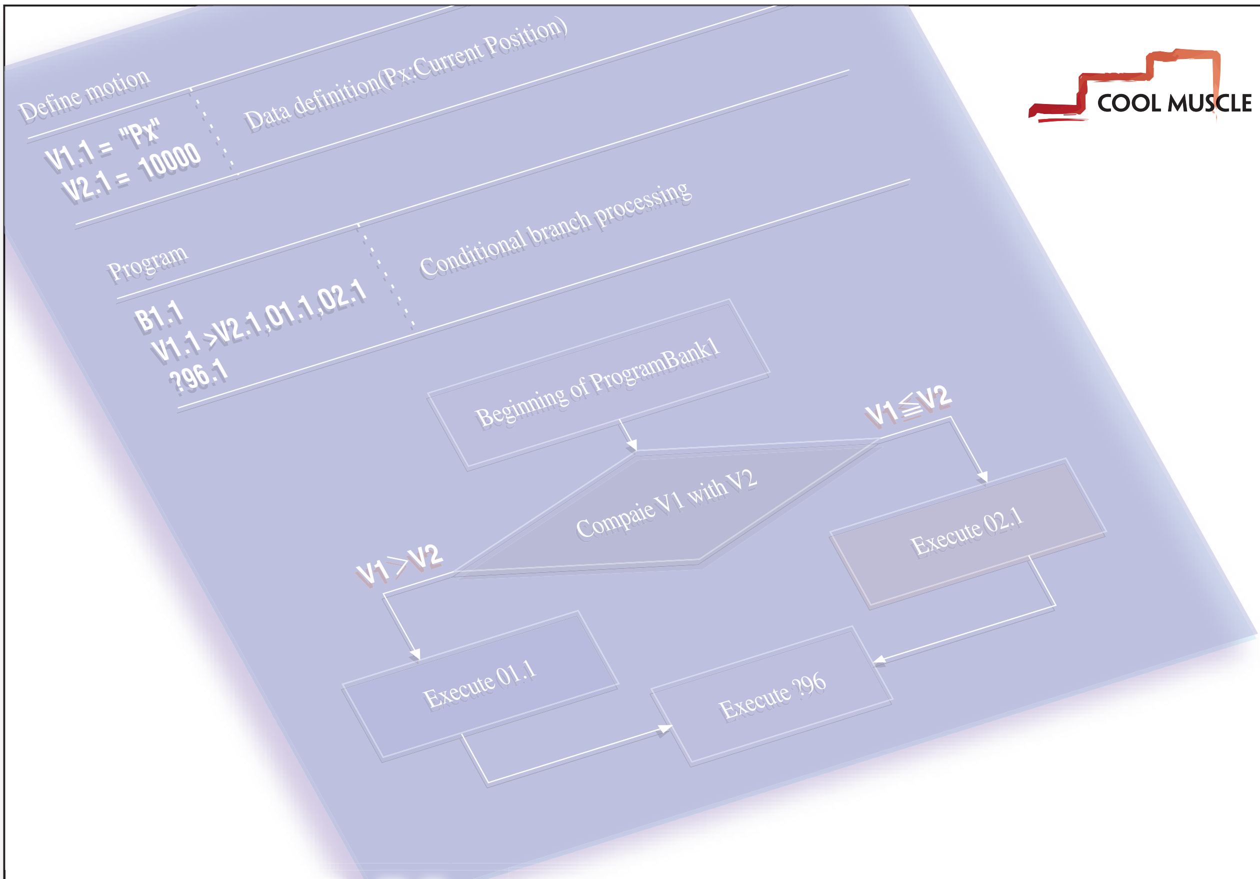
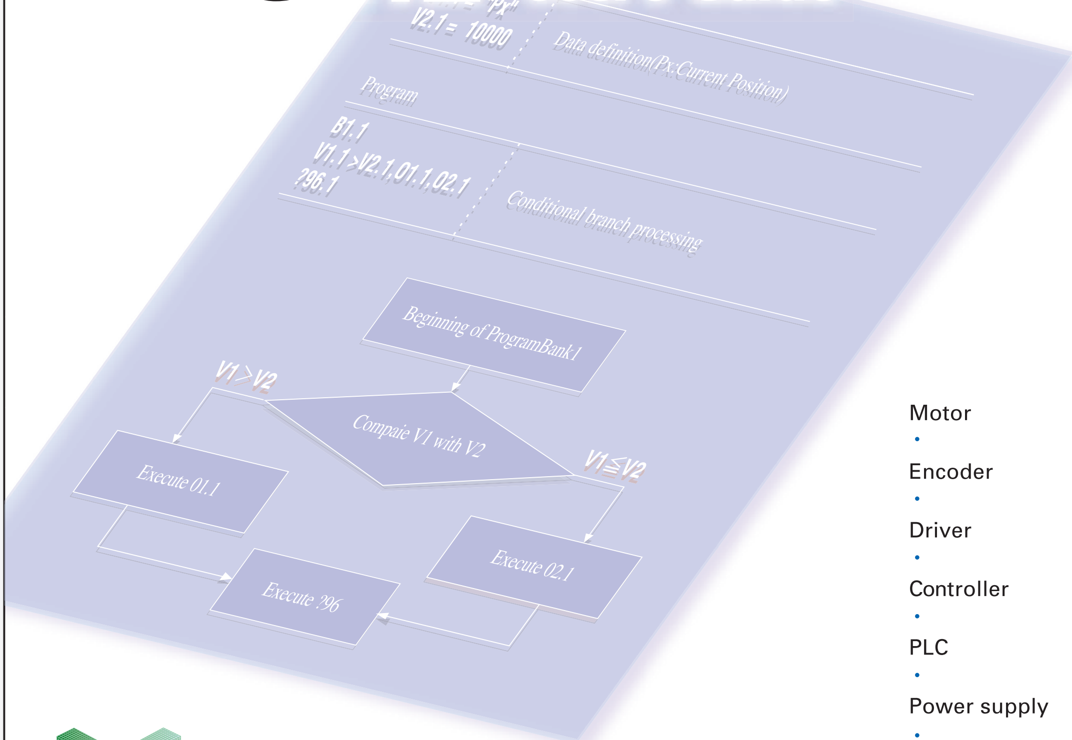
COOL MUSCLE




# CML User's Guide

MDUG-CML/09101E-01



Motion
Encoder
Driver
Controller
PLC
Power supply

All in **ONE** Solution

MUSCLE CORPORATION

☐ Before use, read through this User's Guide to ensure proper use.

☐ Keep this User's Guide at an easily accessible place so as to be referred anytime as necessary.

- The contents of this User's Guide are subject to change without notice for the improvement in product, specification, or usability of this User's Guide.
- This User's Guide is only intended to provide information about the product, and dose not guarantee any result from usage of the product. Muscle Corporation is not responsible for any damages and/or injuries resulting from the implementation in accordance with the contents of this User's Guide.
- Please notify our sales representative if you have some doubts or comments with the contents of this User's Guide.
- The contents of this User's Guide do not guarantee or grant rights to patents, copyright, or any other rights to the intellectual property of Muscle Corporation or any third party. Muscle Corporation is not responsible for any problems that may occur concerning the intellectual property rights of third parties resulting from the application of information provided in this User's Guide.
- Cool Muscle is a registered trademark of Muscle Corporation.
- Microsoft and Windows are registered trademarks of Microsoft Corporation in the United States and other countries.
- Other company names and product names described in this User's Guide are trademarks or registered trademarks of their respective holders. The trademark notices (TM, ®) are not necessarily appended to company, system, and product names described in this User's Guide.

●Before reading CML User's Guide, please read “CM2 User's Guide” for installation or operation of Cool Muscle and “CWL Operation Manual” for the usage of “COOLWORKS LITE”, Cool Muscle operation software.

# INDEX

# Chapter 5

# Chapter 6



# Explanation of icon

Icons used in this User's Guide.

 Warnings and notices

 Important points

 Supplemental explanations

Chapter 1

# CML Overview

## 1.1. What is CML?

CML is a short form of "Cool Muscle Language", which is a collection of commands used to control the motion of Cool Muscle. CML consists of the following commands.

● Parameters

Parameters set Cool Muscle's operating conditions. Do not change parameters while the motor is in motion. Please refer to Section 3.

● Data Commands

Data commands define the data for Cool Muscle's motion and support various kind of motion. Please refer to section 2.1.1, 2.2.1.

● Bank Commands

Bank Commands define motion logic. Program Banks are executed by the Execution commands.
Please refer to 2.2.2, 2.2.3.

● Execution Commands

Execution commands execute or stop motion of Cool Muscle.
Please refer to 2.1.2.

● Query

Query commands confirm Cool Muscle's current status (defined value as position, speed etc).
Please refer to 6.6.

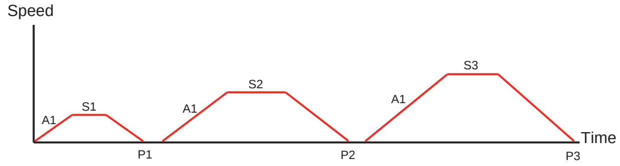
● Operator (Arithmetic / Logic / Comparison)

By using both data and bank commands, more complex motions are possible.
Please refer to 6.67, 6.78 for more detailed information.

Please use 1 byte character fonts only.
CML does not distinguish between upper case and lower case characters.

The following motion can be created by CML

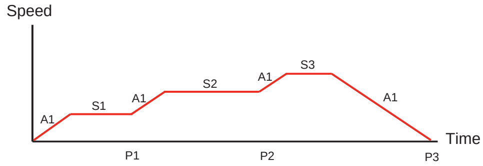
● PTP motion with different speeds




Example:

From the origin, the motor accelerates/decelerates using A1. Move with stops at each point (P1,P2,P3) changing the speed (S1, S2, S3).

● Merged Motion




Example:

From the origin, the motor moves to P3 with the acceleration/deceleration A1, changing speeds (S1,S2,S3) at each point (P1, P2) without stop.

● Motion Control for Multiple motors

By specifying the Motor ID, up to 15 motors can be controlled on a single network. 3 Dimensional motions can be accomplished on a single network for X, Y, Z applications.

● Circular / Linear Interpolation

Using the new interpolation commands, 2 axis systems can be coordinated and trace arcs and lines. Ovals are also possible.

● Conditional Branching

Using New logic operators, branching by multiple input or motor status is possible. It supports various branching as motion branching and conditional branching.

# 1.2. Motion Mode

There are 2 modes of operation in the Cool Muscle.

● Direct Mode

Like chatting online, you can control the Cool Muscle directly. Direct Mode is useful for an instant control, debugging, or the interrupt handling in a program (ex. forced termination). Direct Mode is available in all types of Cool Muscle.

● Program Mode

By using Bank commands, Cool Muscle executes motion according to the block of predefined motion logic (Bank command). There are Program Bank and Ladder Logic Bank as a block of motion logic. They can be stored in Cool Muscle's memory and executed by execution command or digital signal.

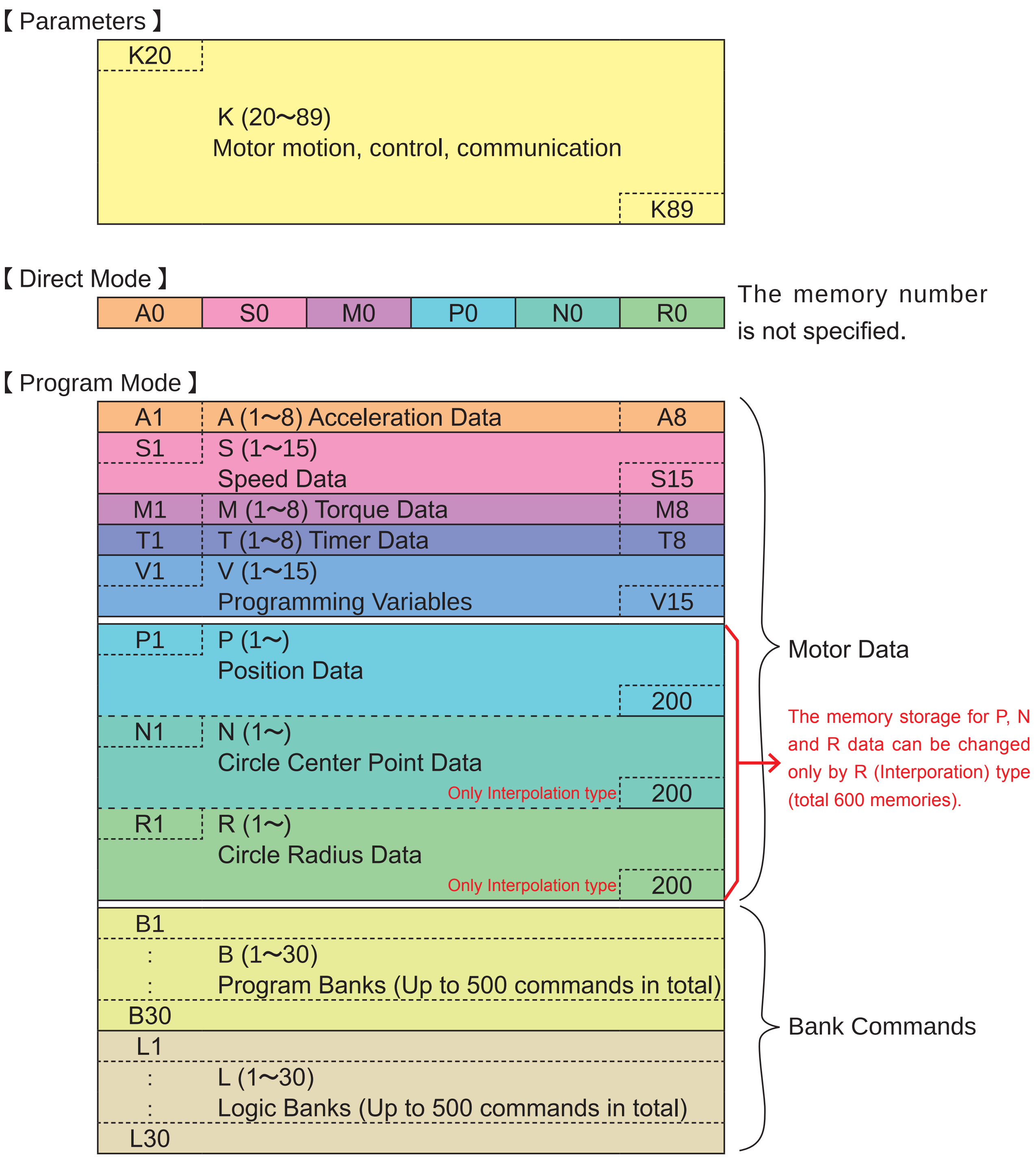
Program bank is useful for repetitive motion applications.

The process depending on input or motor status is described in Ladder Logic Bank. Ladder Logic Bank is scanned continuously in the background per set time by a parameter. It works as a simple sequencer or PLC.

*Program mode is not available with the P type.

# 1.3. Memory Map

By parameter commands, specified numbers of pre-set value can be stored in the Memory of Cool Muscle. Indicate a memory number following parameter commands to read or save the pre-set value. The following diagram outlines the memory composition.



Chapter **2**

# Operation by CML

## 2.1. Direct Mode

In Direct Motion, Position, Speed and Acceleration need to be predefined. Motion based on these predefined data is executed by execution command.

(n: Motor ID, ↵ : Enter Key input)

S.n= Value↵ …Define speed

A.n= Value↵ … Define Acceleration

P.n= Value ↵… Define Target Position

M.n= Value↵ … Define Torque Limit

^.n …Execute action based on the above values

### [Operation Example]

Let's operate Cool Muscle

First of all, define the data by entering numbers as below.

S.1=100
A.1=100
P.1=10000
M.1=100

Defined data can be confirmed by sending the query “?.1”

?.1 sent command to Cool Muscle

P.1=10000, S.1=100, A.1=100, M.1=100 replied data from Cool Muscle

Cool Muscle's default setting is Resolution 1000[ppr], Speed Unit 100[pps], so that the example above should be

Speed = S.1 value x Speed Unit = 100 x 100[pps] = 10000[pps]

Acceleration = A.1 value = 100[kpps2]

Target Position = P.1 value = 10000[pulse]

Torque Limit = M.1 value = 100[%].

Then operate Cool Muscle by entering the command as below.

^.1

Cool Muscle moves to the target position 10000[pulse] with the set speed and acceleration. After completion of positioning, Cool Muscle replies Ux.1=8 that means in-position status.

Current position can be confirmed by the query command ?96.1.

?96.1 sent command to Cool Muscle

Px.1=10000 replied data from Cool Muscle

### 2.1.1. Data Commands in Direct Mode

Motion commands are explained in the format below.

| Data Defining Commands | Functions |
|---|---|
| Unit | Description |
| Example | Description of example |

| P | Position Data Definition |
|---|---|
| Unit: pulse | This command defines Target Position. The value can be defined as relative against current position by using += or -=. If the value is set to 1000000000, the motor will run continuously.<br>Min: -1000000000<br>Max: 1000000000<br>*The setting range depends on K37. |
| P.1=10000 | Set Target Position to 10000 pulses for Motor 1. |
| P.1=-5000 | Set Target Position to -5000 pulses for Motor 1. |
| P.1+=100 | Add 100 pulses to the current position and set it as Target Position for Motor 1. |
| P.1-=200 | Deduct 200 pulses from the current position and set it as Target Position for Motor 1. |
| P.1=1000000000 | Set endless position as target position for Motor 1. |

| S | Speed Data Definition |
|---|---|
| Unit:<br>100pps or 10pps or 1pps<br>(Set by K37) | This command sets the motor Speed as an absolute value. As example, value is treated as +100 even if -100 is set.<br>Only when the motor is running continuously, set Speed to a positive number for CW direction motion, and set Speed to a negative number for CCW direction motion.<br>Min: -32767<br>Max: 32767 |
| S.1=250 | Set Motor 1 Speed to 25000/2500/250pps. |

| A | Acceleration Data Definition |
|---|---|
| Unit : $Kpps^2$ | This command sets Acceleration.<br>Min: -32767<br>Max: 32767 |
| A.1=100 | Set Motor 1 Acceleration to 100 $Kpps^2$. |

| M | Torque Limit Data Definition |
|---|---|
| Unit : % | This command sets Torque Limit using a percentage (0-100%) of the maximum motor torque.<br>Soon after setting M data, the motor torque should be limited by M data.<br>Min: 0<br>Max: 100 |
| M.1=50 | Set Motor 1 Torque Limit to 50% of the maximum motor torque. |

| N | Center Point Data of Circle Definition | |
|---|---|---|
| Unit: pulse | Only interpolation type can be used.<br>This command defines Center of an arc (circles, ovals, arcs) with 2 axes. | Min: -1000000000<br>Max: 1000000000<br>*The setting range depends on K37. |
| N.1=50, N.2=30 | Set Center of a circle to 50pulses for Motor 1 (X axis), and 30 for Motor 2 (Y axis) | |

| R | Radius Data of Circle Definition | |
|---|---|---|
| Unit: pulse | Only interpolation type can be used.<br>This command defines Radius for an arc (circles, ovals, arcs) with 2 axes.<br>When R values for both 2 axes are set to equal, then it will draw a circle.<br>When they are different, it will draw an oval.<br>When R is set to a positive number, a longer arc will be drawn. When it is set to a negative number, a shorter arc will be drawn.<br>When it is set to 0, line will be drawn. | Min: -1000000000<br>Max: 1000000000<br>*The setting range depends on K37. |
| R.1=80, R.2=80 | Set Radius to 80 pulses for Motor 1 (X axis) and Motor 2 (Y axis). | |

## 2.1.2. Execution Commands in Direct Mode

Execution commands are explained in the format below.

| Command | Function |
|---|---|
| Description | |
| Example | Explanation of Example |

| ^ | Execute the Direct Command Motion |
|---|---|
| This command executes motion using predefined Data Commands (S,A,P,M). | |
| S.1=250<br>A.1=100<br>P.1=10000<br>^.1 | Motor 1 moves to position 10000 with the speed 250 and acceleration 100Kpps2. |

| \| | Origin Search |
|---|---|
| This command makes the motor search an Origin based on Origin Search Parameters K42,43,45,46.<br>*This is a bar not the letter I. | |
| \|.1 | Motor 1 starts to search Origin. |

| \|1 | Move to Position 0 |
|---|---|
| This command makes the motor move to an Origin (Position 0). Acceleration and deceleration are set by Parameters K42,43. | |
| \|1.2 | Motor 2 moves to Origin. |

| \|2 | Assign Current Position to 0 |
|---|---|
| This command sets the current position to Origin (Position 0).<br>*No motion. | |
| \|2.3 | Set Motor 3's current potion to Origin |

| ( | Enable Motor |
|---|---|
| This command enables Motor. | |
| (.1 | Enable Motor 1. |

| ) | Motor Free |
|---|---|
| This command makes the motor “Motor Free”. | |
| ).1 | Make Motor 1 Motor Free. |

| O | Output Signal ON |
|---|---|
| This command turns the output on. Parameter K34 needs to be set to 4 (General).<br>Format: O#.n (# = Output No., n = Motor ID) | |
| O2.1 | Output 2 on Motor 1 is set to on. |

| F | Output Signal OFF |
|---|---|
| This command turns the output off. Parameter K34 needs to be set to 4 (General).<br>Format: F#.n (# = Output No., n = Motor ID) | |
| F2.1 | Output 2 on Motor 1 is set to off. |

| $ | Save Data |
|---|---|
| This command saves Parameters, Data Commands, Program Banks and Ladder Logic Banks into Cool Muscle's Memory. When data is saved, a message “saved. Motor ID” is returned.<br>Once saved, the data is kept after the motor is powered off. | |
| $.1 | Save Motor 1's Data like Program Banks. |

| ? | Query |
|---|---|
| This command shows Parameters, Data Commands, Program Banks and Ladder Logic Banks stored in Cool Muscle's Memory. | |
| ?.1 | Display the predefined data of Direct mode of Motor 1 |

| # | Capture Position Data |
|---|---|
| This command sets the current position data to a specified memory. | |
| #2.1 | Take the position memory No.2 from Motor 17s current position. |

| [ | Execute Program Bank |
|---|---|
| This command executes predefined Program Bank. | |
| [1.2 | Execute Motor 2's Program Bank 1 |

| ] | Pause Program Bank |
|---|---|
| This command stops all motors and pauses Program Bank in operation.<br>The “ [ “ re-starts Program Bank in pause.<br>When this command is entered twice, Program Bank is terminated and cannot be resumed. | |
| [1.1 | |
| ] | Stop all motors and pause Program bank 1. |
| ] | Program bank is terminated. |

| ]1 | Pause Specified Motor |
|---|---|
| This command specifies a motor on a daisy chain network to be paused. | |
| ]1.3 | Only Motor 3 pauses on a daisy chain network. |

| } | Stop after Completing Current Line |
|---|---|
| This command pauses the program bank after completing the current line in Program Bank.<br>The " [ " command re-starts the program bank in pause.<br>When this command is entered twice, Program Bank is terminated and cannot be resumed. | |
| }.1 | Motor 1 stops after completing the current line in Program Bank. |

| * | Emergency Stop |
|---|---|
| This command makes all motors stop with the maximum deceleration. This is used when emergency stop is required. To re-start the motion, you have to cancel Emergency Stop using *1 Command. The program is resumed with the next executable line.<br>Program Bank stops when this command is transmitted twice, and Program Bank operates from the beginning with command [ after canceling the emergency stop by command *1.<br>This command can be assigned to inputs. | |
| * | Execute an emergency stop |

| *1 | Cancel Emergency Stop |
|---|---|
| This command cancel Emergency Stop * and enable the motor. | |
| *1 | Cancel an emergency stop |

| > | Execute Next Line |
|---|---|
| This command executes the next line of Program Bank in pause.<br>After executing the last line of Program Bank, the motor executes no motion and reply "End!". | |
| >.1 | Execute the next line of Program Bank of Motor 1 in pause |

| < | Execute Previous Line |
|---|---|
| This command executes the previous line in Program bank in pause.<br>When execution is impossible, a message [Can't back!] is displayed. | |
| <.1 | Execute the previous line of Program Bank of Motor 1 in pause |

| [L | Execute Ladder Logic Bank |
|---|---|
| This command executes the specified Ladder Logic Bank in the background.<br>Format: [L#.n (#=Program Bank No., n=Motor ID) | |
| [L2.1 | Motor 1 executes Ladder Logic Bank 2 in the background. |

| ]L | Stop Ladder Logic Bank |
|---|---|
| This command stops Ladder Logic Bank running in the background. | |
| ]L.1 | Motor 1 stops Ladder Logic Bank running in the background. |

| @ | Execute Circular and Linear Interpolation Motion |
|---|---|
| Only Interpolation type can be used.<br>The starting point is the current position. Motors execute Circular or Linear Interpolation motion toward the set position based on set R or N data.<br>Format: @#.n modifier <+/-> (#=P memory No., n=Motor ID)<br>The modifier should be set to + for CW direction, and - for CCW direction. | |
| @.1+, @.2+<br>@3.1-, @4.2- | Motors execute Circular Interpolation motion for CW direction toward P positions of Motor 1 and Motor 2.<br>Motors execute Circular Interpolation motion for CCW direction toward P3 of Motor 1 and P4 of Motor 2. |

| \ (¥ or ₩) | Area division of Data Command |
|---|---|
| Only Interpolation type can be used.<br>The Data Command of P, N, and R in total 600 are divided the area.<br>The occupancy priority: P, N, R<br>After allocation of P, N should be allocated within the rest area. The rest area after allocation of N should be allocated for R automatically.<br>If the maximum number is allocated for P, N and R should be 0. | |
| \P300<br>\N200 | 300 pieces are allocated for P as a data definition area.<br>200 pieces are allocated for N as a data definition area.<br>The definition area of R becomes "600 - Number of P - Number of N" without specification. (R area should be 100 pieces in example.) |

## 2.2. Program Mode

In program mode, positions, speeds, accelerations need to be predefined. Using these predefined motion data, Program Banks can be created. Program Banks are executed by execution commands. Please refer to Chapter 4 for program examples for different applications.

In program mode, memory numbers should be specified after command.

The following shows basic structure of CML in program mode.
(#=Memory No., n=Motor ID, =Enter Key input)

S#.n=Value
A#.n=Value
P#.n=Value
V#.n=Value

Define each data using Motion commands

B#.n
P#.n=V#.n
S#.n, A#.n, P#.n
P#.n
END

Define motion order using Bank Commands.
⇒Bank Commands are described after B command (starting with Command B).

Transfer the predefined text files to the motor or input data directly via COOLWORKS LITE or Hyper Terminal.

[#.n …The specified Program Bank (B#) is executed based on the above definition.

It is suggested to create, edit and save Program Bank data as text files because whole Program Bank data should be transferred even though there is a small change.
Please save the file as .txt.

# [Operation Example]

Let's make a Program Bank, download it to Cool Muscle and execute it.

First of all, define the data that is necessary for Program Bank as below.

S1.1=100

A1.1=100

P1.1=10000

P2.1=0

T1.1=1000

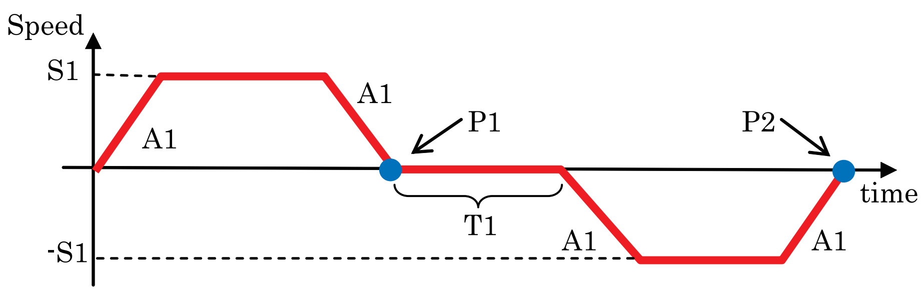
Then define the Program Bank as below.

| | |
|---|---|
| B1.1 | Start of Program Bank definition |
| S1.1, A1.1, P1.1 | Move to P1 with speed S1 and acceleration A1 |
| T1.1 | Timer for T1 |
| P2.1 | Move to P2 with the same speed and acceleration |
| END | The end of Program Bank definition |

After defining all data, execute the Program Bank 1 by entering a command as below.

[1.1

The motion in the diagram is executed as defined in Program Bank.



## 2.2.1. Data Commands in Program Mode

Data Commands can define multiple motion patterns. Each Data Command requires a memory number. The capacity of available memory space depends on the command.

Data Commands are explained in the format below.

| Motion Commands | Function | Available memory space |
|---|---|---|
| Unit | Description | |
| Example | Explanation of Example | |

| P | Position Data Definition | 1 ~ 200 |
|---|---|---|
| Unit: pulse | This command defines Target Position. The value can be defined as relative against set position by using += or -=. If the value is set to 1000000000, the motor will run continuously.<br>It can be defined up to 600 including Data Command N and R. (Interpolation type)<br>*When executing an relative positioning after rotating a motor shaft in motor free status, motor moves to a relative position from the current position before motor free. Execute "Counter Reset" in this case. | Min: -1000000000<br>Max: 1000000000<br>*The setting range depends on K37. |
| P2.1=10000 | Save the value of 10000 to Motor 1's P memory 2. | |
| P2.1=-5000 | Save the value of -5000 to Motor 1's P memory 2. | |
| P2.1+=1000 | Save the value of 1000 as the relative one to Motor 1's P memory 2. | |

| S | Speed Data Definition | 1 ~ 15 |
|---|---|---|
| Unit:<br>100pps or<br>10pps or<br>1pps<br>(Set by K37) | This command sets the motor Speed as an absolute value.<br>As example, value is treated as +100 even if -100 is set.<br>Only when the motor is running continuously, set Speed to a positive number for CW direction motion, and set Speed to a negative number for CCW direction motion. | Min: -32767<br>Max: 32767 |
| S2.1=250 | Save the value of 250 to Motor 1's S memory 2. | |

| A | Acceleration Data Definition | 1 ~ 8 |
|---|---|---|
| Unit: $Kpps^2$ | This command defines Acceleration. | Min: -32767<br>Max: 32767 |
| A2.1=100 | Save the value of 100 to Motor 1's A memory 2. | |

| T | Timer Data Definition | 1 ~ 8 |
|---|---|---|
| Unit: msec | This command defines Timer. | Min: 0<br>Max: 32767 |
| T2.1=1000 | Save the value of 1000 to Motor 1's T memory 2. | |

| M | Torque Limit Data Definition | 1 ~ 8 |
|---|---|---|
| Unit: % | This command sets Torque Limit using a percentage (0-100%) of the maximum motor torque.<br>Soon after setting M data, the motor torque should be limited by M data. | Min: 0<br>Max: 100 |
| M2.1=50 | Save the value of 50 to Motor 1's M memory 2. | |

| V | Variable Data Definition | 1 ~ 15 |
|---|---|---|
| Unit: - | This command is for mathematics operation or conditional branching by the value.<br>General Data can be defined up to 4 digit numbers like 4 characters or motor's internal state value. Note that " (double quotation) is needed to use characters and motor's internal state value.<br>Followings are motor internal state values.<br>Px…Current Position<br>Sx…Current Speed<br>Ix…Current Iq<br>Ux…Current Motor Status<br>Pe…Position Error<br>AIN…Analog Input<br>PT…Target Position<br>ST…Target Speed | Min: -1000000000<br>Max: 1000000000 |
| V2.1=12345678<br>V3.1="abcd"<br>V4.1="Px" | Save 12345678 to Motor 1's V memory 2.<br>Save abcd to Motor 1's V memory 3.<br>Save Px to Motor 1's V memory 4. | |

| N | Center Point Data of Circle Definition | 1 ~ 200 |
|---|---|---|
| Unit: Pulses | Only interpolation type can be used.<br>This command defines Center Point of an arc (circles, ovals) using 2 axes.<br>It can be defined up to 600. | Min: -1000000000<br>Max: 1000000000<br>*The setting range depends on K37. |
| N2.1=50,N2.2=30 | Save the values of 50 for Motor 1 (X axis) and 30 for Motor 2 (Y axis) to N memory 2 of each motor. | |

| R | Radius Data of Circle Definition | 1 ~ 200 |
|---|---|---|
| Unit: Pulses | Only interpolation type can be used.<br>This commands defines Radius of an arc (circles, ovals) using 2 axes.<br>0 must be set for Linear Interpolation.<br>Only interpolation type can be used.<br>It can be defined up to 600. | Min: -1000000000<br>Max: 1000000000<br>*The setting range depends on K37. |
| R2.1=80,R2.2=80 | Save the values of 80 for Motor 1 (X axis) and Motor 2 (Y axis) to R memory 2 of each motor. | |

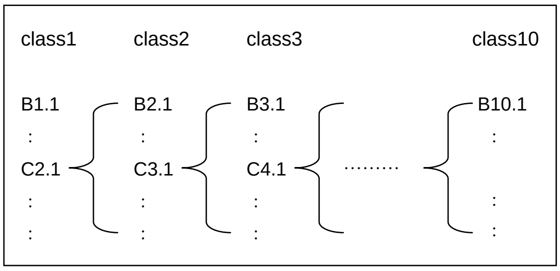
## 2.2.2. Program Bank Commands

Program Bank must start with the B command and end with End command. Program Bank is terminated also with the linefeed and without any command. Multiple commands in a single line are available and should be separated with commas. The maximum number of commands per motor is 500 commands in total.

Bank Commands are explained in the format below.

| Program Commands | Function | Available memory space |
|---|---|---|
| Description | | |
| Example | Explanation of Example | |

| B | Beginning of Program Bank | 1 ~ 30 |
|---|---|---|
| This command defines the beginning of Program Bank.<br>Format: B#.n (# = Program Bank No., n = Motor ID) | | |
| B2.1 | Define the beginning of Motor 1's Program Bank 2. | |

| C | Call other Program Bank | 1 ~ 30 |
|---|---|---|
| This command calls and executes the specific Program Bank, and back to the original Program Bank line after completing the called Program Bank.<br>C command can not be used to call the other ID's Program Bank. Program Bank where C command executes cannot be called again.<br>The maximum layer (nesting) should be under 10. |  | |
| B1.1<br>C2.1 | Motor 1's Program Bank1 calls and executes Motor 1's Program Bank 2. | |



| J | Jump to other Program Bank | 1 ~ 30 |
|---|---|---|
| This command jumps to and executes specific Program Bank.<br>But different from C command, it will not go back to the original Program Bank.<br>J command can be used to jump out of a looped program bank.<br>J command can not be used to jump to the other ID's program bank. | | |
| B1.1<br>J2.1 | Motor 1's Program Bank 1 jumps to and executes Motor 1's Program Bank 2. | |

| S | Speed | 1 ~ 15 |
|---|---|---|
| This command defines Speed in S memory space. This command needs to be defined before motion commands (P, Q, Y, Z, @). If S command is not defined, the previously used value will be applied. The specified memory value can be changed by the value from Arithmetic Operator. | | |
| S2.1, A2.1, P2.1 | Use the value defined in Motor 1's S memory 2 as Speed when Motor 1 moves to P2. | |
| S3.1=S2.1+V2.1 | Save the total value of the value defined in Motor 1's S memory 2 plus the value defined in Motor 1's V memory 2 to Motor 1's S memory 3. | |

| A | Acceleration | 1 ~ 7 |
|---|---|---|
| This command sets the acceleration value in a specified memory space.<br>This command needs to be defined before motion commands (P, Q, Y, Z, @). If the A command is not defined, the previously used acceleration will be applied. The specified memory value can be changed by the value from Arithmetic Operator. | | |
| S2.1, A2.1, P2.1 | Use the acceleration value stored in Motor 1's memory 2, the motor moves to position 2. | |
| A3.1=A2.1+V2.1 | Save the total acceleration value (acceleration value stored in motor1's acceleration memory position 2 plus the value stored in motor 1's general memory position 2 ) to motor 1's acceleration memory position 3. | |

| P | Position | 1 ~ 200 |
|---|---|---|
| This command saves the position value in a specified memory. Use + or - after Motor ID to make the value relative. This value can be added or subtracted from the current position.<br>The specified memory value can be changed by the value from Mathematics Operator. | | |
| S2.1, A2.1, P2.1 | Motor moves to P memory 2 with Acceleration memory 2 and Speed memory 2 respectively. | |
| P2.1+ | Motor moves from the current position by the travel distance defined by position memory 2. | |
| P3.1=V1.1+V2.1 | Save the total values stored general memory 1 and 2 to motor 1's position memory 3.<br>*When executing an relative positioning after rotating a motor shaft in motor free status, motor moves to a relative position from the current position before motor free. Execute "Counter Reset" in this case. | |

| Y | Execute next line without in-position queuing |
|---|---|
| Use this command instead of P to execute motion. However execute the next line of Program Bank without the in-position of Y command. Note that Program Bank may not be resumed after stop command during the operation of several Y commands. | |
| S2.1, A2.1, Y2.1<br>S3.2, A3.2, P3.2 | Motor 2 starts executing line 2 without waiting for Motor 1 to complete line 1. |

| Q | Push Motion |
|---|---|
| Use this command instead of P to execute push motion toward the target position.<br>If the motor reaches the target position before completing push motion, an error occurs (message, Ux.n=256).<br>To avoid this error, set the target position well behind the object that the motor pushes. Torque value and push time are defined by parameter K60 and K61. | |
| S2.1, A2.1, Q2.1 | The motor performs push motion from the current position to P memory No.2. |

| Z | Execute next line without push motion completion |
|---|---|
| Use this command instead of Q to execute push motion. However execute the next line of Program Bank without the in-position of Z command. | |
| S2.1, A2.1, Z2.1<br>S3.2, A3.2, P3.2 | Motor 2 starts executing line 2 without waiting for Motor 1 to complete line 1. |

| M | Torque Limit | 1 ~ 8 |
|---|---|---|
| This command sets Torque Limit using a percentage (0-100%) of the maximum motor torque. | | |
| M1.1=V5.1+V6.1 | Set the operated value from V5.1+V6.1 as value for M1.1. | |

| I | Conditional Branching on Input Status | 1 ~ 6 |
|---|---|---|
| This command makes conditional branching based on the specified input status. Conditional branching is possible based on the status of all Motors' ID on daisy chain network.<br>Use a logic operator when an action is based on the status of 2 inputs. | | |
| I2.1, C3.1, C4.1<br>I1.2 && I2.3, C3.1, C4.1 | If Motor 1's input 2 is on (true) then execute Program Bank No.3, if off (false) then call execute Program Bank No.4.<br>If Motor 1's input 1 and Motor 3's input 2 are on (true) then execute Program Bank No.3, else execute Program Bank No.4. | |

| T | Timer | 0 ~ 8 |
|---|---|---|
| This command sets the timer in timer memory locations. T0 means no action.<br>* Please specify same Motor ID for T command and B command. | | |
| T2.1 | Motor 1 waits for the time defined by Timer memory No.2. | |

| W | Timer in Conditional Branching | 0 ~ 8 |
|---|---|---|
| Use this command instead of T to wait for the time defined by T command while the specified input status is true. If the input status changes while the motor is waiting, then it resumes motion. If it is set to 0 then the motor waits indefinitely.<br>* Please specify same Motor ID for W command and B command. | | |
| I2.1, W2.1, ?99.1<br>P2.1 | If motor1's input 2 is on (true) then the motor waits for the time defined by T memory No.2. If the input status changes during the wait then the motor executes ?99 and the next line (move to P memory No.2). | |

| X | Looping | 0 ～ 255 |
|---|---|---|

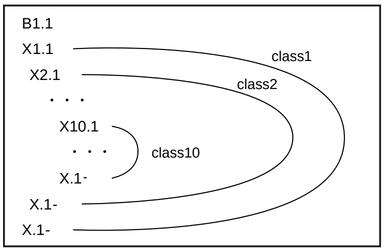
The program lines located between X and X- will be looped specified times.

