YRC1000micro 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
YRC1000micro INSTRUCTIONS
YRC1000micro OPERATOR'S MANUAL
YRC1000micro MAINTENANCE MANUAL
YRC1000micro ALARM CODES (MAJOR ALARMS) (MINOR ALARMS)

The YRC1000micro alarm codes above consist of "MAJOR ALARMS" and "MINOR ALARMS".

Please have the following information available when contacting Yaskawa Customer Support:

- System
- Primary Application
- Software Version (Located on Programming Pendant by selecting: {Main Menu} - {System Info} - {Version})
- Robot Serial Number (Located on robot data plate)
- Robot Sales Order Number (Located on controller data plate)

Part Number: 181300-1CD
Revision: 1
**DANGER**

- This manual explains MotoPlus of the YRC1000micro system. Read this manual carefully and be sure to understand its contents before handling the YRC1000micro. Any matter not described in this manual must be regarded as "prohibited" or "improper".
- General information related to safety are described in "Chapter 1. Safety" of the YRC1000micro INSTRUCTIONS. To ensure correct and safe operation, carefully read "Chapter 1. Safety" of the YRC1000micro INSTRUCTIONS.

**CAUTION**

- In some drawings in this manual, protective covers or shields are removed to show details. Make sure that all the covers or shields are installed in place before operating this product.
- YASKAWA is not responsible for incidents arising from unauthorized modification of its products. Unauthorized modification voids the product warranty.

**NOTICE**

- 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.
Notes for Safe Operation

Read this manual carefully before installation, operation, maintenance, or inspection of the YRC1000micro.

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

**DANGER**
Indicates an imminently hazardous situation which, if not avoided, will result in death or serious injury. Safety Signs identified by the signal word DANGER should be used sparingly and only for those situations presenting the most serious hazards.

**WARNING**
Indicates a potentially hazardous situation which, if not avoided, will result in death or serious injury. Hazards identified by the signal word WARNING present a lesser degree of risk of injury or death than those identified by the signal word DANGER.

**CAUTION**
Indicates a hazardous situation, which if not avoided, could result in minor or moderate injury. It may also be used without the safety alert symbol as an alternative to “NOTICE”.

**NOTICE**
NOTICE is the preferred signal word to address practices not related to personal injury. The safety alert symbol should not be used with this signal word. As an alternative to “NOTICE”, the word “CAUTION” without the safety alert symbol may be used to indicate a message not related to personal injury.

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”.

---

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”.

---
Before operating the manipulator, make sure the servo power is turned OFF by performing the following operations. When the servo power is turned OFF, the SERVO ON LED on the programming pendant is turned OFF.

- Press the emergency stop button on the programming pendant or on the external control device, etc.
- Disconnect the safety plug of the safety fence. (when in the play mode or in the remote mode)

If operation of the manipulator cannot be stopped in an emergency, personal injury and/or equipment damage may result. 

Fig.: Emergency Stop Button

Before releasing the emergency stop, make sure to remove the obstacle or error caused the emergency stop, if any, and then turn the servo power ON.

Failure to observe this instruction may cause unintended movement of the manipulator, which may result in personal injury.

Fig.: Release of Emergency Stop

Observe the following precautions when performing a teaching operation within the manipulator's operating range:

- Be sure to perform lockout by putting a lockout device on the safety fence when going into the area enclosed by the safety fence. In addition, the operator of the teaching operation must display the sign that the operation is being performed so that no other person closes the safety fence.
- View the manipulator from the front whenever possible.
- Always follow the predetermined operating procedure.
- Always keep in mind emergency response measures against the manipulator’s unexpected movement toward a person.
- Ensure a safe place to retreat in case of emergency.

Failure to observe this instruction may cause improper or unintended movement of the manipulator, which may result in personal injury.

Confirm that no person is present in the manipulator's operating range and that the operator is in a safe location before:

- Turning ON the YRC1000micro power
- Moving the manipulator by using the programming pendant
- Running the system in the check mode
- Performing automatic operations

Personal injury may result if a person enters the manipulator’s operating range during operation. Immediately press an emergency stop button whenever there is a problem. The emergency stop button is located on the right of the programming pendant.

Read and understand the Explanation of the Warning Labels before operating the manipulator.
DANGER

• 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 4-14 pin and 5-15 pin of the Safety connector (Safety).

• Upon shipment of the YRC1000micro, this signal is connected by a jumper cable in the dummy connector. To use the signal, make sure to supply 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.

