
<td>DT ... Step number (0 to 999).</td>
</tr>
<tr>
<td>DT</td>
<td>The program jumps backward DT-1 steps.</td>
</tr>
<tr>
<td></td>
<td>DT ... Step number (0 to 999).</td>
</tr>
<tr>
<td>DT</td>
<td>The program jumps to the DTth command step from the top of the program.</td>
</tr>
<tr>
<td></td>
<td>DT ... Absolute address (1 to 999).</td>
</tr>
<tr>
<td>DT</td>
<td>The program jumps to the DTth command step from the label command &quot;/GLB&quot;.</td>
</tr>
<tr>
<td></td>
<td>The DT value to specify the &quot;/GLB&quot; itself is 0 (zero).</td>
</tr>
<tr>
<td></td>
<td>If no label of &quot;/GLB&quot; is found, a command error occurs.</td>
</tr>
<tr>
<td>LB</td>
<td>LB ... Label symbol: alphabet (a to z), figure (0 to 9), and other symbols except A to Z, $, #, and /.</td>
</tr>
<tr>
<td>DT</td>
<td>DT ... Step number (1 to 999).</td>
</tr>
</tbody>
</table>

Example: / J T A 2

If the timer that has been set by the command "/TB" is not expired, the program jumps to the second command from the top of the User Program.
Command /JW

Format 1

The program jumps to the address (JUMP) if the logic derived from the 4-byte data buffer W(BF), the comparison symbol (SN), and the comparison value (DT) are true.

W(BF)... 4-byte data buffer (WA to WF)
Each data in the 4-byte data buffer is -999,999,999 to +999,999,999.
SN... Comparison condition (<, =, or >)
DT... Comparison value (-999,999,999 to +999,999,999)
Signed number up to 9 digits
JUMP... See the table in the previous section.

Format 2

The program jumps to the address (JUMP) if the logic derived from the 4-byte data buffer W(BF), the comparison symbol (SN), and the other 4-byte data buffer W(B2) are true.

W(BF)... 4-byte data buffer (WA to WF)
SN... Comparison condition (<, =, or >)
W(B2)... 4-byte data buffer (WA to WF)
JUMP... See the table in the previous section.

Format 3

The program jumps to the address (JUMP) if the logic derived from the 4-byte data buffer W(BF), the comparison symbol (SN), and the target motor (MT) are true.

W(BF)... 4-byte data buffer (WA to WF)
SN... Comparison condition (<, =, or >)
MT... Target motor (1 or 2)
JUMP... See the table in the previous section.

Ex. /JWB<1 A3
If the value in the 4-byte data buffer WB is less than 1000, the program jumps to the third command from the top of the User Program.

/JWA=W F + 5
If the value in the 4-byte data buffer WA is the same as WF’s, the program jumps forward to the sixth (5+1) command from the current step.
### Command \( / L D \)

**Data buffer Load**

<table>
<thead>
<tr>
<th>Format 1</th>
<th>/</th>
<th>L</th>
<th>D</th>
<th>BF</th>
<th>.</th>
<th>DU</th>
<th>DL</th>
</tr>
</thead>
</table>

Loads the data buffer (BF) with the value (DU and DL).

- **BF...** Data buffer (A to F)
  - Each data buffer has 1-byte data.
- **DU, DL...** One-byte data to be written into the data buffer.
  - DU represents the upper four bits of the data, and DL, the lower.
  - Each constant must range from "0"(zero) to "F" in hexadecimal.

<table>
<thead>
<tr>
<th>Format 2</th>
<th>/</th>
<th>L</th>
<th>D</th>
<th>BF</th>
<th>.</th>
<th>B2</th>
</tr>
</thead>
</table>

Copies the value in the data buffer (B2) to the data buffer (BF).

- **BF...** Data buffer (A to F)
- **B2...** Data buffer (A to F)
  - Each data buffer has 1-byte data.

<table>
<thead>
<tr>
<th>Format 3</th>
<th>/</th>
<th>L</th>
<th>D</th>
<th>W</th>
<th>BF</th>
<th>.</th>
<th>DT</th>
</tr>
</thead>
</table>

Loads the 4-byte data buffer W(BF) with the value (DT).

- **W(BF)...** 4-byte data buffer (WA to WF)
  - Each data in the 4-byte data buffer is -999,999,999 to +999,999,999.
- **DT...** Data to be written into the 4-byte data buffer (-999,999,999 to +999,999,999)
  - Signed number up to 9 digits

<table>
<thead>
<tr>
<th>Format 4</th>
<th>/</th>
<th>L</th>
<th>D</th>
<th>W</th>
<th>BF</th>
<th>.</th>
<th>W</th>
<th>B2</th>
</tr>
</thead>
</table>

Copies the value in the 4-byte data buffer W(B2) to the 4-byte data buffer W(BF).

- **W(BF)...** 4-byte data buffer (WA to WF)
- **W(B2)...** 4-byte data buffer (WA to WF)
  - Each data in the 4-byte data buffer is -999,999,999 to +999,999,999.

<table>
<thead>
<tr>
<th>Format 5</th>
<th>/</th>
<th>L</th>
<th>D</th>
<th>W</th>
<th>BF</th>
<th>.</th>
<th>P</th>
<th>MT</th>
</tr>
</thead>
</table>

Copies the position data of target motor to the 4-byte data buffer W(BF).

- **W(BF)...** 4-byte data buffer (WA to WF)
- **MT...** Target motor (1 or 2)
Command /L D

Note
- The 4-byte data buffer WA to WF can be used by a speed-rated command, such as OH, OL, OX, OS, OC and a position-rated command, such as A, MA, SL, R, etc.

Ex.)
/L DWA, 1 0 0 0 Sets 1,000 to WA.
/O HWA Sets the value of WA to OH
/L DWF, 2 0 0 0 Sets WF to 2,000
/A 1 00WF Sets the data of position number 100 to WF.
/L DWB, P 1 Loads the current position data of motor 1 in WB.
Command /M U L

Multiplication in the data buffer data

Format 1 / M U L BF DU DL

Multiplies the data in the data buffer (BF) and the data (DU and DL) and writes the result to the data buffer (BF).

BF... Data buffer (A to F)
  Each data buffer has 1-byte data.
DU, DL... One-byte data for multiplication.
  DU represents the upper four bits of the data, and DL, the lower. Each constant must range from "0" (zero) to "F" in hexadecimal.

Format 2 / M U L BF . B2

Multiplies the data in the data buffer (BF) and the data buffer (B2) and writes the result to the data buffer (BF).

BF... Data buffer (A to F)
B2... Data buffer (A to F)
  Each data buffer has 1-byte data.

Format 3 / M U L W BF DT

Multiplies the data in the 4-byte data buffer W(BF) and the data (DT) and writes the result to the 4-byte data buffer W(BF).

W(BF)... 4-byte data buffer (WA to WF)
  Each data in the 4-byte data buffer is -999,999,999 to +999,999,999.
DT... Data for multiplication (-999,999,999 to +999,999,999)
  Signed number up to 9 digits

Format 4 / M U L W BF . W B2

Multiplies the data in the 4-byte data buffer W(BF) and the data in the 4-byte data buffer W(B2) and writes the result to the 4-byte data buffer W(BF).

W(BF)... 4-byte data buffer (WA to WF)
W(B2)... 4-byte data buffer (WA to WF)
  Each data in the 4-byte data buffer is -999,999,999 to +999,999,999.

Note
  - In the case of overflow, a command error will occur.
**Command /O**

**Format**

\[
/\ O \ PT \ BF
\]

Outputs the value in the data buffer (BF) to the output port or a PC.

- **PT**: Option parameter that specifies the destination of the transmission.
  - PT=1: The data is output to the output port.
  - 3: The data is sent to a PC via RS-232C.
- **BF**: Data buffer (A to F)
  - Each data buffer has 1-byte data.

**Transfer Format**

\[
DU \ DL
\]

When “3” is chosen as PT, the data is transmitted to a PC in the above format.

- **DU, DL**: One-byte value stored in the data buffer.
  - DU represents the upper four bits of the data, and DL, the lower.
  - The range of each constant is from “0”(zero) to “F” in hexadecimal.

---

**Command /OR**

**Format 1**

\[
/\ O \ R \ BF \ , \ DU \ DL
\]

Implements a logical OR operation in the data buffer (BF) and the value (DU and DL), and writes the result to the buffer (BF).

- **BF**: Data buffer (A to F)
  - Each data buffer has 1-byte data.
- **DU, DL**: One-byte data for which an OR operation is performed.
  - DU represents the upper four bits of the data, and DL, the lower.
  - Each constant must range from “0”(zero) to “F” in hexadecimal.

**Format 2**

\[
/\ O \ R \ BF \ , \ B2
\]

Implements a logical OR operation in the data buffer (BF) and the other data buffer (B2), and writes the result to the former one (BF).

- **BF**: Data buffer (A to F)
- **B2**: Data buffer (A to F)
  - Each data buffer has 1-byte data.
Command \( /R \ R \ /R \ L \)

<table>
<thead>
<tr>
<th>Data buffer</th>
<th>Rotation operation</th>
</tr>
</thead>
</table>

Format 1

\[
/R \ R \ BF
\]

Rotates the data buffer (BF) rightward by one bit.

BF... Data buffer (A to F)

Format 2

\[
/R \ L \ BF
\]

Rotates the specified buffer (BF) leftward by one bit.

BF... Data buffer (A to F)
Command  /SC

Subroutine  Call

Format  / S  C  JUMP

Calls the subroutine specified by JUMP. A quintuple nest of the calls is allowed.

<Description of JUMP>

<table>
<thead>
<tr>
<th>Description</th>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+[DT]</td>
<td>/S C JUMP</td>
<td>The program jumps forward DT+1 steps. DT ... Step number (0 to 999).</td>
</tr>
<tr>
<td>−[DT]</td>
<td>/S C JUMP</td>
<td>The program jumps backward DT-1 steps. DT ... Step number (0 to 999).</td>
</tr>
<tr>
<td>A[DT]</td>
<td>/S C JUMP</td>
<td>The program jumps to the DTth command step from the top of the program. A[DT] value of the first command in the program is 1 (one). DT ... Absolute address (1 to 999).</td>
</tr>
<tr>
<td>G[LB][DT]</td>
<td>/S C JUMP</td>
<td>The program jumps to the DTth command step from the label command &quot;/G LB&quot;. The DT value to specify the &quot;/G LB&quot; itself is 0 (zero). If no label of &quot;/G LB&quot; is found, a command error occurs. LB ... Label symbol: alphabet (a to z), figure (0 to 9), and other symbols except A to Z, $, #, and /. DT ... Step number (1 to 999).</td>
</tr>
</tbody>
</table>

Example:  /SCG11/END/G1/D00/SR/END
The first command “SCG11” calls the subroutine “/G1” that turns all bits of the output port OFF.

Command  /SR

Subroutine  Return

Format  / S  R

Ends the subroutine and jumps back to the subsequent step to the command that called the subroutine.
Command / S U B

Subtraction in the data buffer data

Format 1

/ S U B BF , DU DL

Subtracts the data (DU, DL) from the data in the data buffer (BF) and writes the result to the data buffer (BF).

BF... Data buffer (A to F)
Each data buffer has 1-byte data.
DU, DL... One-byte data for subtraction.
DU represents the upper four bits of the data, and DL, the lower. Each constant must range from "0" (zero) to "F" in hexadecimal.

Format 2

/ S U B BF , B2

Subtracts the data in the data buffer (B2) from the data in the data buffer (BF) and writes the result to the data buffer (BF).

BF... Data buffer (A to F)
B2... Data buffer (A to F)
Each data buffer has 1-byte data.

Format 3

/ S U B W BF , DT

Subtracts the data (DT) from the data in the 4-byte data buffer W(BF) and writes the result to the 4-byte data buffer W(BF).

W(BF)... 4-byte data buffer (WA to WF)
Each data in the 4-byte data buffer is -999,999,999 to +999,999,999.
DT... Data for subtraction (-999,999,999 to +999,999,999)
Signed number up to 9 digits

Format 4

/ S U B W BF , W B2

Subtracts the data in the 4-byte data buffer W(B2) from the data in the 4-byte data buffer W(BF) and writes the result in the 4-byte data buffer W(BF).

W(BF)... 4-byte data buffer (WA to WF)
W(B2)... 4-byte data buffer (WA to WF)
Each data in the 4-byte data buffer is -999,999,999 to +999,999,999.

Note

- In the case of overflow, a command error will occur.
12. User Program Command Reference

**Command /T**

<table>
<thead>
<tr>
<th>Format</th>
<th>/</th>
<th>T</th>
<th>D T</th>
</tr>
</thead>
</table>

Waits for the specified period (DT) and then proceeds to the next command.

**DT...** Timer data (0 to 32,767(approx. 54 minutes)).
The unit is 0.1 second. That is, the maximum setting is 327.67 seconds or about 54 minutes.

**Note:**
- The communication command “T” differs in function from the User Program command "/T".
- The timer resource inside RC-234 is shared by the command "T", "P", "/T", "/TB", and "/TS". Therefore these commands can not be used simultaneously.

**Command /TB**

<table>
<thead>
<tr>
<th>Format 1</th>
<th>/</th>
<th>T</th>
<th>B</th>
<th>NO</th>
<th>D T</th>
</tr>
</thead>
</table>

Starts to count after setting timer (NO).
The command is used in conjunction with the command "JT".