The number of loops is defined between 0 and 255. When it is set to 0, it loops indefinitely.

The repeatable layer (nesting) is up to 10.

If the layer is over 10, the motion is not guranteed.

* Please specify same Motor ID for X command and B command.




| | |
|---|---|
| X3.1<br>S2.1, A2.1, P2.1<br>X.1- | The lines between X and X- will be looped three times. |

| V | Conditional Branching, calculation and data display using general data | 1 ～ 15 |
|---|---|---|

1) Conditional branching can be executed using general data value.
   Arithmetic or Logic operators can realize conditional branching with 2 general data values.
2) Arithmetic operator performs data calculations.
3) When this command is used alone, it displays the specified general data value.
   This is used for a message sent to a host.

* Please specify same Motor ID for V command and B command.

B1.3
•
•
•
V1.3 > V2.3, ～ , ～

| | |
|---|---|
| V2.1, ?99.1, ?98.1 | If V2.1>0, then execute ?99.1. If not, execute ?98.1. |
| V2.1== V3.1, ?99.1, ?98.1 | If V2.1 equals V3.1, then execute?99.1. if not, execute?98.1 |
| P2.1= P3.1+ V2.1 | Save the total value of P3 and V2 to Motor 1's position memory No.2. |
| V2.1 | Display motor 1's general data value 2 |

| N | Center Point of Circle | 1 ～ 200 |
|---|---|---|

Only interpolation type can be used.

When this command is described before @ command, it defines the specified N memory values as the center of a circle.

| | |
|---|---|
| N2.1, N2.2 | Set the center values stored in motor 1 and 2's center memory No.2 as the center position of a circle. |

| R | Radius of Circle | 1 ～ 200 |
|---|---|---|

Only interpolation type can be used.

When this command is described before @ command, it defines the specified R memory valuea as the radius of a circle.

The modifier after Motor ID, + or -, defines the arc size.

When R is set to a positive number, a major arc will be drawn and when it is set to a negative number, a minor arc will be drawn. If the values are set to 0, linear interpolation will be executed.

| | |
|---|---|
| R2.1, R2.2 | Set the values stored in Motor 1 and 2's Radius memory No.2 as the radius for a circle. |

| END | End of Program Bank |
|---|---|
| This command defines the end of each Program Bank. | |
| B1.1<br>S2.1, A2.1, P2.1<br>END | <br><br>End of Program Bank No.1 |

| , (comma) | Command Concatenation / Merge Motion / Simultaneous Motion Execution |
|---|---|
| When multiple commands are listed in a single line, each command need to be separated by a comma.<br>This allows for merge motion, instantaneous motion and dimultaneous motion by multiple axes. | |
| S2.1, A2.1, P2.1<br>A2.1, S2.1, P2.1, S3.1, P3.1<br>P2.1, P3.2 | Combining commands: move to P2 with Acceleration A2 and Speed S2.<br>Merge motion: Move to P3 without stopping at P2. Speed changes to S3 when passing P2.<br>Synchronous motion: Motor 1 moves to P2 and Motor 2 moves to P3 at the same time. |

| ; (semi colon) | Command Concatenation in Multiple Lines |
|---|---|
| This allows for multiple commands to combine over multiple lines. This can be used for combining commands, Merge motion and Synchronous motion. | |
| S2.1, A2.1, P2.1;<br>S3.1, P3.1 | Merge motion: Motor 1 moves to P3 without stopping P2. Speed changes to S3 when passing P2. (same as in a single line with commas.) |

| : (colon) | Command Concatenation in Branching |
|---|---|
| This command can realize to execute multiple commands in conditional branching. | |
| V1.1> V2.1, ?99.1: O1.1, ?96.1: F1.1 | If V1.1>V2.1, then execute ?99.1 and O1.1. If V1.1≦V2.1, then execute ?96.1 and F1.1. |

| // | Comment |
|---|---|
| This command allows you to write comments in Program Bank files. The description between this command and CRLF is not recognized as commands. Comments are not stored into Cool Muscle memory. | |
| // Comments here | Comments |

| Execute Commands | Execute commands within Program Bank |
|---|---|
| Various commands for Direct Mode are available in Program Bank.<br>Please refer to 2.1.2. ]1, [L, ]L, >, <, }, $ commands can not be used. | |

## 2.2.3. Ladder Logic Bank Commands

Ladder Logic Bank is independent from Program Bank and can be executed in the background. Therefore Cool Muscle can execute PLC functions in standalone mode, because they can execute the operations with defined data like Positions, Speeds and Accelerations. Ladder logic Bank execution cycle time is set by K63.
Ladder Logic Bank definition must start with the L1 command and finish with the End command. Ladder Logic Bank also finishes with two CRLFs without any command. Multiple commands in a single line must be separated by a comma. The maximum number of commands per motor is 500 commands in total.

Basic format for CML Ladder Logic Bank is as below.
(#: Memory No. , n: Motor ID. ↵ : Enter Key Input)

```
L#.n
P#.n = V#.n + V#.n
I#.n && I#.n, V#.n = V#.n, T0
END
```

Transfer the predefined text files to the motor or input data directly via COOLWORKS LITE or Hyper Terminal.

[L#.n ↵ …A specified Ladder Logic Bank performs operator processing based on predefined data.

# [Operation Example]

Let's make a Ladder Logic Bank, download to Cool Muscle and execute it.

First of all, define the data as below.

S1.1=50

S2.1=10

V1.1=”Px”

V2.1=5000

Then define a Ladder Logic Bank

| | |
|---|---|
| L1.1 | Start definition of Ladder Logic Bank |
| V1.1> V2.1, S.1= S1.1, S.1= S2.1 | |
| END | End of definition of Ladder Logic Bank |

This example shows Cool Muscle 2 moves with S1 (50) when V1 (current position) is bigger than V2 (5000) but with S2 (10) when V1 is smaller than V2.

After the definition, enter the command as below and execute the Ladder Logic Bank 1.

[L1.1

Lets make a motion in Direct Mode as follows.

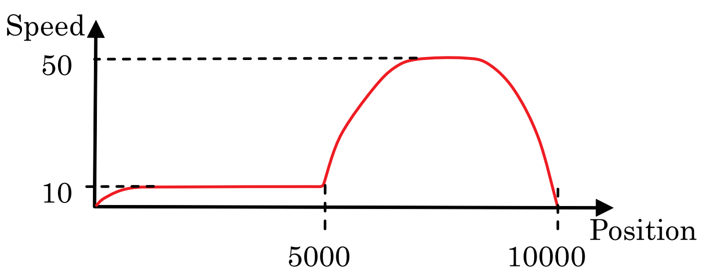
A.1=100

P.1=10000

^.1

This example shows Cool Muscle 2 moves with Speed 10 when the current position is smaller than 5000 and with Speed 50 when the current position is bigger than 5000 by executing Ladder Logic Bank in the background.

To stop or pause Ladder Logic Bank, ]L command is needed

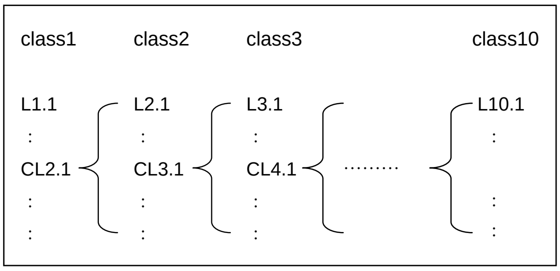


Ladder logic bank commands are explained in the format below.

| Bank command | Function | Available memory space |
|---|---|---|
| Description | | |
| Example | Explanation of Example | |

| L | Beginning of Ladder Logic Bank | 1 ~ 30 |
|---|---|---|
| This command defines the beginning of a Ladder Logic Bank.<br>Format: L#.n (#=Program Bank No., n=Motor ID) | | |
| L2.1 | Begin the definition of Motor 1's Ladder Logic Bank 2. | |

| CL | Call other Ladder Logic Bank | 1 ~ 30 |
|---|---|---|
| This command calls and executes the specific Ladder Logic Bank, and back to the original Ladder Logic Bank line after completing the called Ladder Logic Bank.<br>CL command can not be used to call the other ID's Ladder Logic Bank.<br>The maximum layer (nesting) should be under 10. | | |
| L1.1<br>CL2.1 | Motor 1's Ladder Logic Bank No.1 calls Motor 1's Ladder Logic Bank No.2 and executes it. | |




| JL | Jump to other Ladder Logic Bank | 1 ~ 30 |
|---|---|---|
| This command jumps to and executes specific Ladder Logic Bank.<br>But different from CL command, it will not go back to the original Ladder Logic Bank.<br>JL command can be used to jump out of a looped Ladder Logic Bank.<br>JL command can not be used to jump to the other ID's Ladder Logic Bank. | | |
| L1.1<br>JL2.1 | Motor 1's Ladder Logic Bank No.1 calls Motor 1's Ladder Logic Bank No.2 and executes it. | |

| I | Conditional Branching on Input Status | 1 ~ 6 |
|---|---|---|
| This command makes conditional branching based on the specified input status. Conditional branching is possible based on the status of all Motors' ID on daisy chain network.<br>Use a logic operator when an action is based on the status of 2 inputs. | | |
| I2.1, CL3.1, CL4.1 | If Motor1's input 2 is on(true), then execute Ladder Logic Bank No.3. if off(false), then call execute Ladder Logic Bank No.4 | |
| I1.2 && I2.3, CL3.1, CL4.1 | If Motor 1's input 1 and 2 are on(true), then execute Ladder Logic Bank No.3. if not then, execute Ladder Logic Bank No.4. | |

| T | Timer | 0 ～ 8 |
|---|---|---|
| This command sets the timer in timer memory locations. T0 means no action.<br>* Please specify same Motor ID for T command and L command. | | |
| T2.1 | Motor 1 waits for the time defined by Timer memory No.2. | |

| W | Timer in Conditional Branching | 0 ～ 8 |
|---|---|---|
| Use this command instead of T to wait for the time defined by T command while the specified input status is true. If the input status changes while the motor is waiting, then it resumes motion. If it is set to 0 then the motor waits indefinitely.<br>* Please specify same Motor ID for W command and L command. | | |
| I2.1, W2.1, ?99.1<br>O1.1 | If motor1's input 2 is on (true) then the motor waits for the time defined by T memory No.2. If the input status changes during the wait then the motor executes ?99 and the next line (move to P memory No.2). | |

| # | Capture Position Data |
|---|---|
| This command sets the current position data to a specified memory.<br>This function is the same as the position teaching. | |
| #2.1 | Take the position memory No.2 from Motor 1's current position. |

| V | If then motion, calculation and data display using general data | 1 ～ 15 |
|---|---|---|
| 1) Conditional branching can be executed using general data value.<br>Arithmetic or Logic operators can realize conditional branching with 2 general data values.<br>2) Arithmetic operator performs data calculations.<br>3) When this command is used alone, it means the specified general data value.<br>This is used for a message sent to a host.<br>* Please specify same Motor ID for V command and L command.<br>B1.3<br>•<br>•<br>•<br>V1.3 > V2.3, ～ , ～ | | |
| V2.1, ?99.1, ?98.1<br>V2.1== V3.1, ?99.1, ?98.1<br>P2.1= P3.1 + V2.1<br>V2.1 | If V2.1>0, then execute ?99.1. If not, execute ?98.1.<br>If V2.1 equals V3.1, then execute?99.1. if not, execute?98.1<br>Define the value of P3 + V2 as Motor 1's P memory 2.<br>Motor 1 shows the data defined in General Data memory 2. | |

| END | End of Ladder Logic Bank |
|---|---|
| This command defines the end of each Ladder Logic Bank. | |
| L1.1<br>V2.1= V2.1 + V3.1<br>END | End of Ladder Logic Bank No.1. |

| , (comma) | Command Concatenation |
|---|---|
| When multiple commands are listed in a single line, each command need to be separated with a comma. | |
| V2.1>V3.1, V2.1=V3.1, T0.1 | Combines commands |

| ; (semi colon) | Command Concatenation in Multiple Lines |
|---|---|
| This allows for multiple commands to combine over multiple lines. This can be used for combining commands, Merge motion. | |
| V2.1>V3.1;<br>V2.1=V3.1, T0.1 | Combines commands over several lines. |

| : (colon) | Command Concatenation in Branching |
|---|---|
| This command can realize to execute multiple commands in conditional branching. | |
| V1.1> V2.1, ?99.1: O1.1, ?96.1: F1.1 | If V1.1>V2.1, then execute ?99.1 and O1.1. If V1.1≦V2.1, then execute ?96.1 and F1.1. |

| // | Comment |
|---|---|
| This command allows you to write comments in Ladder Logic Program files. The description between this command and CRLF is not recognized as commands. Comments are not stored into Cool Muscle memory. | |
| // Comments here | Comments |

| Execute Commands | Execute commands within program bank |
|---|---|
| Various commands for Direct Mode are available in Ladder Logic Bank.<br>Please refer to 2.1.2. ]1, [L, ]L, >, <, }, $ commands can not be used. | |

Chapter 3

# Parameter Setting

The Cool Muscle has initial settings that can be adjusted based on your application. Please refer 6.1.
Each parameter is identified by a unique number and has a specific function. To set a parameter, enter a desired value following the = sign as below.

K [Parameter No.] . [Motor ID] =value

Each parameter has individual setting range.
The value out of the range will not be reflected. The changed value is saved automatically.

## 3.1. K parameters

The following chart outlines each K Parameter's usage

| Parameter No. | Setting Item | Unit |
|---|---|---|
| Parameter Description | | |

[Setting Example]

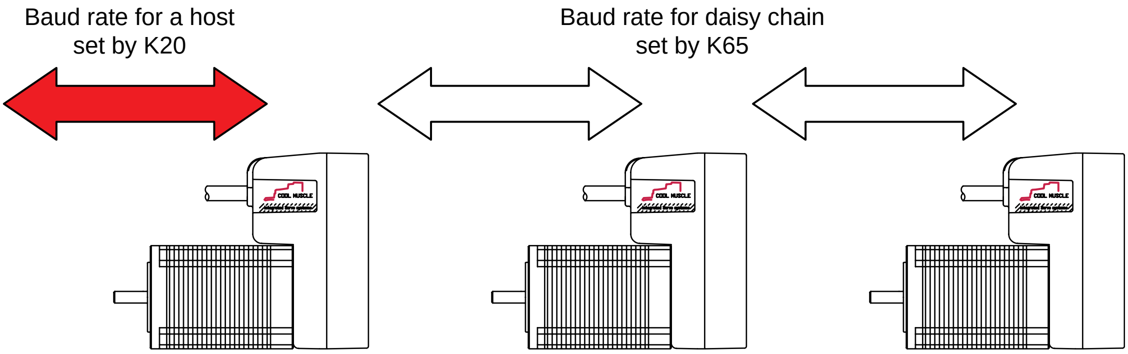
The example and explanation about parameters.

| K20 | Baud Rate | Unit : — |
|---|---|---|

Set the baud rate for the communication between Cool Muscle and a host.
When changed, the host baud rate needs to be changed to match Cool Muscle's changed baud rate.

| Value | Baud rate |
|---|---|
| 0 | 38.4Kbps |
| 1 | 9.6 Kbps |
| 2 | 19.2 Kbps |
| 3 | 57.6 Kbps |
| 4 | 115.2 Kbps |
| 5 | 230.4 Kbps |

Some PCs and host instruments can not be adapt the set baud rate.
Please enter a proper value that matches their specifications.




[Setting Example]

K20.1=1 Set 9.6kbps to the baud rate.

Cool Muscle's communication buffer could be overflowed by a delay of communication data processing when a lot of data are transferred to Cool Muscle and over-written at a time with high communication speed.

| K23 | Status Report | Unit : — |
|---|---|---|

Defines the status report method as an automatic report by each event when status changes. Local echo of sent data from a host, confirmation messages or error massages for mis-operation can be set by this parameter as well. It can be set by the addition of the function No. 1-16 (Max. value is 31)

| Value | Status Report Method |
|---|---|
| 0 | No status report |
| 1 | Automatically report to a host when in-position and alarm occur. |
| 2 | Automatically report to a host when input status changes. |
| 4 | Automatically report to a host when output status changes. |
| 8 | No local echo |
| 16 | Various confirmation messages and error messages will be reported to a host. |

[Confirmation Messages]

| Messages | Description |
|---|---|
| [End of Bank] | Program Bank input is finished properly. |
| Change Baud Rate ??<br>XXX kbps (Y/N) | Confirmation message when the baud rate<br>is changed by K20 |

[Error message]

| Messages | Description |
|---|---|
| error00.n: Out Of Range!! | K Parameter value is out of range |
| error01.n: syntax error!! | Program Bank syntax error |
| error02.n: too many steps!! | Program Bank steps exceed max. numbers. |
| error03.n: XX is not allowed in bank.1 | XX command can not be defined. |
| error04.n: XX can not be followed by DD | XX command can not be defined before DD. |
| error05.n: Program Bank does not exist!! | Program Bank does not exist. |
| error06.n: Ladder Bank does not exist!! | Ladder Logic Bank does not exist. |
| error07.n: CW Limit!! | CW limit sensor is on |
| error08.n: CCW Limit!! | CCW limit sensor is on |

## [Setting Example]

K23.1=13

1: Automatically report to a host when in-position and alarm occur.

4: Automatically report to a host when output status changes.

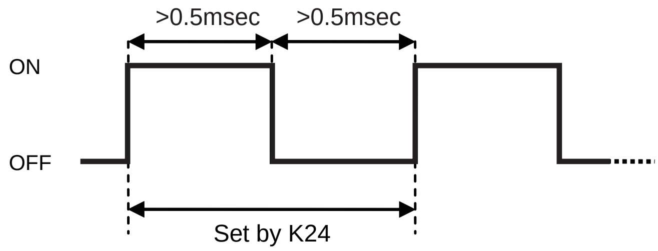
8: No local echo

When 3 functions are combined, the value shall be 1+4+8=13 by addition.

| K24 | Rotational Pulse Output | Unit : pulses |
|---|---|---|

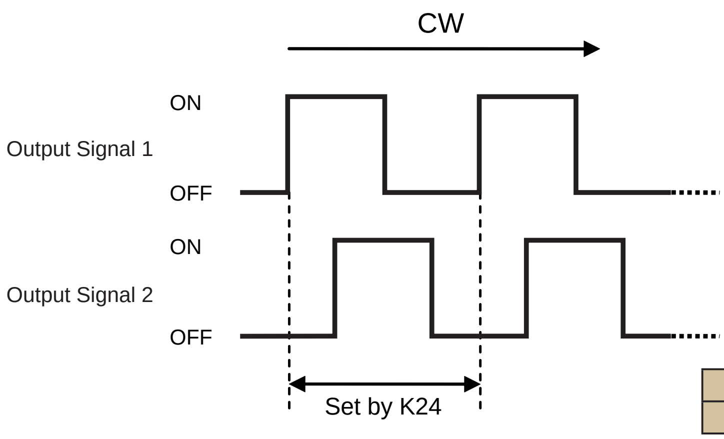
When Output Functions Parameter K34=7, the output is turned ON/OFF at regular interval of pulses set by this parameter.

The signal wave is as shown in the diagram below. The output is turned ON at the first half of the set pulses, then OFF at the last half.




Note: The ON/OFF output pulse width is required to be set more than 0.5 msec as shown in the diagram.

When K34=77, the outputs are guadrature encoder pulse signals, in which the phase between two signals is different by 90-degrees, as shown below.




| | |
|---|---|
| Min | 10 |
| Max | 32767 |

## [Setting Example]

K24.1=1000 The output, set by K34=7, turns ON and OFF every 1000 pulses.

| K25 | Delay Time for Slow Response Signal | Unit : 0.1sec |
|---|---|---|

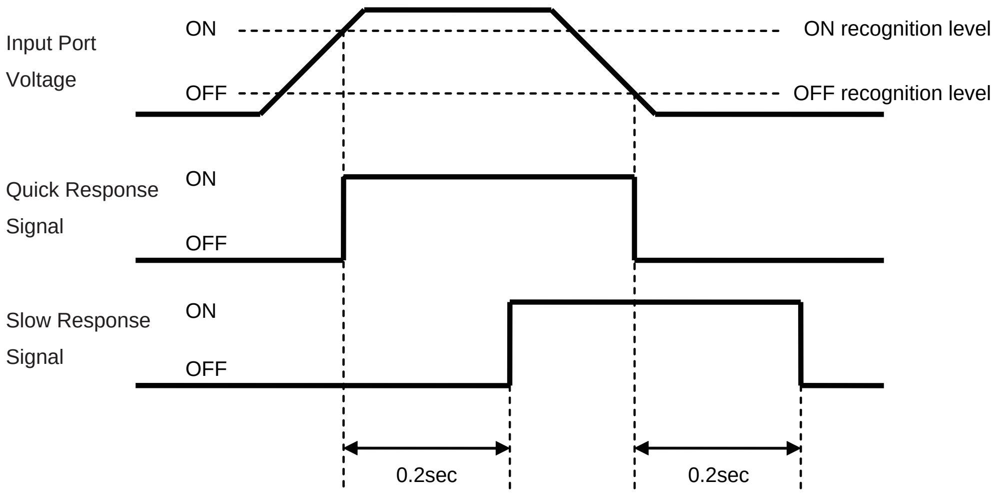
Based on the original signal, 2 signals of Quick and Slow Responses can be recognized. Slow Response is a virtual signal that is generated after a specified delay time. This increases the number of input points to which functions can be assigned. This parameter sets the offset time for Slow Response Signal to be recognized after Quick Response Signal.

Each digit must be set individually in order of Input 6,5,4,3,2,1.

- Quick / Slow Response Signals

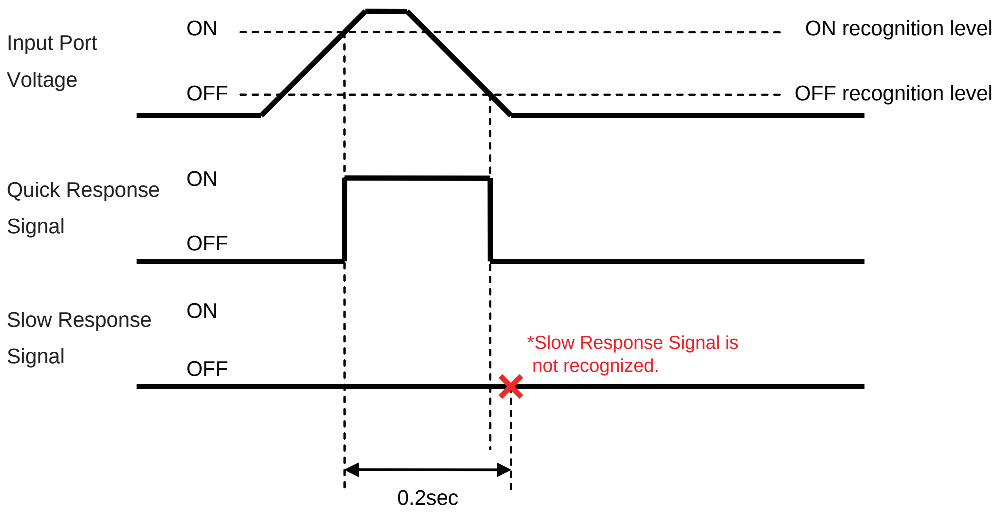
The diagrams below show Quick Response Signal is recognized as ON when the actual input voltage to the input port exceeds the ON recognition level, and as OFF when the voltage falls below the OFF recognition level. (In case of K26=0) The input logic is set by K26.

When delay time for Slow Response Signal is set to 0.2sec, Slow Response Signal is generated 0.2sec after the Quick Response Signal is recognized. Functions can be assigned to the rising edge, the target level and the falling edge of each signal.




- Short signal width

When a signal disappears within the offset time, only a Quick Response Signal is recognized.



## [Setting Example]

K25.1 = 976532

- INPUT 1: Set 0.2sec to the delay time for Slow Response Signal
- INPUT 2: Set 0.3sec to the delay time for Slow Response Signal
- INPUT 3: Set 0.5sec to the delay time for Slow Response Signal
- INPUT 4: Set 0.6sec to the delay time for Slow Response Signal
- INPUT 5: Set 0.7sec to the delay time for Slow Response Signal
- INPUT 6: Set 0.9sec to the delay time for Slow Response Signal

| K26 | Input Logic / P type Operation | Unit : — |
|---|---|---|

This parameter sets

① Input Logic (the logic for input signals and the effective edge for command pulse inputs)

② Execution of P type Operation (applied to C/R type)

Set each function by the digit in order of Input 6,5,4,3,2,1

To each Input 1 ~ 6,

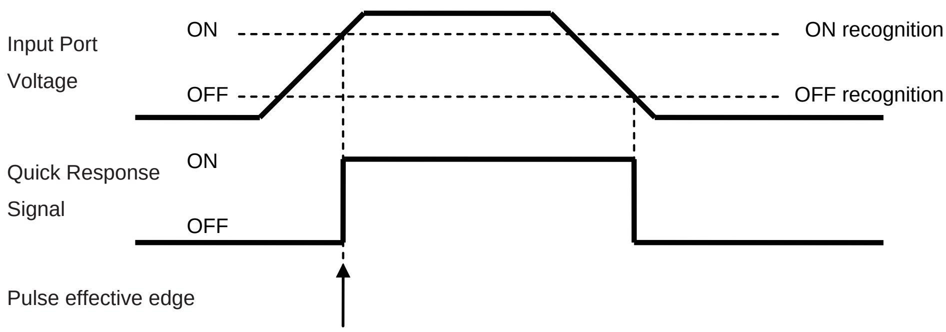
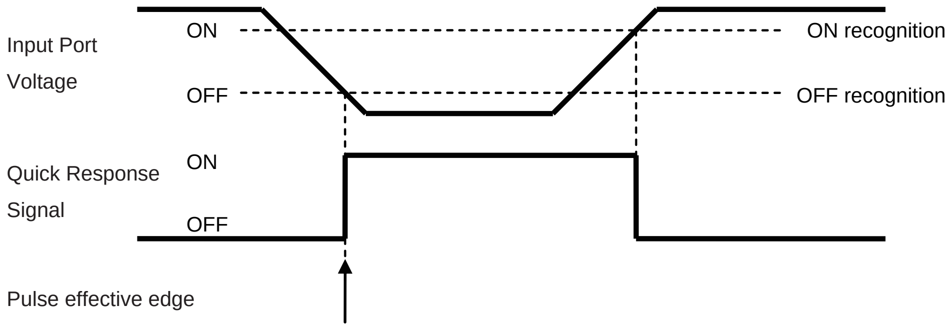
- Set "0" or "1" for only setting Input Logic
- Set "2" or "3" for setting the execution of P type Operation besides Input Logic.

The setting value of "2" or "3" should be used when the rotation control of motor, P type Operation, by the command pulse train to Input 1 and Input 2 is needed for C/R type Cool Muscle.

The execution of P type Operation is enabled by the input of which the setting value is "2" or "3" in Input 3, 4, 5 and 6.

① Setting for Input Logic

<table>
<tr><th>Value</th><th colspan="2">Judgment of "Input Signal ON"</th><th colspan="2">Effective edge of Command Pulse Input</th></tr>
<tr><td>0 or 2</td><td rowspan="2">When the specified Input port is</td><td>ON</td><td>Rising Edge of</td><td rowspan="2">Input Port</td></tr>
<tr><td>1 or 3</td><td>OFF</td><td>Falling Edge of</td></tr>
</table>

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0 or 2</td><td>Input signal is ON when input port is ON<br>Effective edge: Rising edge of input port<br><br></td></tr>
<tr><td>1 or 3</td><td>Input signal is ON when input port is OFF.<br>Effective edge: Falling edge of input port<br><br></td></tr>
</table>

[Setting Example]

K26.1 = 000010

INPUT 1: Input signal is ON when Input port is ON.
INPUT 2: Input signal is ON when Input port is OFF.
INPUT 3: Input signal is ON when Input port is ON.
INPUT 4: Input signal is ON when Input port is ON.
INPUT 5: Input signal is ON when Input port is ON.
INPUT 6: Input signal is ON when Input port is ON.

② Setting for execution of P type Operation (Applies to C/R type)

The input to switch either P type Operation is available or unavailable is specified by this setting.
Set the value "2" or "3" to the input for switching use in Input 3 to Input 6.
The execution of P type Operation is available or unavailable by the state of specified input signal.
P type Operation is executed as long as the input signal is ON, and then the command pulse input to Input 1 and Input 2 is effective.

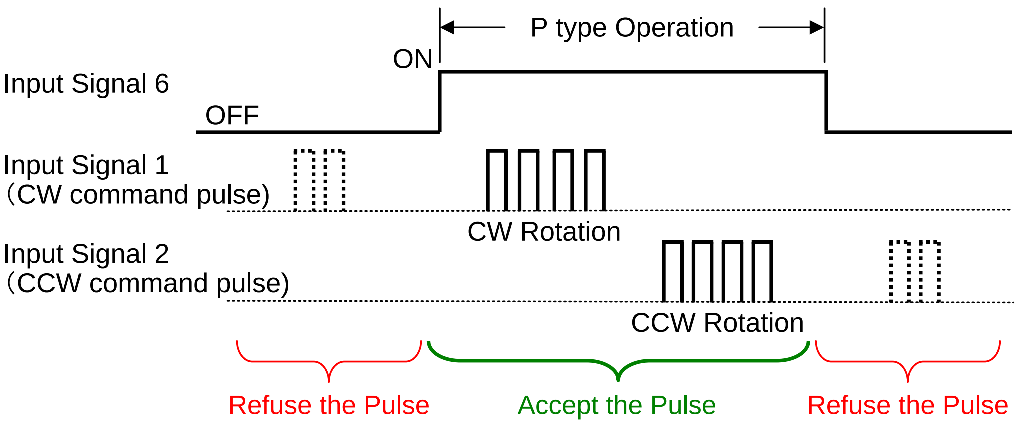
| When Input Signal is | Execution of P type Operation |
|---|---|
| ON | P type Operation is valid and accept the Command Input Pulse |
| OFF | P type Operation is Invalid and refuse the Command Input Pulse |

・When multiple inputs are set to "2" or "3", P type Operation is executed as long as the signal of any input of them is ON.
・When Cool Muscle receives commands from the host during P type Operation, the processing of the command is given priority and executed. (C type Operation priority)

When any Program Bank is not executed, the execution of P type Operation is permitted.
During the execution of Bank Program, it can not be switched to P type Operation even if the specified input signal is ON.

[Setting Example]

K26.1 = 2XXX00



* When the setting value of Input 1 or Input 2 is "2" or "3".
P type Operation is executed at all times and the motor rotation is controlled by only the command pulse input to Input 1 and Input 2.

[Setting Example]

K26.1 = XXXX22
K26.1 = XXXXX2
K26.1 = XXXX2X

P type Operation Only (C/R type Operation is not available)

| K27 | Input Functions at the Quick Response Target Voltage (QTV) | Unit : — |
|---|---|---|
| K30 | Input Functions at the Slow Response Target Voltage (STV) | Unit : — |

These parameters assign functions performed at the target voltage level of quick and slow response signals.

Please note that input functions should not be interfered with each other, when assinging.

(Ref the diagram in K25 description)

Set each function by the digit order of Input 6,5,4,3,2,1.

| # | Functions | Description |
|---|---|---|
| 0 | No Function | — |
| 1 | General Use | Used by Command I in program execution. |
| 2 | Origin Sensor Signal (K27)<br>— (K30) | The signal from Origin Sensor. (K27)<br>— (K30) |
| 3 | Manual feed CW | Motor rotates in CW direction while the input signal is ON, with the speed and acceleration set by K49 and K43. |
| 4 | Manual Feed CCW | Motor rotates in CCW direction while the input signal is ON, with the speed and acceleration set by K49 and K43. |
| 5 | Stop Ladder Logic Bank | Stop Ladder Logic Bank |
| 6 | CW Direction Limit Sensor (CW Origin Sensor combined use) | Usually used for a CW direction limit sensor.<br>When an origin sensor signal is not assigned to other inputs, this input works as an origin sensor signal for the origin search motion to CW direction. |
| 7 | Emergency Stop | Emergency Stop by an input signal on (stop by Max. deceleration)<br>Emergency Stop is canceled by an input signal off. |
| 8 | Stop Program Bank | Stops motion and Program Bank execution. Same as ]] command. |
| 9 | CCW Direction Limit Sensor (CCW Origin Sensor combined use) | Usually used for a CCW direction limit sensor.<br>When an origin sensor signal is not assigned to other inputs, this input works as an origin sensor signal for the origin search motion to CCW direction. |

## [Setting Example]

K27.1=871243

INPUT 1: Manual Feed CW

INPUT 2: Manual Feed CCW

INPUT 3: Origin Sensor

INPUT 4: General Use

INPUT 5: Emergency Stop

INPUT 6: Stop Program Bank

| | | |
|---|---|---|
| K28 | Input Functions at the Quick Response Rising Edge (QR) | Unit : — |
| K31 | Input Functions at the Slow Response Rising Edge (SR) | Unit : — |

These parameters assign functions performed at the Quick and Slow rising edges of signals.

Please note that input functions should not be interfered with each other, when assinging.