WARNING

• Perform the following inspection procedures prior to conducting manipulator teaching. If there is any problem, immediately take necessary steps to solve it, such as maintenance and repair.
  – Check for a problem in manipulator movement.
  – Check for damage to insulation and sheathing of external wires.

• Always return the programming pendant to the hook on the YRC1000micro cabinet after use.

If the programming pendant is left unattended on the manipulator, on a fixture, or on the floor, etc., the Enable Switch may be activated due to surface irregularities of where it is left, and the servo power may be turned ON. In addition, in case the operation of the manipulator starts, the manipulator or the tool may hit the programming pendant left unattended, which may result in personal injury and/or equipment damage.
## Definition of Terms Used Often in This Manual

The MOTOMAN is the YASKAWA industrial robot product.

The MOTOMAN usually consists of the manipulator, the YRC1000micro controller, manipulator cables, the YRC1000micro programming pendant (optional), and the YRC1000micro 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>YRC1000micro controller</td>
<td>YRC1000micro</td>
</tr>
<tr>
<td>YRC1000micro programming pendant</td>
<td>Programming pendant (optional)</td>
</tr>
<tr>
<td>Cable between the manipulator and the controller</td>
<td>Manipulator cable</td>
</tr>
<tr>
<td>YRC1000micro programming pendant dummy connector</td>
<td>Programming pendant dummy connector (optional)</td>
</tr>
</tbody>
</table>

Descriptions of the programming pendant keys, buttons, and displays are shown as follows:

<table>
<thead>
<tr>
<th>Equipment</th>
<th>Character Keys /Symbol Keys</th>
<th>Manual Designation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming Pendant</td>
<td></td>
<td>The keys which have characters or symbols printed on them are denoted with [, ex. [ENTER]</td>
</tr>
<tr>
<td>Axis Keys /Number Keys</td>
<td>[Axis Key] and [Numeric Key] are generic names for the keys for axis operation and number input.</td>
<td></td>
</tr>
<tr>
<td>Keys pressed simultaneously</td>
<td>When two keys are to be pressed simultaneously, the keys are shown with a &quot;+&quot; sign between them, ex. [SHIFT]+[COORD]</td>
<td></td>
</tr>
<tr>
<td>Mode Key</td>
<td>Three kinds of modes that can be selected by the mode key are denoted as follows: REMOTE, PLAY, or TEACH</td>
<td></td>
</tr>
<tr>
<td>Button</td>
<td>Three buttons on the upper side of the programming pendant are denoted as follows: HOLD button START button EMERGENCY STOP button</td>
<td></td>
</tr>
<tr>
<td>Displays</td>
<td>The menu displayed in the programming pendant is denoted with {. e.g. {JOB}</td>
<td></td>
</tr>
<tr>
<td>PC Keyboard</td>
<td>The name of the key is denoted. e.g. Ctrl key on the keyboard</td>
<td></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 (R) and TM are omitted.
## Contents

1. List of Usable API in Each Mode

2. Task Control API
   - mpCreateTask
   - mpDeleteTask
   - mpTaskSuspend
   - mpTaskResume
   - mpMsgQCreate
   - mpMsgQDelete
   - mpMsgQReceive
   - mpMsgQSend
   - mpErrMsgQRcv
   - mpErrMsgQSnd
   - mpSemBCreate
   - mpSemDelete
   - mpSemTake
   - mpSemGive
   - mpErrSemTake
   - mpTaskDelay
   - mpGetRtc
   - mpGetSysClkRate
   - mpClkAnnounce
   - mpStopWatchCreate
   - mpStopWatchDelete
   - mpStopWatchStart
   - mpStopWatchStop
   - mpStopWatchLap
   - mpStopWatchReset
   - mpStopWatchGetTime
   - mpStopWatchGetLapNum
   - mpStopWatchGetAliveLapNo

3. User Watchdog API
   - mpUsrWdogCreate
   - mpUsrWdogDelete
   - mpUsrWdogStart
   - mpUsrWdogClear

4. Network API
   - mpSocket
   - mpListen
   - mpAccept
   - mpBind
   - mpConnect
   - mpRecv
   - mpRecvFrom
Contents

mpSend ........................................................................................................... 4-7
mpSendTo ....................................................................................................... 4-7
mpClose ........................................................................................................ 4-8
mpHtonl ........................................................................................................... 4-9
mpHtons .......................................................................................................... 4-9
mpNtohl ........................................................................................................... 4-9
mpNtohs .......................................................................................................... 4-9
mpInetAddr .................................................................................................... 4-10
mpInetNtoa .................................................................................................... 4-10
mpInetNtoaB ................................................................................................. 4-10
mpGetsockname ........................................................................................... 4-11
mpGetpeername ........................................................................................... 4-12
mpSetsockopt ............................................................................................... 4-13
mpIoctl ........................................................................................................... 4-14
mpSelect ....................................................................................................... 4-15

