FS100 OPTIONS
INSTRUCTIONS

(API Function Specifications)

Upon receipt of the product and prior to initial operation, read these instructions thoroughly, and retain for future reference.

MOTOMAN INSTRUCTIONS
MOTOMAN- INSTRUCTIONS
FS100 INSTRUCTIONS
FS100 OPERATOR'S MANUAL
FS100 MAINTENANCE MANUAL

Part Number: 162001-1CD
Revision: 9
MANDATORY

- This manual explains MotoPlus of the FS100 system. Read this manual carefully and be sure to understand its contents before handling the FS100.
- General items related to safety are listed in the Chapter 1: Safety of the FS100 Instructions. To ensure correct and safe operation, carefully read the FS100 Instructions before reading this manual.

CAUTION

- Some drawings in this manual are shown with the protective covers or shields removed for clarity. Be sure all covers and shields are replaced before operating this product.
- The drawings and photos in this manual are representative examples and differences may exist between them and the delivered product.
- Yaskawa may modify this model without notice when necessary due to product improvements, modifications, or changes in specifications. If such modification is made, the manual number will also be revised.
- If your copy of the manual is damaged or lost, contact a Yaskawa representative to order a new copy. The representatives are listed on the back cover. Be sure to tell the representative the manual number listed on the front cover.
- Yaskawa is not responsible for incidents arising from unauthorized modification of its products. Unauthorized modification voids your product’s warranty.
We suggest that you obtain and review a copy of the ANSI/RIA National Safety Standard for Industrial Robots and Robot Systems (ANSI/RIA R15.06-2012). You can obtain this document from the Robotic Industries Association (RIA) at the following address:

Robotic Industries Association
900 Victors Way
P.O. Box 3724
Ann Arbor, Michigan 48106
TEL: (734) 994-6088
FAX: (734) 994-3338
www.roboticsonline.com

Ultimately, well-trained personnel are the best safeguard against accidents and damage that can result from improper operation of the equipment. The customer is responsible for providing adequately trained personnel to operate, program, and maintain the equipment. NEVER ALLOW UNTRAINED PERSONNEL TO OPERATE, PROGRAM, OR REPAIR THE EQUIPMENT!

We recommend approved Yaskawa training courses for all personnel involved with the operation, programming, or repair of the equipment.

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications.
NOTES FOR SAFE OPERATION

Read this manual carefully before installation, operation, maintenance, or inspection of the FS100.

In this manual, the Notes for Safe Operation are classified as “DANGER”, “WARNING”, “CAUTION”, “MANDATORY”, or “PROHIBITED”.

DANGER
Indicates an imminent hazardous situation which, if not avoided, could result in death or serious injury to personnel.

WARNING
Indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury to personnel.

CAUTION
Indicates a potentially hazardous situation which, if not avoided, could result in minor or moderate injury to personnel and damage to equipment. It may also be used to alert against unsafe practices.

MANDATORY
Always be sure to follow explicitly the items listed under this heading.

PROHIBITED
Must never be performed.

Even items described as “CAUTION” may result in a serious accident in some situations. At any rate, be sure to follow these important items.

NOTE
To ensure safe and efficient operation at all times, be sure to follow all instructions, even if not designated as “DANGER”, “WARNING” and “CAUTION”.

WARNING

• Maintenance and inspection must be performed by specified personnel.
Failure to observe this WARNING may result in electric shock or injury.

• For disassembly or repair, contact your Yaskawa representative.
• Do not remove the motor, and do not release the brake.
Failure to observe these WARNINGs may result in death or serious injury from unexpected turning of the manipulator's arm.
WARNING

• Before operating the manipulator, check that servo power is turned off when the emergency stop button on the programming pendant is pressed. When the servo power is turned off, the SERVO ON LED on the programming pendant is turned off.

Injury or damage to machinery may result if the emergency stop circuit cannot stop the manipulator during an emergency. The manipulator should not be used if the emergency stop buttons do not function.

Fig. : Emergency Stop Button

• In the case of not using the programming pendant, be sure to supply the emergency stop button on the equipment. Then before operating the manipulator, check to be sure that the servo power is turned OFF by pressing the emergency stop button. Connect the external emergency stop button to the 5-6 pin and 16-17 pin of the robot system signal connector (CN2).

• Upon shipment of the FS100, this signal is connected by a jumper cable in the dummy connector. To use the signal, make sure to prepare a new connector, and then input it.

If the signal is input with the jumper cable connected, it does not function, which may result in personal injury or equipment damage.

• Once the emergency stop button is released, clear the cell of all items which could interfere with the operation of the manipulator. Then turn the servo power ON.

Injury may result from unintentional or unexpected manipulator motion.

Fig. : Release of Emergency Stop Button

• Observe the following precautions when performing teaching operations within the P-point maximum envelope of the manipulator:
  – Be sure to use a lockout device to the safeguarding when going inside. Also, display the sign that the operation is being performed inside the safeguarding and make sure no one closes the safeguarding.
  – View the manipulator from the front whenever possible.
  – Always follow the predetermined operating procedure.
  – Ensure that you have a safe place to retreat in case of emergency.

Improper or unintended manipulator operation may result in injury.
Definition of Terms Used Often in This Manual

The MOTOMAN is the YASKAWA industrial robot product.

The MOTOMAN usually consists of the manipulator, the FS100 controller, manipulator cables, the FS100 programming pendant (optional), and the FS100 programming pendant dummy connector (optional).

In this manual, the equipment is designated as follows:

<table>
<thead>
<tr>
<th>Equipment</th>
<th>Manual Designation</th>
</tr>
</thead>
<tbody>
<tr>
<td>FS100 controller</td>
<td>FS100</td>
</tr>
<tr>
<td>FS100 programming pendant</td>
<td>Programming Pendant</td>
</tr>
<tr>
<td>Cable between the manipulator and the controller</td>
<td>Manipulator Cable</td>
</tr>
<tr>
<td>FS100 programming pendant dummy connector</td>
<td>Programming pendant dummy connector</td>
</tr>
</tbody>
</table>
Descriptions of the programming pendant keys, buttons, displays and keyboard of the PC are shown as follows:

<table>
<thead>
<tr>
<th>Equipment</th>
<th>Manual Designation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming Pendant</td>
<td></td>
</tr>
<tr>
<td>Character Keys</td>
<td>The keys which have characters printed on them are denoted with [ ] .</td>
</tr>
<tr>
<td></td>
<td>e.g. [ENTER]</td>
</tr>
<tr>
<td>Symbol Keys</td>
<td>The keys which have a symbol printed on them are not denoted with [ ] but depicted</td>
</tr>
<tr>
<td></td>
<td>with a small picture.</td>
</tr>
<tr>
<td></td>
<td>e.g. PAGE key</td>
</tr>
<tr>
<td></td>
<td>The cursor key is an exception, and a picture is not shown.</td>
</tr>
<tr>
<td>Axis Keys</td>
<td></td>
</tr>
<tr>
<td>Numeric Keys</td>
<td>“Axis keys” and “Numeric keys” are generic names for the keys for axis operation</td>
</tr>
<tr>
<td></td>
<td>and number input.</td>
</tr>
<tr>
<td>Keys Pressed Simultaneously</td>
<td>When two keys are to be pressed simultaneously, the keys are shown with a “+”</td>
</tr>
<tr>
<td></td>
<td>sign between them.</td>
</tr>
<tr>
<td></td>
<td>e.g. SHIFT key + COORD key</td>
</tr>
<tr>
<td>Mode Key</td>
<td>Three kinds of modes that can be selected by the mode key are denoted as follows:</td>
</tr>
<tr>
<td></td>
<td>REMOTE, PLAY, or TEACH</td>
</tr>
<tr>
<td>Button</td>
<td>Three buttons on the upper side of the programming pendant are denoted as follows:</td>
</tr>
<tr>
<td></td>
<td>HOLD button</td>
</tr>
<tr>
<td></td>
<td>START button</td>
</tr>
<tr>
<td></td>
<td>EMERGENCY STOP button</td>
</tr>
<tr>
<td>Displays</td>
<td>The menu displayed in the programming pendant is denoted with { } .</td>
</tr>
<tr>
<td></td>
<td>e.g. {JOB}</td>
</tr>
<tr>
<td>PC Keyboard</td>
<td>The name of the key is denoted.</td>
</tr>
<tr>
<td></td>
<td>e.g. Ctrl key on the keyboard</td>
</tr>
</tbody>
</table>

**Description of the Operation Procedure**

In the explanation of the operation procedure, the expression “Select • • •” means that the cursor is moved to the object item and [SELECT] is pressed, or that the item is directly selected by touching the screen.

**Registered Trademark**

In this manual, names of companies, corporations, or products are trademarks, registered trademarks, or brand names for each company or corporation. The indications of ® and ™ are omitted.
Safeguarding Tips

All operators, programmers, maintenance personnel, supervisors, and anyone working near the system must become familiar with the operation of this equipment. All personnel involved with the operation of the equipment must understand potential dangers of operation. General safeguarding tips are as follows:

• Improper operation can result in personal injury and/or damage to the equipment. Only trained personnel familiar with the operation of this equipment, the operator's manuals, the system equipment, and options and accessories should be permitted to operate this equipment.

• Improper connections can damage the equipment. All connections must be made within the standard voltage and current ratings of the equipment.

• The system must be placed in Emergency Stop (E-Stop) mode whenever it is not in use.

• In accordance with ANSI/RIA R15.06-2012, section 4.2.5, Sources of Energy, use lockout/tagout procedures during equipment maintenance. Refer also to Section 1910.147 (29CFR, Part 1910), Occupational Safety and Health Standards for General Industry (OSHA).

Mechanical Safety Devices

The safe operation of this equipment is ultimately the users responsibility. The conditions under which the equipment will be operated safely should be reviewed by the user. The user must be aware of the various national codes, ANSI/RIA R15.06-2012 safety standards, and other local codes that may pertain to the installation and use of this equipment.

Additional safety measures for personnel and equipment may be required depending on system installation, operation, and/or location. The following safety equipment is provided as standard:

• Safety barriers
• Door interlocks
• EMERGENCY STOP button

Check all safety equipment frequently for proper operation. Repair or replace any non-functioning safety equipment immediately.
Programming, Operation, and Maintenance Safety

All operators, programmers, maintenance personnel, supervisors, and anyone working near the system must become familiar with the operation of this equipment. Improper operation can result in personal injury and/or damage to the equipment. Only trained personnel familiar with the operation, manuals, electrical design, and equipment interconnections of this equipment should be permitted to program, or maintain the system. All personnel involved with the operation of the equipment must understand potential dangers of operation.

- Inspect the equipment to be sure no potentially hazardous conditions exist. Be sure the area is clean and free of water, oil, debris, etc.
- Be sure that all safeguards are in place. Check all safety equipment for proper operation. Repair or replace any non-functioning safety equipment immediately.
- Check the EMERGENCY STOP buttons for proper operation before programming. The equipment must be placed in the Emergency Stop (E-Stop) mode whenever it is not in use.
- Back up all programs and jobs onto suitable media before making program changes. To avoid loss of information, programs, or jobs, a backup must always be made before performing any service procedures or making any changes to options, accessories, or equipment.
- Any modifications to the Controller can cause severe personal injury or death, as well as damage to the Manipulator! Do not make any modifications to the Controller unit. Making any changes without written permission from Yaskawa will void the warranty.
- Some operations require a standard passwords and some require special passwords.
- The equipment allows modifications of the software for maximum performance. Use Care when making these modifications. All modifications made to the software will change the way the equipment operates and can cause severe personal injury or death, as well as damage parts of the system. Double check all modifications under every mode of operation to ensure changes do not create a hazard or dangerous situation.
- This equipment has multiple sources of electrical supply. Electrical interconnections are made between the Controller and other equipment. Disconnect and lockout/tagout all electrical circuits before making any modifications or connections.
- Do not perform any maintenance procedures before reading and understanding the proper procedures in the appropriate manual.
- Use proper replacement parts.
- Improper connections can damage equipment. All connections must be made within the standard voltage and current ratings of the equipment.
Maintenance Safety

Turn power OFF and disconnect and lockout/tagout all electrical circuits before making any modifications or connections.

Perform only the maintenance described in this manual. Maintenance other than specified in this manual should be performed only by Yaskawa-trained, qualified personnel.

Summary of Warning Information

This manual is provided to help users establish safe conditions for operating the equipment. Specific considerations and precautions are also described in the manual, but appear in the form of DANGERS, WARNINGS, CAUTIONS, and Notes.

It is important that users operate the equipment in accordance with this instruction manual and any additional information which may be provided by Yaskawa. Address any questions regarding the safe and proper operation of the equipment to Customer Support.
Motoman Disposal

**WARNING**

- Take precautionary measures to prevent the manipulator from overturning, such as anchoring it firmly, etc., even when temporarily storing it before disposal.

Failure to observe this instruction may cause overturning of the Manipulator, which may result in personal injury.

**CAUTION**

- Do not modify the Manipulator or the Controller

Failure to observe this instruction can cause fire, mechanical failure, or malfunction, which may result in personal injury and/or equipment damage.

**MANDATORY**

When disposing of or recycling the MOTOMAN, follow the applicable national/local laws and regulations.

**PROHIBITED**

This symbol is applicable in some locations.

The wheelie bin symbol on this product, manual or its packaging indicates that at the end of life the product should enter the recycling system. It must be disposed at an appropriate collection point for electrical and electronic equipment (EEE) and should not be put in the normal waste stream.
Customer Support Information

If you need assistance with any aspect of your FS100 system, please contact Customer Support at the following 24-hour telephone number:

(937) 847-3200

For routine technical inquiries, you can also contact Customer Support at the following e-mail address:

techsupport@motoman.com

When using e-mail to contact Customer Support, please provide a detailed description of the issue, along with complete contact information. Please allow approximately 24 to 36 hours for a response.

Please use e-mail for routine inquiries only. If you have an urgent or emergency need for service, replacement parts, or information, contact Customer Support at the telephone number shown above.

Please have the following information ready before you call Customer Support:

- System
- Primary Application
- Controller
- Software Version
- Manipulator Serial Number
- Manipulator Sales Order Number
- Positioner

Access this information on the Programming Pendant’s LCD display screen by selecting {MAIN MENU} - {SYSTEM INFO} - {VERSION}

Located on the Manipulator data plate

Located on the Controller data plate
# Table of Contents

1 List of Usable API in Each Mode ........................................................................................................1-1

2 Task Control API .............................................................................................................................2-1
   mpCreateTask .................................................................................................................................2-1
   mpDeleteTask .................................................................................................................................2-3
   mpTaskSuspend ...............................................................................................................................2-4
   mpTaskResume ...............................................................................................................................2-5
   mpMsgQCreate ...............................................................................................................................2-6
   mpMsgQDelete .................................................................................................................................2-7
   mpMsgQReceive ..............................................................................................................................2-8
   mpMsgQSend ....................................................................................................................................2-9
   mpErrMsgQRcv ...............................................................................................................................2-11
   mpErrMsgQSnd ...............................................................................................................................2-13
   mpSemBCreate ...............................................................................................................................2-15
   mpSemDelete ....................................................................................................................................2-16
   mpSemTake .......................................................................................................................................2-17
   mpSemGive .......................................................................................................................................2-18
   mpErrSemTake ...................................................................................................................................2-19
   mpTaskDelay .....................................................................................................................................2-21
   mpGetRtc ..........................................................................................................................................2-22
   mpClkAnnounce ...............................................................................................................................2-23
   mpStopWatchCreate .........................................................................................................................2-25
   mpStopWatchDelete ..........................................................................................................................2-26
   mpStopWatchStart .............................................................................................................................2-27
   mpStopWatchStop ...............................................................................................................................2-28
   mpStopWatchLap ...............................................................................................................................2-29
   mpStopWatchReset ............................................................................................................................2-30
   mpStopWatchGetTime .........................................................................................................................2-31
   mpStopWatchGetLapNum .....................................................................................................................2-32
   mpStopWatchGetLapTime ....................................................................................................................2-33
   mpStopWatchGetAliveLapNo ..............................................................................................................2-34

3 User Watchdog API ..........................................................................................................................3-1
   mpUsrWdogCreate ..............................................................................................................................3-1
   mpUsrWdogDelete ..............................................................................................................................3-3
   mpUsrWdogStart ...............................................................................................................................3-4
   mpUsrWdogClear ..............................................................................................................................3-5

4 Network API .....................................................................................................................................4-1
   mpSocket .........................................................................................................................................4-1
   mpListen ..........................................................................................................................................4-2
   mpAccept .........................................................................................................................................4-3
   mpBind .............................................................................................................................................4-4
   mpConnect .......................................................................................................................................4-5
   mpRecv ............................................................................................................................................4-6
Table of Contents

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpRecvFrom</td>
<td>4-6</td>
</tr>
<tr>
<td>mpSend</td>
<td>4-7</td>
</tr>
<tr>
<td>mpSendTo</td>
<td>4-7</td>
</tr>
<tr>
<td>mpClose</td>
<td>4-8</td>
</tr>
<tr>
<td>mpHtonl</td>
<td>4-9</td>
</tr>
<tr>
<td>mpHtons</td>
<td>4-9</td>
</tr>
<tr>
<td>mpNtohl</td>
<td>4-9</td>
</tr>
<tr>
<td>mpNtohs</td>
<td>4-9</td>
</tr>
<tr>
<td>mpInetAddr</td>
<td>4-10</td>
</tr>
<tr>
<td>mpInetNtoa</td>
<td>4-10</td>
</tr>
<tr>
<td>mpInetNtoaB</td>
<td>4-10</td>
</tr>
<tr>
<td>mpGetsockname</td>
<td>4-11</td>
</tr>
<tr>
<td>mpGetpeermaname</td>
<td>4-12</td>
</tr>
<tr>
<td>mpSetsockopt</td>
<td>4-13</td>
</tr>
<tr>
<td>mpIoctl</td>
<td>4-14</td>
</tr>
<tr>
<td>mpSelect</td>
<td>4-15</td>
</tr>
</tbody>
</table>

5 Serial Communication API .............................................................................................................. 5-1

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpRsInit</td>
<td>5-1</td>
</tr>
<tr>
<td>mpRsOpen</td>
<td>5-3</td>
</tr>
<tr>
<td>mpRsClose</td>
<td>5-4</td>
</tr>
<tr>
<td>mpRsSend</td>
<td>5-5</td>
</tr>
<tr>
<td>mpRsRecv</td>
<td>5-6</td>
</tr>
</tbody>
</table>

6 Memory Pool Control API ............................................................................................................... 6-1

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpMalloc</td>
<td>6-1</td>
</tr>
<tr>
<td>mpFree</td>
<td>6-2</td>
</tr>
</tbody>
</table>

7 File Control API ............................................................................................................................... 7-1

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpCreate</td>
<td>7-1</td>
</tr>
<tr>
<td>mpOpen</td>
<td>7-3</td>
</tr>
<tr>
<td>mpRemove</td>
<td>7-5</td>
</tr>
<tr>
<td>mpClose</td>
<td>7-6</td>
</tr>
<tr>
<td>mpRename</td>
<td>7-7</td>
</tr>
<tr>
<td>mpRead</td>
<td>7-8</td>
</tr>
<tr>
<td>mpWrite</td>
<td>7-9</td>
</tr>
<tr>
<td>mpIoctl</td>
<td>7-10</td>
</tr>
<tr>
<td>mpLseek</td>
<td>7-12</td>
</tr>
<tr>
<td>mpFstat</td>
<td>7-13</td>
</tr>
<tr>
<td>mpStat</td>
<td>7-13</td>
</tr>
<tr>
<td>mpOpendir</td>
<td>7-15</td>
</tr>
<tr>
<td>mpReaddir</td>
<td>7-16</td>
</tr>
<tr>
<td>mpRewinddir</td>
<td>7-17</td>
</tr>
<tr>
<td>mpClosedir</td>
<td>7-18</td>
</tr>
</tbody>
</table>
8 Existing File Access API ................................................................. 8-1
   mpLoadFile  ................................................................................. 8-1
   mpSaveFile  ............................................................................... 8-3
   mpRefreshFileList ................................................................. 8-5
   mpGetFileCount .................................................................... 8-6
   mpGetFileName ..................................................................... 8-7
   mpFdWriteFile ...................................................................... 8-8
   mpFdReadFile ....................................................................... 8-10
   mpFdGetJobList ................................................................... 8-12

9 System Monitor API ................................................................. 9-1
   mpGetVarData ........................................................................ 9-1
   mpGetSVarInfo ...................................................................... 9-3
   mpReadIO ............................................................................... 9-5
   mpMonitor ........................................................................ 9-7
   mpGetPosVarData .................................................................. 9-9
   mpGetUserVars .................................................................. 9-11
   mpGetAlarmStatus .............................................................. 9-15
   mpGetAlarmCode .................................................................. 9-16
   mpGetMode ........................................................................... 9-17
   mpGetCycle .......................................................................... 9-18
   mpGetServoPower .................................................................. 9-19
   mpGetPlayStatus .................................................................. 9-20
   mpGetMasterJob .................................................................... 9-21
   mpGetCurJob .......................................................................... 9-23
   mpGetSpecialOpStatus .......................................................... 9-25
   mpGetJobDate ....................................................................... 9-26
   mpGetCartPos ....................................................................... 9-28
   mpGetCartPosEx .................................................................... 9-30
   mpGetPulsePos ...................................................................... 9-32
   mpGetRadPos ......................................................................... 9-34
   mpGetRadPosEx .................................................................... 9-36
   mpGetDegPos ......................................................................... 9-38
   mpGetDegPosEx .................................................................... 9-40
   mpGetFBPulsePos ................................................................... 9-42
   mpGetServoSpeed .................................................................. 9-44
   mpGetFBSpeed ....................................................................... 9-46
   mpGetTorque ......................................................................... 9-48
   mpGetTorqueEx ..................................................................... 9-50
   mpGetSysTimes .................................................................... 9-52
   mpGetSysVersionNo .............................................................. 9-54
   mpGetJogSpeed ..................................................................... 9-55
   mpGetJogCoord ..................................................................... 9-56
   mpCtrlGrpId2GrpNo ............................................................... 9-58
# Table of Contents

## 10 System Control API

- mpPutVarData ......................................................... 10-1
- mpPutSVarInfo ....................................................... 10-3
- mpWriteIO ............................................................... 10-4
- mpPutPosVarData .................................................... 10-6
- mpPutUserVars ......................................................... 10-9
- mpSetAlarm .............................................................. 10-14
- mpCancelError ......................................................... 10-15
- mpResetAlarm ......................................................... 10-16
- mpSetCycle ............................................................. 10-17
- mpSetServoPower .................................................... 10-19
- mpSetMasterJob ....................................................... 10-21
- mpSetCurJob ........................................................... 10-23
- mpStartJob ............................................................. 10-25
- mpHold ................................................................. 10-27
- mpWaitForJobEnd ...................................................... 10-29
- mpDeleteJob ........................................................... 10-31

## 11 Motion Control API

- mpMotStart ........................................................... 11-1
- mpMotStop ............................................................. 11-2
- mpMotTargetClear ................................................... 11-3
- mpMotTargetSend .................................................... 11-4
- mpMotTargetReceive ............................................... 11-7
- mpMotSetCoord ........................................................ 11-10
- mpMotSetSpeed ....................................................... 11-12
- mpMotSetOrigin ....................................................... 11-14
- mpMotSetTask .......................................................... 11-16
- mpMotSetSync ........................................................ 11-18
- mpMotResetSync ...................................................... 11-20

## 12 Step Move API

- mpiMOV ................................................................. 12-1
- mpMOVJ ................................................................. 12-6
- mpMOVL ................................................................. 12-10
- mpPulseMOVJ ........................................................ 12-14
- mpPulseMOVL ........................................................ 12-17

## 13 Sensor Control API

- mpReceiveSkillCommand ............................................ 13-1
- mpEndSkillCommandProcess ....................................... 13-4
- mpMeiGetJobExecTask ............................................... 13-7
- mpMeiGetInterpolation .............................................. 13-8
- mpMeiGetExecControlGroup ....................................... 13-9
- mpMeiPutCorrPath ...................................................... 13-11
- mpMeiPutForcePathEnd ............................................... 13-15
- mpMeiPutSpdOverride ............................................... 13-17
14  Robot Control API ...............................................................................................................................14-1
    mpExRcsIncrementMove .......................................................................................................................14-1

15  Servo Monitor/Control API ..................................................................................................................15-1
    mpSvsGetVelTrqFb ..............................................................................................................................15-1
    mpSvsStartTrqLimit ..........................................................................................................................15-3
    mpSvsSetTrqLimit ............................................................................................................................15-6
    mpSvsEndTrqLimit ............................................................................................................................15-10
    mpSvsStartTrqCtrl ..............................................................................................................................15-12
    mpSvsSetTrqCtrl .................................................................................................................................15-15
    mpSvsEndTrqCtrl .................................................................................................................................15-18
    mpSvsForceInit ...................................................................................................................................15-20

16  Kinematics API ....................................................................................................................................16-1
    mpConvAxesToCartPos .........................................................................................................................16-1
    mpConvCartPosToAxes ..........................................................................................................................16-4
    mpConvPulseToAngle ............................................................................................................................16-8
    mpConvAngleToPulse ............................................................................................................................16-10
    mpConvFBPulseToPulse .........................................................................................................................16-12
    mpMakeFrame ..................................................................................................................................16-14
    mplInvFrame ....................................................................................................................................16-16
    mpRotFrame ......................................................................................................................................16-18
    mpMulFrame ......................................................................................................................................16-20
    mpZYXeulerToFrame .............................................................................................................................16-22
    mpFrameToZYXeuler ............................................................................................................................16-24
    mpCrossProduct ................................................................................................................................16-26
    mplInnerProduct .................................................................................................................................16-27

17  ForceSensor Monitor/Control API ....................................................................................................17-1
    mpFcsStartMeasuring ..........................................................................................................................17-1
    mpFcsGetForceData .............................................................................................................................17-3
    mpFcsStartImp ...................................................................................................................................17-5
    mpFcsSetReferenceForce .......................................................................................................................17-8
    mpFcsEndImp ......................................................................................................................................17-10

18  MotomanSync IF Access API .............................................................................................................18-1
    mpMotoSyncIFOpen ...............................................................................................................................18-1
    mpMotoSyncIFClose ..............................................................................................................................18-2
    mpMotoSyncIFLock ...............................................................................................................................18-3
    mpMotoSyncIFUnlock ............................................................................................................................18-5

19  Alarm List ..............................................................................................................................................19-1
## 1 List of Usable API in Each Mode

<table>
<thead>
<tr>
<th>API Name</th>
<th>Not when an alarm or error occurs</th>
<th>Teach mode</th>
<th>Play mode</th>
<th>When an alarm or error occurs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Not in operation</td>
<td>In operation</td>
<td>Not in operation</td>
</tr>
<tr>
<td>Task control API</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpCreateTask</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpDeleteTask</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpTaskSuspend</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpTaskResume</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpMsgQCreate</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpMsgQDelete</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpMsgQReceive</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpMsgQSend</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpErrMsgQRcv</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpErrMsgQSnd</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpSemBCreate</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpSemDelete</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpSemTake</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpSemGive</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpErrSemTake</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpTaskDelay</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetRtc</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpClkAnnounce</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpStopWatchCreate</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpStopWatchDelete</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpStopWatchStart</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpStopWatchStop</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpStopWatchLap</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpStopWatchReset</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpStopWatchGetTime</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpStopWatchGetLapNum</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpStopWatchGetLapTime</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpStopWatchGetAliveLapNo</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>User watchdog API</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpUsrWdogCreate</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpUsrWdogDelete</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpUsrWdogStart</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpUsrWdogClear</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>API Name</td>
<td>Not when an alarm or error occurs</td>
<td>When an alarm or error occurs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------------</td>
<td>-------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Teach mode</td>
<td>Play mode</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Not in operation</td>
<td>In operation</td>
<td>Not in operation</td>
<td>In operation</td>
</tr>
<tr>
<td>mpSocket</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpListen</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpAccept</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpBind</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpConnect</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpRecv</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpRecvFrom</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpSend</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpSendTo</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpClose</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpHtonl</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpHtons</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpNtohl</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpNtohs</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpNetAddr</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpinetNtoa</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpinetNtoaB</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetsockopt</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetpeername</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpSocketopt</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mplocal</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpSelect</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpRsInit</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpRsOpen</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpRsClose</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpRsSend</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpRsRecv</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpMalloc</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpFree</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpCreate</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpOpen</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpRemove</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpClose</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpRename</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpRead</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpWrite</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mployctl</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpLseek</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpFstat</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
</tbody>
</table>
## List of Usable API in Each Mode

<table>
<thead>
<tr>
<th>API Name</th>
<th>Not when an alarm or error occurs</th>
<th>When an alarm or error occurs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FS100</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Teach mode</td>
<td>Play mode</td>
</tr>
<tr>
<td></td>
<td>Not in operation</td>
<td>Not in operation</td>
</tr>
<tr>
<td><strong>File control API</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpStat</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpOpendir</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpReaddir</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpRewinddir</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpClosedir</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td><strong>Existing file access</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpLoadFile</td>
<td>o</td>
<td>2010</td>
</tr>
<tr>
<td>mpSaveFile</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpRefreshFileList</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetFileCount</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetFileName</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpFdWriteFile</td>
<td>o</td>
<td>2010</td>
</tr>
<tr>
<td>mpFdReadFile</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpFdGetJobList</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td><strong>System monitor API</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpGetVarData</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetSVarInfo</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpReadIO</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpMonitor</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetPosVarData</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetUserVars</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetAlarmStatus</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetAlarmCode</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetMode</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetCycle</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetServoPower</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetPlayStatus</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetMasterJob</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetCurJob</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetSpecialOpStatus</td>
<td></td>
<td>o</td>
</tr>
<tr>
<td>mpGetJobDate</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetCartPos</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetCartPosEx</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetPulsePos</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetRadPos</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetRadPosEx</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetDegPos</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetDegPosEx</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetFPulsePos</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetServoSpeed</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetFBSpeed</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetTorque</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetTorqueEx</td>
<td>o</td>
<td>o</td>
</tr>
</tbody>
</table>
## List of Usable API in Each Mode

<table>
<thead>
<tr>
<th>API Name</th>
<th>Teach mode</th>
<th>Play mode</th>
<th>When an alarm or error occurs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Not in operation</td>
<td>In operation</td>
<td>Not in operation</td>
</tr>
<tr>
<td><strong>System monitor API</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpGetSysTimes</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetSysVersionNo</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetJogSpeed</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpGetJogCoord</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpCtrlGrpId2GrpNo</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td><strong>System control API</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpPutVarData</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpPutSVarInfo</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpWriteIO</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpPutPosVarData</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpPutUserVars</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpSetAlarm</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpCancelError</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpResetAlarm</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpSetCycle</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpSetServoPower</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpSetMasterJob</td>
<td>o</td>
<td>2010</td>
<td>o</td>
</tr>
<tr>
<td>mpSetCurJob</td>
<td>o</td>
<td>2010</td>
<td>o</td>
</tr>
<tr>
<td>mpStartJob</td>
<td>2080</td>
<td>2080</td>
<td>o</td>
</tr>
<tr>
<td>mpHold</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpWaitForJobEnd</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpDeleteJob</td>
<td>o</td>
<td>2010</td>
<td>2080</td>
</tr>
<tr>
<td><strong>Motion control API</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpMotStart</td>
<td>r</td>
<td>r</td>
<td>o</td>
</tr>
<tr>
<td>mpMotStop</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpMotTargetClear</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpMotTargetSend</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpMotTargetReceive</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpMotSetCoord</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpMotSetSpeed</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpMotSetOrigin</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpMotSetTask</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpMotSetSync</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpMotResetSync</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td><strong>Step Move API</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpMOV</td>
<td>2080</td>
<td>2080</td>
<td>o</td>
</tr>
<tr>
<td>mpMOVJ</td>
<td>2080</td>
<td>2080</td>
<td>o</td>
</tr>
<tr>
<td>mpMOVL</td>
<td>2080</td>
<td>2080</td>
<td>o</td>
</tr>
<tr>
<td>mpPulseMOVJ</td>
<td>2080</td>
<td>2080</td>
<td>o</td>
</tr>
<tr>
<td>mpPulseMOVL</td>
<td>2080</td>
<td>2080</td>
<td>o</td>
</tr>
<tr>
<td>API Name</td>
<td>Teach mode</td>
<td>Play mode</td>
<td>When an alarm or error occurs</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------</td>
<td>-----------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td></td>
<td>Not in operation</td>
<td>In operation</td>
<td>Not in operation</td>
</tr>
<tr>
<td>Sensor control API</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpReceiveSkillCommand</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpEndSkillCommandProcess</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpMeiGetJobExecTask</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpMeiGetInterpolation</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpMeiGetExecControlGroup</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpMeiPutCorrPath</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpMeiPutForcePathEnd</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpMeiPutSpdOverride</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>Robot control API</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpExRcsIncrementMove</td>
<td>r</td>
<td>i</td>
<td>r</td>
</tr>
<tr>
<td>Servo monitor/control API</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpSvsGetVelTrqFb</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpSvsStartTrqLimit</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpSvsSetTrqLimit</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpSvsEndTrqLimit</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpSvsStartTrqCtrl</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpSvsSetTrqCtrl</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpSvsEndTrqCtrl</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpSvsForceInit</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>Kinematics API</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpConvAxesToCartPos</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpConvCartPosToAxes</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpConvPulseToAngle</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpConvAngleToPulse</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpConvFPulseToPulse</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpMakeFrame</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpInvFrame</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpRotFrame</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpMulFrame</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpZYXeulerToFrame</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpFrameToZYXeuler</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpCrossProduct</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpInnerProduct</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>Force sensor Monitor/Control API</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpFcsStartMeasuring</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpFcsGetForceData</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpFcsStartImp</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpFcsSetReferenceForce</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpFcsEndImp</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>MotomanSync I/F access API</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpMotoSyncIFOpen</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpMotoSyncIFClose</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpMotoSyncIFLock</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>mpMotoSyncIFUnlock</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
</tbody>
</table>
### List of Usable API in Each Mode

<table>
<thead>
<tr>
<th>Alarm code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐</td>
<td>: Executable</td>
</tr>
<tr>
<td>✗</td>
<td>: Inexecutable (API Error)</td>
</tr>
<tr>
<td>2060</td>
<td>: Alarm or error occurrence</td>
</tr>
<tr>
<td>2080</td>
<td>: Different mode</td>
</tr>
<tr>
<td>2010</td>
<td>: Manipulator in operation</td>
</tr>
<tr>
<td>3450</td>
<td>: Failed to turn on the servo power</td>
</tr>
</tbody>
</table>
mpCreateTask

Creates a task.

**Syntax**

```c
int mpCreateTask(
    int priority, /* priority of new task */
    int stackSize, /* size (bytes) of stack */
    FUNCPTR entryPt, /* entry point of new task */
    int arg1, /* 1st of 10 req'd task args to pass to func */
    int arg2,
    int arg3,
    int arg4,
    int arg5,
    int arg6,
    int arg7,
    int arg8,
    int arg9,
    int arg10
)
```

**Parameter**

[priority]

One of the following 4 types can be selected. If a type other than these is specified, the function returns an error.

- **MP_PRI_IO_CLK_TAKE**
  This priority level is a special value for a task started every I/O cycle. Since the task of this priority is read on each execution cycle of the concurrent I/O ladder, the I/O status can be referred immediately after the I/O update.

- **MP_PRI_IP_CLK_TAKE**
  This priority level is a special value for a task started every interpolation cycle.

- **MP_PRI_TIME_CRITICAL**
  High priority task: Response time is from several milliseconds to several tens of milliseconds. The latency is minimal, but the processing time of this task must be short (1 msec or less) because it influences the whole system performance.

- **MP_PRI_TIME_NORMAL**
  Normal priority task: Normally use this level. Response time is from several tens of milliseconds to several hundreds of milliseconds. The CPU time of this task is scheduled by the time-sharing system (TSS) with the system task.
[stack size]

If set MP_STACK_SIZE, then stack size is automatically set as optimum value.

Make sure to set MP_STACK_SIZE for the stackSize. The stack size is set at 20 Kbytes. If another size is set, the function returns an error.

- Return Value

Task ID (TID)  Normal end
-1  Error (cannot create a task due to insufficient memory; exceeding the maximum task number; wrong priority; or the specified stack size is not MP_STACK_SIZE.)

- Note

This routine creates and starts a task. The creation and the start-up are always done simultaneously. Normally, the task started can be suspended by becoming the waiting state after initialization (if needed).

The number of tasks which can be created is limited as shown below.

<table>
<thead>
<tr>
<th>Priority</th>
<th>Description</th>
<th>Max. task number</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_PRI_IO_CLK_TAKE</td>
<td>Notification of I/O control period</td>
<td>1</td>
</tr>
<tr>
<td>MP_PRI_IP_CLK_TAKE</td>
<td>Notification of interpolation period</td>
<td>1</td>
</tr>
<tr>
<td>MP_PRI_TIME_CRITICAL</td>
<td>High priority for general use</td>
<td>1</td>
</tr>
<tr>
<td>MP_PRI_TIME_NORMAL</td>
<td>Normal priority for general use</td>
<td>3</td>
</tr>
</tbody>
</table>

Use precautions when you select a priority other than MP_PRI_TIME_NORMAL. The priority MP_PRI_IO_CLK_TAKE, MP_PRI_IP_CLK_TAKE, and MP_PRI_TIME_CRITICAL have higher priorities than the man-machine control task. Thus, if unusually long processing time (100 microseconds or more) is used to describe a program with these priorities, critical problems may occur. (e.g. The system alarm occurs due to insufficient processing time for the robot control task; The programming pendant or the HOLD button freezes because the man-machine control task cannot be operated.)

MotoPlus detects such status (when the man-machine task does not run for 3 seconds or more), stops the robot motion by turning the servo power OFF, and turns ON the system output ($50901) as the protective function to take countermeasures for peripheral devices. Thus, make sure to set 100 microseconds as the maximum processing time in the high-priority task (100 microseconds equals to approx. 1000 lines in the C language). The OS waiting time, the message/semaphore waiting time, and the task delay time are not included in this processing time.
mpDeleteTask

Deletes the task.

- **Syntax**

  ```c
  STATUS mpDeleteTask
  (  
    int tid /* task ID of task to delete */
  )
  ```

- **Parameter**

  * [tid]_

  Set TID_SELF to delete the calling task itself.

- **Return Value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>-1</td>
<td>Error</td>
</tr>
</tbody>
</table>

- **Note**

  To delete a task other than the calling task itself by this routine, the task ID returned when the task is created must be retained in the application.
mpTaskSuspend

Suspends the task.

■ Syntax

```c
STATUS mpTaskSuspend
(  int tid /* task ID of task to suspend */
)
```

■ Parameter

[tid]
Suspends the task specified by tid. If tid is TID_SELF, the calling task itself is suspended.

■ Return value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>-1</td>
<td>Error</td>
</tr>
</tbody>
</table>

■ Note

The specified task is suspended regardless of its status. Thus, if the task which occupies the system resource (e.g. the network or the system memory partition) is suspended, the function regarding the resource cannot be used. Then the system will deadlock.
mpTaskResume

Resumes the task.

■ Syntax

STATUS mpTaskResume
(int tid /* task ID of task to resume */)

■ Description

This routine releases the specified task from the suspended state.

■ Parameter

[tid]
Task ID

■ Return Value

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>-1</td>
<td>Error</td>
</tr>
</tbody>
</table>
mpMsgQCreate

Creates and initializes a message queue.

- **Syntax**

  ```c
  MSG_Q_ID mpMsgQCreate
  (  
    int maxMsgs, /* max messages that can be queued */
    int maxMsgLength, /* max bytes in a message */
    int options /* message queue options */
  )
  ```

- **Description**

  This routine creates a message queue which can keep messages up to the number of `maxMsgs` (the size of each message is up to `maxMsgLength` bytes).

- **Parameter**

  - `[maxMsgs]`  
    The max. number of messages that can be queued

  - `[maxMsgLength]`  
    The max. size of a message queue

  - `[options]`
    - MSG_Q_FIFO (0x00)  
      Suspended tasks wait in the sequence of FIFO.
    - MSG_Q_PRIORITY (0x01)  
      Suspended tasks wait in the sequence of priority.

- **Return Value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nonzero</td>
<td>MSG_Q_ID (Normal end)</td>
</tr>
<tr>
<td>0</td>
<td>Error</td>
</tr>
</tbody>
</table>
mpMsgQDelete

Deletes the message queue.

■ Syntax

```c
STATUS mpMsgQDelete
(  
  MSG_Q_ID msgQId /* message queue to delete */
)
```

■ Parameter

[msgQId]
The ID of the message queue to be deleted

■ Return Value

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>-1</td>
<td>Error</td>
</tr>
</tbody>
</table>

■ Note

This routine deletes a message queue. If the specified task is blocked by mpMsgQSend() or mpMsgQReceive(), the task is released from the blocked state.
mpMsgQReceive

Receives a message.

■ Syntax

```c
int mpMsgQReceive
(
    MSG_Q_ID msgQId,  /* message queue from which to receive */
    char * buffer,     /* buffer to receive message */
    UINT maxNBytes,   /* length of buffer */
    int timeout       /* ticks to wait */
)
```

■ Description

This routine receives the message from the message queue msgQId. The received message is written to the specified buffer. The length of the buffer is maxNBytes. If the message is longer than maxNBytes, the excess is discarded. If no message is registered in the queue when the mpMsgQReceive() is called, the system waits for the registration for the number of ticks set to `timeout`.

■ Parameter

- `[msgQId]`
The ID of the message queue to be received
- `[buffer]`
The buffer to receive the message
- `[maxNBytes]`
The max length of the message to be received
- `[timeout]`
The number of ticks to wait for the registration

Also the following can be set:

- **NO_WAIT (0)**
  End immediately even if the message is not arrived
- **WAIT_FOREVER (-1)**
  Keep waiting without timeout

■ Return Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;= 0</td>
<td>Data length copied to buffer (bytes)</td>
</tr>
<tr>
<td>-1</td>
<td>Error</td>
</tr>
</tbody>
</table>

To retrieve the error cause, use mpErrMsgQRcv.

■ Note

The period of time per tick (msec) can be retrieved by the following API:

```c
int mpGetRtc(void)
```

Return value: 1 (minimum) or more
(unit: msec)
mpMsgQSend

Sends a message.

- **Syntax**

  ```c
  STATUS mpMsgQSend
  (  
    MSG_Q_ID msgQId, /* message queue on which to send */
    char * buffer, /* message to send */
    UINT nBytes, /* length of message */
    int timeout, /* ticks to wait */
    int priority /* MSG_PRI_NORMAL or MSG_PRI_URGENT */
  )
  ```

- **Description**

  This routine sends the message written to the buffer (length: nBytes) to the message queue msgQId. If the message queue is full, the system waits for the queue to become available for the number of ticks set to `timeout`. The priority of the message to be sent is set to `priority`.

- **Parameter**

  - `[msgQId]`
    The message queue ID
  
  - `[buffer]`
    The message buffer to be sent
  
  - `[nBytes]`
    The length (byte) of the message to be sent
  
  - `[timeout]`
    The number of ticks to wait for the queue to be available if the message queue is full

  Also the following can be set:

  - **NO_WAIT (0)**
    End immediately even if the message is not sent
  
  - **WAIT_FOREVER (-1)**
    Keep waiting without timeout

  - `[priority]`
    Message priority can be set.

    - **MSG_PRI_NORMAL (0)**
      Normal: The message is set to the end of the queue.
    
    - **MSG_PRI_URGENT (1)**
      Urgent: The message is set to the top of the queue.
2 Task Control API
mpMsgQSend

- **Return Value**
  - 0 Normal end
  - -1 Error
    To retrieve the error cause, use mpErrMsgQRcv.

- **Note**
  The period of time per tick (msec) can be retrieved by the following API:
  ```c
  int mpGetRtc(void)
  ```
  Return value: 1 (minimum) or more (unit: msec)
mpErrMsgQRcv

Retrieves an error when a message queue is received.

■ Syntax

int mpErrMsgQRcv(void)

■ Description

This API retrieves the cause of the error which occurs when mpMsgQReceive is executed. Only when retrieving it immediately after mpMsgQReceive returns -1 (Error), the reference value is correct.

■ Parameter

None

■ Return Value

One of the following values returns.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_TIME_OUT 0x00000001</td>
<td>Timeout</td>
</tr>
<tr>
<td>E_NONEXIST 0x00000002</td>
<td>The message queue was deleted</td>
</tr>
<tr>
<td>E_OBJECTS_ID 0x00000004</td>
<td>Object ID error</td>
</tr>
<tr>
<td>E_OBJ_NO_METHOD 0x00000005</td>
<td>Other than the above</td>
</tr>
</tbody>
</table>

■ Note

<Sample Code>

```c
int foo(void)
{
    if (mpMsgQReceive(msgQId, (char *)buffer, maxNBytes,timeout) == ERROR)
    {
        switch (mpErrMsgQRcv())
        {
            case E_TIME_OUT:
                // Some error processing is done.
                break;
            case E_NONEXIST:
                // Some error processing is done.
                break;
            case E_OBJECTS_ID:
                // Some error processing is done.
                break;
        }
    }
```

default: //case E_OBJ_NO_METHOD:
    // Some error processing is done.
    break;
    return (NG);
}
mpErrMsgQSnd

Retrieves an error when a message queue is sent.

■ Syntax

int mpErrMsgQSnd(void)

■ Description

This API retrieves the cause of the error which occurs when
mpMsgQSnd is executed. Only when retrieving it immediately after
mpMsgQSnd returns -1 (Error), the reference value is correct.

■ Parameter

None

■ Return Value

One of the following values returns.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_TIME_OUT 0x00000001</td>
<td>A timeout occurs when other than NO_WAIT is set to “timeout” of mpMsgQSnd.</td>
</tr>
<tr>
<td>E_NONEXIST 0x00000002</td>
<td>The message queue was deleted</td>
</tr>
<tr>
<td>E_LIMIT_EXCEEDED 0x00000003</td>
<td>A timeout occurs when NO_WAIT is set to “timeout” of mpMsgQSnd, or,</td>
</tr>
<tr>
<td></td>
<td>a value larger than maxMsgLength of mpMsgQCreate is set to “nBytes” of</td>
</tr>
<tr>
<td></td>
<td>mpMsgQSnd.</td>
</tr>
<tr>
<td>EOBJECTS_ID 0x00000004</td>
<td>Object ID error</td>
</tr>
<tr>
<td>EOBJ_NO_METHOD 0x00000005</td>
<td>Other than the above</td>
</tr>
</tbody>
</table>

■ Note

<Sample Code>

int foo(void)
{
    if (mpMsgQSnd(msgQId, (char *)buffer, nBytes, timeout,
                  MSG_PRI_NORMAL) == ERROR)
    {
        switch (mpErrMsgQSnd())
        {
            case E_TIME_OUT:
                // Some error processing is done.
                break;
            case E_NONEXIST:
                // Some error processing is done.
                break;
            // Other cases...
        }
    }
}
// Some error processing is done.
break;
case E_LIMIT_EXCEEDED:// Overflow message queue num.
    // Some error processing is done.
    break;
case E_OBJECTS_ID:
    // Some error processing is done.
    break;
default://case E_OBJ_NO_METHOD:
    // Some error processing is done.
    break;
return (NG);
}

return (OK);
mpSemBCreate

Creates and initializes a binary semaphore.

- **Syntax**

  ```c
  SEM_ID mpSemBCreate
  (  
     int options, /* semaphore options */
     SEM_B_STATE initialState /* initial semaphore state */
  )
  ```

- **Description**

  This routine allocates memory to a binary semaphore and initializes it. The semaphore is initialized to the value of `initialState`. Set `SEM_FULL(1)` or `SEM_EMPTY(0)` to `initialState`. Set the queuing procedure of the blocked task to `options`. Two queuing procedures are available: the priority procedure and the FIFO procedure. For the priority procedure, set `SEM_Q_PRIORITY(0x1)`. For the FIFO procedure, set `SEM_Q_FIFO(0x0)`. 

- **Parameter**

  - **[options]**
    The queuing procedure of the blocked task
    - `SEM_Q_PRIORITY(0x1)`: priority procedure
    - `SEM_Q_FIFO(0x0)`: FIFO procedure
  
  - **[initialState]**
    The initialized value: 0 or 1
    - `SEM_FULL(1)`
    - `SEM_EMPTY(0)`

- **Return Value**

  - Nonzero Semaphore ID
  - 0 ERROR: no memory
mpSemDelete

Deletes the semaphore.

- **Syntax**

  ```c
  STATUS mpSemDelete
  (    
      SEM_ID semId /* semaphore ID to delete */
  )
  ```

- **Description**

  This routine deletes the specified semaphore and releases the memory allocated to it. The suspended task is released from the blocked state.

- **Parameter**

  `[semId]`

  The semaphore ID to be deleted

- **Return Value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>-1</td>
<td>Error (invalid semaphore ID)</td>
</tr>
</tbody>
</table>
mpSemTake

Takes a semaphore.

- Syntax

```c
STATUS mpSemTake
(
    SEM_ID semId, /* semaphore ID to take */
    int timeout /* timeout in ticks */
)
```

- Description

This routine takes the specified semaphore. The timeout can be specified in ticks. If the task times out, mpSemTake() returns an error. As `timeout`, WAIT_FOREVER (keep waiting without timeout) and NO_WAIT (end immediately even if the semaphore is not taken) can also be set.

- Parameter

  - **[semId]**
    The ID of the semaphore to be taken
  
  - **[timeout]**
    Specifies the timeout in ticks.
    If the task times out, mpSemTake() returns an error.
    - NO_WAIT (0)
      End immediately even if the semaphore is not taken
    - WAIT_FOREVER (-1)
      Keep waiting without timeout

- Return Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>-1</td>
<td>Error (when the semaphore ID is invalid or the task times out)</td>
</tr>
</tbody>
</table>
  
To retrieve the cause of the error, use mpErrSemTake.
mpSemGive

Gives a semaphore.

- **Syntax**
  
  ```c
  STATUS mpSemGive
  (  
    SEM_ID semId /* semaphore ID to give */
  )
  ```

- **Parameter**
  
  `[semId]`
  
  The ID of the semaphore to be given

- **Return Value**
  
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>-1</td>
<td>Error (when the semaphore ID is invalid)</td>
</tr>
</tbody>
</table>
mpErrSemTake

Retrieves an error when a semaphore is retrieved.

- **Syntax**

  int mpErrSemTake(void)

- **Description**

  This API retrieves the cause of the error which occurs when mpSemTake is executed. Only when retrieving it immediately after mpSemTake returns -1 (Error), the reference value is correct.