**NO...** Number of timer (1 to 5)
**DT...** Value to be set for the timer (0 to 32,767).
The unit is 0.1 second. That is, the maximum setting is 327.67 seconds or about 54 minutes.

**Format 2**

| / | T | B | D T |

Sets the timer to the period (DT) and starts to count.
The command is used in conjunction with the command "JT".

**DT...** Value to be set for the timer (0 to 32,767).
The unit is 0.1 second. That is, the maximum setting is 327.67 seconds or about 54 minutes.

**Example:** / T B 1 0 0 / G 1 / 4 / J T G 1 1 / E N D

An action--a motor starts, rotates clockwise by certain pulses, and stops--is repeated for 10 seconds.

**Note:**
- In format 1, the numbered timer is independent respectively, but in format 2, the timer resource inside RC-234 is shared by the command "T", "P", "/T", "/TB", and "/TS". Therefore these commands can not be used simultaneously.
Command /TS

Timer  Idling after motor stops

Format / T S DT

The command enhances the function of command "/(null)". Whenever a motor stops, the program waits for the period specified by DT, and then proceeds to the next command. Once the command "/TS" is executed, the function is applied to all of the "/(null)" commands that subsequently appear in the program.

DT... Time for idling (0 to 32,767).
The unit is 0.1 second. That is, the maximum setting is 327.67 seconds or about 54 minutes.

Example: /TS 10 /0 /2 2 0 0 0 /3 /END
This example has the same function as that of another User Program--"/ 0 / T 1 0 / 2 2 0 0 0 / 3 / T 1 0 / E N D". The action is that the program performs home-search, waits for one second, moves to the 2,000-pulse position, waits for one second, and ends.

Note:
・ The timer resource inside RC-234 is shared by the command "T", "P", "/T", "/TB", and "/TS". Therefore these commands can not be used simultaneously.

Command /WAW /WDW

EEPROM  Write

Format 1 / W A W
Starts to write the position data to EEPROM, waits for the completion, and then proceeds to the next command. The function is equivalent to that of the communication command "AW".

Format 2 / W D W
Starts to write information such as speed data to EEPROM, waits for the completion, and then proceeds to the next command. The function is equivalent to that of the communication command "DW".

Command /?

Termination due to error

Format /?
Terminates the User Program if the condition flag contains any of errors. Otherwise the subsequent command will be performed in the following execution step.
For more information about the condition flag, see the section "/JE" or "/9".
<Appendix>

13. Program
14. Link Master RC-002
15. Control Command List
16. Command Default Setting
Appendix. Contents by Command Function
13. Program

13-1 Notes for programming

- Flow command

The communication protocol must follow RS232C protocol with two lines RXD and TXD. The programmer must monitor the out overflow of the buffer while programming. Be careful not to send the next command before receiving reply.

**Correct Flow of Commands**

1. Send "$10"
2. Wait for RC-234 reply
3. Send "$1"
4. Wait for RC-234 reply

**Incorrect Flow of commands**

1. Send "$10"
2. Send "$1"
3. Wait for RC-234 reply

Sending "$1" before RC-234 returns the reply "->" for the command "$10" like the above "incorrect flow of commands" will result in a communication error.

- Program for sending commands

  Be sure to use the correct format when sending commands to the RC-234. The following table describes typical codes.

<table>
<thead>
<tr>
<th>No</th>
<th>Sample code</th>
<th>Quick Basic</th>
<th>N88Basic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Propriety</td>
<td>Transmitted characters</td>
</tr>
<tr>
<td>1</td>
<td>PRINT #1,&quot;$10&quot;;CHR$(&amp;HD);</td>
<td>O $10 CR</td>
<td>O $10 CR</td>
</tr>
<tr>
<td>2</td>
<td>PRINT #1,&quot;$10&quot;;CHR$(&amp;HD)</td>
<td>× $10 CR CR</td>
<td>× $10 CR CR LF</td>
</tr>
<tr>
<td>3</td>
<td>PRINT #1,&quot;$10&quot;</td>
<td>O $10 CR</td>
<td>× $10 CR LF</td>
</tr>
<tr>
<td>4</td>
<td>PRINT #1,&quot;$10&quot;;</td>
<td>× $10</td>
<td>× $10</td>
</tr>
</tbody>
</table>

O : Valid command  
× : Invalid command  
CR : Carriage return  ASCII code: 13(HEX 0DH)  
LF : Line feed  ASCII code: 10(HEX 0AH)

Valid commands are marked by "O". The RC-234 treats other commands as invalid commands.
13-2 Sample Program using Quick Basic

Commands for each mode and Sample Program

Mode 0 (Control of stepping motor driver)

<Configuration>

In mode 0, RC-234 can control two stepping motor drivers alternately using stall disc. Please equip a stall disc according your needs. (When “EM”=1, the simultaneous 2-axis control is available.)

As for a stall disc, refer to the command “Q” in the chapter 10 “Command Reference”
To set the mode, use command “E”.

Commands exclusively used in mode 0, 2

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Q *****”</td>
<td>Sets the on-off period of a stall sensor in units of pulse.</td>
</tr>
<tr>
<td></td>
<td>• 100 to 65,535 pulse</td>
</tr>
<tr>
<td></td>
<td>• The setting should be severally designated for each motor.</td>
</tr>
<tr>
<td>“Q D”</td>
<td>Queries the current stall data for a specified motor.</td>
</tr>
<tr>
<td>“Q D 1”</td>
<td>Queries the current stall data for motor 1.</td>
</tr>
<tr>
<td>“Q D 2”</td>
<td>Queries the current stall data for motor 2.</td>
</tr>
<tr>
<td>“Q S E **”</td>
<td>Enable a stall check with encoder for each motor in mode 0.</td>
</tr>
<tr>
<td>“Q S E”</td>
<td>Queries whether the stall check is enable for each motor in mode 0.</td>
</tr>
<tr>
<td>“θ Q”</td>
<td>Locates the stall sensor at the middle of stall slit for stall detection.</td>
</tr>
<tr>
<td>“θ Q *****”</td>
<td>Set the pulses during which the stall input is assuredly activated.</td>
</tr>
<tr>
<td></td>
<td>• 1 to 65,535 pulse</td>
</tr>
<tr>
<td>“θ Q D”</td>
<td>Queries the setting value of pulses during which the stall input is activated.</td>
</tr>
<tr>
<td>“θ Q W”</td>
<td>Queries the actual pulse number during which the stall input has been kept activated.</td>
</tr>
</tbody>
</table>

For more details, refer to the chapter 10 “Command Reference”.
The above commands except Command “QSE” can be used in mode 2.
This is the sample program when wiring like the configuration in the front page. We assume that the body number of RC-234 is 1. When executing this program, set the body number (yellow rotary switch) to 1.

Sequence of moves:

<table>
<thead>
<tr>
<th>Motor 1</th>
<th>Motor 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>① Home Search</td>
<td>③ Home Search</td>
</tr>
<tr>
<td>② Search stall sensor</td>
<td>④ Search stall sensor</td>
</tr>
<tr>
<td>⑥ Move to position 101</td>
<td>⑤ Move to position 201</td>
</tr>
<tr>
<td>(5,000 steps from home position)</td>
<td>(8,000 steps from home position)</td>
</tr>
<tr>
<td>⑦ Return to home position</td>
<td>⑧ Move to position 202</td>
</tr>
<tr>
<td></td>
<td>(12,000 steps from home position)</td>
</tr>
<tr>
<td>⑨ Move to position 101</td>
<td></td>
</tr>
<tr>
<td>(5,000 steps from home position)</td>
<td></td>
</tr>
<tr>
<td>⑩ Return to home position</td>
<td></td>
</tr>
</tbody>
</table>

Repeat the sequence if sequence was executed without limit, stall or emergency stop error.

' RC-234 MODE 0 & 2MOTOR RUNNING (234E0.BAS)
CLS
',
'---open com port for communication---
OPEN "com1:9600,n,8,1" FOR RANDOM AS #1
CR$ = CHR$(13)    'carriage return (CR)
',
'-------start of the main program-------
MAIN:
READ D$
WHILE D$ <> "END"
  'continue running the program until "END" encountered
  CMMD$ = "$1" + D$
  'formulate the command
  PRINT " "; CMMD$
  'print the command on screen before sending to RC-234
  PRINT #1, CMMD$ + CR$;   'send the command to RC-234
  RCV = 1
  GOSUB RECEIVE

  PRINT REPLY$  'print the reply form controller on the screen
  GOSUB CHECK   'error check
  READ D$
WEND
E = E + 1
PRINT "No of cycles "; E  'print the number of cycles
RESTORE REPEATDATA
GOTO MAIN
',
'------Wait for RC-234 reply-----
RECEIVE:
WHILE LOC(1) <> RCV  'wait for RC-234 reply
  REPLY$ = INPUT$(LOC(1), #1)
WEND
RETURN
'-----Error check routine-----

CHECK:  'check error buffer while motor is running
CD0 = 1  'check error buffer while motor is spinning CD0=1
         'check error buffer while motor is idle CD=0

WHILE CD0 = 1
   PRINT #1, "$1"; CR$;
   RCV = 5  'number of character returned by controller
   GOSUB RECEIVE
   IF MID$(REPLY$, 1, 1) <> ">" THEN PRINT "Communication Error": END
   B$ = MID$(REPLY$, 4, 1): CMNDD = VAL("&H" + B$)
   CD0 = CMNDD AND 1  'check error buffer while motor is running
   CDE = CMNDD AND &HE  'error check buffer
   IF CDE <> 0 THEN GOTO STR ELSE GOTO IFEND

STR:
   CD1 = CDE AND 2  'limit/stall error
   CD2 = CDE AND 4  'emergency stop error
   CD3 = CDE AND 8  'command error
   IF CD1 = 2 THEN PRINT "LIMIT/STALL ERROR"
   IF CD2 = 4 THEN PRINT "EMERGENCY STOP ERROR"
   IF CD3 = 8 THEN PRINT "COMMAND ERROR"
   END

IFEND:
   WEND
   RETURN

DATA "E0","EP0","QS11"  'set stall data
DATA "F1","Q1024"
DATA "F2","Q1024"
DATA "A1015000","A2018000","A20212000"  'set position data
DATA "OC50","OX300"  'set speed data
DATA "F1","OL500","OH4000","OS930"
DATA "F2","OL500","OH4000","OS930"
DATA "F1","0S50","0","0Q","OL5"  'home search for motor1
DATA "F2","0S50","0","0Q","OL5"  'home search for motor2

REPEATDATA:  'loop action
   DATA "F2","B201"  'motor2
   DATA "F1","B101","1"  'motor1
   DATA "F2","B202"  'motor2
   DATA "F1","B101","1"  'motor1
   DATA "F2","1"  'motor2
   DATA "END"

Explanation of command used in program:

[DATA "E0","EP0","QS11"]

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0</td>
<td>set mode 0</td>
</tr>
<tr>
<td>EP0</td>
<td>set uni-directional counting 0 to 16,777,215</td>
</tr>
<tr>
<td>QS11</td>
<td>enable stall detection for motor 1 and motor 2</td>
</tr>
</tbody>
</table>

Remarks:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EP0</td>
<td>0 to 16,777,215</td>
</tr>
<tr>
<td>EP1</td>
<td>-8,388,608 to +8,388,607</td>
</tr>
<tr>
<td>QS10</td>
<td>Execute stall detection for motor2</td>
</tr>
<tr>
<td>QSR</td>
<td>Disable stall detection for motor1, 2</td>
</tr>
<tr>
<td>QS</td>
<td>Query the status of the stall detection</td>
</tr>
</tbody>
</table>
Mode 0

[DATA "F1","Q1024"]

F1  select motor 1
Remarks:  F2  select motor 2
Q1024  stall sensor ON-OFF pulse distance to 1,024
Remarks:  QD  Query the setting value of on-off pulse distance
          QD1 Query the setting value of on-off pulse distance for motor 1
          QD2 Query the setting value of on-off pulse distance for motor 2

<Position data>

[DATA "A1015000","A2018000","A20212000"]

A1015000  A  command to set position data
101  position no. (000 to 999 up to 1,000 points)
5,000  position pulse number
Remarks:  A101D  Query the setting value of the position pulse number

<Speed data>

For more details to set the speed data, refer to the chapter 7 “Stepping Motor Speed Adjustment”.

[DATA "OC50","OX300"]

OC50  set S-curve parameter to 50%
The setting value of command “OC” is common data for motor 1,2
(The setting can’t be severally designated for each motor.)
OX300  set multiplication factor 300
The setting value of command “OX” is common data for motor 1,2
(The setting can’t be severally designated for each motor.)

[DATA "F1","OL500","OH4000","OS930"]

OL500  set low speed data to 500
The setting should be severally designated for each motor.
OH4000  set high speed data to 4,000
The setting should be severally designated for each motor.
OS930  set accel. data to 930
The setting should be severally designated for each motor.