5 Memory Pool Control API ............................................................................................................... 5-1
mpMalloc ......................................................................................................... 5-1
mpFree ............................................................................................................ 5-2

6 File Control API ........................................................................................................... .................... 6-1
mpCreate ........................................................................................................ 6-1
mpOpen .......................................................................................................... 6-3
mpRemove ...................................................................................................... 6-5
mpClose ........................................................................................................ 6-6
mpRename ...................................................................................................... 6-7
mpRead ........................................................................................................... 6-8
mpWrite ........................................................................................................... 6-9
mpIoctl ........................................................................................................... 6-10
mpLseek ........................................................................................................ 6-12
mpFstat ......................................................................................................... 6-13
mpStat ........................................................................................................... 6-13
mpOpendedir .................................................................................................. 6-15
mpReaddir ..................................................................................................... 6-16
mpRewinddir ................................................................................................. 6-17
mpClosedir .................................................................................................... 6-18
mpCreatedir .................................................................................................. 6-19
mpRemovedir ................................................................................................ 6-20
mpRenamedir ................................................................................................ 6-21

7 Existing File Access API ................................................................................................................. 7-1
mpLoadFile ..................................................................................................... 7-1
mpSaveFile ..................................................................................................... 7-3
mpRefreshFileList ........................................................................................... 7-5
mpGetFileCount .............................................................................................. 7-6
mpGetFileName .............................................................................................. 7-7
mpFdWriteFile ............................................................................................... 7-8
Contents

8 System Monitor API .......................................................................................................................... 8-1
  mpGetVarData .............................................................................................................................. 8-1
  mpGetSVarInfo ............................................................................................................................. 8-3
  mpReadIO ....................................................................................................................................... 8-5
  mpMonitor ...................................................................................................................................... 8-7
  mpGetPosVarData .......................................................................................................................... 8-9
  mpGetUserVars .............................................................................................................................. 8-11
  mpGetAlarmStatus ......................................................................................................................... 8-15
  mpGetAlarmCode ............................................................................................................................ 8-16
  mpGetMode .................................................................................................................................... 8-17
  mpGetCycle ..................................................................................................................................... 8-18
  mpGetServoPower .......................................................................................................................... 8-19
  mpGetPlayStatus ............................................................................................................................ 8-20
  mpGetMasterJob .............................................................................................................................. 8-21
  mpGetCurJob ................................................................................................................................... 8-23
  mpGetSpecialOpStatus ..................................................................................................................... 8-25
  mpGetJobDate ................................................................................................................................ 8-26
  mpGetCartPos ................................................................................................................................ 8-28
  mpGetCartPosEx .............................................................................................................................. 8-30
  mpGetPulsePos ................................................................................................................................ 8-32
  mpGetRadPos .................................................................................................................................... 8-34
  mpGetRadPosEx ............................................................................................................................... 8-37
  mpGetDegPos .................................................................................................................................. 8-40
  mpGetDegPosEx ............................................................................................................................... 8-43
  mpGetFBPulsePos ............................................................................................................................. 8-46
  mpGetFBPulsePosEx .......................................................................................................................... 8-49
  mpGetServoSpeed ............................................................................................................................ 8-51
  mpGetFBSpeed .................................................................................................................................. 8-53
  mpGetTorque ................................................................................................................................... 8-56
  mpGetTorqueEx .................................................................................................................................. 8-59
  mpGetSysTimes ................................................................................................................................. 8-62
  mpGetSysVersionNo ......................................................................................................................... 8-64
  mpGetJogSpeed ................................................................................................................................ 8-65
  mpGetJogCoord ................................................................................................................................ 8-67
  mpCtrlGrpId2GrpNo ............................................................................................................................ 8-69
  mpGetToolData ................................................................................................................................ 8-71
  mpGetCalendar .................................................................................................................................. 8-73
  mpGetJobStepData .............................................................................................................................. 8-74
  mpGetToolNo .................................................................................................................................... 8-80
  mpGetEncoderTemp ............................................................................................................................ 8-81