- **Parameter**

  None

- **Return Value**

  One of the following values returns.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_TIME_OUT</td>
<td>0x00000001 Timeout</td>
</tr>
<tr>
<td>E_NONEXIST</td>
<td>0x00000002 The semaphore was deleted</td>
</tr>
<tr>
<td>E_OBJECTS_ID</td>
<td>0x00000004 Object ID error</td>
</tr>
<tr>
<td>E_OBJ_NO_METHOD</td>
<td>0x00000005 Other than the above</td>
</tr>
</tbody>
</table>

- **Note**

  <Sample Code>

  ```c
  int foo(void)
  {
      if (mpSemTake(semId, timeout) == ERROR)
      {
          switch (mpErrSemTake())
          {
              case E_TIME_OUT:
                  // Some error processing is done.
                  break;
              case E_NONEXIST:
                  // Some error processing is done.
                  break;
              case E_OBJECTS_ID:
                  // Some error processing is done.
                  break;
          }
      }
  ```
Task Control API

mpErrSemTake

default://case E_OBJ_NO_METHOD:
  // Some error processing is done.
  break;
  return (NG);
}

...
  return (OK);
}
mpTaskDelay

Delays a task.

- **Syntax**

  STATUS mpTaskDelay
  (  
    int ticks /* number of ticks to delay task */  
  )

- **Description**

  This routine prevents the CPU from performing the task for the time (the number of ticks) specified in the calling task. This is generally called “manual schedule”, and can also be used when waiting for the external event which is not defined as an interruption.

- **Parameter**

  [ticks]
  
  The number of ticks to delay the task
  
  Cannot set to an endless wait.

- **Return Value**

  0 Normal end
  -1 Error

- **Note**

  The period of time per tick (msec) can be retrieved by the following API:

  int mpGetRtc(void)
  
  Return value: 1 (minimum) or more  
  (unit: msec)
mpGetRtc

Gets the clock cycle (1 tick) of the OS (VxWorks).

- **Syntax**
  
  int mpGetRtc(void)

- **Parameter**
  
  No parameter

- **Return Value**
  
  The OS clock cycle (unit: msec)
  
  Minimum value: 1
mpClkAnnounce

Notifies the I/O or interpolation cycle event.

- **Syntax**

  STATUS mpClkAnnounce
  (  
  int clk_id  
  )

- **Parameter**

  - **[ticks]**
    - clk_id
      - • MP_IO_CLK
        The I/O cycle is notified.
      - • MP_INTERPOLATION_CLK
        The interpolation cycle is notified.

  The task which called this function is suspended until each clock (event) is notified. (synchronized semaphore)

- **Return Value**

  Same as mpSemTake

- **Note**

  [example code]

  void mpUsrRoot(int arg1, int arg2,....,int arg10)
  {
    int tid;

    tid = mpCreateTask(MP_PRI_IO_CLK TAKE, MP_STACK_SIZE,
                        (FUNCPTR)mp_io_mon_task, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
  }

  void mp_io_mon_task(void)
  {
    FOREVER
    {
      mpClkAnnounce(MP_IO_CLK); // pend and arriving of clock

      // Something is done.
    }
  }

  // The interpolation cycle task is also similar.
Use precautions when you select a priority other than MP_PRI_TIME_NORMAL. The priority MP_PRI_IO_CLK_TAKE and MP_PRI_IP_CLK_TAKE have higher priorities than the man-machine control task. Thus, if unusually long processing time (100 microseconds or more) is used to describe a program with these priorities, critical problems may occur. (e.g. The system alarm occurs due to insufficient processing time for the robot control task; The programming pendant or the HOLD button freezes because the man-machine control task cannot be operated.)

MotoPlus detects such status (when the man-machine task does not run for 3 seconds or more), stops the robot motion by turning the servo power OFF, and turns ON the system output ($50901) as the protective function to take countermeasures for peripheral devices. Thus, make sure to set 100 microseconds as the maximum processing time in the high-priority task (100 microseconds equals to approx. 1000 lines in the C language). The OS waiting time, the message/semaphore waiting time, and the task delay time are not included in this processing time.

The following APIs can be used in the task which performs fixed-cycle processing by using mpClkAnnounce.

<table>
<thead>
<tr>
<th>Priority</th>
<th>Usable API</th>
</tr>
</thead>
<tbody>
<tr>
<td>For the notification task of the I/O control cycle</td>
<td>mpReadIO, mpWriteIO, mpGetUserVars, mpPutUserVars, mpMsgQReceive, mpMsgQSend, mpErrMsgQSend, mpSemTake, mpSemGive, mpTaskDelay, mpGetRtc, mpClkAnnounce, mpStopWatchCreate, mpStopWatchDelete, mpStopWatchStart, mpStopWatchStop, mpStopWatchLap, mpStopWatchReset, mpStopWatchGetTime, mpStopWatchGetLapNum, mpStopWatchGetLapTime, mpStopWatchGetAliveLapNo</td>
</tr>
<tr>
<td>For the notification task of the interpolation control cycle</td>
<td>Above APIs (APIs usable for the notification task of the I/O control cycle), mpGetVarData, mpMeiGetJobExecTask, mpMeiGetInterpolation, mpMeiGetExecControlGroup, mpMeiPutCorrPath, mpMeiPutForcePathEnd, mpMeiPutSpdOverride</td>
</tr>
</tbody>
</table>

If an API other than listed above is used, fixed-cycle processing cannot be performed due to prolonged processing time.
mpStopWatchCreate

Creates a stopwatch.

- Syntax

```c
void * mpStopWatchCreate
(unsigned long maxLapTimeNum /* Number of lap counter */)
```

- Description

APIs named mpStopWatchXxx() provide stopwatch functions using CPU time base. These functions are used to measure the time to execute processes. The data size of the time base is 64bits, and it is added by each CPU clock. mpStopWatchXxx APIs do not stop task scheduling and interruption. Thus, the measurement time may include the processing time of another task and interruption. The measurement time is in milliseconds (msec) and the floating point type.

For information on how to use mpStopWatchXxx() API, refer to “Chapter 2 Task, 2.6 Processing Time Measurement” of “FS100 OPTIONS INSTRUCTIONS Programmer’s Manual for New Language Environment MotoPlus (HW1480841)”.

- Parameter

`[maxLapTimeNum] Number of created lap time counter`

API creates the specified number of the stopwatches to store the lap time. When 0 is specified, the lap time cannot be measured, but only the time from start to stop can be measured.

- Return Value

<table>
<thead>
<tr>
<th>Stopwatch object ID</th>
<th>Normal end</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Error: The stopwatch cannot be created. (insufficient memory)</td>
</tr>
</tbody>
</table>

- Note

Make sure to call this routine before using the stopwatch. After that, all of the stopwatch routines use the stopwatch object ID created in this routine as the first parameter.
mpStopWatchDelete

Deletes the stopwatch.

- **Syntax**
  
  ```c
  void mpStopWatchDelete
  (  
    void * objStopWatch /* Stopwatch object ID */
  )
  ```

- **Parameter**
  
  `[objStopWatch]` *Stopwatch object ID*

- **Return Value**
  
  None

- **Note**
  
  This routine releases the memory area allocated for the stopwatch.
mpStopWatchStart

Starts the stopwatch.

- **Syntax**
  
  ```c
  void mpStopWatchStart
  (    
    void * objStopWatch    /* Stopwatch object ID */
  )
  ```

- **Parameter**
  
  `[objStopWatch] Stopwatch object ID`

- **Return Value**
  
  None
mpStopWatchStop

Stops the stopwatch.

- **Syntax**

  ```c
  void mpStopWatchStop
  (  
    void * objStopWatch /* Stopwatch object ID */
  )
  ```

- **Parameter**

  - `[objStopWatch]` Stopwatch object ID

- **Return Value**

  None
mpStopWatchLap

Records the lap time.

- **Syntax**
  
  ```c
  void mpStopWatchLap
  ( void * objStopWatch/* Stopwatch object ID */
  )
  ```

- **Parameter**
  
  `[objStopWatch] Stopwatch object ID`

- **Return Value**
  
  None

- **Note**
  
  If the lap time is measured multiple times and the limit of the lap time counter is exceeded, the lap time record is deleted in order from the oldest (from No. 0 of the lap time counter). Thus, the lap time records to be stored are of the latest few minutes of the lap time counter.
mpStopWatchReset

Resets the stopwatch.

- **Syntax**
  ```c
  void mpStopWatchReset
  (  
    void * objStopWatch /* Stopwatch object ID */  
  )
  ```

- **Parameter**
  
  `[objStopWatch] Stopwatch object ID`

- **Return Value**
  None

- **Note**
  This routine clears the lap time counter of the stopwatch. If the stopwatch does not have any lap time counter, this routine does not run.
mpStopWatchGetTime

Gets the elapsed time.

- **Syntax**

  ```c
  float mpStopWatchGetTime
  ( const void * objStopWatch /* Stopwatch object ID */ )
  ```

- **Parameter**

  `[objStopWatch] Stopwatch object ID`

- **Return Value**

  The measured elapsed time is returned in milliseconds. The elapsed time is the time from when the latest `mpStopWatchStart()` is called to when `mpStopWatchStop()` is called. However, even when `mpStopWatchStop()` is not called after the latest `mpStopWatchStart()`, if `mpStopWatchLap()` is called, the elapsed time until the latest `mpStopWatchLap()` is returned. In this case, the time measurement continues.
mpStopWatchGetLapNum

Gets the number of the lap time counter.

- Syntax

  unsigned long mpStopWatchGetLapNum
  (const void * objStopWatch /* Stopwatch object ID */)

- Parameter

  [objStopWatch] Stopwatch object ID

- Return Value

  The number of the lap time counter specified by mpStopWatchCreate().
mpStopWatchGetLapTime

Gets lap time.

- Syntax

```c
float mpStopWatchGetLapTime
  (const void * objStopWatch, /* Stopwatch object ID */
   unsigned long lapNo/* Lap number (zero origin) */)
```

- Parameter

  - `[objStopWatch] Stopwatch object ID`
  - `[lapNo] Lap time number`

When a number between 0 and the latest lap time number is specified, the lap time of the specified number is returned. To get the latest lap time and the oldest lap time, the following macros are provided.

- `MP_STOP_WATCH_GET_LAP_NEWEST` magic number to get the latest lap time
- `MP_STOP_WATCH_GET_LAP_OLDEST` magic number to get the oldest lap time

If the stopwatch does not have any lap time counter, 0 is always returned.

- Return value

  The lap time of the specified number is returned in milliseconds.
mpStopWatchGetAliveLapNo

Gets the range of the valid lap time numbers.

■ Syntax

```c
void mpStopWatchGetAliveLapNo
    (const void * objStopWatch, /* Stopwatch object ID */
     unsigned long *oldestNo, /* Number of the oldest lap */
     unsigned long *newestNo  /* Number of the last lap */
    )
```

■ Parameter

- `[objStopWatch] Stopwatch object ID`
- `[oldestNo] Oldest number of lap time set area`
- `[newestNo] Latest number of lap time set area`

■ Return value

None
3 User Watchdog API

mpUsrWdogCreate

Creates a user watchdog timer.

Syntax

```c
MP_WDG_HANDLE mpUsrWdogCreate
(
    int delay, /* timer value(0.001[sec]) */
    MP_USR_WDG_ROUTINE wdRoutine /* routine to call upon expiration */
)
```

Description

Up to 2 user watchdog timers can be created.

If the user watchdog timer times out, the system turns OFF the servo power of all control groups. In addition to turning OFF the servo power, if necessary, the user can define another process to be executed when the timer times out. To do so, specify the process in the parameter, user watchdog time-out routine.

Parameter

- **[delay]**
  Specify the timer value of the user watchdog timer in milliseconds (1 msec or more).

- **[wdRoutine]**
  Specify a desired user watchdog time-out routine. Specify the process necessary when the user watchdog timer times out. If no process is necessary at the time-out, specify NULL.

  The time-out routine does not have a return value. The time-out routine receives the handle value of the user watchdog timer which timed out as the parameter. Thus, since the timed-out user watchdog timer can be distinguished in the time-out routine, one time-out routine can be shared by two watchdog timers.

  In addition, the time-out routine is executed in the task of the system. Therefore, if the task of the time-out routine is tried to be deleted in the time-out routine itself (if mpDeleteTask(TID_SELF) is tried to be executed), ERROR is returned.

  - MP_USR_WDG_ROUTINE
    Time-out routine
    Syntax:
    ```c
typedef void (*MP_USR_WDG_ROUTINE) (MP_WDG_HANDLE);
    ```

    Example of time-out routine:
    ```c
    void fncwdRoutine0( MP_WDG_HANDLE handle )
    {
        ...
    }
    ```
### Return Value

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 or more</td>
<td>Normal end</td>
</tr>
<tr>
<td></td>
<td>The characteristic value of the created user watchdog timer is returned as the handle value.</td>
</tr>
<tr>
<td>ERROR</td>
<td>Error</td>
</tr>
<tr>
<td></td>
<td>In the case when the delay value is 0 or less, 2 user watchdog timers are already created, and the like.</td>
</tr>
</tbody>
</table>
mpUsrWdogDelete

Deletes the user watchdog timer.

- **Syntax**
  ```c
  int mpUsrWdogDelete
  (MP_WDG_HANDLE handle /* specified watchdog timer */)
  ```

- **Description**
  Deletes the user watchdog timer specified in “handle”.

- **Parameter**
  ```c
  [handle]
  ```
  Specify the handle value of the user watchdog timer to be deleted.

- **Return Value**
  - **OK**: Normal end
  - **ERROR**: Error
    In the case when an invalid handle value is specified, and the like.
mpUsrWdogStart

Starts the user watchdog timer.

- **Syntax**
  ```c
  int mpUsrWdogStart
  (    
    MP_WDG_HANDLE handle /* specified watchdog timer */
  )
  ```

- **Description**
  Starts the user watchdog timer specified in “handle”.
  Also, this API is used to start the user watchdog timer again when the user watchdog timer times out.

- **Parameter**
  - **[handle]**
    Specify the handle value of the user watchdog timer to be started.

- **Return Value**
  - **OK** Normal end
  - **ERROR** Error
    In the case when an invalid handle value is specified, and the like.
mpUsrWdogClear

Clears the user watchdog timer.

- **Syntax**
  ```c
  int mpUsrWdogClear
  (MP_WDG_HANDLE handle /* specified watchdog timer */)
  ```

- **Description**
  Clears the user watchdog timer specified in “handle”.
  This API must be executed periodically after the user watchdog timer is started.

- **Parameter**
  - **[handle]**
    Specify the handle value of the user watchdog timer to be cleared.

- **Return Value**
  - **OK** Normal end
  - **ERROR** Error
    In the case when an invalid handle value is specified, and the like.
mpSocket

Opens a socket.

- **Syntax**
  
  ```c
  int mpSocket (int domain, int type, int protocol);
  ```

- **Description**
  
  This routine opens a socket and returns a socket descriptor. The socket descriptor is passed to the other socket routines to identify the socket. The socket descriptor is a standard I/O system file descriptor (fd).

- **Parameter**
  
  - **[domain]**
    
    Specifies the protocol family for communication. The following can be set.

    | Value       | Meaning          |
    |-------------|------------------|
    | AF_INET     | IPv4 internet protocol |

  - **[type]**
    
    Specifies the communication type.

    | Value          | Meaning |
    |----------------|---------|
    | SOCK_STREAM    | TCP     |
    | SOCK_DGRAM     | UDP     |

  - **[protocol]**
    
    Protocol number (fixed by 0)

- **Return Value**
  
  - Value >=0  Socket descriptor
  - -1          Error
mpListen

Gives the server attribute to the socket and waits for connection.

- **Syntax**
  ```c
  int mpListen(int s, int backlog);
  ```

- **Description**
  This routine enables connections to a socket. It also specifies the maximum number of unaccepted connections that can be suspended at one time (backlog). After enabling connections with `mpListen()`, connections are actually accepted by `mpAccept()`.

- **Parameter**
  - `[s]`  
    The descriptor which refers to the socket
  - `[backlog]`  
    The number of the I/O data

- **Return Value**
  - 0  Normal end
  - -1  Error
mpAccept

Accepts a connection from the client to the socket.

- **Syntax**
  
  ```c
  int mpAccept(int s, struct sockaddr *addr, int *addrlen);
  ```

- **Description**
  
  This routine accepts a connection on a socket, and returns a new socket created for the connection. The socket must be bound to an address with `mpBind()`, and enabled for connections by a call to `mpListen()`. The `mpAccept()` routine dequeues the first connection and creates a new socket with the same properties as `s`. It blocks the caller until a connection is present, unless the socket is marked as non-blocking.

  The parameter `addrlen` must be initialized to the size of the available buffer pointed to by `addr`. Upon return, `addrlen` contains the size in bytes of the peer's address stored in `addr`.

- **Parameter**

  - **[s]**
    
    The socket descriptor waiting for a connection by `mpListen`

  - **[addr]**
    
    [out] The pointer to the sockaddr structure for the socket address of the communication target

  - **[addrlen]**
    
    [in] [out] At calling, this must be initialized with the structure size (in bytes) specified by `addr`. At return, the actual size of the connecting target address is set.

- **Return Value**

  - `Value >= 0` The file descriptor of the socket
  - `-1` Error
mpBind

Sets an address to the socket.

- **Syntax**
  ```c
  int mpBind(int s, struct sockaddr *name, int namelen);
  ```

- **Description**
  This routine associates a network address (also referred to as its “name”) with a specified socket so that other processes can connect or send to it. When a socket is created with mpSocket(), it belongs to an address family but has no assigned name.

- **Parameter**
  - **[s]**
    The socket descriptor created by mpSocket
  - **[name]**
    The pointer of the sockaddr structure kept socket address.
    The actual pointer of name depends on address family but MotoPlus supports only sockaddr_in.
  - **[namelen]**
    The size of the address structure of the name (unit: byte)

- **Return Value**
  - 0 Normal end
  - -1 Error
mpConnect

Initiates a connection to the socket.

- **Syntax**

  ```c
  int mpConnect(int s, struct sockaddr *name, int namelen);
  ```

- **Description**

  If `s` is a socket of type SOCK_STREAM, this routine establishes a virtual circuit between `s` and another socket specified by the name. If `s` is of type SOCK_DGRAM, it permanently specifies the peer to which messages are sent. The name parameter specifies the address of the other socket.

- **Parameter**

  - `[s]`
    The socket descriptor created by mpSocket
  
  - `[addr]`
    [in] The pointer of the sockaddr structure which keeps the socket address
    The actual pointer of `addr` depends on address family but MotoPlus supports only sockaddr_in.
  
  - `[addrlen]`
    The size of the address structure of `addr` (unit: byte)

- **Return Value**

  - 0   Normal end
  
  - -1  Error
mpRecv
mpRecvFrom

Receives a message from the socket.

- **Syntax**

  ```c
  int mpRecv(int s, char *buf, int bufLen, int flags);
  int mpRecvFrom(int s, char *buf, int bufLen, int flags,
                  struct sockaddr *from, int *pFromLen);
  ```

- **Description**

  This routine receives a message from a connection-based (stream) socket. mpRecv is mainly used by TCP, mpRecvFrom is mainly used by UDP.

- **Parameter**

  - `[s]`  
    Socket descriptor
  
  - `[buf]`  
    The pointer to the buffer for the received data
  
  - `[bufLen]`  
    The maximum allowable length of received data to be stored in `buf` (unit: byte)
  
  - `[flags]`  
    Set "0".
  
  - `[from]`  
    [in] The pointer to the sockaddr structure which stores the socket address
  
  - `[pFromLen]`  
    [in] [out] At calling, this must be initialized with the structure size (in bytes) specified by `from`. At return, the actual size of the connecting target address is set.

- **Return Value**

  - `Value >= 0`  
    The number of bytes received (normal end)
  
  - `-1`  
    Error
mpSend
mpSendTo

Sends a message to the socket.

- **Syntax**

  ```c
  int mpSend(int s, const char *buf, int bufLen, int flags);
  int mpSendTo(int s, char *buf, int bufLen, int flags,
               struct sockaddr *to, int tolen);
  ```

- **Description**

  Sends data to a socket. mpSend() is mainly used by TCP. mpSendTo() is mainly used by UDP.

- **Parameter**

  - **[s]**  
    Socket descriptor
  - **[buf]**  
    The pointer to the buffer for the sent data
  - **[bufLen]**  
    The length of the sent data stored in buf (unit: byte)
  - **[flags]**  
    Set "0".
  - **[to]**  
    [in] The pointer to the sockaddr structure which stores the socket address
  - **[toLen]**  
    The size of the address structure specified by to (unit: byte)

- **Return Value**

  - Value \(\geq 0\)  
    The number of bytes sent (normal end)
  - -1  
    Error
mpClose

Closes the socket descriptor.

■ Syntax

STATUS mpClose(int fd);

■ Description

Closes the socket descriptor.

■ Parameter

[fd]
Socket descriptor

■ Return Value

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>-1</td>
<td>Error</td>
</tr>
</tbody>
</table>

For mpSocket() and mpAccept(), memory in the heap area is temporarily used. Thus, if mpSocket() and mpAccept() is repeated without doing mpClose(), the above working memory is consumed and the system memory becomes insufficient. Then the FS100 hangs up, the communication error of the programming pendant occurs, and the robot stops its operation. Thus, if you need to repeat mpSocket() and mpAccept(), make sure to do mpClose() each time.
mpHtonl
mpHtons
mpNtohl
mpNtohs

Converts a value between the host byte order and the network byte order.

- **Syntax**

  ULONG  mpHtonl(ULONG  hostlong);
  USHORT mpHtons(USHORT  hostshort);
  ULONG  mpNtohl(ULONG  netlong);
  USHORT mpNtohs(USHORT  netshort);

- **Description**

  mpHtonl() converts the *hostlong* from the host byte order to the network byte order.
  mpHtons() converts the *hostshort* from the host byte order to the network byte order.
  mpNtohl() converts the *netlong* from the network byte order to the host byte order.
  mpNtohs() converts the *netshort* from the network byte order to the host byte order.
mpInetAddr
mpInetNtoa
mpInetNtoaB

Converts the internet host address between the dot-decimal notation and the binary representation in network byte order.

■ Syntax

unsigned long mpInetAddr(char *host);
char *mpInetNtoa(struct in_addr in);
void mpInetNtoaB(struct in_addr inetAddress, char *pString);

■ Description

mpInetAddr() converts the internet host address *host from the dot-decimal notation of IPv4 to the binary representation in network byte order.

mpInetNtoa() converts the internet host address *in from the binary representation in network byte order to the dot-decimal notation of IPv4.

mpInetNtoaB() converts the internet host address *inetAddress from the binary representation in network byte order to the dot-decimal notation of IPv4, and returns the result to *pString.

Although the functions of mpInetNtoa() and mpInetNtoaB() are the same, it is recommended to use mpInetNtoaB() because mpInetNtoa() is not thread-safe.
mpGetsockname

Gets the socket name.

- **Syntax**

  ```c
  int mpGetsockname(int s, struct sockaddr *name, int *namelen);
  ```

- **Description**

  The current socket address specified by `s` is returned by storing in the buffer specified by `name`. The `namelen` parameter must be initialized to the data size (in bytes) specified by `name`. When the parameter returns, the actual size (in bytes) of the socket address is stored in `namelen`.

- **Parameter**

  - **[s]**
    
    Socket descriptor
  
  - **[name]**
    
    [out] The pointer of the sockaddr structure for the socket address
  
  - **[namelen]**
    
    [in] [out] At calling, this must be initialized to the data size (in bytes) specified by `name`. At return, the actual size of the address is stored.

- **Return Value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>-1</td>
<td>Error</td>
</tr>
</tbody>
</table>
mpGetpeername

Gets the name of the peer connected to a socket.

- **Syntax**
  
  ```c
  int mpGetpeername(int s, struct sockaddr *name, int *namelen);
  ```

- **Parameter**
  
  - `[s]`
    
    Specifies the socket descriptor to get its peer name.
  
  - `[name]`
    
    Specifies the address of the buffer which stores the address of the peer.
  
  - `[namelen]`
    
    Specifies the size of the area which `name` indicates.

- **Return Value**
  
  - `0` Normal end
  - `-1` Error
mpSetsockopt

Sets socket options. (Sets options for disconnection.)

- **Syntax**

  ```c
  int mpSetsockopt(int s, int level, int optname, char *optval, int optlen);
  ```

- **Parameter**

  - **[s]**
    The socket descriptor manipulating options
  
  - **[level]**
    Specify `SOL_SOCKET` for the manipulating option layer `mpSetsockopt()`.
  
  - **[optname]**
    Only `SO_KEEPALIVE` can be set for `mpSetsockopt()` specifying the manipulated option name

  **SO_KEEPALIVE -- Disconnection detection**

  The disconnection detection time, which enables the timer detecting disconnection at TCP level, is fixed to two hours.

  If a disconnection is detected, the socket specified as `s` is closed.

  - **[optval]**
    Specifies an appropriate type depending on the area address `optname` that stores the manipulating option values

  - **[optlen]**
    Specifies the area size specified by the manipulating option size `optval`

<table>
<thead>
<tr>
<th>Option name: Optname</th>
<th>The area type specified by optval</th>
<th>The value of optval</th>
<th>Optlen</th>
</tr>
</thead>
<tbody>
<tr>
<td>SO_KEEPALIVE</td>
<td>int</td>
<td>1: ON</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: OFF</td>
<td></td>
</tr>
</tbody>
</table>

- **Return Value**

  0       Normal end
  
  -1      Error
mpioctl

Controls I/O devices (clears buffers, sets blocking / non-blocking)

■ Syntax

```c
int mpioctl
  (  
    int fd,          /* file descriptor (target socket) */
    int function,   /* function code */
    int arg         /* arbitrary argument */
  )
```

■ Parameter

**[fd]**
The file descriptor of the controlled I/O device
Set the socket descriptor, here.

**[function]**
Controlled function
- FIOFLUSH  -- Buffer flush
- FIONBIO   -- Blocking / non-blocking setting

**[arg]**
Control function option
Set the value passed to the controlled function.

<table>
<thead>
<tr>
<th>Function name: Function</th>
<th>The area type specified by arg</th>
<th>The value of arg</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIOFLUSH</td>
<td>int</td>
<td>1: ON</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: OFF</td>
</tr>
<tr>
<td>FIONBIO</td>
<td>int</td>
<td>0: Blocking</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Non-blocking</td>
</tr>
</tbody>
</table>

■ Return Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>-1</td>
<td>Error (Incorrect file descriptor, incorrect parameter specification)</td>
</tr>
</tbody>
</table>
mpSelect

Keeps a task waiting for multiple sockets or examines whether mpSend and mpRecv can be executed to the socket with non-blocking

- **Syntax**

```c
int mpSelect
(
    int width,    /* number of bits to examine from 0 */
    fd_set * pReadFds,    /* read fds */
    fd_set * pWriteFds,    /* write fds */
    fd_set * pExceptFds,    /* exception fds (unsupported) */
    struct timeval * pTimeOut    /* max time to wait, NULL = forever */
)
```

- **Parameter**

1. **[width]**
   
The value to define the number of bits examined in fd_set. Set the value of the current maximum socket file descriptor value + 1, or specify FD_SETSIZE.

2. **[pReadFds]**
   
   Specifies the fd_set structure that sets the bit of the socket file descriptor waiting for reading. When not waiting for reading, set NULL. At the end of execution, mpSelect() clears the socket file descriptor bit to 0, and sets only the bit corresponding to the socket file descriptor in ready state.

```c
struct fd_set
{
    fd_mask fds_bits[howmany(FD_SETSIZE, NFDBITS)];
};
```

   
```c
#define FD_SETSIZE2048
typedef long fd_mask;
#define howmany(x, y)((unsigned int)(((x)+((y)-1)))/(unsigned int)(y))
```

   
To set a bit to the fd_set structure, use the following macros.

```c
FD_SET(fd, &fdset)
FD_CLR(fd, &fdset)
FD_ZERO(&fdset)
```
To check the socket file descriptor usable when finishing mpSelect(), use the following macro.

FD_ISSET(fd, &fdset)

[pWriteFds]
Specifies the fd_set structure that sets the bit of the socket file descriptor waiting for writing. When not waiting for writing, set NULL. At the end of execution, mpSelect() clears the socket file descriptor bit of the fd_set structure to 0, and sets only the bit corresponding to the socket file descriptor in ready state.

[pExeptFds]
Set NULL, always.

[pTimeout]
Specifies the timeval structure setting the maximum waiting time. When waiting without timeout, set NULL. The following members are provided for the timeval structure.

struct timeval {
    long tv_sec; /* second */
    long tv_usec; /* micro second */
};

■ Return Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Timeout</td>
</tr>
<tr>
<td>Value ≥ 0</td>
<td>The number of available sockets</td>
</tr>
<tr>
<td>-1</td>
<td>Error</td>
</tr>
</tbody>
</table>
5 Serial Communication API

mpRsInit

Initializes the specified serial communication port.

- Syntax

```c
int mpRsInit
(
    const char * port, /* name of port */
    const MP_RS_CONFIG * rs_config /* COM port parameter */
)
```

- Parameter

The transmission parameter of the serial communication port with the character string specified by `port` is set to the conditions specified by `rs_config` parameter. This API must be done before using another serial communication API. If the target serial communication port is already opened, this API returns an error. Only the ports defined by the macro definition `MP_RS_COM1` are currently supported. Make sure to set `MP_RS_COM1` as `port`.

`MP_RS_CONFIG`, the structure of the `rs_config` parameter, has the following members.

```c
<MP_RS_CONFIG structure>

typedef struct {
    MP_RS_DATA_BIT  dataBit;
    MP_RS_STOP_BIT  stopBit;
    MP_RS_PARITY    parity;
    MP_RS_BAUDRATE  baudRate;
} MP_RS_CONFIG;
```

- `.dataBit` : Data bit num
  - `mpRsDataBit_7` : 7bits
  - `mpRsDataBit_8` : 8bits

- `.stopBit` : Stop bit mode (length)
  - `mpRsStopBit_one` : 1bit
  - `mpRsStopBit_1point5` : 1.5bits
  - `mpRsStopBit_two` : 2bits
Serial Communication API

mpRsInit

- parity : Parity
  - mpRsParity_none : None
  - mpRsParity_odd : odd parity
  - mpRsParity_even : even parity

- baudRate : Baud rate
  - mpRsBaudrate_150 : 150 bps
  - mpRsBaudrate_300 : 300 bps
  - mpRsBaudrate_600 : 600 bps
  - mpRsBaudrate_1200 : 1200 bps
  - mpRsBaudrate_2400 : 2400 bps
  - mpRsBaudrate_4800 : 4800 bps
  - mpRsBaudrate_9600 : 9600 bps
  - mpRsBaudrate_19200 : 19200 bps

Return Value

0 Normal end
-1 Error
mpRsOpen

Opens the specified serial communication port.

- **Syntax**

  ```c
  int mpRsOpen
  (const char * port, /* name of port */
   )
  ```

- **Parameter**

  This API opens the serial communication port with the character string specified by `port`. This API must be done before using the send/receive API. If the target serial communication port is already opened, this API returns an error. Make sure to set MP_RS_COM1 as `port`.

- **Return Value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value &gt;= 0</td>
<td>Handle value (normal end)</td>
</tr>
<tr>
<td>-1</td>
<td>Error</td>
</tr>
</tbody>
</table>
mpRsClose

Closes the specified serial communication port.

- **Syntax**
  ```c
  int mpRsClose
  (  
    int port_handle,  /* handle value of port opened */
  )
  ```

- **Parameter**
  This API closes the specified serial communication port by `port_handle`.

- **Return Value**
  - 0 Normal end
  - -1 Error
mpRsSend

Sends data to the serial communication port.

- **Syntax**

```c
int mpRsClose
(
    int port_handle, /* handle value of port opened */
    const char *buf, /* send data buffer pointer */
    unsigned int buf_len /* length of buffer data */
)
```

- **Description**

This function sets the send data to the sending FIFO buffer (1 Kbyte). If the buffer is not enough, this API returns error.

- **Parameter**

  - `port_handle`: Handle value returned from mpRsOpen execution
  - `buf`: Pointer of the send data
  - `buf_len`: Buffer size (unit: byte)

- **Return Value**

  - `0`: Normal end
  - `-10`: The length of the send data exceeds the available buffer size. The buffer size is 1 Kbyte.
  - `-1`: Error (another parameter error)
mpRsRecv

Receives data from the serial communication port.
This function reads the receive data from the receiving FIFO buffer (1024 bytes).

- **Syntax**

  ```c
  int mpRsRecv
  (  
    int   port_handle, /* handle value of port opened */
    char  * buf,       /* receive data buffer pointer */
    unsigned int buf_len /* length of buffer area */
  )
  ```

- **Parameter**

  - `port_handle`: Handle value returned from mpRsOpen execution
  - `buf`: Pointer of the receive data readout area
  - `buf_len`: Allowable readout buffer size (Max `buf` size)

- **NOTE**

  If nothing is received (nothing in the buffer), the return value is "0", not an error.

- **Return Value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;=0</td>
<td>Received data size (copied from the FIFO buffer)</td>
</tr>
<tr>
<td>-1</td>
<td>Error</td>
</tr>
</tbody>
</table>
### mpMalloc

Allocates a block with the size bytes from the heap memory. The reserved block is not cleared to 0.

This function makes it possible to reserve the memory with any necessary size at any necessary location, without allocating the memory in advance by using arrays in the program.  

1 MB area is prepared in advance for MotoPlus. However, since the management domain is included in this area, all the 1 MB area cannot be used.

Release the memory, which is reserved with the mpMalloc function, after the memory is used, by using the mpFree function.

#### Syntax

```c
void *mpMalloc(size_t size);
```

#### Parameter

[size]

The memory byte sizes to be reserved

#### Return Value

- **Success**: The pointer to the reserved memory block
- **Failure**: NULL (when the memory with the specified size cannot be reserved: insufficient memory)
mpFree

(Free function) Releases the area specified by ptr to make it available for subsequent allocations. The function does nothing when ptr is an empty pointer.

- Syntax
  
  ```c
  void mpFree(void *ptr);
  ```

- Parameter
  
  `[ptr]`
  
  The pointer to the area to be released

- Return Value
  
  None

**NOTE**

If the area not reserved is specified, a serious problem such as task interruption may occur.
mpCreate

Creates a new file.
If an existing file name is specified, the old file is overwritten.

- Syntax

```c
int mpCreate /* return : fd */
(const char * name, /* name of the file to create */
  int flags /* O_RDONLY, O_WRONLY, or O_RDWR */);
```

- Parameter

  - [name]
  The file name to be created
  Make sure to specify the name with the full pathname. Use
  MP_SRAM_DEV_DOS ("MPSRAM1:0") or
  MP_DRAM_DEV_DOS ("MPRAM1:0")
  as the drive name.
  If an existing file name is specified, the file is overwritten, and the previous
  file contents are deleted.

  - [flags]
  The access right to the file

<table>
<thead>
<tr>
<th>Access right</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>O_RDONLY</td>
<td>0</td>
<td>Read only</td>
</tr>
<tr>
<td>O_WRONLY</td>
<td>1</td>
<td>Write only</td>
</tr>
<tr>
<td>O_RDWR</td>
<td>2</td>
<td>Read and write</td>
</tr>
</tbody>
</table>

- Return Value

  - fd (>=0) Normal end (file descriptor number)
  - -1 Error (Insufficient existing files or file descriptors, no
    specified directory)

- Note

  The maximum number of file descriptors that can be used simultaneously
  is MP_FILES (16).
FileControl API is executed only with the priority MP_PRI_TIME_NORMAL. When a task with a higher priority than MP_PRI_TIME_NORMAL calls FileControl API, and the task priority limitation is set to "Automatic Adjustment" (S2C1101 = 1), the system temporarily lowers the priority to MP_PRI_TIME_NORMAL, and executes the function. If the task priority limitation is set to "limitation" (S2C1101 = 0), the API returns an error.
mpOpen

Opens a file.

- **Syntax**

  ```c
  int mpOpen /* return : fd */
  (const char* name, /* name of the file to open */
   int flags, /* O_RDONLY, O_WRONLY, O_RDWR, or O_CREAT */
   int mode /* mode of file to create (UNIX chmod style) */
  );
  ```

- **Parameter**

  - **[name]**
    The name of the file to be opened
    Make sure to specify the name with the full pathname. Use MP_SRAM_DEV_DOS ("MPSRAM1:0") or MPDRAM_DEV_DOS ("MPRAM1:0") as the drive name.

  - **[flags]**
    The access right to the file
    | Access right | Value | Description |
    |---------------|-------|-------------|
    | O_RDONLY      | 0     | Read only   |
    | O_WRONLY      | 1     | Write only  |
    | O_RDWR        | 2     | Read and write |
    | O_CREAT       | 0x0200| Creates and opens a file (Use this in combination with one of the above three access rights.) |

  - **[mode]**
    Not used

- **Return Value**

  - `fd (>=0)`: Normal end (file descriptor number)
  - `-1`: Error (Insufficient existing files or file descriptors, no specified directory)

- **Note**

  The maximum number of file descriptors that can be used simultaneously is MP_FILES (16).
File Control API is executed only with the priority MP_PRI_TIME_NORMAL. When a task with a higher priority than MP_PRI_TIME_NORMAL calls FileControl API, and the task priority limitation is set to "Automatic Adjustment" (S2C1101 = 1), the system temporarily lowers the priority to MP_PRI_TIME_NORMAL, and executes the function. If the task priority limitation is set to "limitation" (S2C1101 = 0), the API returns an error.
mpRemove

Removes the file.

- **Syntax**
  
  ```c
  STATUS mpRemove /* return : status */
  (const char * name /* name of the file to remove */);
  ```

- **Parameter**
  
  `[name]`

  The file name to be removed

  Make sure to specify the name with the full pathname. Use `MP_SRAM_DEV_DOS ("MPSRAM1:0")` or `MP_DRAM_DEV_DOS ("MPRAM1:0")` as the drive name.

- **Return Value**
  
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>-1</td>
<td>Error (Incorrect specified file descriptor)</td>
</tr>
</tbody>
</table>

**NOTE**

FileControl API is executed only with the priority `MP_PRI_TIME_NORMAL`. When a task with a higher priority than `MP_PRI_TIME_NORMAL` calls FileControl API, and the task priority limitation is set to "Automatic Adjustment" (S2C1101 = 1), the system temporarily lowers the priority to `MP_PRI_TIME_NORMAL`, and executes the function. If the task priority limitation is set to "limitation" (S2C1101 = 0), the API returns an error.
mpClose

Closes the file or the socket.

**Syntax**

```c
STATUS mpClose /* return : status */
    ( int fd /* file descriptor to close */ );
```

**Parameter**

- **[fd]**
  - The file descriptor number of the file to be closed

**Return Value**

- 0 Normal end
- -1 Error (Incorrect specified file descriptor)

---

**NOTE**

FileControl API is executed only with the priority MP_PRI_TIME_NORMAL. When a task with a higher priority than MP_PRI_TIME_NORMAL calls FileControl API, and the task priority limitation is set to "Automatic Adjustment" (S2C1101 = 1), the system temporarily lowers the priority to MP_PRI_TIME_NORMAL, and executes the function. If the task priority limitation is set to "limitation" (S2C1101 = 0), the API returns an error.
mpRename

Renames a file.

- **Syntax**

  ```c
  int mpRename /* return : OK or ERROR */
  (  
     const char * oldName, /* name of file to rename */
     const char * newName /* name with which to rename file */
  );
  ```

- **Parameter**

  - **[oldName]**
    Old file name
  
  - **[newName]**
    New file name

- **Return Value**

  - 0 Normal end
  - -1 Error (Existing opened file, no specified file)

**NOTE**

FileControl API is executed only with the priority MP_PRI_TIME_NORMAL. When a task with a higher priority than MP_PRI_TIME_NORMAL calls FileControl API, and the task priority limitation is set to "Automatic Adjustment" (S2C1101 = 1), the system temporarily lowers the priority to MP_PRI_TIME_NORMAL, and executes the function. If the task priority limitation is set to "limitation" (S2C1101 = 0), the API returns an error.
mpRead

Reads data from a file.

**Syntax**

```c
int mpRead /* return : Number of bytes read (between 1 and maxbytes, */
   ( /*          0 if end of file), or ERROR, by read()) */
   int    fd, /* file descriptor from which to read */
   char * buffer, /* pointer to buffer to receive bytes */
   size_t maxBytes /* max no. of bytes to read into buffer */
);```

**Parameter**

- **[fd]**
  The file descriptor number of the file to be read

- **[buffer]**
  The address storing the data

- **[maxBytes]**
  The number of maximum bytes

**Return Value**

- `>=0` Normal end (The number of actually read bytes)
- `-1` Error (Incorrect specified file descriptor, unopened file for reading)

**Note**

When this API is used, the file pointer of the file specified as fd moves to the end of the read data. When reading or writing after that, the process starts from there. To read or write from the beginning of the file, re-open the file, or move the file pointer by using mpLseek.

---

FileControl API is executed only with the priority MP_PRI_TIME_NORMAL. When a task with a higher priority than MP_PRI_TIME_NORMAL calls FileControl API, and the task priority limitation is set to "Automatic Adjustment" (S2C1101 = 1), the system temporarily lowers the priority to MP_PRI_TIME_NORMAL, and executes the function. If the task priority limitation is set to "limitation" (S2C1101 = 0), the API returns an error.
mpWrite

Writes data to a file.

- **Syntax**
  ```c
  int mpWrite /* return : Number of bytes written (if not equal to *nbytes, an error has occurred), or ERROR */
  ( /* file descriptor on which to write */
    int fd, /* file descriptor number of the file to be written */
    char * buffer, /* buffer containing bytes to be written */
    size_t nBytes /* number of bytes to write */
  );
  ```

- **Parameter**
  
  - **[fd]**
    The file descriptor number of the file to be written
  
  - **[buffer]**
    The address where the data is written
  
  - **[nBytes]**
    The number of bytes

- **Return Value**
  
  - >=0 Normal end (The number of actually written bytes)
  - -1 Error (Incorrect specified file descriptor, unopened file for writing, insufficient disk capacity)

- **Note**
  
  When this API is used, the file pointer of the file specified as fd moves to the end of the written data. When reading or writing after that, the process starts from there. To read or write from the beginning of the file, re-open the file, or move the file pointer by using mpLseek.

---

**NOTE**

FileControl API is executed only with the priority MP_PRI_TIME_NORMAL. When a task with a higher priority than MP_PRI_TIME_NORMAL calls FileControl API, and the task priority limitation is set to "Automatic Adjustment" (S2C1101 = 1), the system temporarily lowers the priority to MP_PRI_TIME_NORMAL, and executes the function. If the task priority limitation is set to "limitation" (S2C1101 = 0), the API returns an error.
mpioctl

Controls devices.

- **Syntax**

  ```c
  int mpioctl  /* return : Return value of the driver, or ERROR */
  (    int fd, /* file descriptor */
       int function, /* function code */
       int arg    /* arbitrary argument */
    );
  ```

- **Parameter**

  - `[fd]`
    The file descriptor number of the file to be written

  - `[function]`
    The request code depending on the device
    The support function is different according to the fd type.
### File Control API

**mpIoctl**

**fd:** RAM drive

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>ret = mpIoctl (fd, function, arg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIOATTRIBSET</td>
<td>Changes the file attribute byte in the directory entry to the attribute specified as newAttrib. MotoPlus supports only 0 (No attribute) or MP_RAMDRV_ATTR_RDONLY (read-only).</td>
<td>status = mpIoctl (fd, FIOATTRIBSET, newAttrib);</td>
</tr>
<tr>
<td>FIOGETNAME</td>
<td>Retrieves the file name specified by fd.</td>
<td>status = mpIoctl (fd, FIOGETNAME, &amp;nameBuf);</td>
</tr>
<tr>
<td>FIOMOVE</td>
<td>Moves the file without renaming it.</td>
<td></td>
</tr>
<tr>
<td>FIONCONTIG</td>
<td>Retrieves the maximum continuous area size on the device.</td>
<td>status = mpIoctl (fd, FIONCONTIG, bytesRequested);</td>
</tr>
<tr>
<td>FIONFREE</td>
<td>Retrieves the number of bytes in the unused area on the volume.</td>
<td>status = mpIoctl (fd, FIONFREE, &amp;freeCount);</td>
</tr>
<tr>
<td>FIONREAD</td>
<td>Retrieves the number of unread bytes.</td>
<td>status = mpIoctl (fd, FIONREAD, &amp;nBytesUnread);</td>
</tr>
<tr>
<td>FIOTRUNC</td>
<td>Truncates the file to the specified length.</td>
<td>status = mpIoctl (fd, FIOTRUNC, newLength);</td>
</tr>
<tr>
<td>FIOWHERE</td>
<td>Returns the current byte position of the file.</td>
<td>position = mpIoctl (fd, FIOWHERE, 0);</td>
</tr>
</tbody>
</table>

**[arg]**

The parameter corresponding to the request code

- **Return Value**
  
  >=0     Normal end (The return value depending on the device)
  
  -1      Error (Incorrect specified file descriptor)

---

**NOTE**

FileControl API is executed only with the priority MP_PRI_TIME_NORMAL. When a task with a higher priority than MP_PRI_TIME_NORMAL calls FileControl API, and the task priority limitation is set to "Automatic Adjustment" (S2C1101 = 1), the system temporarily lowers the priority to MP_PRI_TIME_NORMAL, and executes the function. If the task priority limitation is set to "limitation" (S2C1101 = 0), the API returns an error.
mpLseek

Moves a file pointer.

Syntax

```c
int mpLseek /* return : New offset from the beginning of the file, or ERROR */
    (int fd, /* file descriptor */
     long offset, /* new byte offset to seek to */
     int origin /* relative file position */);
```

Parameter

- **[fd]**
  File descriptor number

- **[offset]**
  The number of bytes from origin

- **[origin]**
  Reference position
  - SEEK_SET: The top of the file
  - SEEK_CUR: The current position of the file pointer
  - SEEK_END: The end of the file

Return Value

- `>=0`: Normal end (The number of offset bytes from the top of the file to the new position)
- `-1`: Error (Incorrect specified file descriptor)

**NOTE**

FileControl API is executed only with the priority MP_PRI_TIME_NORMAL. When a task with a higher priority than MP_PRI_TIME_NORMAL calls FileControl API, and the task priority limitation is set to "Automatic Adjustment" (S2C1101 = 1), the system temporarily lowers the priority to MP_PRI_TIME_NORMAL, and executes the function. If the task priority limitation is set to "limitation" (S2C1101 = 0), the API returns an error.
mpFstat

mpStat

Retrieves file status information.

- Syntax

```c
STATUS mpFstat /* return : OK or ERROR by fstat() */
    (
        int    fd, /* file descriptor for file to check */
        struct stat * pStat /* pointer to stat structure */
    );

STATUS mpStat /* return : OK or ERROR by stat() */
    (
        char *    name, /* name of file to check */
        struct stat * pStat /* pointer to stat structure */
    );
```

- Parameter

  - [fd]
    File descriptor number
  - [name]
    File name
  - [pStat]
    The address to the stat structure

The structure of the file information status

```c
struct  stat
{
    ULONG st_dev; /* device ID number */
    ULONG st_ino; /* file serial number */
    USHORT st_mode; /* file mode (see below) */
    short st_nlink; /* number of links to file */
    short st_uid; /* user ID of file's owner */
    short st_gid; /* group ID of file's group */
    ULONG st_rdev; /* device ID, only if special file */
    ULONG st_size; /* size of file, in bytes */
    TIME st_atime; /* time of last access */
    TIME st_mtime; /* time of last modification */
    TIME st_ctime; /* time of last change of file status */
    long  st_blksize;
    long  st_blocks;
    UINT8 st_attrib; /* file attribute byte (dosFs only) */
    int    reserved1; /* reserved for future use */
    int    reserved2; /* reserved for future use */
}
```
## File Control API

### mpStat

```c
int reserved3; /* reserved for future use */
int reserved4; /* reserved for future use */
int reserved5; /* reserved for future use */
int reserved6; /* reserved for future use */
};
```

/* File mode (st_mode) bit masks */
#define S_IFMT 0xf000 /* file type field */
#define S_IFIFO 0x1000 /* fifo */
#define S_IFCHR 0x2000 /* character special */
#define S_IFDIR 0x4000 /* directory */
#define S_IFBLK 0x6000 /* block special */
#define S_IFREG 0x8000 /* regular */
#define S_IFLNK 0xa000 /* symbolic link */
#define S_IFSOCK 0xc000 /* socket */

### Return Value

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>-1</td>
<td>Error (Incorrect specified file descriptor)</td>
</tr>
</tbody>
</table>

**NOTE**

FileControl API is executed only with the priority MP_PRI_TIME_NORMAL. When a task with a higher priority than MP_PRI_TIME_NORMAL calls FileControl API, and the task priority limitation is set to "Automatic Adjustment" (S2C1101 = 1), the system temporarily lowers the priority to MP_PRI_TIME_NORMAL, and executes the function. If the task priority limitation is set to "limitation" (S2C1101 = 0), the API returns an error.
mpOpendir

Opens the specified directory to examine the entry in the directory.

- **Syntax**

```c
DIR * mpOpendir (char * dirName); /* name of directory to open */
```

```c
#define NAME_MAX 99 /* The file name length excluding the path */
struct dirent { /* dirent */
    char d_name [NAME_MAX + 1]; /* file name, null-terminated */
};
```

```c
typedef struct /* DIR */
{ /* DIR */
    int dd_fd; /* file descriptor for open directory */
    int dd_cookie; /* filesys-specific marker within dir */
    struct dirent dd_dirent; /* obtained directory entry */
} DIR;
```

- **Parameter**

  `[dirName]`

  Directory name

- **Return Value**

  - >0 Normal end (Directory descriptor address)
  - 0 Error (no specified directory)

---

**NOTE**

- FileControl API is executed only with the priority MP_PRI_TIME_NORMAL. When a task with a higher priority than MP_PRI_TIME_NORMAL calls FileControl API, and the task priority limitation is set to "Automatic Adjustment" (S2C1101 = 1), the system temporarily lowers the priority to MP_PRI_TIME_NORMAL, and executes the function. If the task priority limitation is set to "limitation" (S2C1101 = 0), the API returns an error.
- After opening a directory by mpOpendir, be sure to close it by mpClosedir when it is no longer needed.
mpReaddir

Reads one entry from the opened directory.

**Syntax**

```c
struct dirent * mpReaddir /* return : A pointer to a dirent structure, */
    ( /* or NULL if there is an end-of-directory */
      /* marker or error */
        DIR * pDir /* pointer to directory descriptor */
    );
```

```c
#define NAME_MAX 99 /* The file name length excluding the path */
struct dirent /* dirent */
    { char d_name [NAME_MAX + 1]; /* file name, null-terminated */
    }

typedef struct /* DIR */
    { int dd_fd; /* file descriptor for open directory */
      int dd_cookie; /* filesys-specific marker within dir */
      struct dirent dd_dirent; /* obtained directory entry */
    } DIR;
```

**Parameter**

[rDir]

Directory descriptor

**Return Value**

> 0 Normal end (The address of the read directory entry)

0 Error (Incorrect specified file descriptor)

---

**NOTE**

FileControl API is executed only with the priority MP_PRI_TIME_NORMAL. When a task with a higher priority than MP_PRI_TIME_NORMAL calls FileControl API, and the task priority limitation is set to "Automatic Adjustment" (S2C1101 = 1), the system temporarily lowers the priority to MP_PRI_TIME_NORMAL, and executes the function. If the task priority limitation is set to "limitation" (S2C1101 = 0), the API returns an error.
mpRewinddir

Resets the directory descriptor, and returns the read position to the top of the directory.

- **Syntax**