<Home search, stall sensor>

As for home search, refer to the command “0” in the chapter 10 “Command Reference”

[DATA "F1","0S50","0","0Q","OL5"]

0S50  distance from home (ORG) sensor edge that will be home position
Remarks:  set a multiplication factor for the “0S” parameter.
0  search for the home (ORG) sensor and establish the home position,
where the pulse counter equals to 0.
Remarks:  Command “(NULL)” query the status of operation-busy, idle,
limit error, command error, etc.
This sample program always sends command “(NULL)” to observe the motor rotation
and when the motor is running, waits until the motor is stopped and sends
a next command.
If a move command is sent when the motor is running, command error will occur
and the sent command is invalid. Therefore, the procedure to observe the motor
rotation is needed.
0Q  rotate the stall sensor at the middle of the slit for stall detection.

<Execution of motor move>

[DATA "F1","B101","1"]

B101  move to absolute position 101
1  return to home
Remarks:  11  motor 1 returns to home position
          12  motor 2 returns to home position
Mode 1 (Control of stepping motor using encoder)

<Configuration>

In mode 1, RC-234 can control a stepping motor using encoder. Also, the controller can control two stepping motor drivers alternately. (When “EM”=1, the simultaneous 2-axis control is available.)

This section describes the program to control one stepping motor.
In this mode, the position pulse number is managed by the input pulse from encoder and as for the data of speed and home search, the output pulse from RC-234 is used.

Commands exclusively used in mode 1

<table>
<thead>
<tr>
<th>Command</th>
<th>Describes</th>
</tr>
</thead>
<tbody>
<tr>
<td>“QJT”</td>
<td>Check the delay time after which the automatic stall adjustment is carried out.</td>
</tr>
<tr>
<td>“QJT***”</td>
<td>Set the delay time after which the automatic stall adjustment is carried out.</td>
</tr>
<tr>
<td>“QJ”</td>
<td>Query the current adjust data.</td>
</tr>
<tr>
<td>“QJ***”</td>
<td>Enable or disable automatic stall adjust feature.</td>
</tr>
<tr>
<td>“QJR”</td>
<td>Disable automatic adjust.</td>
</tr>
<tr>
<td>“PA”</td>
<td>Query the current mode of counting.</td>
</tr>
<tr>
<td>“PA***”</td>
<td>Set the mode of counting. (Uni., Bi. or quadrature counting) The setting should be severally designated for each motor.</td>
</tr>
<tr>
<td>“PB”</td>
<td>Query the current ratio setting.</td>
</tr>
<tr>
<td>“PB***”</td>
<td>Set the ratio between motor resolution and encoder resolution. The setting should be severally designated for each motor.</td>
</tr>
<tr>
<td>“QE”</td>
<td>Set stall data.</td>
</tr>
<tr>
<td>“QE***”</td>
<td>Query the current stall data setting.</td>
</tr>
</tbody>
</table>

Refer to the chapter 10 “Command Reference” for more details.
<Sample program in mode 1>

This is the sample program when wiring like the configuration in the front page. We assume that the body number of RC-234 is 1. When executing this program, set the body number (yellow rotary switch) to 1.

Sequence of moves:

<table>
<thead>
<tr>
<th>Motor 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home search ↓</td>
</tr>
<tr>
<td>Move to position 102 (120,000 steps from home position) ↓</td>
</tr>
<tr>
<td>Return to home position ↓</td>
</tr>
<tr>
<td>Move to position 101 (20,000 steps from home position) ↓</td>
</tr>
<tr>
<td>Return to home position</td>
</tr>
</tbody>
</table>

The input pulse from encoder is used to manage the position and detect a stall in mode 1.

The sample is similar to “Sample program in mode 0” except lines for setting data.

DATA “E1”
DATA “QE8”, “QS01”, “QJ01”, “QJT30”
DATA “A10120000”, “A102120000”
DATA “F1”, “OC0”, “OX100”, “OL500”, “OH5000”, “OS100”
DATA “PA4”, “PB10”
DATA “0S100”, “0”
REPEATDATA:
DATA “OH5000”, “B102”, “1”
DATA “OH7000”, “B101”, “1”
DATA “END”

Explanation of command used in program:
<Setting Data>
[DATA "E1"]
E1 set mode 1

<Stall detection & Adjust feature>
[DATA “QE8”, “QS01”, “QJ01”, “QJT30”]
QE8 set the maximum limit of the difference to 8 (±1,024 pulses)
Remarks: If the “difference between a pulse count and an encoder count” exceed the data specified by a command “QE”, RC-234 recognizes that a stall error has occurred.
The controller checks for a stall error every 50msec.
QE Query the current maximum limit of the difference
QS01 enable stall check for motor 1
QJ01 enable stall adjust for motor 1
Remarks: QJ10 disable stall adjust for motor2
QJR disable stall adjust for motor1 and 2
QJT30 set start time for stall adjust to 300 msec after the stopping
Remarks: QJT Query the delay time

<Setting regarding encoder>
[DATA “PA4”, “PB10”]
PA4 set quadrature counting for encoder feedback
PB10 set encoder ratio to 10
(encoder ratio = motor resolution / encoder resolution)
Mode 2 (Control of pulse input servo motor driver)

<Configuration>

This section describes the program to control servo motor by connecting with pulse input servo motor driver.

<Servo Driver Terminals>
- **INP** --- The INP input is connected COIN output of servo driver amplifier. The servo amplifier turns on COIN output whenever a commanded move is compelled or motor is stopped. Use “C1” or “C11” command to check status of INP input.

- **CLR** --- The CLR is an output on RC-234 controller and is connected to CLR input (counter clear) of servo amplifier. Turn CLR ON by the time set by the command “DCT” when following conditions are encountered.
  1. When home sensor is ON. CLR output will reset servo drivers internal counter register.
  2. When any of the limit sensor is ON.
  3. When Emergency Stop is ON.
  4. After every move the controller waits for 3 seconds to receive CION output (INP ON).
     If the controller receives no input the limit error flag set.
  5. When RC-234 receives a Stop command “S” during a move.

Use command “DS01” to set CLR output ON for motor 1.

**Commands exclusively used in mode 2**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“C I”</td>
<td>Queries the status of INP inputs bits.</td>
</tr>
<tr>
<td>“C I 1”</td>
<td>Queries the status of INP1 inputs bits.</td>
</tr>
<tr>
<td>“C I 2”</td>
<td>Queries the status of INP2 inputs bits.</td>
</tr>
<tr>
<td>“D S”</td>
<td>Queries the status of CLR output bits.</td>
</tr>
<tr>
<td>“D S * *”</td>
<td>Turns ON/OFF the CLR for each motor compulsively.</td>
</tr>
<tr>
<td>“D S R”</td>
<td>Turns OFF the CLR for each motor compulsively.</td>
</tr>
</tbody>
</table>

Refer to the chapter 10 “Command Reference” for more details.

**Note:** In mode 2, the stall detection using Command “Q”, “Q” is valid.
For more details, refer to the explanation of mode 0 in this chapter.
Mode 2

Sample program in mode 2>

Sequence of moves:

| Motor 1 |  
|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|
| Set Home search data using index (EZ) channel | ↓ | Move to position 101 (5,000 steps from home position) | ↓ | Return to home position | ↓ |
| Home search using index channel and home (ORG) sensor | ↓ | Move to position 201 (8,000 steps from home position) | ↓ | Return to home position | ↓ |

Repeat the sequence if sequence was executed without limit, stall or emergency stop error.

The sample is similar to “Mode 0 sample program” except the lines for setting data.

DATA "E2","EA51B"
DATA "A1015000","A2018000"
DATA "F1","OC100","OX100","OL50","OH4000","OS1240"
DATA "OZ3","OZ"
REPEATDATA:
DATA "OH5000","B101","1"
DATA "OH7000","B201","1"
DATA "END"

Explanation of command used in program:

<Data Setting>

[DATA "E2","EA51B"]

| E2 | set mode 2 |
| EA51B | set the input logic of INP to active high |
| Remarks: EA50B | set the input logic of INP to active low |
| EA | Query the current input logic for each sensor |

[DATA "OZ3","OZ"]

| OZ3 | set Home search data using index pulse – 3 |
| OZ3 | means the controller will look for 3 index pulses after detecting home sensor leading edge. |
| 0Z | home search using encoder index pulse (EZ) |
**Example Program for Special Function 1**

1. **Sum Check Program**
   This example illustrates the sum check and detects errors in communication.

```basic
'RC-234 SUM CHECK PROGRAM (234SUM.BAS)
'Set RC-234 body number to 1
CLS

'---open com port for communication---
OPEN "com1:9600,n,8,1" FOR RANDOM AS #1

CR$ = CHR$(13) 'carriage return
PRINT #1, "$1EL1"; CR$;    'set carriage return for all commands
GOSUB RECEIVE

PRINT #1, "$1SUM1"; CR$;    'set sum check ON
GOSUB RECEIVE

'-----MAIN-----
PRINT
LINE INPUT "COMMAND = ", CMMD$ 'input command

WHILE LEFT$(CMMD$, 2) = "$1" 'repeat the program as long as the command is preceded by $1
    GOSUB SUM 'add sum check data to the command
    PRINT " - "; CMMD$;
    PRINT #1, CMMD$;  'send the command to RC-234
    GOSUB RECEIVE
    GOSUB SUMCK 'call SUMCK subroutine to check if command was sent correctly
    PRINT
    LINE INPUT "COMMAND = ", CMMD$
WEND

'-----The program is terminated-----
PRINT #1, "$1SUM0"; CR$;  'cancel sum check
GOSUB RECEIVE
CLOSE
END

'-----Wait for RC-234 reply-----
RECEIVE:
   REPLY$ = ""
   RCV = 0 'Buffer to check the reply
   'RCV=0: not receive the reply to the end
   'RCV=1: receive only '?' for reply
   'RCV>1: receive the reply to the end

   WHILE RCV = 0
       WHILE LOC(1) = 0 'wait for controller reply
           WEND
       REPLY$ = REPLY$ + INPUT$(LOC(1), #1)
       RCV = INSTR(REPLY$, CR$) 'check if carriage return is accompanied or not
       IF REPLY$ = "?" THEN RCV = 1
   WEND
RETURN

'-----Change command to sum check function-----
   'This subroutine creates the last two hex digit of the sum check data.
SUM:
   SUM1 = 0
```
Example Program for Special Function 1

```
SUM = LEN(CMMD$) 'The number of command characters

FOR J = 1 TO SUM 'calculate the sum of ASCII Code of commands
    SUM1 = SUM1 + ASC(MID$(CMMD$, J, 1))
NEXT J

SUM$ = RIGHT$(HEX$(SUM1), 2)
CMMD$ = CMMD$ + SUM$ + CR$ 'Add ASCII Code at the end of the command
RETURN

'-----display sum check and reply-----
'This subroutine check by comparing the actual sum check data and command received
'by the controller.
SUMCK:
    IF RCV > 1 THEN GOTO STR ELSE GOTO IFEND 'If the reply is ?, go to IFEND
STR:
    CHECK = LEN(REPLY$)
    SUMCK$ = MID$(REPLY$, CHECK - 2, 2) 'actual reply
    PRINT SUMCK$
    SUM1 = 0
    FOR J = 1 TO CHECK - 3 'check sum calculation
        SUM1 = SUM1 + ASC(MID$(REPLY$, J, 1))
    NEXT J
    SUM$ = RIGHT$(HEX$(SUM1), 2) 'sum check reply
    PRINT SUM$
    REPLY$ = LEFT$(REPLY$, CHECK - 3)
    'Compare the actual reply and sum check reply
    'and they are different, display the error on the screen
    IF SUM$ <> SUMCK$ THEN PRINT "SUM CHECK ERROR"
IFEND:
PRINT REPLY$
RETURN
```
Example Program for Special Function 2

2. Echo Back Program

This example will illustrate the program to replace carriage return (CR), reply from RC-234, to a mark and display it.

' RC-234 ECHO BACK PROGRAM (234EE.BAS)
CLS
'
'---open com port for communication---
OPEN "com1:9600,n,8,1" FOR RANDOM AS #1
ON COM(1) GOSUB RECEIVE
COM(1) ON
'
CR$ = CHR$(13) 'carriage return
'
'-----MAIN-----
PRINT
LINE INPUT "COMMAND = ", CMMD$
WHILE LEFT$(CMMD$, 1) = "$" 'if command prefix is other than $ the program is terminated
PRINT " "; CMMD$
RCV = 0
REPLY$ = ""
PRINT #1, CMMD$; CR$; 'send the command to RC-234
'
WHILE RCV = 0 'wait for controller reply
'RECV=0: receive no reply
'RECV=1: receive reply
'
WEND
'
PRINT REPLY$
PRINT
LINE INPUT "COMMAND = ", CMMD$
WEND
'
CLOSE
END
'
'-----Wait for controller reply-----
RECEIVE:
WHILE LOC(1) <> 0
  N$ = INPUT$(LOC(1), #1)
  RCV = 1 'carriage return check
  A = LEN(N$)
  FOR J = 1 TO A
    B$ = MID$(N$, J, 1)
    IF B$ = CR$ THEN B$ = "\n"
    REPLY$ = REPLY$ + B$
  NEXT J
WEND
RETURN
Example Program for Special Function 3
3. Sample for Down Loading Program

This example will illustrate to execute the user program after storing it in EEPROM of RC-234.