For example, assign “Motor Free” to a rising edge of Quick Response Signal and “Start Origin Search” to a falling edge of Slow Response Signal, Cool Muscle goes into motor free state before starting the origin search. (Ref the diagram in K25 description)

Set each function by the digit in order of Input 6,5,4,3,2,1.

| Value | Function | Description |
|---|---|---|
| 0 | No Function | — |
| 1 | Alarm Reset / Program Bank Pause | This resets alarms, and pauses motion. Pause Program Bank being executed.<br>Re-start from paused position is possible by 6: Execute Program Bank 1. |
| 2 | Motor Free | Make a motor go into motor free state and servo OFF. |
| 3 | Position Counter Reset | Make the current position to 0 (the Origin) |
| 4 | Execute Next Program Bank Line | Execute the next line in a Program Bank<br>B1<br>S1,A1,P3 (Line 1)<br>S2,A2,P2 (Line 2)<br>Rising Edge: Execute line 1<br>Next Rising Edge : Execute Line 2 |
| 5 | Execute Previous Program Bank Line | Execute a previous line in a Program Bank<br>This function could not be performed depending on the content of Program Bank. |
| 6 | Execute Program Bank 1 | Execute Program Bank 1. |
| 7 | Start Origin Search | Start Origin Search. |
| 8 | Manual Jog CW / Execute Program Bank 2 | Motor rotates to the amount of feed pulses set in parameter K50, in CW direction.<br>When K36=2 or 3, execute Program Bank 2. |
| 9 | Manual Jog CCW / Execute Program Bank 3 | Motor rotates to the amount of feed pulses set in parameter K50, in CCW direction.<br>When K36=2 or 3, execute Program Bank 3. |

## [Setting Example]

K28.1 = 987612

INPUT 1: Motor Free
INPUT 2: Alarm Reset / Program Bank Pause
INPUT 3: Execute Program Bank 1
INPUT 4: Start Origin Search
INPUT 5: Manual Jog CW
INPUT 6: Manual Jog CCW

| | | |
|---|---|---|
| K29 | Input Functions at the Quick Response Falling Edge (QF) | Unit : — |
| K32 | Input Functions at the Slow Response Falling Edge (SF) | Unit : — |

These parameters assign functions performed at the Quick and Slow falling edges of a signals.

Please note that input functions should not be interfered with each other, when assinging.

For example, assign “Motor Free” to a rising edge of Quick Response Signal and “Start Origin Search” to a falling edge of Slow Response Signal, Cool Muscle goes into motor free state before starting the origin search. (Ref the diagram in K25 description)

Set each function by the digit in order of Input 6,5,4,3,2,1.

| Value | Function | Description |
|---|---|---|
| 0 | No Function | — |
| 1 | Alarm Reset / Program Bank Pause | This resets alarms, and pauses motion. Pause Program Bank being executed.<br>Re-start from paused position is possible by 6: Execute Program Bank 1. |
| 2 | Enable Motor | Cancel “Motor Free” and servo ON. |
| 3 | Position Counter Reset | Make the current position to 0 (the Origin). |
| 4 | Execute Next Program Bank Line | Execute the next line in a Program Bank.<br>B1<br>S1,A1,P3 (Line 1)<br>S2,A2,P2 (Line 2)<br>Rising Edge: Execute line 1<br>Next Rising Edge : Execute Line 2 |
| 5 | Execute Previous Program Bank Line | Execute the previous line in a Program Bank.<br>This function could not be performed depending on the content of Program Bank. |
| 6 | Execute Program Bank 1 | Execute Program Bank 1. |
| 7 | Start Origin Search | Start Origin Search. |
| 8 | Manual Jog CW / Execute Program Bank 2 | Motor rotates to the amount of feed pulses set in parameter K50, in CW direction.<br>When K36=2 or 3, execute Program Bank 2. |
| 9 | Manual Jog CCW / Execute Program Bank 3 | Motor rotates to the amount of feed pulses set in parameter K50, in CCW direction.<br>When K36=2 or 3, execute Program Bank 3. |

[Setting Example]

K29.1 = 987612

INPUT 1: Enable Motor
INPUT 2: Alarm Reset / Program Bank Pause
INPUT 3: Execute Program Bank 1
INPUT 4: Start Origin Search
INPUT 5: Manual Jog CW
INPUT 6: Manual Jog CCW

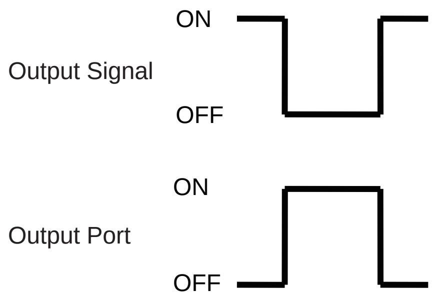
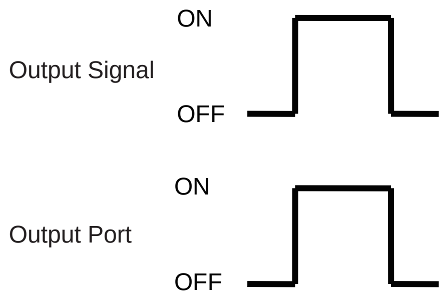
| K33 | Output Logic | Unit : — |
|---|---|---|

This parameter sets the output logic.

0 (Low Active): Command F or when output signal by output function is OFF, turn ON the output port.

1 (High Active): Command O or when output signal by output function is ON, turn ON the output port.

Set each function by the digit in order of Output 4,3,2,1.

| Value | Description |
|---|---|
| 0 | Low Active.<br>Output port is ON<br>when output signal is OFF.<br> |
| 1 | High Active.<br>Output port is ON<br>when output signal is ON.<br> |

[Setting Example]

K33.1 = 0001

- OUTPUT 1: High Active
- OUTPUT 2: Low Active
- OUTPUT 3: Low Active
- OUTPUT 4: Low Active

| K34 | Output Functions | Unit : — |
|---|---|---|

This parameter assigns a function to an output.

Set each function by the degit in order of Output 4, 3, 2, 1.

| Value | Functions | Description |
|---|---|---|
| 0 | No Function | — |
| 1 | In-Position | In-Position signal. |
| 2 | Alarm | Alarm signal. |
| 3 | General Use | Output by Command O / Command F. |
| 4 | Completion of Origin Search | Output In-Position signal only when the origin search is completed. |
| 5 | — | — |
| 6 | In-position Signal in Merge Motion | Output In-Position signal at the passing points in merge motion.<br>Set a signal width by parameter K73. |
| 7 | Rotation Pulse Output | Output a signal at certain intervals. Set its interval by parameter K24.<br>When Output 1 and Output 2 are set to 7 , they are the quadrature encoder outputs. |
| 8 | In Motor Free | Output a signal during motor free state. |
| 9 | In Push Motion | Output a signal during push motion. |

## [Setting Example]

K34.1=2177

OUTPUT 1: Quadrature Encoder Output

OUTPUT 2: Quadrature Encoder Output

OUTPUT 3: In-position

OUTPUT 4: Alarm

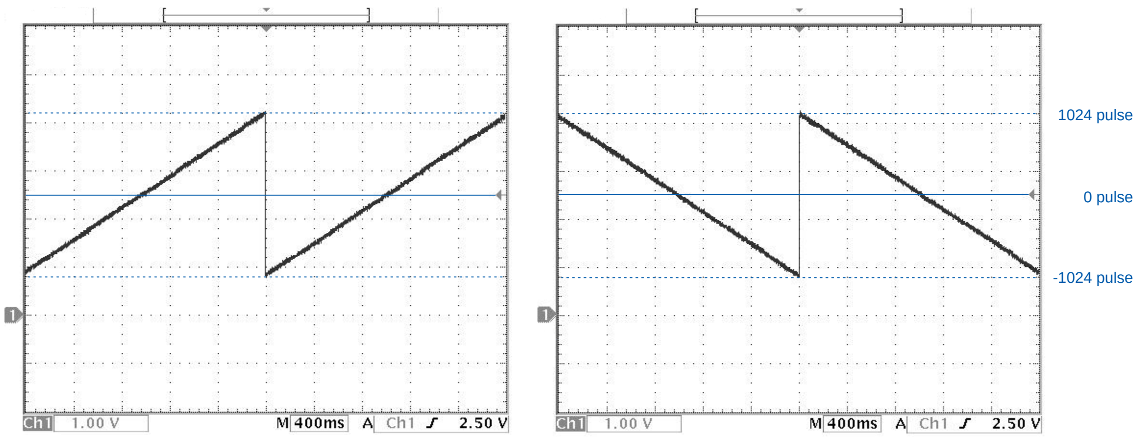
| K35 | Analog Output Functions | Unit : — |
|---|---|---|

This parameter sets a function to Analog Output.

| Value | Analog Output Types | Unit |
|---|---|---|
| 0 | Target Position | ± 1024 pulses/ ± 1.67V |
| 1 | Target Position data magnified by 8 | ± 128 pulses/± 1.67V |
| 2 | Current Position | ± 1024 pulses/± 1.67V |
| 3 | Current Position data magnified by 8 | ± 128 pulses/± 1.67V |
| 4 | Position Error | ± 1024 pulses/ ± 1.67V |
| 5 | Position Error data magnified by 8 | ± 128 pulses/ ± 1.67V |
| 6 | Current Speed | ± 2400min$^{-1}$/ ± 1.67V |
| 7 | Current Speed data magnified by 8 | ± 300min$^{-1}$/ ± 1.67V |
| 8 | Current Torque | ±100%/ ± 1.67V |
| 9 | Current Torque data magnified by 8 | ±12.5% / ± 1.67V |

The analog output that you select can be monitored by an oscilloscope. The output wave is plus minus 1.67V based on the center value of DC+2.5V. When the wave goes over the range 2.5V, it will be replicated.

[Position Data output example]



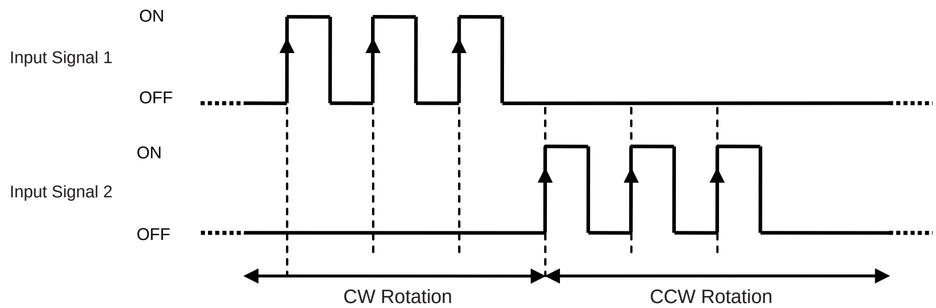

## [Setting Example]

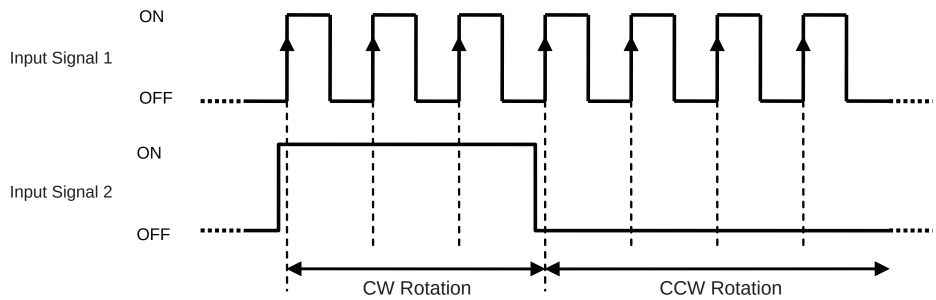
K35.1=3 Set “Current Position data magnified by 8” to Analog Output

| K36 | Command Pulse Format | Unit : — |
|---|---|---|

This parameter sets the format of command pulse Input.
Either CW/CCW or Pulse/Direction can be chosen according to the command pulse train to Input 1 and Input 2.
When K36 is set to 2 or 3 for the other type of Cool Muscle than P type, the function of "Execute Program Bank 2" and "Execute Program Bank 3" can be assigned to the rising edge and/or the falling edge of input signals. (Refer to K28, K29, K31, K32)

| Value | Input form of Input signal | Execute Program Bank 2 and 3 |
|---|---|---|
| 0 | CW/CCW | — |
| 1 | Pulse/Direction | — |
| 2 | CW/CCW | Can be assigned by K28, K29, K31, K32 |
| 3 | Pulse/Direction | Can be assigned by K28, K29, K31, K32 |

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0 or 2</td><td>CW / CCW Pulse<br>Set Input Signal 1 to CW pulse, Input Signal 2 to CCW pulse.<br>

<br>When detecting the rising edge of Input Signal 1, count up in CW direction<br>When detecting the rising edge of Input Signal 2, count up in CCW direction</td></tr>
<tr><td>1 or 3</td><td>Pulse / Direction<br>Set Input Signal 1 to command pulse, Input Signal 2 to direction pulse.<br>

<br>When detecting the rising edge of Input Signal 1, count up in CW direction in case Input Signal 2 is ON<br>When detecting the rising edge of Input Signal 2, count up in CCW direction in case Input Signal 2 is OFF</td></tr>
<tr><td>2 or 3</td><td>Execute Program Bank 2 and Program Bank 3 (Can be assigned by K28, K29, K31, K32)<br>*Except for P type</td></tr>
</table>

[Setting Example]

K36.1=0 Set "CW/CCW Pulse" for Command Pulse Format

K36.1=3 Set "Pulse/Direction" for Command Pulse Format
"Execute Program Bank 2 and Program Bank 3" is available by Input Functions

| K37 | Resolution / Speed Unit | Unit : — |
|---|---|---|

This parameter sets the motor's resolution and the speed unit that is used by S command.

Each value of 0-10 or 40-50 sets 100pps as the speed unit, each value of 20-30 or 60-70 sets 10pps as the speed unit and each value of 80-90 sets 1pps as the speed unit. The maximum position data is limited depending on the Motor Resolution.

| Speed unit 100pps | | | Speed unit 10pps | | | Speed unit 1pps | | |
|---|---|---|---|---|---|---|---|---|
| # | Resolution | Max Position (±) | # | Resolution | Max Position (±) | # | Resolution | Max Position (±) |
| 0 | 200 | 8,589,934 | 20 | 200 | 8,589,934 | 80 | 200 | 8,589,934 |
| 1 | 400 | 17,179,869 | 21 | 400 | 17,179,869 | 81 | 400 | 17,179,869 |
| 2 | 500 | 21,474,836 | 22 | 500 | 21,474,836 | 82 | 500 | 21,474,836 |
| 3 | 1000 | 42,949,672 | 23 | 1000 | 42,949,672 | 83 | 1000 | 42,949,672 |
| 4 | 2000 | 85,899,345 | 24 | 2000 | 85,899,345 | 84 | 2000 | 85,899,345 |
| 5 | 2500 | 107,374,182 | 25 | 2500 | 107,374,182 | 85 | 2500 | 107,374,182 |
| 6 | 5000 | 214,748,364 | 26 | 5000 | 214,748,364 | 86 | 5000 | 214,748,364 |
| 7 | 10000 | 429,496,729 | 27 | 10000 | 429,496,729 | 87 | 10000 | 429,496,729 |
| 8 | 25000 | 999,999,999 | 28 | 25000 | 999,999,999 | 88 | 25000 | 999,999,999 |
| 9 | N/A | N/A | 29 | N/A | N/A | 89 | N/A | N/A |
| 10 | 50000 | 999,999,999 | 30 | 50000 | 999,999,999 | 90 | 50000 | 999,999,999 |
| 40 | 300 | 12,884,901 | 60 | 300 | 12,884,901 | | | |
| 41 | 400 | 17,179,869 | 61 | 400 | 17,179,869 | | | |
| 42 | 600 | 25,769,803 | 62 | 600 | 25,769,803 | | | |
| 43 | 800 | 34,359,738 | 63 | 800 | 34,359,738 | | | |
| 44 | 1200 | 51,539,607 | 64 | 1200 | 51,539,607 | | | |
| 45 | 1500 | 64,424,509 | 65 | 1500 | 64,424,509 | | | |
| 46 | 3000 | 128,849,018 | 66 | 3000 | 128,849,018 | | | |
| 47 | 4000 | 171,798,691 | 67 | 4000 | 171,798,691 | | | |
| 48 | 6000 | 257,698,037 | 68 | 6000 | 257,698,037 | | | |
| 49 | 8000 | 343,597,383 | 69 | 8000 | 343,597,383 | | | |
| 50 | 12000 | 515,396,075 | 70 | 12000 | 515,396,075 | | | |

When it is set to 40-70, the incremental motion cannot be executed.

Continuous motion (P=1000000000) is still available in any resolution.

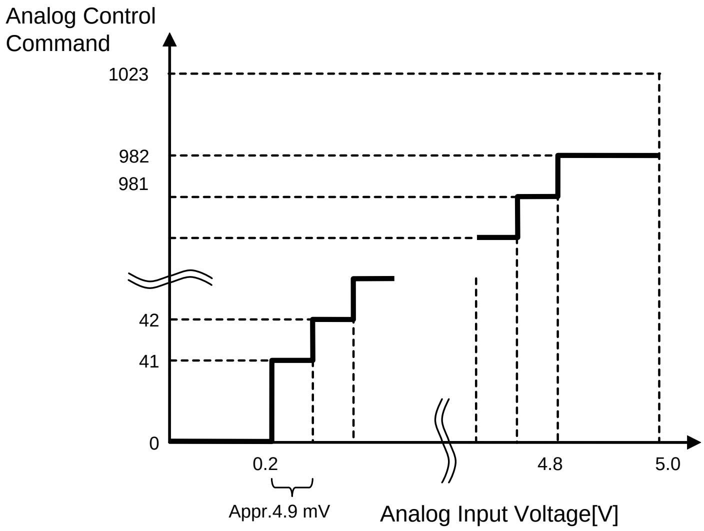
## [Setting Example]

K37.1=3 Set 1000ppr to the Motor Resolution, 100pps as the Speed Unit

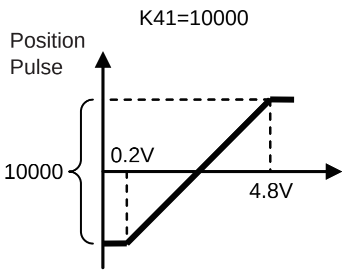
| K38 | Analog Control Type | Unit : — |
|---|---|---|

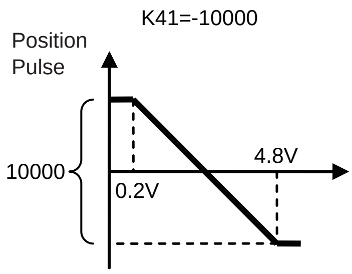
This parameter sets the control type for Analog Input.

Cool Muscle is equipped with 10 bit A/D converter. The analog input voltage 0 to 5V is divided into 1024, and utilized as an analog control command from 0 to 1023. Therefore the control command changes per approximately 4.9mV. However, from 0 to 0.2V and from 4.8 to 5V are the dead zone so that actual control range is from 0.2V to 4.8V.

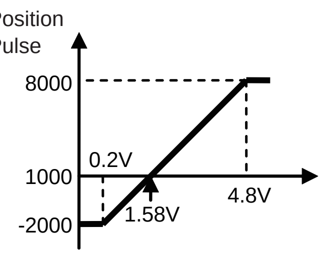


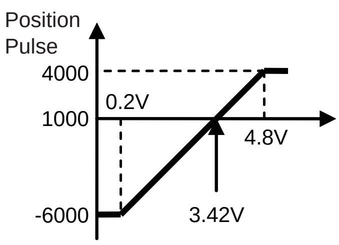

<table>
<tr><th>#</th><th>Functions</th><th>Description</th></tr>
<tr><td>0</td><td>Not Function</td><td>Analog control is not applied.</td></tr>
<tr><td>1</td><td>Position control</td><td>Control the position in the range of K41 by changing the analog input voltage when motor is not rotating.



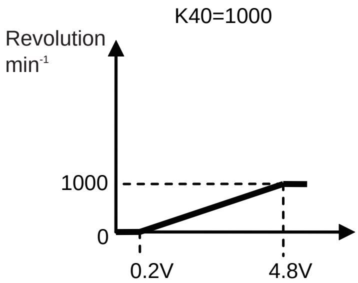

* The plus/minus control range in the set range is different depending on the current position and the applied voltage when the analog control is activated.

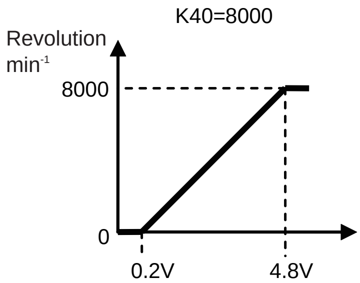


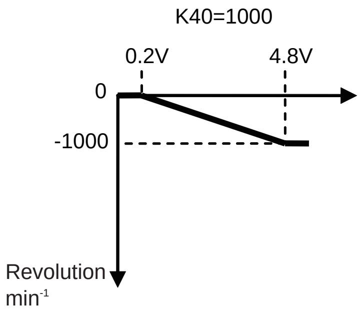
</td></tr>
</table>

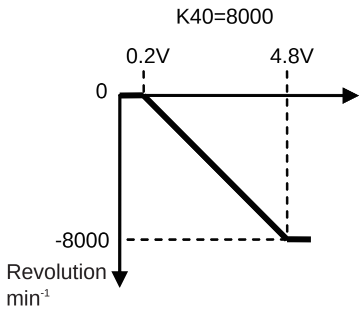
| | | |
|---|---|---|
| 2 | Speed control for CW | Control the speed in CW direction in the range of K40 by changing the analog input voltage.<br>* Will not be functional unless set to 0 [min$^{-1}$] (analog voltage less than 0.2V) only for the first time in control. |



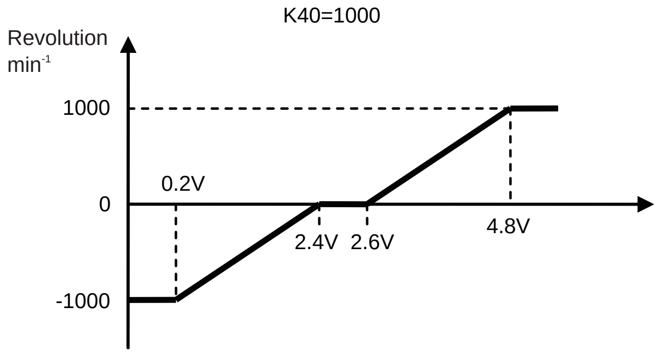




| | | |
|---|---|---|
| 3 | Speed control for CCW | Control the speed in CCW direction in the range of K40 by changing the analog input voltage.<br>* Will not be functional unless set to 0 [min$^{-1}$] (analog voltage less than 0.2V) only for the first time in control. |

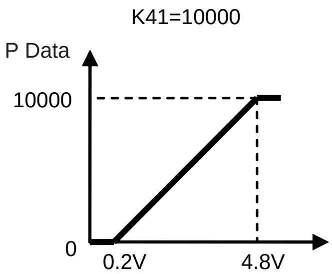


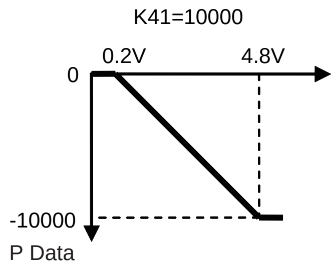




| | | |
|---|---|---|
| 4 | Speed control for CW / CCW | Control the speed in CW/CCW direction in the range of K40 by changing the analog input voltage.<br>2.5V for 0 [min$^{-1}$], rotate in CW direction by Input Voltage>2.5V, rotate in CCW direction by Input Voltage < 2.5V. From 2.4V to 2.6V is dead zone.<br>* Will not be functional unless set to 0 [min$^{-1}$] (analog voltage from 2.4V to 2.6V) only for the first time in control. |

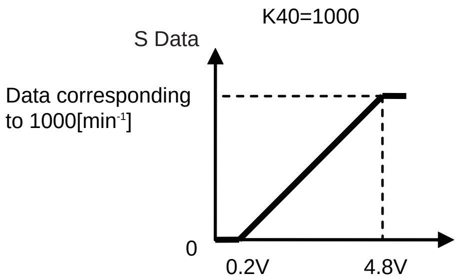



| | | |
|---|---|---|
| 5 | P data for Direct Mode | Change the Position Data P for Direct Mode in the range of K41 by changing the analog input voltage. |

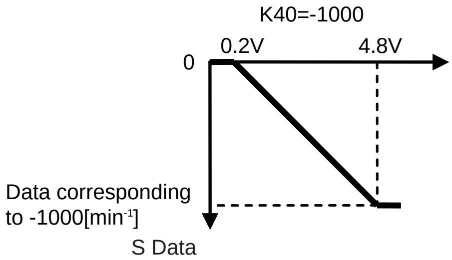


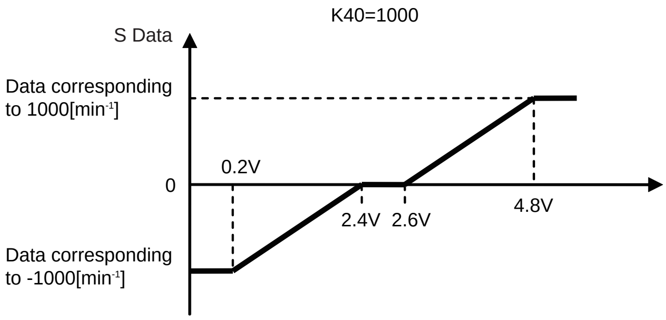
<table>
<tr><td>6</td><td>S data (+) for Direct Mode</td><td>Change the Speed Data S for Direct mode in plus(+) direction in the range of K40 by changing the analog input voltage.



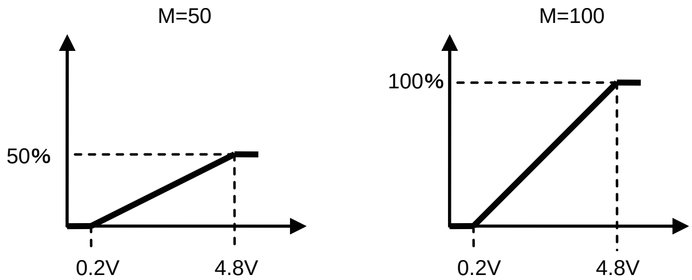
</td></tr>
<tr><td>7</td><td>S data (-) for Direct Mode</td><td>Change the Speed Data S for Direct mode in minus(-) direction in the range of K40 by changing the analog input voltage.



</td></tr>
<tr><td>8</td><td>S data (+/-) for Direct Mode</td><td>Change the Speed Data S for Direct Mode in the range of K40 by changing the analog input voltage.
Change S data in plus(+) direction when Input Voltage>2.5V. In minus(-) direction when Input Voltage<2.5V. (Between 2.4V and 2.6V is the dead zone).



</td></tr>
<tr><td>9</td><td>Torque control</td><td>Control the torque within the range of set value by M command.



</td></tr>
<tr><td>10</td><td>Torque feedback control</td><td>Feedback control by inputting the output analog voltage from an external torque sensor to keep stable torque on the control target that the sensor is equipped with.
Ref, Section 5.4 Torque Feedback Control.
Ref, Parameters K74, K75, K76, K77 for the Gain (proportional and integral gain), Torque Sensor Input Offset Value and Torque Sensor Input Range.</td></tr>
</table>

## [Setting Example]

When using the speed control for CW/CCW direction

K38.1=4 Set "Speed Control for CW/CCW" as the analog control type

K40.1=2000 Set 2,000min-1 to Max. Speed

Increase the speed in CW direction by applying analog input voltage from 2.6V to 4.8V and increase the speed in CCW direction when applying analog input voltage from 2.4V to 0.2V.

Reach the Max. Speed 2000min$^{-1}$ in each direction when 0.2V or 4.8V is applied.

When using the position control

K38.1=1 Set "Position Control" as the analog control type

K41.1=10000 Set 10000 pulse to the travel range

Move 0 to 10000 pulses when changing analog input voltage from 0.2V to 4.8V.

| K39 | Low Pass Filter Cut-off Frequency | Unit : ×5rad/s |
|---|---|---|

Cut-off frequency of low pass filter for the analog input

There is no filter when the value of 1024 is set.

(unit: 5[rad/sec] = 5000[times/sec]/1024)

| | |
|---|---|
| Min | 0 |
| Max | 1024 |

[Setting Example]

K39.1=128 Set 640[rad/sec] to the analog input cut-off frequency

| K40 | Maximum Speed | Unit : $min^{-1}$ |
|---|---|---|

This parameter sets motor's maximum speed.

For the speed control by the analog input, this parameter sets the maximum speed when the maximum analog voltage is applied.

The conversion from the speed unit [$min^{-1}$] to S value is as show in below.

S value = speed[$min^{-1}$] x resolution[ppr] / speed unit[100pps or 10pps] / 60

Ex) K40=2000, K37=3 (Resolution 1000ppr, Speed Unit 100pps)

Max S value = 2000 x 1000 / 100 / 60 = 333

| | |
|---|---|
| Min | 1 |
| Max | Depends on motor type |

## [Setting Example]

K40=2000

Set 2000[$min^{-1}$] to the motor's maximum speed.

When using the speed control by the analog input through setting parameter K38, the motor's maximum speed reaches to 2000[$min^{-1}$] when the maximum analog voltage is applied.

| K41 | Analog Travel Range | Unit : pulses |
|---|---|---|

This parameter sets the maximum travel range in the position control by the analog input, where the input voltage varies from 0.2V to 4.8V.

| | |
|---|---|
| Min | -32767 |
| Max | 32767 |

[Setting Example]

K38=1

K41=4000

If the current position is 0, the position of motor will be controlled in the range from 0 to 4000 according to an analog input voltage level (0.2V-4.8V)

| K42 | Origin Search Speed | Unit : 100pps<br>10pps<br>1pps<br>(Depends on K37) |
|---|---|---|
| This parameter sets the speed for Origin Search. | | Min 1<br>Max 32767 |

[Setting Example]

K37=3

K42.1=50 Set 5000pps to Origin Search Speed

| K43 | Acceleration for Origin Search / Manual Feed | Unit : $Kpps^2$ |
|---|---|---|
| This parameter sets the acceleration for Origin Search.<br>This is also used for the acceleration for Manual Feed. | | Min 1<br>Max 32767 |

[Setting Example]

K43.1=100 Set 100 $Kpps^2$ to Origin Search Acceleration

| K44 | Deceleration Ratio | Unit : % |
|---|---|---|
| This parameter sets the deceleration ratio relatibe to the acceleration in percentage. Acceleration and deceleration are the same when 100% is set.<br>This parameter is applied to all motion. When individual deceleration is needed, use CML command. | | Min: 10<br>Max: 500 |

[Setting Example]

K44.1=100 Set 100% to the Deceleration Ratio. (Deceleration is the same as acceleration)

| K45 | Origin Search Direction / Reverse Coordinates / Measure of Offset and Software limit | Unit : — |
|---|---|---|

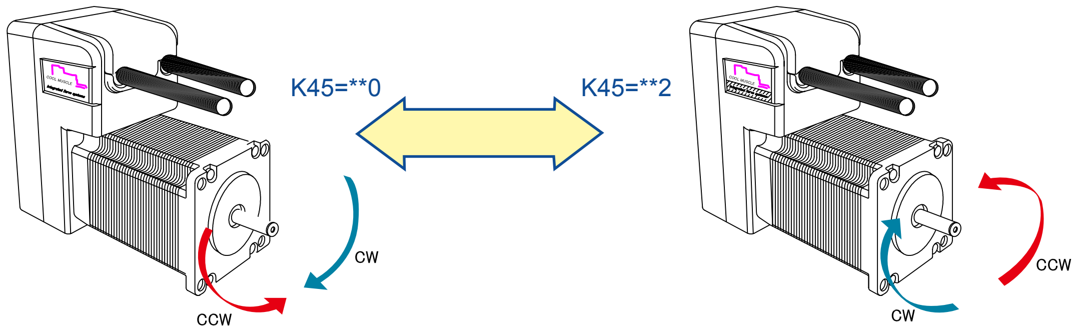
Set by using three digits and setting divides by each digit.

K45.1=□□□

| Digit | Functions | Value | Description |
|---|---|---|---|
| First Digit | Setting of Origin search Direction and Reverse Coordinates | 0 | CW direction |
| | | 1 | CCW direction |
| | | 2 | CW direction Reverse Coordinates |
| | | 3 | CCW direction Reverse Coordinates |
| Second Digit | Unit of offset by K48 | 0 | 100 pulse unit |
| | | 1 | 10 pulse unit |
| | | 2 | 1 pulse unit |
| Third Digit | Unit of software limit by K58, K59 | 0 | 100 pulse unit |
| | | 1 | 10 pulse unit |
| | | 2 | 1 pulse unit |

・First Digit･･Setting of Origin search Direction and Reverse Coordinates

This parameter sets the direction for the Origin Search and Reverse Coordinates. The CW direction usually corresponds to the positive in the coordinate system, but the Reverse Coordinates setting make the CCW direction correspond to the positive. This feature applies for the symmetric machinery without changing signs of all position date but just setting this parameter.




Caution for the Origin Search with an origin sensor

In case the origin search is executed when the origin sensor is ON, the motion to get out of the sensor signal region, to the opposite direction set by K45, is performed for a precise origin search.

(Ref: Section 5.3 Origin Search for the detailed information)

・Second Digit･･･Unit of offset by K48

To set the offset sensitively, set with second digit.

・Third Digit･･･Unit of software limit by K58,K59

To set the software limit sensitively, set with third digit.

[Setting Example]

K45.1=102

Set origin search direction to CW direction and Reverse Coordinates.

The unit of the offset set with K48 is set to 100 pulses.

The unit of software limit with K58 and K59 are set to 10 pulses.

| K46 | Origin Signal Source | Unit : — |
|---|---|---|

This parameter specifies the method for the origin search.

| # | Origin Signal Sources |
|---|---|
| 0 | Stopper Detection |
| 1 | Stopper Detection<br>(automatically starts an origin search when powered on) |
| 2 | Origin Sensor |
| 3 | Origin Sensor<br>(automatically starts an origin search when powered on) |
| 4 | Z Phase Signal |
| 5 | Z Phase Signal (automatically starts an origin search when powered on) |
| 6 | Origin Sensor & Z-Phase Signal |
| 7 | Origin Sensor & Z-Phase Signal (automatically starts an origin search when powered on) |

In case of stopper-detecting origin search, the origin search operation is completed when a pushing torque to a stopper reaches the set torque level.

In case of using an origin sensor, the origin search operation is completed when detecting the rising edge of signal from an external origin sensor.

Origin Search that starts automatically when powered ON can be set as well.

Z-phase signal is generated by the internal position sensor of Cool Muscle and output once per revolution.

Usage of Z-phase signal to detect an origin makes a precise origin search possible that always detects the same origin without an external origin sensor even in a rotative motion.

Furthermore, it is possible to detect an origin by using AND condition with an origin sensor signal. Therefore an origin search with higher repeatability accuracy is realized.

The following related parameters shall be set separately

| Stopper Detection | | Origin Sensor | |
|---|---|---|---|
| K42 | Origin Search Speed | K27 | Origin Sensor Signal |
| K43 | Acceleration for Origin Search | K42 | Origin Search Speed |
| K45 | Origin Search Direction | K43 | Acceleration for Origin Search |
| K47 | Stopper Detection Torque for Origin Search | K45 | Origin Search Direction |

## [Setting Example]

K46.1=3 Set the origin search by an origin sensor that starts automatically when powered ON for Origin Signal Source.

| K47 | Stopper-Detecting Torque for Origin Search | Unit : % |
|---|---|---|

This parameter sets the torque level to complete the stopper-detecting origin search.
The torque is relative to the rated torque of the motor in percentage.

| | |
|---|---|
| Min | 10 |
| Max | 150 |

When the acceleration is set too high, the torque required when starting motion reaches the set torque level and could incorrectly detect the origin. Please decrease K43 value.