  ```c
  void mpRewinddir /* return : N/A */
  (    
   DIR * pDir /* pointer to directory descriptor */
  );
  ```

  ```c
  #define NAME_MAX 99 /* The file name length excluding the path */
  struct dirent /* dirent */
  {
   char    d_name [NAME_MAX + 1]; /* file name, null-terminated */
  };
  ```

  ```c
  typedef struct  /* DIR */
  {    
   int     dd_fd; /* file descriptor for open directory */
   int     dd_cookie; /* filesys-specific marker within dir */
   struct dirent dd_dirent; /* obtained directory entry */
  } DIR;
  ```

- **Parameter**

  **[rDir]**

  Directory descriptor

  **NOTE**

  FileControl API is executed only with the priority MP_PRI_TIME_NORMAL. When a task with a higher priority than MP_PRI_TIME_NORMAL calls FileControl API, and the task priority limitation is set to "Automatic Adjustment" (S2C1101 = 1), the system temporarily lowers the priority to MP_PRI_TIME_NORMAL, and executes the function. If the task priority limitation is set to "limitation" (S2C1101 = 0), the API returns an error.
**mpClosedir**

Closes the directory descriptor.

- **Syntax**

  ```c
  STATUS mpClosedir /* return : OK or ERROR by closedir() */
  (DIR * pDir /* pointer to directory descriptor */);
  ```

- **Parameter**

  - **pDir**
    Directory descriptor

- **Return Value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>-1</td>
<td>Error (Incorrect specified directory descriptor)</td>
</tr>
</tbody>
</table>

**NOTE**

- FileControl API is executed only with the priority MP_PRI_TIME_NORMAL. When a task with a higher priority than MP_PRI_TIME_NORMAL calls FileControl API, and the task priority limitation is set to "Automatic Adjustment" (S2C1101 = 1), the system temporarily lowers the priority to MP_PRI_TIME_NORMAL, and executes the function. If the task priority limitation is set to "limitation" (S2C1101 = 0), the API returns an error.

- After opening a directory by mpOpendir, be sure to close it by mpClosedir when it is no longer needed.
mpLoadFile

writes file data in the specified RAM drive to the controller.

- **Syntax**
  ```c
  LONG mpLoadFile /* return : status */
  (  
    LONG            mpRamDriveId, /* ram drive id to read from */
    CONST CHAR *loadPath, /* file path name */
    CONST CHAR *fileName /* file name */
  );
  ```

- **Parameter**
  - **[mpRamDriveId]**
    The ID of the RAM drive that reads the written file data
    ```c
    #define MP_DRV_ID_DRAM 1
    #define MP_DRV_ID_SRAM 2
    ```
  - **[loadPath]**
    The character string of the folder name where the written file exists
    - When the file exists in the route folder of the specified media, specify an empty character string (" ").
  - **[fileName]**
    The character string of the written file name
    In the case with a job file, an existing job cannot be overwritten. ("2040 Defined JOB name" returns as the return value.)
    Delete the job by using mpDeleteJob, etc., then use this API.
8 Existing File Access API

mpLoadFile

■ Return Value

0 Normal end

One of the following values  Error

<table>
<thead>
<tr>
<th>Value (hexadecimal)</th>
<th>Value (decimal)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1086 or less</td>
<td>4230 or less</td>
<td>Error related to the file For details, refer to the error numbers (decimal) in the chapter 9 Error, 9.1 Error Message of “FS100 MAINTENANCE MANUAL (RE-CHO-A111).”</td>
</tr>
<tr>
<td>0x2010</td>
<td>8208</td>
<td>The manipulator is in motion</td>
</tr>
<tr>
<td>0x2060</td>
<td>8288</td>
<td>Alarm or error status</td>
</tr>
<tr>
<td>0x2080</td>
<td>8320</td>
<td>Wrong operation mode</td>
</tr>
<tr>
<td>0x2090</td>
<td>8336</td>
<td>Data being accessed by another function Retry later.</td>
</tr>
<tr>
<td>0xffff</td>
<td>65535</td>
<td>Other than the above</td>
</tr>
</tbody>
</table>

• FileControl API is executed only with the priority MP_PRI_TIME_NORMAL. When a task with a higher priority than MP_PRI_TIME_NORMAL calls FileControl API, and the task priority limitation is set to "Automatic Adjustment" (S2C1101 = 1), the system temporarily lowers the priority to MP_PRI_TIME_NORMAL, and executes the function. If the task priority limitation is set to "limitation" (S2C1101 = 0), the API returns an error.

• Calling mpLoadFile, mpSaveFile, mpFdWriteFile, or mpFdReadFile to load or save an existing file prohibits editing operation to prevent conflicts in the system’s internal information. Thus, the editing mode of the JOB, variable, or I/O is initialized, and the programming pendant cannot be temporarily operated. Therefore, so as not to interfere with editing operation by using the programming pendant, use these APIs in the teach mode only when necessary, and do not load or save successively.
mpSaveFile

Saves the specified file from the controller to the RAM drive.

■ Syntax

```c
LONG mpSaveFile /* return : status */
{
    LONG mpRamDriveId, /* ram drive id to write to */
    CONST CHAR *savePath, /* file path name */
    CONST CHAR *fileName /* file name */
};
```

■ Parameter

- **[mpRamDriveId]**
The ID of the RAM drive that stores the read file data

```c
#define MP_DRV_ID_DRAM 1
#define MP_DRV_ID_SRAM 2
```

- **[savePath]**
The character string of the folder name where the saved file is stored
  - When the file exists in the route folder of the specified media, specify an empty character string (" ").

- **[fileName]**
The character string of the saved file name

If an existing file on the RAM drive is read, the file is overwritten.

■ Return Value

0 Normal end

One of the following values Error

<table>
<thead>
<tr>
<th>Value (hexadecimal)</th>
<th>Value (decimal)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1086 or less</td>
<td>4230 or less</td>
<td>Error related to the file Error Message of &quot;FS100 MAINTENANCE MANUAL (RE-CHO-A111)&quot;.</td>
</tr>
<tr>
<td>0x2060</td>
<td>8288</td>
<td>Alarm or error status</td>
</tr>
<tr>
<td>0x2090</td>
<td>8336</td>
<td>Data being accessed by another function Retry later.</td>
</tr>
<tr>
<td>0xffff</td>
<td>65535</td>
<td>Other than the above</td>
</tr>
</tbody>
</table>
Existing File Access API

mpSaveFile

- FileControl API is executed only with the priority MP_PRI_TIME_NORMAL. When a task with a higher priority than MP_PRI_TIME_NORMAL calls FileControl API, and the task priority limitation is set to “Automatic Adjustment” (S2C1101 = 1), the system temporarily lowers the priority to MP_PRI_TIME_NORMAL, and executes the function. If the task priority limitation is set to “limitation” (S2C1101 = 0), the API returns an error.

- Calling mpLoadFile, mpSaveFile, mpFdWriteFile, or mpFdReadFile to load or save an existing file prohibits editing operation to prevent conflicts in the system’s internal information. Thus, the editing mode of the JOB, variable, or I/O is initialized, and the programming pendant cannot be temporarily operated. Therefore, so as not to interfere with editing operation by using the programming pendant, use these APIs in the teach mode only when necessary, and do not load or save successively.

NOTE
mpRefreshFileList

Updates a file list.

- **Syntax**
  
  ```c
  LONG mpRefreshFileList /* return : status */
  {
    SHORT extensionId /* file extension index */
  };
  ```

- **Description**
  
  Make sure to execute this API before executing mpGetFileCount() or mpGetFileName.

- **Parameter**
  
  `[extensionId]`  
  File type ID
  
  ```
  #define MP_EXT_ID_JBI 1 Independent job file (JBI)
  #define MP_EXT_ID_JBR 2 Related job file (JBR)
  ```

- **Return Value**
  
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>-1</td>
<td>Error</td>
</tr>
</tbody>
</table>

*NOTE*

FileControl API is executed only with the priority MP_PRI_TIME_NORMAL. When a task with a higher priority than MP_PRI_TIME_NORMAL calls FileControl API, and the task priority limitation is set to “Automatic Adjustment” (S2C1101 = 1), the system temporarily lowers the priority to MP_PRI_TIME_NORMAL, and executes the function. If the task priority limitation is set to “limitation” (S2C1101 = 0), the API returns an error.
mpGetFileCount

Retrieves a number of files of the file list.

- **Syntax**

```c
LONG mpGetFileCount( /* return : status */
        void);
```

- **Description**

  This API returns the number of files in the file list retrieved by `mpRefreshFileList()`.
  Thus, to retrieve the latest information, make sure to execute `mpRefreshFileList()` before executing this API.

- **Parameter**

  None

- **Return Value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;=0</td>
<td>Number of files</td>
</tr>
<tr>
<td>-1</td>
<td>Error</td>
</tr>
</tbody>
</table>

---

**NOTE**

FileControl API is executed only with the priority `MP_PRI_TIME_NORMAL`. When a task with a higher priority than `MP_PRI_TIME_NORMAL` calls FileControl API, and the task priority limitation is set to “Automatic Adjustment” (`S2C1101 = 1`), the system temporarily lowers the priority to `MP_PRI_TIME_NORMAL`, and executes the function. If the task priority limitation is set to “limitation” (`S2C1101 = 0`), the API returns an error.
mpGetFileName

Retrieves a file name from the file list.

- **Syntax**

  ```c
  LONG  mpGetFileName  /* return : status */
  (  
   LONG  index,   /* index of file list */
   CHAR  *fileName    /* pointer to buffer to write the file name */
  );
  ```

- **Description**

  This API retrieves the file name specified by `index` from the file list retrieved by `mpRefreshFileList()`. Thus, to retrieve the latest information, make sure to execute `mpRefreshFileList()` before executing this API.

- **Parameter**

  - `[index]`
    The index in the list of the retrieved file name
  - `[fileName]`
    The pointer to the buffer where the saved file name is output

- **Return Value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>-1</td>
<td>Error</td>
</tr>
</tbody>
</table>

---

**FileControl API is executed only with the priority MP_PRI_TIME_NORMAL. When a task with a higher priority than MP_PRI_TIME_NORMAL calls FileControl API, and the task priority limitation is set to "Automatic Adjustment" (S2C1101 = 1), the system temporarily lowers the priority to MP_PRI_TIME_NORMAL, and executes the function. If the task priority limitation is set to "limitation" (S2C1101 = 0), the API returns an error.**
mpFdWriteFile

Writes file data in the specified fd to the controller.

- **Syntax**
  ```c
  int mpFdWriteFile /* return : status */
  (int fd, /* file descriptor to read from */
   const MP_FILE_NAME_SEND_DATA* sendData /* pointer to send file information */);
  ```

- **Parameter**
  - **[fd]**
    The file descriptor that reads the written file data
    Use the value that is retrieved in advance with mpOpen() of FileControl API.
  - **[sendData]**
    The pointer to the structure that stores the written file information
    (The following members are provided.)
    ```c
    cFileName[TRANS_FILE_LEN + 1]
    ```
    The array storing the character string of the written file
    The file name is created when it is saved with the external memory function (name.JBI, etc.)
    ```c
    #define TRANS_FILE_LEN (8 + 1 + 3)
    ```

- **Return Value**
  - **0** Normal end
  - One of the following values Error

<table>
<thead>
<tr>
<th>Value (hexadecimal)</th>
<th>Value (decimal)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1086 or less</td>
<td>4230 or less</td>
<td>Error related to the file Error related to the file For details, refer to the error numbers (decimal) in the chapter 9 Error, 9.1 Error Message of &quot;FS100 MAINTENANCE MANUAL (RE-CHO-A111)&quot;.</td>
</tr>
<tr>
<td>0x2010</td>
<td>8208</td>
<td>The manipulator is in motion</td>
</tr>
<tr>
<td>0x2060</td>
<td>8288</td>
<td>Alarm or error status</td>
</tr>
<tr>
<td>0x2080</td>
<td>8320</td>
<td>Wrong operation mode</td>
</tr>
<tr>
<td>0x2090</td>
<td>8336</td>
<td>Data being accessed by another function Retry later.</td>
</tr>
<tr>
<td>0xffff</td>
<td>65535</td>
<td>Other than the above</td>
</tr>
</tbody>
</table>
• FileControl API is executed only with the priority MP_PRI_TIME_NORMAL. When a task with a higher priority than MP_PRI_TIME_NORMAL calls FileControl API, and the task priority limitation is set to "Automatic Adjustment" (S2C1101 = 1), the system temporarily lowers the priority to MP_PRI_TIME_NORMAL, and executes the function. If the task priority limitation is set to "limitation" (S2C1101 = 0), the API returns an error.

• Calling mpLoadFile, mpSaveFile, mpFdWriteFile, or mpFdReadFile to load or save an existing file prohibits editing operation to prevent conflicts in the system’s internal information. Thus, the editing mode of the JOB, variable, or I/O is initialized, and the programming pendant cannot be temporarily operated. Therefore, so as not to interfere with editing operation by using the programming pendant, use these APIs in the teach mode only when necessary, and do not load or save successively.
mpFdReadFile

Reads file data from the controller to the specified fd.

- **Syntax**

```c
int mpFdWriteFile /* return : status */
(int  fd, /* file descriptor to read from */
const MP_FILE_NAME_SEND_DATA *sendData, /* pointer to send file information */
```

- **Parameter**

- **[fd]**
The file descriptor that writes the read file data

  Use the value that is retrieved in advance with mpOpen() or mpCreate() of FileControl API.

- **[sendData]**
The pointer to the structure that stores the read file information

  (The following members are provided.)

  cFileName[TRANS_FILE_LEN + 1]

  The array storing the character string of the read file

  The file name is created when it is saved with the external memory function (name.JBI, etc.)

- **Return Value**

<table>
<thead>
<tr>
<th>Value (hexadecimal)</th>
<th>Value (decimal)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1086 or less</td>
<td>4230 or less</td>
<td>Error related to the file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For details, refer to the error numbers (decimal) in the chapter 9 Error, 9.1 Error Message of &quot;FS100 MAINTENANCE MANUAL (RE-CHO-A111)&quot;.</td>
</tr>
<tr>
<td>0x2060</td>
<td>8288</td>
<td>Alarm or error status</td>
</tr>
<tr>
<td>0x2090</td>
<td>8336</td>
<td>Data being accessed by another function</td>
</tr>
<tr>
<td>0xffff</td>
<td>65535</td>
<td>Other than the above</td>
</tr>
</tbody>
</table>

  Retry later.
• FileControl API is executed only with the priority MP_PRI_TIME_NORMAL. When a task with a higher priority than MP_PRI_TIME_NORMAL calls FileControl API, and the task priority limitation is set to "Automatic Adjustment" (S2C1101 = 1), the system temporarily lowers the priority to MP_PRI_TIME_NORMAL, and executes the function. If the task priority limitation is set to "limitation" (S2C1101 = 0), the API returns an error.

• Calling mpLoadFile, mpSaveFile, mpFdWriteFile, or mpFdReadFile to load or save an existing file prohibits editing operation to prevent conflicts in the system's internal information. Thus, the editing mode of the JOB, variable, or I/O is initialized, and the programming pendant cannot be temporarily operated. Therefore, so as not to interfere with editing operation by using the programming pendant, use these APIs in the teach mode only when necessary, and do not load or save successively.
mpFdGetJobList

Retrieves a job list to the specified fd.

- Syntax

```c
int mpFdGetJobList /* return : status */
(int fd, /* file descriptor to write to */
MP_GET_JOBLIST_RSP_DATA *responseData /* pointer to response data */
)
```

- Parameter

[fd]
The file descriptor that writes the job list
Use the value that is retrieved in advance with mpOpen() or mpCreate() of
FileControl API.

[responseData]
The pointer to the structure that stores the read result
(The following members are provided.)

typedef struct
{
USHORT err_no Error number
USHORT ulsEndFlag Always 1 with the normal end
USHORT uListDataNum The number of the read job names
UCHAR cListData[] The work area used for reading
CHAR reserved[2]
} MP_GET_JOBLIST_RSP_DATA

- Return Value

0 Normal end
-1 Error (Usually, it does not occur)

- Note

- Job list format
  - Two-dimensional array with 32 characters +1 (NULL character is filled
to the job name trail.)

- FileControl API is executed only with the priority
  MP_PRI_TIME_NORMAL. When a task with a higher
  priority than MP_PRI_TIME_NORMAL calls FileControl
  API, and the task priority limitation is set to "Automatic
  Adjustment" (S2C1101 = 1), the system temporarily lowers
  the priority to MP_PRI_TIME_NORMAL, and executes the
  function. If the task priority limitation is set to "limitation"
  (S2C1101 = 0), the API returns an error.
System Monitor API

mpGetVarData

Retrieves variables B, I, D, R (byte, integer, double precision, real). Specifies multiple variables.

**Syntax**

```c
LONG  mpGetVarData ( 
    MP_VAR_INFO*  sData, 
    LONG*         rData, 
    LONG          num 
);
```

**Parameter**

**[sData]**

Pointer to the data structure which specifies variable

- **MP_VAR_INFO**
  
  Variable setting data structure
  
  Syntax: `
  typedef struct{
      USHORT  usType; 
      USHORT  usIndex; 
  } MP_VAR_INFO;
  `

  Member: `<usType>`
  
  Variable type

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_RESTYPE_VAR_B</td>
<td>Byte type</td>
</tr>
<tr>
<td>MP_RESTYPE_VAR_I</td>
<td>Integer type</td>
</tr>
<tr>
<td>MP_RESTYPE_VAR_D</td>
<td>Double-precision type</td>
</tr>
<tr>
<td>MP_RESTYPE_VAR_R</td>
<td>Real type</td>
</tr>
</tbody>
</table>

  `<usIndex>`
  
  Variable index

To set multiple variables, define `num` arrays, set the variable type and variable index of each array, then set the starting address of each array to `sData`.

**NOTE**

If multiple variable data are retrieved by this API, these data are not synchronized. Thus, while retrieving multiple values, these may be changed by another application task or system task.

**[rData]**

Pointer to the variable data

**[num]**

Number of variable data (up to 252)
System Monitor API

mpGetVarData

- **Return Value**
  - 0  Normal end
  - -1  Error
mpGetSVarInfo

Reads S variable (fixed length). Specifies multiple variables.

**NOTE**
The system software version FS1.03-00 and IDE version 1.10 or later versions support this API.

**Syntax**

```
LONG mpGetSVarInfo
(
    MP_VAR_INFO  *sData,
    MP_SVAR_RECV_INFO *rData,
    LONG num
)
```

**Parameter**

**[sData]**
Pointer to the data structure which specifies variable
typedef struct
{
    USHORT usType; /* Variable type (only MP_RESTYPE_VAR_S is available) */
    USHORT usIndex; /* Variable index */
} MP_VAR_INFO;
To set multiple variables, define `num` arrays, set the variable type and variable index of each array, and set the starting address of each array to `sData`.

**[rData]**
Pointer to the storage area for variable data
typedef struct
{
    UCHAR ucValue[S_VAR_SIZE+1]; /*S_VAR_SIZE (16 characters) + delimiter\0*/
    UCHAR reserved[3];
} MP_SVAR_RECV_INFO;
To set multiple variables, define `num` arrays.

**[num]**
Number of S variables (up to 10)

**Return Value**

- 0 Normal end
- -1 Error
• Since this API’s processing time is a few milliseconds, do not use this API in a task with the I/O control cycle or interpolation control cycle. If it is used in such a task, the process cannot be completed within the control cycle.

• If multiple S variables are retrieved by this API, these data are not synchronized. Thus, while retrieving multiple values, these may be changed by another application task or system task.

• When changing a parameter or loading a file from the external memory device, the process of this API cannot be executed because the system is busy. In this case, the error (-1) is returned. Retry this API after a while.
mpReadIO

Retrieves several I/O values at a time.

- **Syntax**
  ```c
  LONG  mpReadIO ( 
        MP_IO_INFO*  sData, 
        USHORT*  rData, 
        LONG  num 
    );
  ```

- **Parameter**

  - **sData**
    Data structure that specifies I/O address
    - **MP_IO_INFO**
      Data structure that specifies I/O address
      - **Syntax:**
        ```c
        typedef struct 
        { 
            ULONG  ulAddr; 
        } MP_IO_INFO;
        ```
      - **Member:** `<ulAddr>`
        I/O address

      | Value     | Description                              |
      |-----------|------------------------------------------|
      | 10 - 1287 | Universal input #00010 - #01287(1024)    |
      | 10010 - 11287 | Universal output #10010 - #11287(1024) |
      | 20010 - 1287 | External input #20010 - #21287(1024)    |
      | 30010 - 31287 | External output #30010 - #31287(1024)  |
      | 40010 - 41607 | Specific input #40010 - #41607(1280)   |
      | 50010 - 52007 | Specific output #50010 - #52007(1600) |
      | 60010 - 60647 | Interface panel #60010 - #60647(512)   |
      | 70010 - 79997 | Auxiliary relay #70010 - #79997(7992)  |
      | 80010 - 80647 | Control input #80010 - #80647(512)     |
      | 82010 - 82207 | Pseudo input #82010 - #82207(160)      |
      | 25010 - 26287 | Network input #25010 - #26287(1024)    |
      | 35010 - 36287 | Network output #35010 - #36287(1024)  |
      | 1000000 - 1000559 | Register #1000000 - #1000559(560) |

  To set multiple I/Os, define `num` arrays, set the I/O address of each array, then set the starting address of each array to `sData`.

- **NOTE**
  When retrieving multiple values simultaneously, this API temporarily makes the task priority of the application higher than the robot motion task, etc. so that the values will not be changed by another task while retrieving the values. Thus, reading many values may affect the robot motion. **DO NOT specify a large value as `num` during the robot motion.**
**mpReadIO**

**[rData]**
I/O value

**[num]**
Number of I/O (up to 253)

- **Return Value**
  - 0  Normal end
  - -1  Error
mpMonitor

Retrieves a variable and an I/O value at a time.

- **Syntax**

  ```c
  LONG  mpMonitor ( 
      MP_MONITOR_INFO*  sData, 
      LONG*              rData, 
      LONG               infonum
  );
  ```

- **Parameter**

  **[sData]**
  
  Pointer to the monitor information structure

  - **MP_MONITOR_INFO**
    
    Monitor information structure
    
    Syntax:  `typedef struct 
    { 
      ULONG  ulType; 
      ULONG  ulIndex; 
    } MP_MONITOR_INFO;`
    
    Member:  `<ulType>`
    
    Variable, I/O type

  - `<ulIndex>`
    
    Variable number, I/O address

  To set multiple variables and I/Os, define `infonum` arrays, set the variable type and variable index of each array, then set the starting address of each array to `sData`.

- **[rData]**

  Monitor information value

- **[infonum]**

  Number of monitor (up to 126)

---

**NOTE**

When retrieving multiple values simultaneously, this API temporarily makes the task priority of the application higher than the robot motion task, etc. so that the values will not be changed by another task while retrieving the values. Thus, reading many values may affect the robot motion. DO NOT specify a large value as `num` during the robot motion.
### Return Value

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>-1</td>
<td>Error</td>
</tr>
</tbody>
</table>
mpGetPosVarData

Retrieves the position-type variable.

**Syntax**

```c
LONG   mpGetPosVarData ( 
MP_VAR_INFO* sData, 
LONG* rData, 
LONG num 
);
```

**Parameter**

*sData*  
[in] The pointer to the data structure array that sets the position-type variable

- **MP_VAR_INFO**
  
  [in] The data structure that sets the position-type variable
  
  Syntax:
  ```c
typedef struct
  {
      USHORT usType;
      USHORT usIndex;
  } MP_VAR_INFO;
  ```
  
  **Member:** *<usType>*
  
  Variable type
  
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_RESTYPE_VAR_ROBOT</td>
<td>Robot</td>
</tr>
<tr>
<td>MP_RESTYPE_VAR_BASE</td>
<td>Base</td>
</tr>
<tr>
<td>MP_RESTYPE_VAR_STATION</td>
<td>Station</td>
</tr>
</tbody>
</table>

- **<usIndex>**
  
  Variable number

To set multiple position-type variables, define `num` arrays, set the variable type and variable index of each array, then set the starting address of each array to `sData`.

**NOTE**

- If multiple position-type variables are retrieved by this API, these data are not synchronized. Thus, while retrieving multiple values, these may be changed by another application task or system task.

- Since this API’s processing time is a few milliseconds, do not use this API in a task with the I/O control cycle or interpolation control cycle. If it is used in such a task, the process cannot be completed within the control cycle.
System Monitor API
mpGetPosVarData

[rData]
[out] The pointer to the position-type variable information (requires area for 10*num)

<table>
<thead>
<tr>
<th>Array</th>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rData[0]</td>
<td>D05 - D00</td>
<td>Variable type</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>Pulse</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>Cartesian (base coordinates)</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>Cartesian (robot coordinates)</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>Cartesian (tool coordinates)</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>Cartesian (user coordinates)</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>Cartesian (reserved for master tool)</td>
</tr>
<tr>
<td>D07 - D06</td>
<td></td>
<td>Reserved by manufacturer</td>
</tr>
<tr>
<td>D08</td>
<td>0: Front</td>
<td>1: Back</td>
</tr>
<tr>
<td>D09</td>
<td>0: Upper arm</td>
<td>1: Lower arm</td>
</tr>
<tr>
<td>D10</td>
<td>0: Flip</td>
<td>1: No flip</td>
</tr>
<tr>
<td>D11</td>
<td>0: R &lt; 180 deg</td>
<td>1: R &gt;= 180 deg</td>
</tr>
<tr>
<td>D12</td>
<td>0: T &lt; 180 deg</td>
<td>1: T &gt;= 180 deg</td>
</tr>
<tr>
<td>D13</td>
<td>0: S &lt; 180 deg</td>
<td>1: S &gt;= 180 deg</td>
</tr>
<tr>
<td>D14 - D15</td>
<td></td>
<td>Reserved by manufacturer</td>
</tr>
<tr>
<td>D16 - D21</td>
<td></td>
<td>Tool number (0 - 23)</td>
</tr>
<tr>
<td>D22 - D27</td>
<td></td>
<td>User coordinate number</td>
</tr>
<tr>
<td>D28 - D31</td>
<td></td>
<td>Reserved by manufacturer</td>
</tr>
<tr>
<td>rData[1]</td>
<td></td>
<td>(Extended attribute)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Array</th>
<th>Pulse</th>
<th>Cartesian</th>
</tr>
</thead>
<tbody>
<tr>
<td>rData[2]</td>
<td>1st axis (S) pulse value</td>
<td>X-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>rData[3]</td>
<td>2nd axis (L) pulse value</td>
<td>Y-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>rData[4]</td>
<td>3rd axis (U) pulse value</td>
<td>Z-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>rData[5]</td>
<td>4th axis (R) pulse value</td>
<td>Wrist angle Rx (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>rData[6]</td>
<td>5th axis (B) pulse value</td>
<td>Wrist angle Ry (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>rData[7]</td>
<td>6th axis (T) pulse value</td>
<td>Wrist angle Rz (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>rData[8]</td>
<td>7th axis (E) pulse value</td>
<td>Angle Re (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>rData[9]</td>
<td>8th axis pulse value</td>
<td>8th axis pulse value (micron in the case of traveling axis)</td>
</tr>
</tbody>
</table>

[num]
Number of array (up to 25)

- Return Value
  - 0 Normal end
  - -1 Error
mpGetUserVars

Retrieves the user variable.

To compare mpGetUserVars with the other variable access APIs, such as mpGetVarData(), mpGetPosVarData() and mpGetSVarInfo(), mpGetUserVars can retrieves user variable in high speed.

**Syntax**

```c
STATUS mpGetUserVars(
    MP_USR_VAR_INFO*mp_usr_var_info
);
```

**Parameter**

- **mp_usr_var_info**
  
  [in] User variable access API interface (Input/Output)

- **MP_USR_VAR_INFO**
  
  [in] The data structure that sets the user variable

  Syntax:

  ```c
typedef struct {
    int var_Type;
    int var_no;
    union {
      MP_B_VAR_BUFF b;
      MP_I_VAR_BUFF i;
      MP_D_VAR_BUFF d;
      MP_R_VAR_BUFF r;
      MP_S_VAR_BUFF s;
      MP_P_VAR_BUFF p;
      MP_P_VAR_BUFF bp;
      MP_P_VAR_BUFF ex;
    } val;
  } MP_USR_VAR_INFO;
```

*Member:* `<var_Type>`

Variable type (Input)

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_VAR_B</td>
<td>Byte Variable</td>
</tr>
<tr>
<td>MP_VAR_I</td>
<td>Integer Variable</td>
</tr>
<tr>
<td>MP_VAR_D</td>
<td>Double Variable</td>
</tr>
<tr>
<td>MP_VAR_R</td>
<td>Real Variable</td>
</tr>
<tr>
<td>MP_VAR_S</td>
<td>Character Variable</td>
</tr>
<tr>
<td>MP_VAR_P</td>
<td>Robot position Variable</td>
</tr>
<tr>
<td>MP_VAR_BP</td>
<td>Base position Variable</td>
</tr>
<tr>
<td>MP_VAR_EX</td>
<td>Station (LEX) Variable</td>
</tr>
</tbody>
</table>

*<var_no>*

Variable number (Input)
 Variable data buffer (output)

Syntax:
```
define STR_VAR_SIZE 16
#define MP_GRP_AXES_NUM 8
```

```c
typedef unsigned char MP_B_VAR_BUFF;
typedef short MP_I_VAR_BUFF;
typedef long MP_D_VAR_BUFF;
typedef float MP_R_VAR_BUFF;
typedef char MP_S_VAR_BUFF[STR_VAR_SIZE];
```

```c
typedef struct {
    UNIT dtype;
    int tool_no;
    int uf_no;
    BITSTRING fig_ctrl;
    long data[MP_GRP_AXES_NUM];
} MP_P_VAR_BUFF;
```

Member: **MP_P_VAR_BUFF**

- `<dtype>`
  - Coordinate system type

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_PULSE_DTYPE</td>
<td>Pulse coordinate</td>
</tr>
<tr>
<td>MP_BASE_DTYPE</td>
<td>Base coordinate</td>
</tr>
<tr>
<td>MP_ROBO_DTYPE</td>
<td>Robot coordinate</td>
</tr>
<tr>
<td>MP_TOOL_DTYPE</td>
<td>Tool coordinate</td>
</tr>
<tr>
<td>MP_USER_DTYPE</td>
<td>User coordinate</td>
</tr>
<tr>
<td>MP_MTOOL_DTYPE</td>
<td>Master tool coordinate</td>
</tr>
</tbody>
</table>

- `<tool_no>`
  - Tool file No. (0 to 15)

- `<uf_no>`
  - User coordinate system file No. (0 to 15)

- `<fig_ctrl>`
  - Figure information (bits)

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_FIG_SIDE</td>
<td>0: Front</td>
</tr>
<tr>
<td></td>
<td>1: Back</td>
</tr>
<tr>
<td>MP_FIG_ELBOW</td>
<td>0: Upper elbow</td>
</tr>
<tr>
<td></td>
<td>1: Lower elbow</td>
</tr>
<tr>
<td>MP_FIG_FLIP</td>
<td>0: Flip</td>
</tr>
<tr>
<td></td>
<td>1: No flip</td>
</tr>
<tr>
<td>MP_FIG_R180</td>
<td>0: R&lt;180</td>
</tr>
<tr>
<td></td>
<td>1: R&gt;=180</td>
</tr>
<tr>
<td>MP_FIG_T180</td>
<td>0: T&lt;180</td>
</tr>
<tr>
<td></td>
<td>1: T&gt;=180</td>
</tr>
<tr>
<td>MP_FIG_S180</td>
<td>0: S&lt;180</td>
</tr>
<tr>
<td></td>
<td>1: S&gt;=180</td>
</tr>
</tbody>
</table>

- `<data>`
  - Position data
Return value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_MP_VARS_ACCESS</td>
<td>An error except followings</td>
</tr>
<tr>
<td>E_MP_VAR_TYPE</td>
<td>Incorrect variable type</td>
</tr>
<tr>
<td>E_MP_VAR_NO_RANGE</td>
<td>Incorrect variable No.</td>
</tr>
<tr>
<td>E_MP_PVAR_DTYPE</td>
<td>Incorrect coordinate type of the position variable</td>
</tr>
<tr>
<td>E_MP_PVAR_TOOL_NO</td>
<td>Incorrect tool No. of the position variable</td>
</tr>
<tr>
<td>E_MP_PVAR_USER_NO</td>
<td>Incorrect user coordinate type of the position variable</td>
</tr>
<tr>
<td>E_MP_PVAR_UNSETUP</td>
<td>Undefined data of the position variable</td>
</tr>
<tr>
<td>E_MP_PMS_RET_VAL</td>
<td>Failure to retrieve the position variable</td>
</tr>
</tbody>
</table>
Data types which are used in this API are different from ones in the mpGetVarData(), mpGetPosVarData() and mpGetSVarInfo(). The following table shows the differences of the position variable.

**Difference of the API Parameter**

<table>
<thead>
<tr>
<th>Position Variable Retrieved API</th>
<th>User Variable Retrieved API</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpGetPosVarData(</td>
<td>mpGetUserVars(</td>
</tr>
<tr>
<td>MP_VAR_INFO *sData;</td>
<td>MP_USR_VAR_INFO *mp_usr_var_info;</td>
</tr>
<tr>
<td>LONG *rData;</td>
<td>)</td>
</tr>
<tr>
<td>LONG num;</td>
<td>Syntax</td>
</tr>
<tr>
<td>)</td>
<td>typedef struct {</td>
</tr>
<tr>
<td></td>
<td>int var_type;</td>
</tr>
<tr>
<td></td>
<td>int var_no;</td>
</tr>
<tr>
<td></td>
<td>union {</td>
</tr>
<tr>
<td></td>
<td>MP_B_VAR_BUFF b;</td>
</tr>
<tr>
<td></td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>MP_P_VAR_BUFF p;</td>
</tr>
<tr>
<td></td>
<td>MP_P_VAR_BUFF bp;</td>
</tr>
<tr>
<td></td>
<td>MP_P_VAR_BUFF ex;</td>
</tr>
<tr>
<td></td>
<td>} val;</td>
</tr>
<tr>
<td></td>
<td>} MP_USR_VAR_INFO;</td>
</tr>
</tbody>
</table>

**Setting Content of the Parameter**

<table>
<thead>
<tr>
<th>Item</th>
<th>mpGetPosVarData()</th>
<th>mpGetUserVars()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordinate system</td>
<td>rData[0]</td>
<td>p, bp, ex</td>
</tr>
<tr>
<td>Type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Front/Back</td>
<td>d0 to d5</td>
<td></td>
</tr>
<tr>
<td>Upper elbow/ Lower elbow</td>
<td>d08 .dtype d08</td>
<td></td>
</tr>
<tr>
<td>Flip/No flip</td>
<td>d10</td>
<td>d2</td>
</tr>
<tr>
<td>R&lt;180/R&gt;=180</td>
<td>d11</td>
<td>d3</td>
</tr>
<tr>
<td>T&lt;180/T&gt;=180</td>
<td>d12</td>
<td>d4</td>
</tr>
<tr>
<td>S&lt;180/S&gt;=180</td>
<td>d13</td>
<td>d5</td>
</tr>
<tr>
<td>Tool No.</td>
<td>d16 to d21</td>
<td>.tool_no</td>
</tr>
<tr>
<td>User coordinate No.</td>
<td>d22 to d27</td>
<td>.uf_no</td>
</tr>
</tbody>
</table>
mpGetAlarmStatus

Retrieves the alarm status of system.

- **Syntax**

  
  ```c
  LONG mpGetAlarmStatus(
      MP_ALARM_STATUS_RSP_DATA* rData
  );
  ```

- **Parameter**

  - **[rData]**
    [out] The pointer to the data structure which receives the alarm status
    
    - **MP_ALARM_STATUS_RSP_DATA**
      [out] The data structure which receives the alarm status
      
      Syntax: 
      ```c
typedef struct {
    SHORT sIsAlarm;
    CHAR reserved[2];
  } MP_ALARM_STATUS_RSP_DATA;
  ```
      
      Member: 
      ```c
<isAlarm>
    Alarm status
  ```

- **Return Value**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D00</td>
<td>Error</td>
</tr>
<tr>
<td>D01</td>
<td>Alarm</td>
</tr>
<tr>
<td>D02 - D15</td>
<td>Reserved by manufacturer</td>
</tr>
</tbody>
</table>

  | 0    | Normal end                      |
  | -1   | Error                           |
mpGetAlarmCode

Retrieves the error code and alarm code of the system.

**Syntax**

```c
LONG   mpGetAlarmCode
       (MP_ALARM_CODE_RSP_DATA*   rData);
```

**Parameter**

`[rData]`

[out] The pointer to the data structure which receives the alarm code

- **MP_ALARM_CODE_RSP_DATA**
  [out] The data structure which receives the alarm code
  Syntax: `typedef struct {
    USHORT  usErrorNo;
    USHORT  usErrorData;
    USHORT  usAlarmNum;
    MP_ALARM_DATA  AlarmData;
  } MP_ALARM_CODE_RSP_DATA;`

  Member:
  - `<usErrorNo>` Error number
  - `<usErrorData>` Error data
  - `<usAlarmNum>` Number of alarm
  - `<AlarmData>` Alarm data

- **MP_ALARM_DATA**
  [out] Alarm data structure
  Syntax: `#define MAX_ALARM_COUNT (4)
           typedef struct {
             USHORT  usAlarmNo[MAX_ALARM_COUNT];
             USHORT  usAlarmData[MAX_ALARM_COUNT];
           } MP_ALARM_DATA;`

  Member:
  - `<usAlarmNo[MAX_ALARM_COUNT]>` Alarm number (up to 4 numbers)
  - `<usAlarmData[MAX_ALARM_COUNT]>` Alarm data (up to 4 data)

**Return Value**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>-1</td>
<td>Error</td>
</tr>
</tbody>
</table>
mpGetMode

Retrieves the operation mode of the system.

- Syntax

LONG  mpGetMode
(MP_MODE_RSP_DATA*   rData);

- Parameter

[rData]
[out] The pointer to the data structure which receives the operation mode
- MP_MODE_RSP_DATA
[out] The data structure which receives the operation mode

Syntax:
typedef struct
{
    SHORT  sMode;
    SHORT  reserved;
} MP_MODE_RSP_DATA;

Member:  <sMode>
        Operation mode

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TEACH</td>
</tr>
<tr>
<td>2</td>
<td>PLAY</td>
</tr>
</tbody>
</table>

- Return Value

0    Normal end
-1   Error
mpGetCycle

Retrieves the cycle mode of the system.

- Syntax
  ```c
  LONG   mpGetCycle(
           MP_CYCLE_RSP_DATA*   rData
  );
  ```

- Parameter
  **[rData]**
  [out] The pointer to the data structure which receives the cycle mode
  - **MP_CYCLE_RSP_DATA**
    [out] The data structure which receives the cycle mode
    Syntax: typedef struct
    ```c
    {
    SHORT  sCycle;
    CHAR   reserved[2];
    } MP_CYCLE_RSP_DATA;
    ```
    Member:  `<sCycle>`
    Cycle
    **Value**  |  **Description**
    --- | ---
    1     |  Step
    2     |  1Cycle
    3     |  Auto

- Return Value
  0     |  Normal end
  -1    |  Error
mpGetServoPower

Retrieves the ON/OFF status of the servo power.

- **Syntax**

  ```
  LONG    mpGetServoPower
          (MP_SERVO_POWER_RSP_DATA*   rData);
  ```

- **Parameter**

  **[rData]**

  [out] The pointer to the data structure which receives the ON/OFF status of servo power

  ```
  typedef struct { 
   SHORT   sServoPower; 
   CHAR     reserved[2]; 
  } MP_SERVO_POWER_RSP_DATA;
  ```

- **Return Value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Servo power OFF</td>
</tr>
<tr>
<td>1</td>
<td>Servo power ON</td>
</tr>
</tbody>
</table>

- **[Error]**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>Error</td>
</tr>
</tbody>
</table>
System Monitor API

mpGetPlayStatus

Retrieves the play status of the job.

**Syntax**

```
LONG mpGetPlayStatus ( 
    MP_PLAY_STATUS_RSP_DATA* rData 
);
```

**Parameter**

```
[rData]
[out] The pointer to the data structure which receives the play status
```

- **MP_PLAY_STATUS_RSP_DATA**
  [out] The data structure which receives the play status
  
  ```
  typedef struct
  {
    SHORT sStart;
    SHORT sHold;
  } MP_PLAY_STATUS_RSP_DATA
  ```

  **Member:**
  
  - `<sStart>`
    Operation status
    
    | Value | Description |
    |-------|-------------|
    | 1     | Start ON    |
    | 0     | Start OFF   |

  - `<sHold>`
    Hold status
    
    | Value | Description |
    |-------|-------------|
    | 1     | Hold ON     |
    | 0     | Hold OFF    |

**Return Value**

- 0    Normal end
- -1   Error
mpGetMasterJob

Retrieves the master job name.

■ Syntax

```c
LONG mpGetMasterJob(
    MP_TASK_SEND_DATA* sData,
    MP_JOB_NAME_RSP_DATA* rData
);
```

■ Parameter

- **[sData]**
  - **[in]** The pointer to the data structure which specifies the task number
    - **MP_TASK_SEND_DATA**
      - **[in]** The data structure which specifies the task number
        - Syntax:
          ```c
typedef struct {
    SHORT  sTaskNo;
    CHAR   reserved[2];
  } MP_TASK_SEND_DATA;
```
        - Member: `<sTaskNo>`
          Task number

- **[rData]**
  - **[out]** The pointer to the data structure which receives the job name
    - **MP_JOB_NAME_RSP_DATA**
      - **[out]** The data structure which receives the job name
        - Syntax:
          ```c
#define MAX_JOB_NAME_LEN (33)
typedef struct {
    CHAR  cJobName[MAX_JOB_NAME_LEN];
    CHAR  reserved[3];
} MP_JOB_NAME_RSP_DATA;
```
        - Member: `<cJobName[MAX_JOB_NAME_LEN]>`
          Job name (up to 32 characters for a job name)
<table>
<thead>
<tr>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>-1</td>
<td>Error</td>
</tr>
</tbody>
</table>
mpGetCurJob

Retrieves the name, line number, step number of the current job.

**Syntax**

```c
LONG mpGetCurJob (    
MP_TASK_SEND_DATA*    sData,  
MP_CUR_JOB_RSP_DATA*  rData  
);
```

**Parameter**

*sData*

[in] The pointer to the data structure which specifies the task number

- **MP_TASK_SEND_DATA**
  [in] The data structure which specifies the task number
  Syntax:  
  ```c
typedef struct  
{  
    SHORT sTaskNo;  
    CHAR reserved[2];  
} MP_TASK_SEND_DATA;
```
  Member:  
  ```c
  <sTaskNo>
  Task number
  ```

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Master task</td>
</tr>
<tr>
<td>1</td>
<td>Subtask 1</td>
</tr>
<tr>
<td>2</td>
<td>Subtask 2</td>
</tr>
<tr>
<td>3</td>
<td>Subtask 3</td>
</tr>
<tr>
<td>4</td>
<td>Subtask 4</td>
</tr>
<tr>
<td>5</td>
<td>Subtask 5</td>
</tr>
</tbody>
</table>

*rData*

[-out] The pointer to the data structure which receives the current job information

- **MP_CUR_JOB_RSP_DATA**
  [out] The data structure which receives the current job information
  Syntax:  
  ```c
#define MAX_JOB_NAME_LEN (33)  
typedef struct  
{  
    USHORT usJobLine;  
    USHORT usStep;  
    CHAR cJobName[MAX_JOB_NAME_LEN];  
    CHAR reserved[3];  
} MP_CUR_JOB_RSP_DATA;
```
System Monitor API

mpGetCurJob

Member:  
\(<usJobLine>\)
  Job line
\(<usStep>\)
  Step
\(<cJobName[\text{MAX\_JOB\_NAME\_LEN}]>\)
  Job name (up to 32 characters for a job name)

**Return Value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>-1</td>
<td>Error</td>
</tr>
</tbody>
</table>
mpGetSpecialOpStatus

Retrieves the special operation status.

For more information on the special operation, refer to section 4.2.2 “Special Playback Operations” of “FS100 OPERATOR'S MANUAL (RE-CSO-A043)”.

Syntax

```c
LONG  mpGetSpecialOpStatus ( 
    MP_SPECIAL_OP_RSP_DATA*  rData 
); 
```

Parameter

[rData]

[out] The pointer to the data structure which receives the special operation status

• MP_SPECIAL_OP_RSP_DATA

[out] The data structure which receives the special operation status

Syntax: typedef struct
{  
    SHORT  sSpecialOpStatus;  
    CHAR    reserved[2];  
} MP_SPECIAL_OP_RSP_DATA;

Member: <sSpecialOpStatus>
Special operation (play mode only)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D00</td>
<td>Check operation</td>
</tr>
<tr>
<td>D01</td>
<td>Safety speed operation</td>
</tr>
<tr>
<td>D02</td>
<td>Dry-run speed operation</td>
</tr>
<tr>
<td>D03</td>
<td>Machine lock operation</td>
</tr>
<tr>
<td>D04</td>
<td>(Reserved)</td>
</tr>
<tr>
<td>D05</td>
<td>Low speed operation</td>
</tr>
<tr>
<td>D06</td>
<td>Weaving prohibited</td>
</tr>
<tr>
<td>D07-D09</td>
<td>(Reserved)</td>
</tr>
<tr>
<td>D010</td>
<td>Pressure instruction prohibited</td>
</tr>
<tr>
<td>D11-D15</td>
<td>(Reserved)</td>
</tr>
</tbody>
</table>

Return Value

0    Normal end
-1   Error
mpGetJobDate

Retrieves the date when the job is changed.

- **Syntax**
  
  ```c
  LONG mpGetJobDate ( 
  MP_JOB_NAME_SEND_DATA* sData, 
  MP_SYS_TIME_RSP_DATA* rData 
  );
  ```

- **Parameter**

  **[sData]**
  
  [in] The pointer to the data structure which specifies the job name
  
  - `MP_JOB_NAME_SEND_DATA`
  
  [in] The data structure which specifies the job name
  
  Syntax:
  ```c
  #define MAX_JOB_NAME_LEN (33)
  typedef struct {
    CHAR cJobName[MAX_JOB_NAME_LEN];
    CHAR reserved[3];
  } MP_JOB_NAME_SEND_DATA;
  ```

  **Member:**
  `<cJobName[MAX_JOB_NAME_LEN]>`
  
  Job name (up to 32 characters for a job name)

  **[rData]**
  
  [out] The pointer to the data structure which receives the system time
  
  - `MP_SYS_TIME_RSP_DATA`
  
  [out] The data structure which receives the system time
  
  Syntax:
  ```c
  typedef struct {
    SHORT sStartYear;
    SHORT sStartMonth;
    SHORT sStartDay;
    SHORT sStartHour;
    SHORT sStartMin;
    SHORT sStartSec;
    LONG lElapsedTime;
  } MP_SYS_TIME_RSP_DATA;
  ```

  **Member:**
  `<sStartYear>`
  
  Year

  `<sStartMonth>`
  
  Month

  `<sStartDay>`
  
  Day

  `<sStartHour>`
  
  Hour
System Monitor API

mpGetJobDate

<sStartMin>
Minute
</sStartMin>

<sStartSec>
Second
</sStartSec>

<lElapsedTime>
Not used
</lElapsedTime>

Return Value

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>-1</td>
<td>Error</td>
</tr>
</tbody>
</table>
mpGetCartPos

Retrieves the current position in Cartesian coordinates.

**Syntax**

```c
LONG mpGetCartPos(
    MP_CTRL_GRP_SEND_DATA* sData,
    MP_CART_POS_RSP_DATA* rData
);
```

**Parameter**

[sData]

[in] The pointer to the data structure which specifies the control group

- MP_CTRL_GRP_SEND_DATA

[in] The data structure which specifies the control group

Syntax:

```c
typedef struct {
    CTRLG_T sCtrlGrp;
} MP_CTRL_GRP_SEND_DATA;
```

Member: `<sCtrlGrp>`

<table>
<thead>
<tr>
<th>Control group</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>R1 (Robot 1)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>R2 (Robot 2)</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>B1 (Base 1)</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>B2 (Base 2)</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12</td>
<td></td>
</tr>
<tr>
<td></td>
<td>13</td>
<td></td>
</tr>
<tr>
<td></td>
<td>14</td>
<td></td>
</tr>
<tr>
<td></td>
<td>15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>S1 (Station 1)</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>S2 (Station 2)</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>S3 (Station 3)</td>
</tr>
</tbody>
</table>
**System Monitor API**

```c
mpGetCartPos
```

### [rData]

[out] The pointer to the data structure which receives the cartesian position

- **MP_CART_POS_RSP_DATA**
  [out] The data structure which receives the cartesian position

**Syntax:**

```c
#define MAX_CART_AXES (6)

typedef struct {
    LONG lPos[MAX_CART_AXES];
    SHORT sConfig;
    CHAR reserved[2];
} MP_CART_POS_RSP_DATA;
```

**Member:**

- `<lPos[MAX_CART_AXES]>` Cartesian position (up to 6 arrays)

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lPos[0]</td>
<td>X-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>lPos[1]</td>
<td>Y-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>lPos[2]</td>
<td>Z-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>lPos[3]</td>
<td>Wrist angle Rx (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>lPos[4]</td>
<td>Wrist angle Ry (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>lPos[5]</td>
<td>Wrist angle Rz (unit: 0.0001 deg)</td>
</tr>
</tbody>
</table>

- `<sConfig>` Relative position information

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D00</td>
<td>0: Front</td>
</tr>
<tr>
<td></td>
<td>1: Back</td>
</tr>
<tr>
<td>D01</td>
<td>0: Upper arm</td>
</tr>
<tr>
<td></td>
<td>1: Lower arm</td>
</tr>
<tr>
<td>D02</td>
<td>0: Flip</td>
</tr>
<tr>
<td></td>
<td>1: No flip</td>
</tr>
<tr>
<td>D03</td>
<td>0: R &lt; 180</td>
</tr>
<tr>
<td></td>
<td>1: R &gt;= 180</td>
</tr>
<tr>
<td>D04</td>
<td>0: T &lt; 180</td>
</tr>
<tr>
<td></td>
<td>1: T &gt;= 180</td>
</tr>
<tr>
<td>D05</td>
<td>0: S &lt; 180</td>
</tr>
<tr>
<td></td>
<td>1: S &gt;= 180</td>
</tr>
<tr>
<td>D06 - D15</td>
<td>Reserved by manufacturer</td>
</tr>
</tbody>
</table>

For more information on the position data, refer to Chapter 2 "Manipulator Coordinate Systems and Operations" of "FS100 OPERATOR’S MANUAL (RE-CSO-A043)".

**Return Value**

- 0 Normal end
- -1 Error
mpGetCartPosEx

Retrieves the current position in Cartesian coordinates with a specified coordinate frame (Robot, Base, User).

**Syntax**