' RC-234 USER PROGRAM (234I.G.BAS)
CLS
'
'---open com port for communication---
OPEN "com1:9600,n,8,1" FOR RANDOM AS #1
CR$ = CHR$(13)   'carriage return
'
'-----MAIN-----
READ D$
WHILE D$ <> "END"  
  DT = 0
    'If first character of data is "/" and also if the last four character
    'is not equal to /END then do not add carriage return.
    IF LEFT$(D$, 1) = "/" AND RIGHT$(D$, 4) <> "/END" THEN DT = 1
    IF DT = 0 THEN CMMD$ = D$ + CR$ ELSE CMMD$ = D$
    PRINT " - "; CMMD$
    PRINT #1, CMMD$;  'send the command to RC-234
    'if DT=1 the PC need not wait for response from the controller
    IF DT = 0 THEN GOTO STR ELSE GOTO IFEND
    IF D$ = "$1IW" THEN RCV = 5 ELSE RCV = 1
    GOSUB RECEIVE
    IFEND:

RECEIVE:
WHILE LOC(1) <> RCV   
  'wait for the controller reply
  WEND
  REPLY$ = INPUT$(LOC(1), #1)
RETURN

DATA "$1GES","$1I"
DATA "/E0/OH8000/OL300/OS300/OX60/A00130000/CS1,10"
DATA "/0//B001//1//CD1/JC1+1/J-7/END"
DATA "$1IW","$1G"
DATA "END"

Explanation of command used in program:

DATA "$1GES","$1I"
$1GES suspend any running program
$1I load user program to RAM

DATA "$1IW","$1G"
$1IW transfer the user program RAM to EEPROM
$1G run user program
14. Link Master RC-002

Introduction

Since RS-232C used for the data transfer in general transmits signals by voltage level, it is subject to voltage drop by line resistor or noise voltage and not adequate to long-distance data transfer and generally transfer the data to one data receiver within 10 m.

The adapter that converts this RS-232C voltage level signal into the current transmission signal that has many advantages is a link master RC-002.

The current voltage signal outputted and inputted from a link master is isolated by the high-speed photo-coupler of Generate master and I/O master and signals are transmitted. And since link master, Generate master and I/O master have a built-in stabilized power supply of wide operating range, they are less subject to potential difference and noise and the power supply can be used with common power supply of a stepping motor driver, a solenoid valve, etc.

As mentioned above, the long-distance data transfer is available by combining this link master RC-002 with generate masters or I/O masters, because a link master can transfer the data to two or more receivers and it is less subject to noise.
### Specification

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Supply voltage</strong></td>
<td>18 to 40VDC (including ripple)</td>
</tr>
<tr>
<td><strong>Supply current</strong></td>
<td>20mA or less at 24V</td>
</tr>
<tr>
<td><strong>Method of communication</strong></td>
<td>Current loop transmission</td>
</tr>
<tr>
<td>Personal Computer Side</td>
<td>RXD input level: 2.2V ON, 0.6V OFF (hysteresis input) TXD output level: ±11V(±10%) TXD current level: ±7mA(±10%)max.</td>
</tr>
<tr>
<td>Generate Master Side</td>
<td>RXD output current: ±20mA(±20%) TXD output current: ±20mA(±20%)</td>
</tr>
<tr>
<td><strong>Baud rate</strong></td>
<td>40 kbps max.</td>
</tr>
<tr>
<td><strong>Weight</strong></td>
<td>Approx. 250g</td>
</tr>
<tr>
<td><strong>Outside dimensions</strong></td>
<td>27.5(H) x 105(W) x 56(D) mm</td>
</tr>
</tbody>
</table>

### Dimensions

![Dimensions Diagram](image)

(Mm)
### 15. Control Command List

EP  ····· Command attached circle (O) symbol can be used by [command] format in user program.
$  ····· Command prefix
B#  ····· Body No. (0 to E) This is the number set by rotary switch of RC-234.
\[ \]  ····· Carriage Return (0DH by ASCII) This means the termination of command.

Reply from RC-234  ····· One letter ‘>‘ is returned. This reply means that RC-234 received a command.

**Common General Command (Part 1)**

<table>
<thead>
<tr>
<th>Contents</th>
<th>Command</th>
<th>Reply</th>
<th>EP</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode setting</td>
<td>$ B#</td>
<td></td>
<td>&gt;</td>
<td>Sets motion mode of RC-234 &quot;Control stepping motor, Control with encoder, Using servo motor&quot;, etc.</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td></td>
<td>O</td>
<td>Sets the sensor input logic in the lump.</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td></td>
<td>O</td>
<td>Sets the sensor input logic every sensors.</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td></td>
<td>O</td>
<td>Sets the sensor input logic with the same as RC-231HA.</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td></td>
<td>O</td>
<td>Sets the sensor input logic with the same as RC-231.</td>
</tr>
<tr>
<td></td>
<td>A H</td>
<td></td>
<td>O</td>
<td>Sets the pulse output with 2 clock logic(CW, CCW).</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td></td>
<td>O</td>
<td>Sets the pulse output with PULSE &amp; DIR logic.</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td></td>
<td>O</td>
<td>Uses echo function. Sends format and responses return together.</td>
</tr>
<tr>
<td></td>
<td>E 1</td>
<td></td>
<td>O</td>
<td>Resets echo function.</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td></td>
<td>O</td>
<td>Adds [ ] after general command response [&gt;] and sets output function.</td>
</tr>
<tr>
<td></td>
<td>E L</td>
<td></td>
<td>O</td>
<td>Adds [ ] after general command response [&gt;] and resets output function.</td>
</tr>
<tr>
<td></td>
<td>L 1</td>
<td></td>
<td>O</td>
<td>Enables the simultaneous 2-axis interpolation.</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td></td>
<td>O</td>
<td>Enables the alternate 2-axis interpolation.</td>
</tr>
<tr>
<td></td>
<td>E R</td>
<td></td>
<td>O</td>
<td>Sets “Command error display function” to return [&gt;@] as the response when the executed command is command error.</td>
</tr>
<tr>
<td></td>
<td>E R 0</td>
<td></td>
<td>O</td>
<td>Resets “Command error display function”.</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td></td>
<td>O</td>
<td>Amends communication baud rate. After that, when you turn on the</td>
</tr>
<tr>
<td></td>
<td>S</td>
<td></td>
<td>O</td>
<td>Selects motor 1 or 2.</td>
</tr>
<tr>
<td></td>
<td>U M</td>
<td></td>
<td>O</td>
<td>Uses sum check function. After you execute this command.</td>
</tr>
<tr>
<td></td>
<td>S U M</td>
<td></td>
<td>O</td>
<td>Resets sum check function.</td>
</tr>
</tbody>
</table>
# 15. Control Command List

## Descriptions

<table>
<thead>
<tr>
<th>Command</th>
<th>Common General Command (Part 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EP</td>
<td>Related Home Search</td>
</tr>
<tr>
<td>Rally</td>
<td>Move to the home position at High Speed.</td>
</tr>
<tr>
<td></td>
<td>Motor 1 moves to the home position at High Speed.</td>
</tr>
<tr>
<td></td>
<td>Motor 2 moves to the home position at High Speed.</td>
</tr>
<tr>
<td></td>
<td>Following the home position, the motor will decelerate to zero speed.</td>
</tr>
<tr>
<td></td>
<td>Sets the multiplication factor.</td>
</tr>
<tr>
<td></td>
<td>Sets the number of EZ pulses when ORG signal is ON.</td>
</tr>
<tr>
<td></td>
<td>Executes command ( \Theta ) and sets the distance the motor would move after sensing the leading edge of the home sensor.</td>
</tr>
<tr>
<td></td>
<td>Searches the home position using Z-p of encoder (EZ) input and home ORG sensor input signal.</td>
</tr>
<tr>
<td></td>
<td>Stops the motor immediately. (It would cause system damage if high inertia load is being moved.)</td>
</tr>
<tr>
<td></td>
<td>Stops the motor slowly. The motor will decelerate to zero speed.</td>
</tr>
<tr>
<td></td>
<td>Less than 5 digits</td>
</tr>
<tr>
<td></td>
<td>Less than 8 digits</td>
</tr>
</tbody>
</table>

## Related Home Search

- Performing Home Search (1) for Motor 1.
- Performing Home Search (2) for Motor 1.
- Performing Home Search (1) for Motor 2.
- Performing Home Search (2) for Motor 2.
- Performing High Speed Home Search (1) for Motor 1.
- Performing High Speed Home Search (1) for Motor 2.
- Performing High Speed Home Search (2) for Motor 1.
- Performing High Speed Home Search (2) for Motor 2.
- Executing command \( \Theta \) and sets the distance the motor would move after sensing the leading edge of the home sensor.
- Searches the home position using Z-p of encoder (EZ) input and home ORG sensor input signal.
- Sets the multiplication factor.
- Sets the number of EZ pulses when ORG signal is ON.
- Moves to the home position at High Speed.
- Motor 1 moves to the home position at High Speed.
- Motor 2 moves to the home position at High Speed.
- Resets motion counter to zero.
- Stops the motor immediately. (It would cause system damage if high inertia load is being moved.)
- Stops the motor slowly. The motor will decelerate to zero speed.
### Common General Command (Part 3)

#### Contents

<table>
<thead>
<tr>
<th>Command</th>
<th>Reply</th>
<th>EP</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B#\ C\ Y\ [3\ digits]\ \ R\</td>
<td>&gt;</td>
<td>O</td>
<td>Enables the interruption input and stores the data to the specified position number.</td>
</tr>
<tr>
<td>$B#\ C\ Y\ \ R\</td>
<td>&gt;</td>
<td>O</td>
<td>Enables the interruption input and turns the specified general output ON/OFF when the interruption of specified bit has occurred.</td>
</tr>
<tr>
<td>$B#\ C\ Y\ \ S\</td>
<td>&gt;</td>
<td>O</td>
<td>Enables the interruption input and stops the designated motor when the interruption of specified bit has occurred.</td>
</tr>
<tr>
<td>$B#\ C\ Y\ \ L\</td>
<td>&gt;</td>
<td>O</td>
<td>Disables the interruption input.</td>
</tr>
<tr>
<td>$B#\ C\ Y\ \ L\ \ [Less\ than\ 8\ digits]\ \ R\</td>
<td>&gt;</td>
<td>O</td>
<td>Sets the data memory position range that enables the interruption input.</td>
</tr>
<tr>
<td>$B#\ D\</td>
<td>&gt;</td>
<td>O</td>
<td>Sets the whole output ports.</td>
</tr>
<tr>
<td>$B#\ D\ \ B\</td>
<td>&gt;</td>
<td>O</td>
<td>Sets the individual bits.</td>
</tr>
<tr>
<td>$B#\ P\ \ [Less\ than\ 5\ digits]\ \ R\</td>
<td>&gt;</td>
<td>O</td>
<td>Turns OFF outputs after being ON for certain time. The time is set by P command.</td>
</tr>
<tr>
<td>$B#\ K\ \ R\</td>
<td>&gt;</td>
<td>O</td>
<td>When setting error has occurred, turn plural bits of setting output port ON.</td>
</tr>
<tr>
<td>$B#\ K\ \ [Less\ than\ 5\ digits]\ \ R\</td>
<td>&gt;</td>
<td>O</td>
<td>When setting error has occurred, turn one bit of setting output port ON.</td>
</tr>
<tr>
<td>$B#\ K\ \ 0\ \ R\</td>
<td>&gt;</td>
<td>O</td>
<td>Resets the command “K” function.</td>
</tr>
<tr>
<td>$B#\ K\ \ C\ \ R\</td>
<td>&gt;</td>
<td>O</td>
<td>Turns setting output port bit OFF</td>
</tr>
</tbody>
</table>

#### Position data setting

<table>
<thead>
<tr>
<th>Command</th>
<th>Reply</th>
<th>EP</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B#\ 2\ \ [Less\ than\ 8\ digits]\ \ R\</td>
<td>&gt;</td>
<td>O</td>
<td>Sets motion counter register for command “3”, “4”, “5”.</td>
</tr>
<tr>
<td>$B#\ 2\ \ R\</td>
<td>&gt;</td>
<td>O</td>
<td>Sets the current position as motion counter register for command “3”, “4”, “5”.</td>
</tr>
<tr>
<td>$B#\ [3\ digits]\ \ L\</td>
<td>&gt;</td>
<td>O</td>
<td>Sets motion counter register for command “B”. (Max. 1,000 points)</td>
</tr>
<tr>
<td>$B#\ [3\ digits]\ \ R\</td>
<td>&gt;</td>
<td>O</td>
<td>Sets the current position as motion counter register for command “B”.</td>
</tr>
<tr>
<td>$B#\ S\ L\ \ R\</td>
<td>&gt;</td>
<td>O</td>
<td>Enables/disables the software limit function for motor 1 and motor 2 severally.</td>
</tr>
<tr>
<td>$B#\ S\ L\ \ [Less\ than\ 8\ digits]\ \ R\</td>
<td>&gt;</td>
<td>O</td>
<td>Sets the data memory position of the software limit for CW direction and CCW direction</td>
</tr>
</tbody>
</table>
### Common General Command (Part 4)