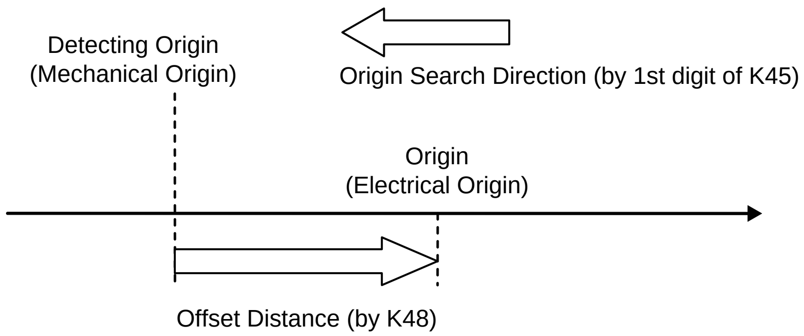
[Setting Example]

K47.1=30 Set the 30% of motor's rated torque for stopper detection torque level.

| K48 | Offset Distance Between Mechanical and Electrical Origins | Unit : 100 pulses<br>10 pulses<br>1 pulse<br>(Depends on 2nd digit of K45) |
|---|---|---|

This parameter sets the offset distance between the mechanical and electrical origins.
When it is set to 0, the motor stops at the mechanical origin. When it is set to other than 0, the motor automatically goes to the electrical origin set by this parameter. The speed moving from the mechanical origin to the electrical origin is the same as K42(return to origin speed)

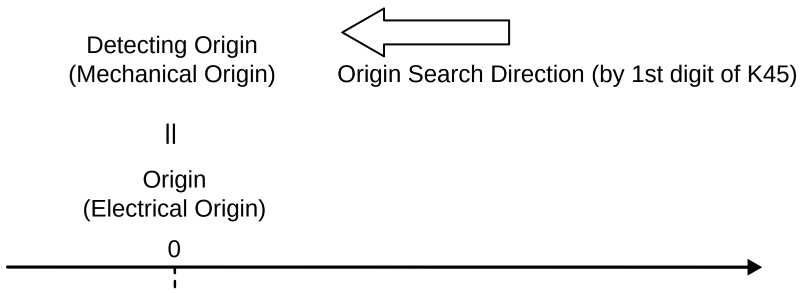
| | |
|---|---|
| Min | -32767 |
| Max | 32767 |



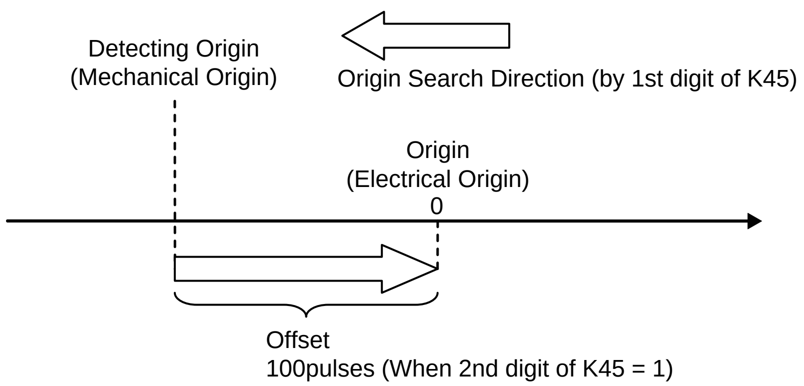

[Setting Example]

K48.1=0

The mechanical and electrical origins are the same




K48.1=10



| K49 | Speed for Manual Feed | Unit : 100pps<br>10pps<br>1pps<br>(Depends on K37) |
|---|---|---|
| This parameter sets the speed for manual feed.<br>Acceleration for manual feed can be set by K43. | | Min: 1<br>Max: 32767 |

[Setting Example]

K49.1=100 Set 100 x 100pps = 10000pps for the speed for manual feed.

| K50 | Feed Pulses for Manual Jog | Unit : pulses |
|---|---|---|
| This parameter sets the numbers of feed pulses for manual jog in the pulse unit. | | Min: 1<br>Max: 100 |

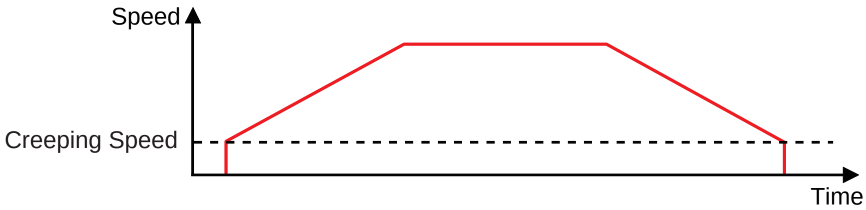
[Setting Example]

K50.1=10 Set 10 pulses for the numbers of feed pulses in manual jog operation.

| K51 | Creeping Speed | Unit : 100pps<br>10pps<br>1pps<br>(Depends on K37) |
|---|---|---|

This parameter sets creeping speed for the initial and terminal speed for a motion. Creeping speed is the speed from which motor starts to move and stop. The motor response or tact time will be adjusted finely by changing creeping speed. Setting creeping speed too high may cause the motor to vibrate.

| | |
|---|---|
| Min | 0 |
| Max | 1000 |




The value that can be used for this parameter depends on K37 (motor resolution).

| K37 | | | K51 |
|---|---|---|---|
| Value | | Resolution | Usable Values |
| 7 | 27 | 10000 | 2 and over |
| 8 | 28 | 25000 | 5 and over |
| 10 | 30 | 50000 | 10 and over |
| 48 | 68 | 6000 | 2 and over |
| 49 | 69 | 8000 | 2 and over |
| 50 | 70 | 12000 | 5 and over |

| K55 | In-Position Range | Unit : pulses |
|---|---|---|

This parameter sets the range for In-position in the pulse unit.

| Min | 1 |
|---|---|
| Max | 100 |

Different from motion completion signal, in-position is detected when the current position is within the set range against the target position.

When stopping the motor by a stop command, the stopped position is recognized as the target position, therefore In-position is detected within the set range against the current position.

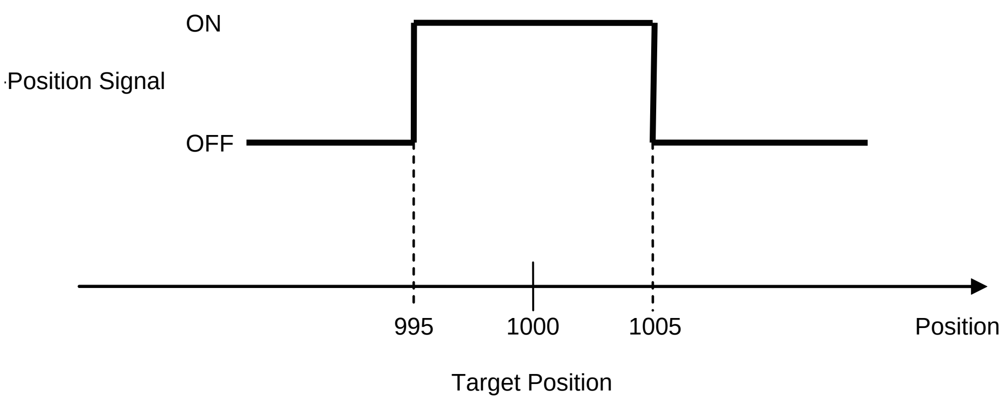
When recognized as In-position, In-position signal is ON and the motor status goes in Ux.n=8 (Ref: K23, n: Motor ID).

In-position signal can be output by assigning an output function (Ref: K34).

When the range is set to small, In-Position may not be detected and can not execute the next step in a program.
When the range is set too big, the resolution is too small and the speed is too slow, In-Position may be detected before reaching the target position.

## [Setting Example]

K55=5 In-Position signal range is set to 5 pulses. In-position signals will be sent out between 995 and 1005, when the target position is 1000.



| K56 | Position Error Overflow Threshold Level | Unit : 100 pulses |
|---|---|---|

This parameter sets a threshold value for the position error overflow in the 100-pulse units. When the deviation between the current position and command position exceeds the threshold level, the motor outputs an alarm and goes into motor free state.

| | |
|---|---|
| Min | 1 |
| Max | 32767 |

### [Setting Example]

K56.1=50

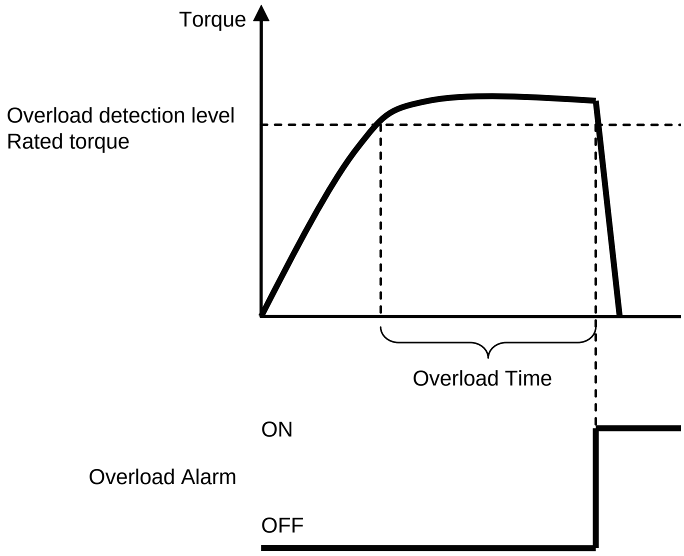
Set 50K pulses to Position Error Overflow Threshold Level.

When the deviation between the current position and command position exceeds 5000 pulses. Motor goes into Position Error Overflow alarm (Ux=1) and stops in motor free state.

| K57 | Overload Detection Time | Unit : msec |
|---|---|---|

When the overload condition for motor continues more than set time (msec), the overload alarm is detected. Then motor outputs overload alarm (Ux=4) and stops in motor free state.

| | |
|---|---|
| Min | 100 |
| Max | 5000 |




## [Setting Example]

K57.1=3000

Motor outputs overload alarm when the overload condition continues more than 3000msec, and stops in motor free state.

| K58 | Software Limit (+) | Unit : 100 pulses<br>10 pulses<br>1 pulse<br>(Depends on 3rd digit of K45) |
|---|---|---|

K58 sets the software limit in the positive direction, to prevent the motion over the set position.
There is no software limit available when 0 is set.
This function provides the safety stop and cost reduction without an external hardware as limit sensor.

| Value | Description |
|---|---|
| 1 ~ 32767 | + side software limit |
| 0 | No software limit |

| K59 | Software Limit (-) | Unit : 100 pulses<br>10 pulses<br>1 pulse<br>(Depends on 3rd digit of K45) |
|---|---|---|

K59 sets the software limit in negative direction in the 100-pulse units, to prevent the motion over the set position.
There is no software limit available when 0 is set.

| Value | Description |
|---|---|
| -32767 ~ -1 | - side software limit |
| 0 | No software limit |

[Setting Example]

K58.1=200 Set 200 pulse to + direction software limit. (When 3rd digit of K45 = 2)

K59.1=0 Set no software limit in - direction.

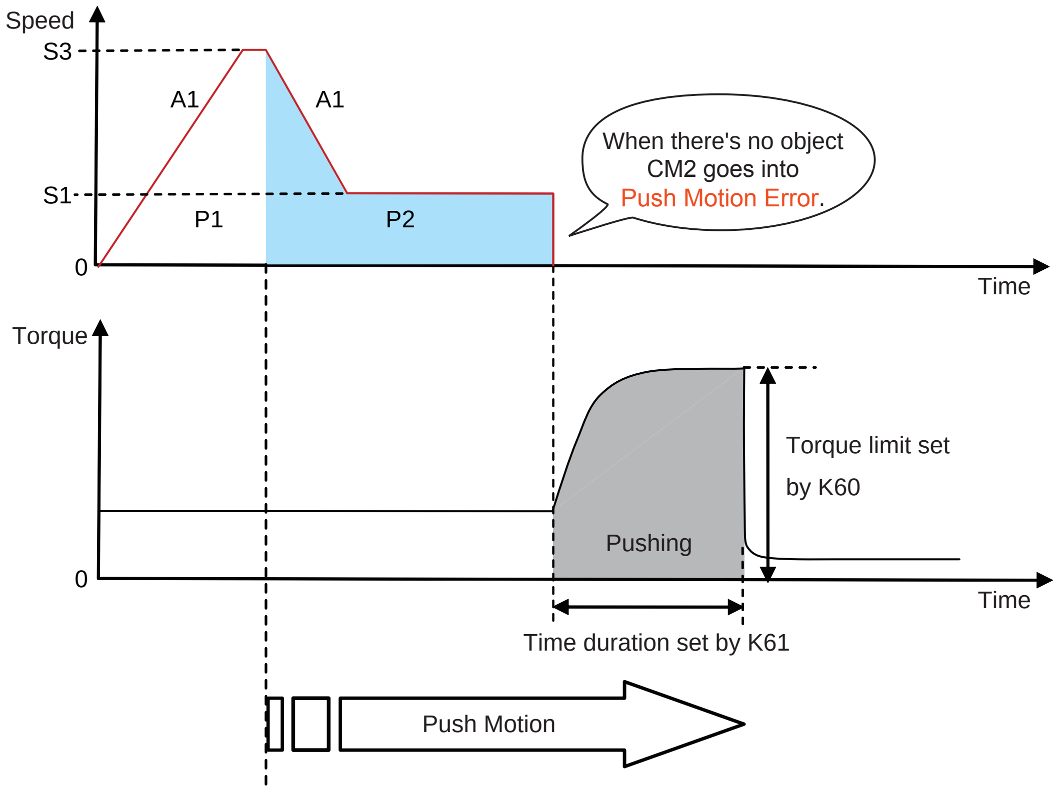
| K60 | Push Motion Torque Level | Unit : % |
|---|---|---|
| This parameter sets the torque level for the Push Motion, that is relative to the motor's rated torque in percentage.<br>When the odd number is set the push motion error will not occur. | | Min 10<br>Max 100 |

| K61 | Push Motion Holding Time | Unit : msec |
|---|---|---|
| This parameter sets the holding time for the Push Motion.<br>The endless Push Motion can be applied by setting K61=0 | | Min 0<br>Max 30000 |

[Setting Example]

K60.1 = 50 Set 50% of rated torque to Push Motion Torque Level

K61.1 = 5000 Motor keeps pushing an object for 5000msec



| K62 | Ladder Logic Bank No. Executed when Powered ON | Unit : — |
|---|---|---|
| Set a Ladder Logic Bank No. that is executed automatically when powered ON.<br>No Ladder Logic Bank will be executed when 0 is set. | | Min: 0<br>Max: 30 |

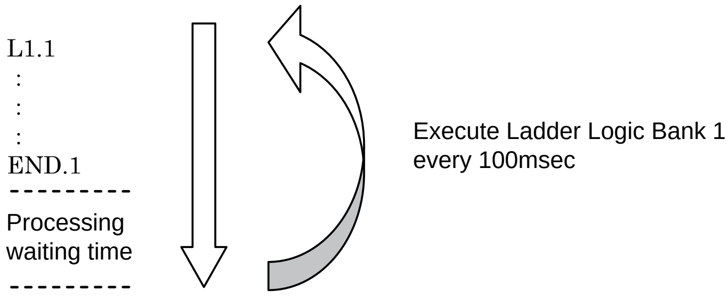
[Setting Example]

K62.1=2 Ladder Logic Bank 2 is executed automatically when powered ON.
(The same as [L2.1)

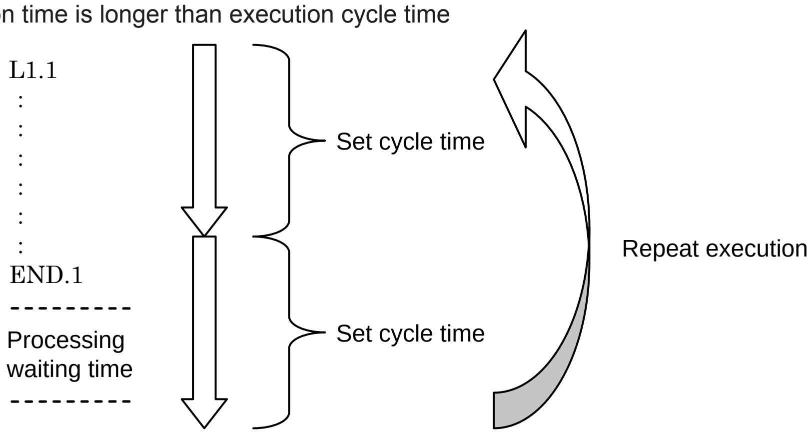
| K63 | Ladder Logic Bank execution cycle time | Unit : msec |
|---|---|---|
| Sets the execution cycle time for Ladder Logic Bank.<br>When all processing in a Ladder Logic Bank is finished within a set cycle time, the execution of processing is forced to wait until the next cycle. When all processing is not finished within a set cycle time, remaining processing is carried over to the next cycle.<br>When K63=0, a Ladder Logic Bank is not executed. | | Min: 0<br>Max: 30000 |

[Setting Example]

K63=100 Execute Ladder Logic Bank every 100msec




When execution time is longer than execution cycle time



| K64 | Status LED Setting | Unit : — |
|---|---|---|

This parameter sets either the status LED is activated or inactivated.

The default value is 0 (Activated).

When setting 1 (Inactivated), the LED will be off all the time including an alarm status.

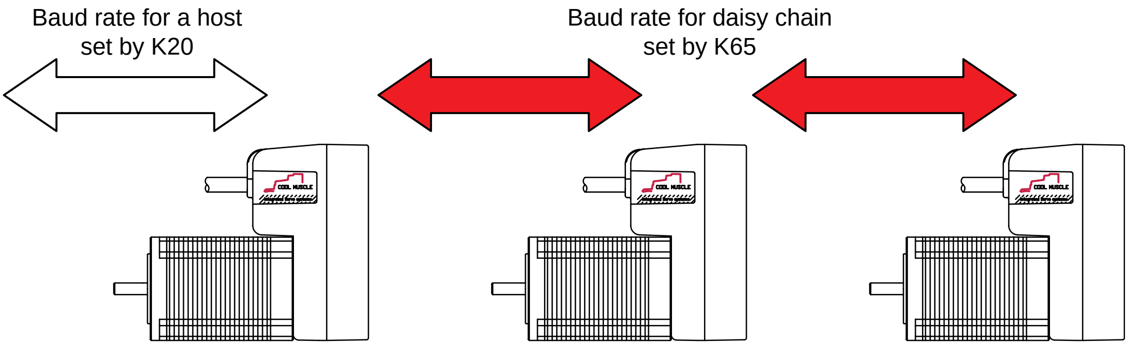
CM2 User's Guide shall be referred to for the LED activated pattern.

| # | Description |
|---|---|
| 0 | Status LED Activated |
| 1 | Status LED Inactivated |

| K65 | Baud Rate Between Slave Motors | Unit : — |
|---|---|---|

This parameter sets the baud rate between slave motors in the daisy chain operation.
When changing the setting of ID1, the settings of other motors will be changed automatically. When changing the setting of slave motors except ID1, the baud rate between slave motors will not be changed.

| # | Baud rate |
|---|---|
| 0 | 38.4Kbps |
| 1 | 9.6Kbps |
| 2 | 19.2Kbps |
| 3 | 57.6Kbps |
| 4 | 115.2 Kbps |
| 5 | 230.4 Kbps |




When setting a high baud rate, the synchronization will be better but the communication will become sensitive against the noise.

## [Setting Example]

K65.1=5 Set "230.4 Kbps" to the baud rate between slave motors.

| K68 | Motor Free when Powered ON | Unit : — |
|---|---|---|

This parameter sets either servo ON or motor free when powered ON.

| Value | Set content |
|---|---|
| 0 | Motor free when powered ON |
| 1 | Servo ON when powered ON |

[Setting Example]

K68.1=1 Servo on when powered on

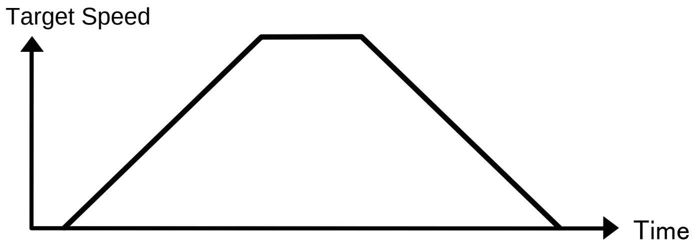
| K69 | S-Curve gain | Unit : — |
|---|---|---|
| This parameter sets the S-curve gain in positioning.<br>By setting S-curve, the form of target speed for acceleration and deceleration periods will be S-shaped according to its gain. Therefore it may effect a smoother positioning or vibration reduction.<br>When 0, motor makes a trapezoidal motion. | | |

| | |
|---|---|
| Min | 0 |
| Max | 1024 |

[Setting Example]

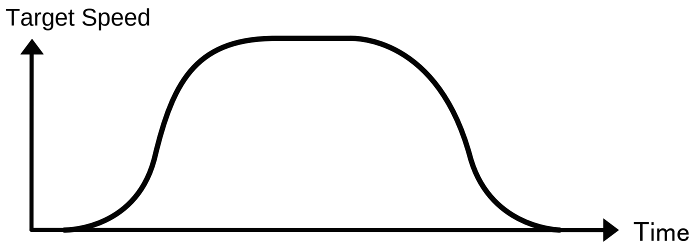
K69.1=0

Trapezoidal Motion




K69.1=1024

S-curve Motion



| K70 | Delimiter | Unit : — |
|---|---|---|
| This parameter sets the delimiter type at the end of replied data. | | |

| Value | Delimiter |
|---|---|
| 0 | CR |
| 1 | CRLF |

[Setting Example]

K70.1=1 Set “CRLF” to the delimiter.

| K71 | External Encoder Type | Unit : — |
|---|---|---|

Set the external encoder type.

| Value | Set content |
|---|---|
| 0 | No external encoder |
| 1 | A-phase index |
| 2 | A-phase index, B-phase rotation direction |
| 3 | A-phase & B-phase index |
| 4 | A-phase & B-phase feedback |
| 5 | A-phase pulse counting |
| 6 | A-phase pulse measuring B-phase rotation direction |
| 7 | A-phase & B-phase pulse counting |

Index :

Motor keeps rotating until the numbers of pulses from an external encoder reaches the specified numbers of palses.(It will not adjust the overrun pulses)

It will be useful for the motion winds in specified amount in one direction without loosening as used in a winding machine.

Feedback :

By the feedback pulses from the external encoder equipped for the control target, the whole system can be controlled as a full closed-loop system.

Pulse Counting :

Input the pulses from an external encoder to Cool Muscle and only count the numbers of pulses.

This feature is useful for the control according to the amount of movement or speed of the control target.

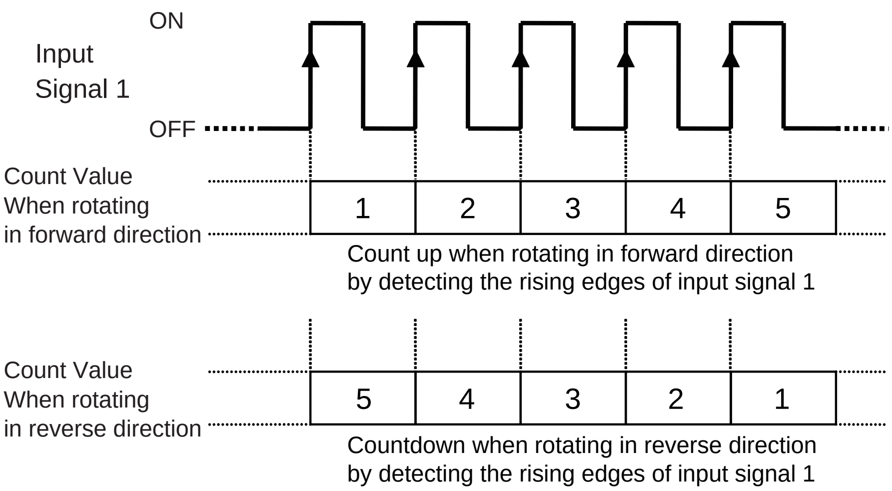
Count Timing for the external encoder depends on input type and is shown as the diagram below.

Note: The input logic for the input voltage can be set by the parameter K26.

[A-Phase Index]

Input encoder pulse to input port 1.

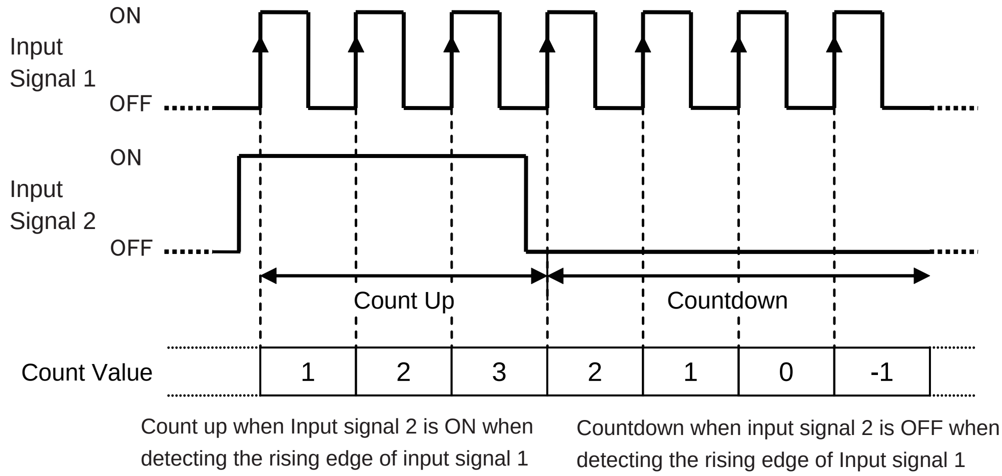
Count the pulse when detecting the rising edge. Count up when rotating in forward direction and countdown when rotating in reverse direction.



[A-Phase Index, B-Phase Direction]

Input encoder pulse to input port 1 and direction pulse to input port 2.

Count the pulse by the rising edge of input signal 1. Count up when input signal 2 is ON when detecting the rising edge and countdown when input signal 2 is OFF.

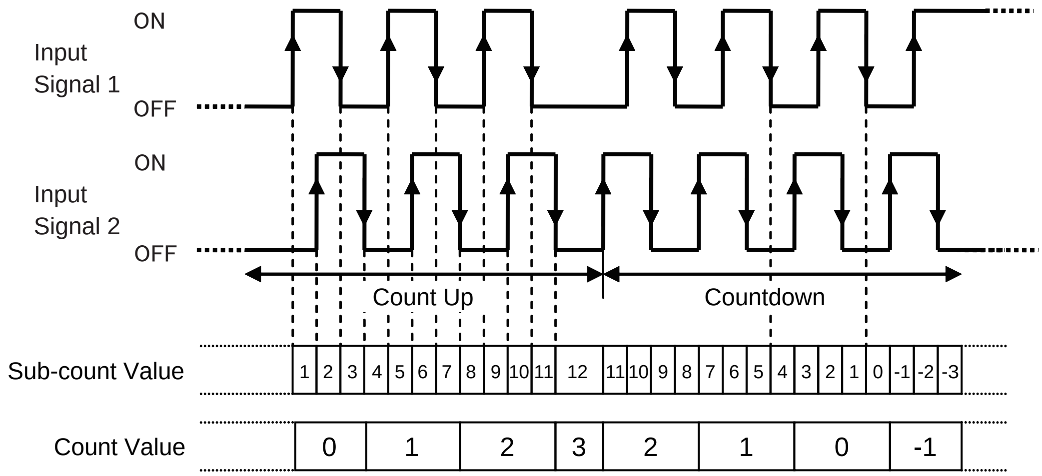



[A-Phase & B-Phase Phase Index]

Count the pulse automatically recognizing count up or countdown by input signals that phases are shifted 90 degrees to each input port 1 and input port 2 as the timing shown in the diagram below.

In this mode, sub-count value will be 4 for 1 cycle of input signal because the rising and falling edges of 2 phase signals for input are counted.

Encoder pulse count value will be 1/4 integral part of sub-count value.



| K72 | External Encoder Resolution | Unit : ppr |
|---|---|---|
| This parameter sets the resolution for the external encoder. | | Min: 0<br>Max: 32767 |

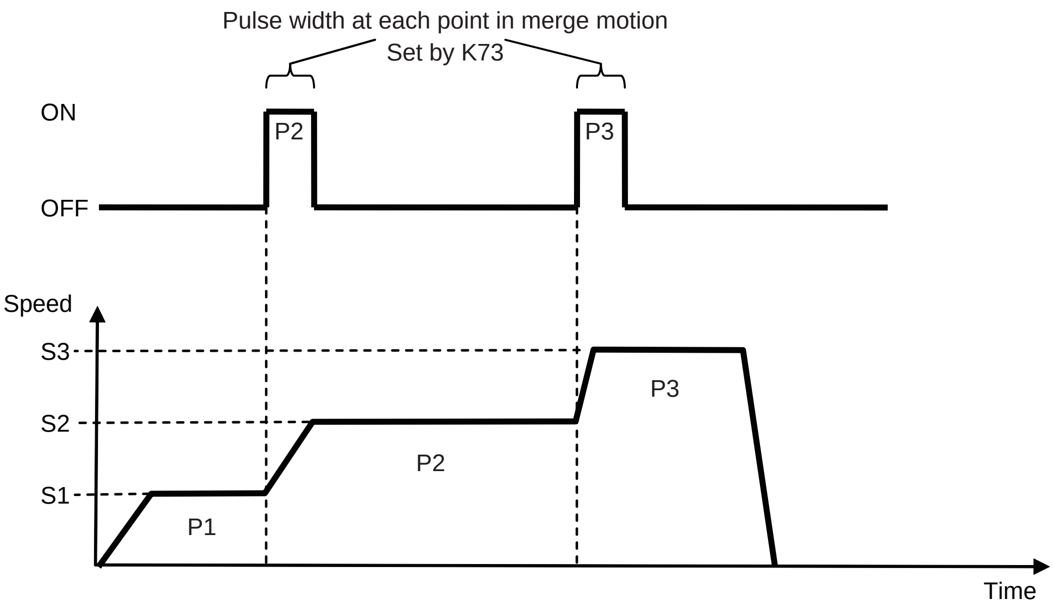
[Setting Example]

K72.1=1000 Set 1000ppr to the external encoder resolution

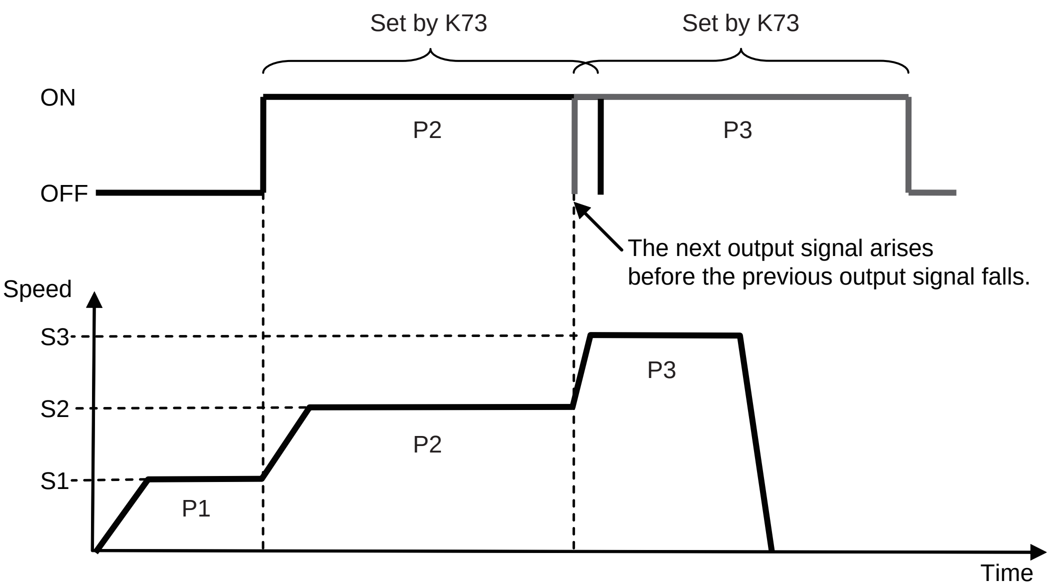
| K73 | Output Pulse Width at Passing Point in Merge Motion | Unit : msec |
|---|---|---|

This parameter sets the pulse width in msecs that is sent out when the motor passes each point in merge motion. Pulse signals are sent out only at the passing points but not at the starting and end points.

| | |
|---|---|
| Min | 0 |
| Max | 1000 |




When the pulse width is set too big, the motor can reach the next passing point and the signal arises before the signal falls down. This results in one signal covering multiple points. When this happen, please reset the pulse width smaller.




[Setting Example]

K73.1=100 Set 100msec to the pulse width at passing point in merge motion

| K74 | Torque Control P Gain | Unit : — |
|---|---|---|
| Set proportional gain for external torque sensor feedback. | Min 0<br>Max 1000 | |

| K75 | Torque Control I Gain | Unit : — |
|---|---|---|
| Set integral gain for external torque sensor feedback. | Min 0<br>Max 500 | |

| K76 | Input Offset for Torque Sensor | Unit : 0.01V |
|---|---|---|

Set the offset value of an external torque sensor input for torque feedback control.
The offset value is the output voltage of external torque sensor when torque sensor is 0[N.m].

| | |
|---|---|
| Min | 0 |
| Max | 500 |

| K77 | Input Range for Torque Sensor | Unit : 0.01V |
|---|---|---|

Set the input range of an external torque sensor for torque feedback control.
In the torque feedback control, the motor output can be controlled in accordance with K74(P gain) and K75(I gain), where the feedback data from external torque sensor equipped for the control target track the torque command value specified in the range of 0~±100 by Variable 15.
This parameter sets the voltage level in the unit of 0.01V. The value is the output voltage of torque sensor when the torque command value is 100.

| | |
|---|---|
| Min | -1000 |
| Max | 1000 |

[Setting Example]

K76.1=250

K77.1=200

Output 1[V] for 0.5[N.m], connected to a torque sensor with offset voltage 2.5[V].

Since the offset voltage is 2.5[V], set K76=250.

When set command torque 100 as 1.0[N.m]

Since the torque sensor output when 10[N.m] is 2[V], set K77=200.

| K78 | Input Address for Modbus Host Communication | Unit : — |
|---|---|---|

Set the Modbus input address for the host communication.

Set K78=0 for relative address. When K78=-1 is set, this function is not activated.

| | |
|---|---|
| Min | -1 |
| Max | 32767 |

| K79 | Input Address for Modbus Slave Communication | Unit : — |
|---|---|---|

Set the Modbus input address for the slave communication.

Set K79=0 for relative address. When K78=-1 is set, this function is not activated.

| | |
|---|---|
| Min | -1 |
| Max | 32767 |

| K80 | Output Address for Modbus Slave Communication | Unit : — |
|---|---|---|

Set the Modbus output address for the slave communication.

Set K80=0 for relative address. When K80=-1 is set, this function is not activated.

| | |
|---|---|
| Min | -1 |
| Max | 32767 |

| K81 | Slave Address | Unit : — |
|---|---|---|

Set the equipment ID number for Modbus or RS-485 communication.

Set the value from -255 to -1 in query mode, from 1 to 255 in event mode for RS-485 communication.

| | |
|---|---|
| Min | -255 |
| Max | 255 |

For Modbus communication, set the value from 1 to 255 in the following way.

Set Cool Muscle's ID No. when Cool Muscle is receiving side, and set the equipment ID No. of receiving side when Cool Muscle is transmitting side.

⚠ This parameter changes the communication method. Please change K81 after adjusting, K78, K79 and K80.

## [Setting Example]

Set the parameters as shown in the diagram below.

| Communication Mode | Parameters | | | |
|---|---|---|---|---|
| | K78 | K79 | K80 | K81 |
| RS-232C | × | × | × | 0 (zero) |
| RS-485 | -1 | -1 | -1 | Query Mode : -255～-1<br>Event Mode : 1～255 |
| Modbus | Host Communication<br>Input :<br>0～32767 | Slave Communication<br>Input :<br>0～32767 | Slave Communication<br>Output :<br>0～32767 | 1～255 |

⚠ CML command communication during RS-232C can not be established under RS-485 communication mode. When canceling RS-485 communication, temporary CML command communication will be enabled by 2 times transferring " {equipment ID number (absolute value) " command to Cool Muscle after setting a proper baud rate. RS-485 communication will be canceled by setting K81=0.

Example of transferring command : K81= 123 "{123"

K81= -123 "{123"

| K82 | Parity | Unit : — |
|---|---|---|

Set the parity when transferring data.

| Value | Description |
|---|---|
| 0 | None |
| 1 | Even |
| 2 | Odd |

Chapter **4**

# Sample Program

In this section, we will show some program examples by CML that is explained in the section 2.
This section is comprehensive to learn basic to advanced CML.
* Please use Cool Muscle 2 alone since those are sample programs.

## 4.1. Various PTP motion

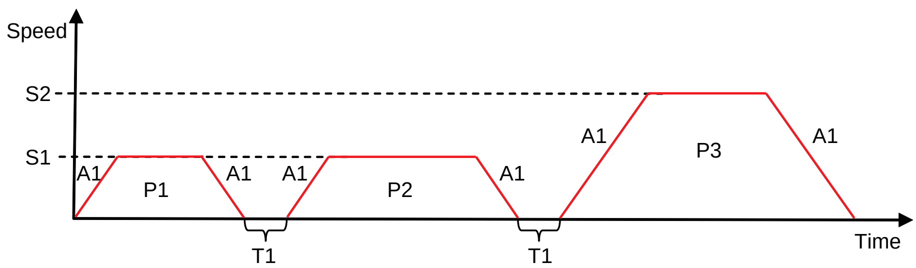
Using one motor, basic single axis point to point motion (one point on one straight line to another point) is executed.