```c
LONG mpGetCartPos (
    MP_CARTPOS_EX_SEND_DATA* sData,
    MP_CART_POS_RSP_DATA_EX* rData
);
```

**Parameter**

**[sData]**

[in] The pointer of the control group send data structure
- **MP_CARTPOS_EX_SEND_DATA**
  - [in] The present position send data structure
    - Syntax: `typedef struct {
        SHORT sRobotNo;
        SHORT sFrame;
        SHORT sToolNo;
        CHAR reserved[2];
    } MP_CARTPOS_EX_SEND_DATA;`
    - Member: `<sRobotNo>`
      - **Control Group**
      - **Value** | **Description**
        - 0 | R1 (Robot 1)
        - 1 | R2 (Robot 2)

    - Member: `<sFrame>`
      - **Specified coordinate frame**
      - **Value** | **Description**
        - 0 | Base coordinate
        - 1 | Robot coordinate
        - 2, 3, ... | User coordinate number

    - Member: `<sToolNo>`
      - **Tool number(0 - 23)**
mpGetCartPosEx

[rData]
[out] The pointer of the Cartesian coordinate position receive data structure

- MP_CART_POS_RSP_DATA_EX
  [out] The Cartesian coordinate position receive data structure
  Syntax: #define MAX_CART_AXES_EX (12)
  
  typedef struct
  {
    LONG lPos[MAX_CART_AXES_EX];
    SHORT sConfig;
    CHAR reserved[2];
  } MP_CART_POS_RSP_DATA_EX;
  
  Member: <lPos[MAX_CART_AXES_EX]>
  Cartesian coordinate position (Maximum is 12)

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lPos[0]</td>
<td>X-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>lPos[1]</td>
<td>Y-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>lPos[2]</td>
<td>Z-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>lPos[3]</td>
<td>Wrist angle Rx (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>lPos[4]</td>
<td>Wrist angle Ry (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>lPos[5]</td>
<td>Wrist angle Rz (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>lPos[6]</td>
<td>Elbow angle E (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>lPos[7]</td>
<td>Reserved</td>
</tr>
<tr>
<td>lPos[8]</td>
<td>1st external axis pulse value (micron in the case of traveling axis)</td>
</tr>
<tr>
<td>lPos[9]</td>
<td>2nd external axis pulse value (micron in the case of traveling axis)</td>
</tr>
<tr>
<td>lPos[10]</td>
<td>3rd external axis pulse value (micron in the case of traveling axis)</td>
</tr>
</tbody>
</table>

<sConfig>
Relative position information

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D00</td>
<td>0: Front 1: Back</td>
</tr>
<tr>
<td>D01</td>
<td>0: Upper arm 1: Lower arm</td>
</tr>
<tr>
<td>D02</td>
<td>0: Flip 1: No flip</td>
</tr>
<tr>
<td>D03</td>
<td>0: R &lt; 180 1: R &gt;= 180</td>
</tr>
<tr>
<td>D04</td>
<td>0: T &lt; 180 1: T &gt;= 180</td>
</tr>
<tr>
<td>D05</td>
<td>0: S &lt; 180 1: S &gt;= 180</td>
</tr>
<tr>
<td>D06</td>
<td>0: L &lt; 0 1: L &gt;= 0</td>
</tr>
<tr>
<td>D07 - D15</td>
<td>Reserved by manufacturer</td>
</tr>
</tbody>
</table>

For more information on the position data, refer to Chapter 2 “Manipulator Coordinate Systems and Operations” of “FS100 OPERATOR’S MANUAL (RE-CSO-A043)”.

Return Value

0   Normal end
-1   Error
mpGetPulsePos

Retrieves the current position in pulse counts.

- **Syntax**

  ```c
  LONG mpGetPulsePos ( 
  MP_CTRL_GRP_SEND_DATA* sData, 
  MP_PULSE_POS_RSP_DATA* rData );
  ```

- **Parameter**

  **[sData]**

  [in] The pointer to the data structure which specifies the control group

  • **MP_CTRL_GRP_SEND_DATA**

  [in] The data structure which specifies the control group

  Syntax:

  ```c
  typedef struct {
    CTRLG_T sCtrlGrp;
  } MP_CTRL_GRP_SEND_DATA;
  ```

  Member: `<sCtrlGrp>`

  Control group

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>R1 (Robot 1)</td>
</tr>
<tr>
<td>1</td>
<td>R2 (Robot 2)</td>
</tr>
<tr>
<td>2</td>
<td>Not in use</td>
</tr>
<tr>
<td>3</td>
<td>Not in use</td>
</tr>
<tr>
<td>4</td>
<td>Not in use</td>
</tr>
<tr>
<td>5</td>
<td>Not in use</td>
</tr>
<tr>
<td>6</td>
<td>Not in use</td>
</tr>
<tr>
<td>7</td>
<td>B1 (Base 1)</td>
</tr>
<tr>
<td>8</td>
<td>B2 (Base 2)</td>
</tr>
<tr>
<td>9</td>
<td>Not in use</td>
</tr>
<tr>
<td>10</td>
<td>Not in use</td>
</tr>
<tr>
<td>11</td>
<td>Not in use</td>
</tr>
<tr>
<td>12</td>
<td>S1 (Station 1)</td>
</tr>
<tr>
<td>13</td>
<td>S2 (Station 2)</td>
</tr>
<tr>
<td>14</td>
<td>S3 (Station 3)</td>
</tr>
<tr>
<td>15</td>
<td>Not in use</td>
</tr>
<tr>
<td>16</td>
<td>S1 (Station 1)</td>
</tr>
<tr>
<td>17</td>
<td>S2 (Station 2)</td>
</tr>
<tr>
<td>18</td>
<td>S3 (Station 3)</td>
</tr>
</tbody>
</table>
System Monitor API

mpGetPulsePos

[rData]
[out] The pointer to the data structure which receives the position coordinates in pulse

• MP_PULSE_POS_RSP_DATA
[out] The data structure which receives the position coordinates in pulse

Syntax:
#define MAX_PULSE_AXES (8)

typedef struct
{
    LONG lPos[MAX_PULSE_AXES];
} MP_PULSE_POS_RSP_DATA;

Member: <lPos[MAX_PULSE_AXES]>
Position coordinates in pulse (up to 8 arrays)

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lPos[0]</td>
<td>1st axis (S) pulse value</td>
</tr>
<tr>
<td>lPos[1]</td>
<td>2nd axis (L) pulse value</td>
</tr>
<tr>
<td>lPos[2]</td>
<td>3rd axis (U) pulse value</td>
</tr>
<tr>
<td>lPos[3]</td>
<td>4th axis (R) pulse value</td>
</tr>
<tr>
<td>lPos[4]</td>
<td>5th axis (B) pulse value</td>
</tr>
<tr>
<td>lPos[5]</td>
<td>6th axis (T) pulse value</td>
</tr>
<tr>
<td>lPos[6]</td>
<td>7th axis (E) pulse value</td>
</tr>
<tr>
<td>lPos[7]</td>
<td>8th axis pulse value</td>
</tr>
</tbody>
</table>

Return Value

0 Normal end

-1 Error
mpGetRadPos

Retrieves the current position in radians. When linear motion axes such as servo tracks are included, -1 (error) is returned.

**Syntax**

```c
LONG  mpGetRadPos ( 
MP_CTRL_GRP_SEND_DATA* sData,  
MP_RAD_POS_RSP_DATA* rData 
);
```

**Parameter**

**[sData]**

[in] The pointer to the data structure which specifies the control group

- **MP_CTRL_GRP_SEND_DATA**
  [in] The data structure which specifies the control group
  Syntax: `typedef struct {
            CTRLG_T sCtrlGrp;
        } MP_CTRL_GRP_SEND_DATA;`
  Member: `<sCtrlGrp>`

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>R1 (Robot 1)</td>
</tr>
<tr>
<td>1</td>
<td>R2 (Robot 2)</td>
</tr>
<tr>
<td>2</td>
<td>Not in use</td>
</tr>
<tr>
<td>3</td>
<td>Not in use</td>
</tr>
<tr>
<td>4</td>
<td>Not in use</td>
</tr>
<tr>
<td>5</td>
<td>Not in use</td>
</tr>
<tr>
<td>6</td>
<td>Not in use</td>
</tr>
<tr>
<td>7</td>
<td>Not in use</td>
</tr>
<tr>
<td>8</td>
<td>B1 (Base 1)</td>
</tr>
<tr>
<td>9</td>
<td>B2 (Base 2)</td>
</tr>
<tr>
<td>10</td>
<td>Not in use</td>
</tr>
<tr>
<td>11</td>
<td>Not in use</td>
</tr>
<tr>
<td>12</td>
<td>Not in use</td>
</tr>
<tr>
<td>13</td>
<td>Not in use</td>
</tr>
<tr>
<td>14</td>
<td>Not in use</td>
</tr>
<tr>
<td>15</td>
<td>Not in use</td>
</tr>
<tr>
<td>16</td>
<td>S1 (Station 1)</td>
</tr>
<tr>
<td>17</td>
<td>S2 (Station 2)</td>
</tr>
<tr>
<td>18</td>
<td>S3 (Station 3)</td>
</tr>
</tbody>
</table>
**[rData]**

[out] The pointer to the data structure which receives the position coordinates in radians

- **MP_RAD_POS_RSP_DATA**
  
  [out] The data structure which receives the position coordinates in radians

**Syntax:**
```
#define MAX_PULSE_AXES (8)
typedef struct
{
    LONG lRadPos[MAX_PULSE_AXES];
} MP_RAD_POS_RSP_DATA;
```

**Member:** `<lRadPos[MAX_PULSE_AXES]>`
Position coordinates in radians (up to 8 arrays)

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lRadPos[0]</td>
<td>1st axis (S) rad value</td>
</tr>
<tr>
<td>lRadPos[1]</td>
<td>2nd axis (L) rad value</td>
</tr>
<tr>
<td>lRadPos[2]</td>
<td>3rd axis (U) rad value</td>
</tr>
<tr>
<td>lRadPos[3]</td>
<td>4th axis (R) rad value</td>
</tr>
<tr>
<td>lRadPos[4]</td>
<td>5th axis (B) rad value</td>
</tr>
<tr>
<td>lRadPos[5]</td>
<td>6th axis (T) rad value</td>
</tr>
<tr>
<td>lRadPos[6]</td>
<td>7th axis (E) rad value</td>
</tr>
<tr>
<td>lRadPos[7]</td>
<td>8th axis rad value</td>
</tr>
</tbody>
</table>

**Return Value**

- 0  Normal end
- -1  Error
mpGetRadPosEx

Retrieves the current position in radians. Linear motion axes such as servo tracks are retrieved not in radians, but in position (unit: 0.001mm).

### Syntax

```c
LONG mpGetRadPosEX(
    MP_CTRL_GRP_SEND_DATA* sData,
    MP_RAD_POS_RSP_DATA_EX* rData);
```

### Parameter

**[sData]**

[in] The pointer to the data structure which specifies the control group

- **MP_CTRL_GRP_SEND_DATA**
  
  [in] The data structure which specifies the control group

  Syntax: `typedef struct {
    CTRLG_T sCtrlGrp;
  } MP_CTRL_GRP_SEND_DATA;`

  Member: `<sCtrlGrp>`

  Control group

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>R1 (Robot 1)</td>
</tr>
<tr>
<td>1</td>
<td>R2 (Robot 2)</td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Not in use</td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>B1 (Base 1)</td>
</tr>
<tr>
<td>9</td>
<td>B2 (Base 2)</td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Not in use</td>
</tr>
<tr>
<td>14</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>S1 (Station 1)</td>
</tr>
<tr>
<td>17</td>
<td>S2 (Station 2)</td>
</tr>
<tr>
<td>18</td>
<td>S3 (Station 3)</td>
</tr>
</tbody>
</table>
mpGetRadPosEx

**[rData]**

[out] The pointer to the data structure (expanded) which receives the position coordinates in radians

- **MP_RAD_POS_RSP_DATA_EX**
  [out] The data structure (expanded) which receives the position coordinates in radians

Syntax:
```
define MAX_PULSE_AXES (8)
typedef struct
{
    LONG lRadPos[MAX_PULSE_AXES];
    LONG lRadUnit[MAX_PULSE_AXES];
} MP_RAD_POS_RSP_DATA_EX;
```

Member: `<lRadPos[MAX_PULSE_AXES]>`
Position coordinates in radians (up to 8 arrays)

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lRadPos[0]</td>
<td>1st axis (S) rad or mm value</td>
</tr>
<tr>
<td>lRadPos[1]</td>
<td>2nd axis (L) rad or mm value</td>
</tr>
<tr>
<td>lRadPos[2]</td>
<td>3rd axis (U) rad or mm value</td>
</tr>
<tr>
<td>lRadPos[3]</td>
<td>4th axis (R) rad or mm value</td>
</tr>
<tr>
<td>lRadPos[4]</td>
<td>5th axis (B) rad or mm value</td>
</tr>
<tr>
<td>lRadPos[5]</td>
<td>6th axis (T) rad or mm value</td>
</tr>
<tr>
<td>lRadPos[6]</td>
<td>7th axis (E) rad or mm value</td>
</tr>
<tr>
<td>lRadPos[7]</td>
<td>8th axis rad or mm value</td>
</tr>
</tbody>
</table>

`<lRadUnit[MAX_PULSE_AXES]>`
Units (up to 8 arrays)

- **MP_POS_UNIT_DEGREE** 1 Degree (0.0001 deg)
- **MP_POS_UNIT_DISTANCE** 2 Distance (0.001 mm)
- **MP_POS_UNIT_RADIAN** 3 Radian (0.0001 rad)

### Return Value

- **0** Normal end
- **-1** Error
mpGetDegPos

Retrieves the current position in degrees. When linear motion axes such as servo tracks are included, -1 (error) is returned.

- **Syntax**

  ```c
  LONG  mpGetDegPos (
      MP_CTRL_GRP_SEND_DATA*  sData,
      MP_DEG_POS_RSP_DATA*  rData
  );
  ```

- **Parameter**

  `[sData]`

  [in] The pointer to the data structure which specifies the control group

  - **MP_CTRL_GRP_SEND_DATA**
    [in] The data structure which specifies the control group
    Syntax:  typedef struct
    
    ```
    {
        CTRLG_T  sCtrlGrp;
    }
    ```
    ```
    MP_CTRL_GRP_SEND_DATA;
    ```
    Member:  `<sCtrlGrp>`
    Control group

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>R1 (Robot 1)</td>
</tr>
<tr>
<td>1</td>
<td>R2 (Robot 2)</td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>B1 (Base 1)</td>
</tr>
<tr>
<td>9</td>
<td>B2 (Base 2)</td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>S1 (Station 1)</td>
</tr>
<tr>
<td>17</td>
<td>S2 (Station 2)</td>
</tr>
<tr>
<td>18</td>
<td>S3 (Station 3)</td>
</tr>
</tbody>
</table>
System Monitor API

mpGetDegPos

[rData]
[out] The pointer to the data structure which receives the position coordinates in degrees

• MP_DEG_POS_RSP_DATA
  [out] The data structure which receives the position coordinates in degrees

Syntax:
#define MAX_PULSE_AXES (8)
typedef struct
{
  LONG lDegPos[MAX_PULSE_AXES];
} MP_DEG_POS_RSP_DATA;

Member:
  <lDegPos[MAX_PULSE_AXES]>
  Position coordinates in degrees (up to 8 arrays)

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lDegPos[0]</td>
<td>1st axis (S) deg value</td>
</tr>
<tr>
<td>lDegPos[1]</td>
<td>2nd axis (L) deg value</td>
</tr>
<tr>
<td>lDegPos[2]</td>
<td>3rd axis (U) deg value</td>
</tr>
<tr>
<td>lDegPos[3]</td>
<td>4th axis (R) deg value</td>
</tr>
<tr>
<td>lDegPos[4]</td>
<td>5th axis (B) deg value</td>
</tr>
<tr>
<td>lDegPos[5]</td>
<td>6th axis (T) deg value</td>
</tr>
<tr>
<td>lDegPos[6]</td>
<td>7th axis (E) deg value</td>
</tr>
<tr>
<td>lDegPos[7]</td>
<td>8th axis deg value</td>
</tr>
</tbody>
</table>

■ Return Value

0    Normal end
-1   Error
mpGetDegPosEx

Retrieves the current position in degrees. Linear motion axes such as servo tracks are retrieved not in degrees (unit: 0.0001deg), but in position (unit: 0.001mm).

- **Syntax**

  ```c
  LONG  mpGetDegPosEX (  
    MP_CTRL_GRP_SEND_DATA*  sData,  
    MP_DEG_POS_RSP_DATA_EX*  rData  
  );
  ```

- **Parameter**

  **[sData]**

  - [in] The pointer to the data structure which specifies the control group
    
    ```c
    typedef struct  
    {  
      CTRLG_T  sCtrlGrp;  
    } MP_CTRL_GRP_SEND_DATA;
    ```

    **Member:** `<sCtrlGrp>`

    Control group

    | Value | Description       |
    |-------|-------------------|
    | 0     | R1 (Robot 1)      |
    | 1     | R2 (Robot 2)      |
    | 2     |                   |
    | 3     |                   |
    | 4     | Not in use        |
    | 5     |                   |
    | 6     |                   |
    | 7     |                   |
    | 8     | B1 (Base 1)       |
    | 9     | B2 (Base 2)       |
    | 10    |                   |
    | 11    | Not in use        |
    | 12    |                   |
    | 13    |                   |
    | 14    |                   |
    | 15    |                   |
    | 16    | S1 (Station 1)    |
    | 17    | S2 (Station 2)    |
    | 18    | S3 (Station 3)    |
**mpGetDegPosEx**

[**rData**]

[out] The pointer to the data structure (expanded) which receives the position coordinates in degrees

- MP_DEG_POS_RSP_DATA_EX
  
  [out] The data structure (expanded) which receives the position coordinates in degrees

Syntax:

```c
#define MAX_PULSE_AXES (8)
typedef struct {
    LONG lDegPos[MAX_PULSE_AXES];
    LONG lDegUnit[MAX_PULSE_AXES];
} MP_DEG_POS_RSP_DATA_EX;
```

Member: `<lDegPos[MAX_PULSE_AXES]>`

- Position coordinates in degrees (up to 8 arrays)

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lDegPos[0]</td>
<td>1st axis (S) deg or mm value</td>
</tr>
<tr>
<td>lDegPos[1]</td>
<td>2nd axis (L) deg or mm value</td>
</tr>
<tr>
<td>lDegPos[2]</td>
<td>3rd axis (U) deg or mm value</td>
</tr>
<tr>
<td>lDegPos[3]</td>
<td>4th axis (R) deg or mm value</td>
</tr>
<tr>
<td>lDegPos[4]</td>
<td>5th axis (B) deg or mm value</td>
</tr>
<tr>
<td>lDegPos[5]</td>
<td>6th axis (T) deg or mm value</td>
</tr>
<tr>
<td>lDegPos[6]</td>
<td>7th axis (E) deg or mm value</td>
</tr>
<tr>
<td>lDegPos[7]</td>
<td>8th axis deg or mm value</td>
</tr>
</tbody>
</table>

Member: `<lDegUnit[MAX_PULSE_AXES]>`

- Position coordinates in degrees (up to 8 arrays)

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_POS_UNIT_DEGREE</td>
<td>1 Degree (0.0001 deg)</td>
</tr>
<tr>
<td>MP_POS_UNIT_DISTANCE</td>
<td>2 Distance (0.001 mm)</td>
</tr>
<tr>
<td>MP_POS_UNIT_RADIAN</td>
<td>3 Radian (0.0001 rad)</td>
</tr>
</tbody>
</table>

- **Return Value**

  - 0  Normal end
  - -1  Error
mpGetFBPulsePos

Retrieves the feedback position in pulse counts.

■ Syntax

```c
LONG mpGetFBPulsePos(
    MP_CTRL_GRP_SEND_DATA* sData,
    MP_FB_PULSE_POS_RSP_DATA* rData);
```

■ Description

mpGetPulsePos retrieves the current commanded position, and mpGetFBPulsePos retrieves the current actual position of the manipulator (feedback value).

■ Parameter

**[sData]**

[in] The pointer to the data structure which specifies the control group

- **MP_CTRL_GRP_SEND_DATA**

[in] The data structure which specifies the control group

```c
typedef struct {
    CTRLG_T sCtrlGrp;
} MP_CTRL_GRP_SEND_DATA;
```

Member: `<sCtrlGrp>`

Control group

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>R1 (Robot 1)</td>
</tr>
<tr>
<td>1</td>
<td>R2 (Robot 2)</td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Not in use</td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>B1 (Base 1)</td>
</tr>
<tr>
<td>9</td>
<td>B2 (Base 2)</td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Not in use</td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>S1 (Station 1)</td>
</tr>
<tr>
<td>17</td>
<td>S2 (Station 2)</td>
</tr>
<tr>
<td>18</td>
<td>S3 (Station 3)</td>
</tr>
</tbody>
</table>
### [rData]

[out] The pointer to the data structure which receives the feedback position coordinates in pulse

- **MP_FB_PULSE_POS_RSP_DATA**
  
  [out] The data structure which receives the feedback position coordinates in pulse

Syntax:

```c
#define MAX_PULSE_AXES (8)

typedef struct {
    LONG lPos[MAX_PULSE_AXES];
} MP_FB_PULSE_POS_RSP_DATA;
```

Member: `<lPos[MAX_PULSE_AXES]>`

Position coordinates in pulse (up to 8 arrays)

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lPos[0]</td>
<td>1st axis (S) pulse value</td>
</tr>
<tr>
<td>lPos[1]</td>
<td>2nd axis (L) pulse value</td>
</tr>
<tr>
<td>lPos[2]</td>
<td>3rd axis (U) pulse value</td>
</tr>
<tr>
<td>lPos[3]</td>
<td>4th axis (R) pulse value</td>
</tr>
<tr>
<td>lPos[4]</td>
<td>5th axis (B) pulse value</td>
</tr>
<tr>
<td>lPos[5]</td>
<td>6th axis (T) pulse value</td>
</tr>
<tr>
<td>lPos[6]</td>
<td>7th axis (E) pulse value</td>
</tr>
<tr>
<td>lPos[7]</td>
<td>8th axis pulse value</td>
</tr>
</tbody>
</table>

### Return Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>-1</td>
<td>Error</td>
</tr>
</tbody>
</table>
mpGetServoSpeed

Retrieves the current servo speed in pulse counts (unit: second).

Syntax

```c
LONG mpGetServoSpeed (MP_CTRL_GRP_SEND_DATA* sData,
                        MP_SERVO_SPEED_RSP_DATA* rData);
```

Parameter

**[sData]**

[in] The pointer to the data structure which specifies the control group

- **MP_CTRL_GRP_SEND_DATA**

  [in] The data structure which specifies the control group

  Syntax:

  ```c
typedef struct {
    CTRLG_T sCtrlGrp;
  } MP_CTRL_GRP_SEND_DATA;
  ```

  Member: `<sCtrlGrp>`

  **Control group**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>R1 (Robot 1)</td>
</tr>
<tr>
<td>1</td>
<td>R2 (Robot 2)</td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Not in use</td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>B1 (Base 1)</td>
</tr>
<tr>
<td>9</td>
<td>B2 (Base 2)</td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Not in use</td>
</tr>
<tr>
<td>13</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>S1 (Station 1)</td>
</tr>
<tr>
<td>17</td>
<td>S2 (Station 2)</td>
</tr>
<tr>
<td>18</td>
<td>S3 (Station 3)</td>
</tr>
</tbody>
</table>
mpGetServoSpeed

[out] The pointer to the data structure which receives the servo speed

- MP_SERVO_SPEED_RSP_DATA

[out] The data structure which receives the servo speed

Syntax: #define MAX_PULSE_AXES (8)

typedef struct
  LONG  lSpeed[MAX_PULSE_AXES];
} MP_SERVO_SPEED_RSP_DATA;

Member: <lSpeed[MAX_PULSE_AXES]>
  Speed (up to 8 arrays)

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lSpeed[0]</td>
<td>In 1st axis (S) direction</td>
</tr>
<tr>
<td></td>
<td>(unit: pulses per second - pps)</td>
</tr>
<tr>
<td>lSpeed[1]</td>
<td>In 2nd axis (L) direction</td>
</tr>
<tr>
<td></td>
<td>(unit: pps)</td>
</tr>
<tr>
<td>lSpeed[2]</td>
<td>In 3rd axis (U) direction</td>
</tr>
<tr>
<td></td>
<td>(unit: pps)</td>
</tr>
<tr>
<td>lSpeed[3]</td>
<td>In 4th axis (R) direction</td>
</tr>
<tr>
<td></td>
<td>(unit: pps)</td>
</tr>
<tr>
<td>lSpeed[4]</td>
<td>In 5th axis (B) direction</td>
</tr>
<tr>
<td></td>
<td>(unit: pps)</td>
</tr>
<tr>
<td>lSpeed[5]</td>
<td>In 6th axis (T) direction</td>
</tr>
<tr>
<td></td>
<td>(unit: pps)</td>
</tr>
<tr>
<td>lSpeed[6]</td>
<td>In 7th axis (E) direction</td>
</tr>
<tr>
<td></td>
<td>(unit: pps)</td>
</tr>
<tr>
<td>lSpeed[7]</td>
<td>In 8th axis direction</td>
</tr>
<tr>
<td></td>
<td>(unit: pps)</td>
</tr>
</tbody>
</table>

Return Value

0 Normal end

-1 Error
mpGetFBSpeed

Retrieves the feedback speed in pulse counts (unit: second).

Syntax

```c
LONG mpGetFBSpeed (MP_CTRL_GRP_SEND_DATA* sData, MP_FB_SPEED_RSP_DATA* rData);
```

Description

mpGetServoSpeed retrieves the current commanded speed, and mpGetFBSpeed retrieves the current actual speed of the manipulator (feedback value).

Parameter

[sData]

[in] The pointer to the data structure which specifies the control group

- MP_CTRL_GRP_SEND_DATA

[in] The data structure which specifies the control group

Syntax:
```c
typedef struct {
    CTRLG_T sCtrlGrp;
} MP_CTRL_GRP_SEND_DATA;
```

Member: <sCtrlGrp>
Control group

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>R1 (Robot 1)</td>
</tr>
<tr>
<td>1</td>
<td>R2 (Robot 2)</td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Not in use</td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>B1 (Base 1)</td>
</tr>
<tr>
<td>9</td>
<td>B2 (Base 2)</td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Not in use</td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>S1 (Station 1)</td>
</tr>
<tr>
<td>17</td>
<td>S2 (Station 2)</td>
</tr>
<tr>
<td>18</td>
<td>S3 (Station 3)</td>
</tr>
</tbody>
</table>

[rData]
The pointer to the data structure which receives the feedback speed

- **MP_FB_SPEED_RSP_DATA**

[out] The data structure which receives the feedback speed

Syntax:
```c
#define MAX_PULSE_AXES (8)

typedef struct {
    LONG lSpeed[MAX_PULSE_AXES];
} MP_FB_SPEED_RSP_DATA;
```

Member: `<lSpeed[MAX_PULSE_AXES]>`
- Speed (up to 8 arrays)

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lSpeed[0]</td>
<td>In 1st axis (S) direction (unit: pulses per second - pps)</td>
</tr>
<tr>
<td>lSpeed[1]</td>
<td>In 2nd axis (L) direction (unit: pps)</td>
</tr>
<tr>
<td>lSpeed[2]</td>
<td>In 3rd axis (U) direction (unit: pps)</td>
</tr>
<tr>
<td>lSpeed[3]</td>
<td>In 4th axis (R) direction (unit: pps)</td>
</tr>
<tr>
<td>lSpeed[4]</td>
<td>In 5th axis (B) direction (unit: pps)</td>
</tr>
<tr>
<td>lSpeed[5]</td>
<td>In 6th axis (T) direction (unit: pps)</td>
</tr>
<tr>
<td>lSpeed[6]</td>
<td>In 7th axis (E) direction (unit: pps)</td>
</tr>
<tr>
<td>lSpeed[7]</td>
<td>In 8th axis direction (unit: pps)</td>
</tr>
</tbody>
</table>

### Return Value

- **0** Normal end
- **-1** Error
mpGetTorque

Retrieves the percentage of the maximum current servo torque value.

- **Syntax**

  ```c
  LONG mpGetTorque (MP_CTRL_GRP_SEND_DATA* sData,
                    MP_TORQUE_RSP_DATA* rData);
  ```

- **Parameters**

  **[sData]**
  
  [in] The pointer to the data structure which specifies the control group

  - **MP_CTRL_GRP_SEND_DATA**
    
    [in] The data structure which specifies the control group

    ```c
    typedef struct {
      CTRLG_T sCtrlGrp;
    } MP_CTRL_GRP_SEND_DATA;
    ```

    Member: `<sCtrlGrp>`

    Control group

    | Value | Description |
    |-------|-------------|
    | 0     | R1 (Robot 1) |
    | 1     | R2 (Robot 2) |
    | 2     |             |
    | 3     |             |
    | 4     | Not in use  |
    | 5     |             |
    | 6     |             |
    | 7     |             |
    | 8     | B1 (Base 1) |
    | 9     | B2 (Base 2) |
    | 10    |             |
    | 11    |             |
    | 12    |             |
    | 13    |             |
    | 14    |             |
    | 15    |             |
    | 16    | S1 (Station 1) |
    | 17    | S2 (Station 2) |
    | 18    | S3 (Station 3) |
mpGetTorque

[Data]
[out] The pointer to the data structure which receives the torque value

• MP_TORQUE_RSP_DATA
  [out] The data structure which receives torque the value

Syntax:
#define MAX_PULSE_AXES (8)

typedef struct
{
  LONG iTorquePcnt[MAX_PULSE_AXES];
} MP_TORQUE_RSP_DATA;

Member: <iTorquePcnt[MAX_PULSE_AXES]>
  Torque percentage (up to 8 arrays)

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iTorquePcnt[0]</td>
<td>In 1st axis (S) direction (unit: 1% of max. torque)</td>
</tr>
<tr>
<td>iTorquePcnt[1]</td>
<td>In 2nd axis (L) direction (unit: 1% of max. torque)</td>
</tr>
<tr>
<td>iTorquePcnt[2]</td>
<td>In 3rd axis (U) direction (unit: 1% of max. torque)</td>
</tr>
<tr>
<td>iTorquePcnt[3]</td>
<td>In 4th axis (R) direction (unit: 1% of max. torque)</td>
</tr>
<tr>
<td>iTorquePcnt[4]</td>
<td>In 5th axis (B) direction (unit: 1% of max. torque)</td>
</tr>
<tr>
<td>iTorquePcnt[5]</td>
<td>In 6th axis (T) direction (unit: 1% of max. torque)</td>
</tr>
<tr>
<td>iTorquePcnt[6]</td>
<td>In 7th axis (E) direction (unit: 1% of max. torque)</td>
</tr>
<tr>
<td>iTorquePcnt[7]</td>
<td>In 8th axis direction (unit: 1% of max. torque)</td>
</tr>
</tbody>
</table>

Return Value

0       Normal end
-1      Error
mpGetTorqueEx

Retrieves the absolute value (Nm) of the maximum current servo torque.

■ Syntax

LONG mpGetTorqueEx ( 
    MP_CTRL_GRP_SEND_DATA* sData, 
    MP_TORQUEEX_RSP_DATA* rData 
);

■ Parameters

[sData]
[in] The pointer to the data structure which specifies the control group

• MP_CTRL_GRP_SEND_DATA
[in] The data structure which specifies the control group

Syntax: typedef struct {
    CTRLG_T sCtrlGrp;
} MP_CTRL_GRP_SEND_DATA;

Member: <sCtrlGrp>
Control group

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>R1 (Robot 1)</td>
</tr>
<tr>
<td>1</td>
<td>R2 (Robot 2)</td>
</tr>
<tr>
<td>2</td>
<td>Not in use</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Not in use</td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>B1 (Base 1)</td>
</tr>
<tr>
<td>9</td>
<td>B2 (Base 2)</td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Not in use</td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>S1 (Station 1)</td>
</tr>
<tr>
<td>17</td>
<td>S2 (Station 2)</td>
</tr>
<tr>
<td>18</td>
<td>S3 (Station 3)</td>
</tr>
</tbody>
</table>
### [rData]

[out] The pointer to the data structure which receives the torque value

- **MP_TORQUE_EX_RSP_DATA**
  
  [out] The data structure which receives torque the value

**Syntax:**
```
#define MAX_PULSE_AXES (8)

typedef struct
{
  LONG iTorqueExPcnt[MAX_PULSE_AXES);
} MP_TORQUE_EX_RSP_DATA;
```

**Member:** 
```
<iTorqueExPcnt[MAX_PULSE_AXES]>
    Torque percentage (up to 8 arrays)
```

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iTorqueExPcnt[0]</td>
<td>In 1st axis (S) direction</td>
</tr>
<tr>
<td></td>
<td>(unit: Nm)</td>
</tr>
<tr>
<td>iTorqueExPcnt[1]</td>
<td>In 2nd axis (L) direction</td>
</tr>
<tr>
<td></td>
<td>(unit: Nm)</td>
</tr>
<tr>
<td>iTorqueExPcnt[2]</td>
<td>In 3rd axis (U) direction</td>
</tr>
<tr>
<td></td>
<td>(unit: Nm)</td>
</tr>
<tr>
<td>iTorqueExPcnt[3]</td>
<td>In 4th axis (R) direction</td>
</tr>
<tr>
<td></td>
<td>(unit: Nm)</td>
</tr>
<tr>
<td>iTorqueExPcnt[4]</td>
<td>In 5th axis (B) direction</td>
</tr>
<tr>
<td></td>
<td>(unit: Nm)</td>
</tr>
<tr>
<td>iTorqueExPcnt[5]</td>
<td>In 6th axis (T) direction</td>
</tr>
<tr>
<td></td>
<td>(unit: Nm)</td>
</tr>
<tr>
<td>iTorqueExPcnt[6]</td>
<td>In 7th axis (E) direction</td>
</tr>
<tr>
<td></td>
<td>(unit: Nm)</td>
</tr>
<tr>
<td>iTorqueExPcnt[7]</td>
<td>In 8th axis direction</td>
</tr>
<tr>
<td></td>
<td>(unit: Nm)</td>
</tr>
</tbody>
</table>

#### Return Value

- **0** Normal end
- **-1** Error
mpGetSysTimes

Retrieves the current system time.

- **Syntax**

```c
LONG mpGetSysTimes ( 
    MP_SYS_TIME_SEND_DATA* sData, 
    MP_SYS_TIME_RSP_DATA* rData 
);
```

- **Parameters**

  - **[sData]**
    - **[in]** The pointer to the data structure which specifies the system time to be retrieved
      - **MP_SYS_TIME_SEND_DATA**
        - **[in]** The data structure which specifies the system time to be retrieved
          - Syntax:
            ```c
            typedef struct {
                SHORT sTimeType;
                CHAR reserved[2];
            } MP_SYS_TIME_SEND_DATA;
            ```

          - **Member:** `<sTimeType>`
            - Type of system time

        - **Value** | **Description**
          |------------------------------------------------|
          | MP_POWER_ON_TIME_ID | Control power time |
          | MP_SERVO_ON_TIME_ID | Servo power time |
          | MP_PLAYBACK_TIME_ID | Playback time |
          | MP_MOVING_TIME_ID | Moving time |
          | MP_OPERATION_TIME_ID | Operating time |

  - **[rData]**
    - **[out]** The pointer to the data structure which receives the system time
      - **MP_SYS_TIME_RSP_DATA**
        - **[out]** The data structure which receives the system time
          - Syntax:
            ```c
            typedef struct {
                SHORT sStartYear;
                SHORT sStartMonth;
                SHORT sStartDay;
                SHORT sStartHour;
                SHORT sStartMin;
                SHORT sStartSec;
                LONG lElapsedTime;
            } MP_SYS_TIME_RSP_DATA;
            ```
Member: <sStartYear>
    Year
<sStartMonth>
    Month
<sStartDay>
    Day
<sStartHour>
    Hour
<sStartMin>
    Minute
<sStartSec>
    Second
<lElapsedTime>
    Elapsed time

■ Return Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>-1</td>
<td>Error</td>
</tr>
</tbody>
</table>
mpGetSysVersionNo

Retrieves the version number of system software (main CPU) in character strings.

■ Syntax

```c
LONG mpGetSysVersionNo
    (MP_SYS_VERSION_NO_RSP_DATA* rData
    )
```

■ Parameters

[rData]
[out] The pointer to the data structure which retrieves the system version

- MP_SYS_VERSION_NO_RSP_DATA
  [out] The data structure which retrieves the system version

Syntax:
```c
typedef struct
    {
        CHAR sys_version[OFFLINE_SYSTEM_VERSION_SIZE]; /*system version*/
        CHAR reserved[3];
    } MP_SYS_VERSION_NO_RSP_DATA;
```

#define OFFLINE_SYSTEM_VERSION_SIZE:
the number of system version character strings: 22

Member: <sys_version>
System version
(e.g.) "FS1.00.00A(JP/US)-00"

■ Return Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>-1</td>
<td>Error (Version number could not be retrieved)</td>
</tr>
</tbody>
</table>

mpGetJogSpeed

Retrieves the current jog speed.

**Syntax**

```
LONG  mpGetJogSpeed ( 
    MP_JOGSPEED_RSP_DATA*  rData 
);
```

**Parameter**

[rData]

[**out**] The pointer to the data structure which receives the jog speed

- **MP_JOGSPEED_RSP_DATA**
  [**out**] The data structure which receives the jog speed

Syntax:
```
typedef struct
{
    SHORT  sJogSpeed;
    CHAR    reserved[2];
} MP_JOGSPEED_RSP_DATA;
```

Member: <sJogSpeed>
- Jog speed

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Inching</td>
</tr>
<tr>
<td>1</td>
<td>Low speed</td>
</tr>
<tr>
<td>2</td>
<td>Medium speed</td>
</tr>
<tr>
<td>3</td>
<td>High speed</td>
</tr>
<tr>
<td>4</td>
<td>Maximum speed ([HIGH SPEED] key)</td>
</tr>
</tbody>
</table>

**Return Value**

- 0 Normal end
- -1 Error

---

**Note**

Exclusive control by using semaphores is executed between this API and the APIs listed below. Thus, if this API is executed while one of the following APIs is already being executed, the task may be suspended to wait for the API to be completed. (waiting for semaphore acquisition)

Therefore, DO NOT use the following APIs for the task which must be performed periodically with high priority:

mpCancelError, mpResetAlarm, mpSetCycle, mpSetServoPower, mpSetMasterJob, mpSetCurJob, mpStartJob, mpHold, mpWaitForJobEnd, mpDeleteJob, mpGetJogSpeed, mpGetJogCoord, mpiMOV, mpiMOVJ, mpiMOVL, mpPulseMOVJ, mpPulseMOVL
mpGetJogCoord

Retrieves the current operational coordinates.

■ Syntax

LONG  mpGetJogCoord ( 
    MP_JOGCOORD_RSP_DATA*   rData 
);

■ Parameter

[rData]
[out] The pointer to the data structure which receives the operational coordinates

- MP_JOGCOORD_RSP_DATA
[out] The data structure which receives the operational coordinates

Syntax: typedef struct
{
    SHORT   sJogCoord;
    CHAR     reserved[2];
} MP_JOGCOORD_RSP_DATA;

Member: <sJogCoord>
Operational coordinates

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Joint coordinates</td>
</tr>
<tr>
<td>1</td>
<td>Cartesian coordinates</td>
</tr>
<tr>
<td>2</td>
<td>Cylindrical coordinates</td>
</tr>
<tr>
<td>3</td>
<td>Tool coordinates</td>
</tr>
<tr>
<td>4</td>
<td>User coordinates</td>
</tr>
<tr>
<td>5</td>
<td>External reference point</td>
</tr>
<tr>
<td>6</td>
<td>Torch coordinates</td>
</tr>
</tbody>
</table>

■ Return Value

0     Normal end
-1    Error
System Monitor API
mpGetJogCoord

Exclusive control by using semaphores is executed between this API and the APIs listed below. Thus, if this API is executed while one of the following APIs is already being executed, the task may be suspended to wait for the API to be completed. (waiting for semaphore acquisition)

Therefore, DO NOT use the following APIs for the task which must be performed periodically with high priority.

mpCancelError, mpResetAlarm, mpSetCycle, mpSetServoPower, mpSetMasterJob, mpSetCurJob, mpStartJob, mpHold, mpWaitForJobEnd, mpDeleteJob, mpGetJogSpeed, mpGetJogCoord, mpIMOV, mpMOVJ, mpMOVL, mpPulseMOVJ, mpPulseMOVL
mpCtrlGrpId2GrpNo

Converts the control group ID to the control group number.

- **Syntax**
  ```c
  int mpCtrlGrpId2GrpNo
  (MP_GRP_ID_TYPE grp_id /* specified control group id's */)
  ```

- **Description**
  The control group ID is the characteristic number assigned to each control group. This value is not changed according to the configuration of the control group in the system, but is always the same value. For example, the control group ID for S1 is the same either in the system with R1+S1 or R1+R2+S1.

  The control group number is the number assigned sequentially to the control group in the system. Thus, the number assigned to each control group differs according to the configuration of the control group in the system. For example, the control group number for S1 is 1 in the system with R1+S1, but it is 2 in the system with R1+R2+S1.

  For some of the sensor control APIs or the servo control APIs, the control group number must be specified by the parameter. To use one of those APIs, retrieve the appropriate control group number by using this API (mpCtrlGrpId2GrpNo).

- **Parameter**
  
  **[grp_id]**

  Specify the value in the following table.

<table>
<thead>
<tr>
<th>Group ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_R1_GID</td>
<td>R1 (Robot 1)</td>
</tr>
<tr>
<td>MP_R2_GID</td>
<td>R2 (Robot 2)</td>
</tr>
<tr>
<td>MP_R3_GID</td>
<td>R3 (Robot 3)</td>
</tr>
<tr>
<td>MP_R4_GID</td>
<td>R4 (Robot 4)</td>
</tr>
<tr>
<td>MP_B1_GID</td>
<td>B1 (Base 1)</td>
</tr>
<tr>
<td>MP_B2_GID</td>
<td>B2 (Base 2)</td>
</tr>
<tr>
<td>MP_B3_GID</td>
<td>B3 (Base 3)</td>
</tr>
<tr>
<td>MP_B4_GID</td>
<td>B4 (Base 4)</td>
</tr>
<tr>
<td>MP_S1_GID</td>
<td>S1 (Station 1)</td>
</tr>
<tr>
<td>MP_S2_GID</td>
<td>S2 (Station 2)</td>
</tr>
<tr>
<td>MP_S3_GID</td>
<td>S3 (Station 3)</td>
</tr>
</tbody>
</table>

- **Return Value**
  
  0 or more Normal end
  Returns the control group number which begins with 0.

  ERROR Error
mpPutVarData

Writes a variable.

**Syntax**

```c
LONG   mpPutVarData (  
    MP_VAR_DATA*     sData,  
    LONG             num  
);  
```

**Parameter**

*sData*

Pointer to the data structure that writes the variable

- **MP_VAR_DATA**
  
  Data structure that writes the variable
  
  Syntax:
  ```c
  typedef struct
  {
    USHORT   usType;
    USHORT   usIndex;
    LONG     ulValue;
  } MP_VAR_DATA;
  ```
  
  Member: <usType>
  
  Variable type
  
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_RESTYPE_VAR_B</td>
<td>Byte type</td>
</tr>
<tr>
<td>MP_RESTYPE_VAR_I</td>
<td>Integer type</td>
</tr>
<tr>
<td>MP_RESTYPE_VAR_D</td>
<td>Double-precision type</td>
</tr>
<tr>
<td>MP_RESTYPE_VAR_R</td>
<td>Real type</td>
</tr>
</tbody>
</table>

- <usIndex>
  
  Variable index

- <ulValue>
  
  Variable value

To set multiple variables, define `num` arrays, set the variable type, variable index, and variable value of each array, then set the starting address of each array to `sData`.

*num*

Number of the variable data (up to 126)
## Return Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>-1</td>
<td>Error</td>
</tr>
</tbody>
</table>

- Immediately after the job starts, the system has access to the variable area, and the API to write to the variable area such as `mpPutVarData` or `mpPutPosVarData` may return an error. Thus, for the application which requires the job start-up, describe a code with which the value is repeatedly retrieved until the error does not return.

- There is no interlock between the MotoPlus API which writes variables, I/O, or registers and the following modification methods other than MotoPlus, thus the MotoPlus API can write them. Pay attention to the interference with the other methods so that the modification by MotoPlus will not affect the system’s operation.

### Note

1. Edit of variables, I/O, or registers in the teach mode by using the programming pendant
2. Change of variables, I/O, or registers by job execution
3. Change of I/O or registers by the concurrent I/O output

- Since this API’s processing time is a few milliseconds, do not use this API in a task with the I/O control cycle or interpolation control cycle. If it is used in such a task, the process cannot be completed within the control cycle.

- When changing a parameter or loading a file from the external memory device, the process of this API cannot be executed because the system is busy. In this case, the error (-1) is returned. Retry this API after a while.
mpPutSVarInfo

 Writes S variable (fixed length). Specifies multiple variables.

**NOTE**

The system software version FS1.03-00 and IDE version 1.10 or later versions support this API.

- **Syntax**

  ```c
  int mpPutSVarInfo
  (    
       MP_SVAR_SEND_INFO *sData,
       LONG num
  )
  ```

- **Parameter**

  **[sData]**

  Pointer to the data structure which specifies variable

  ```c
typedef struct
  {
      USHORT usType;    /* Variable type (only MP_RESTYPE_VAR_S is available) */
      USHORT usIndex;   /* Variable index */
      UCHAR ucValue[S_VAR_SIZE+1]; /*S_VAR_SIZE (16 characters) + delimiter0*/
      CHAR reserved[3];
 } MP_SVAR_SEND_INFO;
  ```

  To set multiple variables, define *num* arrays, set the variable type and variable index of each array, and set the starting address of each array to *sData*.

  **[num]**

  Number of S variables (up to 10)

- **Return Value**

  0 Normal end
  -1 Error

**NOTE**

- Since this API’s processing time is a few milliseconds, do not use this API in a task with the I/O control cycle or interpolation control cycle. If it is used in such a task, the process cannot be completed within the control cycle.

- When changing a parameter or loading a file from the external memory device, the process of this API cannot be executed because the system is busy. In this case, the error (-1) is returned. Retry this API after a while.
mpWriteIO

Writes I/O.

**Syntax**

```c
LONG mpWriteIO (MP_IO_DATA* sData, LONG num);
```

**Parameter**

*sData*

Pointer to the data structure that writes I/O

- MP_IO_DATA
  - Data structure that writes I/O
  - **Syntax:**
    ```
    typedef struct {
        ULONG ulAddr;
        ULONG ulValue;
    } MP_IO_DATA;
    ```
  - **Member:**
    - `<ulAddr>`: I/O address
    - `<ulValue>`: I/O value

To set multiple I/Os, define *num* arrays, set the I/O address and I/O value of each array, then set the starting address of each array to *sData*.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10010 - 12567</td>
<td>Universal output #10010 - #12567(2048)</td>
</tr>
<tr>
<td>60010 - 60647</td>
<td>Interface panel #60010 - #60647(512)</td>
</tr>
<tr>
<td>25010 - 27567</td>
<td>Network input #25010 - #27567(2048)</td>
</tr>
<tr>
<td>1000000 - 1000559</td>
<td>Register #1000000 - #1000559(560)</td>
</tr>
</tbody>
</table>

<num>

Number of the I/O data (up to 126)

---

When retrieving multiple values simultaneously, this API temporarily makes the task priority of the application higher than the robot motion task, etc., so that the values will not be changed by another task while retrieving the values. Thus, reading many values may affect the robot motion. DO NOT specify a large value as *num* during the robot motion.
## Return Value

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>-1</td>
<td>Error</td>
</tr>
</tbody>
</table>

There is no interlock between the MotoPlus API which writes variables, I/O, or registers and the following modification methods other than MotoPlus, thus the MotoPlus API can write them. Pay attention to the interference with the other methods so that the modification by MotoPlus will not affect the system’s operation.

### NOTE

1. Edit of variables, I/O, or registers in the teach mode by using the programming pendant
2. Change of variables, I/O, or registers by job execution
3. Change of I/O or registers by the concurrent I/O output
mpPutPosVarData

Sets the position-type variable.

- **Syntax**