<table>
<thead>
<tr>
<th>Contents</th>
<th>Command</th>
<th>Reply</th>
<th>EP</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absolute movement</td>
<td>$ B# 3</td>
<td>&gt;</td>
<td>O</td>
<td>The motor moves to absolute position.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The position data was set earlier by command “2”.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td>O</td>
<td>Motor 1 moves to absolute position.</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td>O</td>
<td>Motor 2 moves to absolute position.</td>
</tr>
<tr>
<td></td>
<td>B [3 digits]</td>
<td>&gt;</td>
<td>O</td>
<td>The motor moves to absolute position.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The position data was set earlier by command “A”.</td>
</tr>
<tr>
<td></td>
<td>B [3 digits]</td>
<td>1</td>
<td>O</td>
<td>Motor 1 moves to absolute position.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The position data was set earlier by command “A”.</td>
</tr>
<tr>
<td></td>
<td>B [3 digits]</td>
<td>2</td>
<td>O</td>
<td>Motor 2 moves to absolute position.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The position data was set earlier by command “A”.</td>
</tr>
<tr>
<td></td>
<td>B #</td>
<td>&gt;</td>
<td>O</td>
<td>The motor moves the load to the next absolute position.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The position data was set earlier by command “A”.</td>
</tr>
<tr>
<td></td>
<td>B #</td>
<td>&gt;</td>
<td>O</td>
<td>Motor 1 moves the load to the next absolute position.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The position data was set earlier by command “A”.</td>
</tr>
<tr>
<td></td>
<td>B #</td>
<td>&gt;</td>
<td>O</td>
<td>Motor 2 moves the load to the next absolute position.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The position data was set earlier by command “A”.</td>
</tr>
</tbody>
</table>
### Incremental Movement

<table>
<thead>
<tr>
<th>Contents</th>
<th>Command</th>
<th>Reply</th>
<th>EP</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ B# 4</td>
<td></td>
<td></td>
<td></td>
<td>The motor moves to CW Incremental position from current position. The position data was set earlier by command &quot;2&quot;.</td>
</tr>
<tr>
<td>$ B# 4 1</td>
<td></td>
<td></td>
<td></td>
<td>Motor 1 moves to CW Incremental position from current position. The position data was set earlier by command &quot;2&quot;.</td>
</tr>
<tr>
<td>$ B# 4 2</td>
<td></td>
<td></td>
<td></td>
<td>Motor 2 moves to CW Incremental position from current position. The position data was set earlier by command &quot;2&quot;.</td>
</tr>
<tr>
<td>$ B# 5</td>
<td></td>
<td></td>
<td></td>
<td>The motor moves to CCW Incremental position from current position. The position data was set earlier by command &quot;2&quot;.</td>
</tr>
<tr>
<td>$ B# 5 1</td>
<td></td>
<td></td>
<td></td>
<td>Motor 1 moves to CCW Incremental position from current position. The position data was set earlier by command &quot;2&quot;.</td>
</tr>
<tr>
<td>$ B# 5 2</td>
<td></td>
<td></td>
<td></td>
<td>Motor 2 moves to CCW Incremental position from current position. The position data was set earlier by command &quot;2&quot;.</td>
</tr>
<tr>
<td>$ B# B [3 digits]</td>
<td></td>
<td></td>
<td></td>
<td>The motor moves to CW (in case of +) or CCW (in case of -) Incremental position from current position. The position data was set earlier by command &quot;A&quot;.</td>
</tr>
<tr>
<td>$ B# B [3 digits]</td>
<td>1 + or -</td>
<td></td>
<td></td>
<td>Motor 1 moves to CW (in case of +) or CCW (in case of -) Incremental position from current position. The position data was set earlier by command &quot;A&quot;.</td>
</tr>
<tr>
<td>$ B# B [3 digits]</td>
<td>2 + or -</td>
<td></td>
<td></td>
<td>Motor 2 moves to CW (in case of +) or CCW (in case of -) Incremental position from current position. The position data was set earlier by command &quot;A&quot;.</td>
</tr>
<tr>
<td>$ B# B + or -</td>
<td></td>
<td></td>
<td></td>
<td>The motor moves to CW (in case of +) or CCW (in case of -) Incremental position from current position. The position data was read out the position data set by command &quot;A&quot; in turn. The controller automatically increments the position number data by adding 001 to the previous position number.</td>
</tr>
<tr>
<td>$ B# B 1 + or -</td>
<td></td>
<td></td>
<td></td>
<td>Motor 1 moves to CW (in case of +) or CCW (in case of -) Incremental position from current position. The position data was read out the position data set by command &quot;A&quot; in turn. The controller automatically increments the position number data by adding 001 to the previous position number.</td>
</tr>
<tr>
<td>$ B# B 2 + or -</td>
<td></td>
<td></td>
<td></td>
<td>Motor 2 moves to CW (in case of +) or CCW (in case of -) Incremental position from current position. The position data was read out the position data set by command &quot;A&quot; in turn. The controller automatically increments the position number data by adding 001 to the previous position number.</td>
</tr>
</tbody>
</table>
## Common General Command (Part 6)

<table>
<thead>
<tr>
<th>Contents</th>
<th>Command</th>
<th>Reply</th>
<th>EP</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Direction movement</strong></td>
<td>$B# 7$</td>
<td>&gt;</td>
<td>○</td>
<td>Move motor in CW direction at Low speed. The motor doesn’t stop until command “S” or “SS” is sent or the limit sensor is turned ON.</td>
</tr>
<tr>
<td></td>
<td>$B# 1$</td>
<td>&gt;</td>
<td>○</td>
<td>Move motor1 in CW direction at Low speed.</td>
</tr>
<tr>
<td></td>
<td>$B# 2$</td>
<td>&gt;</td>
<td>○</td>
<td>Move motor2 in CW direction at Low speed.</td>
</tr>
<tr>
<td></td>
<td>$B# *$</td>
<td>&gt;</td>
<td>○</td>
<td>Single step motor in CW direction.</td>
</tr>
<tr>
<td></td>
<td>$B# 8$</td>
<td>&gt;</td>
<td>○</td>
<td>Move motor in CCW direction at Low speed. The motor doesn’t stop until command “S” or “SS” is sent or the limit sensor is turned ON.</td>
</tr>
<tr>
<td></td>
<td>$B# 1$</td>
<td>&gt;</td>
<td>○</td>
<td>Move motor1 in CCW direction at Low speed.</td>
</tr>
<tr>
<td></td>
<td>$B# 2$</td>
<td>&gt;</td>
<td>○</td>
<td>Move motor2 in CCW direction at Low speed.</td>
</tr>
<tr>
<td></td>
<td>$B# *$</td>
<td>&gt;</td>
<td>○</td>
<td>Single step motor in CCW direction.</td>
</tr>
<tr>
<td><strong>Speed adjustment</strong></td>
<td>$B# H$</td>
<td>&gt;</td>
<td>○</td>
<td>Change rotating motor speed to High speed using command “7”, “8”.</td>
</tr>
<tr>
<td></td>
<td>$B# L$</td>
<td>&gt;</td>
<td>○</td>
<td>Change rotating motor speed to Low speed using command “7”, “8”.</td>
</tr>
<tr>
<td></td>
<td>$B# N$</td>
<td>&gt;</td>
<td>○</td>
<td>Set the data to change moving motor speed on the way.</td>
</tr>
<tr>
<td></td>
<td>$B# N C$</td>
<td>&gt;</td>
<td>○</td>
<td>Clear the data set by command “N”.</td>
</tr>
<tr>
<td></td>
<td>$B# N S$</td>
<td>&gt;</td>
<td>○</td>
<td>Use the function to change moving motor speed on the way on the basis of the data set by command “N”. After that if execute command “B”, etc. and let the motor move, the speed is changed as the set data.</td>
</tr>
<tr>
<td></td>
<td>$B# N R$</td>
<td>&gt;</td>
<td>○</td>
<td>Disable the function of command “NS”.</td>
</tr>
<tr>
<td></td>
<td>$B# O$</td>
<td>&gt;</td>
<td>○</td>
<td>Set the S-curve data.</td>
</tr>
<tr>
<td></td>
<td>$B# O H$</td>
<td>&gt;</td>
<td>○</td>
<td>Set the speed at High speed.</td>
</tr>
<tr>
<td></td>
<td>$B# O H I$</td>
<td>&gt;</td>
<td>○</td>
<td>Increment the high speed by a certain value.</td>
</tr>
<tr>
<td></td>
<td>$B# O H D$</td>
<td>&gt;</td>
<td>○</td>
<td>Decrement the high speed by a certain value.</td>
</tr>
</tbody>
</table>
### Common General Command (Part 7)

<table>
<thead>
<tr>
<th>Contents</th>
<th>Command</th>
<th>Reply</th>
<th>EP</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Speed adjustment</strong></td>
<td>$ B# O L [Less than 5 digits] &lt;br&gt; $ B# O L I [Less than 5 digits] &lt;br&gt; $ B# O L D [Less than 5 digits] &lt;br&gt; $ B# O S [Less than 5 digits] &lt;br&gt; $ B# O S A [Less than 5 digits] &lt;br&gt; $ B# O S B [Less than 5 digits] &lt;br&gt; $ B# O S I [Less than 5 digits] &lt;br&gt; $ B# O S D [Less than 5 digits] &lt;br&gt; $ B# O X [Less than 5 digits]</td>
<td>&gt;</td>
<td>o</td>
<td>Set the speed at Low speed. &lt;br&gt; Increment the low speed by a certain value. &lt;br&gt; Decrement the low speed by a certain value. &lt;br&gt; Set the accel./decel. data. &lt;br&gt; Sets the acceleration data in acceleration. &lt;br&gt; Sets the acceleration data in deceleration. &lt;br&gt; Increment the accel./decel. data by a certain value. &lt;br&gt; Decrement the accel./decel. data by a certain value. &lt;br&gt; Set the multiplication factor data.</td>
</tr>
<tr>
<td><strong>Related matter to EEPROM</strong></td>
<td>$ B# A W</td>
<td>&gt;</td>
<td></td>
<td>Store the position data set by command “A” into EEPROM. &lt;br&gt; When storing has done, return the response of the format “$ B# * [③]”.</td>
</tr>
<tr>
<td></td>
<td>$ B# A L</td>
<td>&gt;</td>
<td>o</td>
<td>Read out the position data set by command “A” stored into EEPROM to RAM.</td>
</tr>
<tr>
<td></td>
<td>$ B# A L *</td>
<td>&gt;</td>
<td>o</td>
<td>Initializes the current position data of command “A” and “MA” stored into RAM.</td>
</tr>
<tr>
<td></td>
<td>$ B# D W</td>
<td>&gt;</td>
<td></td>
<td>Store the data to set speed, function, etc. into EEPROM. &lt;br&gt; When storing has done, return the response of the format “$ B# * [③]”.</td>
</tr>
<tr>
<td></td>
<td>$ B# D L</td>
<td>&gt;</td>
<td>o</td>
<td>Read out the data stored in EEPROM by command “DW” to RAM.</td>
</tr>
<tr>
<td></td>
<td>$ B# D L *</td>
<td>&gt;</td>
<td>o</td>
<td>Initializes the setting values of speed and functions written in RAM.</td>
</tr>
<tr>
<td></td>
<td>$ B# E E / / /</td>
<td>&gt;</td>
<td></td>
<td>Erase the data stored in EEPROM. &lt;br&gt; It takes about 5 seconds to erase. &lt;br&gt; When storing has done, return the response of the format “$ B# * [③]”.</td>
</tr>
<tr>
<td></td>
<td>$ B# I W</td>
<td>&gt;</td>
<td></td>
<td>Store the user program stored in RAM into EEPROM. &lt;br&gt; When storing has done, return the response of the format “$ B# * [③]”.</td>
</tr>
<tr>
<td></td>
<td>$ B# I L</td>
<td>&gt;</td>
<td>o</td>
<td>Read out the user program stored in EEPROM to RAM.</td>
</tr>
<tr>
<td><strong>Related matter to Stall</strong></td>
<td>$ B# Q S □ □ □</td>
<td>&gt;</td>
<td>o</td>
<td>Enable or disable stall check of each motor 1 and 2.</td>
</tr>
<tr>
<td></td>
<td>$ B# Q S R</td>
<td>&gt;</td>
<td>o</td>
<td>Disable stall check of both motor 1 and 2.</td>
</tr>
<tr>
<td>Contents</td>
<td>Command</td>
<td>Reply</td>
<td>EP</td>
<td>Descriptions</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------</td>
<td>-------</td>
<td>----</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Related to User program control</td>
<td>S B# G □</td>
<td>&gt;</td>
<td></td>
<td>Execute the user program stored by command “1”.</td>
</tr>
<tr>
<td></td>
<td>S B# G □</td>
<td>&gt;</td>
<td></td>
<td>Start the program from the part of format “G□”.</td>
</tr>
<tr>
<td></td>
<td>S B# G A S □</td>
<td>&gt;</td>
<td></td>
<td>Enable autostart. After this when turning On the power of RC-234, execute the user program automatically.</td>
</tr>
<tr>
<td></td>
<td>S B# G A R □</td>
<td>&gt;</td>
<td></td>
<td>Disable autostart.</td>
</tr>
<tr>
<td></td>
<td>S B# G C □</td>
<td>&gt;</td>
<td></td>
<td>Restart the user program after temporary stoppage.</td>
</tr>
<tr>
<td></td>
<td>S B# G E □</td>
<td>&gt;</td>
<td></td>
<td>End the user program.</td>
</tr>
<tr>
<td></td>
<td>S B# G E S □</td>
<td>&gt;</td>
<td></td>
<td>End the user program immediately.</td>
</tr>
<tr>
<td></td>
<td>S B# G S □</td>
<td>&gt;</td>
<td></td>
<td>Stop the user program temporarily.</td>
</tr>
<tr>
<td></td>
<td>S B# G W □  [Less than 5 digits] □</td>
<td>&gt;</td>
<td></td>
<td>Queries the cause of error of user program.</td>
</tr>
<tr>
<td></td>
<td>S B# G W □  [2 digits] □</td>
<td>&gt;</td>
<td></td>
<td>Change the buffer data.</td>
</tr>
<tr>
<td></td>
<td>S B# G W X □  [Less than 5 digits] □</td>
<td>&gt;</td>
<td></td>
<td>Changes the counter number in loop counter for the position number used in the user program.</td>
</tr>
<tr>
<td></td>
<td>S B# G W W □  [± Less than 9 digits] □</td>
<td>&gt;</td>
<td></td>
<td>Changes the 4-byte buffer data used in the user program.</td>
</tr>
<tr>
<td></td>
<td>S B# I □</td>
<td>&gt;</td>
<td></td>
<td>Store the user program into RC-234’s RAM. Just after executing this command, the user program is stored by the format “[USERPROGRAM] □”.</td>
</tr>
<tr>
<td>Others</td>
<td>S B# G W T  [Less than 5 digits] □</td>
<td>&gt;</td>
<td></td>
<td>Change the timer.</td>
</tr>
<tr>
<td></td>
<td>S B# G W T □  [Less than 5 digits] □</td>
<td>&gt;</td>
<td></td>
<td>Specifies the number of timer and change the timer.</td>
</tr>
<tr>
<td></td>
<td>S B# T *  [Less than 5 digits] □</td>
<td>&gt;</td>
<td></td>
<td>Set the timer and count the time.</td>
</tr>
<tr>
<td></td>
<td>S B# Q B  [Less than 5 digits] □  [Less than 5 digits] □</td>
<td>&gt;</td>
<td></td>
<td>Enables a “vibration suppression” function.</td>
</tr>
<tr>
<td></td>
<td>S B# Q B R □  [Less than 4 digits] □</td>
<td>&gt;</td>
<td></td>
<td>Enables a backlash offset/slip offset.</td>
</tr>
<tr>
<td></td>
<td>S B# Q B R 0 □  [Less than 5 digits] □</td>
<td>&gt;</td>
<td></td>
<td>Disable a backlash offset/slip offset.</td>
</tr>
<tr>
<td></td>
<td>S B# Q B R O L  [Less than 5 digits] □</td>
<td>&gt;</td>
<td></td>
<td>Sets the speed data when the offset is enabled.</td>
</tr>
</tbody>
</table>
### Inquiry command and its response (Part 1)