### 4.1.1. Basic PTP motion

| | | |
|---|---|---|
| P1.1=20000 | Position Data | Motion Data Definition |
| P2.1=72000 | | |
| P3.1=120000 | | |
| S1.1=100 | Speed Data | |
| S2.1=200 | | |
| S3.1=300 | | |
| A1.1=100 | Acceleration Data | |
| A2.1=200 | | |
| T1.1=500 | … Timer Data | |

| | | |
|---|---|---|
| B1.1 | …Beginning of Program Bank 1 | Program Bank |
| S1.1, A1.1, P1.1 | …Move to P1 with speed S1 and acceleration A1 | |
| T1.1 | …Timer for T1, motor no action | |
| P2.1 | …with the same acceleration and speed as in line 2 move to P2 | |
| T1.1 | | |
| S2.1, P3.1 | …Change speed to S2 and move to P3 | |
| END | …End of Program Bank | |

One line represents one motion. When speed and acceleration are not specified, the previously used speed and acceleration are applied. In the example above, the same acceleration A1 is used for the entire program and the same speed S1 is applied until the motor reaches P2 (line 4). In line 6, the speed changes to S2 and motor moves to P3.



## 4.1.2. Merge Motion

```
B2.1
S1.1, A1.1, P1.1, S2.1, P2.1, S3.1, A2.1, P3.1
```

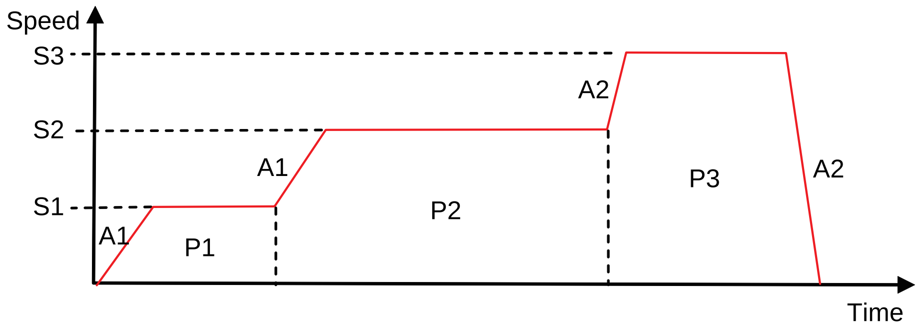
…Beginning of Program Bank 2

Motion data are same as 4.1.1.

Motion before P1 | Motion before P2 | Motion before P3

When multiple P commands are used in a single line, the motor does not stop at each position that is called merge motion. In Merge Motion, A and S commands can be specified, changing speeds and accelerations at passing points.

In the example program above, the motor passes P1 and P2 and moves to the final destination.




But when a movement direction is turn over, Merge Motion is removed and it performs normal motion.

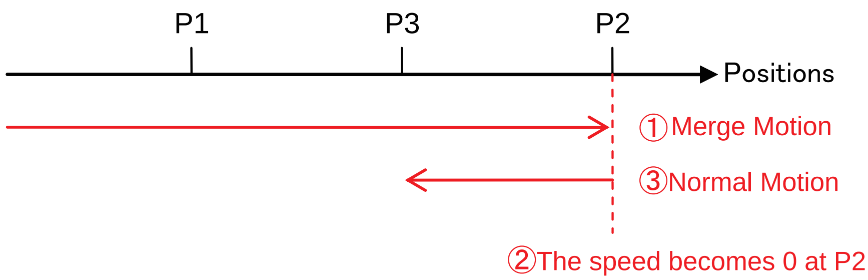
```
P1.1=10000
P2.1=30000
P3.1=20000
```

Position Data

```
B2.1
S1.1, A1.1, P1.1, S2.1, P2.1, S3.1, A2.1, P3.1
```

…Beginning of Program Bank 2

Motion before P1 | Motion before P2 | Motion before P3



### 4.1.3. PTP motion with different Accelerations and Decelerations

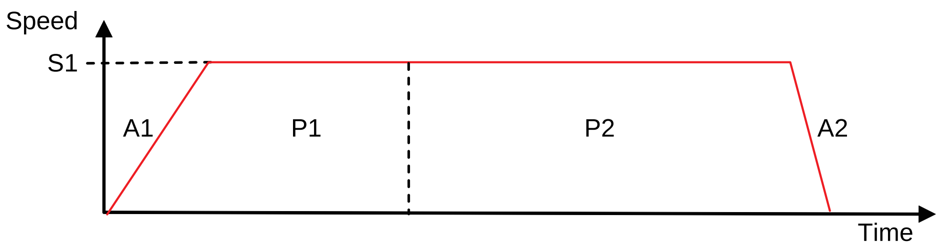
Related parameter
K44

```
B3.1                              ...Beginning of Program Bank 3
S1.1, A1.1, P1.1, A2.1, P2.1      Motion data are same in the 4.1.1.
```

Motion before P1 | Motion before P2

Acceleration and Speed remain the same unless specified otherwise.

When multiple A commands are used in a single line, you can set accelerations and decelerations independently. As the chart below shows the motor reaches the final destination with a slow acceleration and a quick deceleration. Another way to set deceleration separately is to use parameter K44. (by a percentage of acceleration.)




### 4.1.4. Push Motion

Related parameter
K60, K61

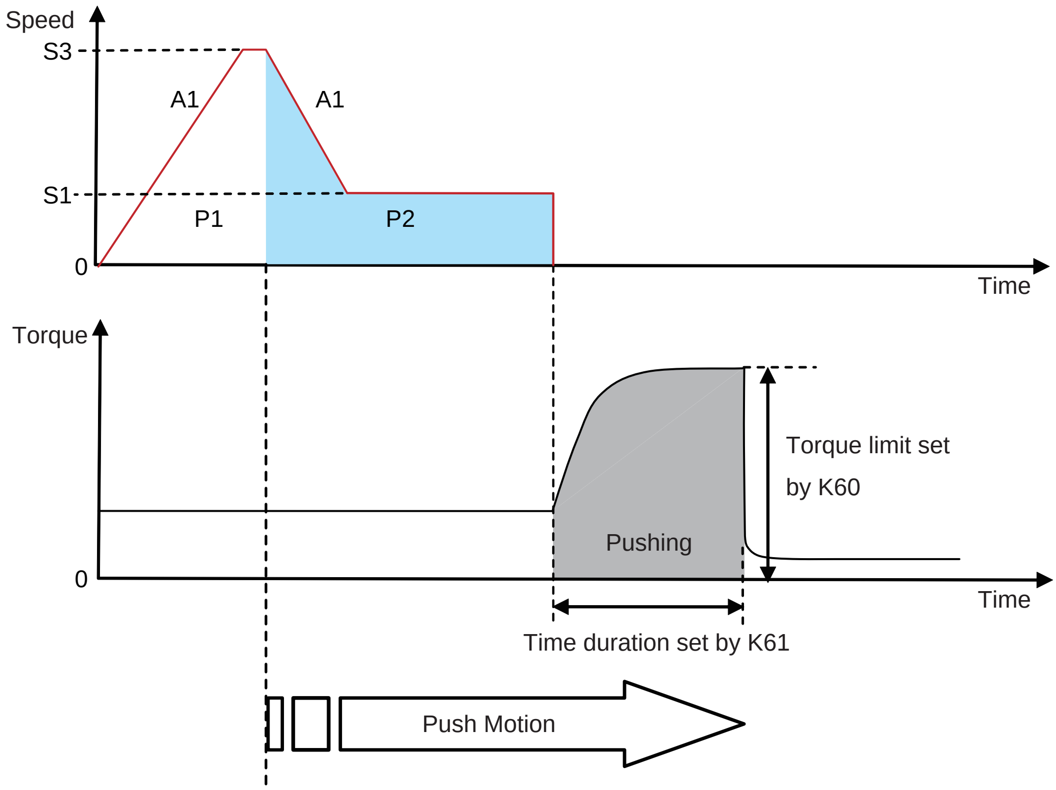
```
B4.1                              ...Beginning of Program Bank 4
S3.1, A1.1, P1.1, S1.1, Q2.1      Motion data are same as 4.1.1.
```

Motion before P1 | Motion before P2

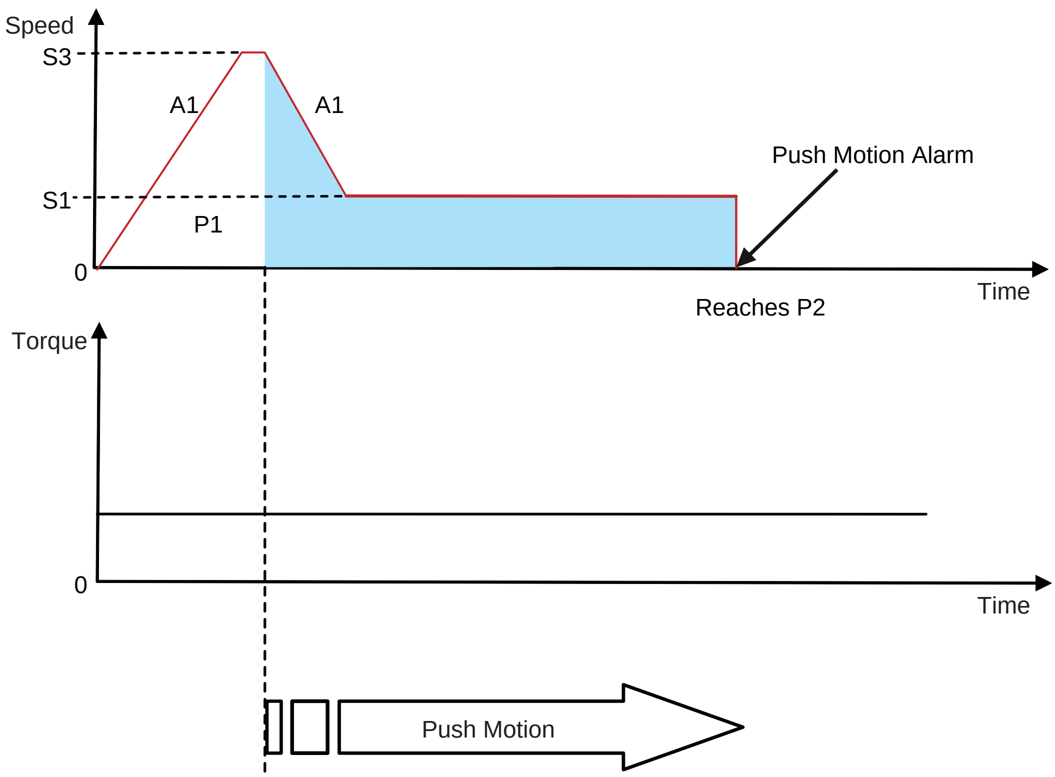
Using Q command instead of P command, it performs Push Motion within the torque limit designed by parameters.

The CML program above shows the motion that the motor changes the speed to S1 at P1 and start performing Push Motion toward P2.

Torque limit and Push Motion duration time need to be defined by Parameter K60 and 61. The following charts show relationship between the motion and torque.




During the Push Motion, Cool Muscle 2 goes into an alarm state (Ux=256) being in Push Motion when reaches a target position by the reason that pushing object does not exist or push torque is too high.



# 4.2. Various Processing

More complex CML program flows are introduced and described in this section.

## 4.2.1. Loop Processing

```
P1.1=20000
P2.1=72000
P3.1=120000
P4.1=100000
S1.1=100
A1.1=100
```

Position Data

…Speed Data

…Acceleration Data

Data definition.

```
B1.1
X0.1
 S1.1,A1.1,P1.1
 X2.1
  P3.1
  P4.1
 X.1-
 P2.1
X.1-
```

Repeat 2 times between X2.1 and X.1-

Loop(X2.1) within loop(X0.1)

Infinite Loop between X0.1 and X.1-

The lines between [ X loop count . Motor ID] command and [X . Motor ID - ] command are repeated the number of times that is specified by Loop Count. By using command X between loops, it performs multiple loops up to 10 classes.

## 4.2.2. Basic Branch Processing

By specifying branching condition, different processes can be executed by conditions true or false.

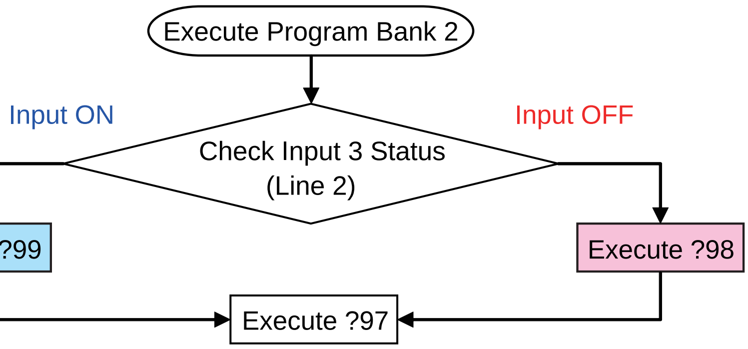
When defining a branching processing as below, describe a condition ( I or V command ), true condition and false condition dividing with comma.

[ Format ] Branching Condition, True Condition, False Condition

When using I command, execute the conditional branching by the specified input status.

| | | |
|---|---|---|
| B2.1 | | Program Bank |
| I3.1, ?99.1, ?98.1 | …Conditional branch Processing | |
| ?97.1 | …Merge back to normal command processing | |
| END | | |

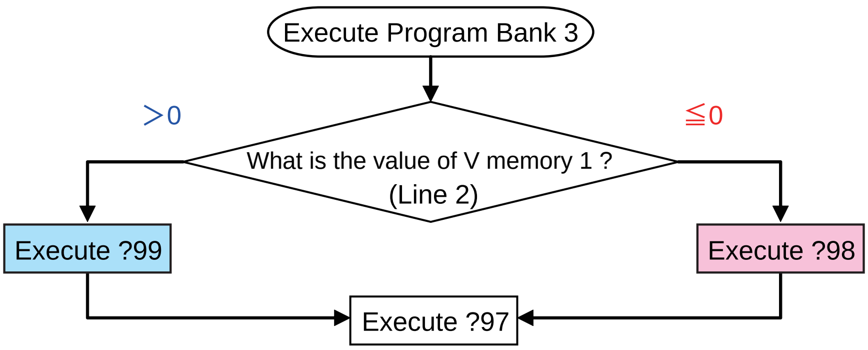
The flow of above CML program is as shown in below.




When using only V command, the branch processing depends the specified V data is larger than 0 or not. When larger than 0, execute the true condition otherwise false condition.

| | | |
|---|---|---|
| V1.1="Sx" | Data definition. Define the current speed to V1 | |
| B3.1 | | Program Bank |
| V1.1, ?99.1, ?98.1 | …Conditional Branching Processing | |
| ?97.1 | …Return from Conditional Branching | |
| END | | |

In the above program bank, execute the true condition when the current speed > 0, and the false condition when the current speed < 0. The processing shall be shown in below.



### 4.2.3. Branch Processing using Logic Operator

Using a logic operator, more complicated branch processing than the programs in section 4.2.2 is possible (Ref Section 6.7 for Mathematical Operator, Ref Section 6.8 for Logic Operator).

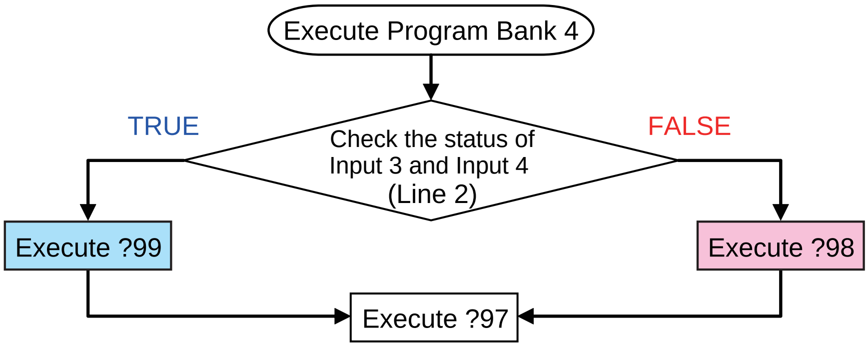
When executing branching processing, two conditions (I or V command), mathematical or logic operator between two conditions, true condition and false condition dividing by comma shall be described.

[ Format ] Branching Condition 1, Operation, Branching Condition 2, True Condition, False Condition.

A CML program example using the I command and its flow is as below.

```
B4.1
I3.1 && I4.1, ?99.1, ?98.1   …Conditional Branching with Logic Operator
?97.1                        …Merge back to a normal command processing.
```

Program Bank




The criteria of condition of 2 input status and Logic Operator is as shown in below.

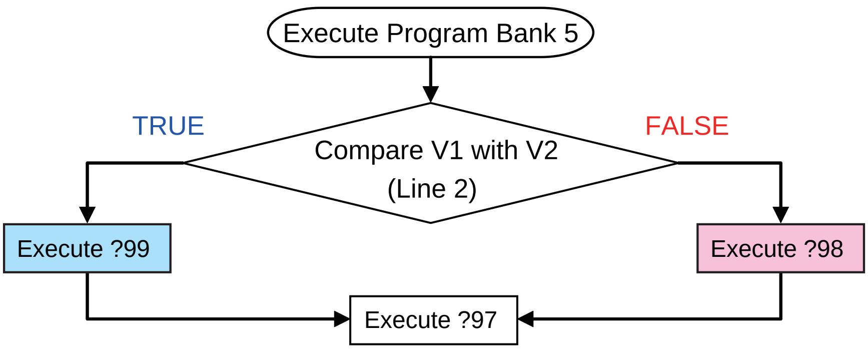
| Logic Operator && (AND) | | INPUT 2 | |
|---|---|---|---|
| | | ON | OFF |
| INPUT 1 | ON | TRUE | FALSE |
| | OFF | FALSE | FALSE |

| Logic Operator || (OR) | | INPUT 2 | |
|---|---|---|---|
| | | ON | OFF |
| INPUT 1 | ON | TRUE | TRUE |
| | OFF | TRUE | FALSE |

A CML program example using the V command as a condition and its flow is as below.

```
B5.1
V1.1> V2.1, ?99.1, ?98.1   …Conditional Branching with Logic Operator
?97.1                      …Merge back to normal command processing.
END
```

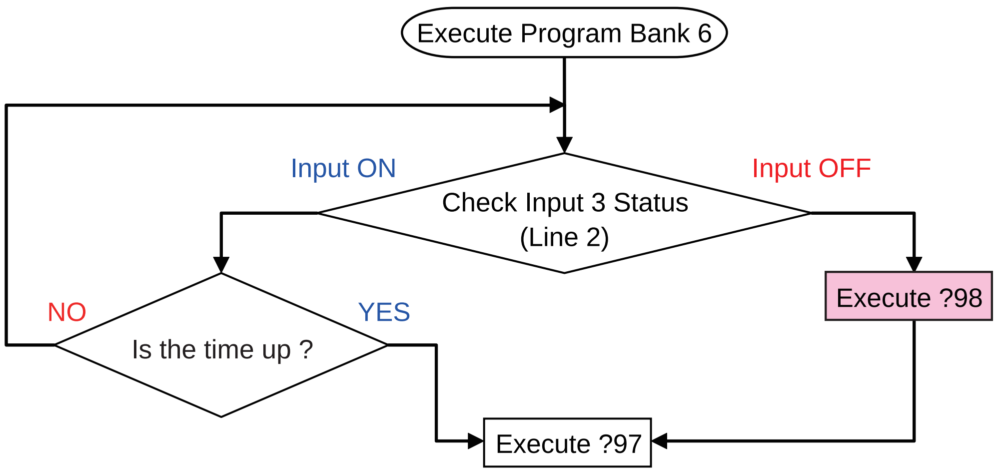
Program Bank



## 4.2.4. Branch Processing with Wait function

| Program Bank | |
|---|---|
| B6.1 | |
| I3.1, W1, ?98.1 | …Branching with timer function |
| ?97.1 | …Merge back to normal command processing. |

The W command can be used for branching with wait function (line 2). The motor waits for the time specified by the timer memory to pass and keeps on monitoring the status of the specified input for that duration. When the time is up, the motor finishes the branch processing and executes the next command line. The flow of the CML program above is as below.

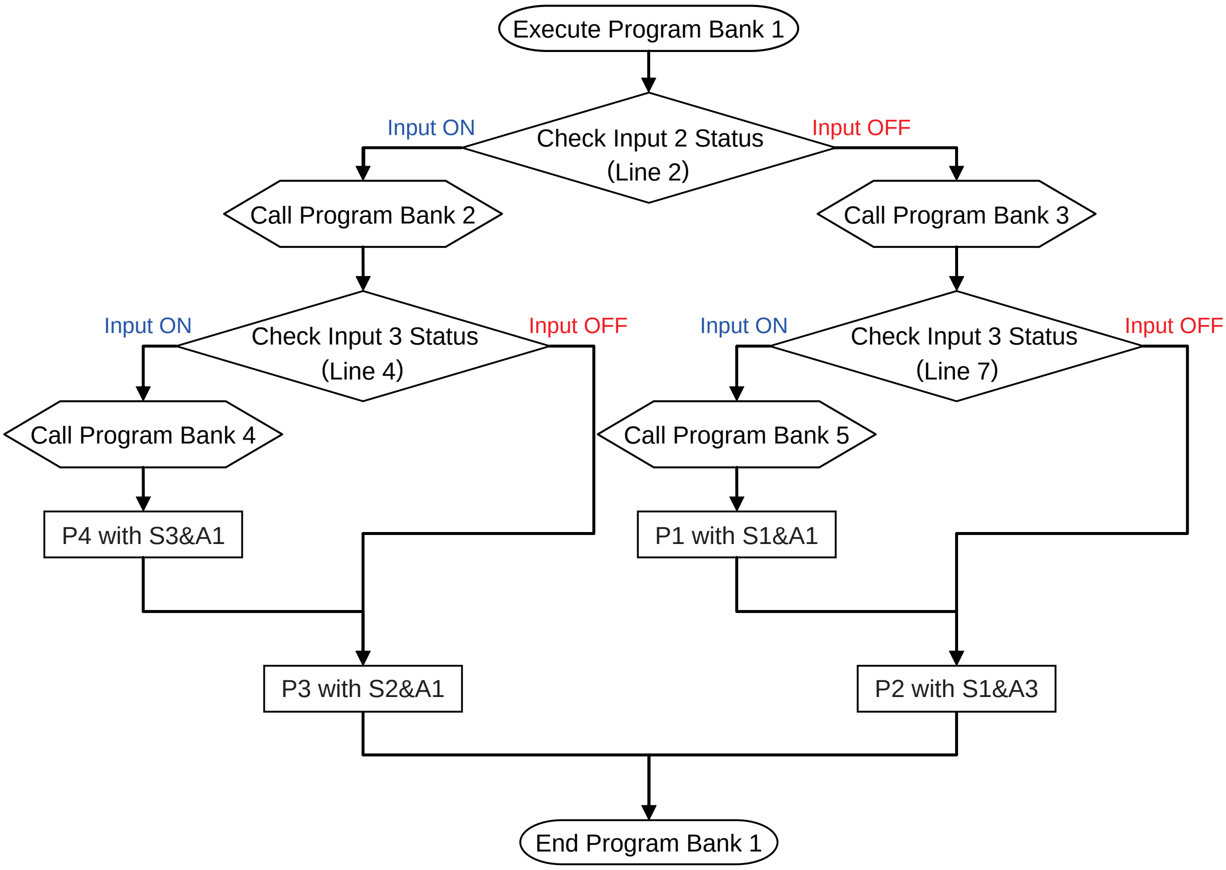


## 4.2.5. Nesting

By using C command (call), Program Bank goes deeper and its depth is called “Nesting”. Depending on how to compose of a program, the programming that a hierarchy becomes deeper is possible.
The maximum nesting level for COOL MUSCLE 2's programming is up to 10.

By combining nesting and branching, a specified program bank can be executed according to the specified input status.
The CML program below shows how one of the 4 program banks is executed according to the status of Input 2 and 3.

| | | |
|---|---|---|
| B1.1 | | Program Bank 1 |
| I2.1, C2.1, C3.1 | …Conditional branch Processing | |
| B2.1 | | Program Bank 2 |
| I3.1, C4.1, T0.1 | …Conditional branch Processing | |
| S2.1, A1.1, P3.1 | | |
| B3.1 | | Program Bank 3 |
| I3.1, C5.1, T0.1 | …Conditional branch Processing | |
| S1.1, A3.1, P2.1 | | |
| B4.1 | Program Bank 4 | |
| S3.1, A1.1, P4.1 | | |
| B5.1 | Program Bank 5 | |
| S1.1, A1.1, P1.1 | | |



# 4.3. Controlling Multiple Motors

The CML program examples introduced in the section 4.1, 4.2 use a single motor. CML programs using multiple motors are introduced in this section.
To control multiple motors, various data and commands have to be defined for each motor.

## 4.3.1. Synchronized motion in Two Dimensions

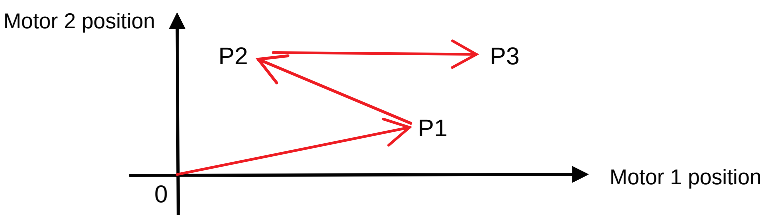
| Program | Description |
|---|---|
| P1.1=2500<br>P2.1=1000<br>P3.1=3000<br>S1.1=200<br>A1.1=100 | Motor 1 Data definitions |
| P1.2=1000<br>P2.2=2000<br>S1.2=100<br>A1.2=50 | Motor 2 Data definitions |

| Program | Description | |
|---|---|---|
| B1.1 | | Program Bank 1 |
| A1.1, S1.1, P1.1, A1.2, S1.2, P1.2 | …Motor 1 and Motor 2 move to P1 at the same time | |
| P2.1, P2.2 | …Motor 1 and Motor 2 move to P2 at the same time | |
| P3.1 | …Motor 1 moves to P2. | |

Program description
Line 2 : Motor 1 and Motor 2 start to move at the same time. Motor 1 moves to position 2500 with the speed of 200 and acceleration of 100. Motor 2 moves to position 1000 with the speed of 100 and acceleration of 50.

Line 3 : Once both Motor 1 and Motor 2 complete the motion defined by line 2, Motor 1 and Motor 2 start to move at the same time. Motor 1 moves to position 1000 with the same speed and acceleration as in the previous motion. Motor 2 moves to position 2000 with the same speed and acceleration as in the previous motion.
The line 3 is not executed until both Motor 1 and Motor 2 complete the current motion (line 2). One motor waits until the motion of another is completed.

Line 4 : When Motor 1 and Motor 2 complete the motion defined by line 3 in Bank 1, only Motor 1 moves to position 3000.



## 4.3.2. Non-synchronized motion in Two Dimensions

In the previous CML program example, either motor does not initiate the next motion until both motors complete the current motion. In this CML program, both motors independently initiate their own motion without waiting for the completion of motion each other.

| | | |
|---|---|---|
| B2.1 | …Use the same data as in section 4.3.1 | Program Bank 2 |
| A1.1, S1.1, P1.1, A1.2, S1.2, P1.2 | …Motor 1 and Motor 2 move to P2 at the same time. | |
| P2.1, Y2.2 | …Substitute Y command for P command to Motor 2 | |
| P3.1 | …Motor 1 moves without waiting for Motor 2 | |

Description of the program above

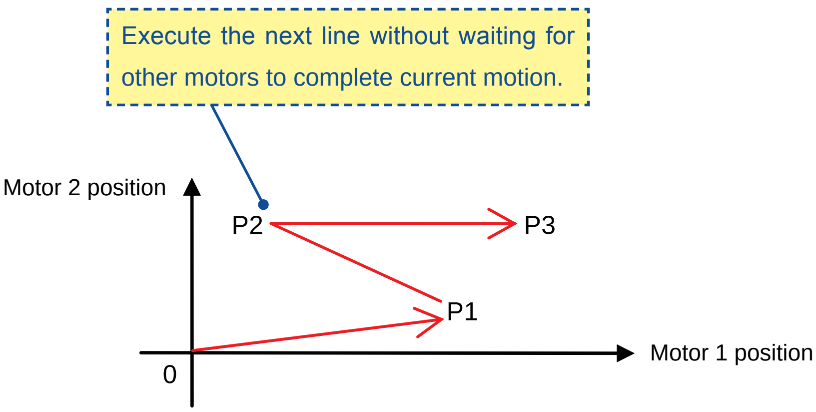
Line 3 : When Motor 1 and Motor 2 complete the current motion (line 2), then Motor 1 moves to P2 with the same speed and acceleration as in the previous line, and Motor 2 moves to P2 with the same speed and acceleration as in the previous line

Line 4 : Motor 1 starts to move to P3 without waiting for Motor 2 to reach P2 (line 3)

When Y command is used instead of P command, the command in the next line is enabled to execute without waiting for the completion of the motion by Y command.

For performing Push Motion, substitute Z command for Q command to allow the motor to perform the next motion independently.

Note that the motor completes one motion before executing the next command when Y commands or Z commands is used continuously. In series of Y commands or Z commands, the last command is effective for non-synchronized motion, although commands other than the last one complete the positioning motion.



# 4.4. Interpolation (Optional: R Type only)

In this section interpolation programs for two motors are introduced. In order to make sure of synchronization, the condition that adjacent Motor IDs are assigned to two motors needs to be met.

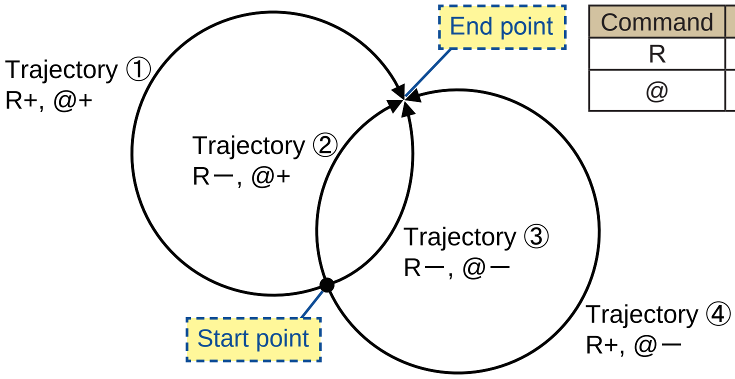
Using two motors, assign the current position as a starting point, and set the end point by P command, then circular interpolation is possible with specifying radius or center point of circle. Linear interpolation is performed when radius is set to 0 (zero).

> Interpolation should be performed by the adjacent motors for ensuring of synchronization.
> CM2 can operate merge motion, during even at the interpolation.
> By using the " ; " command, multiple commands can be concatenated in multiple lines.

## 4.4.1. Circular Interpolation by Specifying Radius

There are 4 different motion trajectories when starting point (current position), end point and radius are specified in the circular interpolation. See diagram below. Select one of the trajectories by combining R command (specify radius), @ command (execute interpolation) and + or - modifier.

In this case, the center of a circle is automatically calculated.




| Command | + | - |
|---|---|---|
| R | Long Arc | Short Arc |
| @ | CW | CCW |

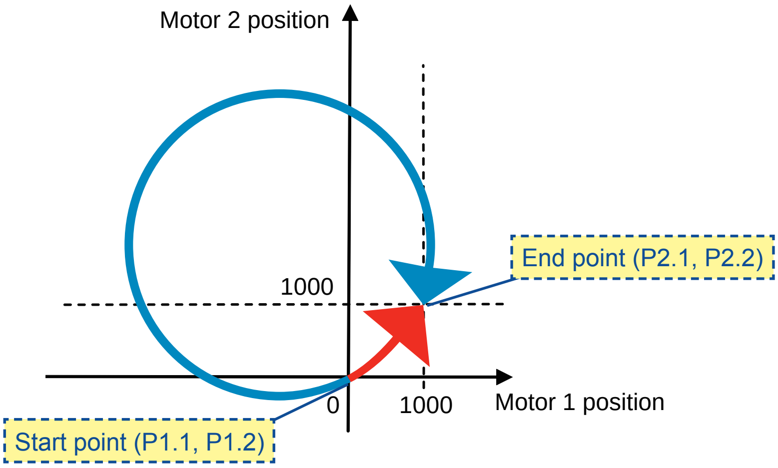
```
A1.1=100
S1.1=100
P1.1=0
P2.1=1000
R1.1=2000                 …Definition of Radius
A1.2=100
S1.2=100
P1.2=0
P2.2=1000
R1.2=2000                 …Definition of Radius
```

A1.1 to R1.1: Motor 1 (X axis) Data definition

A1.2 to R1.2: Motor 2 (Y axis) Data definition

```
B1.1
A1.1,S1.1,P1.1,A1.2,S1.2,P1.2      …Move to the starting point of circular arc.
R1.1,R1.2,@2.2+,@2.1+              …Move to the end point (P2.1,P2.2).
                                     Draw a long arc in the CW direction with Radius R1.
END
```

The Program Bank above draws a circular arc trajectory outlined in blue, where the modifier for R command is + (Long Arc) and the modifier for @ command is + (CW).




In contrast, the program below draws a circular arc trajectory outlined in red, where the modifier for R command is - (Short Arc) and the modifier for @ command is - (CCW).

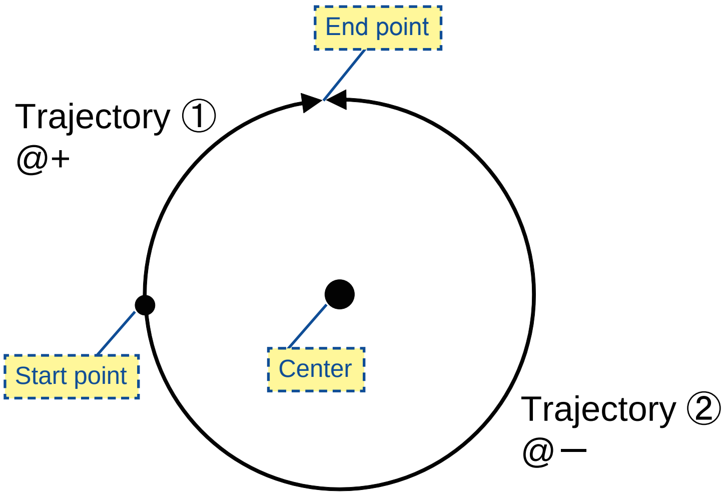
```
B2.1
A1.1,S1.1,P1.1,A1.2,S1.2,P1.2    …Move to the starting point
R1.1-,R1.2-,@2.2-,@2.1-          …Move to the end point (P2.1,P2.2).
END                                Draw a short arc in the CCW direction with radius R1.
```

## 4.4.2. Circular Interpolation by Specifying Center Point

By specifying starting point (current position), end point and center point (N), circular interpolation is possible with the motors. There are 2 different motion trajectories by combining N command (specify center point), @ command (execute interpolation) and + or - modifier.

In this case, the radius of a circle is automatically calculated.

| Command | + | - |
|---|---|---|
| @ | CW | CCW |




```
A1.1=100
S1.1=100
P1.1=0
P2.1=1000
N1.1=-823
A1.2=100
S1.2=100
P1.2=0
P2.1=1000
N1.2=1823
```

Lines A1.1=100 through N1.1=-823: Motor 1 (X axis) Data definition (N1.1=-823 …Definition of center)

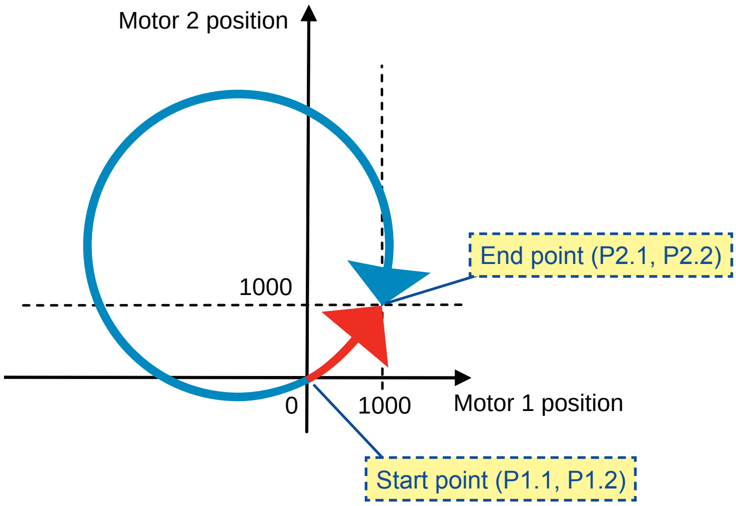
Lines A1.2=100 through N1.2=1823: Motor 2 (Y axis) Data definition (N1.2=1823 …Definition of center)

```
B1.1
A1.1, S1.1, P1.1, A1.2, S1.2, P1.2
N1.1, N1.2, @2.2+, @2.1+
END
```

A1.1, S1.1, P1.1, A1.2, S1.2, P1.2 …Move to the starting point of circular arc.