```c
LONG   mpPutPosVarData(
    MP_POSVAR_DATA*   sData,
    LONG    num
);
```

- **Parameter**

**[sData]**

[in] The pointer to the data structure array which specifies the position-type variable

- **MP_POSVAR_DATA**

[in] The data structure array which specifies the position-type variable

Syntax:
```c
typedef struct
{
    USHORT   usType;
    USHORT   usIndex;
    LONG     ulValue[10];
} MP_POSVAR_DATA;
```

Member:

- `<usType>`
  Variable type

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_RESTYPE_VAR_ROBOT</td>
<td>Robot</td>
</tr>
<tr>
<td>MP_RESTYPE_VAR_BASE</td>
<td>Base</td>
</tr>
<tr>
<td>MP_RESTYPE_VAR_STATION</td>
<td>Station</td>
</tr>
</tbody>
</table>

- `<usIndex>`
  Variable number

- `<ulValue[10]>`
  Setting value

To set multiple position-type variables, define `num` arrays, set the variable type, variable index, and position data of each array, then set the starting address of each array to `sData`.

---

**NOTE**

Since this API's processing time is a few milliseconds, do not use this API in a task with the I/O control cycle or interpolation control cycle. If it is used in such a task, the process cannot be completed within the control cycle.
### System Control API

#### mpPutPosVarData

<table>
<thead>
<tr>
<th>Array</th>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rData[0]</td>
<td>D05 - D00</td>
<td>Variable type</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>Pulse</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>Cartesian (base coordinates)</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>Cartesian (robot coordinates)</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>Cartesian (tool coordinates)</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>Cartesian (user coordinates)</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>Cartesian (reserved for master tool)</td>
</tr>
<tr>
<td></td>
<td>D07 - D06</td>
<td>Reserved by manufacturer</td>
</tr>
<tr>
<td></td>
<td>D08</td>
<td>0: Front 1: Back</td>
</tr>
<tr>
<td></td>
<td>D09</td>
<td>0: Upper arm 1: Lower arm</td>
</tr>
<tr>
<td></td>
<td>D10</td>
<td>0: Flip 1: No flip</td>
</tr>
<tr>
<td></td>
<td>D11</td>
<td>0: R &lt; 180 deg 1: R &gt;= 180 deg</td>
</tr>
<tr>
<td></td>
<td>D12</td>
<td>0: T &lt; 180 deg 1: T &gt;= 180 deg</td>
</tr>
<tr>
<td></td>
<td>D13</td>
<td>0: S &lt; 180 deg 1: S &gt;= 180 deg</td>
</tr>
<tr>
<td></td>
<td>D14 - D15</td>
<td>Reserved by manufacturer</td>
</tr>
<tr>
<td></td>
<td>D16 - D21</td>
<td>Tool number (0 - 23)</td>
</tr>
<tr>
<td></td>
<td>D22 - D27</td>
<td>User coordinate number</td>
</tr>
<tr>
<td></td>
<td>D28 - D31</td>
<td>Reserved by manufacturer</td>
</tr>
<tr>
<td>rData[1]</td>
<td>(Extended attribute)</td>
<td></td>
</tr>
</tbody>
</table>

#### Array

<table>
<thead>
<tr>
<th>Array</th>
<th>Pulse</th>
<th>Cartesian</th>
</tr>
</thead>
<tbody>
<tr>
<td>rData[2]</td>
<td>1st axis (S) pulse value</td>
<td>X-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>rData[3]</td>
<td>2nd axis (L) pulse value</td>
<td>Y-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>rData[4]</td>
<td>3rd axis (U) pulse value</td>
<td>Z-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>rData[5]</td>
<td>4th axis (R) pulse value</td>
<td>Wrist angle Rx (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>rData[6]</td>
<td>5th axis (B) pulse value</td>
<td>Wrist angle Ry (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>rData[7]</td>
<td>6th axis (T) pulse value</td>
<td>Wrist angle Rz (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>rData[8]</td>
<td>7th axis (E) pulse value</td>
<td>Angle Re (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>rData[9]</td>
<td>8th axis pulse value</td>
<td>8th axis pulse value (micron in the case of traveling axis)</td>
</tr>
</tbody>
</table>

#### Return Value

- **0** Normal end
- **-1** Error

**NOTE**

Immediately after the job starts, the system has access to the variable area, and the API to write to the variable area such as mpPutVarData or mpPutPosVarData may return an error. Thus, for the application which requires the job startup, describe a code with which the value is repeatedly retrieved until the error does not return.
• There is no interlock between the MotoPlus API which writes variables, I/O, or registers and the following modification methods other than MotoPlus, thus the MotoPlus API can write them. Pay attention to the interference with the other methods so that the modification by MotoPlus will not affect the system’s operation.

**NOTE**

1. Edit of variables, I/O, or registers in the teach mode by using the programming pendant
2. Change of variables, I/O, or registers by job execution
3. Change of I/O or registers by the concurrent I/O output

• When changing a parameter or loading a file from the external memory device, the process of this API cannot be executed because the system is busy. In this case, the error (-1) is returned. Retry this API after a while.
mpPutUserVars

Writes the user variable.

To compare mpGetUserVars with the other variable access APIs, such as mpPutVarData(), mpPutPosVarData and mpPutSVarInfo(), mpGetUserVars can writes user variable in high speed.

Syntax

STATUS mpPutUserVars(
    MP_USR_VAR_INFO*mp_usr_var_info
);

Parameter

[mp_usr_var_info]
[in] User variable access API interface (Input/Output)

• MP_USR_VAR_INFO

[in] The data structure that sets the user variable

Syntax:

typedef struct

{  
   var_Type;
   int var_no;
   union {
       MP_B_VAR_BUFF b;
       MP_I_VAR_BUFF i;
       MP_D_VAR_BUFF d;
       MP_R_VAR_BUFF r;
       MP_S_VAR_BUFF s;
       MP_P_VAR_BUFF p;
       MP_P_VAR_BUFF bp;
       MP_P_VAR_BUFF ex;
   } val;
} MP_USR_VAR_INFO;

Member:  <var_Type>

Variable type (Input)

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_VAR_B</td>
<td>Byte Variable</td>
</tr>
<tr>
<td>MP_VAR_I</td>
<td>Integer Variable</td>
</tr>
<tr>
<td>MP_VAR_D</td>
<td>Double Variable</td>
</tr>
<tr>
<td>MP_VAR_R</td>
<td>Real Variable</td>
</tr>
<tr>
<td>MP_VAR_S</td>
<td>Character Variable</td>
</tr>
<tr>
<td>MP_VAR_P</td>
<td>Robot position Variable</td>
</tr>
<tr>
<td>MP_VAR_BP</td>
<td>Base position Variable</td>
</tr>
<tr>
<td>MP_VAR_EX</td>
<td>Station (LEX) Variable</td>
</tr>
</tbody>
</table>
System Control API

mpPutUserVars

Syntax:

```c
#define STR_VAR_SIZE 16
#define MP_GRP_AXES_NUM 8

typedef unsigned char MP_B_VAR_BUFF;
typedef short MP_I_VAR_BUFF;
typedef long MP_D_VAR_BUFF;
typedef float MP_R_VAR_BUFF;
typedef char MP_S_VAR_BUFF[STR_VAR_SIZE];

typedef struct {
    UINT dtype;
    int tool_no;
    int uf_no;
    BITSTRING fig_ctrl;
    long data[MP_GRP_AXES_NUM];
} MP_P_VAR_BUFF;
```

Member: `MP_P_VAR_BUFF`

- `<dtype>`
  - Coordinate system type

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_PULSE_DTYPE</td>
<td>Pulse coordinate</td>
</tr>
<tr>
<td>MP_BASE_DTYPE</td>
<td>Base coordinate</td>
</tr>
<tr>
<td>MP_ROBO_DTYPE</td>
<td>Robot coordinate</td>
</tr>
<tr>
<td>MP_TOOL_DTYPE</td>
<td>Tool coordinate</td>
</tr>
<tr>
<td>MP_USER_DTYPE</td>
<td>User coordinate</td>
</tr>
<tr>
<td>MP_MTOOL_DTYPE</td>
<td>Master tool coordinate</td>
</tr>
</tbody>
</table>

- `<tool_no>`
  - Tool file No. (0 to 15)

- `<uf_no>`
  - User coordinate system file No. (0 to 15)

- `<fig_ctrl>`
  - Figure information (bits)

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_FIG_ SIDE</td>
<td>0: Front 1: Back</td>
</tr>
<tr>
<td>MP_FIG_ ELBOW</td>
<td>0: Upper elbow 1: Lower elbow</td>
</tr>
<tr>
<td>MP_FIG_ FLIP</td>
<td>0: Flip 1: No flip</td>
</tr>
<tr>
<td>MP_FIG_ R180</td>
<td>0: R&lt;180 1: R&gt;=180</td>
</tr>
<tr>
<td>MP_FIG_ T180</td>
<td>0: T&lt;180 1: T&gt;=180</td>
</tr>
<tr>
<td>MP_FIG_ S180</td>
<td>0: S&lt;180 1: S&gt;=180</td>
</tr>
</tbody>
</table>
<data>
    Position data
### Return value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_MP_VARS_ACCESS</td>
<td>An error except followings</td>
</tr>
<tr>
<td>E_MP_VAR_TYPE</td>
<td>Incorrect variable type</td>
</tr>
<tr>
<td>E_MP_VAR_NO_RANGE</td>
<td>Incorrect variable No.</td>
</tr>
<tr>
<td>E_MP_PVAR_DTYPE</td>
<td>Incorrect coordinate type of the position variable</td>
</tr>
<tr>
<td>E_MP_PVAR_TOOL_NO</td>
<td>Incorrect tool No. of the position variable</td>
</tr>
<tr>
<td>E_MP_PVAR_USER_NO</td>
<td>Incorrect user coordinate type of the position variable</td>
</tr>
<tr>
<td>E_MP_PVAR_UNSETUP</td>
<td>Undefined data of the position variable</td>
</tr>
<tr>
<td>E_MP_PMS_RET_VAL</td>
<td>Failure to retrieve the position variable</td>
</tr>
</tbody>
</table>
Beware of changing the variable from this API during the JOB operation which uses a position variable as a target position of the move instruction. There is no guarantee for changing the position variable value which will reflect the motion of the robot.

Data types which are used in this API are different from ones in the mpPutVarData(), mpPutPosVarData() and mpPutSVarInfo(). The following table shows the differences of the position variable.

### Difference of the API Parameter

<table>
<thead>
<tr>
<th>Position Variable Retrieved API</th>
<th>User Variable Retrieved API</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpPutPosVarData(</td>
<td>mpGetUserVars(</td>
</tr>
<tr>
<td>MP_POSVAR_DATA *sData;</td>
<td>MP_USR_VAR_INFO *mp_usr_var_info;</td>
</tr>
<tr>
<td>LONG num;</td>
<td>)</td>
</tr>
<tr>
<td>Syntax</td>
<td>Syntax</td>
</tr>
<tr>
<td>typedef struct {</td>
<td>typedef struct {</td>
</tr>
<tr>
<td>USHORT usType;</td>
<td>int var_type;</td>
</tr>
<tr>
<td>USHORT usIndex;</td>
<td>int var_no;</td>
</tr>
<tr>
<td>USHORT ulValue[10];</td>
<td>union {</td>
</tr>
<tr>
<td>} MP_POSVAR_DATA;</td>
<td>MP_B_VAR_BUFF b;</td>
</tr>
<tr>
<td></td>
<td>.</td>
</tr>
<tr>
<td>mpGetUserVars()</td>
<td>MP_P_VAR_BUFF p;</td>
</tr>
<tr>
<td>mpGetPosVarData()</td>
<td>MP_P_VAR_BUFF bp;</td>
</tr>
<tr>
<td></td>
<td>MP_P_VAR_BUFF ex;</td>
</tr>
<tr>
<td>mpGetUserVars()</td>
<td>} val;</td>
</tr>
<tr>
<td></td>
<td>} MP_USR_VAR_INFO;</td>
</tr>
</tbody>
</table>

### Setting Content of the Parameter

<table>
<thead>
<tr>
<th>Item</th>
<th>mpGetPosVarData()</th>
<th>mpGetUserVars()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordinate system</td>
<td>rData[0]</td>
<td>p, bp, ex</td>
</tr>
<tr>
<td>Type</td>
<td>rData[0]</td>
<td>p, bp, ex</td>
</tr>
<tr>
<td>Type</td>
<td>rData[0]</td>
<td>p, bp, ex</td>
</tr>
<tr>
<td>Type</td>
<td>rData[0]</td>
<td>p, bp, ex</td>
</tr>
<tr>
<td>Type</td>
<td>rData[0]</td>
<td>p, bp, ex</td>
</tr>
<tr>
<td>Type</td>
<td>rData[0]</td>
<td>p, bp, ex</td>
</tr>
<tr>
<td>Type</td>
<td>rData[0]</td>
<td>p, bp, ex</td>
</tr>
</tbody>
</table>
mpSetAlarm

Generates an application alarm.

- **Syntax**

  ```c
  int mpSetAlarm(
    short alm_code, 
    char *alm_msg, 
    UCHAR sub_code 
  );
  ```

- **Description**

  Generates the alarm of the specified alarm code, alarm message, and subcode.
  If an alarm is generated using this API, all the jobs under execution are stopped.

- **Parameter**

  `[alm_code]`
  Alarm code: 8000 to 8999

  `[msg]`
  Alarm message: Up to 32 characters

  `[msg]`
  Alarm sub code

- **Return Value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>ERROR</td>
<td>Error (Alarm code range exceeded)</td>
</tr>
</tbody>
</table>

**NOTE**

- Executing this API generates an alarm, and thus some operations are restricted, for example, editing is prohibited, JOB cannot be started, servo power cannot be turned ON. Reset the alarm to cancel the restriction.
- The alarm generated by this API cannot be read by an alarm-reading API “mpGetAlarmStatus” or “mpGetAlarmCode” immediately after executing this API. This is because there is a time lag between the request of generating the alarm by this API and the actual detection of the alarm.
mpCancelError

Releases from the error status.

- **Syntax**

```c
LONG mpCancelError (MP_STD_RSP_DATA* rData);
```

- **Parameter**

[rData]

[out] The pointer to the data structure which receives the error information

- **Parameter**

MP_STD_RSP_DATA

[out] The data structure which receives the error information

```c
typedef struct
{
    USHORT err_no;
    CHAR reserved[2];
} MP_STD_RSP_DATA;
```

- **Return Value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>Normal end</td>
</tr>
</tbody>
</table>

**NOTE**

Exclusive control by using semaphores is executed between this API and the APIs listed below. Thus, if this API is executed while one of the following APIs is already being executed, the task may be suspended to wait for the API to be completed. (waiting for semaphore acquisition)

Therefore, DO NOT use the following APIs for the task which must be performed periodically with high priority.

mpCancelError, mpResetAlarm, mpSetCycle, mpSetServoPower, mpSetMasterJob, mpSetCurJob, mpStartJob, mpHold, mpWaitForJobEnd, mpDeleteJob, mpGetJogSpeed, mpGetJogCoord, mpiMOV, mpiMOVJ, mpiMOVL, mpiPulseMOVJ, mpiPulseMOVL
mpResetAlarm

Resets the robot alarm.

Syntax

LONG mpResetAlarm (MP_STD_RSP_DATA* rData);

Parameter
[rData]
[out] The pointer to the data structure which receives the error information
- MP_STD_RSP_DATA
[out] The data structure which receives the error information

Syntax:
ttypedef struct
{
USHORT err_no;
CHAR reserved[2];
} MP_STD_RSP_DATA;

Member: <err_no>
Error number

Value | Description
---|---
0x0000 | Normal end

Return Value

0 Normal end
-1 Error

Exclusive control by using semaphores is executed between this API and the APIs listed below. Thus, if this API is executed while one of the following APIs is already being executed, the task may be suspended to wait for the API to be completed. (waiting for semaphore acquisition)

Therefore, DO NOT use the following APIs for the task which must be performed periodically with high priority.

mpCancelError, mpResetAlarm, mpSetCycle, mpSetServoPower, mpSetMasterJob, mpSetCurJob, mpStartJob, mpHold, mpWaitForJobEnd, mpDeleteJob, mpGetJogSpeed, mpGetJogCoord, mpiMOV, mpMOVJ, mpMOVL, mpPulseMOVJ, mpPulseMOVL
mpSetCycle

Sets the cycle mode.

- **Syntax**

  ```c
  LONG  mpSetCycle ( 
    MP_CYCLE_SEND_DATA*  sData, 
    MP_STD_RSP_DATA*  rData 
  );
  ```

- **Parameter**

  - **[sData]**

    [in] The pointer to the data structure which transmits the cycle mode to be set

    - MP_CYCLE_SEND_DATA

      [in] The data structure which transmits the cycle mode to be set

      ```c
typedef struct 
  {
    SHORT  sCycle;
    CHAR    reserved[2];
  }  MP_CYCLE_SEND_DATA;
  ```

      - **Member:** `<sCycle>`

        - **Cycle**

          | Value | Description |
          |-------|-------------|
          | 1     | Step        |
          | 2     | 1Cycle      |
          | 3     | Auto        |

  - **[rData]**

    [out] The pointer to the data structure which receives the error information

    - MP_STD_RSP_DATA

      [out] The data structure which receives the error information

      ```c
typedef struct 
  {
    USHORT   err_no;
    CHAR     reserved[2];
  }  MP_STD_RSP_DATA;
  ```

      - **Member:** `<err_no>`

        - **Error number**

          | Value   | Description                      |
          |---------|----------------------------------|
          | 0x0000  | Normal end                       |
          | 0x2060  | In error/alarm status            |
### Return Value

0  API execution processor received the command.

Return value = 0 and err_no = 0  Normal end

Return value = 0 and err_no ! = 0  Error (Check err_no.)

-1  API execution processor does not respond.

---

**NOTE**

Exclusive control by using semaphores is executed between this API and the APIs listed below. Thus, if this API is executed while one of the following APIs is already being executed, the task may be suspended to wait for the API to be completed. (waiting for semaphore acquisition)

Therefore, DO NOT use the following APIs for the task which must be performed periodically with high priority.

mpCancelError, mpResetAlarm, mpSetCycle, mpSetServoPower, mpSetMasterJob, mpSetCurJob, mpStartJob, mpHold, mpWaitForJobEnd, mpDeleteJob, mpGetJogSpeed, mpGetJogCoord, mpiMOV, mpMOVJ, mpMOVL, mpPulseMOVJ, mpPulseMOVL
mpSetServoPower

Sets the ON/OFF of the servo power.

- **Syntax**

  ```c
  LONG mpSetServoPower (  
    MP_SERVO_POWER_SEND_DATA* sData,  
    MP_STD_RSP_DATA* rData
  );
  ```

- **Parameter**

  - **[sData]**
    
    [in] The pointer to the data structure which transmits the ON/OFF of the servo power
    
    - **MP_SERVO_POWER_SEND_DATA**
      
      [in] The data structure which transmits the ON/OFF of the servo power
      
      Syntax: `typedef struct
        {
          SHORT sServoPower;
          CHAR reserved[2];
        } MP_SERVO_POWER_SEND_DATA;`
      
      Member: `<sServoPower>`
      
      Servo power
      
      | Value  | Description          |
      |--------|----------------------|
      | 1      | Servo power ON       |
      | 0      | Servo power OFF      |

  - **[rData]**
    
    [out] The pointer to the data structure which receives the error information
    
    - **MP_STD_RSP_DATA**
      
      [out] The data structure which receives the error information
      
      Syntax: `typedef struct
        {
          USHORT err_no;
          CHAR reserved[2];
        } MP_STD_RSP_DATA;`
      
      Member: `<err_no>`
      
      Error number
      
      | Value   | Description                                |
      |---------|--------------------------------------------|
      | 0x0000  | Normal end                                 |
      | 0x2060  | In error/alarm status                      |
      | 0x3450  | Failed (Unable to turn servo on)           |
**Return Value**

0  API execution processor received the command.
    Return value = 0 and err_no = 0  Normal end
    Return value = 0 and err_no ! = 0  Error (Check err_no.)

-1  API execution processor does not respond.

---

**NOTE**

Exclusive control by using semaphores is executed between this API and the APIs listed below. Thus, if this API is executed while one of the following APIs is already being executed, the task may be suspended to wait for the API to be completed. (waiting for semaphore acquisition)

Therefore, DO NOT use the following APIs for the task which must be performed periodically with high priority.

mpCancelError, mpResetAlarm, mpSetCycle, mpSetServoPower, mpSetMasterJob, mpSetCurJob, mpStartJob, mpHold, mpWaitForJobEnd, mpDeleteJob, mpGetJogSpeed, mpGetJogCoord, mpiMOV, mpMOVJ, mpMOVL, mpPulseMOVJ, mpPulseMOVL
mpSetMasterJob

Registers the specified job as the master job.

- **Syntax**
  
  ```c
  LONG  mpSetMasterJob ( 
    MP_MASTER_JOB_SEND_DATA*   sData, 
    MP_STD_RSP_DATA*           rData
  );
  ```

- **Parameter**

  - **[sData]**
    
    [in] The pointer to the data structure which transmits the master job information

    - **MP_MASTER_JOB_SEND_DATA**
      
      [in] The data structure which transmits the master job information

      Syntax:  
      ```
      #define MAX_JOB_NAME_LEN (33)
      
      typedef struct
      {
        SHORT     sTaskNo;
        CHAR      cJobName[MAX_JOB_NAME_LEN];
        CHAR      reserved[5];
      } MP_MASTER_JOB_SEND_DATA;
      ```

      Member:  
      ```
      <sTaskNo>
      Task number
      ```

      - **Value**  
      - **Description**
        
        | Value | Description |
        |-------|------------|
        | 0     | Master task |
        | 1     | Subtask 1   |
        | 2     | Subtask 2   |
        | 3     | Subtask 3   |
        | 4     | Subtask 4   |
        | 5     | Subtask 5   |

      - **<cJobName[MAX_JOB_NAME_LEN]>**
        
        Job name (up to 32 characters for a job name)
System Control API
mpSetMasterJob

[rData]
[out] The pointer to the data structure which receives the error information

- MP_STD_RSP_DATA
[out] The data structure which receives the error information

Syntax:
```c
typedef struct {
    USHORT err_no;
    CHAR reserved[2];
} MP_STD_RSP_DATA;
```

Member:  

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>Normal end</td>
</tr>
<tr>
<td>0x2060</td>
<td>In error/alarm status</td>
</tr>
<tr>
<td>0x2010</td>
<td>Robot is in operation</td>
</tr>
<tr>
<td>0x3400</td>
<td>Cannot operate MASTER JOB</td>
</tr>
<tr>
<td>0x3410</td>
<td>The JOB name is already registered in another task.</td>
</tr>
<tr>
<td>0x4040</td>
<td>Specified JOB not found</td>
</tr>
</tbody>
</table>

Return Value

0    API execution processor received the command.
Return value = 0 and err_no = 0    Normal end
Return value = 0 and err_no ! = 0    Error (Check err_no.)

-1    API execution processor does not respond.

- Exclusive control by using semaphores is executed between this API and the APIs listed below.
Thus, if this API is executed while one of the following APIs is already being executed, the task may be suspended to wait for the API to be completed. (waiting for semaphore acquisition)
Therefore, DO NOT use the following APIs for the task which must be performed periodically with high priority.

mpCancelOnError, mpResetAlarm, mpSetCycle,
mpSetServoPower, mpSetMasterJob, mpSetCurJob,
mpStartJob, mpHold, mpWaitForJobEnd, mpDeleteJob,
mpGetJogSpeed, mpGetJogCoord, mpiMOV, mpiMOVJ,
mpiMOL, mpiPulseMOVJ, mpiPulseMOV

- In case a task No. which exceeds the sub task number designated by the independent control function is specified, it is not registered as the master job.
mpSetCurJob

Sets the job name and line number to the current job of the master task.

- Syntax

```
LONG mpSetCurJob (MP_CUR_JOB_SEND_DATA* sData,
                  MP_STD_RSP_DATA* rData);
```

- Parameter

  - [sData]

    - [in] The pointer to the data structure which transmits the current job information
      - MP_CUR_JOB_SEND_DATA
        - [in] The data structure which transmits the current job information
          Syntax: `#define MAX_JOB_NAME_LEN (33)`
          ```
          typedef struct
          {
            USHORT  usJobLine;
            CHAR    cJobName[MAX_JOB_NAME_LEN];
            CHAR    reserved[5];
          } MP_CUR_JOB_SEND_DATA;
          ```
          Member:
          - `<usJobLine>`
            - Job line
          - `<cJobName[MAX_JOB_NAME_LEN]>`
            - Job name (up to 32 characters for a job name)

  - [rData]

    - [out] The pointer to the data structure which receives the error information
      - MP_STD_RSP_DATA
        - [out] The data structure which receives the error information
          Syntax: `typedef struct`
          ```
          {
            USHORT  err_no;
            CHAR    reserved[2];
          } MP_STD_RSP_DATA;
          ```
          Member:
          - `<err_no>`
            - Error number

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>Normal end</td>
</tr>
<tr>
<td>0x2060</td>
<td>In error/alarm status</td>
</tr>
<tr>
<td>0x2010</td>
<td>Robot is in operation</td>
</tr>
<tr>
<td>0x2110</td>
<td>Inaccessible data</td>
</tr>
<tr>
<td>0x4040</td>
<td>Specified JOB not found</td>
</tr>
<tr>
<td>0x5200</td>
<td>Over data range</td>
</tr>
</tbody>
</table>
Return Value

0  API execution processor received the command.
   Return value = 0 and err_no = 0       Normal end
   Return value = 0 and err_no != 0      Error (Check err_no.)

-1  API execution processor does not respond.

Exclusive control by using semaphores is executed between this API and the APIs listed below. Thus, if this API is executed while one of the following APIs is already being executed, the task may be suspended to wait for the API to be completed. (waiting for semaphore acquisition)
Therefore, DO NOT use the following APIs for the task which must be performed periodically with high priority.

mpCancelError, mpResetAlarm, mpSetCycle, mpSetServoPower, mpSetMasterJob, mpSetCurJob, mpStartJob, mpHold, mpWaitForJobEnd, mpDeleteJob, mpGetJogSpeed, mpGetJogCoord, mpiMOV, mpMOVJ, mpMOVL, mpPulseMOVJ, mpPulseMOVL
mpStartJob

Starts a job.

- **Syntax**

  ```c
  LONG  mpStartJob ( 
      MP_START_JOB_SEND_DATA*  sData, 
      MP_STD_RSP_DATA*  rData 
  );
  ```

- **Description**

  This API executes the job specified as `cJobName` from the beginning. If NULL is specified as `cJobName`, the job suspended by using the hold function, etc., resumes from where it stopped. If multiple jobs are operating before the hold function is executed, all of the suspended jobs resume.

- **Parameter**

  **[sData]**

  [in] The pointer to the data structure which transmits the job to be started

  - MP_START_JOB_SEND_DATA

    [in] The data structure which transmits the job to be started

    ```c
    #define MAX_JOB_NAME_LEN (33)
    typedef struct 
      {
        SHORT  sTaskNo;
        CHAR  cJobName[MAX_JOB_NAME_LEN];
        CHAR  reserved[5];
      } MP_START_JOB_SEND_DATA;
    ```

    **Member:**

    - `<sTaskNo>`

      Task number: Always specify the master task, “0”.

    - `<cJobName[MAX_JOB_NAME_LEN]>`

      Job name (up to 32 characters for a job name)

  **[rData]**

  [out] The pointer to the data structure which receives the error information

  - MP_STD_RSP_DATA

    [out] The data structure which receives the error information

    ```c
    typedef struct 
      {
        USHORT  err_no;
        CHAR  reserved[2];
      } MP_STD_RSP_DATA;
    ```
**System Control API**

**mpStartJob**

**Member:** `<err_no>`

**Error number**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>Normal end</td>
</tr>
<tr>
<td>0x2010</td>
<td>Robot is in operation</td>
</tr>
<tr>
<td>0x2030</td>
<td>In HOLD status (PP)</td>
</tr>
<tr>
<td>0x2040</td>
<td>In HOLD status (External)</td>
</tr>
<tr>
<td>0x2050</td>
<td>In HOLD status (Command)</td>
</tr>
<tr>
<td>0x2060</td>
<td>In error/alarm status</td>
</tr>
<tr>
<td>0x2070</td>
<td>In SERVO OFF status</td>
</tr>
<tr>
<td>0x2080</td>
<td>Wrong operation mode</td>
</tr>
<tr>
<td>0x3040</td>
<td>The home position is not registered</td>
</tr>
<tr>
<td>0x3050</td>
<td>Out of range (ABSO data)</td>
</tr>
<tr>
<td>0x3400</td>
<td>Cannot operate MASTER JOB</td>
</tr>
<tr>
<td>0x3410</td>
<td>The JOB name is already registered in another task.</td>
</tr>
<tr>
<td>0x4040</td>
<td>Specified JOB not found</td>
</tr>
</tbody>
</table>

**Return Value**

- **0** API execution processor received the command.
  - Return value = 0 and `err_no` = 0 Normal end
  - Return value = 0 and `err_no` ! = 0 Error (Check `err_no`.)

- **-1** API execution processor does not respond.

---

**NOTE**

Exclusive control by using semaphores is executed between this API and the APIs listed below. Thus, if this API is executed while one of the following APIs is already being executed, the task may be suspended to wait for the API to be completed. (waiting for semaphore acquisition)

Therefore, DO NOT use the following APIs for the task which must be performed periodically with high priority.

- mpCancelError, mpResetAlarm, mpSetCycle, mpSetServoPower, mpSetMasterJob, mpSetCurJob, mpStartJob, mpHold, mpWaitForJobEnd, mpDeleteJob, mpGetJogSpeed, mpGetJogCoord, mpiMOV, mpiMOVJ, mpiMOVL, mpPulseMOVJ, mpPulseMOVL
mpHold

Turns ON/OFF the hold function.

Syntax

```c
LONG mpHold ( 
MP_HOLD_SEND_DATA* sData, 
MP_STD_RSP_DATA* rData );
```

Parameter

**[sData]**

[in] The pointer to the data structure which transmits the ON/OFF of the hold function

- **MP_HOLD_SEND_DATA**
  [in] The data structure which transmits the ON/OFF of the hold function
  Syntax: ```
  typedef struct 
  { 
  SHORT sHold; 
  CHAR reserved[2]; 
  } MP_HOLD_SEND_DATA;
  ```
  Member: `<sHold>`
  Hold status

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hold ON</td>
</tr>
<tr>
<td>0</td>
<td>Hold OFF</td>
</tr>
</tbody>
</table>

**[rData]**

[out] The pointer to the data structure which receives the error information

- **MP_STD_RSP_DATA**
  [out] The data structure which receives the error information
  Syntax: ```
  typedef struct 
  { 
  USHORT err_no; 
  CHAR reserved[2]; 
  } MP_STD_RSP_DATA;
  ```
  Member: `<err_no>`
  Error number

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>Normal end</td>
</tr>
</tbody>
</table>
## Return Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>-1</td>
<td>Error</td>
</tr>
</tbody>
</table>

Exclusive control by using semaphores is executed between this API and the APIs listed below. Thus, if this API is executed while one of the following APIs is already being executed, the task may be suspended to wait for the API to be completed. (waiting for semaphore acquisition)

Therefore, DO NOT use the following APIs for the task which must be performed periodically with high priority:

- mpCancelError
- mpResetAlarm
- mpSetCycle
- mpSetServoPower
- mpSetMasterJob
- mpSetCurJob
- mpStartJob
- mpHold
- mpWaitForJobEnd
- mpDeleteJob
- mpGetJogSpeed
- mpGetJogCoord
- mpiMOV
- mpiMOVJ
- mpiMOVL
- mpPulseMOVJ
- mpPulseMOVL
mpWaitForJobEnd

Waits for the job completion or the expiration of the specified time.

- **Syntax**

  ```c
  LONG  mpWaitForJobEnd ( 
    MP_WAIT_JOB_SEND_DATA*  sData, 
    MP_STD_RSP_DATA*        rData 
  );
  ```

- **Parameter**

  **[sData]**
  
  [in] The pointer to the data structure which transmits the WaitJOB

  - MP_WAIT_JOB_SEND_DATA

    [in] The data structure which transmits the WaitJOB

    Syntax:  typedef struct 
            { 
              SHORT    sTaskNo; 
              SHORT    sTime; 
            } MP_WAIT_JOB_SEND_DATA;

    Member:  <sTaskNo>
             Task number

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Master task</td>
</tr>
<tr>
<td>1</td>
<td>Subtask 1</td>
</tr>
<tr>
<td>2</td>
<td>Subtask 2</td>
</tr>
<tr>
<td>3</td>
<td>Subtask 3</td>
</tr>
<tr>
<td>4</td>
<td>Subtask 4</td>
</tr>
<tr>
<td>5</td>
<td>Subtask 5</td>
</tr>
</tbody>
</table>

- **<sTime>**

  Time (unit: sec, from 0 to 32767)
10 System Control API

mpWaitForJobEnd

[rData]
[out] The pointer to the data structure which receives the error information

• MP_STD_RSP_DATA
[out] The data structure which receives the error information

Syntax:
```c
typedef struct
{
    USHORT  err_no;
    CHAR    reserved[2];
} MP_STD_RSP_DATA;
```

Member: `<err_no>`
Error number

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>Normal end</td>
</tr>
<tr>
<td>0x2030</td>
<td>In HOLD status (PP)</td>
</tr>
<tr>
<td>0x2040</td>
<td>In HOLD status (External)</td>
</tr>
<tr>
<td>0x2050</td>
<td>In HOLD status (Command)</td>
</tr>
<tr>
<td>0x2060</td>
<td>In error/alarm status</td>
</tr>
<tr>
<td>0x2070</td>
<td>In SERVO OFF status</td>
</tr>
<tr>
<td>0xFFFF</td>
<td>mpWaitForJobEnd timed out or the job execution interrupted</td>
</tr>
</tbody>
</table>

Return Value

0      API execution processor received the command.
      Return value = 0 and err_no = 0        Normal end
      Return value = 0 and err_no ! = 0     Error (Check err_no.)
-1     API execution processor does not respond.

Exclusive control by using semaphores is executed between this API and the APIs listed below. Thus, if this API is executed while one of the following APIs is already being executed, the task may be suspended to wait for the API to be completed. (waiting for semaphore acquisition)

Therefore, DO NOT use the following APIs for the task which must be performed periodically with high priority.

mpCancelError, mpResetAlarm, mpSetCycle, mpSetServoPower, mpSetMasterJob, mpSetCurJob, mpStartJob, mpHold, mpWaitForJobEnd, mpDeleteJob, mpGetJogSpeed, mpGetJogCoord, mpMOV, mpMOVJ, mpMOVL, mpPulseMOVJ, mpPulseMOVL
mpDeleteJob

Deletes the job.

### Syntax

```c
LONG mpDeleteJob ( MP_DELETE_JOB_SEND_DATA* sData, MP_STD_RSP_DATA* rData );
```

### Parameter

**[sData]**

- **[in]** The pointer to the data structure which transmits the job name to be deleted
  - • MP_DELETE_JOB_SEND_DATA
    - **[in]** The data structure which transmits the job name to be deleted
      - Syntax: 
        ```c
        #define MAX_JOB_NAME_LEN (33)
        typedef struct {
            CHAR    cJobName[MAX_JOB_NAME_LEN];
            CHAR    reserved[7];
        } MP_DELETE_JOB_SEND_DATA;
        ```
      - Member: `cJobName[MAX_JOB_NAME_LEN]` Job name (up to 32 characters for a job name)

**[rData]**

- **[out]** The pointer to the data structure which receives the error information
  - • MP_STD_RSP_DATA
    - **[out]** The data structure which receives the error information
      - Syntax: 
        ```c
        typedef struct {
            USHORT   err_no;
            CHAR     reserved[2];
        } MP_STD_RSP_DATA;
        ```
      - Member: `<err_no>` Error number

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>Normal end</td>
</tr>
<tr>
<td>0x2010</td>
<td>Robot is in operation; or cannot delete JOBS during the robot operation</td>
</tr>
<tr>
<td>0x2060</td>
<td>In error/alarm status</td>
</tr>
<tr>
<td>0x2080</td>
<td>Wrong operation mode</td>
</tr>
<tr>
<td>0x4020</td>
<td>Edit lock job</td>
</tr>
<tr>
<td>0x4040</td>
<td>Specified JOB not found</td>
</tr>
</tbody>
</table>
### mpDeleteJob

#### Return Value

0  
API execution processor received the command.
Return value = 0 and err_no = 0  Normal end
Return value = 0 and err_no ! = 0  Error (Check err_no.)

-1  
API execution processor does not respond.

---

**NOTE**

Exclusive control by using semaphores is executed between this API and the APIs listed below. Thus, if this API is executed while one of the following APIs is already being executed, the task may be suspended to wait for the API to be completed. (waiting for semaphore acquisition)
Therefore, DO NOT use the following APIs for the task which must be performed periodically with high priority.

mp CancelError, mp ResetAlarm, mp SetCycle, mp SetServoPower, mp SetMasterJob, mp SetCurJob, mp StartJob, mp Hold, mp WaitForJobEnd, mp DeleteJob, mp GetJogSpeed, mp GetJogCoord, mpiMOV, mpiMOVJ, mpiMOVL, mp PulseMOVJ, mp PulseMOVL
11 Motion Control API

mpMotStart

Starts the operation (target execution).

- **Syntax**

  ```c
  int mpMotStart
  (  
      int options /* option */
  )
  ```

- **Description**

  This routine allows the operation start, and successively executes the sent targets stored in the buffer. The target that has been executed (arrived) is deleted, then the next target is executed. When all the stored targets are executed, the manipulator stops operation. When new targets are sent and stored in the buffer, the manipulator starts operation automatically.

- **Parameter**

  **[options]**

  Specify 0.

- **Return Value**

  - >= 0  Success
  - < 0   Failure

  Negative value means failure.

<table>
<thead>
<tr>
<th>Returned Value</th>
<th>Meaning</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_MP_MOT_SVOFF</td>
<td>Servo power is OFF.</td>
<td>Turn ON the servo power.</td>
</tr>
<tr>
<td>E_MP_MOT_HOLD</td>
<td>In HOLD status.</td>
<td>Release the HOLD status.</td>
</tr>
<tr>
<td>E_MP_MOT_ALARM_OCCUR</td>
<td>In alarm or error status.</td>
<td>Release the alarm or error.</td>
</tr>
<tr>
<td>E_MP_MOT_HOME_UNSET</td>
<td>Home position is unset.</td>
<td>Register the home position.</td>
</tr>
<tr>
<td>E_MP_MOT_HOME_UNCHK</td>
<td>Position confirmation is undone.</td>
<td>Move to the second position, and confirm the position.</td>
</tr>
<tr>
<td>E_MP_MOT_START_UNREADY</td>
<td>Not in play mode.</td>
<td>Set to the play mode.</td>
</tr>
<tr>
<td>E_MP_MOT_START_COLLISION</td>
<td>During job playback.</td>
<td>Start operation after the job playback.</td>
</tr>
<tr>
<td>E_MP_MOT_GRP_DUPLICATE</td>
<td>Duplicates to the control group of the job.</td>
<td>Select job or master job.</td>
</tr>
</tbody>
</table>
mpMotStop

Stops the operation (target execution).

- **Syntax**

```c
int mpMotStop
(  
    int options /* option */
)
```

- **Description**

This routine stops the manipulator’s operation and the successive execution of targets.

- **Parameter**

  `[options]`

Specify 0.

- **Return Value**

  >= 0  Success
  < 0  Failure
mpMotTargetClear

Clears the target already sent.

■ Syntax

```c
int mpMotTargetClear
(
    CTRLG_T grp, /* target control group */
    int options /* option */
)
```

■ Description

This routine clears the target buffer of the control group that is specified in `grp`.

■ Parameter

**[grp]**

Specify the target control group. Each bit indicates the control group. Specify in bits by using the control group number retrieved by the `mpCtrlGrpId2GrpNo()`.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>d0</td>
<td>The first control group</td>
</tr>
<tr>
<td>d1</td>
<td>The second control group</td>
</tr>
<tr>
<td>d2</td>
<td>The third control group</td>
</tr>
<tr>
<td>d3</td>
<td>The fourth control group</td>
</tr>
</tbody>
</table>

**[options]**

Specify 0.

■ Return Value

```c
>= 0  Success
< 0   Failure
```
mpMotTargetSend

Sends the target.

- Syntax

```c
int mpMotTargetSend
(
    CTRLG_T grp,  /*!< target control group */
    MP_TARGET * target,  /*!< target pointer */
    int timeout  /*!< waiting time before the completion of sending */
)
```

- Description

This routine sends the target to the control group specified in `grp`.

The `target` is the head pointer of target array. It composes the array by making the first control group being the first element and the second control group being the second element,..., then sends each head pointer.

In the system with only a manipulator(R1), sends the pointer of the target to be sent.

In the system with multiple control groups, sends the head pointer of the array to be sent. This routine refers to the element that `grp` indicates.

When specify the 0x02 to `grp`, the array that `target` indicates needs to have at least two elements.

`timeout` is the waiting time(the number of ticks) before completion of sending. If the buffer is full, waits until there is a vacancy.

- Parameter

`[grp]`

Specify the target control group. Each bit indicates the control group. Specify in bits by using the control group number retrieved by the `mpCtrlGrpId2GrpNo()`.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>d0</td>
<td>The first control group</td>
</tr>
<tr>
<td>d1</td>
<td>The second control group</td>
</tr>
<tr>
<td>d2</td>
<td>The third control group</td>
</tr>
<tr>
<td>d3</td>
<td>The fourth control group</td>
</tr>
</tbody>
</table>
[**target**]

Specify the pointer that indicates the target to be sent (or the array if there are multiple control groups).

```c
typedef struct {
    int id; /* target ID. */
    MP_INTP_TYPE intp; /* interpolation type. */
    MP_POS dst; /* destination position. */
    MP_POS aux; /* passing(auxiliary) position. */
} MP_TARGET;
```

```c
typedef enum {
    MP_MOV_NOP_TYPE, /* NOP */
    MP_MOVJ_TYPE, /* link interpolation type. */
    MP_MOVL_TYPE, /* linear interpolation type. */
    MP_MOVC_TYPE, /* circular interpolation type. */
} MP_INTP_TYPE;
```

```c
typedef union {
    MP_COORD coord;
    MP_JOINT joint;
} MP_POS;
```

```c
typedef struct {
    long x, y, z; /* XYZ position(0.001[mm]) */
    long rx, ry, rz; /* RxRyRz rotation(0.0001[deg]) */
    long e1, e2;
} MP_COORD;
```

```c
typedef long MP_JOINT[MP_GRP_AXES_NUM]; /* pulse or angle(0.0001[deg]) */
```

[**timeout**]

Specify the waiting time in ticks. Able to specify "NO_WAIT" (doesn’t wait more than the set number of ticks) or "WAIT_FOREVER" (keep waiting forever).

<table>
<thead>
<tr>
<th>NO_WAIT</th>
<th>Doesn’t wait</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAIT_FOREVER</td>
<td>Keep waiting forever</td>
</tr>
</tbody>
</table>

```
### Return Value

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Meaning</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_MP_MOT_TIMEOUT</td>
<td>Exceeded the waiting time.</td>
<td>Lengthen in waiting time. Execute or clear target to enable to send targets.</td>
</tr>
<tr>
<td>E_MP_MOT_GRP_NOT_EXIST</td>
<td>Specified control group doesn’t exist.</td>
<td>Specify the correct control group.</td>
</tr>
<tr>
<td>E_MP_MOT_BASE_LACK</td>
<td>The base axis control group is lacking. The manipulator with base axis needs the target of both manipulator and the base axis.</td>
<td>Send the target to both manipulator and the base axis.</td>
</tr>
<tr>
<td>E_MP_MOT_INTP_OUTRNG</td>
<td>The interpolation command is out of range. The external axis cannot be moved by the APIs except for MOVJ(joint interpolation).</td>
<td>Specify the correct interpolation command.</td>
</tr>
<tr>
<td>E_MP_MOT_BASE_TASK_UNMATCH</td>
<td>The task number of the manipulator with base axis doesn’t accord with the task number of the base axis.</td>
<td>Specify a same task number to the manipulator and the base axis.</td>
</tr>
<tr>
<td>E_MP_MOT_MASTER_INTP_OUTRNG</td>
<td>The coordinated motion is unable. The interpolation command of the master is out of range.</td>
<td>Specify MOVJ (joint interpolation) or MOVL (linear interpolation) to the master.</td>
</tr>
<tr>
<td>E_MP_MOT_SLAVE_INTP_OUTRNG</td>
<td>The coordinated motion is unable. The interpolation command of the slave is out of range.</td>
<td>Specify APIs except for MOVJ (joint interpolation) to the slave.</td>
</tr>
<tr>
<td>E_MP_MOT_MASTER_LACK</td>
<td>The coordinated motion is unable. Targets of master and slave are required for coordinated motion.</td>
<td>Send targets to the master and the slave.</td>
</tr>
<tr>
<td>E_MP_MOT_SYNC_TASK_UNMATCH</td>
<td>The coordinated motion is unable. The task number of the master doesn’t accord with the task number of the slave.</td>
<td>Specify the same task number to the master and the slave.</td>
</tr>
<tr>
<td>E_MP_MOT_SYNC_ILLEGAL</td>
<td>Specification for coordinated motion is wrong. Multiple masters are specified to one task.</td>
<td>Specify one master to one task.</td>
</tr>
</tbody>
</table>

Negative value means failure.
mpMotTargetReceive

Receives the report of having arrived to the target position.

**Syntax**

```c
int mpMotTargetReceive
(
    int grpNo, /* control group number */
    int id, /* target ID to receive */
    int * recvId, /* received target ID */
    int timeout, /* waiting time for the completion of receiving */
    int options /* option */
)
```

**Description**

This routine receives the report of the control group specified in `grp` having arrived to the target of `id`. The received target ID is stored in the `recvId`.

Specify the waiting time before receiving to `timeout`.

If operation is interrupted because of some factor (mpMotTargetStop(), emergency stop, alarm, etc.), the waiting status is canceled, and `E_MP_NOT_INTERRUPT` is returned.

Not all the arriving history, but only the latest arriving report can be received. The following figure shows the movement of manipulator (having arrived to id=1, then id=2, and moving toward id=3). At this time, if specify id=1 and then call this routine, receiving report is not done immediately and wait for the next arriving id=1. This is because id=1 has already been arrived but the latest arrived target is id=2.

On the other hand, if specify id=2 while moving toward id=3, receives the report immediately because the id=2 is the latest arrived target.
 Parameter

**[grpNo]**
Specify the target control group.
Specify the control group number retrieved by the mpCtrlGrpId2GrpNo().

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The first control group</td>
</tr>
<tr>
<td>1</td>
<td>The second control group</td>
</tr>
<tr>
<td>2</td>
<td>The third control group</td>
</tr>
<tr>
<td>3</td>
<td>The fourth control group</td>
</tr>
</tbody>
</table>

**[id]**
Specify the target ID to receive.

**[recvId]**
Specify the destination of the received target ID.
NULL can be also specified.

**[timeout]**
Specify the waiting time in ticks. Able to specify “NO_WAIT” (doesn’t wait more than the set number of ticks) or “WAIT_FOREVER” (keep waiting forever).

- NO_WAIT: End immediately even if the report is not received
- WAIT_FOREVER: Keep waiting without timeout

**[options]**
Specify 0.
### Return Value

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Meaning</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_MP_MOT_TIMEOUT</td>
<td>Exceeded the waiting time.</td>
<td>Lengthen in waiting time. Specify the receivable target ID.</td>
</tr>
<tr>
<td>E_MP_MOT_INTERRUPT</td>
<td>Operation was interrupted.</td>
<td>Take steps not to occur the factor for interruption.</td>
</tr>
<tr>
<td>E_MP_MOT_GRP_NOT_EXIST</td>
<td>Specified control group doesn't exist.</td>
<td>Specify the correct control group.</td>
</tr>
</tbody>
</table>
mpMotSetCoord

Specifies the coordinate system of the target position.

■ Syntax

```c
int mpMotSetCoord
(
    int grpNo, /* control group number */
    MP_COORD_TYPE grp, /* target ID to receive */
    int aux /* option */
)
```

■ Description

Set the type of coordinate system to the control group specified in `grp`. The set details are reflected in the target that is sent after this routine is called.

■ Parameter

- **[grpNo]**
  Specify the target control group.
  Specify the control group number retrieved by the `mpCtrlGrpId2GrpNo()`.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The first control group</td>
</tr>
<tr>
<td>1</td>
<td>The second control group</td>
</tr>
<tr>
<td>2</td>
<td>The third control group</td>
</tr>
<tr>
<td>3</td>
<td>The fourth control group</td>
</tr>
</tbody>
</table>

- **[type]**
  Specify the type of the coordinate system.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
<th>Auxiliary Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_PULSE_TYPE</td>
<td>Pulse</td>
<td>Not used</td>
</tr>
<tr>
<td>MP_ANGLE_TYPE</td>
<td>Angle</td>
<td>Not used</td>
</tr>
<tr>
<td>MP_BASE_TYPE</td>
<td>Base coordinate system</td>
<td>Not used</td>
</tr>
<tr>
<td>MP_ROBOT_TYPE</td>
<td>Robot System</td>
<td>Not used</td>
</tr>
<tr>
<td>MP_USER_TYPE</td>
<td>User coordinate system</td>
<td>User coordinate system number 0<del>15(#01</del>#16)</td>
</tr>
</tbody>
</table>
Specify the auxiliary data of the type of coordinate system.
When specify the user coordinate system to type, specify the user coordinate system number 0 - 15(#01 -#16) to aux.

**Return Value**

- >= 0    Success
- < 0     Failure

Negative value means failure.

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Meaning</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_MP_MOT_GRP_NOT_EXIST</td>
<td>Specified control group doesn't exist.</td>
<td>Specify the correct control group.</td>
</tr>
<tr>
<td>E_MP_MOT_USER_OUTRNG</td>
<td>The specified user coordinate system number is out of range.</td>
<td>Specify the correct user coordinate system number.</td>
</tr>
</tbody>
</table>
mpMotSetSpeed

Specifies speed of the target.

- **Syntax**

```c
int mpMotSetSpeed(
    int grpNo,  /* control group number */
    MP_SPEED * spd /* speed */
)
```

- **Description**

Set the speed specified in `spd` to the control group specified in `grpNo`. `spd` includes joint speed, orbital speed, and posture speed in each member. This routine sets all these speed. If the set value is 0, the speed is not changed and keeps the previous status.

The set details are reflected in the target that is sent after this routine is called.

- **Parameter**

  - **[grpNo]**
    
    Specify the target control group.
    
    Specify the control group number retrieved by the `mpCtrlGrpId2GrpNo()`.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The first control group</td>
</tr>
<tr>
<td>1</td>
<td>The second control group</td>
</tr>
<tr>
<td>2</td>
<td>The third control group</td>
</tr>
<tr>
<td>3</td>
<td>The fourth control group</td>
</tr>
</tbody>
</table>

  - **[spd]**
    
    Specify the type of the coordinate system.

```c
typedef struct {
    long vj,  /* joint velocity(0.01[%]). */
    long v,   /* trajectory velocity(0.1[mm/sec]). */
    long vr,  /* orientation velocity(0.1[deg/sec]). */
    MP_SPEED;
}
```

<table>
<thead>
<tr>
<th>Member</th>
<th>Meaning</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_SPEED.vj</td>
<td>Axis speed</td>
<td>0.01[%]</td>
</tr>
<tr>
<td>MP_SPEED.v</td>
<td>Orbital speed</td>
<td>0.1[mm/sec]</td>
</tr>
<tr>
<td>MP_SPEED.vr</td>
<td>Posture speed</td>
<td>0.1[deg/sec]</td>
</tr>
</tbody>
</table>
Return Value

>= 0  Success
< 0   Failure

Negative value means failure.

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Meaning</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_MP_MOT_GRP_NOT_EXIST</td>
<td>Specified control group doesn't exist.</td>
<td>Specify the correct control group.</td>
</tr>
<tr>
<td>E_MP_MOT_USER_OUTRNG</td>
<td>The specified user coordinate system number is out of range.</td>
<td>Specify the correct user coordinate system number.</td>
</tr>
</tbody>
</table>
mpMotSetOrigin

Specifies the basis of the target position.

**Syntax**

```c
int mpMotSetOrigin
(
    int grpNo, /* control group number */
    int options /* option */
);
```

**Description**

Set the basis specified in `options` to the control group specified in `grpNo`. When specify MP_ABSO_VAL to `options`, base on the followings. For example, when the each axis target position of the target is 10[deg], the manipulator moves to the position of 10[deg].

<table>
<thead>
<tr>
<th>Target position</th>
<th>Base position</th>
</tr>
</thead>
<tbody>
<tr>
<td>Each axis value (pulse, angle, etc.)</td>
<td>0 pulse position of each axis</td>
</tr>
<tr>
<td></td>
<td>(The position of absolute angle 0[deg])</td>
</tr>
<tr>
<td>Cartesian value (base coordinate system, user coordinate system, etc.)</td>
<td>The defined position of coordinate system</td>
</tr>
</tbody>
</table>

When specify the MP_INC_VAL to `options`, base on the start position. For example, when the each axis target position of the target is 10[deg] and each axis angle is 5[deg], the manipulator moves to the position of 15[deg].