<table>
<thead>
<tr>
<th>Contents</th>
<th>Command</th>
<th>Reply</th>
<th>No. of Character</th>
<th>EP</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request for the condition</td>
<td>$ B# $</td>
<td>$ B# $</td>
<td>5</td>
<td>5</td>
<td>Queries the existence of an error and whether the motor has stopped or not.</td>
</tr>
<tr>
<td></td>
<td>$ B# $ 9</td>
<td>$ B# $</td>
<td>6</td>
<td>6</td>
<td>Queries the condition data (the existence of an error).</td>
</tr>
<tr>
<td></td>
<td>$ B# $ 9</td>
<td>$ B# $</td>
<td>5</td>
<td>5</td>
<td>Set the data bit of the condition data and inquire it.</td>
</tr>
<tr>
<td></td>
<td>$ B# $ 9 A L L</td>
<td>$ B# $</td>
<td>16</td>
<td>16</td>
<td>Queries the status of status flags, condition flags, general inputs/outputs and sensors all together.</td>
</tr>
<tr>
<td></td>
<td>$ B# $ 9 D D</td>
<td>$ B# $</td>
<td>6</td>
<td>6</td>
<td>Queries the status and the cause of error for each motor.</td>
</tr>
<tr>
<td>Related matter to EEPROM</td>
<td>$ B# $ E E C H E C K</td>
<td>$ B# $ D A T A _ O K !</td>
<td>11</td>
<td>11</td>
<td>Checks stored data when some data is stored in EEPROM by the command &quot;DW&quot;, &quot;AW&quot;, and &quot;IW&quot;.</td>
</tr>
<tr>
<td>Request for the mode setting data</td>
<td>$ B# $ E</td>
<td>$ B# $</td>
<td>5</td>
<td>5</td>
<td>Queries the setting of command &quot;EM&quot;.</td>
</tr>
<tr>
<td></td>
<td>$ B# $ E A</td>
<td>$ B# $</td>
<td>6</td>
<td>6</td>
<td>Queries the setting of command &quot;EM&quot;.</td>
</tr>
<tr>
<td></td>
<td>$ B# $ E D</td>
<td>$ B# $</td>
<td>5</td>
<td>5</td>
<td>Queries the setting of command &quot;EM&quot;.</td>
</tr>
<tr>
<td></td>
<td>$ B# $ E E</td>
<td>$ B# $ 0 $ B# $ 1</td>
<td>5</td>
<td>5</td>
<td>If echo function is OFF, the upper 5 character response is returned and if ON, the lower 9 character response is returned.</td>
</tr>
<tr>
<td></td>
<td>$ B# $ E L</td>
<td>$ B# $</td>
<td>5</td>
<td>5</td>
<td>Queries whether the command &quot;EL&quot; is set or not.</td>
</tr>
<tr>
<td></td>
<td>$ B# $ E M</td>
<td>$ B# $</td>
<td>5</td>
<td>5</td>
<td>Queries the setting of command &quot;EM&quot;.</td>
</tr>
<tr>
<td></td>
<td>$ B# $ E R</td>
<td>$ B# $</td>
<td>5</td>
<td>5</td>
<td>Queries whether &quot;command error display function&quot; is set or not.</td>
</tr>
<tr>
<td></td>
<td>$ B# $ E S</td>
<td>$ B# $</td>
<td>5</td>
<td>5</td>
<td>Queries the current baud rate.</td>
</tr>
<tr>
<td></td>
<td>$ B# $ F</td>
<td>$ B# $</td>
<td>5</td>
<td>5</td>
<td>Queries which is the object to control, motor 1 or motor 2.</td>
</tr>
<tr>
<td></td>
<td>$ B# $ S U M</td>
<td>$ B# $ 0 $ B# $ 1</td>
<td>5</td>
<td>5</td>
<td>If sum check function is OFF, the upper 5 character response is returned and if ON, the lower 9 character response is returned.</td>
</tr>
<tr>
<td>Related matter to Home seach</td>
<td>$ B# $ 0 $ S</td>
<td>$ B# $ [5 digits]</td>
<td>9</td>
<td>9</td>
<td>Queries the data set by command &quot; $ S&quot;.</td>
</tr>
<tr>
<td></td>
<td>$ B# $ 0 $ B</td>
<td>$ B# $</td>
<td>5</td>
<td>5</td>
<td>Queries the multiplication factor of command &quot; $ S&quot;.</td>
</tr>
<tr>
<td></td>
<td>$ B# $ 0 $ Z D</td>
<td>$ B# $</td>
<td>5</td>
<td>5</td>
<td>Queries the Z-ph ON number set by command &quot; $ Z&quot;.</td>
</tr>
<tr>
<td>Command</td>
<td>Contents</td>
<td>No. of Character</td>
<td>Descriptions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td>-----------------</td>
<td>--------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$ $2 $</td>
<td>Request for the position data</td>
<td>8 digits</td>
<td>Queries the current setting value of command &quot;$2&quot;.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$ $6 $</td>
<td>Request for the Input/Output ports</td>
<td>8 digits</td>
<td>Queries the current position of command &quot;$6&quot;.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$ $12 $</td>
<td></td>
<td></td>
<td>Queries the position number of command &quot;A&quot; to refer next by command &quot;$12&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$ $2 $</td>
<td>Request for command &quot;$2&quot;</td>
<td>8 digits</td>
<td>Queries the current position of command &quot;$2&quot;.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$ $6 $</td>
<td>Request for command &quot;$6&quot;</td>
<td>8 digits</td>
<td>Queries the status of ORG and limit sensor input of motor 1 all together.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$ $12 $</td>
<td></td>
<td>8 digits</td>
<td>Queries the status of ORG and limit sensor input of motor 2 all together.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 15: Control Command List**

- **Command**: 2, D, S
- **No. of Character**: 8 digits
- **Descriptions**:
  - Queries the current setting value of command "$2".
  - Selects a motor 1 or 2 and inquires the current position.
  - Queries the current position of command "$6".
  - Queries the position number of command "A" to refer next by command "$12".
  - Queries the current position of command "$2".
  - Queries the status of ORG and limit sensor input of motor 1 all together.
  - Queries the status of ORG and limit sensor input of motor 2 all together.
  - Queries the status of ORG or limit sensor input bits.
  - Queries the status of ORG or limit sensor input bits.
  - Queries the status of output ports all together.
  - Queries the data set by command "K".
  - Queries the number of times of interruption.
  - Queries the data set by command "K".
  - Queries the current S-curve data.
  - Queries the current speed at High speed.
  - Queries the current speed at Low speed.

- **Reply**: 8 digits, 5 digits
- **Descriptions**: (continued)
<table>
<thead>
<tr>
<th>Contents</th>
<th>Command</th>
<th>Reply</th>
<th>No. of Character</th>
<th>EP</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed adjustment</td>
<td>$B#O$</td>
<td>$O$</td>
<td>[5 digits]</td>
<td>9</td>
<td>Queries the current accel./decel. data.</td>
</tr>
<tr>
<td></td>
<td>$B#O$</td>
<td>$S$</td>
<td>[5 digits]</td>
<td>9</td>
<td>Queries acceleration data in acceleration motion.</td>
</tr>
<tr>
<td></td>
<td>$B#O$</td>
<td>$S$</td>
<td>[5 digits]</td>
<td>9</td>
<td>Queries acceleration data in deceleration motion.</td>
</tr>
<tr>
<td>Related matter to stall</td>
<td>$B#Q$</td>
<td>$O$</td>
<td>[5 digits]</td>
<td>5</td>
<td>Queries whether the stall has been detected or not.</td>
</tr>
<tr>
<td></td>
<td>$B#Q$</td>
<td>$S$</td>
<td>[5 digits]</td>
<td>6</td>
<td>Queries whether the stall detection has been executed or not.</td>
</tr>
<tr>
<td>Related matter to the user program control</td>
<td>$B#G$</td>
<td>$A$</td>
<td>[5 digits]</td>
<td>5</td>
<td>Queries the position to execute the command next when the user program is running.</td>
</tr>
<tr>
<td></td>
<td>$B#G$</td>
<td>$N$</td>
<td>[5 digits]</td>
<td>9</td>
<td>Queries the next position where the error has occurred when the user program is ended because of error.</td>
</tr>
<tr>
<td></td>
<td>$B#G$</td>
<td>$R$</td>
<td>[5 digits]</td>
<td>9</td>
<td>Queries the current counter number in loop counter used in the user program.</td>
</tr>
<tr>
<td></td>
<td>$B#G$</td>
<td>$R$</td>
<td>[2 digits]</td>
<td>6</td>
<td>Queries the contents of data buffer used in the user program.</td>
</tr>
<tr>
<td></td>
<td>$B#G$</td>
<td>$W$</td>
<td>[±9 digits]</td>
<td>14</td>
<td>Queries the current status of 3-byte data buffer used in the user program.</td>
</tr>
<tr>
<td></td>
<td>$B#G$</td>
<td>$T$</td>
<td>[5 digits]</td>
<td>9</td>
<td>Queries the current time left.</td>
</tr>
<tr>
<td></td>
<td>$B#G$</td>
<td>$T$</td>
<td>[5 digits]</td>
<td>9</td>
<td>Specificed the number of timer and queries the current time left.</td>
</tr>
<tr>
<td></td>
<td>$B#G$</td>
<td>$S$</td>
<td>[5 digits]</td>
<td>5</td>
<td>Queries the status of user program. (The user program is running, ended because of error, etc.)</td>
</tr>
<tr>
<td></td>
<td>$B#G$</td>
<td>$S$</td>
<td>[5 digits]</td>
<td>5</td>
<td>Queries the cause of stop in the user program.</td>
</tr>
<tr>
<td></td>
<td>$B#I$</td>
<td>$R$</td>
<td>[User program]</td>
<td></td>
<td>Queries the contents of user program stored into RAM. The number of returned character is changed depending on the quantity of stored program.</td>
</tr>
<tr>
<td>Others</td>
<td>$B#Q$</td>
<td>$B$</td>
<td>[5 digits]</td>
<td>15</td>
<td>Queries the output timing of additional pulses.</td>
</tr>
<tr>
<td></td>
<td>$B#Q$</td>
<td>$R$</td>
<td>[5 digits]</td>
<td>11</td>
<td>Queries the status of the backlash/slip offset setting.</td>
</tr>
<tr>
<td></td>
<td>$B#Q$</td>
<td>$R$</td>
<td>[5 digits]</td>
<td>9</td>
<td>Queries the speed when offsetting.</td>
</tr>
<tr>
<td></td>
<td>$B#T$</td>
<td>[Less than 5 digits]</td>
<td>[5 digits]</td>
<td>1(4)</td>
<td>Set the time of the timer of RC-234 and use it. If timer has timed out, return the response of the format &quot;$B#T$&quot;.</td>
</tr>
<tr>
<td></td>
<td>$B#T$</td>
<td>$B$</td>
<td>[5 digits]</td>
<td>5</td>
<td>Queries whether timer has timed out or not.</td>
</tr>
<tr>
<td></td>
<td>$B#V$</td>
<td>$B$</td>
<td>$RC-234$ Ver ...</td>
<td>3 9</td>
<td>Queries the version of RC-234.</td>
</tr>
</tbody>
</table>
### 15. Control Command List