9 System Control API .......................................................................................................................... 9-1
  mpPutVarData ................................................................................................................................... 9-1
  mpPutSVarInfo .................................................................................................................................. 9-3
Contents

mpWriteIO ....................................................................................................... 9-4
mpPutPosVarData .......................................................................................... 9-6
mpPutUserVars ............................................................................................... 9-9
mpSetAlarm .................................................................................................. 9-13
mpCancelError .............................................................................................. 9-14
mpResetAlarm .............................................................................................. 9-15
mpSetCycle ................................................................................................... 9-16
mpSetServoPower ........................................................................................ 9-18
mpSetMasterJob ........................................................................................... 9-20
mpSetCurJob ................................................................................................ 9-22
mpStartJob .................................................................................................... 9-24
mpHold .......................................................................................................... 9-26
mpWaitForJobEnd ........................................................................................ 9-28
mpDeleteJob ................................................................................................. 9-30
mpSetJogCoord ............................................................................................ 9-32
mpSetToolNo ................................................................................................ 9-35
mpSetJobPosData ........................................................................................ 9-37
mpParallelShiftJob .................................................................................... 9-41
mpApplicationInfoNotify ................................................................................ 9-45

10    Motion Control API........................................................................................................ .............. 10-1

mpMotStart.................................................................................................... 10-1
mpMotStop .................................................................................................... 10-2
mpMotTargetClear ........................................................................................ 10-3
mpMotTargetSend ........................................................................................ 10-4
mpMotTargetReceive .................................................................................... 10-7
mpMotSetCoord .......................................................................................... 10-10
mpMotSetSpeed ......................................................................................... 10-12
mpMotSetTool ............................................................................................. 10-14
mpMotSetOrigin .......................................................................................... 10-15
mpMotSetTask ............................................................................................ 10-17
mpMotSetSync ............................................................................................ 10-19
mpMotResetSync ......................................................................................... 10-21
mpMotSetConfig ......................................................................................... 10-22
mpMotSetAccuracy ..................................................................................... 10-24
mpMotSetAccel ........................................................................................... 10-26
mpMotSetDecel ........................................................................................... 10-28

11    Motion Monitor/Control API................................................................................................ ......... 11-1

mplMOV ........................................................................................................ 11-1
mpMOVJ ....................................................................................................... 11-6
mpMOVL ..................................................................................................... 11-11
mpPulseMOVJ ............................................................................................ 11-17
mpPulseMOVL ............................................................................................ 11-21
mpManualMOV .......................................................................................... 11-26

12    Sensor Control API ..................................................................................................................... 12-1
Contents

mpReceiveSkillCommand ................................................................. 12-1
mpEndSkillCommandProcess .............................................................. 12-5
mpMeiGetJobExecTask ......................................................................... 12-8
mpMeiGetInterpolation ....................................................................... 12-9
mpMeiGetExecControlGroup ................................................................. 12-10
mpMeiPutCorrPath ............................................................................. 12-12
mpMeiPutForcePathEnd ....................................................................... 12-16
mpMeiPutSpdOverride ......................................................................... 12-18
mpMeiIncrementMove ......................................................................... 12-20

13 Robot Motion Control API ...........................................................................................................13-1
mpExRcsIncrementMove ........................................................................ 13-1

14 Servo Monitor/Control API ..........................................................................................................14-1
mpSvsGetVelTrqFb ............................................................................. 14-1
mpSvsStartTrqLimit ........................................................................... 14-3
mpSvsSetTrqLimit .............................................................................. 14-6
mpSvsEndTrqLimit .............................................................................. 14-10
mpSvsStartTrqCtrl ............................................................................. 14-12
mpSvsSetTrqCtrl ............................................................................... 14-15
mpSvsEndTrqCtrl ............................................................................... 14-18
mpSvsForceInit ................................................................................ 14-20

15 Kinematics API.................................................................................................................................15-1
mpConvAxesToCartPos ........................................................................ 15-1
mpConvCartPosToAxes ....................................................................... 15-4
mpConvPulseToAngle ......................................................................... 15-8
mpConvAngleToPulse ......................................................................... 15-10
mpConvFBPulseToPulse ...................................................................... 15-12
mpMakeFrame .................................................................................... 15-14
mplInvFrame ..................................................................................... 15-16
mpRotFrame ...................................................................................... 15-18
mpMulFrame ..................................................................................... 15-20
mpZYXeulerToFrame .......................................................................... 15-22
mpFrameToZYXeuler .......................................................................... 15-24
mpCrossProduct ............................................................................... 15-26
mplInnerProduct ............................................................................. 15-27