<table>
<thead>
<tr>
<th>Target position</th>
<th>Base position</th>
</tr>
</thead>
<tbody>
<tr>
<td>Each axis value (pulse, angle, etc.)</td>
<td>Each axis start position</td>
</tr>
<tr>
<td>Cartesian value (base coordinate system, user coordinate system, etc.)</td>
<td>Cartesian value start position</td>
</tr>
</tbody>
</table>

The set details are reflected to the sent target after this routine had been called.
11 Motion Control API

mpMotSetOrigin

- **Parameter**

  **[grpNo]**
  Specify the target control group.
  Specify the control group number retrieved by the mpCtrlGrpId2GrpNo().

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The first control group</td>
</tr>
<tr>
<td>1</td>
<td>The second control group</td>
</tr>
<tr>
<td>2</td>
<td>The third control group</td>
</tr>
<tr>
<td>3</td>
<td>The fourth control group</td>
</tr>
</tbody>
</table>

  **[options]**
  Specify the basis.
  MP_ABSO_VAL  Absolute value
  MP_INC_VAL   Incremental value

- **Return Value**

  >= 0  Success
  < 0   Failure
  Negative value means failure.

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Meaning</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_MP_MOT_GRP_NOT_EXIST</td>
<td>Specified control group doesn't exist.</td>
<td>Specify the correct control group.</td>
</tr>
</tbody>
</table>
mpMotSetTask

Specifies a task number of the target. To use this function, the independent control function (optional) is required.

- **Syntax**

  ```c
  int mpMotSetTask(
      int grpNo, /* control group number */
      int taskNo /* task number */
  )
  ```

- **Description**

  Set the task number specified in `taskNo` to the control group specified in `grpNo`. The set details are reflected to the target that is sent after this routine is called.
Parameter

[grpNo]
Specify the target control group.
Specify the control group number retrieved by the mpCtrlGrpId2GrpNo().

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The first control group</td>
</tr>
<tr>
<td>1</td>
<td>The second control group</td>
</tr>
<tr>
<td>2</td>
<td>The third control group</td>
</tr>
<tr>
<td>3</td>
<td>The fourth control group</td>
</tr>
</tbody>
</table>

[taskNo]
Specify the task number.

Return Value

>= 0  Success
< 0   Failure

Negative value means failure.

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Meaning</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_MP_MOT_GRP_NOT_EXIST</td>
<td>Specified control group doesn’t exist.</td>
<td>Specify the correct control group.</td>
</tr>
<tr>
<td>E_MP_MOT_TASK_OUTRNG</td>
<td>Specified task number is out of range.</td>
<td>Specify the correct task number.</td>
</tr>
</tbody>
</table>

CAUTION

- If specify any task number except for 0 when the independent control function (optional) is invalid, “E_MP_MOT_TASK_OUTRNG” is returned.
  To use this function, the independent control function (optional) is required separately.
mpMotSetSync

Specifies the coordinated motion of the target. To use this function, the coordinated control function (optional) is required.

■ Syntax

```c
int mpMotSetSync(
    int grpNo,  /* control group number */
    int aux,    /* coordinate system auxiliary data */
    int options /* option */
);
```

■ Description

Specify the control group, which is specified in `grpNo`, for coordinated motion. The target control group becomes a slave control group, and executes the coordinated motion on the basis of the master tool coordinates. The set details are reflected to the target that is sent after this routine.
Parameter

**[grpNo]**
Specify the slave control group.
Specify the control group number retrieved by the mpCtrlGrpId2GrpNo().

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The first control group</td>
</tr>
<tr>
<td>1</td>
<td>The second control group</td>
</tr>
<tr>
<td>2</td>
<td>The third control group</td>
</tr>
<tr>
<td>3</td>
<td>The fourth control group</td>
</tr>
</tbody>
</table>

**[aux]**
Specify the master control group.
Specify the control group number retrieved by the mpCtrlGrpId2GrpNo().

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The first control group</td>
</tr>
<tr>
<td>1</td>
<td>The second control group</td>
</tr>
<tr>
<td>2</td>
<td>The third control group</td>
</tr>
<tr>
<td>3</td>
<td>The fourth control group</td>
</tr>
</tbody>
</table>

**[options]**
Specify 0.

Return Value

- >= 0 Success
- < 0 Failure

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Meaning</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_MP_MOT_MASTER_GRP_ILLEGAL</td>
<td>Specified coordinated master is wrong. The control group is out of range or the base axis is specified to the master.</td>
<td>Specify the correct control group.</td>
</tr>
<tr>
<td>E_MP_MOT_SLAVE_GRP_ILLEGAL</td>
<td>Specified coordinated slave is wrong. The control group is out of range or other than manipulator is specified to the slave.</td>
<td>Specify the correct control group.</td>
</tr>
</tbody>
</table>

**CAUTION**

- If specify or cancel coordinated motion when the coordinated control function (optional) is invalid, ERROR is returned. To use this function, the independent control function (optional) is required separately.
mpMotResetSync

Cancels the coordinated motion specification of the target. To use this function, the coordinated control function (optional) is required.

- Syntax

```c
int mpMotResetSync(
    int grpNo /* control group */
);
```

- Description

This routine cancels the coordinated motion of the control group specified in `grpNo`. The set details are reflected to the target which is sent after this routine.

- Parameter

  `[grpNo]`

Specify the target control group. Specify the control group number retrieved by the `mpCtrlGrpId2GrpNo()`.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The first control group</td>
</tr>
<tr>
<td>1</td>
<td>The second control group</td>
</tr>
<tr>
<td>2</td>
<td>The third control group</td>
</tr>
<tr>
<td>3</td>
<td>The fourth control group</td>
</tr>
</tbody>
</table>

- Return Value

  => 0 Success
  < 0 Failure

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Meaning</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_MP_MOT_GRP_NOT_EXIST</td>
<td>Specified control group doesn’t exist.</td>
<td>Specify the correct control group.</td>
</tr>
</tbody>
</table>

- CAUTION

  • If specify or cancel coordinated motion when the coordinated control function (optional) is invalid, ERROR is returned. To use this function, the independent control function (optional) is required separately.
mpIMOV

Moves the manipulator one step to the position where the incremental value is added to the current position by linear interpolation.

■ Syntax

```c
LONG  mpIMOV ( 
    MP_IMOV_SEND_DATA*   sData,
    MP_STD_RSP_DATA*     rData
);
```

■ Parameter

[sData]

[in] The pointer to the data structure which transmits the information of incremental move

- MP_IMOV_SEND_DATA
  [in] The data structure which transmits the information of incremental move
  Syntax:  #define MAX_NO_OF_AXES (12)

```c
typedef struct  
{    ULONG   sCtrlGrp;
    LONG    lSpeed;
    SHORT   sVType;
    SHORT   sFrame;
    SHORT   sToolNo;
    CHAR    reserved[2];
    LONG    lPos[MAX_NO_OF_AXES];
} MP_IMOV_SEND_DATA;
```
### Member: `<sCtrlGrp>`

**Control group**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>R1 (Robot 1)</td>
</tr>
<tr>
<td>1</td>
<td>R2 (Robot 2)</td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>B1 (Base 1)</td>
</tr>
<tr>
<td>9</td>
<td>B2 (Base 2)</td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Not in use</td>
</tr>
<tr>
<td>13</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>S1 (Station 1)</td>
</tr>
<tr>
<td>17</td>
<td>S2 (Station 2)</td>
</tr>
<tr>
<td>18</td>
<td>S3 (Station 3)</td>
</tr>
</tbody>
</table>

### `<lSpeed>`

**Motion speed**

0.1 to SIC\*G17 (linear speed for registration) [mm/s]  
0.1 to SIC\*G25 (position angle speed) [deg/s]  
For more information on each parameter, refer to “8.2.0.4 Linear Speed for Registration” and “8.2.0.5 Position Angle Speed” of “FS100 OPERATOR’S MANUAL (RE-CSO-A043)".

### `<sVType>`

**Selection of motion speed**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Control point</td>
</tr>
<tr>
<td>1</td>
<td>Position angular</td>
</tr>
<tr>
<td>2, 3,...,17</td>
<td>User coordinate 1, 2,...,16</td>
</tr>
</tbody>
</table>

### `<sFrame>`

**Coordinate system ID**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Base coordinate</td>
</tr>
<tr>
<td>1</td>
<td>Robot coordinate</td>
</tr>
<tr>
<td>2, 3,...</td>
<td>User coordinate 1, 2,...</td>
</tr>
</tbody>
</table>

### `<sToolNo>`

**Tool number (0 - 15)**
Specified position (up to 12 arrays)

<<Robot axis>>

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lPos[0]</td>
<td>X-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>lPos[1]</td>
<td>Y-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>lPos[2]</td>
<td>Z-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>lPos[3]</td>
<td>Wrist angle Rx (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>lPos[4]</td>
<td>Wrist angle Ry (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>lPos[5]</td>
<td>Wrist angle Rz (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>lPos[6]</td>
<td>Elbow angle E (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>lPos[7]</td>
<td>8th axis pulse value (micron in the case of traveling axis)</td>
</tr>
<tr>
<td>lPos[8]</td>
<td>Not in use</td>
</tr>
<tr>
<td>lPos[9]</td>
<td>Not in use</td>
</tr>
<tr>
<td>lPos[10]</td>
<td>Not in use</td>
</tr>
</tbody>
</table>


<<External axis>>

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lPos[0]</td>
<td>Not in use</td>
</tr>
<tr>
<td>lPos[1]</td>
<td>Not in use</td>
</tr>
<tr>
<td>lPos[2]</td>
<td>Not in use</td>
</tr>
<tr>
<td>lPos[3]</td>
<td>Not in use</td>
</tr>
<tr>
<td>lPos[4]</td>
<td>Not in use</td>
</tr>
<tr>
<td>lPos[5]</td>
<td>Not in use</td>
</tr>
<tr>
<td>lPos[6]</td>
<td>1st traveling axis pulse value</td>
</tr>
<tr>
<td>lPos[7]</td>
<td>2nd traveling axis pulse value</td>
</tr>
<tr>
<td>lPos[8]</td>
<td>3rd traveling axis pulse value</td>
</tr>
<tr>
<td>lPos[9]</td>
<td>Not in use</td>
</tr>
<tr>
<td>lPos[10]</td>
<td>Not in use</td>
</tr>
</tbody>
</table>

**12 Step Move API**  
**mpIMOV**

**[rData]**  
[out] The pointer to the data structure which receives the error information  

- **MP_STD_RSP_DATA**  
  [out] The data structure which receives the error information

Syntax:  
```c
typedef struct
{
    USHORT   err_no;
    CHAR     reserved[2];
} MP_STD_RSP_DATA;
```

Member:  
- `<err_no>`  
  Error number

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>Normal end</td>
</tr>
<tr>
<td>0x2010</td>
<td>Robot is in operation</td>
</tr>
<tr>
<td>0x2030</td>
<td>In HOLD status (PP)</td>
</tr>
<tr>
<td>0x2040</td>
<td>In HOLD status (External)</td>
</tr>
<tr>
<td>0x2050</td>
<td>In HOLD status (Command)</td>
</tr>
<tr>
<td>0x2060</td>
<td>In error/alarm status</td>
</tr>
<tr>
<td>0x2070</td>
<td>In SERVO OFF status</td>
</tr>
<tr>
<td>0x2080</td>
<td>Wrong operation mode</td>
</tr>
<tr>
<td>0x3040</td>
<td>The home position is not registered</td>
</tr>
<tr>
<td>0x3050</td>
<td>Out of range (ABSO data)</td>
</tr>
<tr>
<td>0x3400</td>
<td>Cannot operate MASTER JOB</td>
</tr>
<tr>
<td>0x3410</td>
<td>The JOB name is already registered in another task.</td>
</tr>
</tbody>
</table>

**Return Value**

- **0**  
  API execution processor received the command.  
  Return value = 0 and err_no = 0  
  Normal end  
  Return value = 0 and err_no ! = 0  
  Error (Check err_no.)

- **-1**  
  API execution processor does not respond.

---

**NOTE**

Two or more move instructions such as mpIMOV, mpMOVJ, mpMOVL, mpPulseMOVJ, and mpPulseMOVL cannot be called.
If one of the manipulators is in operation by the move instruction above, the move instruction to the other manipulators cannot be issued.
If done so, "0x2060: In error/alarm status" returns in err_no.
Exclusive control by using semaphores is executed between this API and the APIs listed below. Thus, if this API is executed while one of the following APIs is already being executed, the task may be suspended to wait for the API to be completed. (waiting for semaphore acquisition)

Therefore, DO NOT use the following APIs for the task which must be performed periodically with high priority.

mpCancelError, mpResetAlarm, mpSetCycle, mpSetServoPower, mpSetMasterJob, mpSetCurJob, mpStartJob, mpHold, mpWaitForJobEnd, mpDeleteJob, mpGetJogSpeed, mpGetJogCoord, mpIMOV, mpMOVJ, mpMOVL, mpPulseMOVJ, mpPulseMOVL
mpMOVJ

Moves the manipulator one step to the target cartesian value by joint interpolation.

**Syntax**

LONG mpMOVJ (MP_MOVJ_SEND_DATA* sData, MP_STD_RSP_DATA* rData);

**Parameter**

**[sData]**

[in] The pointer to the data structure which transmits the MOVJ information

- **MP_MOVJ_SEND_DATA**
  
  [in] The data structure which transmits the MOVJ information

  Syntax: #define MAX_NO_OF_AXES (12)

  ```c
  typedef struct {
    ULONG sCtrlGrp;
    LONG lSpeed;
    SHORT sFrame;
    SHORT sConfig;
    SHORT sToolNo;
    CHAR reserved[2];
    LONG lPos[MAX_NO_OF_AXES];
  } MP_MOVJ_SEND_DATA;
  ```

  Member: <sCtrlGrp>

  Control group

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>R1 (Robot 1)</td>
</tr>
<tr>
<td>1</td>
<td>R2 (Robot 2)</td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>B1 (Base 1)</td>
</tr>
<tr>
<td>9</td>
<td>B2 (Base 2)</td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
</tr>
</tbody>
</table>

  Not in use
12 Step Move API
mpMOVJ

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>S1 (Station 1)</td>
</tr>
<tr>
<td>17</td>
<td>S2 (Station 2)</td>
</tr>
<tr>
<td>18</td>
<td>S3 (Station 3)</td>
</tr>
</tbody>
</table>

</Speed>
Motion speed (1 - 10000 representing 0.01 to 100.0%)

<sFrame>
Coordinate system ID

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Base coordinate</td>
</tr>
<tr>
<td>1</td>
<td>Robot coordinate</td>
</tr>
<tr>
<td>2, 3,...,17</td>
<td>User coordinate 1, 2,...,16</td>
</tr>
</tbody>
</table>

<sConfig>
Configuration

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D00</td>
<td>0: Front 1: Back</td>
</tr>
<tr>
<td>D01</td>
<td>0: Upper arm 1: Lower arm</td>
</tr>
<tr>
<td>D02</td>
<td>0: Flip 1: No flip</td>
</tr>
<tr>
<td>D03</td>
<td>0: R &lt; 180 1: R &gt;= 180</td>
</tr>
<tr>
<td>D04</td>
<td>0: T &lt; 180 1: T &gt;= 180</td>
</tr>
<tr>
<td>D05</td>
<td>0: S &lt; 180 1: S &gt;= 180</td>
</tr>
<tr>
<td>D06 - D15</td>
<td>Reserved by manufacturer</td>
</tr>
</tbody>
</table>

<sToolNo>
Tool number (0 - 15)

<lPos[MAX_NO_OF AXES]>
<< Robot axis >>

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lPos[0]</td>
<td>X-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>lPos[1]</td>
<td>Y-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>lPos[2]</td>
<td>Z-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>lPos[3]</td>
<td>Wrist angle Rx (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>lPos[4]</td>
<td>Wrist angle Ry (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>lPos[5]</td>
<td>Wrist angle Rz (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>lPos[6]</td>
<td>Elbow angle E (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>lPos[7]</td>
<td>8th axis pulse value (micron in the case of traveling axis)</td>
</tr>
<tr>
<td>lPos[8]</td>
<td>Not in use</td>
</tr>
<tr>
<td>lPos[9]</td>
<td>Not in use</td>
</tr>
<tr>
<td>lPos[10]</td>
<td>Not in use</td>
</tr>
</tbody>
</table>

<< External axis >>

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPos[0]</td>
<td>Not in use</td>
</tr>
<tr>
<td>IPos[1]</td>
<td>Not in use</td>
</tr>
<tr>
<td>IPos[2]</td>
<td>Not in use</td>
</tr>
<tr>
<td>IPos[3]</td>
<td>Not in use</td>
</tr>
<tr>
<td>IPos[4]</td>
<td>Not in use</td>
</tr>
<tr>
<td>IPos[5]</td>
<td>Not in use</td>
</tr>
<tr>
<td>IPos[6]</td>
<td>1st traveling axis pulse value</td>
</tr>
<tr>
<td>IPos[7]</td>
<td>2nd traveling axis pulse value</td>
</tr>
<tr>
<td>IPos[8]</td>
<td>3rd traveling axis pulse value</td>
</tr>
<tr>
<td>IPos[9]</td>
<td>Not in use</td>
</tr>
<tr>
<td>IPos[10]</td>
<td>Not in use</td>
</tr>
</tbody>
</table>

* Set “0” for data IPos[0] to IPos[5], and IPos[9] to IPos[11].

[rData]

[out] The pointer to the data structure which receives the error information

- MP_STD_RSP_DATA

Syntax:
```c
typedef struct
{
    USHORT   err_no;
    CHAR     reserved[2];
} MP_STD_RSP_DATA;
```

Member:  
<err_no>
Error number

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>Normal end</td>
</tr>
<tr>
<td>0x2010</td>
<td>Robot is in operation</td>
</tr>
<tr>
<td>0x2030</td>
<td>In HOLD status (PP)</td>
</tr>
<tr>
<td>0x2040</td>
<td>In HOLD status (External)</td>
</tr>
<tr>
<td>0x2050</td>
<td>In HOLD status (Command)</td>
</tr>
<tr>
<td>0x2060</td>
<td>In error/alarm status</td>
</tr>
<tr>
<td>0x2070</td>
<td>In SERVO OFF status</td>
</tr>
<tr>
<td>0x2080</td>
<td>Wrong operation mode</td>
</tr>
<tr>
<td>0x3040</td>
<td>The home position is not registered</td>
</tr>
<tr>
<td>0x3050</td>
<td>Out of range (ABSO data)</td>
</tr>
<tr>
<td>0x3400</td>
<td>Cannot operate MASTER JOB</td>
</tr>
<tr>
<td>0x3410</td>
<td>The JOB name is already registered in another task.</td>
</tr>
</tbody>
</table>
Return Value

0  API execution processor received the command.
   Return value = 0 and err_no = 0  Normal end
   Return value = 0 and err_no ! = 0  Error (Check err_no.)

-1  API execution processor does not respond.

• Two or more move instructions such as mpIMOV, mpMOVJ, mpMOVL, mpPulseMOVJ, and mpPulseMOVL cannot be called.
  If one of the manipulators is in operation by the move instruction above, the move instruction to the other manipulators cannot be issued.
  If done so, “0x2060: In error/alarm status” returns in err_no.

• Exclusive control by using semaphores is executed between this API and the APIs listed below.
  Thus, if this API is executed while one of the following APIs is already being executed, the task may be suspended to wait for the API to be completed. (waiting for semaphore acquisition)
  Therefore, DO NOT use the following APIs for the task which must be performed periodically with high priority.

  mpCancelError, mpResetAlarm, mpSetCycle, mpSetServoPower, mpSetMasterJob, mpSetCurJob, mpStartJob, mpHold, mpWaitForJobEnd, mpDeleteJob, mpGetJogSpeed, mpGetJogCoord, mpIMOV, mpMOVJ, mpMOVL, mpPulseMOVJ, mpPulseMOVL
mpMOVL

Moves the manipulator one step to the target cartesian value by linear interpolation.

- **Syntax**

  ```c
  LONG mpMOVL(
      MP_MOVL_SEND_DATA* sData,
      MP_STD_RSP_DATA* rData
  );
  ```

- **Parameter**

  `[sData]`

  [in] The pointer to the data structure which transmits the MOVL information

  - **MP_MOVL_SEND_DATA**

    [in] The data structure which transmits the MOVL information

    Syntax:
    ```c
    #define MAX_NO_OF_AXES (12)
    typedef struct {
        ULONG sCtrlGrp;
        LONG lSpeed;
        SHORT sVType;
        SHORT sFrame;
        SHORT sConfig;
        SHORT sToolNo;
        LONG lPos[MAX_NO_OF_AXES];
    } MP_MOVL_SEND_DATA;
    ```

    Member: `<sCtrlGrp>`
    Control group

    | Value | Description |
    |-------|-------------|
    | 0     | R1 (Robot 1) |
    | 1     | R2 (Robot 2) |
    | 2     |             |
    | 3     |             |
    | 4     |             |
    | 5     | Not in use  |
    | 6     |             |
    | 7     |             |
    | 8     | B1 (Base 1) |
    | 9     | B2 (Base 2) |
    | 10    |             |
    | 11    |             |
    | 12    |             |
    | 13    | Not in use  |
    | 14    |             |
    | 15    |             |
<lSpeed>
Motion speed
0.1 to SIC*G17 (linear speed for registration) [mm/s]
0.1 to SIC*G25 (position angle speed) [deg/s]
For more information on each parameter, refer to
“8.2.0.4 Linear Speed for Registration” and “8.2.0.5
Position Angle Speed” of “FS100 OPERATOR’S
MANUAL (RE-CSO-A043)”.
</lSpeed>

<sVType>
Selection of motion speed

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Control point</td>
</tr>
<tr>
<td>1</td>
<td>Position angular</td>
</tr>
</tbody>
</table>

<sFrame>
Coordinate system ID

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Base coordinate</td>
</tr>
<tr>
<td>1</td>
<td>Robot coordinate</td>
</tr>
<tr>
<td>2, 3,..,17</td>
<td>User coordinate 1, 2,..,16</td>
</tr>
</tbody>
</table>

<sConfig>
Configuration

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D00</td>
<td>0: Front 1: Back</td>
</tr>
<tr>
<td>D01</td>
<td>0: Upper arm 1: Lower arm</td>
</tr>
<tr>
<td>D02</td>
<td>0: Flip 1: No flip</td>
</tr>
<tr>
<td>D03</td>
<td>0: R &lt; 180 1: R &gt;= 180</td>
</tr>
<tr>
<td>D04</td>
<td>0: T &lt; 180 1: T &gt;= 180</td>
</tr>
<tr>
<td>D05</td>
<td>0: S &lt; 180 1: S &gt;= 180</td>
</tr>
<tr>
<td>D06 - D15</td>
<td>Reserved by manufacturer</td>
</tr>
</tbody>
</table>

<sToolNo>
Tool number (0 -15)
## Step Move API

**mpMOVL**

### <lPos[Max_NO_OF_AXES]>

**<< Robot axis >>**

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lPos[0]</td>
<td>X-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>lPos[1]</td>
<td>Y-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>lPos[2]</td>
<td>Z-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>lPos[3]</td>
<td>Wrist angle Rx (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>lPos[4]</td>
<td>Wrist angle Ry (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>lPos[5]</td>
<td>Wrist angle Rz (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>lPos[6]</td>
<td>Elbow angle E (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>lPos[7]</td>
<td>Reserved</td>
</tr>
<tr>
<td>lPos[8]</td>
<td>Not in use</td>
</tr>
<tr>
<td>lPos[9]</td>
<td>Not in use</td>
</tr>
<tr>
<td>lPos[10]</td>
<td>Not in use</td>
</tr>
</tbody>
</table>


**<< External axis >>**

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lPos[0]</td>
<td>Not in use</td>
</tr>
<tr>
<td>lPos[1]</td>
<td>Not in use</td>
</tr>
<tr>
<td>lPos[2]</td>
<td>Not in use</td>
</tr>
<tr>
<td>lPos[3]</td>
<td>Not in use</td>
</tr>
<tr>
<td>lPos[4]</td>
<td>Not in use</td>
</tr>
<tr>
<td>lPos[5]</td>
<td>Not in use</td>
</tr>
<tr>
<td>lPos[6]</td>
<td>1st traveling axis pulse value</td>
</tr>
<tr>
<td>lPos[7]</td>
<td>2nd traveling axis pulse value</td>
</tr>
<tr>
<td>lPos[8]</td>
<td>3rd traveling axis pulse value</td>
</tr>
<tr>
<td>lPos[9]</td>
<td>Not in use</td>
</tr>
<tr>
<td>lPos[10]</td>
<td>Not in use</td>
</tr>
</tbody>
</table>


### [rData]

[out] The pointer to the data structure which receives the error information

- **MP_STD_RSP_DATA**
  - [out] The data structure which receives the error information

```c
typedef struct
{
    USHORT   err_no;
    CHAR     reserved[2];
} MP_STD_RSP_DATA;
```

Member: `<err_no>`

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>Normal end</td>
</tr>
<tr>
<td>0x2010</td>
<td>Robot is in operation</td>
</tr>
<tr>
<td>0x2030</td>
<td>In HOLD status (PP)</td>
</tr>
<tr>
<td>0x2040</td>
<td>In HOLD status (External)</td>
</tr>
<tr>
<td>0x2050</td>
<td>In HOLD status (Command)</td>
</tr>
<tr>
<td>0x2060</td>
<td>In error/alarm status</td>
</tr>
<tr>
<td>0x2070</td>
<td>In SERVO OFF status</td>
</tr>
<tr>
<td>0x2080</td>
<td>Wrong operation mode</td>
</tr>
<tr>
<td>0x3040</td>
<td>The home position is not registered</td>
</tr>
<tr>
<td>0x3050</td>
<td>Out of range (ABSO data)</td>
</tr>
<tr>
<td>0x3400</td>
<td>Cannot operate MASTER JOB</td>
</tr>
<tr>
<td>0x3410</td>
<td>The JOB name is already registered in another task.</td>
</tr>
</tbody>
</table>

**Return Value**

-0 API execution processor received the command.
  - Return value = 0 and err_no = 0   Normal end
  - Return value = 0 and err_no ≠ 0  Error (Check err_no.)

-1 API execution processor does not respond.

• Two or more move instructions such as mpIMOV, mpMOVJ, mpMOVL, mpPulseMOVJ, and mpPulseMOVL cannot be called.
  - If one of the manipulators is in operation by the move instruction above, the move instruction to the other manipulators cannot be issued.
  - If done so, “0x2060: In error/alarm status” returns in err_no.

• Exclusive control by using semaphores is executed between this API and the APIs listed below.
  - Thus, if this API is executed while one of the following APIs is already being executed, the task may be suspended to wait for the API to be completed. (waiting for semaphore acquisition)
  - Therefore, DO NOT use the following APIs for the task which must be performed periodically with high priority.

mpCancelError, mpResetAlarm, mpSetCycle, mpSetServoPower, mpSetMasterJob, mpSetCurJob, mpStartJob, mpHold, mpWaitForJobEnd, mpDeleteJob, mpGetJogSpeed, mpGetJogCoord, mpIMOV, mpMOVJ, mpMOVL, mpPulseMOVJ, mpPulseMOVL
mpPulseMOVJ

Moves the manipulator one step to the target pulse value by joint interpolation.

- **Syntax**

  ```c
  LONG  mpPulseMOVJ ( 
    MP_PMOVJ_SEND_DATA*  sData, 
    MP_STD_RSP_DATA*    rData 
  );
  ```

- **Parameter**

  [sData]

  [in] The pointer to the data structure which transmits the PMOVJ information

- **Syntax**

  ```c
  #define MAX_NO_OF_AXES (12)
  typedef struct
  {
    ULONG     sCtrlGrp;
    LONG      lSpeed;
    SHORT     sToolNo;
    CHAR      reserved0[2];
    LONG      lPos[MAX_NO_OF_AXES];
    CHAR      reserved1[4];
  } MP_PMOVJ_SEND_DATA;
  ```

  **Member:** `<sCtrlGrp>`

  **Control group**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>R1 (Robot 1)</td>
</tr>
<tr>
<td>1</td>
<td>R2 (Robot 2)</td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Not in use</td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>B1 (Base 1)</td>
</tr>
<tr>
<td>9</td>
<td>B2 (Base 2)</td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Not in use</td>
</tr>
</tbody>
</table>
### Step Move API

**mpPulseMOVJ**

- **<lSpeed>**
  - Motion speed (1 - 10000 representing 0.01 to 100.0 %)
- **<sToolNo>**
  - Tool number (0 - 15)
- **<lPos[MAX_NO_OF_AXES]>**
  - **<< Robot axis >>**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>S1 (Station 1)</td>
</tr>
<tr>
<td>17</td>
<td>S2 (Station 2)</td>
</tr>
<tr>
<td>18</td>
<td>S3 (Station 3)</td>
</tr>
</tbody>
</table>


- **<< External axis >>**

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lPos[0]</td>
<td>1st external axis (S) pulse value</td>
</tr>
<tr>
<td>lPos[1]</td>
<td>2nd external axis (L) pulse value</td>
</tr>
<tr>
<td>lPos[2]</td>
<td>3rd external axis (U) pulse value</td>
</tr>
<tr>
<td>lPos[3]</td>
<td>4th external axis (R) pulse value</td>
</tr>
<tr>
<td>lPos[4]</td>
<td>5th external axis (B) pulse value</td>
</tr>
<tr>
<td>lPos[5]</td>
<td>6th external axis (T) pulse value</td>
</tr>
<tr>
<td>lPos[6]</td>
<td>7th external axis (E) pulse value</td>
</tr>
<tr>
<td>lPos[7]</td>
<td>8th external axis pulse value</td>
</tr>
<tr>
<td>lPos[8]</td>
<td>Not in use</td>
</tr>
<tr>
<td>lPos[9]</td>
<td>Not in use</td>
</tr>
<tr>
<td>lPos[10]</td>
<td>Not in use</td>
</tr>
</tbody>
</table>

* Set "0" for data lPos[0] to lPos[7].

**[rData]**

**[out]** The pointer to the data structure which receives the error information

- **MP_STD_RSP_DATA**

**[out]** The data structure which receives the error information
Step Move API

mpPulseMOVJ

Syntax:
```c
typedef struct
{
    USHORT    err_no;
    CHAR       reserved[2];
} MP_STD_RSP_DATA;
```

Member: `<err_no>`

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>Normal end</td>
</tr>
<tr>
<td>0x2010</td>
<td>Robot is in operation</td>
</tr>
<tr>
<td>0x2030</td>
<td>In HOLD status (PP)</td>
</tr>
<tr>
<td>0x2040</td>
<td>In HOLD status (External)</td>
</tr>
<tr>
<td>0x2050</td>
<td>In HOLD status (Command)</td>
</tr>
<tr>
<td>0x2060</td>
<td>In error/alarm status</td>
</tr>
<tr>
<td>0x2070</td>
<td>In SERVO OFF status</td>
</tr>
<tr>
<td>0x2080</td>
<td>Wrong operation mode</td>
</tr>
<tr>
<td>0x3040</td>
<td>The home position is not registered</td>
</tr>
<tr>
<td>0x3050</td>
<td>Out of range (ABSO data)</td>
</tr>
<tr>
<td>0x3400</td>
<td>Cannot operate MASTER JOB</td>
</tr>
<tr>
<td>0x3410</td>
<td>The JOB name is already registered in another task.</td>
</tr>
</tbody>
</table>

Return Value

- **0**: API execution processor received the command.
  - Return value = 0 and `err_no = 0` Normal end
  - Return value = 0 and `err_no ! = 0` Error (Check `err_no`.)

- **-1**: API execution processor does not respond.

- **NOTE**
  - Two or more move instructions such as `mpIMOV`, `mpMOVJ`, `mpMOVL`, `mpPulseMOVJ`, and `mpPulseMOVL` cannot be called.
  - If one of the manipulators is in operation by the move instruction above, the move instruction to the other manipulators cannot be issued.
  - If done so, “0x2060: In error/alarm status” returns in `err_no`.

  - Exclusive control by using semaphores is executed between this API and the APIs listed below.
  - Thus, if this API is executed while one of the following APIs is already being executed, the task may be suspended to wait for the API to be completed. (waiting for semaphore acquisition)
  - Therefore, DO NOT use the following APIs for the task which must be performed periodically with high priority.

  ```c
  mpCancelError, mpResetAlarm, mpSetCycle,
  mpSetServoPower, mpSetMasterJob, mpSetCurJob,
  mpStartJob, mpHold, mpWaitForJobEnd, mpDeleteJob,
  mpGetJogSpeed, mpGetJogCoord, mpIMOV, mpMOVJ,
  mpMOVL, mpPulseMOVJ, mpPulseMOVL
  ```
mpPulseMOVL

Moves the manipulator one step to the target pulse value by linear interpolation.

- **Syntax**

```c
LONG mpPulseMOVL( 
    MP_PMOVL_SEND_DATA* sData, 
    MP_STD_RSP_DATA* rData 
);
```

- **Parameter**

  - `[sData]`

  [in] The pointer to the data structure which transmits the PMOVL information

  - **MP_PMOVL_SEND_DATA**

  [in] The data structure which transmits the PMOVL information

  Syntax:  

  ```c
  #define MAX_NO_OF_AXES (12)
  typedef struct {
    ULONG sCtrlGrp;
    LONG lSpeed;
    SHORT sVType;
    SHORT sToolNo;
    LONG lPos[MAX_NO_OF_AXES];
    CHAR reserved[4];
  } MP_PMOVL_SEND_DATA;
  ```

  Member: `<sCtrlGrp>`

  Control group

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>R1 (Robot 1)</td>
</tr>
<tr>
<td>1</td>
<td>R2 (Robot 2)</td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Not in use</td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>B1 (Base 1)</td>
</tr>
<tr>
<td>9</td>
<td>B2 (Base 2)</td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Not in use</td>
</tr>
<tr>
<td>13</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
</tr>
</tbody>
</table>
**Selection of motion speed**

**Value** | **Description**
---|---
16 | S1 (Station 1)
17 | S2 (Station 2)
18 | S3 (Station 3)

**Motion speed**

0.1 to SIC*G17 (linear speed for registration) [mm/s]
0.1 to SIC*G25 (position angle speed) [deg/s]

For more information on each parameter, refer to section 8.2.0.4 “Linear Speed for Registration” and section “8.2.0.5 Position Angle Speed” of “FS100 OPERATOR’S MANUAL (RE-CSO-A043)”.

**Tool number (0 - 15)**

**Array**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Control point</td>
</tr>
<tr>
<td>1</td>
<td>Position angular</td>
</tr>
</tbody>
</table>

**Array**

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPos[0]</td>
<td>1st axis (S) pulse value</td>
</tr>
<tr>
<td>IPos[1]</td>
<td>2nd axis (L) pulse value</td>
</tr>
<tr>
<td>IPos[2]</td>
<td>3rd axis (U) pulse value</td>
</tr>
<tr>
<td>IPos[3]</td>
<td>4th axis (R) pulse value</td>
</tr>
<tr>
<td>IPos[4]</td>
<td>5th axis (B) pulse value</td>
</tr>
<tr>
<td>IPos[5]</td>
<td>6th axis (T) pulse value</td>
</tr>
<tr>
<td>IPos[6]</td>
<td>7th axis (E) pulse value</td>
</tr>
<tr>
<td>IPos[7]</td>
<td>8th axis pulse value</td>
</tr>
<tr>
<td>IPos[8]</td>
<td>Not in use</td>
</tr>
<tr>
<td>IPos[9]</td>
<td>Not in use</td>
</tr>
<tr>
<td>IPos[10]</td>
<td>Not in use</td>
</tr>
</tbody>
</table>

<< External axis >>

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPos[0]</td>
<td>Not in use</td>
</tr>
<tr>
<td>IPos[1]</td>
<td>Not in use</td>
</tr>
<tr>
<td>IPos[2]</td>
<td>Not in use</td>
</tr>
<tr>
<td>IPos[3]</td>
<td>Not in use</td>
</tr>
<tr>
<td>IPos[4]</td>
<td>Not in use</td>
</tr>
<tr>
<td>IPos[5]</td>
<td>Not in use</td>
</tr>
<tr>
<td>IPos[6]</td>
<td>Not in use</td>
</tr>
<tr>
<td>IPos[7]</td>
<td>Not in use</td>
</tr>
<tr>
<td>IPos[8]</td>
<td>1st external axis pulse value</td>
</tr>
<tr>
<td>IPos[9]</td>
<td>2nd external axis pulse value</td>
</tr>
<tr>
<td>IPos[10]</td>
<td>3rd external axis pulse value</td>
</tr>
</tbody>
</table>

* Set "0" for data IPos[0] to IPos[7].

[rData]

[out] The pointer to the data structure which receives the error information

- MP_STD_RSP_DATA
  [out] The data structure which receives the error information

  Syntax: typedef struct
  {
    USHORT   err_no;
    CHAR      reserved[2];
  } MP_STD_RSP_DATA;

  Member: <err_no>
  Error number

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>Normal end</td>
</tr>
<tr>
<td>0x2010</td>
<td>Robot is in operation</td>
</tr>
<tr>
<td>0x2030</td>
<td>In HOLD status (PP)</td>
</tr>
<tr>
<td>0x2040</td>
<td>In HOLD status (External)</td>
</tr>
<tr>
<td>0x2050</td>
<td>In HOLD status (Command)</td>
</tr>
<tr>
<td>0x2060</td>
<td>In error/alarm status</td>
</tr>
<tr>
<td>0x2070</td>
<td>In SERVO OFF status</td>
</tr>
<tr>
<td>0x2080</td>
<td>Wrong operation mode</td>
</tr>
<tr>
<td>0x3040</td>
<td>The home position is not registered</td>
</tr>
<tr>
<td>0x3050</td>
<td>Out of range (ABSO data)</td>
</tr>
<tr>
<td>0x3400</td>
<td>Cannot operate MASTER JOB</td>
</tr>
<tr>
<td>0x3410</td>
<td>The JOB name is already registered in another task.</td>
</tr>
</tbody>
</table>
■ Return Value

0  API execution processor received the command.
   Return value = 0 and err_no = 0     Normal end
   Return value = 0 and err_no != 0    Error (Check err_no.)

-1  API execution processor does not respond.

• Two or more move instructions such as mpIMOV, mpMOVJ, mpMOVL, mpPulseMOVJ, and mpPulseMOVL cannot be called.
   If one of the manipulators is in operation by the move instruction above, the move instruction to the other manipulators cannot be issued.
   If done so, “0x2060: In error/alarm status” returns in err_no.

• Exclusive control by using semaphores is executed between this API and the APIs listed below.
   Thus, if this API is executed while one of the following APIs is already being executed, the task may be suspended to wait for the API to be completed. (waiting for semaphore acquisition)
   Therefore, DO NOT use the following APIs for the task which must be performed periodically with high priority.

   mpCancelError, mpResetAlarm, mpSetCycle, mpSetServoPower, mpSetMasterJob, mpSetCurJob, mpStartJob, mpHold, mpWaitForJobEnd, mpDeleteJob, mpGetJogSpeed, mpGetJogCoord, mpIMOV, mpMOVJ, mpMOVL, mpPulseMOVJ, mpPulseMOVL
mpReceiveSkillCommand

Receives a sensor command sent by executing SKILLSND instruction.

- **Syntax**

  ```c
  STATUS mpReceiveSkillCommand
  (  
     int sl_id, /* Specifies communication area with the system*/
     SYS2MP_SENS_MSG *msg_p /* Structure of sensor command data */
  )
  ```

- **Description**

  This API receives the command (sensor command), which is sent from the system when SKILLSND instruction is executed or when the JOB execution after that is suspended, from the command area specified by `sl_id`. The character string specified by SKILLSND instruction is sent to SYS2MP_SENS_MSG structure specified by `msg_p`, so it is necessary to describe the process to distinguish its content.

  Note that when an operation which interferes with the JOB execution (e.g. jog operation, JOB cursor move, and JOB selection) is performed after suspending the JOB, and then the JOB is started again, a sensor command is sent as the sub command “MP_SKILL_END”. After this command is sent, the suspended interpolation motion cannot be continued even by restarting, so stop the path correction process and initialize the path correction information.

- **Parameter**

  `[sl_id]`

  Specifies the communication area with the system.
  The following values can be specified.
  Make sure to specify MP_SL_ID1 for the robot 1 (R1), and MP_SL_ID2 for the robot 2 (R2).

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_SL_ID1</td>
<td>Communication area 1</td>
</tr>
<tr>
<td>MP_SL_ID2</td>
<td>Communication area 2</td>
</tr>
</tbody>
</table>
[msg_p]
Structure receiving the message from the system

typedef struct
{
    int main_comm;  // main command
    int sub_comm; // sub command
    int exe_tsk; // task number of job in execution (0-15)
    int exe_apl; // application number of execution control group
    char cmd[256]; // text command data sent by SKILLSND
    int usr_opt;   // for future addition (reserved)
} SYS2MP_SENS_MSG;

To main_comm, one of the following values is sent from the system.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_SKILL_COMM</td>
<td>Skill command</td>
</tr>
<tr>
<td>Other than 0</td>
<td>Reserved by the system</td>
</tr>
<tr>
<td></td>
<td>Do not perform the process to this command.</td>
</tr>
</tbody>
</table>

To sub_comm, one of the following values is sent from the system.

<table>
<thead>
<tr>
<th>main_comm value</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_SKILL_COMM</td>
<td>MP_SKILL_SEND</td>
<td>SKILLSEND</td>
</tr>
<tr>
<td>MP_SKILL_END</td>
<td>2</td>
<td>SKILL command force-quit</td>
</tr>
</tbody>
</table>

SYS2MP_SENS_MSG Structure

- main_command
  1. MP_SKILL_COMM
     This is sent when the SKILLSND command is executed or force-quit. Refer to sub_command to check which is sent.

  2. Other than 0
     This value is reserved by the system. Do not perform the process even if this is sent.

- sub_command
  1. MP_SKILL_SEND
     This is sent when the SKILLSND command is executed. After receiving this, the character string which is set as the tag of the SKILLSND command will be stored in comm_link. Refer to this and perform the process according to the contents of the character string.
(2) MP_SKILL_END

After completing MP_SKILL_SEND command process, if one of the operations listed below which makes the process unable to continue is performed and then the JOB execution is started, this command is sent.

- Job selection
- Job or file edit operation (including changing the cursor location of the job)
- Jog operation
- Step, [FWD], or [BWD] switching

(For example, when the start operation is performed after moving the cursor location of the job after switching the mode from the play mode to the teach mode during playback after executing the MP_SKILL_SEND command.)

After receiving the MP_SKILL_END command, stop the processes performed in real-time such as the path correction process.

- exe_tsk
  The local task of the job to which SKILLSND is executed is stored.

- exe_apl
  The application number of the control group (the application number used to refer to the specific I/O) to which SKILLSND is executed is stored.

- cmd
  The pointer of the command (character string) specified by the SKILLSND command is stored.

- usr_opt
  This is provided for future addition (reserved).

Be sure to pass the content of msg_p to mpEndSkillCommandProcess. If the content of another member is modified, the completion of command receiving process may not be properly notified to the system.

### Return Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>ERROR</td>
<td>Error</td>
</tr>
</tbody>
</table>

**NOTE**

- SKILLSND instruction can be executed only in the JOB in which the robot is defined as the control group. If SKILLSND instruction is executed from the JOB in which the robot is not defined (with no control group), the alarm 4475: “JOB without robot axis cannot be executed” occurs.

- When SKILLSND instruction is executed from the JOB with the control group of R1+R2 (robot 1 and robot 2), the communication area of MP_SL_ID1 is used. Thus, specify “MP_SL_ID1” as the parameter sl_id of mpReceiveSkillCommand().
mpEndSkillCommandProcess

After receiving the sensor command, this API notifies the system that the command process in the application is completed.

- Syntax

```c
void mpEndSkillCommandProcess
(int sl_id, /* Specifies communication area with the system */
SYS2MP_SENS_MSG *msg_p /* Structure of sensor command data */)
```

- Description

This function is called when the task which received a command completes preparations for path correction process according to the content of the command. The system waits without executing SKILLSND instruction until it receives the notification by this function. Upon receiving the notification by this function, the system completes SKILLSND instruction and proceed to the next instruction. With this mechanism, it is possible to prevent the move instruction under path correction or speed change from starting before the application is ready.

- Parameter

**[sl_id]**

Specifies the communication area with the system. The following values can be specified. Make sure to specify MP_SL_ID1 for the robot 1 (R1), and MP_SL_ID2 for the robot 2 (R2).

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_SL_ID1</td>
<td>Communication area 1</td>
</tr>
<tr>
<td>MP_SL_ID2</td>
<td>Communication area 2</td>
</tr>
</tbody>
</table>

**[msg_p]**

Structure receiving the message from the system

typedef struct
{  
    int main_comm;      // main command
    int sub_comm;       // sub command
    int exe_tsk;        // task number of job in execution (0-15)
    int exe_api;        // application number of execution control group
    char cmd[256];      // text command data sent by SKILLSND
    int usr_opt;        // for future addition (reserved)
} SYS2MP_SENS_MSG;
Sample Code

The following is the sample of process to receive the command from the system by using mpReceiveSkillCommand() and mpEndSkillCommandProcess().

```c
void sensCommRcvTask(void)
{
    SYS2MP_SENS_MSG msg;
    memset(&msg, CLEAR, sizeof(SYS2MP_SENS_MSG));

    FOREVER
    {
        // Notification of command process completion. SKILLSND instruction completes the process, and executes the next instruction.
        mpEndSkillCommandProcess (MP_SL_ID1, &msg);

        // Waits for command from the system.
        if (mpReceiveSkillCommand (MP_SL_ID1, &msg) == ERROR)
        {
            printf("mpReceiveSensComm Error\n\r");
            break;
        }
        else
        {
            // Command process is described in the following.
        }
    }
}
```
In the programmer's manual, it is described that mpEndSkillCommandProcess() and mpReceiveSkillCommand() are combined as receiveSkillCommand() and called as follows:

```c
void sensCommandRcvTask(void)
{
    SYS2MP_SENS_MSG msg;
    int rc;
    memset(&msg, CLEAR, sizeof(SYS2MP_SENS_MSG));
    FOREVER
    {
        if (receiveSkillCommand (&msg) == ERROR)
            break;
        :
    }
}
```

// Process to receive command from SKILLSND
STATUS receiveSkillCommand (SYS2MP_SENS_MSG *msg)
{
    // Returns as command-completed response. Execution proceeds to the next step of SKILLSND.
    int rc;
    mpEndSkillCommandProcess (MP_SL_ID1, msg);
    // Receives command sent by SKILLSND.
    rc= mpReceiveSkillCommand(MP_SL_ID1, msg) ;
    return(rc);
}

■ Return Value

None
mpMeiGetJobExecTask

Retrieves the task (local task) number of the JOB in execution.

- **Syntax**

  ```c
  int mpMeiGetJobExecTask
  (  
    int sl_id, /* Specifies communication area with the system */
    EXEJT_T *dst_p /* Pointer to data structure returned by the function */
  )
  ```

- **Description**

  This routine returns the task number (master task and sub 1 task to sub 15 task) of the current JOB in execution as 32-bit status (d0 to d15: ignoring higher 16-bit).

  The task executing a job in the independent control function may be called "local task".

  This word is used descriptively to avoid confusion (conflict) with the application task that the operating system executes, but both words ("task" and "local task") are synonymous.

- **Parameter**

  **[sl_id]**

  Specifies the communication area with the system.

  The following values can be specified.

  Make sure to specify MP_SL_ID1 for the robot 1 (R1), and MP_SL_ID2 for the robot 2 (R2).

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_SL_ID1</td>
<td>Communication area 1</td>
</tr>
<tr>
<td>MP_SL_ID2</td>
<td>Communication area 2</td>
</tr>
</tbody>
</table>

  **[dst_p]**

  The task (local task) executing JOB.

- **Return Value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>ERROR</td>
<td>Error</td>
</tr>
</tbody>
</table>
Retrieves interpolation type of the move instruction of JOB in execution.

**Syntax**

```c
int mpMeiGetInterpolation
(int sl_id, /* Specifies communication area with the system */
int exe_tsk /* Task number of JOB in execution */ )
```

**Description**

This routine returns the interpolation of the move instruction executed in the task (local task) specified by `exe_tsk`.

**Parameter**

`[sl_id]`

Specifies the communication area with the system. The following values can be specified. Make sure to specify MP_SL_ID1 for the robot 1 (R1), and MP_SL_ID2 for the robot 2 (R2).

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_SL_ID1</td>
<td>Communication area 1</td>
</tr>
<tr>
<td>MP_SL_ID2</td>
<td>Communication area 2</td>
</tr>
</tbody>
</table>

`[exe_tsk]`

Specified task (local task)

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Master task</td>
</tr>
<tr>
<td>1</td>
<td>Sub task 1</td>
</tr>
<tr>
<td>2</td>
<td>Sub task 2</td>
</tr>
<tr>
<td>3</td>
<td>Sub task 3</td>
</tr>
<tr>
<td>4</td>
<td>Sub task 4</td>
</tr>
<tr>
<td>5</td>
<td>Sub task 5</td>
</tr>
</tbody>
</table>

**Return Value**

One of the following values Normal end

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_MOVJ_COM</td>
<td>0</td>
</tr>
<tr>
<td>MP_MOVL_COM</td>
<td>1</td>
</tr>
<tr>
<td>MP_MOVC_COM</td>
<td>2</td>
</tr>
<tr>
<td>MP_EIMOVL_COM</td>
<td>3</td>
</tr>
<tr>
<td>MP_EIMOVC_COM</td>
<td>4</td>
</tr>
<tr>
<td>MP_MOVS_COM</td>
<td>5</td>
</tr>
</tbody>
</table>

-1 Error
mpMeiGetExecControlGroup

Retrieves the control group of the path of the move instruction in execution (instruction in interpolation process).

- **Syntax**

  ```c
  int mpMeiGetExecControlGroup
  (  
  int   sl_id,       /* Specifies communication area with the system */
  int   exe_tsk    /* Task number of JOB in execution */
  CTRLG_T *ctrl_grp /* Pointer to data structure returned by the function */
  )
  ```

- **Description**

  This routine returns the control group information executed in the task (local task) specified by `exe_tsk`.

- **Parameter**

  - **[sl_id]**

    Specifies the communication area with the system. The following values can be specified. Make sure to specify MP_SL_ID1 for the robot 1 (R1), and MP_SL_ID2 for the robot 2 (R2).

    | Value     | Description          |
    |-----------|----------------------|
    | MP_SL_ID1 | Communication area 1 |
    | MP_SL_ID2 | Communication area 2 |

  - **[exe_tsk]**

    Specified task (local task)

    | Value | Description |
    |-------|-------------|
    | 0     | Master task |
    | 1     | Sub task 1  |
    | 2     | Sub task 2  |
    | 3     | Sub task 3  |
    | 4     | Sub task 4  |
    | 5     | Sub task 5  |
[ctrl_grp]

Control group
Specify the target group in bits.
Retrieve the control group number of the target group by
mpCtrlGrpId2GrpNo().

- Return Value
  - 0  Normal end
  - -1  Error
mpMeiPutCorrPath

Sets the amount of path correction to the path in interpolation process.