<table>
<thead>
<tr>
<th>Command</th>
<th>Mode setting</th>
<th>Stall detection</th>
<th>Mode 0</th>
<th>Mode 1</th>
<th>Mode 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>E P 0</td>
<td>E P 0 0</td>
<td>E P 0 0</td>
<td>E P 0 0</td>
<td>E P 0 0</td>
<td>E P 0 0</td>
</tr>
<tr>
<td>E P 0</td>
<td>E P 0 0</td>
<td>E P 0 0</td>
<td>E P 0 0</td>
<td>E P 0 0</td>
<td>E P 0 0</td>
</tr>
<tr>
<td>Encoder</td>
<td>Encoder</td>
<td>Encoder</td>
<td>Encoder</td>
<td>Encoder</td>
<td>Encoder</td>
</tr>
<tr>
<td>Mode 0</td>
<td>Mode 0</td>
<td>Mode 0</td>
<td>Mode 0</td>
<td>Mode 0</td>
<td>Mode 0</td>
</tr>
<tr>
<td>Mode 1</td>
<td>Mode 1</td>
<td>Mode 1</td>
<td>Mode 1</td>
<td>Mode 1</td>
<td>Mode 1</td>
</tr>
<tr>
<td>Mode 2</td>
<td>Mode 2</td>
<td>Mode 2</td>
<td>Mode 2</td>
<td>Mode 2</td>
<td>Mode 2</td>
</tr>
</tbody>
</table>

**Descriptions**

- Set the counting to -9,388,608 to +9,388,607.
- Set the counting to 0 to 16,777,215. Can set stall data from 100 to 65,535.
- Position the stall sensor at the middle of stall slit.
- Set the input should be ON. Enable stall checking using encoder in mode “E0”.
- Cancel stall checking using encoder.
- Enable or disable automatic stall adjustment feature for both motor1 and motor 2 respectively. If enable automatic stall adjust, execute stall checking and after that adjust to the object value of encoder.
- If enable automatic stall adjustment, set the delay time after which the automatic stall adjustment is carried out.
- During automatic stall adjust, when and how many pulses (max.) the limit error is set.
- Set the speed during automatic stall adjustment.
- Set stall data. Usually, when stall is detected, a command except *“0”* (home search) isn’t received, but this status is reset.
- Reset stall error. When executing home search command, search the home position by new algorithm.
- When executing home search command, search the home position by old algorithm.
## Control Command List

### Queries command only for modes and its response

<table>
<thead>
<tr>
<th>Contents</th>
<th>Command</th>
<th>Reply</th>
<th>No. of Character</th>
<th>EP</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode setting</td>
<td>Mode 0, 2</td>
<td>$B# E P$</td>
<td>&gt; $B# [\ ] [\ ]</td>
<td>5</td>
<td>Queries bi-direction counting(-8,388,608 to +8,388,607) or uni-direction counting(0 to 16,777,215).</td>
</tr>
<tr>
<td>Stall detection</td>
<td>Mode 0, 2</td>
<td>$B# 0 Q D$</td>
<td>&gt; $B# [\ ] [\ ]</td>
<td>9</td>
<td>Queries the setting value of command “0Q”.</td>
</tr>
<tr>
<td></td>
<td>Mode 0, 2</td>
<td>$B# 0 Q W$</td>
<td>&gt; $B# [5 digits] [\ ]</td>
<td>9</td>
<td>Queries the actually counted pulse number of ON term when positioning the stall sensor at the middle of stall slit.</td>
</tr>
<tr>
<td></td>
<td>Mode 0, 2</td>
<td>$B# Q D$</td>
<td>&gt; $B# [5 digits] [\ ]</td>
<td>9</td>
<td>Queries the pulse number of stall slit ON/OFF cycle.</td>
</tr>
<tr>
<td></td>
<td>Mode 0, 2</td>
<td>$B# Q D 1$</td>
<td>&gt; $B# [5 digits] [\ ]</td>
<td>9</td>
<td>Queries the pulse number of stall slit ON/OFF cycle to Motor1.</td>
</tr>
<tr>
<td></td>
<td>Mode 0, 2</td>
<td>$B# Q D 2$</td>
<td>&gt; $B# [5 digits] [\ ]</td>
<td>9</td>
<td>Queries the pulse number of stall slit ON/OFF cycle to Motor2.</td>
</tr>
<tr>
<td>Mode 0</td>
<td></td>
<td>$B# Q S E$</td>
<td>&gt; $B# [\ ] [\ ] [\ ]</td>
<td>6</td>
<td>Enable stall checking using encoder in mode E0.</td>
</tr>
<tr>
<td>Mode 1</td>
<td></td>
<td>$B# Q J$</td>
<td>&gt; $B# [\ ] [\ ] [\ ]</td>
<td>6</td>
<td>Queries enable or disable automatic stall adjust feature for both motor1 and motor2.</td>
</tr>
<tr>
<td>Encoder</td>
<td>Mode 1</td>
<td>$B# Q J T$</td>
<td>&gt; $B# [3 digits] [\ ]</td>
<td>7</td>
<td>Queries the time set by the command “QJT”.</td>
</tr>
<tr>
<td>Others in</td>
<td>Mode 2</td>
<td>$B# Q J A$</td>
<td>&gt; $B# [5 digits] [\ ]</td>
<td>9</td>
<td>Queries the pulse set by the command “QJA”.</td>
</tr>
<tr>
<td>Others in</td>
<td>Mode 2</td>
<td>$B# Q J O$</td>
<td>&gt; $B# [5 digits] [\ ]</td>
<td>9</td>
<td>Queries the speed set by the command “QJO”.</td>
</tr>
<tr>
<td>Others in</td>
<td>Mode 2</td>
<td>$B# Q E$</td>
<td>&gt; $B# [\ ] [\ ] [\ ]</td>
<td>5</td>
<td>Queries the setting value of the difference between the actual pulse and encoder pulse.</td>
</tr>
<tr>
<td>Mode 1</td>
<td></td>
<td>$B# P A$</td>
<td>&gt; $B# [\ ] [\ ]</td>
<td>5</td>
<td>Queries the mode of counting. (Uni., Bi., quadrature counting.)</td>
</tr>
<tr>
<td>Others in</td>
<td>Mode 2</td>
<td>$B# P B$</td>
<td>&gt; $B# [3 digits] [\ ]</td>
<td>7</td>
<td>Queries the setting value of command “PB”.</td>
</tr>
<tr>
<td>Mode 0, 1</td>
<td></td>
<td>$B# C A$</td>
<td>&gt; $B# [\ ] [\ ] [\ ]</td>
<td>7</td>
<td>Queries the status of INP (imposition) signal, ON or OFF.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$B# C O A$</td>
<td>&gt; $B# [\ ] [\ ] [\ ]</td>
<td>7</td>
<td>Queries the status of CLR terminal output and input ports.</td>
</tr>
</tbody>
</table>
### Control Command List

<table>
<thead>
<tr>
<th>Contents</th>
<th>Command</th>
<th>Reply</th>
<th>EP</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Home search related</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$ B# 0 3 [\pm 7\text{ digits}] $</td>
<td>&gt;</td>
<td>O</td>
<td></td>
<td>Performs Mechanical Home search #1 for motor 1 and motor 2 simultaneously.</td>
</tr>
<tr>
<td>$ B# 0 R 3 [\pm 7\text{ digits}] $</td>
<td>&gt;</td>
<td>O</td>
<td></td>
<td>Performs Mechanical Home search #2 for motor 1 and motor 2 simultaneously.</td>
</tr>
<tr>
<td>$ B# 0 H 3 [\pm 7\text{ digits}] $</td>
<td>&gt;</td>
<td>O</td>
<td></td>
<td>Performs Mechanical Home search #1 at high speed for motor 1 and motor 2 simultaneously.</td>
</tr>
<tr>
<td>$ B# 0 A 3 [\pm 7\text{ digits}] $</td>
<td>&gt;</td>
<td>O</td>
<td></td>
<td>Performs Mechanical Home search #2 at high speed for motor 1 and motor 2 simultaneously.</td>
</tr>
<tr>
<td>$ B# 1 3 [\pm 7\text{ digits}] $</td>
<td>&gt;</td>
<td>O</td>
<td></td>
<td>Moves motor 1 and motor 2 to the Home position simultaneously.</td>
</tr>
<tr>
<td><strong>Stop of motor</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$ B# S 1 [\pm 7\text{ digits}] $</td>
<td>&gt;</td>
<td>O</td>
<td></td>
<td>Immediately stops motor 1. (When the motor is running at an extremely high speed, it may result in a stall.)</td>
</tr>
<tr>
<td>$ B# S 2 [\pm 7\text{ digits}] $</td>
<td>&gt;</td>
<td>O</td>
<td></td>
<td>Immediately stops motor 2. (When the motor is running at an extremely high speed, it may result in a stall.)</td>
</tr>
<tr>
<td>$ B# S 3 [\pm 7\text{ digits}] $</td>
<td>&gt;</td>
<td>O</td>
<td></td>
<td>Immediately stops motor 1 and motor 2 simultaneously. (When the motor is running at an extremely high speed, it may result in a stall.)</td>
</tr>
<tr>
<td>$ B# S S 1 [\pm 7\text{ digits}] $</td>
<td>&gt;</td>
<td>O</td>
<td></td>
<td>Stops motor 1 with ramping deceleration.</td>
</tr>
<tr>
<td>$ B# S S 2 [\pm 7\text{ digits}] $</td>
<td>&gt;</td>
<td>O</td>
<td></td>
<td>Stops motor 2 with ramping deceleration.</td>
</tr>
<tr>
<td>$ B# S S S 3 [\pm 7\text{ digits}] $</td>
<td>&gt;</td>
<td>O</td>
<td></td>
<td>Stops motor 1 and motor 2 simultaneously with ramping deceleration.</td>
</tr>
<tr>
<td><strong>Position data setting</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$ B# M A [3\text{ digits}] [\pm 7\text{ digits}] [\pm 7\text{ digits}] [\pm 7\text{ digits}] $</td>
<td>&gt;</td>
<td>O</td>
<td></td>
<td>Sets the position pulse number which the command “MB”, etc. use.</td>
</tr>
<tr>
<td>$ B# M A [3\text{ digits}] [\pm 7\text{ digits}] [\pm 7\text{ digits}] $</td>
<td>&gt;</td>
<td>O</td>
<td></td>
<td>Sets the current position as position pulse number which the command “MB”, etc. use.</td>
</tr>
<tr>
<td>$ B# M P [3\text{ digits}] [\pm 7\text{ digits}] [\pm 7\text{ digits}] $</td>
<td>&gt;</td>
<td>O</td>
<td></td>
<td>Sets the interpolation pattern. (linear interpolation)</td>
</tr>
<tr>
<td>$ B# M P C [3\text{ digits}] [\pm 7\text{ digits}] [\pm 7\text{ digits}] $</td>
<td>&gt;</td>
<td>O</td>
<td></td>
<td>Sets the interpolation pattern. (circular interpolation)</td>
</tr>
<tr>
<td>$ B# M P C [3\text{ digits}] [\pm 7\text{ digits}] [\pm 7\text{ digits}] $</td>
<td>&gt;</td>
<td>O</td>
<td></td>
<td>Erases specified interpolation pattern.</td>
</tr>
<tr>
<td><strong>Absolute movement</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$ B# 3 3 [\pm 7\text{ digits}] $</td>
<td>&gt;</td>
<td>O</td>
<td></td>
<td>Moves motor 1 and motor 2 simultaneously to the position specified by the command “2”.</td>
</tr>
<tr>
<td>$ B# B [3\text{ digits}] [\pm 7\text{ digits}] [\pm 7\text{ digits}] $</td>
<td>&gt;</td>
<td>O</td>
<td></td>
<td>Moves motor 1 and motor 2 simultaneously to the position specified by the command “A”.</td>
</tr>
<tr>
<td>$ B# B 3 [\pm 7\text{ digits}] [\pm 7\text{ digits}] $</td>
<td>&gt;</td>
<td>O</td>
<td></td>
<td>Moves motor 1 and motor 2 simultaneously to the position specified by the command “B”.</td>
</tr>
<tr>
<td>$ B# M B [3\text{ digits}] [\pm 7\text{ digits}] [\pm 7\text{ digits}] $</td>
<td>&gt;</td>
<td>O</td>
<td></td>
<td>Moves motor 1 and motor 2 simultaneously to the position specified by the command “MA”.</td>
</tr>
<tr>
<td>$ B# M B [\pm 7\text{ digits}] [\pm 7\text{ digits}] $</td>
<td>&gt;</td>
<td>O</td>
<td></td>
<td>Reads the pp of subsequent pn of the last move pn by the command “MB” and moves motor to the position.</td>
</tr>
</tbody>
</table>
## Controls Command List