N1.1, N1.2, @2.2+, @2.1+ …Move to the end point(P2.1,P2.2).
Draw a circular arc in the CCW direction with radius R1.

The program above draws a circular arc outlined in blue.




The program below draws a circular arc outlined in red.

```
B2.1
A1.1,S1.1,P1.1,A1.2,S1.2,P1.2    …Move to the starting point
N1.1,N1.2,@2.2-,@2.1-            …Move to the end point (P2.1,P2.2).
                                   Draw a circular arc in the CCW direction with radius R1.
END
```

## 4.4.3. Linear Interpolation

When R memory (radius data) is set to 0 (zero) in circular interpolation by specified radius, the motors perform linear interpolation.

Define the end point and set R memory to 0. The motors perform linear interpolation starting from the starting point (current position) to the end point. + or - modifier for R command and @ command do not affect the motion trajectory.

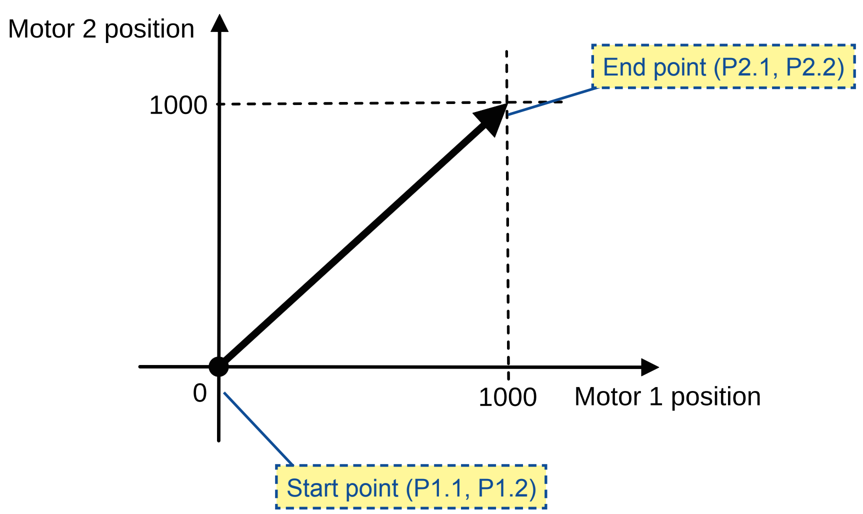
```
A1.1=100
S1.1=100
P1.1=0
P2.1=1000
R1.1=0                      …Specify Linear interpolation
A1.2=100
S1.2=100
P1.2=0
P2.1=1000
R1.2=0                      …Specify Linear interpolation
```

Motor 1 (X axis) Data definition

Motor 2 (Y axis) Data definition

```
B1.1
A1.1,S1.1,P1.1,A1.2,S1.2,P1.2   …Move to the starting point.
R1.1,R1.2,@2.2,@2.1             …Perform linear interpolation to the end point (P2.1,P2.2).
END
```

The program above draws a line outlined in black.



# 4.5. Ladder Logic Banks

In this section Ladder Logic Banks are introduced. Execution of commands in a Ladder Logic Bank does not accompany the motion of motor. Only mathematical and/or logic operations and branch processing are executed in the bank.

## 4.5.1. Basic Operations

| Command | Description | |
|---|---|---|
| L1.1 | | Ladder Logic Bank1 |
| V1=V1+V2 | …Add V2 to V1 | |
| V1>V3, V1=V3, T0 | …Branching without motion | |
| V1<V4, V1=V4, T0 | …Branching without motion | |
| V3=V1 | …Set value of V1 to V3 | |
| P1.1 | …Display P1 value | |
| END | | |

In describing a Ladder Logic Bank, place [ L Bank No. Motor ID ] at the beginning of bank and describe any command lines after that.

When P command is used in a Ladder Logic Bank, it does not cause any motion. It only displays the value of P memory.

As a Ladder Logic Bank is continuously executed in the period of time based on parameter K63, X command can not be available in Ladder Logic Bank.

Chapter **5**

# Setting Examples

In this section, parameter settings or procedures required for realizing various functions are described.

## 5.1. Manual Jog / Feed

【Manual Jog】

Manual jog makes the motor move incrementally by the number of pulses set by parameter, with each input of one-shot signal. This is useful for fine adjustments.

The setting of parameters is as below.

<table>
<tr><th>Parameter</th><th colspan="2">Contents</th></tr>
<tr><td>K28</td><td>Quick Response Rising Edge</td><td rowspan="4">Set to either of followings.<br>8 : Manual Jog in CW direction<br>9 : Manual Jog in CCW direction</td></tr>
<tr><td>K29</td><td>Quick Response Falling Edge</td></tr>
<tr><td>K31</td><td>Slow Response Rising Edge</td></tr>
<tr><td>K32</td><td>Slow Response Falling Edge</td></tr>
<tr><td>K50</td><td colspan="2">Number of pulses for one movement</td></tr>
</table>

【Manual Feed】

Manual feed makes the motor move in a specified direction continuously while the signal is ON. The motor stops when the signal is OFF.

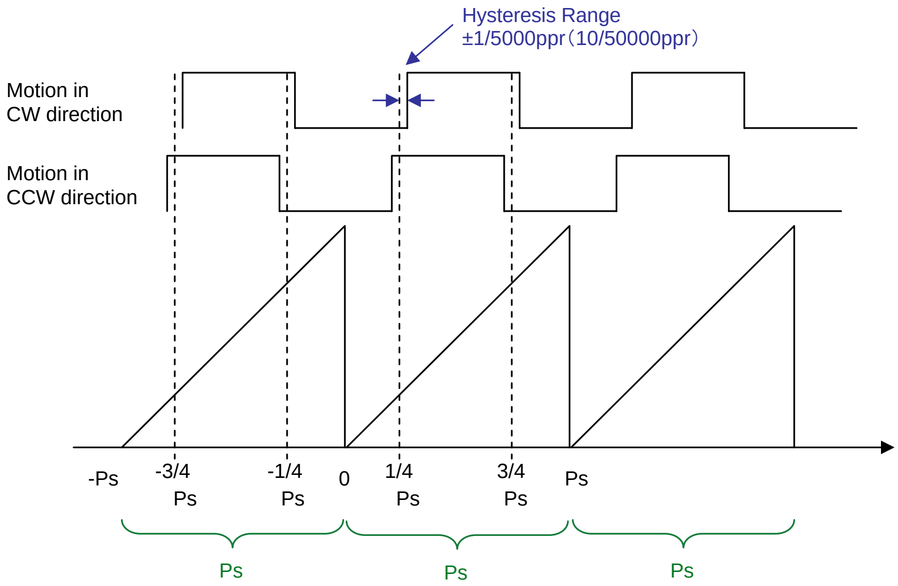
The setting of parameters is as below.

<table>
<tr><th>Parameter</th><th colspan="2">Contents</th></tr>
<tr><td>K27</td><td>at the Quick Response Target Voltage</td><td rowspan="2">Direction of continuous rotation<br>3 : CW direction, 4 : CCW direction</td></tr>
<tr><td>K30</td><td>at the Slow Response Target Voltage</td></tr>
<tr><td>K49</td><td colspan="2">Manual Feed Speed</td></tr>
</table>

# 5.2. Rotation Pulse Output

The motor's current position shall be divided by the range of K24 value, the output will be ON at the first half of set position by K24 then OFF at the last half.

However the output timing will be different in CW and CCW direction because the threshold for output signal ON and OFF has plus minus 1/5000ppr (plus minus 10/50000ppr) hysteresis to the noise.




| Parameter | Contents |
|---|---|
| K34 | Output Functions [ 7 : Rotation Pulse Output ]<br>In case of Quadrature Encoder Output, both Output 1 Function and Output 2 Function should be set to 7. |
| K24 | Position interval (number of pulses) for Rotation Pulse Output |
| K33 | Output logic by ON or OFF. |

Depending on the value of parameter K24 and the rotation speed of motor, the time interval of output pulse may be less than 0.5 msec.
In that case, the Rotation Pulse could not be output correctly.

# 5.3. Origin Search

Origin Search can be executed by transmitting "| (bar)" command or by using the input to which Origin Search Start Function is assigned through setting "7" in parameter K28, K29, K31 or K32.

Origin Search operates according to the following parameter setting.

| Parameter | Contents |
|---|---|
| K42 | Speed for Origin Search |
| K43 | Acceleration for Origin Search |
| K45 | Origin Search Direction : CW or CCW |
| K46 | Origin Signal Source |
| K48 | Offset Distance Between Machine Origin and Electrical Origin |

Besides, "Origin Signal Source" of parameter K46 and related parameter settings are necessary.

## 5.3.1. Origin Search using Stopper

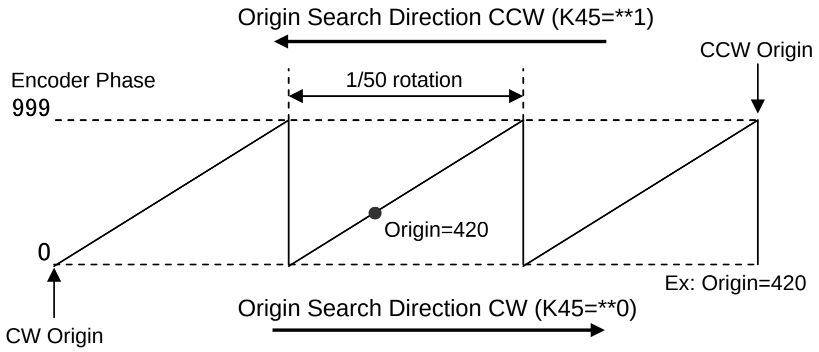
The following parameter setting is also necessary for the Origin Search by Stopper.

| Parameter | Contents |
|---|---|
| K46 | Origin Signal Source 0 or 1: Origin Search by Stopper |
| K47 | Torque Level when searching for origin using a Stopper |

Origin Search completes when the torque pushing against the stopper reaches the set level by K47 and the speed goes 1/16 below the set speed by K42. Then the encoder phase value will be displayed.

For the stable origin search, adjust an attachment as a coupling for the encoder phase value indicated in "Origin=* * * " to be between 200 and 800.

The encoder phase will straightly changes from 0 to 999 per 1/50 rotation.
When the completion of Origin Search, in-position signal will be output and the motor stops at the encoder phase 0 point that is 1 cycle ahead of completion.



### 5.3.2. Origin Search using Sensor

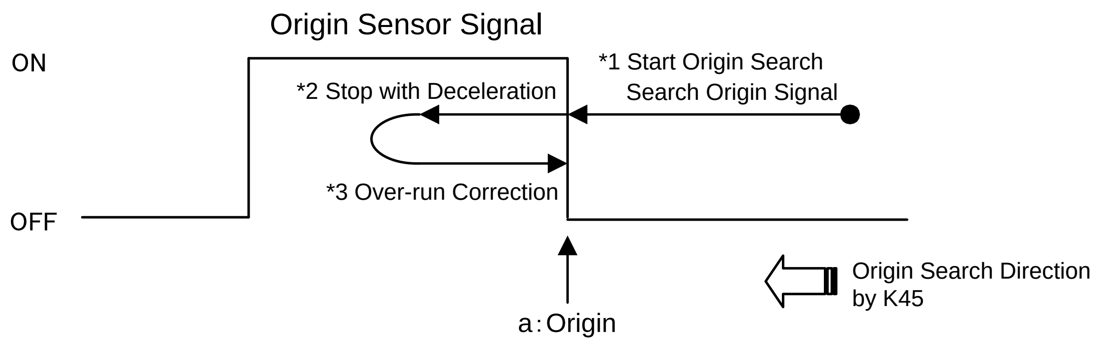
The following parameter setting is also necessary for the Origin Search by sensor.

| Parameter | Contents |
|---|---|
| K27 | Input Functions at the Quick Response Target Voltage “2 : Origin Sensor”<br>Do not set “2 : Origin Sensor” to multiple inputs to prevent abnormal detection of the origin sensor signal caused by the conflict between the inputs. |

Moreover, depending on the status of origin sensor signal input when origin search starts, there are the following differences in the movement of origin search.

【When an origin sensor signal is OFF】

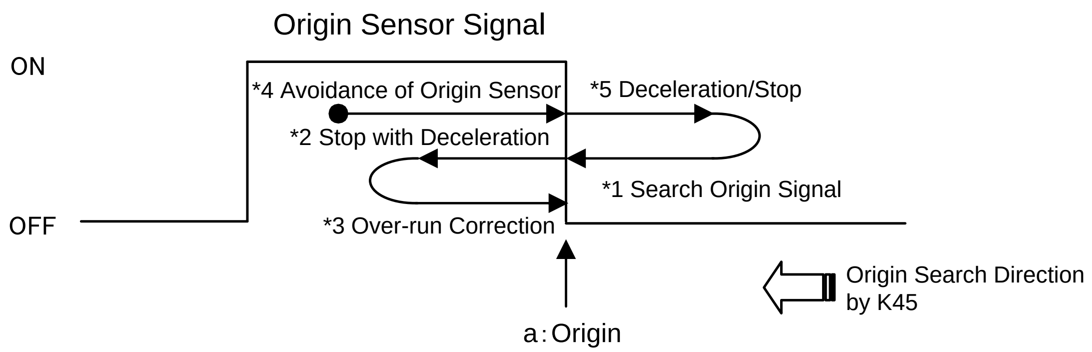
Start Origin Search, move in the direction set by K45, start deceleration at the rising edge of sensor signal and stop. Complete origin search after returning to the point a.




【When an origin sensor signal is ON】

For detecting the rising edge of sensor signal to be possible, move in the opposite direction from what is set by K45 to turn off a sensor signal.

When passing the point a in the figure, start to decelerate after detecting a sensor signal off, then the same motion as “When an origin sensor signal is OFF” in the previous paragraph will be executed.

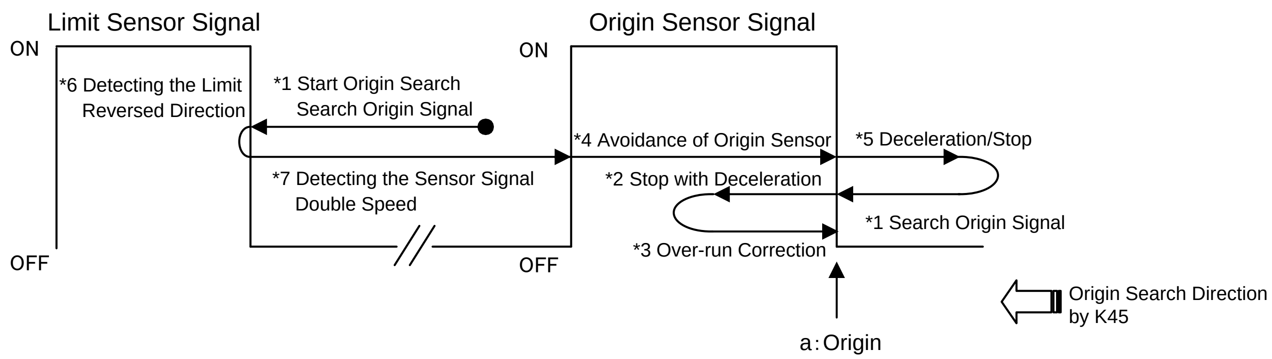


【Use of Limit Sensor concurrently】

| Parameter | Contents |
|---|---|
| K27 | Input Functions at the Quick Response Target Voltage<br>“6 : CW Limit Sensor” or “9 : CCW Limit Sensor” |

It will be operated as below when the Limit Sensor in the same direction as an origin search is assigned to another input.

Start Origin Search, move in the direction set by K45. After detecting the limit sensor signal, start to move in the reverse direction. Move at the double speed of what is set by K42, and detect the origin sensor signal. After detecting the origin sensor signal, then the same motion as "When an origin sensor signal is ON" in the previous paragraph will be executed.




【Example】

Input Functions at the
Quick Response Target Voltage
K27=629000

Origin Search Direction : CW

INPUT6 : CW Limit Sensor
INPUT5 : Origin Sensor
INPUT4 : CCW Limit Sensor
INPUT1~3 : No Operation

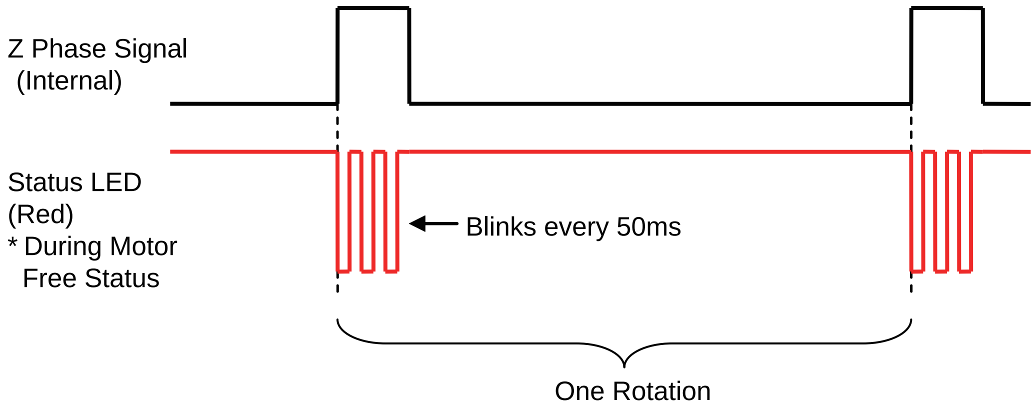
### 5.3.3. Origin Search with Z Phase Signal

The following parameter setting is also necessary for the Origin Search with Z Phase Signal.

| Parameter | Contents |
|---|---|
| K46 | Origin Signal Source 4-7: Z Phase Signal |

Z Phase Signal is the signal generated by an internal position sensor of Cool Muscle 2 and output once per rotation. Usage of Z Phase Signal to detect an origin makes a precise origin search possible that always detects the same origin without an external origin sensor even in a rotary motion. The sequence for the origin search is the same as the origin search with sensor.

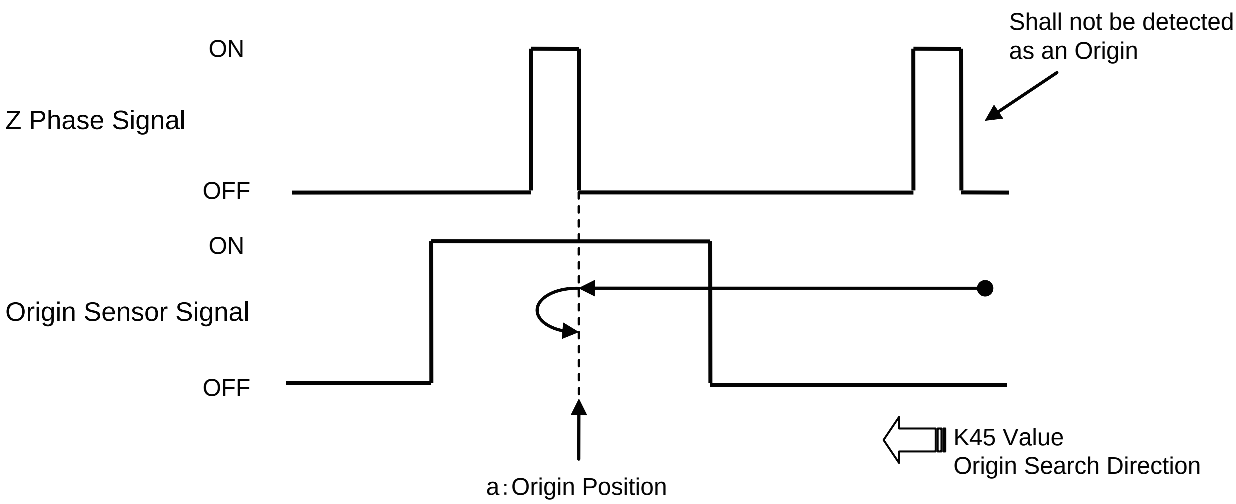
When the motor free by the command " ) " or an input function, the status LED shall be on all the time but when the Z Phase Signal is selected by K46, the status LED shall blinks quickly only during the Z Phase Signal is output.



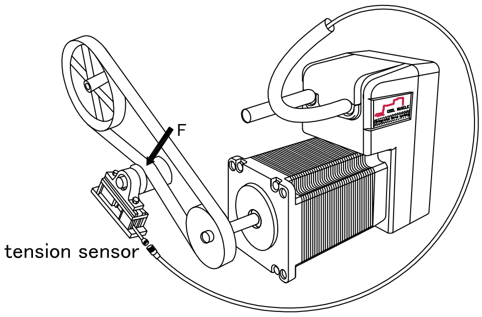

【Concurrent Usage with an Origin Sensor Signal】

it is possible to detect an origin by a combination with the Z Phase signal and an origin sensor signal. Therefore an origin search with better repeat accuracy is possible.

The sequence of the Origin Search is the same as the Origin Search by sensor but the origin shall be the position where the effective edges of both a Z Phase signal and an origin sensor signal are detected.



# 5.4. External Encoder




The full closed-loop position control is available by using the output signal of external encoder equipped for the control target. It is possible to be compatible with the compensation for belt-slipping or backlash of gears, or position control for the stage with linear encoder.

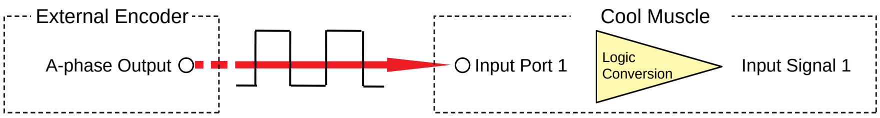
When applying an external encode, the following parameter settings are needed.

| Parameter | Contents |
|---|---|
| K71 | External Encoder Type |
| K72 | External Encoder Resolution |

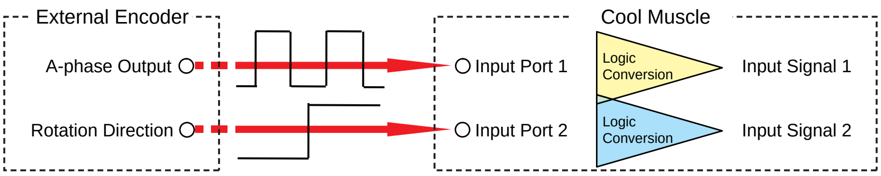
The outputs of external encoder are connected with Input port 1 and Input port 2 of Cool Muscle. Therefore the input functions assigned to Input 1 and Input 2 through parameter settings of K27 – K32 are not available.

【Signals connection between external encoder and Cool Muscle】
Configure the effective edge of input pulse signal by setting of parameter K26.

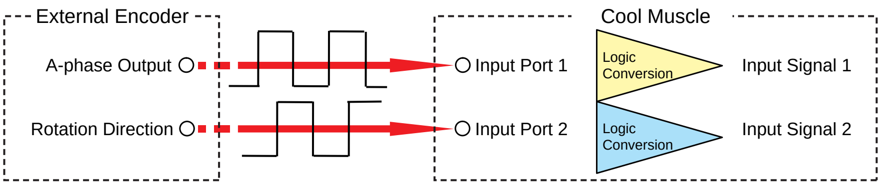
[A-phase signal: pulse input]
Every rising edge of input pulse, pulse counting is performed with either counting-up when moving with increasing position, or countdown when moving with decreasing position. The effectiveness of pulse is determined only at the rising edge of input signal, miscounting caused by noise or vibration of load axis could occur.




[A-phase signal: pulse input, B-phase signal: rotation direction]
Every rising edge of input pulse, either counting-up or countdown is performed in accordance with the rotation direction signal of external encoder. The effectiveness of pulse is determined only at the rising edge of input signal, miscounting caused by noise or vibration of load axis could occur.




[A-phase signal: pulse input, B-phase signal: pulse input]
When two-phase signal, of which phase is shifted by 90 degree each other, is input to Input 1 and Input 2, pulse counting is performed with automatically discriminating whether counting-up or countdown.




*Refer to Parameter K71 for detail information such as the timing of counting.

## 5.4.1. External Encoder / Index Operation

The motor continues to rotate until the count of external encoder pulse reaches the specified number of pulses. Then, the motor stops to rotate when count value reaching the specified number of pulses. (Recovering operation for the amount of overrun is not supported.)
This operation is appropriate for the equipment such as winding machine, where a fixed amount is required to be wound without slack in a fixed direction.
*In Index Operation, the motor operation is not affected by the setting for External Encoder Resolution (K72).

[Example of Use]
Set the parameter K71 according to the pulse type of external encoder.

| | |
|---|---|
| K71.1=1 | : A-phase Index |

Set the data of position, speed and acceleration in the same manner as in normal positioning, and execute the operation.

| | |
|---|---|
| P.1=10000 | : Set the target position of load for the position data. |
| S.1=10 | : Set the speed of Cool Muscle. |
| A.1=100 | : Set the acceleration of Cool Muscle. |
| ^.1 | |

The motor continues to rotate at a set speed of S until the count value of external encoder pulse reaches 10000. Then, the rotation stops when the count value reaching 10000.
Although the actual count could overrun for the target position at this time, the motor stops right there without recovering operation for the amount of overrun.
It is possible to confirm the current count value of external encoder with using query command “?76”.

| | |
|---|---|
| ?76.1 | : Transmission command to Cool Muscle |
| Ecnt.1=10005 | : Receiving data from Cool Muscle |

## 5.4.2. External Encoder / Feedback Operation

The whole system can be controlled as a full closed-loop system by utilizing the feedback pulse from external encoder equipped for the control target.

[Example of Use]
Set the pulse type and resolution of external encoder.

| | |
|---|---|
| K71.1=4 | :A&B phase feedback |
| K72.1=1000 | :1,000 ppr |

Set the data of position, speed and acceleration in the same manner as in normal positioning, and execute the operation.

| | |
|---|---|
| P.1=10000 | : Set the target position of load for the position data. |
| S.1=10 | : Set the speed of Cool Muscle. |
| A.1=100 | : Set the acceleration of Cool Muscle. |
| ^.1 | |

With tracking the command value, the feedback control for the position of control target is performed.
It is possible to confirm the current count value of external encoder with using query command “?76”.

| | |
|---|---|
| ?76.1 | : Transmission command to Cool Muscle |
| Ecnt.1=10000 | : Receiving data from Cool Muscle |

## 5.4.3. External Encoder / Pulse-Counting Operation

The counting operation of pulses input to Cool Muscle from an external encoder is simply performed. The motor operates in the same manner as in normal positioning.
This operation is used for the control with responding to amount of movement or speed of control target.

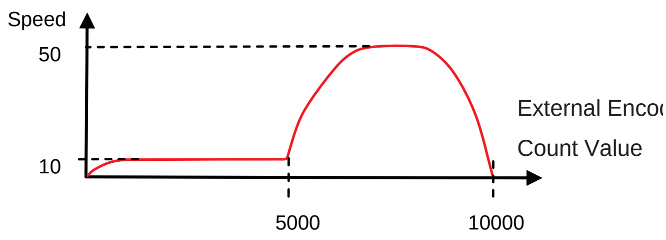
[Example of Use]
In the following example, with using the Ladder Logic Bank described in the section 2.2.3, the motor speed can be changed according to the count of pulses from external encoder equipped for the control target.

Change the setting of General Variable 1, from V1.1=”Px” (current position) to V1.1=”Ecnt” (External Encoder Count).

V1.1 = “Ecnt” : Set the current count value of external encoder into General Variable 1.
Other settings and the definition of Ladder Logic Bank is the same as in section 2.2.3.

After completing all the definitions, execute the Ladder Logic Bank 1 through inputting the command as below.
[L1.1
Operate the motor in the Direct Mode as follows.
A.1=100
P.1=10000
^.1

In this example, the motor operates at the speed of 10 when the count value of external encoder is less than 5000, shown in the right.
However, it operates at the speed of 50 when the count value is over 5000.



# 5.5. Torque feedback control

The torque feedback control is available for applications such as push control common in pneumatic sliders or constant tension control.
It is needed to specify positions and speeds because the control is performed during the positioning operation.

When applying torque feedback control, the following parameter settings are needed.

| Parameter | Contents |
|---|---|
| K38 | Target controlled by Analog Input |
| K74 | Proportional Gain for Torque Control |
| K75 | Integral Gain for Torque Control |
| K76 | Torque Sensor Input offset value |
| K77 | Input range for Torque Sensor Signal |

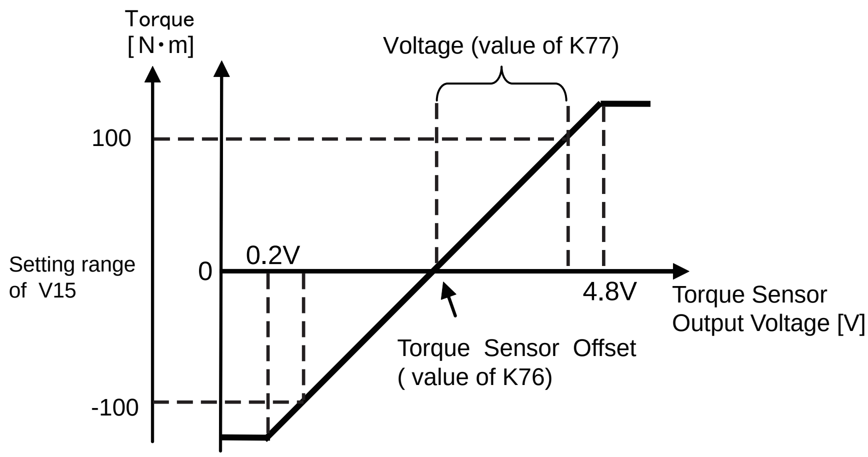
[Example of Setting]
Set Torque Feedback Control into parameter K38, “Target controlled by Analog Input”.
K38.1=10 : Setting the target controlled by Analog Input

Next, set K76, “Torque Sensor Input offset value”, and K77, “Input Range for Torque Sensor Signal”.
In the torque feedback control, the motor output can be controlled in accordance with K74 “Proportional Gain for Torque Control” and K75 “Integral Gain for Torque Control”, for the feedback data from external torque sensor to track the torque command value specified in the range 0 - ±100 by using General Variable 15.

When using the torque sensor with output of 1[V] per 0.2[N･m] and offset voltage of 2.5[V], the controllable range is 0 - ±0.46[N･m] because the analog input voltage is in the range of 0.2[V] - 4.8[V].

For example, the torque command value is required to be maximized (V15=100) when the detected torque of sensor is 0.4[N･m], set parameters as below.

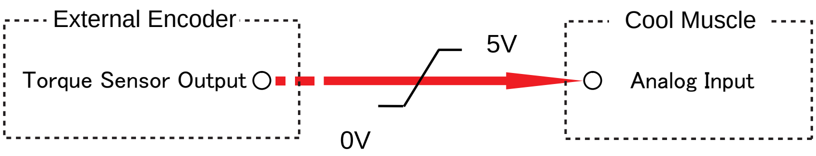
K76.1=250 : Set the offset voltage for Torque Sensor Input. (unit : 0.01 V )
K77.1=200 : Set the difference between the output voltage of torque sensor at maximum torque command value and the offset voltage. (unit : 0.01 V )



At last, set the torque command value and operation range for torque feedback control. The torque command value is set into the General Variable 15 in the range 0 - ±100.
For example, Cool Muscle operates at the speed of 60[min-1] from current position to the position of 10000 pulses, meanwhile the load torque is required to be controlled with the constant torque of 0.1[N･m] , set as below.

| | |
|---|---|
| V15 = 25 | : Setting for torque command value (0.1[N･m]/0.4[N･m] × 100) |
| S0 = 100 | : Speed setting for Cool Muscle (at resolution[K37]=3) |
| P0 = 10000 | : Setting for target position |

After completing every setting, input the output voltage of torque sensor to Analog Input of Cool Muscle, and start operation. Torque feedback control is performed until Cool Muscle reaching the position of 10000.



# 5.6. Modbus Protocol

Modbus protocol is the communication protocol which has developed by Modicon Inc. for PLC applications. Modbus protocol, of which specifications have got open worldwide, is widely used in the FA or PA field due to its simple transmission structure.
Cool Muscle supports Modbus protocol and can be directly connected to Modbus compatible devices with not using the communication converter but just setting the following parameters.

| Parameter | Contents |
|---|---|
| K20 | Communication baud rate (Modbus host→ Cool Muscle) |
| K65 | Slave-to-slave communication baud rate (Cool Muscle → Modbus Slave) |
| K78 | Modbus host communication address (set -1 when not used) |
| K79 | Modbus slave communication – input address (set -1 when not used) |
| K80 | Modbus slave communication – output address (set -1 when not used) |
| K81 | Slave address |
| K82 | Parity |

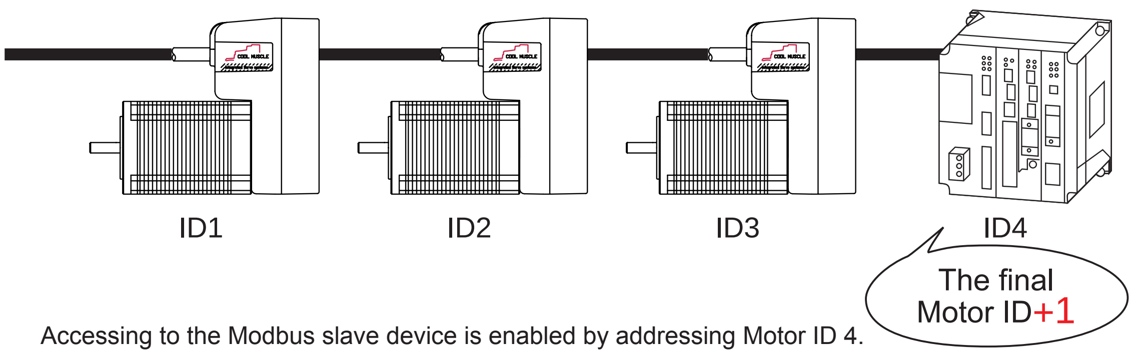
In the Modbus protocol, there are a host and a slave necessarily and the communication can be established through a response of the slave to a communication request from the host.
Cool Muscle can communicate either as a host or as a slave.

[Modbus Host Communication]
Cool Muscle can be used as a Modbus slave by connecting a Modbus host device to the host communication side of Cool Muscle. The host device can transmit commands to Cool Muscle, and read or write the data of Cool Muscle.

[Modbus Slave Communication]
Cool Muscle can be used as a Modbus host by connecting a Modbus slave device to the slave communication side of Cool Muscle. The I/O control or the status read of a slave device can be performed by Cool Muscle.

In the Modbus slave communication, a slave device can be treated as it exist on the daisy-chained network of Cool Muscles through automatically generating a Modbus message from some CML commands related to I/O.
Accessing to a Modbus slave device can be performed by assigning the final Motor ID + 1 for the CML command.

[Example]
When connecting a Modbus slave device onto the daisy-chained network of three axes.




Accessing to the Modbus slave device is enabled by addressing Motor ID 4.

### 5.6.1. Message Transmission Mode

Modbus protocol equipped in Cool Muscle performs the message transmission in RTU (Remote Terminal Unit) mode.

<table>
<tr><th colspan="2">Item</th><th>Contents</th></tr>
<tr><td colspan="2">Communication method</td><td>Half-duplex, Asynchronous method</td></tr>
<tr><td colspan="2">Communication Protocol</td><td>Modbus RTU mode</td></tr>
<tr><td colspan="2">Transmission Speed</td><td>9.6K, 19.2K, 38.4K, 57.6K, 115.2K, 230.4 Kbps<br>(Set by parameter K20 or K65)</td></tr>
<tr><td colspan="2">Transmission Code</td><td>Binary</td></tr>
<tr><td rowspan="2">Error check<br>(Error detection)</td><td>Vertical</td><td>Parity</td></tr>
<tr><td>Horizontal</td><td>CRC-16</td></tr>
<tr><td rowspan="4">Character configuration</td><td>Start Bit</td><td>1 bit</td></tr>
<tr><td>Data Length</td><td>8 bits</td></tr>
<tr><td>Parity Bit</td><td>None/even/odd<br>(Set by parameter K82)</td></tr>
<tr><td>Stop Bit</td><td>1 bit</td></tr>
<tr><td colspan="2">Time Interval between Data</td><td>Less than 8 byte times</td></tr>
</table>

### 5.6.2. Time Interval between Data

When transmitting a message, be sure that the time interval between data constructing a message must not exceed 8 byte times. If a greater interval than specified time occurs, Cool Muscle assumes a transmission has terminated and performs reception of an illegal message.

| Baud Rate | Time Interval between Data |
|---|---|
| 9.6Kbps | Less than 6.66msec |
| 19.2Kbps | Less than 3.33msec |
| 38.4Kbps | Less than 1.66msec |
| 57.6Kbps | Less than 1.11msec |
| 115.2Kbps | Less than 0.55msec |
| 230.4Kbps | Less than 0.27msec |

### 5.6.3. Message Framing

The Modbus message is constructed as below.

| Slave Address (8 bits) | Function Code (8 bits) | Data | Error Check CRC-16 (16 Bits) |
|---|---|---|---|

● Slave Address

The slave address is specified in the range of 1 - 247 decimal by parameter K81.