- **Syntax**

  ```c
  int mpMeiPutCorrPath
  (   
   int   sl_id,   /* Specifies communication area with the system */
   MP_POS_DATA   *src_p,   /* Pointer of structure to set path correction data */
  )
  ```

- **Description**

  This routine sets the amount of path correction to the robot of the control group indicated by src_p->ctrl_grp. The specified correction amount is calculated as the path correction amount, and the corrected value is added by the system. For example, let the interpolation cycle be 4 msec and the correction amount be 1 mm in Y-direction per interpolation control cycle, and then the path is corrected 250 mm in one second.

- **Parameter**

  - **[sl_id]**
    
    Specifies the communication area with the system. The following values can be specified. Make sure to specify MP_SL_ID1 for the robot 1 (R1), and MP_SL_ID2 for the robot 2 (R2).

    | Value       | Description       |
    |-------------|-------------------|
    | MP_SL_ID1   | Communication area 1 |
    | MP_SL_ID2   | Communication area 2 |

  - **[src_p]**
    
    Path correction information

    ```c
typedef struct {
CTRLG_T        ctrl_grp;    /* Control group to perform path correction */
MP_GRP_POS_INFO grp_pos_info[MP_GRP_NUM];
} MP_POS_DATA;
```
13 Sensor Control API

mpMeiPutCorrPath

MP_GRP_AXES_NUM: Maximum number of control axes of control group 8
MP_GRP_NUM: Maximum number of control groups 32

MP_POS_TAG.data[0]: Axis configuration information

```
+----+----+----+----+----+----+
| d7 | d6 | d5 | d4 | d3 | d2 | d1 |
+----+----+----+----+----+----+
| 1st Axis |
+----+----+----+----+----+----+
| 8th Axis |
```

MP_POS_TAG.data[2]: Not in use

MP_POS_TAG.data[3]: Data type

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_CORR_BF_DTYPE</td>
<td>MP_FBASE_TYPE</td>
</tr>
<tr>
<td>MP_CORR_RF_DTYPE</td>
<td>MP_FROBOT_TYPE</td>
</tr>
<tr>
<td>MP_CORR_TF_DTYPE</td>
<td>MP_FTOOL_TYPE</td>
</tr>
<tr>
<td>MP_CORR_UF_DTYPE</td>
<td>MP_FUSER_TYPE</td>
</tr>
</tbody>
</table>

MP_POS_TAG.data[4]: User coordinate number (0 to 63: valid only for user coordinate)

MP_POS_TAG.data[1], [5] to [7]: For future extensions

MP_GRP_POS_INFO pos

<table>
<thead>
<tr>
<th>Array</th>
<th>Pulse</th>
<th>Cartesian</th>
</tr>
</thead>
<tbody>
<tr>
<td>pos [0]</td>
<td>1st axis (S) pulse value</td>
<td>X-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>pos [1]</td>
<td>2nd axis (L) pulse value</td>
<td>Y-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>pos [2]</td>
<td>3rd axis (U) pulse value</td>
<td>Z-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>pos [3]</td>
<td>4th axis (R) pulse value</td>
<td>Wrist angle Rx (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>pos [4]</td>
<td>5th axis (B) pulse value</td>
<td>Wrist angle Ry (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>pos [5]</td>
<td>6th axis (T) pulse value</td>
<td>Wrist angle Rz (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>pos [6]</td>
<td>7th axis (E) pulse value</td>
<td>angle Re (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>pos [7]</td>
<td>8th axis pulse value</td>
<td>8th axis pulse value (micron in the case of traveling axis)</td>
</tr>
</tbody>
</table>

- **Return Value**

  0 Normal end

  One of the following negative values Error

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_MP_ARG_OBJ_ID (-1)</td>
<td>Incorrect sl_id</td>
</tr>
<tr>
<td>E_MP_CTRL_GRP (-3)</td>
<td>Incorrect control group</td>
</tr>
<tr>
<td>E_MP_AXES_CONFIG (-4)</td>
<td>Incorrect axis configuration information</td>
</tr>
<tr>
<td>E_MP_TOOL_NO (-5)</td>
<td>Incorrect tool number</td>
</tr>
<tr>
<td>E_MP_POS_DTYPE (-6)</td>
<td>Incorrect data type</td>
</tr>
<tr>
<td>E_MP_USER_FRAME (-7)</td>
<td>Incorrect user coordinate setting</td>
</tr>
</tbody>
</table>
Sample Code

The following is the sample code for path correction. When correct_req is set by the command receiving task, the path is corrected 1 mm in Y-direction per interpolation control cycle. (For the sample including the command receiving task, refer to the programmer's manual.)

Although the path correction process is FOREVER (infinite loop), mpClkAnnounce (MP_INTERPOLATION_CLK) is described at the starting of the loop so that the path correction process synchronizes with the interpolation control cycle.

```c
int correct_req;
void pathCorrectionTask(void)
{
    MP_POS_DATA corrpath_src_p;

    // Initializes correction amount data
    memset(&corrpath_src_p, CLEAR, sizeof(MP_POS_DATA));
    corrpath_src_p.ctrl_grp = 1; // Robot1
    corrpath_src_p.grp_pos_info[0].pos_tag.data[0] = 0x3f; // Effective axis specification (6 axes)
    corrpath_src_p.grp_pos_info[0].pos_tag.data[2] = 0;
    corrpath_src_p.grp_pos_info[0].pos[0] = 0; //X
    corrpath_src_p.grp_pos_info[0].pos[1] = 0; //Y
    corrpath_src_p.grp_pos_info[0].pos[2] = 0; //Z
    corrpath_src_p.grp_pos_info[0].pos[3] = 0; //TX
    corrpath_src_p.grp_pos_info[0].pos[4] = 0; //TY
    corrpath_src_p.grp_pos_info[0].pos[5] = 0; //TZ

    FOREVER
    {
```

The correction data received with the system, by calling this API, is volatile (disappears, once read and retrieved). Then usually, the source task of calling this API needs to be activated with a constant cycle (such as the interpolation cycle), to achieve real-time tracking. Consequently, the program structure that periodically calls this API is achieved while the path correction is necessary. However, if the program calls the API irregularly (asynchronously), no contradiction occurs in the API.
mpClkAnnounce(MP_INTERPOLATION_CLK); // Performs the following process per interpolation control cycle

if ( correct_req == ON ) // Request for correction? Set by command receiving task.
{
  //PutCorrPath (1 [mm] correction in Y-direction per interpolation control cycle
  corrpath Src_p.grp_pos_info[0].pos[1] = 1000;
  mpMeiPutCorrPath(MP_SL_ID1, &corrpath Src_p);
}
}
mpMeiPutForcePathEnd

Force-quits the move instruction in execution, and changes the motion to the next step.

- **Syntax**
  
  ```c
  int mpMeiPutForcePathEnd
  (int sl_id, /* Specifies communication area with the system */
   CTRLG_T ctrl_grp /* Effective control groups specification */
  )
  ```

- **Description**
  
  This routine force-quits the move instruction of the robot specified by `ctrl_grp` currently in execution. Thus, the software of the system (simply called “system” hereafter) stops calculating the position command data of the current step, and starts calculating the position command data from the current position to the next step (teaching position). At this time, the current step terminates without deceleration, and motion to the next step starts. This request is received only once by the move instruction in execution (current step). Thus, the following step is not affected.

- **Parameter**

  `[sl_id]`

  Specifies the communication area with the system. The following values can be specified.

  Make sure to specify MP_SL_ID1 for the robot 1 (R1), and MP_SL_ID2 for the robot 2 (R2).

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_SL_ID1</td>
<td>Communication area 1</td>
</tr>
<tr>
<td>MP_SL_ID2</td>
<td>Communication area 2</td>
</tr>
</tbody>
</table>
[ctrl_grp]

Control group
Specify the target group in bits.
Retrieve the control group number of the target group by
mpCtrlGrpId2GrpNo().

- Return Value

0  Normal end
ERROR  Error
mpMeiPutSpdOverride

Sets a speed override amount.

**Syntax**

```c
int mpMeiPutSpdOverride
(int sl_id, /* Specifies communication area with the system */
CTRLG_T ctrl_grp, /* Effective control groups specification */
long src_p[MP_GRP_NUM] /* Speed override ratio(0~15,000) */
)
```

- **MP_GRP_NUM**: Maximum number of control groups 32

**Description**

This routine sets a speed override amount (ratio) to the robot specified by `ctrl_grp`, in the range of 0% to 150.00% (minimum resolution: 1/100 [%], valid integer value: 0 to 15,000).

Once this value is set, the value is used consecutively for the move instruction in motion. To cancel this, turn OFF all the bits specified by `ctrl_grp` and call this API. Then, the speed override amount from the application is canceled.

```c
// speed override cancel
rc = mpMeiPutSpdOverride(sl_id, OFF, NULL);
```

**Parameter**

**[sl_id]**

Specifies the communication area with the system. The following values can be specified. Make sure to specify MP_SL_ID1 for the robot 1 (R1), and MP_SL_ID2 for the robot 2 (R2).

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_SL_ID1</td>
<td>Communication area 1</td>
</tr>
<tr>
<td>MP_SL_ID2</td>
<td>Communication area 2</td>
</tr>
</tbody>
</table>
### Control group

Specify the target group in bits.
Retrieve the control group number of the target group by `mpCtrlGrpId2GrpNo()`.

Specify OFF (all bits turned OFF) to make all the speed override amount from MotoPlus invalid.

### Speed override amount (0 to 15,000, unit: 1/100 [%])

- **Return Value**
  - 0: Normal end
  - -1: Error
mpExRcsIncrementMove

Sets the increment move amount per every interpolation period to the manipulator executing the WAIT instruction. This API has a function that a slave side manipulator can be operated by being linked and coordinated to the operation to the master control group.

- **Syntax**

```c
int mpExRcsIncrementMove
(
    MP_EXPOS_DATA *src_p /* pointer of the structure which sets the path correcting data */
)
```

- **Description**

This API sets the increment move amount to the manipulator of the control group that `src_p->ctrl_grp` indicates. Since this API adds the specified increment move amount to the current position, the increment move (such as the jog movement) is available. However, as a necessary condition, the target control group needs to be executing the WAIT instruction in the job. When the job execution stops because of the emergency stop, the hold ON, or the mode change, this correction also stops. When the job is started again, this API starts the increment move again.

In the following job example, by the execution of “SET B000 1”, it prepares to receive the specification of increment value move from the MotoPlus application. And then, the increment value is specified by the execution of “mpRcsIncrementMove()”. The specification of increment value is received while the job is executing WAIT instruction. However, in case input IN#(1) is turned ON, the following MOVL instruction is executed after WAIT instruction is passed through.

At this time, the correction is ignored even if the increment value is specified by mpRcsIncrementMove(). (mpRcsIncrementMove() API returns “E_EXRCS_IMOV_UNREADY(-1): not under executing WAIT” as its return value.)
<JOB example>

: 

MOVL C000 V=100.0

SET B000 1

WAIT IN#(1) = ON        // The increment value move is possible
only while this WAIT instruction is being executed.

MOVL C001 V=100.0

:

This specification is available not only for the manipulator but also to the
external axis (station axis).

■ Parameter

[src_p]

The increment move value data
typedef struct {
CTRLG_T ctrl_grp; /* Target control group which executes the
            increment value move*/
CTRLG_T m_ctrl_grp; /* Master side control group for coordinated
            (synchronized) operation */
CTRLG_T s_ctrl_grp; /* Slave side control group for coordinated
            (synchronized) operation */
MP_GRP_POS_INFO grp_pos_info[MP_GRP_NUM];
}MP_EXPOS_DATA;

• ctrl_grp
  Control group of the object operation

• m_ctrl_grp
  Master side control group for coordinated (synchronized) operation.
  It must be the same control group as "ctrl_grp". Set “0” in case
  synchronized operation is not executed.

• s_CTRL_g
  Slave side control group for coordinated (synchronized) operation.
  Set “0” in case synchronized operation is not executed.
Followings are the functional errors when setting the control group data.

1. When the specified “m_ctrl_grp” is not found (does not exist) in the system configuration.
   - When “s_ctrl_grp” is set, “m_ctrl_grp” is “0”, or more than one control groups are specified.
     → E_MXRCS_M_CTRLGRP (-11)

2. When “m_ctrl_grp” is set, “s_ctrl_grp” is “0”, or it is not found (does not exist or “0” is set) in the system configuration.
   - E_EXRCS_S_CTRLGRP (-12)

3. When overlapped control groups exist in the specified “m_ctrl_grp” and “s_ctrl_grp”.
   - E_MXRCS_SYNC_MOVE_CTRLGRP (-10)

(e.g.) The control group setting and the action on the R1+R2 system

<table>
<thead>
<tr>
<th>ctrl_grp</th>
<th>m_ctrl_grp</th>
<th>s_ctrl_grp</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>R1 independent operation</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>Operate R1 and R2 synchronizes</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>Error: E_EXRCS_SYNC_MOVE_CTRLGRP(-10) Control group specification error in the coordinated operation</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Error: E_EXRCS_M_CTRLGRP(-11) Master side control group specification error</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Error: E_EXRCS_S_CTRLGRP(-12) Slave side control group specification error</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Error: E_EXRCS_SYNC_MOVE_CTRLGRP(-10) Control group specification error in the coordinated operation</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>Error: E_EXRCS_M_CTRLGRP(-11) Master side control group specification error</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>Error: E_EXRCS_S_CTRLGRP(-12) Slave side control group specification error</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>R2 independent operation</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>Error: E_EXRCS_SYNC_MOVE_CTRLGRP(-10) Control group specification error in the coordinated operation</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>Operate R1 and R2 synchronizes</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>Error: E_EXRCS_M_CTRLGRP(-11) Master side control group specification error</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>Error: E_EXRCS_S_CTRLGRP(-12) Slave side control group specification error</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
<td>Error: E_EXRCS_M_CTRLGRP(-11) Master side control group specification error</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>Error: E_EXRCS_S_CTRLGRP(-12) Slave side control group specification error</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>Error: E_EXRCS_SYNC_MOVE_CTRLGRP(-10) Control group specification error in the coordinated operation</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>Error: E_EXRCS_M_CTRLGRP(-11) Master side control group specification error</td>
</tr>
</tbody>
</table>
typedef struct {
  MP_POS_TAG pos_tag;
  long pos[MP_GRP_AXES_NUM]; /* Correction amount data*/
} MP_GRP_POS_INFO;

typedef struct {
  UCHAR data[8]; /* For the property and details of the data, see below */
  CTRLG_T ei_ctrl_grp; /* Control group specification when EIMOV robot */
} MP_POS_TAG;

MP_GRP_AXES_NUM: Maximum axis number of the control group (8)
MP_GRP_NUM: Number of the control group (4)

MP_POS_TAG.data[0]: Information of axis configuration

<table>
<thead>
<tr>
<th>Value</th>
<th>real value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_INC_PULSE_DTY</td>
<td>0x80</td>
<td>Increment value specification by pulse data</td>
</tr>
<tr>
<td>MP_INC_ANGLE_DTY</td>
<td>0x81</td>
<td>Increment value specification by angle data</td>
</tr>
<tr>
<td>MP_INC_BF_DTYPE</td>
<td>0x90</td>
<td>Increment value specification by base coordinate system cartesian data</td>
</tr>
<tr>
<td>MP_INC_RF_DTYPE</td>
<td>0x91</td>
<td>Increment value specification by robot coordinate system cartesian data</td>
</tr>
<tr>
<td>MP_INC_TF_DTYPE</td>
<td>0x92</td>
<td>Increment value specification by tool coordinate system cartesian data</td>
</tr>
<tr>
<td>MP_INC_UF_DTYPE</td>
<td>0x93</td>
<td>Increment value specification by user coordinate system cartesian data</td>
</tr>
</tbody>
</table>

MP_POS_TAG.data[2]: Tool file number
The tool file number used when increment value move is executed in the tool coordinate system

MP_POS_TAG.data[3]: Data type

MP_POS_TAG.data[4]: User coordinate number
(0~15: valid only when user coordinate is specified)

MP_POS_TAG.data[1], [5]~[7]: For future expansion
## MP_GRP_POS_INFO pos

<table>
<thead>
<tr>
<th>Array</th>
<th>Pulse type</th>
<th>Angle type</th>
<th>Cartesian type</th>
</tr>
</thead>
<tbody>
<tr>
<td>pos[0]</td>
<td>1st axis (S) pulse</td>
<td>1st axis (S) angle [0.0001deg]</td>
<td>X-axis coordinate [μm]</td>
</tr>
<tr>
<td>pos[1]</td>
<td>2nd axis (L) pulse</td>
<td>2nd axis (L) angle [0.0001deg]</td>
<td>Y-axis coordinate [μm]</td>
</tr>
<tr>
<td>pos[2]</td>
<td>3rd axis (U) pulse</td>
<td>3rd axis (U) angle [0.0001deg]</td>
<td>Z-axis coordinate [μm]</td>
</tr>
<tr>
<td>pos[3]</td>
<td>4th axis (R) pulse</td>
<td>4th axis (R) angle [0.0001deg]</td>
<td>Wrist angle Rx [0.0001deg]</td>
</tr>
<tr>
<td>pos[4]</td>
<td>5th axis (B) pulse</td>
<td>5th axis (B) angle [0.0001deg]</td>
<td>Wrist angle Ry [0.0001deg]</td>
</tr>
<tr>
<td>pos[5]</td>
<td>6th axis (T) pulse</td>
<td>6th axis (T) angle [0.0001deg]</td>
<td>Wrist angle Rz [0.0001deg]</td>
</tr>
<tr>
<td>pos[6]</td>
<td>7th axis (E) pulse</td>
<td>7th axis (E) angle [0.0001deg]</td>
<td>angle Re [0.0001deg]</td>
</tr>
<tr>
<td>pos[7]</td>
<td>8th axis pulse</td>
<td>8th axis angle [0.0001deg]</td>
<td>8th axis pulse value (micron in the case of traveling axis)</td>
</tr>
</tbody>
</table>

### Return value
- 0 : Normal end
- One of the following negative values : Error

### Value Description

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_EXRCS_IMOV_UNREADY(-1)</td>
<td>Not executing WAIT instruction</td>
</tr>
<tr>
<td>E_EXRCS_CTRL_GRP(-3)</td>
<td>Error in control group setting (It occurs when the operation target control group specification is overlapped, too. e.g. When a specified control group is used in other task of the independent control.)</td>
</tr>
<tr>
<td>E_EXRCS_AXES_CONFIG(-4)</td>
<td>Error in axis construction data</td>
</tr>
<tr>
<td>E_EXRCS_TOOL_NO(-5)</td>
<td>Error in tool number</td>
</tr>
<tr>
<td>E_EXRCS_POS_DTYPE(-6)</td>
<td>Error in data type</td>
</tr>
<tr>
<td>E_EXRCS_USER_FRAME(-7)</td>
<td>Error in user coordinate setting</td>
</tr>
<tr>
<td>E_EXRCS_JOG_BUSY(-8)</td>
<td>Operating jog</td>
</tr>
<tr>
<td>E_EXRCS_MOVE_BUSY(-9)</td>
<td>Executing move instruction</td>
</tr>
<tr>
<td>E_EXRCS_SYNC_MOVE_CTRL_GRP(-10)</td>
<td>Error in control group specification in coordinated operation</td>
</tr>
<tr>
<td>E_EXRCS_M_CTRL_GRP(-11)</td>
<td>Error in master side control group specification</td>
</tr>
<tr>
<td>E_EXRCS_S_CTRL_GRP(-12)</td>
<td>Error in slave side control group specification</td>
</tr>
<tr>
<td>E_EXRCS_USING_CTRL_GRP(-13)</td>
<td>Job of the control group specified in <code>mpExRcsIncrementMove()</code> is executing.</td>
</tr>
</tbody>
</table>
Sample code

The following is the sample code for increment value move operation.

When incomove_req is set by the command receiving task, the manipulator moves 1mm to Y-direction per interpolation cycle.

Although the increment value move process is FOREVER (infinite loop), `mpClkAnnounce(MP_INTERPOLATION_CLK);` is described at the starting of the loop so that the increment value move process synchronizes with the interpolation control cycle.

```c
int incmove_req;
void pathCorrectionTask(void)
{
    MP_EXPOS_DATA corrpath_src_p;

    // memset(&corrpath_src_p, CLEAR, sizeof(MP_EXPOS_DATA));
    corrpath_src_p.ctrl_grp = 1; // Robot1
    corrpath_src_p.grp_pos_info[0].pos_tag.data[0] = 0x3f;  // Effective axis specification (6 axes)
    corrpath_src_p.grp_pos_info[0].pos[0] = 0;     //X
    corrpath_src_p.grp_pos_info[0].pos[1] = 0;     //Y
```
corrpath_src_p.grp_pos_info[0].pos[2] = 0;     //Z
corrpath_src_p.grp_pos_info[0].pos[3] = 0;     //Rx
corrpath_src_p.grp_pos_info[0].pos[4] = 0;     //Ry
corrpath_src_p.grp_pos_info[0].pos[5] = 0;     //Rz

FOREVER
{
    mpClkAnnounce(MP_INTERPOLATION_CLK);
    // Execute following process per interpolation control cycle
    if (incmove_req== ON) //Correction required?
        Set by command receiving task .
    {
        // PutCorrPath (Correct 1[mm] in Y-direction per interpolation control cycle
        corrpath_src_p.grp_pos_info[0].pos[1] = 1000;
        mpExRcsIncrementMove(&corrpath_src_p);
    }
}
}
15 Servo Monitor/Control API

mpSvsGetVelTrqFb

Retrieves the feedback speed and torque of all control groups and axes.

- Syntax
  ```c
  int mpSvsGetVelTrqFb
  (    
    MP_GRP_AXES_T dst_vel, /* destination for velocity feedback */
    MP_TRQ_CTL_VAL *dst_trq/* destination for torque feedback */
  )
  ```

- Description
  This API writes the feedback speed and torque of all control groups and axes in the area specified by the parameter. Since the area specified by the parameter is not initialized (0 clear) by this API, initialize the area by the application as necessary. If NULL is set for the area specification of the parameter, the target data are not written.

  The system monitor APIs mpGetFBSpeed(), mpGetTorque(), and mpGetTorqueEx() can also return the feedback speed and torque, but by this API, mpSvsGetVelTrqFb(), the data of speed and torque are updated at shorter intervals.

  Also, the units of the data of speed and torque are different from those of mpGetFBSpeed(), mpGetTorque(), and mpGetTorqueEx(). Regarding the data of torque, the direction (sign) may be different from that of mpGetTorque() and mpGetTorqueEx().

  For details, refer to the following description about the parameter.

- Parameter
  - **[dst_vel]**
    Specify the pointer to the feedback speed receiving array. The feedback speed is written in the array in units of 0.1 pulse/sec.

    - **MP_GRP_AXES_T**
      Data array for all control groups and axes
      Syntax: typedef long MP_GRP_AXES_T[MP_GRP_NUM][MP_GRP_AXES_NUM];
      Data: <dst_vel>
      Feedback speed

      | Array  | Description |
      |--------|-------------|
      | dst_vel[x][y] | The feedback speed of the y-th axis of the x-th control group. Use the value retrieved by mpCtrlGrpId2GrpNo() as the control group number. |
      | x: control group number | y: axis number |


Specify the pointer to the torque receiving array.

- **MP_TRQ_CTL_VAL**
  Torque data structure
  Syntax: typedef struct {
    int unit;
    MP_TRQCTL_DATA data;
  } MP_TRQ_CTL_VAL;
  Member: <unit>
  Torque data unit

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRQ_PERCENTAGE</td>
<td>The torque data can be retrieved in the percentage of the rated torque of the motor. The data is in units of 0.01%.</td>
</tr>
<tr>
<td>TRQ_NEWTON_METER</td>
<td>The torque value of the motor can be retrieved. The data is in units of 0.000001 Nm.</td>
</tr>
</tbody>
</table>

- **MP_TRQCTL_DATA**
  Torque data array
  The torque data are written in this array in the units specified in “unit” above.
  Syntax: #define MP_TRQCTL_DATA MP_GRP_AXES_T
typedef long MP_GRP_AXES_T[MP_GRP_NUM][MP_GRP_AXES_NUM];

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data[x][y]</td>
<td>The torque data of the y-th axis of the x-th control group. Use the value retrieved by mpCtrlGrpId2GrpNo() as the control group number.</td>
</tr>
</tbody>
</table>

The direction (sign) of the torque retrieved by mpSvsGetVelTrqFb() is the same as the direction of the axis rotation. Also, it coincides with the direction of the speed.

However, it may be different from the direction of the torque retrieved by the system monitor API, mpGetTorque() or mpGetTorqueEx().

**NOTE**

For mpGetTorque() and mpGetTorqueEx(), the torque’s normal (forward) direction is not the direction of the axis rotation but the direction of the motor’s normal (forward) rotation.

The direction of the axis rotation and the direction of the motor’s normal rotation may be reversed. If so, the direction of the torque retrieved by mpGetTorque() or mpGetTorqueEx() is the reverse of the direction of the speed and the torque for the servo control API.

**Return Value**
- OK Normal end
- ERROR Error
mpSvsStartTrqLimit

Starts the torque limit. (Only for the external axis)

**CAUTION**

- By using this API, the motor may not be able to maintain its position and may move in an unexpected direction. Carefully read "FS100 OPTIONS INSTRUCTIONS Programmer’s Manual for New Language Environment MotoPlus (HW1480841)", and be sure to confirm safety before using this API.

**Syntax**

```c
MP_SVS_HANDLE mpSvsStartTrqLimit
(
    MP_GRP_CONFIG *arg_grp_config; /* specified effective group axes */
)
```

**Description**

The execution of this API sets the control group and axis for which the torque limit is executed, and the torque limit is made ready to start. However, the torque limit cannot be started under one of the following conditions:

- The manipulator or the base axis is specified as the target group.
- The axis for which the torque limit or the torque control is already executed is present in the specified control group.
- The move command is output to the specified control group.
- One of the following functions, which cannot be executed simultaneously with the torque limit, is already executed:
  - Servo float function
  - Speed control function
  - Search function

Note that the actual torque limit is started after executing this API and then mpSvsSetTrqLimit().

**Parameter**

**[arg_grp_config]**

Specify the control group and axis for which the torque limit is executed.

- **MP_GRP_CONFIG**
  Control group and control axis structure
  Syntax: typedef struct {
    CTRLG_T ctrl_grp;
    UCHAR axes_config[MP_GRP_NUM];
  } MP_GRP_CONFIG;

---

---

---
Member: <ctrl_grp>
Control group
Specify the target group in bits.
Retrieve the control group number of the target group by mpCtrlGrpId2GrpNo().

<axes_config[]>
Axis configuration information
Specify the target axis in bits.
Specify the control group number of the target group retrieved by mpCtrlGrpId2GrpNo() as the array number.

■ Return Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_SVS_HANDLE_FULL</td>
<td>-2 The number of the handle values issued already reaches the limit.</td>
</tr>
<tr>
<td>E_SVS_UNINITIALIZED</td>
<td>-5 The area for the servo control is not initialized.</td>
</tr>
<tr>
<td>E_SVS_USING_CTRL_GRP</td>
<td>-6 The axis for which the torque limit or the torque control is already executed is present in the specified control group.</td>
</tr>
</tbody>
</table>
## Servo Monitor/Control API

### mpSvsStartTrqLimit

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_SVS_NON_CTRL_GRP</td>
<td>-7 No group is specified.</td>
</tr>
<tr>
<td>E_SVS_CTRL_GRP_NOT_EXIST</td>
<td>-8 Non-existent group is specified.</td>
</tr>
<tr>
<td>E_SVS_AXES_SPECIFIED_NON</td>
<td>-9 Non-existent axis is specified.</td>
</tr>
<tr>
<td>E_SVS_MOVE_BUSY</td>
<td>-10 This API is executed for the group to which the move command is output.</td>
</tr>
<tr>
<td>E_SVS_AX_COMM_OBJ_FULL</td>
<td>-11 It was impossible to command the servo-control part to move to the torque limit.</td>
</tr>
<tr>
<td>E_SVS_AX_COMM_TIME_OVER</td>
<td>-12</td>
</tr>
<tr>
<td>E_SVS_AX_COMM_FAILURE</td>
<td>-13</td>
</tr>
<tr>
<td>E_SVS_INVALID_ROB_TRQ_FUNC</td>
<td>-14 This API was executed for the manipulator or the base axis.</td>
</tr>
<tr>
<td>E_SVS_INVALID_ROB_TRQ_AXIS</td>
<td>-15</td>
</tr>
<tr>
<td>E_SVS_AX_TRQ_LMT_SET</td>
<td>-16 The servo-control part was not able to move to the torque limit mode.</td>
</tr>
<tr>
<td>E_SVS_RTP_NOT_SUPPORTED</td>
<td>-25 The system software does not support this API.</td>
</tr>
<tr>
<td>E_SVS_JOG_BUSY</td>
<td>-26 This API was executed for the group in JOG operation.</td>
</tr>
</tbody>
</table>
**mpSvsSetTrqLimit**

Sets the torque limit value. (Only for the external axis)

---

**CAUTION**

- By using this API, the motor may not be able to maintain its position and may move in an unexpected direction. Carefully read "FS100 OPTIONS INSTRUCTIONS Programmer's Manual for New Language Environment MotoPlus (HW1480841)", and be sure to confirm safety before using this API.

### Syntax

```c
int mpSvsSetTrqLimit
(
    MP_SVS_HANDLE handle, /* specified SVS handle */
    MP_GRP_CONFIG *arg_grp_config, /* torque limit set group axes */
    MP_TRQ_LMT_VAL *arg_lmt_val /* torque limit max/min value */
)
```

### Description

This API sets the torque limit value for the control group and axis for which the torque limit is started by mpSvsStartTrqLimit(). The value within ±100% can be set as the torque limit value.

### Parameter

- **[handle]**
  Specify the handle value which is returned after executing mpSvsStartTrqLimit().

- **[arg_grp_config]**
  Specify the control group and axis for which the torque limit value is set.
  It is not necessary to specify all of the control groups and axes set when executing mpSvsStartTrqLimit(), so it is possible to specify only some of them. However, the groups and axes to be specified must be included in the control groups and axes set when executing mpSvsStartTrqLimit().

- **[arg_lmt_val]**
  Torque limit max/min value

---

### MP_GRP_CONFIG

Control group and control axis structure

Syntax: `typedef struct {` 

```c
CTRLG_T          ctrl_grp; 
UCHAR              axes_config[MP_GRP_NUM]; 
}                         MP_GRP_CONFIG;
```

Member: `<ctrl_grp>`

Control group

Specify the target group in bits.

Retrieve the control group number of the target group by `mpCtrlGrpId2GrpNo()`.
mpSvsSetTrqLimit

<axes_config[]>
Axis configuration information
Specify the target axis in bits.
Specify the control group number of the target group retrieved by mpCtrlGrpId2GrpNo() as the array number.

[arg_lmt_val]
Specify the pointer to the torque limit value structure.

- MP_TRQ_LMT_VAL
  Torque limit value structure
  Syntax: typedef struct {
    int unit;
    MP_TRQLMT_RANGE data[MP_GRP_NUM];
  } MP_TRQ_LMT_VAL;
  Member: <unit>
  Torque data unit

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRQ_PERCENTAGE</td>
<td>The torque limit value is specified in the percentage of the rated torque of the motor. The data is in units of 0.01%.</td>
</tr>
<tr>
<td>TRQ_NEWTON_METER</td>
<td>The torque limit value of the motor is specified. The data is in units of 0.000001 Nm.</td>
</tr>
</tbody>
</table>
• MP_TRQLMT_RANGE
  Torque limit value structure
  Syntax: typedef struct {
    long  max[MP_GRP_AXES_NUM];
    long  min[MP_GRP_AXES_NUM];
  }  MP_TRQLMT_RANGE;
  Member: <max[]>
    Torque upper limit value
  <min[]>
    Torque lower limit value

  Array Description
  data[x].max[y]  The torque upper limit value of
                  the y-th axis of the x-th control
                  group. Use the value retrieved by
                  mpCtrlGrpId2GrpNo() as the
                  control group number.
  x: control group number
  y: axis number
  data[x].min[y]  The torque lower limit value of
                  the y-th axis of the x-th control
                  group. Use the value retrieved by
                  mpCtrlGrpId2GrpNo() as the
                  control group number.
  x: control group number
  y: axis number

• It is possible to set positive (plus) values to both of the
  torque upper limit value and the torque lower limit value,
  or negative (minus) values to both of them. However, if
  those values are set as such, the torque cannot take the
  value 0 and a certain torque is always applied to the
  motor, thus the motor may move.

• The direction (sign) of the torque limit value is the same as
  the direction of the axis rotation. Also, it coincides with the
  direction of the speed. However, it may be different from the
direction of the torque retrieved by the system monitor API,
mpGetTorque() or mpGetTorqueEx(). For mpGetTorque() and
mpGetTorqueEx(), the torque’s normal (forward) direction is not
the direction of the axis rotation but the direction of the motor’s
normal (forward) rotation. The direction of the axis rotation and
the direction of the motor’s normal rotation may be reversed. If so,
the direction of the torque retrieved by mpGetTorque() or
mpGetTorqueEx() is the reverse of the direction of the
speed and the torque for this API.
## Return Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>E_SVS_BAD_HANDLE</code></td>
<td>-1 The handle value for servo control is not specified.</td>
</tr>
<tr>
<td><code>E_SVS_NOT_FUNC_CONCERNED</code></td>
<td>-3 The handle value for torque limit is not specified.</td>
</tr>
<tr>
<td><code>E_SVS_NON_CTRL_GRP</code></td>
<td>-7 No group is specified.</td>
</tr>
<tr>
<td><code>E_SVS_CTRL_GRP_NOT_EXIST</code></td>
<td>-8 Non-existent group is specified.</td>
</tr>
<tr>
<td><code>E_SVS_NOT_TRQ_LMT_AXIS</code></td>
<td>-18 An invalid axis is specified.</td>
</tr>
<tr>
<td><code>E_SVS_TRQ_LMT_LARGER_THAN_ULT</code></td>
<td>-20 A value larger than the upper limit is set to the lower limit.</td>
</tr>
<tr>
<td><code>E_SVS_TRQ_LMT_RANGE_MAX</code></td>
<td>-21 The value set to the upper limit exceeds the allowable range.</td>
</tr>
<tr>
<td><code>E_SVS_TRQ_LMT_RANGE_MIN</code></td>
<td>-22 The value set to the lower limit exceeds the allowable range.</td>
</tr>
<tr>
<td><code>E_SVS_RTP_NOT_SUPPORTED</code></td>
<td>-25 The system software does not support this API.</td>
</tr>
</tbody>
</table>
mpSvsEndTrqLimit

Ends the torque limit. (Only for the external axis)

**CAUTION**

- By using this API, the motor may not be able to maintain its position and may move in an unexpected direction. Carefully read "FS100 OPTIONS INSTRUCTIONS Programmer’s Manual for New Language Environment MotoPlus (HW1480841)", and be sure to confirm safety before using this API.

**Syntax**

```c
int mpSvsEndTrqLimit(
    MP_SVS_HANDLE handle /* specified SVS handle */
);
```

**Description**

This API ends the torque limit for all control groups and axes started by mpSvsStartTrqLimit(). During the torque limit, the position is not controlled, thus the command position and the feedback position do not coincide. Therefore, create the command position (current position) of the target group again after the torque limit ends, and adjust it to the feedback position.

Note that this API cannot be executed if the move command is output to the target group.

**NOTE**

Do not output the move command to the target group until the execution of this API is completed. Since the command position (current position) is created again after the execution of this API is completed, the move command may be changed considerably.

**Parameter**

(handle)

Specify the handle value which is returned after executing mpSvsStartTrqLimit().
### Return Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_SVS_BAD_HANDLE -1</td>
<td>The handle value for servo control is not specified.</td>
</tr>
<tr>
<td>E_SVS_NOT_FUNC_CONCERNED -3</td>
<td>The handle value for torque limit is not specified.</td>
</tr>
<tr>
<td>E_SVS_MOVE_BUSY -10</td>
<td>This API is executed for the group to which the move command is output.</td>
</tr>
<tr>
<td>E_SVS_AX_COMM_OBJ_FULL -11</td>
<td>It was impossible to command the servo-control part to end the torque limit.</td>
</tr>
<tr>
<td>E_SVS_AX_COMM_TIME_OVER -12</td>
<td></td>
</tr>
<tr>
<td>E_SVS_AX_COMM_FAILURE -13</td>
<td></td>
</tr>
<tr>
<td>E_SVS_AX_TRQ_LMT_CANCEL -17</td>
<td>The servo-control part was not able to end the torque limit mode.</td>
</tr>
<tr>
<td>E_SVS_PR_MAKE_FAILURE -24</td>
<td>Re-creation of the current position is failed.</td>
</tr>
<tr>
<td>E_SVS_RTP_NOT_SUPPORTED -25</td>
<td>The system software does not support this API.</td>
</tr>
<tr>
<td>E_SVS_JOG_BUSY -26</td>
<td>This API was executed for the group in JOG operation.</td>
</tr>
</tbody>
</table>
mpSvsStartTrqCtrl

Starts the torque control. (Only for the external axis)

CAUTION

- By using this API, the motor may not be able to maintain its position and may move in an unexpected direction. Carefully read “FS100 OPTIONS INSTRUCTIONS Programmer’s Manual for New Language Environment MotoPlus (HW1480841)”, and be sure to confirm safety before using this API.

**Syntax**

```c
MP_SVS_HANDLE mpSvsStartTrqCtrl
(
    MP_GRP_CONFIG *arg_grp_config; /* specified effective group axes */
)
```

**Description**

The execution of this API sets the control group and axis for which the torque control is executed, and the torque control is made ready to start. However, the torque control cannot be started under one of the following conditions:

- The manipulator or the base axis is specified as the target group.
- The axis for which the torque limit or the torque control is already executed is present in the specified control group.
- The move command is output to the specified control group.
- One of the following functions, which cannot be executed simultaneously with the torque control, is already executed:
  - Servo float function
  - Speed control function
  - Search function

Note that the actual torque control is started after executing this API and then mpSvsSetTrqCtrl().

**Parameter**

`[arg_grp_config]`

Specify the control group and axis for which the torque control is executed.

- MP_GRP_CONFIG
  Control group and control axis structure
  Syntax: typedef struct {
    CTRLG_T        ctrl_grp;
    UCHAR          axes_config[MP_GRP_NUM];
  } MP_GRP_CONFIG;

---

**Syntax**

```c
MP_SVS_HANDLE mpSvsStartTrqCtrl
(
    MP_GRP_CONFIG *arg_grp_config; /* specified effective group axes */
)
```

**Description**

The execution of this API sets the control group and axis for which the torque control is executed, and the torque control is made ready to start. However, the torque control cannot be started under one of the following conditions:

- The manipulator or the base axis is specified as the target group.
- The axis for which the torque limit or the torque control is already executed is present in the specified control group.
- The move command is output to the specified control group.
- One of the following functions, which cannot be executed simultaneously with the torque control, is already executed:
  - Servo float function
  - Speed control function
  - Search function

Note that the actual torque control is started after executing this API and then mpSvsSetTrqCtrl().

**Parameter**

`[arg_grp_config]`

Specify the control group and axis for which the torque control is executed.

- MP_GRP_CONFIG
  Control group and control axis structure
  Syntax: typedef struct {
    CTRLG_T        ctrl_grp;
    UCHAR          axes_config[MP_GRP_NUM];
  } MP_GRP_CONFIG;
Member: `<ctrl_grp>`
Control group
Specify the target group in bits.
Retrieve the control group number of the target group by
`mpCtrlGrpId2GrpNo()`.

```
\[\text{\ldots\ldots\ldots\ldots}\]
\text{Control group 0}
\text{Control group 1}
\text{Control group 2}
\ldots
\text{Control group 31}
```

`<axes_config[]>`
Axis configuration information
Specify the target axis in bits.
Specify the control group number of the target group
retrieved by `mpCtrlGrpId2GrpNo()` as the array number.

```
\[\text{\ldots\ldots\ldots\ldots}\]
\text{1st axis}
\text{2nd axis}
\text{3rd axis}
\ldots
\text{8th axis}
```

**Return Value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_SVS_HANDLE_FULL</td>
<td>-2   The number of the handle values issued already reaches the limit.</td>
</tr>
<tr>
<td>E_SVS_UNINITIALIZED</td>
<td>-5   The area for the servo control is not initialized.</td>
</tr>
<tr>
<td>E_SVS_USING_CTRL_GRP</td>
<td>-6   The axis for which the torque limit or the torque control is already executed is present in the specified control group.</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>E_SVS_NON_CTRL_GRP</td>
<td>-7 No group is specified.</td>
</tr>
<tr>
<td>E_SVS_CTRL_GRP_NOT_EXIST</td>
<td>-8 Non-existent group is specified.</td>
</tr>
<tr>
<td>E_SVS_AXES_SPECIFIED_NON</td>
<td>-9 Non-existent axis is specified.</td>
</tr>
<tr>
<td>E_SVS_MOVE_BUSY</td>
<td>-10 This API is executed for the group to which the move command is output.</td>
</tr>
<tr>
<td>E_SVS_AX_COMM_OBJ_FULL</td>
<td>-11 It was impossible to command the servo-control part to move to the torque control.</td>
</tr>
<tr>
<td>E_SVS_AX_COMM_TIME_OVER</td>
<td>-12</td>
</tr>
<tr>
<td>E_SVS_AX_COMM_FAILURE</td>
<td>-13</td>
</tr>
<tr>
<td>E_SVS_INVALID_ROB_TRQ_FUNC</td>
<td>-14 This API was executed for the manipulator or the base axis.</td>
</tr>
<tr>
<td>E_SVS_INVALID_ROB_TRQ_AXIS</td>
<td>-15</td>
</tr>
<tr>
<td>E_SVS_AX_TRQ_LMT_SET</td>
<td>-16 The servo-control part was not able to move to the torque control mode.</td>
</tr>
<tr>
<td>E_SVS_RTP_NOT_SUPPORTED</td>
<td>-25 The system software does not support this API.</td>
</tr>
<tr>
<td>E_SVS_JOG_BUSY</td>
<td>-26 This API was executed for the group in JOG operation.</td>
</tr>
</tbody>
</table>
mpSvsSetTrqCtrl

Sets the torque command value. (Only for the external axis)

CAUTION

- By using this API, the motor may not be able to maintain its position and may move in an unexpected direction. Carefully read "FS100 OPTIONS INSTRUCTIONS Programmer's Manual for New Language Environment MotoPlus (HW1480841)", and be sure to confirm safety before using this API.

- Syntax

```c
int mpSvsSetTrqCtrl
(
    MP_SVS_HANDLE handle, /* specified SVS handle */
    MP_GRP_CONFIG *arg_grp_config, /* torque limit set group axes */
    MP_TRQ_CTL_VAL *argCtl_val /* torque control values */
);
```

- Description

This API sets the torque command value for the control group and axis for which the torque control is started by mpSvsStartTrqCtrl(). The value within ±100% can be set as the torque command value.

- Parameter

  **[handle]**

Specify the handle value which is returned after executing mpSvsStartTrqCtrl().

  **[arg_grp_config]**

Specify the control group and axis for which the torque command value is set.

It is not necessary to specify all of the control groups and axes set when executing mpSvsStartTrqCtrl(), so it is possible to specify only some of them. However, the groups and axes to be specified must be included in the control groups and axes set when executing mpSvsStartTrqCtrl().

- MP_GRP_CONFIG

  Control group and control axis structure

  Syntax: typedef struct {
    CTRLG_T ctrl_grp;
    UCHAR axes_config[MP_GRP_NUM];
  } MP_GRP_CONFIG;

  Member: <ctrl_grp>

  Control group

  Specify the target group in bits.

  Retrieve the control group number of the target group by mpCtrlGrpld2GrpNo().
mpSvsSetTrqCtrl

<axes_config[]>
Axis configuration information
Specify the target axis in bits.
Specify the control group number of the target group
retrieved by mpCtrlGrpId2GrpNo() as the array number.

[arg_ctl_val]
Specify the pointer to the torque data structure.

- MP_TRQ_CTL_VAL
  Torque data structure
  Syntax: typedef struct {
    int unit;
    MP_TRQCTL_DATA data;
  } MP_TRQ_CTL_VAL;
  Member: <unit>
  Torque data unit

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRQ_PERCENTAGE</td>
<td>The torque command value is specified in the percentage of the rated torque of the motor. The data is in units of 0.01%.</td>
</tr>
<tr>
<td>TRQ_NEWTON_METER</td>
<td>The torque command value of the motor is specified. The data is in units of 0.000001 Nm.</td>
</tr>
</tbody>
</table>
15 Servo Monitor/Control API
mpSvsSetTrqCtrl

<data>
- MP_TRQCTL_DATA
  Torque data array
  Syntax:
  #define MP_TRQCTL_DATA MP_GRP_AXES_T
  typedef long
  MP_GRP_AXES_T[MP_GRP_NUM][MP_GRP_AXES_NUM];

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data[x][y]</td>
<td>The torque command value of the y-th axis of the x-th control group. Use the value retrieved by mpCtrlGrpId2GrpNo() as the control group number.</td>
</tr>
</tbody>
</table>

NOTE

The direction (sign) of the torque command value is the same as the direction of the axis rotation. Also, it coincides with the direction of the speed. However, it may be different from the direction of the torque retrieved by the system monitor API, mpGetTorque() or mpGetTorqueEx(). For mpGetTorque() and mpGetTorqueEx(), the torque’s normal (forward) direction is not the direction of the axis rotation but the direction of the motor’s normal (forward) rotation.

The direction of the axis rotation and the direction of the motor’s normal rotation may be reversed. If so, the direction of the torque retrieved by mpGetTorque() or mpGetTorqueEx() is the reverse of the direction of the speed and the torque for this API.

Return Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>Less than 0</td>
<td>Error</td>
</tr>
<tr>
<td>One of the following values is returned according to the error content.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_SVS_BAD_HANDLE</td>
<td>-1 The handle value for servo control is not specified.</td>
</tr>
<tr>
<td>E_SVS_NOT_FUNC_CONCERNED</td>
<td>-3 The handle value for torque limit is not specified.</td>
</tr>
<tr>
<td>E_SVS_NON_CTRL_GRP</td>
<td>-7 No group is specified.</td>
</tr>
<tr>
<td>E_SVS_CTRL_GRP_NOT_EXIST</td>
<td>-8 Non-existent group is specified.</td>
</tr>
<tr>
<td>E_SVS_NOT_TRQ_CTL_AXIS</td>
<td>-19 An invalid axis is specified.</td>
</tr>
<tr>
<td>E_SVS_TRQ_CTL_RANGE</td>
<td>-23 The value set as the torque command value exceeds the allowable range.</td>
</tr>
<tr>
<td>E_SVS_RTP_NOT_SUPPORTED</td>
<td>-25 The system software does not support this API.</td>
</tr>
</tbody>
</table>
mpSvsEndTrqCtrl

Ends the torque control. (Only for the external axis)

**CAUTION**

- By using this API, the motor may not be able to maintain its position and may move in an unexpected direction. Carefully read “FS100 OPTIONS INSTRUCTIONS Programmer’s Manual for New Language Environment MotoPlus (HW1480841)”, and be sure to confirm safety before using this API.

**Syntax**

```c
int mpSvsEndTrqCtrl
(
    MP_SVS_HANDLE handle /* specified SVS handle */
)
```

**Description**

This API ends the torque control for all control groups and axes started by mpSvsStartTrqCtrl(). During the torque control, the position is not controlled, thus the command position and the feedback position do not coincide. Therefore, create the command position (current position) of the target group again after the torque control ends, and adjust it to the feedback position.

Note that this API cannot be executed if the move command is output to the target group.

**NOTE**

Do not output the move command to the target group until the execution of this API is completed. Since the command position (current position) is created again after the execution of this API is completed, the move command may be changed considerably.

**Parameter**

**handle**

Specify the handle value which is returned after executing mpSvsStartTrqCtrl().
Return Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_SVS_BAD_HANDLE</td>
<td>-1</td>
</tr>
<tr>
<td>E_SVS_NOT_FUNC_CONCERNED</td>
<td>-3</td>
</tr>
<tr>
<td>E_SVS_MOVE_BUSY</td>
<td>-10</td>
</tr>
<tr>
<td>E_SVS_AX_COMM_OBJ_FULL</td>
<td>-11</td>
</tr>
<tr>
<td>E_SVS_AX_COMM_TIME_OVER</td>
<td>-12</td>
</tr>
<tr>
<td>E_SVS_AX_COMM_FAILURE</td>
<td>-13</td>
</tr>
<tr>
<td>E_SVS_AX_TRQ_LMT_CANCEL</td>
<td>-17</td>
</tr>
<tr>
<td>E_SVS_PR_MAKE_FAILURE</td>
<td>-24</td>
</tr>
<tr>
<td>E_SVS_RTP_NOT_SUPPORTED</td>
<td>-25</td>
</tr>
<tr>
<td>E_SVS_JOG_BUSY</td>
<td>-26</td>
</tr>
</tbody>
</table>
mpSvsForceInit

Force-quits the servo control (torque limit, torque control). (Only for the external axis)

- **Syntax**
  
  STATUS mpSvsForceInit(void)

- **Description**
  
  This API force-quits the servo control (torque limit, torque control) in execution if any, and initializes the area for servo control (including the handle value or internal objects).

  Use this API in the case when an error occurred during servo control due to the system and recovery is difficult.

  Note that if it is during the execution of the torque limit or torque control, the command position (current position) of the target group is created again to end the torque limit or torque control. Thus, if the move command is output to the control group for which the torque limit or torque control is executed (the command position is changing), the torque limit or torque control cannot be ended. Do not use this API when the move command is output to the control group for which the torque limit or torque control is executed.

- **Parameter**
  
  None

- **Return Value**
  
<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td>Normal end</td>
</tr>
<tr>
<td>ERROR</td>
<td>Error</td>
</tr>
<tr>
<td></td>
<td>In the case when it was impossible to end the servo control in execution, and the like.</td>
</tr>
</tbody>
</table>
16 Kinematics API

mpConvAxesToCartPos

Converts an angle position of the specified control group to a cartesian coordinate position (robot coordinate systems).

Syntax