### Commands for EM (part 2)

<table>
<thead>
<tr>
<th>Contents</th>
<th>Command</th>
<th>Reply</th>
<th>EP</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absolute move</td>
<td>$ B# M C 1 [3 digits], [3 digits]</td>
<td>&gt;</td>
<td>○</td>
<td>Performs CW circular interpolation to the position specified by the command “MA”.</td>
</tr>
<tr>
<td></td>
<td>$ B# M C 2 [3 digits], [3 digits]</td>
<td>&gt;</td>
<td>○</td>
<td>Performs CCW circular interpolation to the position specified by the command “MA”.</td>
</tr>
<tr>
<td></td>
<td>$ B# M C 3 [3 digits], [3 digits]</td>
<td>&gt;</td>
<td>○</td>
<td>Performs CCW circular interpolation to the position specified by the command “MA” via intermediate positions.</td>
</tr>
<tr>
<td></td>
<td>$ B# M S [3 digits]</td>
<td>&gt;</td>
<td>○</td>
<td>Performs a linear interpolation to the position specified by the command “MA”.</td>
</tr>
<tr>
<td></td>
<td>$ B# M S</td>
<td>&gt;</td>
<td>○</td>
<td>Reads the pp of subsequent pn of the last move pn by the command “MS” and moves a motor to the position.</td>
</tr>
<tr>
<td>Incremental move</td>
<td>$ B# 4 3</td>
<td>&gt;</td>
<td>○</td>
<td>Incrementally moves motor 1 and motor 2 simultaneously by the distance set by the command “2” in the CW direction relative to the current position.</td>
</tr>
<tr>
<td></td>
<td>$ B# 5 3</td>
<td>&gt;</td>
<td>○</td>
<td>Incrementally moves motor 1 and motor 2 simultaneously by the distance set by the command “2” in the CCW direction relative to the current position.</td>
</tr>
<tr>
<td></td>
<td>$ B# B 3</td>
<td>&gt;</td>
<td>○</td>
<td>Reads the pp of subsequent pn of the last move pn by the command “B” and incrementally moves motor 1 and motor 2 simultaneously in the CW (in case of +) or CCW (in case of -) direction relative to the current position.</td>
</tr>
<tr>
<td></td>
<td>$ B# M B + [3 digits]</td>
<td>&gt;</td>
<td>○</td>
<td>Incrementally moves motor 1 and motor 2 simultaneously by the distance set by the command “MA” in the CW direction relative to the current position.</td>
</tr>
<tr>
<td></td>
<td>$ B# M B +</td>
<td>&gt;</td>
<td>○</td>
<td>Reads the pp of subsequent pn of the last move pn by the command “B+” and incrementally moves motor 1 and motor 2 simultaneously in the CW (in case of +) or CCW (in case of -) direction relative to the current position.</td>
</tr>
<tr>
<td></td>
<td>$ B# M C 1 + [3 digits], [3 digits]</td>
<td>&gt;</td>
<td>○</td>
<td>Performs CW circular interpolation by certain distance set by the command “MA” relative to current position. (incremental move)</td>
</tr>
<tr>
<td></td>
<td>$ B# M C 2 + [3 digits], [3 digits]</td>
<td>&gt;</td>
<td>○</td>
<td>Performs CCW circular interpolation by certain distance set by the command “MA” relative to current position. (incremental move)</td>
</tr>
<tr>
<td></td>
<td>$ B# M C 3 + [3 digits], [3 digits]</td>
<td>&gt;</td>
<td>○</td>
<td>Performs linear interpolation by certain distance set by the command “MA” via intermediate point relative to current position. (incremental move)</td>
</tr>
<tr>
<td></td>
<td>$ B# M S + [3 digits]</td>
<td>&gt;</td>
<td>○</td>
<td>Reads the position number of subsequent pn of the last move pn by the command “MS+” and moves by the position data.</td>
</tr>
<tr>
<td>Direction move</td>
<td>$ B# 7 3</td>
<td>&gt;</td>
<td>○</td>
<td>Rotates motor 1 and motor 2 simultaneously at low speed in the CW direction.</td>
</tr>
<tr>
<td></td>
<td>$ B# 8 3</td>
<td>&gt;</td>
<td>○</td>
<td>Rotates motor 1 and motor 2 simultaneously at low speed in the CCW direction.</td>
</tr>
<tr>
<td></td>
<td>$ B# M G</td>
<td>&gt;</td>
<td>○</td>
<td>Performs a continuous interpolation move set by the command “MP”.</td>
</tr>
<tr>
<td>Stall detection</td>
<td>$ B# Q S S</td>
<td>&gt;</td>
<td>○</td>
<td>Specifies the target axis when a stall is detected.</td>
</tr>
</tbody>
</table>
### Commands for EM Query command and its reply

<table>
<thead>
<tr>
<th>Contents</th>
<th>Command</th>
<th>Reply</th>
<th>No. of Character</th>
<th>EP</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query of the position</td>
<td>$B#\ M\ A [3 digits\ D\ \square\ ,\ ]$</td>
<td>$&gt;$ $B\ #\ \pm7\ digits\ ,\ \pm7\ digits\ \square\ $</td>
<td>2 1</td>
<td>○ Queries the position data which the command “MB”, etc. use.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$B#\ M\ B\ N\ \square$</td>
<td>$&gt;$ $B\ #\ \square [5\ digits]$</td>
<td>9</td>
<td>○ Queries the position number of command “MB” to move subsequently.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$B#\ M\ S\ N\ \square$</td>
<td>$&gt;$ $B\ #\ \square [5\ digits]$</td>
<td>9</td>
<td>○ Queries the position number of command “MS” to move subsequently.</td>
<td></td>
</tr>
<tr>
<td>Stall detection Mode 0, 2</td>
<td>$B#\ Q\ S\ S\ \square$</td>
<td>$&gt;$ $B\ #\ \square [3\ digits]\ ,\ [3\ digits]\ \square$</td>
<td>13 or 14</td>
<td>○ Specifies the target axis when a stall is detected.</td>
<td></td>
</tr>
</tbody>
</table>
### 16. Command Default Setting

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Default</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;C Y&quot;</td>
<td>S-Curve</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>&quot;C Y&quot;</td>
<td>High speed data</td>
<td>5000</td>
<td></td>
</tr>
<tr>
<td>&quot;C Y&quot;</td>
<td>Low speed data</td>
<td>500</td>
<td></td>
</tr>
<tr>
<td>&quot;C Y&quot;</td>
<td>Accel. data</td>
<td>300</td>
<td></td>
</tr>
<tr>
<td>&quot;C Y&quot;</td>
<td>Multiplication factor</td>
<td>300 $\times$ 1pps</td>
<td></td>
</tr>
<tr>
<td>&quot;C Y&quot;</td>
<td>Encoder counting mode</td>
<td>1</td>
<td>Single Use PA4 for quadrature counting</td>
</tr>
<tr>
<td>&quot;C Y&quot;</td>
<td>Encoder ratio</td>
<td>10</td>
<td>1 : 10</td>
</tr>
<tr>
<td>&quot;C Y&quot;</td>
<td>Circle of stall detection sensor ON and OFF</td>
<td>400</td>
<td>400 pulses</td>
</tr>
<tr>
<td>&quot;C Y&quot;</td>
<td>Vibration suppression function output setting</td>
<td>0,0</td>
<td>Disabled</td>
</tr>
<tr>
<td>&quot;C Y&quot;</td>
<td>Backlash offset</td>
<td>0</td>
<td>Disabled</td>
</tr>
<tr>
<td>&quot;C Y&quot;</td>
<td>Offset pulse number</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>&quot;C Y&quot;</td>
<td>Speed when offsetting</td>
<td>500</td>
<td></td>
</tr>
<tr>
<td>&quot;C Y&quot;</td>
<td>Stall error interval</td>
<td>4</td>
<td>±64 pulses</td>
</tr>
<tr>
<td>&quot;C Y&quot;</td>
<td>Stall adjust</td>
<td>0</td>
<td>No default (unused)</td>
</tr>
<tr>
<td>&quot;C Y&quot;</td>
<td>Stall adjust max. migration pulse</td>
<td>800</td>
<td>800 pulses</td>
</tr>
<tr>
<td>&quot;C Y&quot;</td>
<td>Speed in adjusting</td>
<td>0</td>
<td>Present low speed</td>
</tr>
<tr>
<td>&quot;C Y&quot;</td>
<td>Stall adjust time</td>
<td>3</td>
<td>30 msec</td>
</tr>
<tr>
<td>&quot;C Y&quot;</td>
<td>Stall detection of motor 1, 2</td>
<td>0</td>
<td>No stall detection</td>
</tr>
<tr>
<td>&quot;C Y&quot;</td>
<td>Target axis when a stall is detected.</td>
<td>0</td>
<td>Stops the axis detecting a stall</td>
</tr>
<tr>
<td>&quot;C Y&quot;</td>
<td>Stall detection using encoder</td>
<td>0</td>
<td>No default</td>
</tr>
<tr>
<td>&quot;C Y&quot;</td>
<td>Software limit function</td>
<td>0</td>
<td>Disabled</td>
</tr>
<tr>
<td>&quot;C Y&quot;</td>
<td>Data memory position range setting</td>
<td>0,0</td>
<td></td>
</tr>
<tr>
<td>&quot;C Y&quot;</td>
<td>Sum check function</td>
<td>0</td>
<td>No default (Unused)</td>
</tr>
</tbody>
</table>
〈Caution〉

- Some of setting commands in “Command Default Setting” will be written the changed setting value in EEPROM if you execute command “A W” and “D W” after changing setting value.

- When you execute command “A W” and “D W”, after that, command “A L” and “D L” is executed automatically every time you turn on the power. Therefore, in case of the command written the setting value in EEPROM, default in “Command Default Setting” is ignored and the value written in EEPROM will become default when you turn on the power supply.

- Some commands can set data of motor 1, 2 separately, but default of motor 1, 2 is the same. However, when you execute command “D W”, the setting each changed value of motor 1, 2 will become default.

- If you have executed command “A W” or “D W” before, you have to execute command “E E/ / /” and clear the data written in EEPROM to return to the original default.
### Appendix. Contents by Command Function

<table>
<thead>
<tr>
<th>Speed</th>
<th>Home search</th>
<th>Encoder</th>
<th>Sensor Input Logic</th>
<th>MotorControl Method</th>
<th>Communication</th>
<th>User Program</th>
<th>Stall Detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>0</td>
<td>Q J O</td>
<td>B- 145</td>
<td>GW</td>
<td>E</td>
<td>G</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>0 A</td>
<td>Q J R</td>
<td>B- 144</td>
<td>T</td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>0 B</td>
<td>Q J T</td>
<td>B- 144</td>
<td>R</td>
<td>E</td>
<td>G</td>
<td></td>
</tr>
<tr>
<td>N C</td>
<td>0 H</td>
<td>Q S</td>
<td>B- 147</td>
<td>G</td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>N R</td>
<td>0 R</td>
<td>Q S E</td>
<td>B- 149</td>
<td>G</td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>N S</td>
<td>0 S</td>
<td>Q S R</td>
<td>B- 147</td>
<td>E</td>
<td>E</td>
<td>G</td>
<td></td>
</tr>
<tr>
<td>O C</td>
<td>0 X</td>
<td>Q S S</td>
<td>B- 147</td>
<td>D</td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>O H</td>
<td>0 Z</td>
<td>R D</td>
<td>B- 151</td>
<td>E</td>
<td>E</td>
<td>G</td>
<td></td>
</tr>
<tr>
<td>O H D</td>
<td>0 Z D</td>
<td>B- 24</td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>O H I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>O L</td>
<td>B- 24</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>O L D</td>
<td>B- 24</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>O L I</td>
<td>B- 24</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>O S</td>
<td>B- 26</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>O S A</td>
<td>B- 26</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>O S B</td>
<td>B- 26</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>O S D</td>
<td>B- 103</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>O S I</td>
<td>B- 112</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>O X</td>
<td>B- 129</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>Move</td>
<td>B- 28</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>B- 28</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>B- 30</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>B- 31</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>B- 32</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>B- 34</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>7 *</td>
<td>B- 34</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>B- 35</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>8 *</td>
<td>B- 35</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>B- 46</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>B +</td>
<td>B- 48</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>B -</td>
<td>B- 48</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>B N</td>
<td>B- 46</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>B M</td>
<td>B- 46</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>B M N</td>
<td>B- 46</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>M B</td>
<td>B- 104</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>M B +</td>
<td>B- 105</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>M C 1</td>
<td>B- 106</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>M C 1 +</td>
<td>B- 107</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>M C 2</td>
<td>B- 108</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>M C 2 +</td>
<td>B- 109</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>M C 3</td>
<td>B- 110</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>M C 3 +</td>
<td>B- 111</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>M G</td>
<td>B- 111</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>M S</td>
<td>B- 114</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>M S N</td>
<td>B- 114</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>M S +</td>
<td>B- 115</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>B- 151</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>S S</td>
<td>B- 154</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The table entries represent command functions with their respective values. The values range from B-1 to B-204, indicating various functions and parameters. The table is organized by command functions, with each function listed with its corresponding value. This table is crucial for understanding the operational commands and inputs required for specific operations within a system or device.
* All RORZE products come with a 24-month guarantee.
* Specifications and products are subject to change without any obligation on the part of the manufacturer.