The broadcast mode (slave address 0) is not available.

● Function Code

The function code is classified in outline as below. Refer to section 5.6.4 for detail information.

| Code | Function | Remarks |
|---|---|---|
| 01 | Read the status of slave output | only in the slave communication |
| 02 | Read the status of slave input | only in the slave communication |
| 03 | Read the motor status | only in the host communication |
| 04 | Read the I/O status | only in the host communication |
| 05 | Single ON/OFF for slave outputs | only in the slave communication |
| 15 | Multiple ON/OFF for slave outputs | only in the slave communication |
| 16 | Command transmission to Cool Muscle | only in the host communication |

● Data

The data field is constructed differently according to the function code. Refer to section 5.6.4 "Function Code" for detail information.

● Error Check

The 16-bit error checking code generated by CRC-16 method is appended as the last field in the message.

### 5.6.4. Function Code

In case of using the following parameter settings, an example of a response to each function code is shown as below.

(The slave device ID is 4 when in the Modbus slave communication.)

| Parameter | Contents | Set value |
|---|---|---|
| K78 | Modbus host communication address | 100 |
| K79 | Modbus slave communication - input address | 200 |
| K80 | Modbus slave communication - output address | 300 |
| K81 | Slave address | 1 |

## 【Function Code : 01 (0x01)】

● Function

The output status in the slave can be read.

This function is supported only in the Modbus slave communication.

● Transmission Message

The transmission message to the slave is generated automatically through transmitting Output Status - Query Command “?50.n” to Cool Muscle. (n = the final Motor ID + 1 : indicating the Modbus slave device ID) The number of Read Registers is fixed to 16.

Example : a transmission message to the slave when “?50.4” has transmitted to Cool Muscle.(automatic generation)

When the starting address is set to 300 (0x2C hex) by K80, the 16 output status are read from address 301 in a slave device.

| Field Name | Contents (Hex) | Remarks |
|---|---|---|
| Slave Address | 0x01 | Set by K81 |
| Function Code | 0x01 | |
| Starting Address (Hi) | 0x01 | Set by K80 |
| Starting Address (Lo) | 0x2C | |
| Number of Registers (Hi) | 0x00 | Fixed |
| Number of Registers (Lo) | 0x10 | |
| Error Check | CRC (16 bits) | |

● Response

The response from the slave is interpreted by Cool Muscle automatically.

## 【Function Code : 02 (0x02)】

● Function

The output status in the slave can be read.

This function is supported only in the Modbus slave communication.

● Transmission Message

The transmission message to the slave is generated automatically through transmitting Output Status - Query Command “?50.n” to Cool Muscle. (n = the final Motor ID + 1 : indicating the Modbus slave device ID) The number of Read Registers is fixed to 16.

Example : a transmission message to the slave when “?50.4” has transmitted to Cool Muscle.(automatic generation)

When the starting address is set to 300 (0x2C hex) by K80, the 16 output status are read from address 301 in a slave device.

| Field Name | Contents (Hex) | Remarks |
|---|---|---|
| Slave Address | 0x01 | Set by K81 |
| Function Code | 0x02 | |
| Starting Address (Hi) | 0x00 | Set by K79 |
| Starting Address (Lo) | 0xC8 | |
| Number of Registers (Hi) | 0x00 | Fixed |
| Number of Registers (Lo) | 0x10 | |
| Error Check | CRC (16 bits) | |

● Response

The response from the slave is interpreted by Cool Muscle automatically.

## 【 Function Code : 03 (0x03) 】

● Function

The specified register address (the current motor information) can be read.

This function is supported only in the Modbus host communication.

The number of data bytes is four.

| Motor Information to be read | Register Address | Corresponding CML |
|---|---|---|
| Position Deviation | K78 setting | ?95 |
| Current Position | value of K78 +2 | ?96 |
| Current Speed | value of K78 +4 | ?97 |
| Current Torque | value of K78 +6 | ?98 |
| Motor status | value of K78 +8 | ?99 |

| Motor Information to be read | Register Address | Corresponding CML |
|---|---|---|
| V0 | value of K78 +10 | V0 |
| V1 | value of K78 +12 | V1 |
| V2 | value of K78 +14 | V2 |
| V3 | value of K78 +16 | V3 |
| V4 | value of K78 +18 | V4 |
| V5 | value of K78 +20 | V5 |
| V6 | value of K78 +22 | V6 |
| V7 | value of K78 +24 | V7 |
| V8 | value of K78 +26 | V8 |
| V9 | value of K78 +28 | V9 |
| V10 | value of K78 +30 | V10 |
| V11 | value of K78 +32 | V11 |
| V12 | value of K78 +34 | V12 |
| V13 | value of K78 +36 | V13 |
| V14 | value of K78 +38 | V14 |
| V15 | value of K78 +40 | V15 |

● Transmission Message

Example : a transmission message to Cool Muscle when reading the current position.

Note that the number of registers is 2 and the starting address is 101(0x65 hex : value of K78 + 1).

| Field Name | Contents (Hex) | Remarks |
|---|---|---|
| Slave Address | 0x01 | Set by K81 |
| Function Code | 0x03 | |
| Starting Address (Hi) | 0x00 | |
| Starting Address (Lo) | 0x65 | |
| Number of Registers (Hi) | 0x00 | Fixed |
| Number of Registers (Lo) | 0x02 | |
| Error Check | CRC (16 bits) | |

● Response

Here is an example of a response from Cool Muscle. The slave address and the function code are echoed back without modification. The number of data bytes is four.

| Field Name | Contents (Hex) | Remarks |
|---|---|---|
| Slave Address | 0x01 | |
| Function Code | 0x03 | |
| Number of Data Bytes | 0x04 | Fixed |
| Data1 (Hi) | 0x00 | |
| Data1 (Lo) | 0x01 | |
| Data2 (Hi) | 0x86 | |
| Data2 (Lo) | 0xA0 | |
| Error Check | CRC (16 bits) | |

In above example, the current position is 0x000186A0 (hex), representing 100000 decimal.

## 【Function Code : 04 (0x04)】

● Function

The I/O and the status information can be read.

This function is supported only in the Modbus host communication.

| Motor Information to be Read | Register Address | Corresponding CML |
|---|---|---|
| Input Status (ID1~ID15) | K78 setting ~K78+14 | ?70.n (n : Motor ID) |
| Output Status (ID1~ID15) | K78 setting +16 ~K78+30 | ?50.n (n : Motor ID) |
| Motor Status (ID1~ID15) | K78 setting +32 ~K78+46 | ?99.n (n : Motor ID) |

● Transmission Message

Example : The transmission message to read the 3 input status of ID3 - ID5

The set value of K78 represents ID1, so that ID3 is the set value of K78 + 2.

Note that the starting address is 102 (0x66 hex), which indicates ID3.

| Field Name | Contents (Hex) | Remarks |
|---|---|---|
| Slave Address | 0x01 | Set by K81 |
| Function Code | 0x04 | |
| Starting Address (Hi) | 0x00 | |
| Starting Address (Lo) | 0x66 | |
| Number of Registers (Hi) | 0x00 | |
| Number of Registers (Lo) | 0x03 | |
| Error Check | CRC (16 bits) | |

● Response

Here is an example of a response from Cool Muscle.

The slave address and the function code are echoed back without modification.

| Field Name | Contents (Hex) | Remarks |
|---|---|---|
| Slave Address | 0x01 | |
| Function Code | 0x04 | |
| Number of Data Bytes | 0x06 | |
| Data1 (Hi) | 0x00 | |
| Data1 (Lo) | 0x3F | |
| Data2 (Hi) | 0x00 | |
| Data2 (Lo) | 0x02 | |
| Data3 (Hi) | 0x01 | |
| Data3 (Lo) | 0xFF | |
| Error Check | CRC (16 bits) | |

## 【Function Code : 05 (0x05)】

● Function

Turning ON/OFF the single output in the slave can be performed.

This function is supported only in the Modbus slave communication.

● Transmission Message

The transmission message to the slave is generated automatically through transmitting Output ON/OFF Command “O#.n” or t“F#.n” to Cool Muscle. (n = the final Motor ID + 1 : indicating the Modbus slave device ID)

Example : A transmission message to the slave when transmitting “O7.4” to Cool Muscle. (automatic generation)

Note that the starting address is 206 (0xCE hex), which is the 7th since address 200 (set value of K80) of a slave device.

| Field Name | Contents (Hex) | Remarks |
|---|---|---|
| Slave Address | 0x01 | Set by K81 |
| Function Code | 0x05 | |
| Starting Address (Hi) | 0x00 | |
| Starting Address (Lo) | 0xCE | |
| Preset Data (Hi) | 0x00 | Command O : 1<br>Command F : 0 |
| Preset Data (Lo) | 0x01 | |
| Error Check | CRC (16 bits) | |

● Response

The response from the slave is interpreted by Cool Muscle automatically.

## 【Function Code : 15 (0x0F)】

### ● Function

Turning ON/OFF the multiple outputs in a slave device simultaneously.

This function is supported only in the Modbus slave communication.

### ● Transmission Message

The transmission message to the slave is generated automatically through transmitting Output ON/OFF Command “O#.n=X” to Cool Muscle. (n = the final Motor ID + 1 : indicating the Modbus slave device ID, X=output status) The number of registers is fixed to 16 and the number of bytes is fixed to2.

Here is an example of setting the 16 output status from address 301 in a slave device, as below.

| Address | 301 | 302 | 303 | 304 | 305 | 306 | 307 | 308 | 309 | 310 | 311 | 312 | 313 | 314 | 315 | 316 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Status | ON | ON | OFF | OFF | ON | ON | OFF | ON | OFF | OFF | OFF | OFF | OFF | OFF | OFF | ON |
| Binary | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

C D 0 1

The data is 0xCD01 hex, so that the data to be set to the output status is 52481 decimal.

Example : The transmission message to the slave when transmitting “ O.4=52481” to Cool Muscle.

(automatic generation)

| Field Name | Contents (Hex) | Remarks |
|---|---|---|
| Slave Address | 0x01 | Set by K81 |
| Function Code | 0x05 | |
| Starting Address (Hi) | 0x01 | Set by K80 |
| Starting Address (Lo) | 0x2C | |
| Number of Registers (Hi) | 0x00 | Fixed |
| Number of Registers (Lo) | 0x10 | |
| Number of Bytes | 0x02 | Fixed |
| Preset Data (Hi) | 0xCD | |
| Preset Data (Lo) | 0x01 | |
| Error Check | CRC (16 bits) | |

### ● Response

The response from the slave is interpreted by Cool Muscle automatically.

## 【Function Code : 16 (0x10)】

● Function

The designated command can be transmitted to Cool Muscle.

| Function | Register Address |
|---|---|
| CML Command Transmission | K78 setting +100 |

● Transmission Message

Example : The transmission message of Position Counter Reset Command “|2” when K78=100.

Note that the starting address is 200 (=100+100).

| Field Name | Contents (Hex) | Remarks |
|---|---|---|
| Slave Address | 0x01 | |
| Function Code | 0x10 | |
| Starting Address (Hi) | 0x00 | |
| Starting Address (Lo) | 0xC8 | |
| Number of Registers (Hi) | 0x00 | |
| Number of Registers (Lo) | 0x02 | |
| Number of Bytes | 0x04 | |
| Data1 (Hi) | 0x7C | \| |
| Data1 (Lo) | 0x32 | 2 |
| Data2 (Hi) | 0x0D | CR |
| Data2 (Lo) | 0x00 | Pad with 0 |
| Error Check | CRC (16 bits) | |

● Response

When the message is received correctly, the response is performed by returning the copy of the transmission message with the number of bytes and data area removed.

| Field Name | Contents (Hex) | Remarks |
|---|---|---|
| Slave Address | 0x01 | |
| Function Code | 0x10 | |
| Starting Address (Hi) | 0x00 | |
| Starting Address (Lo) | 0xC8 | |
| Number of Registers (Hi) | 0x00 | |
| Number of Registers (Lo) | 0x02 | |
| Error Check | CRC (16 bits) | |

## 5.6.5. Exception Responses

When a message has transmitted from a Modbus host device, if a slave device receives the massage normally, it returns a normal response. However, it will return an exception response if abnormal event occurs, as below.

The exception response has the following configuration.

| Slave Address (8 bits) | Function Code (8 bits) | Exception Code (8 bits) | Error Check CRC-16 (16 bits) |
|---|---|---|---|

A slave address is placed in the slave address field as well as in the normal response.
A function code, the value of function code of transmission message + 0x80 hexadecimal, is placed in the function code field.

| Function Code | Function Code +0x80 |
|---|---|
| 03 (0x03) | 0x83 |
| 04 (0x04) | 0x84 |
| 16 (0x10) | 0x90 |

The exception code is shown below.

| Exception Code | Name | Meaning |
|---|---|---|
| 01 | Illegal Function | The relevant function is not supported. |
| 02 | Illegal Data Address | The designated data address is non-existent. |

## 5.6.6. Termination of Modbus Mode

The normal RS-232C communication cannot be performed under the Modbus host communication.
Cool Muscle can terminate the Modbus communication mode and perform the normal RS-232C communication through the setting of K81=0 by a Modbus compatible device.

If the Modbus communication mode is required to be terminated by a Modbus incompatible device or the mode has been set accidentally, it is possible to terminate the Modbus mode and perform normal RS-232C communication temporally, through transmitting “FFFFFFFFF” (transmitting F nine times continuously) to Cool Muscle after confirming that the communication baud rate is correct.
In this state, the Modbus communication mode will be terminated by the setting of K81=0.

Chapter 6

# CML List

## 6.1. K Parameter

| K | Parameter | Min | Max | Default | Unit | Description |
|---|---|---|---|---|---|---|
| 20 | Baud Rate | 0 | 5 | 0 | - | The communication baud rate between Cool Muscle and a host.<br>0: 38.4Kbps, 1: 9.6 Kbps, 2: 19.2 Kbps, 3: 57.6 Kbps,<br>4: 115.2 Kbps, 5: 230.4 Kbps |
| 23 | Status Report | 0 | 31 | 1 | - | Event selection for status report setting for Local Echo, confirmation/error messages.<br>0: No status report<br>1: In-position and alarm<br>2: Input status change<br>4: Output status change<br>8: No Local Echo<br>16: Confirmation / error messages |
| 24 | Rotation Pulse Output | 10 | 32767 | 1000 | pulses | Output ON/OFF at regular intervals with pulses.(set K34=7)<br>When both Output 1 and Output 2 in K34 values are set to 7, quadurature encoder pulse is output. |
| 25 | Delay Time for Slow Response Signal | 111111 | 999999 | 333333 | 0.1sec | The delay timel for slow response signal. Each digit must be set individually and assigns Input 6, 5, 4, 3, 2, 1. |
| 26 | Input Logic / P type Operation | 000000 | 333333 | 000000 | - | Input Logic and Execution of P type Operation<br>Each digit must be set individually and assigns Input 6, 5, 4, 3, 2, 1.<br>① Input Logic<br>0 or 2 : Input signal is ON when Input port is ON.<br>(P type effective edge: rising edge)<br>1 or 3 : Input signal is ON when Input port is OFF.<br>(P type effective edge: falling edge)<br>② Execution of P type operation (Apply to C/R type)<br>Set the value "2" or "3" of Input 3 to Input 6 :<br>When Input Signal is ON, P type operation is valid and accept the Pulse signal.<br>When Input Signal is OFF, P type operation is Invalid and refuse the Pulse signal.<br>Set the value "2" or "3" of Input 1 or Input 2 :<br>CM2 operates P type operation when setting values are "2" or "3" to two or more input, during input signal is ON at either Input 1 or Input 2. |

| K | Parameter | Min | Max | Default | Unit | Description |
|---|---|---|---|---|---|---|
| 27 | Input Functions at the Quick Response Target Voltage | 000000 | 999999 | 000000 | - | Assign functions at target voltage level of quick response signal. Each digit must be set individually and assigns Input 6, 5, 4, 3, 2, 1.<br>0: No function<br>1: General Use<br>2: Origin sensor signal<br>3: Manual feed CW<br>4: Manual feed CCW<br>5: Stop Ladder Logic Bank<br>6: CW direction limit sensor<br>(Dual usage as CW origin sensor)<br>7: Emergency stop<br>8: Terminate the Program Bank (same as ] ] )<br>9: CCW direction limit sensor<br>(Dual usage as CCW origin sensor) |
| 28 | Input Function at Rising Edge of Quick Response Signal | 000000 | 999999 | 000000 | - | Assign functions at rising edge of quick response signal. Each digit must be set individually and assigns Input 6, 5, 4, 3, 2, 1.<br>0: No function<br>1: Alarm reset/Program Bank Pause<br>2: Motor free<br>3: Position counter reset<br>4: Execute next Program Bank line<br>5: Execute previous Program Bank line<br>6: Execute Program Bank 1<br>7: Start origin search<br>8: Manual jog CW<br>(K36=2 or 3, execute Program Bank 2)<br>9: Manual jog CCW<br>(K36=2 or 3, execute Program Bank 3) |
| 29 | Input Function at Falling Edge of Quick Response Signal | 000000 | 999999 | 000000 | - | Assign functions at falling edge of quick response signal. Each digit must be set individually and assigns Input 6, 5, 4, 3, 2, 1.<br>0: No function<br>1: Alarm reset/Program Bank Pause<br>2: Enable motor<br>3: Position counter reset<br>4: Execute next Program Bank line<br>5: Execute previous Program Bank line<br>6: Execute Program Bank 1<br>7: Start origin search<br>8: Manual jog CW<br>(K36=2 or 3, execute Program Bank 2)<br>9: Manual jog CCW<br>(K36=2 or 3, execute Program Bank 3) |

| K | Parameter | Min | Max | Default | Unit | Description |
|---|---|---|---|---|---|---|
| 30 | Input Functions at Slow Response Target Voltage | 000000 | 999999 | 000000 | - | Assign functions at target voltage level of quick response signal. Each digit must be set individually and assigns Input 6, 5, 4, 3, 2, 1.<br>0: No function<br>1: General Use<br>2: -<br>3: Manual feed CW<br>4: Manual feed CCW<br>5: Stop Ladder Logic Bank<br>6: CW direction limit sensor<br>7: Emergency stop<br>8: Terminate the Program Bank (same as ] ] )<br>9: CCW direction limit sensor |
| 31 | Input Function at Rising Edge of Slow Response Signal | 000000 | 999999 | 000000 | - | Assign functions at rising edge of quick response signal. Each digit must be set individually and assigns Input 6, 5, 4, 3, 2, 1.<br>0: No function<br>1: Alarm reset/Program Bank Pause<br>2: Motor free<br>3: Position counter reset<br>4: Execute next Program Bank line<br>5: Execute previous Program Bank line<br>6: Execute Program Bank 1<br>7: Start origin search<br>8: Manual jog CW (K36=2 or 3, execute Program Bank 2)<br>9: Manual jog CCW (K36=2 or 3, execute Program Bank 3) |
| 32 | Input Function at Falling Edge of Slow Response Signal | 000000 | 999999 | 000000 | - | Assign functions at falling edge of quick response signal. Each digit must be set individually and assigns Input 6, 5, 4, 3, 2, 1.<br>0: No function<br>1: Alarm reset/Program Bank Pause<br>2: Enable motor<br>3: Position counter reset<br>4: Execute next Program Bank line<br>5: Execute previous Program Bank line<br>6: Execute Program Bank 1<br>7: Start origin search<br>8: Manual jog CW (K36=2 or 3, execute Program Bank 2)<br>9: Manual jog CCW (K36=2 or 3, execute Program Bank 3) |
| 33 | Output logic | 0000 | 1111 | 1111 | - | Set output logic. Each digit must be set individually and assigns Output 4, 3, 2, 1.<br>0: Output port is ON when Output signal is OFF.<br>1: Output port is ON when Output signal is ON. |

| K | Parameter | Min | Max | Default | Unit | Description |
|---|---|---|---|---|---|---|
| 34 | Output Functions | 0000 | 9999 | 0000 | - | Assign Output Functions. Each digit must be set individually and assigns Output 4, 3, 2, 1.<br>0: No function<br>1: In-position<br>2: Alarm<br>3: General Use<br>4: Completion of origin search<br>5: -<br>6: In-position signal in merge motion<br>7: Rotation pulse output. When both Output 1 and Output 2 are set to 7, quadrature encoder pulse output.<br>8: In motor free<br>9: In push motion |
| 35 | Analog Output Functions | 0 | 9 | 0 | - | 0: Target position<br>1: Target position data magnified by 8<br>2: Current position<br>3: Current position data magnified by 8<br>4: Position error<br>5: Position error data magnified by 8<br>6: Current speed<br>7: Current speed data magnified by 8<br>8: Current torque<br>9: Current torque data magnified by 8 |
| 36 | Command Pulse Format | 0 | 3 | 0 | - | Set P type motor to either CW/CCW mode or pulse/direction mode.<br>Or assign functions at rising/falling edge of input signal.<br>0 or 2 : CW / CCW<br>1 or 3 : Pulse / direction<br>2 or 3 : Enable to execute Program Banks 2 and 3 (except for P type) |
| 37 | Resolution/Speed Unit<br>0~10 :<br>speed unit 100pps<br>20~30 :<br>speed unit 10pps<br>40~50 :<br>speed unit 100pps<br>60~70 :<br>speed unit 10pps<br>80~90 :<br>speed unit 1pps | 0 | 90 | 3 | - | Pulses per rotation and speed unit<br>0, 20, 80 : 200    40, 60 : 300<br>1, 21, 81 : 400    41, 61 : 400<br>2, 22, 82 : 500    42, 62 : 600<br>3, 23, 83 : 1000    43, 63 : 800<br>4, 24, 84 : 2000    44, 64 : 1200<br>5, 25, 85 : 2500    45, 65 : 1500<br>6, 26, 86 : 5000    46, 66 : 3000<br>7, 27, 87 : 10000    47, 67 : 4000<br>8, 28, 88 : 25000    48, 68 : 6000<br>9, 29, 89 : -    49, 69 : 8000<br>10, 30, 90 : 50000    50, 70 : 12000<br>When K37 is set to 40-70, incremental motion is not allowable. |

| K | Parameter | Min | Max | Default | Unit | Description |
|---|---|---|---|---|---|---|
| 38 | Analog Control Type | 0 | 10 | 1 | - | Control Target and method for analog input<br>0 : No function<br>1: Position control<br>2: Speed control for CW<br>3: Speed control for CCW<br>4 : Speed control for CW / CCW<br>5 : P data for Direct Mode<br>6 : S data (+) for Direct Mode<br>7 : S data (-) for Direct Mode<br>8 : S data (+/-) for Direct Mode<br>9 : Torque control<br>10 : Torque feedback control |
| 39 | Low Pass Filter Cut-off Frequency | 0 | 1024 | 128 | 5rad/s | Low pass filter cut-off frequency for analog input |
| 40 | Maximum Speed | 1 | Depends on motor type | Depends on motor type | $min^{-1}$ | The maximum speed of motor. Set the speed when the maximum analog voltage is applied in case of speed control with analog input. |
| 41 | Analog Travel range | -32767 | 32767 | 200 | Pulses | The max. travel range in case of position control with analog input |
| 42 | Origin Search Speed | 1 | 32767 | 10 | 100pps<br>10pps<br>1pps | The speed for origin search |
| 43 | Acceleration for Origin search / Manual feed | 1 | 32767 | 100 | $kpps^2$ | Set the acceleration for origin search and manual feed. |
| 44 | Deceleration Ratio | 10 | 500 | 100 | % | Deceleration ratio is relative to the acceleration in percentage.<br>When K44=100, deceleration is the same as acceleration. |
| 45 | Origin Search Direction, Reverse coordinates | 000 | 223 | 001 | - | · First digit ··· Setting of Origin search Direction and Reverse Coordinates<br>0: CW direction<br>1: CCW direction<br>2: CW direction Reverse Coordinates<br>3: CCW direction Reverse Coordinates<br>· Second digit ··· Unit of offset by K48<br>0: 100 pulses<br>1: 10 pulses<br>2: 1 pulse<br>· Third digit ··· Unit of software limit by K58, K59<br>0: 100 pulses<br>1: 10 pulses<br>2: 1 pulse |

| K | Parameter | Min | Max | Default | Unit | Description |
|---|---|---|---|---|---|---|
| 46 | Origin Signal Source | 0 | 7 | 0 | - | Specify the origin signal source.<br>0: Stopper detection<br>1: Stopper detection (Automatic start when powered ON)<br>2: Origin sensor<br>3: Origin sensor (Automatic start when powered ON)<br>4: Z-phase signal<br>5: Z-phase signal (Automatic start when powered ON)<br>6: Origin sensor & Z-phase signal<br>7: Origin sensor & Z-phase signal (Automatic start when powered ON) |
| 47 | Stopper Detecting Torque for Origin Search | 10 | 150 | 30 | % | The torque where the motor will determine that the stopper detection has been completed. It is relative to the rated torque of the motor in percentage. |
| 48 | Offset distance between mechanical and electrical origins | -32767 | 32767 | 0 | 100 pulses<br>10 pulses<br>1 pulse | Offset between the mechanical and electrical origins. When it is not set to 0, the movement to the electrical origin is automatically performed after the detection of mechanical origin. The speed is the same as the origin search speed set by K42. When set to 0, electrical origin and mechanical origins are the same. *Unit depends on 2nd digit of K45. |
| 49 | Speed for Manual Feed | 1 | 32767 | 10 | 100pps<br>10pps<br>1pps | Speed for manual feed |
| 50 | Feed Pulses for Manual Jog | 1 | 100 | 10 | pulses | Feed pulses for manual jog<br>(Speed and acceleration are set automatically and can not be changed.) |
| 51 | Creeping speed | 0 | 1000 | 0 | 100pps<br>10pps<br>1pps | Creeping speed. |
| 55 | In-position Range | 1 | 100 | 5 | Pulse | In-position range. |
| 56 | Position Error Overflow Threshold Level | 1 | 32767 | 50 | 100 pulses | Threshold level for position error Over Flow |
| 57 | Overload Detection Time | 100 | 5000 | 3000 | msec | Overload alarm is recognized after continuation of overload state more than set time. |
| 58 | Software Limit (+) | 0 | 32767 | 0 | 100 pulses<br>10 pulses<br>1 pulse | Movable limit in plus direction in reference to the origin.<br>When set to 0, no software limit.<br>*Unit depends on 3rd digit of K45. |
| 59 | Software Limit (-) | -32767 | 0 | 0 | 100 pulses<br>10 pulses<br>1 pulse | Movable limit in minus direction in reference to the origin.<br>When set to 0, no software limit.<br>*Unit depends on 3rd digit of K45. |
| 60 | Push Motion Torque Level | 10 | 100 | 30 | % | Torque level for push motion is relative to the rated torque in percentage.<br>When set to odd No., push motion error will not occur. |
| 61 | Push Motion Holding Time | 0 | 30000 | 200 | msec | Time for keeping push motion.<br>(When set to 0, push motion will continue without end) |
| 62 | Ladder Logic Bank No. Executed when Powered ON | 0 | 30 | 0 | - | Ladder Logic Bank No. that is executed automatically when powered ON.<br>When K62=0, Ladder Logic Bank will not be executed. |

| K | Parameter | Min | Max | Default | Unit | Description |
|---|---|---|---|---|---|---|
| 63 | Ladder Logic Bank execution cycle time | 0 | 30000 | 100 | msec | Execution cycle time for Ladder Logic Bank.<br>Ladder Logic Bank will be executed repeatedly with set cycle time. |
| 64 | Status LED Setting | 0 | 1 | 0 | - | Status LED setting, either activated or inactivated<br>0 : Status LED activated<br>1 : Status LED inactivated |
| 65 | Baud Rate between Slave Motors | 0 | 5 | 0 | - | Baud rate between the slave motors on the daisy chain network.<br>0:38.4Kbps, 1:9.6 Kbps, 2:19.2 Kbps, 3:57.6 Kbps, 4:115.2 Kbps, 5:230.4Kbps<br>When K65 of ID1 motor is changed, all K65 values of other motors will be automatically changed.<br>If any motor's K65 except for ID1 is changed, the other motors' K65 values are not changed. |
| 68 | Motor Free when Powered ON | 0 | 1 | 1 | - | Select either servo ON or motor free when powered ON<br>0: Motor free when powered ON<br>1: Servo ON when powered ON |
| 69 | S-Curve Gain | 0 | 1024 | 0 | - | S-curve gain in positioning<br>When 0, motor makes trapezoidal motion. |
| 70 | Delimiter | 0 | 1 | 1 | - | Select the delimiter attached to the end of sent data from Cool Muscle.<br>0: CR<br>1: CRLF |
| 71 | External Encoder Type | 0 | 7 | 0 | - | Set the external encoder type<br>0: No external encoder<br>1: A-phase index<br>2: A-phase index, B-phase rotation direction<br>3: A-phase & B-phase index<br>4: A-phase & B-phase feedback<br>5: A-phase pulse counting<br>6: A-phase pulse counting, B-phase rotation direction<br>7: A-phase & B-phase pulse counting |
| 72 | External Encoder Resolution | 0 | 32767 | 400 | ppr | Resolution of external encoder |
| 73 | Output Pulse Width at Passing Point in Merge Motion | 0 | 1000 | 10 | msec | Output pulse width at passing point in merge motion. |
| 74 | Torque Control P Gain | 0 | 1000 | 100 | - | Proportional gain for the torque control using external torque sensor. |
| 75 | Torque Control I Gain | 0 | 500 | 10 | - | Integral gain for the torque control using external torque sensor. |
| 76 | Input Offset for Torque Sensor | 0 | 500 | 250 | 0.01V | Input offset voltage of the external torque sensor for feedback control |

| K | Parameter | Min | Max | Default | Unit | Description |
|---|---|---|---|---|---|---|
| 77 | Input Range for Torque Sensor | -1000 | 1000 | 200 | 0.01V | Input range of the external torque sensor for feedback control. |
| 78 | Input Address for Modbus Host Communication | -1 | 32767 | 0 | - | Input address of Cool Muscle for the Modbus host communication. |
| 79 | Input Address for Modbus Slave Communication | -1 | 32767 | 0 | - | Input address of Cool Muscle for the Modbus slave communication. |
| 80 | Output Address for Modbus Slave Communication | -1 | 32767 | 0 | - | Output address of Cool Muscle for the Modbus slave communication. |
| 81 | Slave Address | -255 | 255 | 0 | - | Equipment ID number for Modbus or RS-485 communication. |
| 82 | Parity | 0 | 2 | 0 | - | Parity setting for data transmission.<br>0: None<br>1: Even<br>2: Odd |

# 6.2. Data Commands

| Com-mands | Function | unit | Format ( n: Motor ID ) | Example | Description |
|---|---|---|---|---|---|
| P | Position Data Definition | pulses | P#.n=Value<br>#: memory No.<br>*(1~200)<br>P0: direct mode<br>( 0 can be omitted ) | <br><br><br>P.2=9000<br>P1.3=9000 | Define the position data in Motor n's P memory #.<br>Note) The max. values of the position data depends on the resolution setting<br>Ex.) Motor 2's P0 is set to 9000.<br>Define Motor 3's P1 is set to 9000.<br>* Memory range can be changed by allocation (R type only) |
| | Relative Position Data Definition | pulses | P#.n+=1000<br>P1.3-=1000 | <br><br><br><br><br><br><br>P.1+=1000<br>P1.3-=1000 | The value can be defined as relative to the current position by using + or – after Motor ID.<br>Note) In direct mode (using P0), it is relative to the current position.<br>In program mode, it is relative to values defined as P1~P200*.<br>Ex.)<br>Motor 1's P0 is set to [current position + 1000].<br>Define Motor 3's P1 as [current position - 1000]. |
| S | Speed Data Definition | 100pps or 10pps or 1pps | S#.n=value<br>#: memory No. (1~15)<br>S0: direct mode<br>(0 can be omitted) | <br><br><br><br>S.2=100<br>S13.3=150 | Define an absolute speed data in Motor n's S memory #.<br>Note) The negative value is treated as absolute value.<br>Ex.) Define Motor 2's S0 as 100.<br>Define Motor 3's S13 as 150. |
| A | Acceleration Data Definition | kpps² | A#.n=value<br>#: memory No. (1~8)<br>A0: direct mode<br>(0 can be omitted) | <br><br><br><br>A.2=10<br>A6.3=100 | Define the absolute acceleration data in Motor n's A memory #.<br>Note) The negative value is treated as absolute value.<br>Ex.) Define Motor 2's A0 as 10.<br>Define Motor 3's A6 as 100. |
| T | Timer Data Definition | msec | T#.n=value<br>#: memory No. (1~8) | <br>T2.1=500 | Define Timer data in Motor n's T memory #<br>Ex.) Define Motor 1's T2 as 500. |
| M | Torque Limit Data Definition | % | M#.n=value<br>#: memory No. (1~8)<br>M0: direct mode<br>(0 can be omitted) | <br><br>M.2=50<br>M2.3=80 | Define Torque limit data in Motor n's M memory #.<br>(0-100% of Max. motor torque can be set)<br>Ex.) Define Motor 2's direct mode M0 as 50.<br>Define Motor 3's M2 as 80. |

| Com-mands | Function | unit | Format ( n: Motor ID ) | Example | Description |
|---|---|---|---|---|---|
| V | Variable Data Definition | | 1) V#.n=value<br>2) V#.n="Characters"<br>3) V#.n="motor's internal variables"<br>#: memory No. (1∼15)<br>Internal variables :<br>Px, Sx, Ix, Ux, Pe, AIN, PT, ST or CT | V1.2=1234<br>V1.2="ABCD"<br>V1.2="Px" | Define Variable data in Motor n's V memory #.<br>Up to 4 digit number or characters can be used.<br>Note) " double quotation is needed to use characters and motor's internal variables.<br>1) use as a number<br>2) use as character<br>3) use as an internal state values as below<br>Px…current position<br>Sx…current speed<br>Ix…current Iq<br>Ux…current motor status<br>Pe…position error<br>AIN…analog input<br>PT…target position<br>ST…target speed<br>CT…external encoder counter<br>Ex.) Define Motor 2's V1 as 1234<br>Define Motor 2's V1 as ABCD<br>Define Motor 2's V1 as Px (current position) |
| N | Center Point data of Circle Definition | | N#.n=value<br>#: memory No.(1∼200)*<br>N0: direct mode<br>(0 can be omitted) | N.1=100,N.2=100<br>N1.1=100,N1.2=100 | Define center point data of circle in Motor n's N memory #.<br>Ex.) Define center point of circle to 100 pulses for X and Y.<br>* Memory range can be changed by allocation (R type only) |
| R | Radius data of Circle Definition | | R#.n=value<br>#: memory No.(1∼200)*<br>R0: direct mode<br>(0 can be omitted) | R.1=100, R.2=100<br>R1.1=100, R1.2=100 | Define radius data of circle in Motor n's R memory #.<br>When the two values are set to 0, linear interpolation is executed.<br>When the two values are different, elliptic trajectory will be drawn.<br>Ex.) Define radius of circle to 100 pulses for X and Y.<br>* Memory range can be changed by allocation (R type only) |

# 6.3. Program Bank Commands

OP···It shows if it is possible to use with operators or not.