```c
int mpConvAxesToCartPos(
    unsigned int    grp_no,
    long            angle[MP_GRP_AXES_NUM],
    unsigned int    tool_no,
    BITSTRING*      fig_ctrl,
    MS_COORD*       coord
);
```

Parameter

**[grp_no]**
Control group number
Acquire the control group number by mpCtrlGrpId2GrpNo().

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>First control group</td>
</tr>
<tr>
<td>1</td>
<td>Second control group</td>
</tr>
<tr>
<td>2</td>
<td>Third control group</td>
</tr>
<tr>
<td>3</td>
<td>Fourth control group</td>
</tr>
</tbody>
</table>

**[angle]**
Angle (degree) position
Syntax: `#define MP_GRP_AXES_NUM (8)`

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>angle[0]</td>
<td>1st axis (S) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>angle[1]</td>
<td>2nd axis (L) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>angle[2]</td>
<td>3rd axis (U) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>angle[3]</td>
<td>4th axis (R) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>angle[4]</td>
<td>5th axis (B) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>angle[5]</td>
<td>6th axis (T) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>angle[6]</td>
<td>7th axis (E) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>angle[7]</td>
<td>8th axis (W) angle value (unit: 0.0001 deg)</td>
</tr>
</tbody>
</table>
Kinematics API

mpConvAxesToCartPos

**[tool_no]**
Tool number

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Tool number 0</td>
</tr>
<tr>
<td>1</td>
<td>Tool number 1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>15</td>
<td>Tool number 15</td>
</tr>
</tbody>
</table>

**[fig_ctrl]**
Pointer to the figure information (output)

Syntax: `typedef unsigned int BITSTRING;`

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D00</td>
<td>0:Front 1:Back</td>
</tr>
<tr>
<td>D01</td>
<td>0:Upper arm 1:Lower arm</td>
</tr>
<tr>
<td>D02</td>
<td>0:Flip 1:No flip</td>
</tr>
<tr>
<td>D03</td>
<td>0:R&lt;180 1:R&gt;=180</td>
</tr>
<tr>
<td>D04</td>
<td>0:T&lt;180 1:T&gt;=180</td>
</tr>
<tr>
<td>D05</td>
<td>0:S&lt;180 1:S&gt;=180</td>
</tr>
<tr>
<td>D06-D31</td>
<td>Reserved by manufacturer</td>
</tr>
</tbody>
</table>

**[coord]**
Pointer to the cartesian coordinate position structure (output)

- `MP_COORD`
  Cartesian coordinate position structure
  Syntax: `typedef struct {
  long x, y, z;
  long x, ry, rz;
  long ex1, ex2;
  } MP_COORD;`

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>X-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>y</td>
<td>Y-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>z</td>
<td>Z-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>rx</td>
<td>Wrist angle Rx (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>ry</td>
<td>Wrist angle Ry (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>rz</td>
<td>Wrist angle Rz (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>ex1</td>
<td>Elbow angle Re (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>ex2</td>
<td>Reserved by manufacturer</td>
</tr>
</tbody>
</table>
### Return Value

0  Normal end

One of the following negative values  Error

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_KINEMA_FAILURE  (-1)</td>
<td>Failed in conversion</td>
</tr>
<tr>
<td>E_KINEMA_CONV_IMPOSSIBLE  (-2)</td>
<td>Conversion impossible</td>
</tr>
<tr>
<td>E_KINEMA_GRP_OUT_RANGE  (-3)</td>
<td>Specified control group out of range</td>
</tr>
<tr>
<td>E_KINEMA_TOOL_OUT_RANGE  (-4)</td>
<td>Specified tool number out of range</td>
</tr>
</tbody>
</table>

In case executing the API in each of the following condition, because the system is busy, correct conversion result cannot be acquired, and then E_KINEMA_FAILURE(-1) is returned to “Return Value”.

- When setting a parameter
- When setting a tool file
- When setting an user coordinate
- When setting a calibration

When E_KINEMA_FAILURE(-1) is returned to “Return Value”, try executing the API again after an interval.
mpConvCartPosToAxes

Converts a cartesian coordinate position (robot coordinate systems) of the specified control group to an angle position.

- Syntax

```c
int mpConvCartPosToAxes(
    unsigned int grp_no,
    MP_COORD* coord,
    unsigned int tool_no,
    BITSTRING fig_ctrl,
    long prev_angle[MP_GRP_AXES_NUM],
    MP_KINEMA_TYPE type,
    long angle[MP_GRP_AXES_NUM]
);
```

- Parameter

[group_no]
Control group number

Acquire the control group number by mpCtrlGrpId2GrpNo()

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>First control group</td>
</tr>
<tr>
<td>1</td>
<td>Second control group</td>
</tr>
<tr>
<td>2</td>
<td>Third control group</td>
</tr>
<tr>
<td>3</td>
<td>Fourth control group</td>
</tr>
</tbody>
</table>
**Kinematics API**

**mpConvCartPosToAxes**

**[coord]**

Pointer to the cartesian coordinate position structure (input)

- **MPCOORD**
  Cartesian coordinate position structure
  Syntax: typedef struct
  ```
  {
    long x, y, z;
    long ry, rz;
    long ex1, ex2;
  }  MPCOORD;
  ```

**[tool_no]**

Tool number

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Tool number 0</td>
</tr>
<tr>
<td>1</td>
<td>Tool number 1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>15</td>
<td>Tool number 15</td>
</tr>
</tbody>
</table>

**[fig_ctrl]**

Figure information

Syntax: typedef unsigned int BITSTRING;

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D00</td>
<td>0:Front 1:Back</td>
</tr>
<tr>
<td>D01</td>
<td>0:Upper arm 1:Lower arm</td>
</tr>
<tr>
<td>D02</td>
<td>0:Flip 1:No flip</td>
</tr>
<tr>
<td>D03</td>
<td>0:R&lt;180 1:R&gt;=180</td>
</tr>
<tr>
<td>D04</td>
<td>0:T&lt;180 1:T&gt;=180</td>
</tr>
<tr>
<td>D05</td>
<td>0:S&lt;180 1:S&gt;=180</td>
</tr>
<tr>
<td>D06 - D31</td>
<td>Reserved by manufacturer</td>
</tr>
</tbody>
</table>
**prev_angle**

Previous angle

It is expressed by the unit “angle”. NULL can be specified to the angle. When NULL is specified, it is regarded as “0” is set to all elements.

Syntax: `#define MP_GRP_AXES_NUM (8)`

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prev_angle[0]</td>
<td>1st axis (S) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>prev_angle[1]</td>
<td>2nd axis (L) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>prev_angle[2]</td>
<td>3rd axis (U) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>prev_angle[3]</td>
<td>4th axis (R) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>prev_angle[4]</td>
<td>5th axis (B) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>prev_angle[5]</td>
<td>6th axis (T) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>prev_angle[6]</td>
<td>7th axis (E) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>prev_angle[7]</td>
<td>8th axis (W) angle value (unit: 0.0001 deg)</td>
</tr>
</tbody>
</table>

**type**

Inverse kinematics method specification

- MP_KINEMA_TYPE
  Inverse kinematics method each
  Syntax: `typedef enum`
    ```c
    {
      MP_KINEMA_DEFAULT,
      MP_KINEMA_DELTA,
      MP_KINEMA_FIG,
    } MP_KINEMA_TYPE;
    ```

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_KINEMA_DEFAULT</td>
<td>Default inverse kinematics</td>
</tr>
<tr>
<td>MP_KINEMA_DELTA</td>
<td>Inverse kinematics to microtargeted value</td>
</tr>
<tr>
<td>MP_KINEMA_FIG</td>
<td>Inverse kinematics to realize the figure</td>
</tr>
</tbody>
</table>

**angle**

Angle (degree) position

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>angle[0]</td>
<td>1st axis (S) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>angle[1]</td>
<td>2nd axis (L) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>angle[2]</td>
<td>3rd axis (U) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>angle[3]</td>
<td>4th axis (R) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>angle[4]</td>
<td>5th axis (B) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>angle[5]</td>
<td>6th axis (T) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>angle[6]</td>
<td>7th axis (E) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>angle[7]</td>
<td>8th axis (W) angle value (unit: 0.0001 deg)</td>
</tr>
</tbody>
</table>
### Return Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_KINEMA_FAILURE</td>
<td>Failed in conversion</td>
</tr>
<tr>
<td>E_KINEMA_CONV_IMPOSSIBLE</td>
<td>Conversion impossible</td>
</tr>
<tr>
<td>E_KINEMA_GRP_OUT_RANGE</td>
<td>Specified control group out of range</td>
</tr>
<tr>
<td>E_KINEMA_TOOL_OUT_RANGE</td>
<td>Specified tool number out of range</td>
</tr>
<tr>
<td>E_KINEMA_KINEMA_TYPE_ERR</td>
<td>Inverse kinematics wrong</td>
</tr>
<tr>
<td>E_KINEMA_INTERPOLATION_INVALID</td>
<td>Interpolation operation at the cartesian position before conversion is invalid.</td>
</tr>
</tbody>
</table>

In case executing the API in each of the following condition, because the system is busy, correct conversion result cannot be acquired, and then E_KINEMA_FAILURE(-1) is returned to “Return Value”.

- When setting a parameter
- When setting a tool file
- When setting an user coordinate
- When setting a calibration

When E_KINEMA_FAILURE(-1) is returned to “Return Value”, try executing the API again after an interval.
mpConvPulseToAngle

Converts a pulse position of the specified control group to an angle position.

Syntax

```c
int mpConvPulseToAngle(
    unsigned int   grp_no,
    long           pulse[MP_GRP_AXES_NUM],
    long           angle[MP_GRP_AXES_NUM]
);
```

Parameter

- **[grp_no]**
  Control group number
  Acquire the control group number by mpCtrlGrpId2GrpNo().

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>First control group</td>
</tr>
<tr>
<td>1</td>
<td>Second control group</td>
</tr>
<tr>
<td>2</td>
<td>Third control group</td>
</tr>
<tr>
<td>3</td>
<td>Fourth control group</td>
</tr>
</tbody>
</table>

- **[pulse]**
  Pulse position
  Syntax: #define MP_GRP_AXES_NUM (8)

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pulse[0]</td>
<td>1st axis (S) pulse value</td>
</tr>
<tr>
<td>pulse[1]</td>
<td>2nd axis (L) pulse value</td>
</tr>
<tr>
<td>pulse[2]</td>
<td>3rd axis (U) pulse value</td>
</tr>
<tr>
<td>pulse[3]</td>
<td>4th axis (R) pulse value</td>
</tr>
<tr>
<td>pulse[4]</td>
<td>5th axis (B) pulse value</td>
</tr>
<tr>
<td>pulse[5]</td>
<td>6th axis (T) pulse value</td>
</tr>
<tr>
<td>pulse[6]</td>
<td>7th axis (E) pulse value</td>
</tr>
<tr>
<td>pulse[7]</td>
<td>8th axis (W) pulse value</td>
</tr>
</tbody>
</table>
Kinematics API

**mpConvPulseToAngle**

### [angle]

Angle (degree) position

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>angle[0]</td>
<td>1st axis (S) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>angle[1]</td>
<td>2nd axis (L) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>angle[2]</td>
<td>3rd axis (U) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>angle[3]</td>
<td>4th axis (R) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>angle[4]</td>
<td>5th axis (B) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>angle[5]</td>
<td>6th axis (T) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>angle[6]</td>
<td>7th axis (E) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>angle[7]</td>
<td>8th axis (W) angle value (unit: 0.0001 deg)</td>
</tr>
</tbody>
</table>

### Return Value

- **0**  Normal end
- One of the following negative values  Error

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_KINEMA_FAILURE (-1)</td>
<td>Failed in conversion</td>
</tr>
<tr>
<td>E_KINEMA_CONV IMPOSSIBLE (-2)</td>
<td>Conversion impossible</td>
</tr>
<tr>
<td>E_KINEMA_GRP_OUT_RANGE (-3)</td>
<td>Specified control group out of range</td>
</tr>
</tbody>
</table>

In case executing the API in each of the following condition, because the system is busy, correct conversion result cannot be acquired, and then E_KINEMA_FAILURE(-1) is returned to “Return Value”.

- When setting a parameter
- When setting a tool file
- When setting an user coordinate
- When setting a calibration

When E_KINEMA_FAILURE(-1) is returned to “Return Value”, try executing the API again after an interval.
mpConvAngleToPulse

Converts an angle position of the specified control group to a pulse position.

- **Syntax**

  ```c
  int mpConvAngleToPulse(
    unsigned int grp_no,
    long angle[MP_GRP_AXES_NUM],
    long pulse[MP_GRP_AXES_NUM]
  );
  ```

- **Parameter**

  - **[grp_no]**
    Control group number
    Acquire the control group number by mpCtrlGrpId2GrpNo().

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>First control group</td>
</tr>
<tr>
<td>1</td>
<td>Second control group</td>
</tr>
<tr>
<td>2</td>
<td>Third control group</td>
</tr>
<tr>
<td>3</td>
<td>Fourth control group</td>
</tr>
</tbody>
</table>

  - **[angle]**
    Angle (degree) position
    Syntax: `#define MP_GRP_AXES_NUM (8)`

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>angle[0]</td>
<td>1st axis (S) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>angle[1]</td>
<td>2nd axis (L) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>angle[2]</td>
<td>3rd axis (U) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>angle[3]</td>
<td>4th axis (R) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>angle[4]</td>
<td>5th axis (B) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>angle[5]</td>
<td>6th axis (T) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>angle[6]</td>
<td>7th axis (E) angle value (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>angle[7]</td>
<td>8th axis (W) angle value (unit: 0.0001 deg)</td>
</tr>
</tbody>
</table>
mpConvAngleToPulse

**[pulse]**
Pulse position

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pulse[0]</td>
<td>1st axis (S) pulse value</td>
</tr>
<tr>
<td>pulse[1]</td>
<td>2nd axis (L) pulse value</td>
</tr>
<tr>
<td>pulse[2]</td>
<td>3rd axis (U) pulse value</td>
</tr>
<tr>
<td>pulse[3]</td>
<td>4th axis (R) pulse value</td>
</tr>
<tr>
<td>pulse[4]</td>
<td>5th axis (B) pulse value</td>
</tr>
<tr>
<td>pulse[5]</td>
<td>6th axis (T) pulse value</td>
</tr>
<tr>
<td>pulse[6]</td>
<td>7th axis (E) pulse value</td>
</tr>
<tr>
<td>pulse[7]</td>
<td>8th axis (W) pulse value</td>
</tr>
</tbody>
</table>

**Return Value**

- **0**: Normal end
- **One of the following negative values**: Error

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_KINEMA_FAILURE (-1)</td>
<td>Failed in conversion</td>
</tr>
<tr>
<td>E_KINEMA_CONV_IMPOSSIBLE (-2)</td>
<td>Conversion impossible</td>
</tr>
<tr>
<td>E_KINEMA_GRP_OUT_RANGE (-3)</td>
<td>Specified control group out of range</td>
</tr>
</tbody>
</table>

In case executing the API in each of the following condition, because the system is busy, correct conversion result cannot be acquired, and then E_KINEMA_FAILURE(-1) is returned to “Return Value”.

- When setting a parameter
- When setting a tool file
- When setting an user coordinate
- When setting a calibration

When E_KINEMA_FAILURE(-1) is returned to “Return Value”, try executing the API again after an interval.

---

**NOTE**
mpConvFBPulseToPulse

Converts a feedback pulse to an arithmetic pulse.

**Syntax**

```c
int mpConvFBPulseToPulse(
    unsigned int   grp_no,
    long           fbpulse[MP_GRP_AXES_NUM],
    long           pulse[MP_GRP_AXES_NUM]
);
```

**Parameter**

**[grp_no]**
Control group number
Acquire the control group number by mpCtrlGrpId2GrpNo().

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>First control group</td>
</tr>
<tr>
<td>1</td>
<td>Second control group</td>
</tr>
<tr>
<td>2</td>
<td>Third control group</td>
</tr>
<tr>
<td>3</td>
<td>Fourth control group</td>
</tr>
</tbody>
</table>

**[fbpulse]**
Feedback pulse position
Syntax:#define MP_GRP_AXES_NUM (8)

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pulse[0]</td>
<td>1st axis (S) pulse value</td>
</tr>
<tr>
<td>pulse[1]</td>
<td>2nd axis (L) pulse value</td>
</tr>
<tr>
<td>pulse[2]</td>
<td>3rd axis (U) pulse value</td>
</tr>
<tr>
<td>pulse[3]</td>
<td>4th axis (R) pulse value</td>
</tr>
<tr>
<td>pulse[4]</td>
<td>5th axis (B) pulse value</td>
</tr>
<tr>
<td>pulse[5]</td>
<td>6th axis (T) pulse value</td>
</tr>
<tr>
<td>pulse[6]</td>
<td>7th axis (E) pulse value</td>
</tr>
<tr>
<td>pulse[7]</td>
<td>8th axis (W) pulse value</td>
</tr>
</tbody>
</table>
## Kinematics API

### mpConvFBPulseToPulse

**[pulse]**

Pulse position

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pulse[0]</td>
<td>1st axis (S) pulse value</td>
</tr>
<tr>
<td>pulse[1]</td>
<td>2nd axis (L) pulse value</td>
</tr>
<tr>
<td>pulse[2]</td>
<td>3rd axis (U) pulse value</td>
</tr>
<tr>
<td>pulse[3]</td>
<td>4th axis (R) pulse value</td>
</tr>
<tr>
<td>pulse[4]</td>
<td>5th axis (B) pulse value</td>
</tr>
<tr>
<td>pulse[5]</td>
<td>6th axis (T) pulse value</td>
</tr>
<tr>
<td>pulse[6]</td>
<td>7th axis (E) pulse value</td>
</tr>
<tr>
<td>pulse[7]</td>
<td>8th axis (W) pulse value</td>
</tr>
</tbody>
</table>

### Return Value

- **0**: Normal end
- One of the following negative values: Error

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_KINEMA_FAILURE (-1)</td>
<td>Failed in conversion</td>
</tr>
<tr>
<td>E_KINEMA_CONV_IMPOSSIBLE (-2)</td>
<td>Conversion impossible</td>
</tr>
<tr>
<td>E_KINEMA_GRP_OUT_RANGE (-3)</td>
<td>Specified control group out of range</td>
</tr>
</tbody>
</table>

In case executing the API in each of the following condition, because the system is busy, correct conversion result cannot be acquired, and then E_KINEMA_FAILURE(-1) is returned to “Return Value”.

- When setting a parameter
- When setting a tool file
- When setting an user coordinate
- When setting a calibration

When E_KINEMA_FAILURE(-1) is returned to “Return Value”, try executing the API again after an interval.
mpMakeFrame

Creates a homogeneous transformation matrix from a three-point position.

- **Syntax**

```c
int mpMakeFrame(
    MP_XYZ* org_vector,
    MP_XYZ* x_vector,
    MP_XYZ* y_vector,
    MP_FRAME* frame
);
```

- **Description**

“org_vector” indicates the home position and “x_vector” indicates a point on the x-axis. By defining “y_vector”, which is a point on the y-axis side, the directions of both y-axis and z-axis are determined.

- **Parameter**

**[org_vector]**

Pointer to the home position vector structure of the coordinate (input)

- **MP_XYZ**
  Position vector structure
  Syntax: typedef struct
    double x;
    double y;
    double z;
  } MP_XYZ;

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>X-component value (unit: micron)</td>
</tr>
<tr>
<td>y</td>
<td>Y-component value (unit: micron)</td>
</tr>
<tr>
<td>z</td>
<td>Z-component value (unit: micron)</td>
</tr>
</tbody>
</table>
mpMakeFrame

[x_vector]
Pointer to the position vector structure on the x-axis of the coordinate (input)

[y_vector]
Pointer to the position vector structure on the y-axis of the coordinate (input)

[frame]
Pointer to the homogeneous transformation matrix structure (output)

- MP_FRAME
  Homogeneous transformation matrix structure
  Syntax: typedef struct
  {
    double nx, ny, nz;
    double ox, oy, oz;
    double ax, ay, az;
    double px, py, pz;
  } MP_FRAME;

### Return Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>E_KINEMA_FAILURE</td>
<td>(-1) Failed in conversion</td>
</tr>
</tbody>
</table>
mpInvFrame

Create an inverse matrix of an optional homogeneous transformation matrix.

- **Syntax**

  ```c
  int mpInvFrame(
    MP_FRAME* org_frame,
    MP_FRAME* frame
  );
  ```

- **Parameter**

  `[org_frame]`
  
  Pointer to the homogeneous transformation matrix structure (input)

  - `MP_FRAME`
    
    homogeneous transformation matrix structure
    
    Syntax: `typedef struct {
      double nx, ny, nz;
      double ox, oy, oz;
      double ax, ay, az;
      double ox, py, pz;
    } MP_FRAME;`

    | Member | Description                               |
    |--------|-------------------------------------------|
    | nx     | normal vector X-component value           |
    | ny     | normal vector Y-component value           |
    | nz     | normal vector Z-component value           |
    | ox     | orientation vector X-component value      |
    | oy     | orientation vector Y-component value      |
    | oz     | orientation vector Z-component value      |
    | ax     | approach vector X-component value         |
    | ay     | approach vector Y-component value         |
    | az     | approach vector Z-component value         |
    | px     | position vector X-component value (unit: micron) |
    | py     | position vector Y-component value (unit: micron) |
    | pz     | position vector Z-component value (unit: micron) |
Kinematics API

mpInvFrame

[frame]
Pointer to the homogeneous transformation matrix structure (output)

Return Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_KINEMA_FAILURE</td>
<td>Failed in conversion</td>
</tr>
</tbody>
</table>

One of the following negative values

- Normal end
- Error
mpRotFrame

Finds a rotated homogeneous transformation matrix around an optional vector.

**Syntax**

```c
int mpRotFrame(
    MP_FRAME* org_frame,
    double angle,
    MP_XYZ* vector,
    MP_FRAME* frame
);
```

**Parameter**

- **[org_frame]**
  Pointer to the homogeneous transformation matrix structure (input)

  - MP_FRAME
    homogeneous transformation matrix structure
  
  ```c
  typedef struct {
    double nx, ny, nz;
    double ox, oy, oz;
    double ax, ay, az;
    double px, py, pz;
  } MP_FRAME;
  ```

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nx</td>
<td>normal vector X-component value</td>
</tr>
<tr>
<td>ny</td>
<td>normal vector Y-component value</td>
</tr>
<tr>
<td>nz</td>
<td>normal vector Z-component value</td>
</tr>
<tr>
<td>ox</td>
<td>orientation vector X-component value</td>
</tr>
<tr>
<td>oy</td>
<td>orientation vector Y-component value</td>
</tr>
<tr>
<td>oz</td>
<td>orientation vector Z-component value</td>
</tr>
<tr>
<td>ax</td>
<td>approach vector X-component value</td>
</tr>
<tr>
<td>ay</td>
<td>approach vector Y-component value</td>
</tr>
<tr>
<td>az</td>
<td>approach vector Z-component value</td>
</tr>
<tr>
<td>px</td>
<td>position vector X-component value (unit: micron)</td>
</tr>
<tr>
<td>py</td>
<td>position vector Y-component value (unit: micron)</td>
</tr>
<tr>
<td>pz</td>
<td>position vector Z-component value (unit: micron)</td>
</tr>
</tbody>
</table>

- **[angle]**
  Rotation angle (degree)
Kinematics API

mpRotFrame

**[vector]**

Pointer to the position vector structure (input)

- **MP_XYZ**
  Position vector structure
  Syntax: typedef struct
  ```c
  { double x;
    double y;
    double z;
  } MP_XYZ;
  ```

**[frame]**

Pointer to the homogeneous transformation matrix structure (output)

### Return Value

0 Normal end

One of the following negative values Error

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_KINEMA_FAILURE</td>
<td>(-1) Failed in conversion</td>
</tr>
</tbody>
</table>
mpMulFrame

Finds a product of two optional homogeneous transformation matrix.

- **Syntax**

```c
int mpMulFrame(
    MP_FRAME* frame1,
    MP_FRAME* frame2,
    MP_FRAME* prod
);
```

- **Description**

Returns the multiplication of frame2 and frame1 to prod.

- **Parameter**

  ```org_frame
  ```

  Pointer to the homogeneous transformation matrix structure (input)

  - **MP_FRAME**
    homogeneous transformation matrix structure
    Syntax: typedef struct
    ```
    {
        double nx, ny, nz;
        double ox, oy, oz;
        double ax, ay, az;
        double px, py, pz;
    }    MP_FRAME;
    ```

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nx</td>
<td>normal vector X-component value</td>
</tr>
<tr>
<td>ny</td>
<td>normal vector Y-component value</td>
</tr>
<tr>
<td>nz</td>
<td>normal vector Z-component value</td>
</tr>
<tr>
<td>ox</td>
<td>orientation vector X-component value</td>
</tr>
<tr>
<td>oy</td>
<td>orientation vector Y-component value</td>
</tr>
<tr>
<td>oz</td>
<td>orientation vector Z-component value</td>
</tr>
<tr>
<td>ax</td>
<td>approach vector X-component value</td>
</tr>
<tr>
<td>ay</td>
<td>approach vector Y-component value</td>
</tr>
<tr>
<td>az</td>
<td>approach vector Z-component value</td>
</tr>
<tr>
<td>px</td>
<td>position vector X-component value (unit: micron)</td>
</tr>
<tr>
<td>py</td>
<td>position vector Y-component value (unit: micron)</td>
</tr>
<tr>
<td>pz</td>
<td>position vector Z-component value (unit: micron)</td>
</tr>
</tbody>
</table>
Kinematics API

mpMulFrame

[frame2]
Pointer to the homogeneous transformation matrix structure (input)

[prod]
Pointer to the homogeneous transformation matrix structure (output)

Return Value

0 Normal end

One of the following negative values Error

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_KINEMA_FAILURE</td>
<td>(-1) Failed in conversion</td>
</tr>
</tbody>
</table>
mpZYXeulerToFrame

Converts an optional XYZ eulerian expression cartesian value to a homogeneous transformation matrix.

- **Syntax**

  ```
  int mpZYXeulerToFrame(
      MP_COORD* coord,
      MP_FRAME* frame
  );
  ```

- **Parameter**

  **[coord]**

  Pointer to the cartesian coordinate position structure (input)

  - **MP_COORD**
    Cartesian coordinate position structure
  
  Syntax: typedef struct
  ```
  {
      long x, y, z;
      long rx, ry, rz;
  } MPCOORD;
  ```

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>X-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>y</td>
<td>Y-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>z</td>
<td>Z-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>rx</td>
<td>Wrist angle Rx (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>ry</td>
<td>Wrist angle Ry (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>rz</td>
<td>Wrist angle Rz (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>ex1</td>
<td>Elbow angle Re (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>ex2</td>
<td>Reserved by manufacturer</td>
</tr>
</tbody>
</table>
Kinematics API

mpZYXeulerToFrame

[frame]

Pointer to the homogeneous transformation matrix structure (output)

- MP_FRAME
  Homogeneous transformation matrix structure
  Syntax: typedef struct
  {
    double nx, ny, nz;
    double ox, oy, oz;
    double ax, ay, az;
    double px, py, pz;
  } MP_FRAME;

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nx</td>
<td>normal vector X-component value</td>
</tr>
<tr>
<td>ny</td>
<td>normal vector Y-component value</td>
</tr>
<tr>
<td>nz</td>
<td>normal vector Z-component value</td>
</tr>
<tr>
<td>ox</td>
<td>orientation vector X-component value</td>
</tr>
<tr>
<td>oy</td>
<td>orientation vector Y-component value</td>
</tr>
<tr>
<td>oz</td>
<td>orientation vector Z-component value</td>
</tr>
<tr>
<td>ax</td>
<td>approach vector X-component value</td>
</tr>
<tr>
<td>ay</td>
<td>approach vector Y-component value</td>
</tr>
<tr>
<td>az</td>
<td>approach vector Z-component value</td>
</tr>
<tr>
<td>px</td>
<td>position vector X-component value (unit: micron)</td>
</tr>
<tr>
<td>py</td>
<td>position vector Y-component value (unit: micron)</td>
</tr>
<tr>
<td>pz</td>
<td>position vector Z-component value (unit: micron)</td>
</tr>
</tbody>
</table>

■ Return Value

0 Normal end

One of the following negative values Error

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_KINEMA_FAILURE (-1)</td>
<td>Failed in conversion</td>
</tr>
</tbody>
</table>
mpFrameToZYXeuler

Converts an optional XYZ eulerian expression cartesian value to a homogeneous transformation matrix.

### Syntax

```c
int mpFrameToZYXeuler(
    MP_FRAME* frame,
    MPCOORD* coord,
);
```

### Parameter

**[frame]**

Pointer to the homogeneous transformation matrix structure (input)

- **MP_FRAME**
  Homogeneous transformation matrix structure
  Syntax: typedef struct
    {
        double nx, ny, nz;
        double ox, oy, oz;
        double ax, ay, az;
        double px, py, pz;
    } MP_FRAME;

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nx</td>
<td>normal vector X-component value</td>
</tr>
<tr>
<td>ny</td>
<td>normal vector Y-component value</td>
</tr>
<tr>
<td>nz</td>
<td>normal vector Z-component value</td>
</tr>
<tr>
<td>ox</td>
<td>orientation vector X-component value</td>
</tr>
<tr>
<td>oy</td>
<td>orientation vector Y-component value</td>
</tr>
<tr>
<td>oz</td>
<td>orientation vector Z-component value</td>
</tr>
<tr>
<td>ax</td>
<td>approach vector X-component value</td>
</tr>
<tr>
<td>ay</td>
<td>approach vector Y-component value</td>
</tr>
<tr>
<td>az</td>
<td>approach vector Z-component value</td>
</tr>
<tr>
<td>px</td>
<td>position vector X-component value (unit: micron)</td>
</tr>
<tr>
<td>py</td>
<td>position vector Y-component value (unit: micron)</td>
</tr>
<tr>
<td>pz</td>
<td>position vector Z-component value (unit: micron)</td>
</tr>
</tbody>
</table>
Kinematics API

mpFrameToZYXeuler

[coord]
Pointer to the cartesian coordinate position structure (output)

- **MP_COORD**
  Cartesian coordinate position structure
  Syntax: `typedef struct {
    long x, y, z;
    long rx, ry, rz;
    long ex1, ex2;
  } MP_COORD;`

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>X-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>y</td>
<td>Y-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>z</td>
<td>Z-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>rx</td>
<td>Wrist angle Rx (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>ry</td>
<td>Wrist angle Ry (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>rz</td>
<td>Wrist angle Rz (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>ex1</td>
<td>Elbow angle Re (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>ex2</td>
<td>Reserved by manufacturer</td>
</tr>
</tbody>
</table>

- **Return Value**

  0 Normal end

  One of the following negative values Error

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_KINEMA_FAILURE</td>
<td>(-1) Failed in conversion</td>
</tr>
</tbody>
</table>
mpCrossProduct

Finds an outer product or two vectors.

- **Syntax**

  ```c
  int mpCrossProduct(
    MP_XYZ* vector1,
    MP_XYZ* vector2,
    MP_XYZ* prod
  );
  ```

- **Description**

  Returns the multiplication of vector 2 and vector1 to prod.

- **Parameter**

  - `[vector1]`  
    Pointer to the position vector structure (input)
    - `MP_XYZ`
      Position vector structure
      Syntax: `typedef struct {
        double x;
        double y;
        double z;
      } MP_XYZ;`
      | Member | Description                  |
      |--------|-----------------------------|
      | x      | X-component value (unit: micron) |
      | y      | Y-component value (unit: micron) |
      | z      | Z-component value (unit: micron) |

  - `[vector2]`  
    Pointer to the position vector structure (input)

  - `[prod]`  
    Pointer to the position vector structure (output)

- **Return Value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_KINEMA_FAILURE (-1)</td>
<td>Failed in conversion</td>
</tr>
</tbody>
</table>

  Normal end

  One of the following negative values

  Error
mpInnerProduct

Finds an inner product of two vectors.

- **Syntax**

```c
int mpInnerProduct(
    MP_XYZ* vector1,
    MP_XYZ* vector2,
    double* prod
);
```

- **Parameter**

  **[vector1]**
  
  Pointer to the position vector structure (input)
  
  - **MP_XYZ**
    
    Position vector structure
    
    Syntax: typedef struct
    
    ```
    {
        double x;
        double y;
        double z;
    } MP_XYZ;
    ```

  **[vector2]**
  
  Pointer to the position vector structure (input)

  **[prod]**
  
  Pointer to the cosine (output)

- **Return Value**

  0  
  
  Normal end

  One of the following negative values  
  
  Error

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_KINEMA_FAILURE</td>
<td>(-1) Failed in conversion</td>
</tr>
</tbody>
</table>
17 ForceSensor Monitor/Control API

To use the force control, the MotoFit function (optional) is necessary.

**mpFcsStartMeasuring**

This routine resets the 6-axis force sensor, and starts measurement.

- **Syntax**

  ```c
  MP_FCS_HANDLE mpInitializeForceSens
  (  
    MP_FCS_ROB_ID rob_id, /* target robot ID specification */
    Int reset_time, /* averaging time */
    MP_FCS_OFFSET_DATA offset_data /* measurement result destination */
  )
  ```

- **Description**

  The execution of this API resets the offset of the force sensor, and starts the signal output of the sensor coordinate system. If a move command (includes the axis operation) is output to the specified robot, this API returns an error. Also, this API does not return the control to the caller during the specified averaging time (until the result output is completed) because this API is a synchronous type. When the force sensor is in a condition that "the offset value calculation is completed" the following system outputs are turned ON:
  
  - Robot 1 (MP_FCS_R1ID): #51110
  - Robot 2 (MP_FCS_R2ID): #51111

- **Parameter**

  - **[rob_id]**
    Specifies the targeted robot (ID) for control (measurement) by using the name defined the following type.

    ```c
    typedef enum {
      MP_FCS_R1ID = 0,
      MP_FCS_R2ID
    } MP_FCS_ROB_ID;
    ```

  - **[reset_time]**
    Specifies the averaging time to calculate the offset value from 10 to 1000 (msec). The value less than the minimum value (10 < reset_time) or the value more than the maximum value (1000 > reset_time) is rounded to this range.
[offset_data]
Returns the offset value of the measurement result. Store it in the 6-
element array of the int type.

- Return Value

0 or more Normal end
The handle value is returned.

Less than 0 Error
One of the following values is returned according to the error content.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_FCS_SPECIFIED_ROB_ID</td>
<td>-1 Robot group ID designation error</td>
</tr>
<tr>
<td>E_FCS_CTRL_GRP_NOT_EXIST</td>
<td>-2 The specified control group does not exist.</td>
</tr>
<tr>
<td>E_FCS_JOG_BUSY</td>
<td>-4 This API cannot be executed during JOG operation.</td>
</tr>
<tr>
<td>E_FCS_MOVE_BUSY</td>
<td>-5 This API cannot be executed while MOV command is executed.</td>
</tr>
<tr>
<td>E_FCS_ARGUMENT_RANGE_OUTSIDE</td>
<td>-17 The value is outside the range.</td>
</tr>
</tbody>
</table>
mpFcsGetForceData

Retrieves the force data in the specified coordinate system.

- **Syntax**

  ```c
  int mpFcsGetForceData
  (     
    MP_FCS_R OB_ID rob_id, /* target robot ID specification */
    int coord_type, /* coordinates specification */
    int uf_no, /* number of user coordinates file */
    MP_FCS_SENS_DATA sens_data /* force data result destination */
  )
  ```

- **Description**

  The execution of this API retrieves the force data in the specified coordinate system of the 6-axis force sensor. However, the force data cannot be retrieved under the following condition:
  
  - mpFcsStartMeasuring() is not completed normally.

- **Parameter**

  - **[rob_id]**
    Specifies the targeted robot (ID) for control (retrieving).
  
  - **[coord_type]**
    Specifies the destination coordinate systems (base coordinate, robot coordinate, tool coordinate, user coordinate) from the force data in the sensor coordinate system.
  
  - **[uf_no]**
    Specifies the user coordinate file number (0 to 15) when the user coordinate is specified as the destination coordinate system.
17 ForceSensor Monitor/Control API

mpFcsGetForceData

### [sens_data]

Returns the force data of the specified coordinate system. It is expected to be stored in the 6-element array of the int type.

<table>
<thead>
<tr>
<th>Specified Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sens_data[0]</td>
<td>Fx, unit (0.1N)</td>
</tr>
<tr>
<td>sens_data[1]</td>
<td>Fy, unit (0.1N)</td>
</tr>
<tr>
<td>sens_data[2]</td>
<td>Fz, unit (0.1N)</td>
</tr>
<tr>
<td>sens_data[3]</td>
<td>Mx, unit (0.01Nm)</td>
</tr>
<tr>
<td>sens_data[4]</td>
<td>My, unit (0.01Nm)</td>
</tr>
<tr>
<td>sens_data[5]</td>
<td>Mz, unit (0.01Nm)</td>
</tr>
</tbody>
</table>

#### Return Value

**0 or more** Normal end

The handle value is returned.

**Less than 0** Error

One of the following values is returned according to the error content.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_FCS_SPECIFIED_ROB_ID</td>
<td>-1 Robot group ID designation error</td>
</tr>
</tbody>
</table>
mpFcsStartImp

Sets the parameters necessary for the impedance control, and starts the impedance control.

- **Syntax**

  ```c
  LONG mpSetImpParam
  (  
    MP_FCS_R OB_ID rob_id, /* target robot ID specification */
    MP_FCS_IMP_COEFF m, /* inertial coefficient */
    MP_FCS_IMP_COEFF d, /* viscosity coefficient */
    MP_FCS_IMP_COEFF k, /* spring coefficient */
    int coord_type, /* coordinates specification */
    int uf_no, /* number of user coordinates file */
    BITSTRING cart_axes, /* specified [X, Y, Z, Rx, Ry, Rz] bits */
    BITSTRING option_ctrl /* option control bits */
  )
  ```

- **Description**

  The execution of this API updates the control parameters of the impedance model on the servo CPU necessary for the high-speed impedance control executed with servo CPU. This API can operate even the impedance control has already been effective. This API returns an error when mpFcsStartMeasuring() API does not complete the calculation of the offset value of the force sensor yet. When the execution of this API changes the condition to that "the impedance control is being operated," the following system outputs are turned ON:

  - Robot 1 (MP_FCS_R1ID): #51100
  - Robot 2 (MP_FCS_R2ID): #51101

- **Parameter**

  - `[rob_id]`
    
    Specifies the targeted robot (ID) for control (start).

  - `[m]`
    
    Specifies the inertia coefficients in the 6-element array of the int type.

<table>
<thead>
<tr>
<th>Specified Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M[0]</td>
<td>Inertia coefficient (X direction), unit (Ns²/m²)</td>
</tr>
<tr>
<td>M[1]</td>
<td>Inertia coefficient (Y direction), unit (Ns²/m²)</td>
</tr>
<tr>
<td>M[2]</td>
<td>Inertia coefficient (Z direction), unit (Ns²/m²)</td>
</tr>
<tr>
<td>M[3]</td>
<td>Inertia coefficient (Rx direction)</td>
</tr>
<tr>
<td>M[4]</td>
<td>Inertia coefficient (Ry direction)</td>
</tr>
<tr>
<td>M[5]</td>
<td>Inertia coefficient (Rz direction)</td>
</tr>
</tbody>
</table>
[d]
Specifies the viscous coefficients in the 6-element array of the int type.

<table>
<thead>
<tr>
<th>Specified Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D[0]</td>
<td>Viscous coefficient (X direction), unit (Ns/m)</td>
</tr>
<tr>
<td>D[1]</td>
<td>Viscous coefficient (Y direction), unit (Ns/m)</td>
</tr>
<tr>
<td>D[2]</td>
<td>Viscous coefficient (Z direction), unit (Ns/m)</td>
</tr>
<tr>
<td>D[3]</td>
<td>Viscous coefficient (Rx direction)</td>
</tr>
<tr>
<td>D[4]</td>
<td>Viscous coefficient (Ry direction)</td>
</tr>
<tr>
<td>D[5]</td>
<td>Viscous coefficient (Rz direction)</td>
</tr>
</tbody>
</table>

[k]
Specifies the spring coefficients in the 6-element array of the int type.

<table>
<thead>
<tr>
<th>Specified Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>K[0]</td>
<td>Spring coefficient (X direction), unit (N/m)</td>
</tr>
<tr>
<td>K[1]</td>
<td>Spring coefficient (Y direction), unit (N/m)</td>
</tr>
<tr>
<td>K[2]</td>
<td>Spring coefficient (Z direction), unit (N/m)</td>
</tr>
<tr>
<td>K[3]</td>
<td>Spring coefficient (Rx direction)</td>
</tr>
<tr>
<td>K[4]</td>
<td>Spring coefficient (Ry direction)</td>
</tr>
<tr>
<td>K[5]</td>
<td>Spring coefficient (Rz direction)</td>
</tr>
</tbody>
</table>

[coord_type]
Specifies the control coordinate system.

<table>
<thead>
<tr>
<th>Specified Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The robot coordinate system is specified as the control coordinate system.</td>
</tr>
<tr>
<td>1</td>
<td>The tool coordinate system is specified as the control coordinate system. The tool file number which is currently activated is specified.</td>
</tr>
<tr>
<td>2</td>
<td>The user coordinate system is specified as the control coordinate system.</td>
</tr>
</tbody>
</table>

[uf_no]
Specifies the user coordinate file number (0 to 15) when the user coordinate is specified.

[cart_axes]
Specifies X, Y, Z, Rx, Ry, Rz (d0 to d5) as the valid control axes.

<table>
<thead>
<tr>
<th>Specified Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>d0 bit of Available_axis</td>
<td>Valid bit of the X axis</td>
</tr>
<tr>
<td>d1 bit of Available_axis</td>
<td>Valid bit of the Y axis</td>
</tr>
<tr>
<td>d2 bit of Available_axis</td>
<td>Valid bit of the Z axis</td>
</tr>
<tr>
<td>d3 bit of Available_axis</td>
<td>Valid bit of the Rx axis</td>
</tr>
<tr>
<td>d4 bit of Available_axis</td>
<td>Valid bit of the Ry axis</td>
</tr>
<tr>
<td>d5 bit of Available_axis</td>
<td>Valid bit of the Rz axis</td>
</tr>
</tbody>
</table>
**[option_ctrl]**

Specifies the optional controls.

<table>
<thead>
<tr>
<th>Specified Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>d0 bit of Option_ctr_req</td>
<td>A request to execute the contact stability process</td>
</tr>
<tr>
<td>d1 bit of Option_ctr_req</td>
<td>A request to retain the previous force command value *1</td>
</tr>
</tbody>
</table>

*1 When executing mpFcsStartImp after mpFcsEndImp is executed, the force command value is not retained even if this bit is specified, and the force command value is set to zero.

#### Return Value

- **0 or more** Normal end
  
  The handle value is returned.

- **Less than 0** Error
  
  One of the following values is returned according to the error content

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_FCS_SPECIFIED_ROB_ID</td>
<td>-1 Robot group ID designation error</td>
</tr>
<tr>
<td>E_FCS_RANGE_USER_FILE_NO</td>
<td>-6 The user coordinate number is outside the range.</td>
</tr>
<tr>
<td>E_FCS_UNREGIST_USER_FRAME</td>
<td>-7 The user coordinate is not set.</td>
</tr>
<tr>
<td>F_FCS_AX_IMP_MODE_START</td>
<td>-9 Setting of the impedance control parameter is failed.</td>
</tr>
<tr>
<td>E_FCS_ARGUMENT_RANGE_OUTSIDE</td>
<td>-17 The value is outside the range.</td>
</tr>
</tbody>
</table>
mpFcsSetReferenceForce

Sets the force command value of the impedance control.

- **Syntax**

  ```c
  LONG mpSetReferenceForce
  (  
    MP_FCS_R OB_ID rob_id, /* target robot ID specification */
    MP_FCS_FREF _DATA fref_data /* force data to set */
  )
  ```

- **Description**

  The execution of this API sets the force command value executed on the servo CPU. This API can operate even the impedance control has already been effective.

  This API returns an error when `mpFcsStartImp()` API does not start the impedance control yet. (Confirm the following system outputs: #51100, #51101)

- **Parameter**

  **[rob_id]**

  Specifies the targeted robot (ID) for control (setting).

  **[fref_data]**

  Passes the force command value which is set in the 6-element array of the int type (X, Y, Z, Rx, Ry, Rz).

<table>
<thead>
<tr>
<th>Specified Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fref_data[0]</td>
<td>Force command value of the X-direction axis, unit (N)</td>
</tr>
<tr>
<td>Fref_data[1]</td>
<td>Force command value of the Y-direction axis, unit (N)</td>
</tr>
<tr>
<td>Fref_data[2]</td>
<td>Force command value of the Z-direction axis, unit (N)</td>
</tr>
<tr>
<td>Fref_data[3]</td>
<td>Force command value of the Rx-direction axis, unit (Nm x 0.01)</td>
</tr>
<tr>
<td>Fref_data[4]</td>
<td>Force command value of the Ry-direction axis, unit (Nm x 0.01)</td>
</tr>
<tr>
<td>Fref_data[5]</td>
<td>Force command value of the Rz-direction axis, unit (Nm x 0.01)</td>
</tr>
</tbody>
</table>
Return Value

0 or more Normal end
  The handle value is returned.

Less than 0 Error
  One of the following values is returned according to the error content

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_FCS_SPECIFIED_ROB_ID</td>
<td>Robot group ID designation error</td>
</tr>
<tr>
<td>F_FCS_AX_IMP_MODE_START</td>
<td>Setting of the impedance control parameter failed.</td>
</tr>
</tbody>
</table>
mpFcsEndImp

Ends the impedance control.

- **Syntax**

  ```c
  LONG mpEndImp
  (MP_FCS_ROB_ID rob_id/* specified FCS handle */)
  ```

- **Description**

  When the impedance control is ended, the execution of this API creates the current position again and activates the position deviation monitoring between the command and feedback.

- **Parameter**

  - `[rob_id]`
    Specifies the targeted robot (ID) for control (end).

- **Return Value**

  - 0 or more Normal end
    The handle value is returned.
  - Less than 0 Error
    One of the following values is returned according to the error content

    | Value                              | Description                                      |
    |------------------------------------|--------------------------------------------------|
    | E_FCS_SPECIFIED_ROB_ID             | -1 Robot group ID designation error              |
    | F_FCS_PR_MAKE_FAILURE              | -16 Re-creation of the current position is failed. |
mpMotoSyncIFOpen

Opens the MotomanSync I/F.

- **Syntax**

  ```c
  MP_MS_IF_DESC mpMotoSyncIFOpen
  (   int options /* option */  
  )
  ```

- **Description**

  This routine opens MotomanSync I/F and retrieves descriptors. The retrieved descriptors are identifiers to access to I/F, and used as the parameters of mpMotoSyncIFLock(). Up to six descriptors can be retrieved.

- **Parameter**

  - `[options]`
    Specify 0.

- **Return Value**

  - `!=0` Success
  - `== 0` Failure

  When the return value is "!=0", descriptors of MotomanSync I/F are returned.
mpMotoSyncIFClose

Closes the MotomanSync I/F.

- **Syntax**

  ```c
  int mpMotoSyncIFClose
  (    
      MP_MS_IF_DESC ifd /* MotomanSync I/F descriptor */
  )
  ```

- **Description**

  This routine closes MotomanSync I/F. Hereafter ifd becomes an invalid descriptor and cannot be used as a valid parameter of mpMotoSyncIFLock().

- **Parameter**

  `[ifd]`

  An identifier (descriptor) to access to MotomanSync I/F.
  Passes the valid descriptor which is retrieved by mpMotoSyncIFOpen().

- **Return Value**

  >=0  Success
  == 0  Failure

  When the return value is “>=0”, the number of remaining open descriptors is returned.

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Meaning</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_MP_MS_IFD_ILLEGAL</td>
<td>Specified ifd (MotomanSync I/F descriptor) is wrong.</td>
<td>Specify the valid descriptor retrieved by mpMotoSyncIFOpen().</td>
</tr>
</tbody>
</table>
mpMotoSyncIFLock

Locks the MotomanSync I/F.

- **Syntax**

  ```c
  int mpMotoSyncIFLock
  (  
    MP_MS_IF_DESC ifd, /* MotomanSync I/F descriptor */
    int options, /* option */
    void ** ope, /* MotomanSync I/F command data head pointer */
    void ** mon /* MotomanSync I/F monitor data head pointer */
  )
  ```

- **Description**

  This routine returns the address of MotomanSync I/F. Hereafter I/F scanning is stopped until mpMotoSyncIFUnlock() is called.

MotomanSync I/F can be locked by multiple MotoPlus user tasks using different ifds (descriptors). I/F scanning is stopped until all the lock condition are released. To scan command data quickly or to get the latest condition, shorten the locking time as much as possible.
## MotomanSync IF Access API

### mpMotoSyncIFLock

#### Parameter

- **[ifd]**
  
  An identifier (descriptor) to access to MotomanSync I/F.
  
  Passes a valid descriptor which is retrieved by mpMotoSyncIFOpen().

- **[options]**
  
  Specify 0.

- **[ope]**
  
  Returns the head pointer of MotomanSync I/F command data. When specify NULL pointer, nothing is returned.

- **[mon]**
  
  Returns the head pointer of MotomanSync I/F monitor data. When specify NULL pointer, nothing is returned.

#### Return Value

- `>=0` Success
- `== 0` Failure

When the return value is “`>=0`”, the number of descriptors which are locking MotomanSync I/F is returned.

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Meaning</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_MP_MS_IFD_ILLEGAL</td>
<td>Specified ifd (MotomanSync I/F descriptor) is wrong.</td>
<td>Specify a valid descriptor retrieved by mpMotoSyncIFOpen().</td>
</tr>
</tbody>
</table>
mpMotoSyncIFUnlock

Unlocks the MotomanSync I/F.

Syntax

```c
int mpMotoSyncIFUnlock
  (MP_MS_IF_DESC ifd /* MotomanSync I/F descriptor */)
```

Description

This routine releases MotomanSync I/F and resumes I/F scanning.

Parameter

[ifd]

An identifier (descriptor) to access to MotomanSync I/F.
Passes the valid descriptor which is retrieved by mpMotoSyncIFOpen().

Return Value

- >=0  Success
- == 0  Failure

When the return value is ">=0", the number of remaining descriptors which are locking MotomanSync I/F is returned.

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Meaning</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_MP_MS_IFD_ILLEGAL</td>
<td>Specified ifd (MotomanSync I/F descriptor) is wrong.</td>
<td>Specify the valid descriptor retrieved by mpMotoSyncIFOpen().</td>
</tr>
</tbody>
</table>
## 19 Alarm List

<table>
<thead>
<tr>
<th>Alarm Number</th>
<th>Alarm Name</th>
<th>Subcode</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1020</td>
<td>MOTOPLUS APPLICATION</td>
<td>1</td>
<td>The number of loaded files exceeds the limit.</td>
</tr>
<tr>
<td></td>
<td>LOAD ERROR</td>
<td>2</td>
<td>The memory is insufficient. (Available memory area is less than 3 Mbyte.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>Failed to open the directory.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>Load failure (Failed to open the file.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>Load failure (Undefined symbol)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>Load failure (Others: application overloaded)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>Failed to initialize the API library.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>No user root task</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
<td>Failed to create the user root task.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>Failed to create RAM-Disk.</td>
</tr>
<tr>
<td>4478</td>
<td>MM TASK NO RESPONSE</td>
<td></td>
<td>The process requested from the MotoPlus application to the MM task was not completed within the specified time.</td>
</tr>
<tr>
<td></td>
<td>(MotoPlus)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4479</td>
<td>MOTOPLUS MM TASK</td>
<td></td>
<td>The man-machine task does not run for 3 seconds or more.</td>
</tr>
<tr>
<td></td>
<td>WATCHDOG ERROR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>