| Com-mands | Function | OP | Format ( n: Motor ID ) | Example | Description |
|---|---|---|---|---|---|
| S | Speed | ○ | S#.n<br>#:memory No. (1~15) | S1.1<br>S1.1= S2.1+ V1.1 | Ex.) Define the specified motion speed as S1.<br>Define the value of S2.1+V1.1 as S1.1. |
| A | Acceleration | ○ | A#.n<br>#:memory No. (1~8) | A1.1<br>A1.1= A2.1+ V2.1 | Ex.) Define the specified motion acceleration as A1.<br>Define the value of A2.1+ V2.1as A1.1. |
| P | Position | ○ | P#.n<br>#:memory No. (1~200)* | P1.1<br>P1.1+<br>P1.1= P3.1+ V3.1 | Ex.) Define target position as P1.<br>Add or subtract P1 to or from the current position and set P1 as the next target position. Define the value of P3.1+ V3.1 as P1.1.<br>* Memory range can be changed by allocation (R type only) |
| Y | Execute next line without in-position queuing | × | Y#.n<br>#:P memory No. | A1.1,S4.1,Y1.1<br>A1.2,S4.2,P1.2 | In Daisy Chain, by using Y command instead of P, the motors move without waiting for in-position of Motor n.<br>Ex.) Motor 2 starts executing the next line without waiting for Motor 1's in-position at P1. |
| Q | Push motion | × | Q#.n<br>#:P memory No. | A1.1,S4.1,Q10.1 | Perform push motion<br>Ex.) Motor 1 performs push motion against P10. |
| Z | Execute next line without push motion completion queuing | × | Z#.n<br>#:P memory No. | A1.1,S4.1,Z1.1<br>A1.2,S4.2,P1.2 | In Daisy Chain, by using Z command instead of Q, the motors move without waiting for completion of specified motor's push motion.<br>Ex.) Motor 2 starts execution the next line without waiting for Motor 1's completion of push motion. |
| M | Torque Limit | ○ | M#.n<br>#:memory No. (1~8) | M1.1<br>M1.1= V5.1+ V6.1 | Define the max. torque in percentage as M# of Motor n.<br>Ex.) The max. torque is set to M1.1. |
| B | Beginning of Program Bank | × | B#.n<br>#: Bank No. (1~30) | B1.1<br>A1.1,S4.1,P12.1 | Define the beginning of a Program Bank and specify the Program Bank number.<br>Note) Program Bank should end with “End”. |
| C | Call other Program Bank | × | C#.n<br>#: Bank No. (1~30) | A1.1,S4.1,P12.1<br>C2.1 | Call the specified Program Bank, execute it and return to the next line of the original Program Bank.<br>Note) It is impossible to call other motor's Program Banks and re-call itself. |
| J | Jump to other Program Bank | × | J#.n<br>#: Bank No. (1~30) | B1.1<br>J2.1 | Jump to the specified Program Bank, execute it and can not return to the next line of the original Program Bank.<br>Note) It is impossible to jump to other motor's Program Banks. |

| Com-mands | Function | OP | Format<br>( n: Motor ID ) | Example | Description |
|---|---|---|---|---|---|
| X | Looping | × | X#.n<br>X#.n<br>~<br>X.n-<br>X.n-<br># is loop count<br>n must be the same as Motor ID of B command. | X0.1<br>A1.1, S1.1, P1.1<br>X2.1<br>P1.1<br>P2.1<br>X.1-<br>X.1- | Execute the lines between X and X- repeatedly up to loop count (1~255).<br>When X0 is set, it loops infinitely.<br>Note) When X-is not placed, lines after X until the end of Program Bank will be looped.<br>· A part of Program Bank can be executed repeatedly.<br>· Up to 10 nestings of X loop are available. |
| I | Conditional Branching on Input Status | | [expression], [action 1], [action 2] | | Depending on the result of operation in expression, if TRUE, execute action 1.<br>if FALSE execute action 2. |
| | Branching on Condition of Single Input | ○ | I#.n,[action 1], [action 2]<br>#:Input No. | I1.1, C2.1, C3.1 | Execute a specified motion according to Input # status.<br>Ex.) If Input 1 is ON (TRUE), Motor 1 calls Program Bank 2. If OFF (FALSE), Motor 1 calls Program Bank 3. |
| | Branching on Condition of 2 Inputs Operation | | I#.n \ I #.n, [action 1], [action 2]<br>\ : Logical Operator | I3.2 && I4.1,?99,?98 | Ex.) If I3.2 & I4.1 are ON (TRUE), Motor 1 executes ?99. If FALSE, Motor 1 executes ?98. |
| V | Conditional Branching on Variable | | [expression], [action 1], [action 2] | | Depending on the result of operation in expression, if TRUE, execute action 1.<br>if FALSE execute action 2. |
| | Branching on Condition of Single Variable | ○ | V#.n,[action 1], [action 2]<br>n must be the same as Motor ID of B command. | V1.1,?99,?98 | For single Variable, the operation of V>0 is applied.<br>Ex.) If V1.1>0, Motor 1 executes ?99.<br>Otherwise, Motor 1 executes ?98. |
| | Branching on Condition of 2 Variables | | V#.n \ V#.n,[action 1], [action 2]<br>\ : Operator<br>n must be the same as Motor ID of B command. | V1.1> V2.1, ?99, ?98<br>V1.1== V2.1, ?99, ?98 | Ex.) If V1.1>V2.1, Motor 1 executes ?99.<br>If V1.1≦V2.1, Motor 1 executes ?98.<br>If V1.1=V2.1, Motor 1 executes ?99.<br>If V1.1≠V2.1, Motor 1 executes ?98. |
| T | Timer | ○ | T#.n<br>n must be the same as Motor ID of B command. | T1.1 | Wait for the time defined by T data.<br>T0 means no action. |
| W | Timer in Conditional Branching | × | W#.n<br>#:T memory No.<br>n must be the same as Motor ID of B command. | I4.1,W1.1,?99<br>A1.1, S1.1, P1.1 | Wait for event to happen for the time defined by T data. If set to W0, then wait continuously.<br>Ex.) While I4.1 is TRUE, Motor 1 waits for the time set by T1. After the time is up, Motor 1 executes next line.<br>If I4.1 turns FALSE during the time set by T1, Motor 1 executes ?99 instantly and then next line. |
| N | Center Point of Circle | ○ | N#.n,N#.n | N1.1,N1.2 | Set the center point of circle to (N1.1, N1.2) (multi-axis application) |
| R | Radius of Circle | ○ | R#.n,R#.n | R1.1,R1.2 | Set X axis' radius of circle to R1.1, and set Y axis' radius to R1.2. (multi-axis application) |
| Execution Command | Refer to 6.5 Chapter | × | Refer to 6.5 Chapter | Refer to 6.5 Chapter | Execution commands can be used within Program Bank. |
| END | End of Program Bank | × | END | END | Define the end of Program Bank. |

| Symbol | Function | Format<br>( n: Motor ID ) | Example | Description |
|---|---|---|---|---|
| // | Comment | Command line // Comment | B1.1//comment | Comments can be written after “//”. |
| ,<br>(comma) | Command Concatenation / Merge Motion / Simultaneous Motion Execution | Command, Command | A1.1, S1.1, P1.1<br><br>A1.1, S1.1, P1.1, S2.1, P2.1<br><br>P1.1, P3.2 | Command concatenation : Multiple commands can be described in a single line.<br>Merge motion : Motor 1 moves to P2 without stopping at P1 smoothly, with speed change to S2 when passing P1.<br>Simultaneous motion : Motor 1 and 2 will start their motion at the same time. |
| ;<br>(semi colon) | Command Concatenation in Multiple Lines | Command; Command | A1.1,S1.1,P1.1;<br>S2.1,P2.1 | By using semicolon instead of comma, multiple commands and merge motion can be described in multiple lines. |
| :<br>(colon) | Command Concatenation in Branching | Command:Command | V1>V2, ?99.1: O1.1, ?96.1: F1.1 | Colon allows the use of multiple commands in branching processing.<br>Ex.) If V1>V2, Motor 1 executes ?99 and O1.1. If V1<=V2, motor 1 executes ?96 and F1.1. |

# 6.4. Ladder Logic Bank Commands

OP・・・It shows if it is possible to use with operators or not.

| Com-mands | Function | OP | Format ( n: Motor ID ) | Example | Description |
|---|---|---|---|---|---|
| L | Begging of Ladder Logic Bank | × | L#.n<br>#: Bank No. (1∼30) | L1.1 | Define the beginning of a Ladder Logic Bank and specify Ladder Logic Bank number.<br>Note) Ladder Logic Bank should end with “End”. |
| CL | Call other Ladder logic Bank | × | CL#.n<br>#: Bank No. (1∼30) | CL2.1 | Call the specified Ladder Logic Bank, execute it and return to the next line of the original Ladder Logic Bank.<br>Note) It is impossible to call other motor's Ladder Logic Banks and re-call itself. |
| JL | Jump to other Ladder Logic Bank | × | JL#.n<br>#: Bank No. (1∼30) | JL2.1 | Jump to the specified Ladder Logic Bank, execute it and can not return to the next line of the original Ladder Logic Bank.<br>Note) It is impossible to jump to other motor's Ladder Logic Banks. |
| I | Conditional Branching on Input Status | ○ | [expression], [action 1], [action 2] | | Depending on the result of operation in expression, if TRUE, execute action 1.<br>if FALSE execute action 2. |
| | Branching on Condition of Single Input | | I#.n,[action 1], [action 2]<br>#:Input No. | I1.1, CL2.1, CL3.1 | Execute a specified motion according to Input # status.<br>Ex.) If Input 1 is ON (TRUE), Motor 1 calls Ladder Logic Bank 2. If OFF (FALSE), Motor 1 calls Ladder Logic Bank 3. |
| | Branching on Condition of 2 Inputs Operation | | I#.n \ I #.n, [action 1], [action 2]<br>\ : Logical Operator | I3.2 && I4.1,?99,?98 | Ex.) If I3.2 & I4.1 are ON (TRUE), Motor 1 executes ?99. If FALSE, Motor 1 executes ?98. |
| V | Conditional Branching on Variable | ○ | [expression], [action 1], [action 2] | | Depending on the result of operation in expression, if TRUE, execute action 1.<br>if FALSE execute action 2. |
| | Branching on Condition of Single Variable | | V#.n,[action 1], [action 2]<br>n must be the same as Motor ID of L command. | V1.1,?99,?98 | For single Variable, the operation of V>0 is applied.<br>Ex.) If V1.1>0, Motor 1 executes ?99.<br>Otherwise, Motor 1 executes ?98. |
| | Branching on Condition of 2 Variables | | V#.n \ V#.n,[action 1], [action 2]<br>\ : Operator<br>n must be the same as Motor ID of L command. | V1.1> V2.1, ?99, ?98<br>V1.1== V2.1, ?99, ?98 | Ex.) If V1.1>V2.1, Motor 1 executes ?99.<br>If V1.1≦V2.1, Motor 1 executes ?98.<br>If V1.1=V2.1, Motor 1 executes ?99.<br>If V1.1≠V2.1, Motor 1 executes ?98. |
| T | Timer | ○ | T#.n<br>n must be the same as Motor ID of L command. | T1.1 | Wait for the time defined by T data.<br>T0 means no action. |
| W | Timer in Conditional Branching | × | W#.n<br>#:T memory No.<br>n must be the same as Motor ID of L command. | I4.1,W1.1,?99<br>CL3.1 | Wait for event to happen for the time defined by T data. If set to W0, then wait coutinuously.<br>Ex.) While I4.1 is TRUE, Motor 1 waits for the time set by T1. After the time is up, Motor 1 executes next line.<br>If I4.1 turns FALSE during the time set by T1, Motor 1 executes ?99 instantly and then next line. |
| # | Capture Position Data | ○ | #x.n<br>x is P memory No. | #2.1 | Capture the current position value and store it to the specified motor's specified P memory. |
| Execution Command | Refer to 6.5 Chapter | × | Refer to 6.5 Chapter | Refer to 6.5 Chapter | Execution commands can be used within Ladder Logic Bank. |
| END | | × | END | END | Define the end of Ladder Logic Bank. |

| Symbol | Function | Format ( n: Motor ID ) | Example | Description |
|---|---|---|---|---|
| // | Comment | Command line // Comment | B1.1//comment | Comments can be written after “//”. |
| , (comma) | Command Concatenation | Command, Command | V2.1>V3.1, V2.1=V3.1, T0.1 | Command concatenation : Multiple commands can be described in a single line. |
| ; (semi colon) | Command Concatenation in Multiple Lines | Command; Command | V2.1>V3.1;<br>V2.1=V3.1, T0.1 | By using semicolon instead of comma, multiple commands can be described in multiple lines. |
| : (colon) | Command Concatenation in Branching | Command:Command | V1>V2,?99.1:O1.1,?96.1: F1.1 | Colon allows the use of multiple commands in branching processing.<br>Ex.) If V1>V2, Motor 1 executes ?99 and O1.1. If V1<=V2, motor 1 executes ?96 and F1.1. |

# 6.5. Execution Commands

P ··· Program Bank, L ··· Ladder Logic Bank, D ··· Direct Mode, indicate the availability of command.

| Com-mands | Function | P | L | D | Format ( n: Motor ID ) | Example | Description |
|---|---|---|---|---|---|---|---|
| \| | Origin Search | ○ | ○ | ○ | \|.n | \|.1 | Origin Search starts. |
| \|1 | Move to Position 0 | ○ | ○ | ○ | \|1.n | \|1.2 | Move to position 0 with the speed and acceleration set by K42 and K43. |
| \|2 | Assign Current Position to 0 | ○ | ○ | ○ | \|2.n | \|2.3 | Assign current position to 0.<br>Set Motor 3's current position to 0. |
| ( | Enable Motor | ○ | ○ | ○ | (.n | (.1 | Enable motor (Servo ON). |
| ) | Motor Free | ○ | ○ | ○ | ).n | ).1 | Motor free. |
| [ | Execute Program Bank | ○ | ○ | ○ | [#.n<br>#: Bank # | [1.2 | Execute the specified Program Bank.<br>Only [ resumes the execution of the Program Bank paused right before. |
| ] | Pause Program Bank | ○ | ○ | ○ | ] | ]CR : pause<br>]CR ]CR : stop | This command stops all motors and pauses Program Bank in operation.<br>Send the command twice to terminate the Program Bank. |
| ]1 | Pause Specified Motor | × | × | ○ | ]1.n | ]1.2 | Pause only specified motor in Daisy Chain connection. In direct mode, pause only Motor 2 in Daisy Chain connection. |
| [L | Execute Ladder Logic Bank | × | ○ | ○ | [L#.n<br>#: Bank # | [L1.1 | Execute the specified Ladder Logic Bank.<br>Only [L restarts the execution of the paused Ladder Logic Bank. |
| ]L | Pause Ladder Logic Bank | × | ○ | ○ | ]L.n | ]L.1CR : pause<br>]L.1CR ]L.1CR : stop | Pause specified Ladder Logic Bank.<br>Send the command once to pause the Ladder Logic Bank.<br>Send the command twice to stop the Ladder Logic Bank. |
| * | Emergency Stop | ○ | ○ | ○ | * | * | Emergency stop of operation with the max. deceleration.<br>Send the command once to pause the Program Bank.<br>Send the command twice to terminate the Program Bank. |
| *1 | Cancel Emergency Stop | ○ | ○ | ○ | *1 | *1 | Cancel emergency stop |
| ^ | Execute the Direct Mode Motion | × | × | ○ | ^.n | ^.1 | Execute the motion in Direct Mode. |
| O | Output Signal ON | ○ | ○ | ○ | O#.n<br>#: Output # | O2.1 | Turn the specified output signal ON.<br>Output signal 2 of Motor 1 is turned ON. |
| F | Output Signal OFF | ○ | ○ | ○ | F#.n<br>#: Output # | F2.1 | Turn the specified output signal OFF.<br>Output signal 2 of Motor 1 is turned OFF. |
| > | Execute Next Line | × | × | ○ | >.n | >.2 | Execute the next line of Program Bank in pause. |
| < | Execute Previous Line | × | × | ○ | <.n | <.2 | Execute the previous line of Program Bank in pause. |
| } | Stop after Completing Current Line | × | × | ○ | }.n | }.1 | Stop motor after completing the current line in Program Bank. |
| $ | Save data | × | × | ○ | $.n | $.1 | Save the data into a specified motor's memory. |
| ? | Query | ○ | ○ | ○ | ?No. | ?96 | Please refer to section 6.6 |
| # | Capture Position Data | ○ | ○ | ○ | #x.n<br>x is P memory No. | #2.1 | Capture the current position value and store it to the specified motor's specified P memory. |
| @ | Execute Circular and Linear Interpolation | ○ | ○ | ○ | @#.n,@#.n | @1.1,@1.2 | Motors execute interpolation motion target to the points (P1.1,P1.2).<br>Only interpolation type can be used. |
| \ (¥or₩) | Allocation of Data Area | × | × | ○ | \P numeric<br>\N numeric | \P300<br>\N200 | The area for 600 data in total is allocated for P, N and R.<br>Only interpolation type can be used. |

## 6.6. Query

Queries can be used in Direct Mode, Program Bank and Ladder Logic Bank.

| Command | Query item | Format<br>( n: Motor ID ) | Response |
|---|---|---|---|
| ? | Direct Mode Data | ?.n | The predefined A,S and P data for Direct mode.<br>Example: ?.1<br>Predefined data of Direct mode of Motor 1?<br>Response data example:<br>S.1=500, A.1=2000, P.1=100000 |
| ?1～30 | Program Bank | ?#.n<br>#: Program Bank No. 1～30 | Predefined program banks 1-30.<br>Example: ?1.1<br>Predefined Program Bank 1 of Motor 1?<br>Response data example:<br>S1.1, A1.1, P1.1<br>P2.1<br>(Only the predefined content after B#.n) |
| ?50 | Output Signal | ?50.n | Current status of all outputs in hexadecimal.<br>Example: ?50.1<br>All the output current status of Motor 1?<br>Response data example:<br>OUT.1=03<br>* 03 means 0011 in binary number and 0 (OFF) or 1 (ON) is responded by one column of unit in order of Out4, 3, 2, 1. |
| ?51 | Output Signal 1 | ?51.n | Current status of output signal 1 by 0 (OFF) or 1 (ON).<br>Example:<br>?51.1<br>Response data example:<br>OUT1.1=0 |
| ?52 | Output Signal 2 | ?52.n | Current status of output signal 2 by 0 (OFF) or 1 (ON).<br>Example:<br>?52.1<br>Response data example:<br>OUT2.1=0 |
| ?53 | Output Signal 3 | ?53.n | Current status of output signal 3 by 0 (OFF) or 1 (ON).<br>Example:<br>?53.1<br>Response data example:<br>OUT3.1=0 |
| ?54 | Output Signal 4 | ?54.n | Current status of output signal 4 by 0 (OFF) or 1 (ON).<br>Example:<br>?54.1<br>Response data example:<br>OUT4.1=0 |

| Command | Query item | Format<br>( n: Motor ID ) | Response |
|---|---|---|---|
| ?70 | Input Signal | ?70.n | Current status of all inputs in hexadecimal.<br>Example: ?70.1<br>All the input current status of Motor 1?<br>Response data example:<br>IN.1=1C<br>* 1C means 011100 in binary number and 0 (OFF) or 1 (ON) is responded by one column of unit in order of In6, 5, 4, 3, 2, 1. |
| ?71 | Temperature in Driver Case | ?71.n | Temperature inside the driver case<br>Example: ?71.1<br>Temperature inside the driver case of Motor 1?<br>Response data example:<br>Temp.1=40 (Unit : ℃) |
| ?72 | Power Supply Voltage | ?72.n | Current power supply voltage level<br>Example: ?72.1<br>Current power supply voltage level of Motor 1?<br>Response data example:<br>VSEN.1=1400 (Unit : 0.1V) |
| ?74 | Analog Input | ?74.n | Analog input voltage value<br>0-5V is divided by 1024 and respond 0 when 0V and 1023 when 5V is applied.<br>Example: ?74.1<br>Analog input voltage value of Motor 1?<br>Response data example:<br>ADI0.1=512 (represents 2.5V) (Unit : 5/1023V) |
| ?76 | External Encoder Counter | ?76.n | Value of counter for an external encoder.<br>Example: ?76.1<br>Response data example:<br>Ecnt.1=100 |
| ?85 | Version Title | ?85.n | Version title<br>Example: ?85.1<br>Version title of Motor 1?<br>Response data example<br>ID1 : CM2v3.10R.1 #0802A12345<br>Serial No.<br>Hardware No.<br>Type<br>Firmware Version<br>Series Name<br>ID No. |

| Command | Query item | Format ( n: Motor ID ) | Response |
|---|---|---|---|
| ?90 | User Parameter | ?90.n | User parameter K20 ~K89<br>Example: ?90.1<br>User parameter's of Motor 1?<br>Response data example:<br>K20.1=0, K21.1=0, K22.1=200, K23.1=1<br>………………..<br>K88.1=0, K89.1=0<br>4 parameters in 1 line, each is separated with a comma(,). |
| ?95 | Position Error | ?95.n | Position error value<br>Example: ?95.1<br>Position error value of Motor 1?<br>Response data example:<br>Pe.1=0 (Unit : pulse) |
| ?96 | Current position | ?96.n | Current position<br>Example: ?96.1<br>Current position of Motor 1?<br>Response data example:<br>Px.1=10000 (Unit : pulse) |
| ?97 | Current speed | ?97.n | Current speed<br>Example: ?97.1<br>Current speed of Motor 1?<br>Response data example:<br>Sx.1=100 (Unit : 100pps/10pps/1pps) |
| ?98 | Current torque | ?98.n | Current torque<br>Example: ?98.1<br>Current torque of Motor 1?<br>Response data example:<br>Ix.1=20 |
| ?99 | Motor Status | ?99.n | Current status<br>Example: ?99.1<br>Response data example:<br>Ux.n=0 motor is running<br>Ux.n=1 position error over flow<br>Ux.n=2 over speed/regenerative voltage<br>Ux.n=4 overload<br>Ux.n=8 In-position<br>Ux.n=16 motor free<br>Ux.n=32 push motion<br>Ux.n=40 push motion completed<br>Ux.n=64 power module over current<br>Ux.n=128 temperature alarm<br>Ux.n=256 push motion error<br>Ux.n=512 emergency stop<br>Multiple status can be responded by addition of above numbers. |
| V1~15 | Specified Variable Data | V#.n<br>#:memory No. | Value of specified V (Variables) memory |

The commands below can not be used in Program Bank and Ladder Logic Bank.

| Command | Query item | Format ( n: Motor ID ) | Response |
|---|---|---|---|
| ?L1~30 | Ladder Logic Bank | ?L#.n<br>#: Ladder Logic Bank No. | Predefined Ladder Logic Banks.<br>Example: ?L1.1<br>Predefined Ladder Logic Bank 1 of Motor 1?<br>Response data example:<br>I1.1&&I2.1, O1.1, O2.1<br>I3.1\|\|I4.1, O3.1, O4.1<br>(Only predefined content after L#.n) |
| A1~8 | Specified Acceleration Data | A#.n<br>#: memory No. | Value of specified A ( Acceleration) memory No.<br>Example: A1.1<br>Predefined acceleration data 1 of Motor 1?<br>Response data example:<br>A1.1=100 |
| S1~15 | Specified Speed Data | S#.n<br>#: memory No. | Value of specified S (Speed) memory No.<br>Example: S1.1<br>Predefined speed data 1 of Motor 1?<br>Response data example:<br>S1.1=500 |
| M1~8 | Specified Torque Limit Data | M#.n<br>#: memory No. | Value of specified M (Torque Limit) memory No.<br>Example: M1.1<br>Predefined torque limit data 1 of Motor 1?<br>Response data example:<br>M1.1=10000 |
| T1~8 | Specified Timer Data | T#.n<br>#: memory No. | Value of specified T (Timer) memory No.<br>Example: T1.1<br>Predefined timer data 1 of Motor 1?<br>Response data example:<br>T1.1=1000 |
| P1~200 | Specified Position Data | P#.n<br>#: memory No. | Value of specified P (Position) memory No.<br>Example: P1.1<br>Predefined position data 1 of Motor 1?<br>Response data example:<br>P1.1=100<br>Only R type, P data up to P600 is available by allocation. |
| N1~200 | Specified Center Point of Circle Data | N#.n<br>#: memory No. | Value of specified N (Center Point of Circle) memory No.<br>Example: N1.1<br>Predefined center point of circle data 1 of Motor 1?<br>Response data example:<br>N1.1=100<br>Only R type, N data up to N600 is available by allocation. |

| Command | Query item | Format ( n: Motor ID ) | Response |
|---|---|---|---|
| R1~200 | Specified Radius of Circle Data | R#.n<br>#: memory No. | Value of specified R (Radius of Circle) memory No.<br>Example: R1.1<br>Predefined radius of circle data 1 of Motor 1?<br>Response data example:<br>R1.1=100<br>Only R type, R data up to R600 is available by allocation. |
| ?A | All Acceleration Data | ?A.n | All acceleration data<br>4 parameters in 1 line, each is separated with a comma(,). |
| ?S | All Speed Data | ?S.n | All speed data<br>4 parameters in 1 line, each is separated with a comma(,). |
| ?M | All Torque Limit Data | ?M.n | All torque limit data<br>4 parameters in 1 line, each is separated with a comma(,). |
| ?T | All Timer Data | ?T.n | All timer data<br>4 parameters in 1 line, each is separated with a comma(,). |
| ?V | All Variable Data | ?V.n | All variable data<br>4 parameters in 1 line, each is separated with a comma(,). |
| ?P | All Position Data | ?P.n | All position data<br>4 parameters in 1 line, each is separated with a comma(,). |
| ?N | All Center Point of Circle Data<br>*Available with R type only | ?N.n | All center point of circle data<br>4 parameters in 1 line, each is separated with a comma(,). |
| ?R | All Radius of Circle Data<br>*Available with R type only | ?R.n | All radius of circle data<br>4 parameters in 1 line, each is separated with a comma(,). |
| ?999 | All Data List | ?999.n | All data of P, S, A, T, M, N, R, V |
| ?1000 | All Banks | ?1000.n | All Program Banks and Ladder Logic Banks |

# 6.7. Arithmetic Operators

These operators perform mathematical calculations.

Any number is required to be integer and defined value as in P(positio data) or V(variable).

| Operator | Functions | Format | Examples | Description |
|---|---|---|---|---|
| = | Sets value | [variable] = [expression] | V1.1=V2.1<br>P1.1=P2.1+P3.1 | = Operator assigns the value on its right to the variable on its left.<br>Ex.) When V2.1=50, V1.1is assigned to 50<br>When P2.1=1000, P3.1=2000, then P1.1=3000 |
| + | Addition | [number1] + [number2] | P1.1=P2.1 + V1.1 | + Operator adds two numbers. The result is their arithmetic sum.<br>Ex.) When P2.1=1000, V1.1=300, then P1.1=1300 |
| - | Subtraction | [number1] - [number2] | P1.1=P2.1 - V1.1 | - Operator returns the difference between two numbers. The result is calculated by subtracting number2 from number1.<br>Ex.) When P2.1=1000, V1.1=300, then P1.1=700 |
| * | Multiplication | [number1] * [number2] | P1.1=P2.1 * V1.1 | * Operator multiplies two numbers. The result is the product of number1 and number2.<br>Ex.) When P2.1=100, V1.1=30, then P1.1=3000 |
| / | Division | [number1] / [number2] | P1.1=P2.1 / V1.1 | / Operator divides two numbers. The result is the quotient of number1 divided by number2, not including any remainder. The decimal fraction part is truncated.<br>Ex.) When P2.1=6000, V1.1=20, then P1.1=300 |
| U1 | Sine | U1([number]) | P1.1=U1(V1.1) | U1 Operator returns 10000 times value of sine operation result in integer as following expression.<br>The decimal fraction part is truncated.<br>$U1(\theta) = 10000 \times \sin(2\pi \times \frac{\theta}{36000})$<br>θis data as V value (Unit:0.01degrees)<br>Ex.) When V1.1=3000 (30 degrees),<br>P1.1= U1(V1.1) = 10000*sin(2π×100/36000) = 5000 |
| U2 | Cosine | U2([number]) | P2.1=U2(V1.1) | U2 Operator returns 10000 times value of cosine operation result in integer as following expression.<br>The decimal fraction part is truncated.<br>$U2(\theta) = 10000 \times \cos(2\pi \times \frac{\theta}{36000})$<br>θis data as V value (Unit:0.01degrees)<br>Ex.) When V1.1=3000 (30 degrees),<br>P2.1=U2(V1.1) = 10000*cos(2π×100/36000) = 8660 |
| U3 | Square Root | U3([number]) | P3.1=U3(V1.1) | U3 Operator returns value of square-root operation result in integer.<br>The decimal fraction part is truncated.<br>$U3(\chi) = \sqrt{\chi}$<br>χ is data as V value (Integer)<br>Ex.) When V1.1=100, P3.1=U3(V1.1)=10 |

# 6.8. Logic Operators

<table>
<tr><th>Operator</th><th>Functions</th><th>Format</th><th>Examples</th><th>Description</th></tr>
<tr><td>&&</td><td>And</td><td>[operand1] && [operand2]</td><td>I4.1 && I3.2</td><td>And(&&) Operator performs a logical conjunction on two Boolean operands.<br>Result is True if and only if both operand1 and operand2 evaluate to True.<br>The following table illustrates how result is determined.
<table>
<tr><th>operand1</th><th>operand2</th><th>the value of result</th></tr>
<tr><td>TRUE</td><td>TRUE</td><td>TRUE</td></tr>
<tr><td>TRUE</td><td>FALSE</td><td>FALSE</td></tr>
<tr><td>FALSE</td><td>TRUE</td><td>FALSE</td></tr>
<tr><td>FALSE</td><td>FALSE</td><td>FALSE</td></tr>
</table>
Ex.) If I4.1=TRUE, I3.2=TRUE, then result is TRUE</td></tr>
<tr><td>||</td><td>Or</td><td>[operand1] || [operand2]</td><td>I4.1 || I3.2</td><td>Or(||) Operator performs an inclusive logical disjunction on two Boolean operands.<br>Result is False if and only if both operand1 and operand2 evaluate to False.<br>The following table illustrates how result is determined.
<table>
<tr><th>operand1</th><th>operand2</th><th>the value of result</th></tr>
<tr><td>TRUE</td><td>TRUE</td><td>TRUE</td></tr>
<tr><td>TRUE</td><td>FALSE</td><td>TRUE</td></tr>
<tr><td>FALSE</td><td>TRUE</td><td>TRUE</td></tr>
<tr><td>FALSE</td><td>FALSE</td><td>FALSE</td></tr>
</table>
Ex.) If I4.1=FALSE, I3.2=FALSE, then result is FALSE</td></tr>
<tr><td>!!</td><td>Negation</td><td>!!(operand)</td><td>!!(I3.2)<br>I4.1 && !!(I3.2)</td><td>Not(!!) Operator performs a logical negation on a Boolean operand.<br>The following table illustrates how result is determined.
<table>
<tr><th>operand</th><th>the value of result</th></tr>
<tr><td>TRUE</td><td>FALSE</td></tr>
<tr><td>FALSE</td><td>TRUE</td></tr>
</table>
Ex.) If I3.2=TRUE, then result is FALSE<br>If I4.1=TRUE, I3.2=FALSE, then result is TRUE<br>[ Correct example ]<br>I2.1 && I4.1 && !!(I3.1), O1.1, F1.1<br>If Input Signal 2 and Input Signal are ON(TRUE), and Input Signal 3 is OFF(FALSE), then Output Signal 1 turns ON(TRUE).<br>Otherwise Output Signal 1 is OFF(FALSE).<br>[ Incorrect example ]<br>I2.1 && !!(I3.1) && I4.1, O1.1, F1.1<br>* The next character following && !!( ) strings should be "," in a statement.<br>I2.1 && !!(I3.1) && !!(I4.1), O1.1, F1.1<br>* Multiple Not(!!) Operators are not permissible in a statement.</td></tr>
</table>

## 6.9. Comparison Operators

These operators compare two numbers to determine whether or not they meet the conditions and return the results of comparison.

The value representing the result of the comparison is Boolean. Any number is required to be integer and defined value in V(variable)

| Operator | Functions | Format | Examples | Description |
|---|---|---|---|---|
| == | Equal to | [number1] == [number2] | V1.1 == V2.1<br>V1.1 == V2.1 + V3.1 | Result is TRUE if number1 is equal to number2.<br>Otherwise FALSE.<br>Ex.) If V1.1=100, V2.1=100, then TRUE<br>If V1.1=100, V2.1=50, V3.1=50, then TRUE |
| != | Not Equal to | [number1] != [number2] | V1.1 != V2.1<br>V1.1 != V2.1 + V3.1 | Result is TRUE if number1 is not equal to number2.<br>Otherwise FALSE.<br>Ex.) If V1.1=100, V2.1=100, then FALSE<br>If V1.1=100, V2.1=50, V3.1=50, then FALSE |
| > | Greater than | [number1] > [number2] | V1.1 > V2.1<br>V1.1 > V2.1 + V3.1 | Result is TRUE if number1 is greater than number2.<br>Otherwise FALSE.<br>Ex.) If V1.1=110, V2.1=100, then TRUE<br>If V1.1=100, V2.1=50, V3.1=50, then FALSE |
| >= | Greater than or equal to | [number1] >= [number2] | V1.1 >= V2.1<br>V1.1 >= V2.1 + V3.1 | Result is TRUE if number1 is greater than or equal to number2.<br>Otherwise FALSE.<br>Ex.) If V1.1=110, V2.1=100, then TRUE<br>If V1.1=100, V2.1=50, V3.1=70, then FALSE |
| < | Smaller than | [number1] < [number2] | V1.1 < V2.1<br>V1.1 < V2.1 + V3.1 | Result is TRUE if number1 is less than number2.<br>Otherwise FALSE.<br>Ex.) If V1.1=110, V2.1=100, then FALSE<br>If V1.1=100, V2.1=50, V3.1=70, then TRUE |
| <= | Smaller than or equal to | [number1] <= [number2] | V1.1 <= V2.1<br>V1.1 <= V2.1 + V3.1 | Result is TRUE if number1 is less than or equal to number2.<br>Otherwise FALSE.<br>Ex.) If V1.1=110, V2.1=100, then FALSE<br>If V1.1=100, V2.1=50, V3.1=50, then TRUE |

The following table contains a list of the relational comparison operators and the conditions that determine whether result is TRUE or FALSE.

| Operator | TRUE if | FALSE if |
|---|---|---|
| == | number1 == number2 | number1 != number2 |
| != | number1 != number2 | number1 == number2 |
| > | number1 > number2 | number1 <= number2 |
| >= | number1 >= number2 | number1 < number2 |
| < | number1 < number2 | number1 >= number2 |
| <= | number1 <= number2 | number1> number2 |

# Revision History

* User's Guide No. is described in the cover of this manual.

| Revised Date | User's Guide No. | Page | Object | Revised Item |
|---|---|---|---|---|
| May, 2007 | MDUG-CML/07525E-01 | | | New Draft |
| Feb., 2008 | MDUG-CML/08215E-01 | CH 3-31～33 | K26 | Parameter name and Description are changed. |
| | | CH 3-35 | K28, K31 | K36=2 change to K36=2 or 3. |
| | | CH 3-36 | K29, K32 | |
| | | CH 3-40 | K36 | Description is changed. |
| | | CH 3-51 | K45 | Function of setting unit is added. |
| | | CH 3-54 | K48 | Unit is changed. |
| | | CH 3-60 | K57 | "80% of peak torque" change to "Rated torque". |
| | | CH 3-61 | K58, K59 | Unit is changed. |
| | | CH 3-62 | K60 | Description of when set to odd No. is added. |
| | | CH 5-96 | K46, K47 | K45=1 change to K45=**1,<br>K45=0 change to K45=**0. |
| | | CH 6-116 | K26 | Max value and Description are changed. |
| | | CH 6-117, 118 | K28, K29<br>K31, K32 | K36=2 change to K36=2 or 3. |
| | | CH 6-119 | K36 | Max value and Description are changed. |
| | | CH 6-120～123 | K41, K51,<br>K60, K63,<br>K68, K72,<br>K77 | Default value is changed. |
| | | CH 6-120 | K45 | Description of 2nd digit and 3rd digit is added. |
| | | CH 6-121 | K48,<br>K58, K59 | Unit is changed.<br>Description of setting unit is added. |
| | | | K60 | Description of when set to odd No. is added. |
| | | CH 6-133 | ?85 | Serial No. is added. |
| Apr., 2008 | MDUG-CML/08215E-02 | CH 6-135 | M | M1 ～ 7 change to M1 ～ 8. |
| Jan., 2009 | MDUG-CML/09101E-01 | CH 2-013 | P | Caution is added. |
| | | CH 2-016 | P | Caution is added. |
| | | CH 3-075 | K81 | Caution is added. |
| | | CH 4-088 | 4.4 | Description of merge motion during even at the interpolation is added. |
| | | CH 6-120 | K45 | Default value is changed. |
| | | CH 6-121 | K48,<br>K58, K59 | "*Unit depends on *** digit of K45." is added. |