16 Alarm List .................................................................................................................................16-1
## 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>Not in operation</td>
<td>In operation</td>
<td>Not in operation</td>
<td>In operation</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>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>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>mpSocket</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpListen</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpAccept</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpBind</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpConnect</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpRecv</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpRecvFrom</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpSend</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpSendTo</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpClose</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpHtonl</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>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>
<td>-------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Not in operation</td>
<td>In operation</td>
<td>Not in operation</td>
</tr>
<tr>
<td>Network API</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpHtons</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpNtohl</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpNtohs</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpInetAddr</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpInetNtoa</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpInetNtoaB</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetsockopt</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetpeername</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpSetsockopt</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpIoctl</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpSelect</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Memory pool control</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpMalloc</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpFree</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>File control API</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpCreate</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpOpen</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpRemove</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpClose</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpRename</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpRead</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpWrite</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpIoctl</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpLseek</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpFstat</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpStat</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpOpendir</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpReaddir</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpRewinddir</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpClosedir</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Existing file access</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpLoadFile</td>
<td>○</td>
<td>2010</td>
<td>2080</td>
<td>2080</td>
</tr>
<tr>
<td>mpSaveFile</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpRefreshFileList</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetFileCount</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetFileName</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpFdWriteFile</td>
<td>○</td>
<td>2010</td>
<td>2080</td>
<td>2080</td>
</tr>
<tr>
<td>mpFdReadFile</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpFdGetJobList</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</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>Teach mode</th>
<th>Play mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Not in operation</td>
<td>In operation</td>
<td>Not in operation</td>
</tr>
<tr>
<td>mpGetVarData</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetSVarInfo</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpReadIO</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpMonitor</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetPosVarData</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetUserVars</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetAlarmStatus</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetAlarmCode</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetMode</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetCycle</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetServoPower</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetPlayStatus</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetMasterJob</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetCurJob</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetSpecialOpStatus</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetJobDate</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetCartPos</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetCartPosEx</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetPulsePos</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetRadPos</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetRadPosEx</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetDegPos</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetDegPosEx</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetFBPulsePos</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetFBPulsePosEx</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetServoSpeed</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetFBSpeed</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetTorque</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetTorqueEx</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetSysTimes</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetSysVersionNo</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetJogSpeed</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetJogCoord</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpCtrlGrpId2GrpNo</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetToolData</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetJobStepData</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetToolNo</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpGetEncoderTemp</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>API Name</td>
<td>Not when an alarm or error occurs</td>
<td>Teach mode</td>
<td>Play mode</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>mpPutVarData</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpPutSVarInfo</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpWriteIO</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpPutPosVarData</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpPutUserVars</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpSetAlarm</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpCancelError</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpResetAlarm</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpSetCycle</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpSetServoPower</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpSetMasterJob</td>
<td>○</td>
<td>2010</td>
<td>○</td>
</tr>
<tr>
<td>mpSetCurJob</td>
<td>○</td>
<td>2010</td>
<td>○</td>
</tr>
<tr>
<td>mpStartJob</td>
<td>2080</td>
<td>2080</td>
<td>○</td>
</tr>
<tr>
<td>mpHold</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpWaitForJobEnd</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>mpDeleteJob</td>
<td>○</td>
<td>2010</td>
<td>2080</td>
</tr>
<tr>
<td>mpSetJogCoord</td>
<td>○</td>
<td>2010</td>
<td>2080</td>
</tr>
<tr>
<td>mpSetToolNo</td>
<td>○</td>
<td>2010</td>
<td>2080</td>
</tr>
<tr>
<td>mpSetJobPosData</td>
<td>○</td>
<td>2010</td>
<td>2080</td>
</tr>
<tr>
<td>mpParallelShiftJob</td>
<td>○</td>
<td>2010</td>
<td>2080</td>
</tr>
<tr>
<td>mpApplicationInfoNotify</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

System control API

Motion control API

Motion monitor/ control API
## 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></td>
<td>Teach mode</td>
<td>Play mode</td>
</tr>
<tr>
<td></td>
<td>In operation</td>
<td>Not in operation</td>
</tr>
<tr>
<td></td>
<td>Not in operation</td>
<td>In operation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor control API</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpReceiveSkillCommand</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>mpEndSkillCommandProcess</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>mpMeiGetJobExecTask</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>mpMeiGetInterpolation</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>mpMeiGetExecControlGroup</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>mpMeiPutCorrPath</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>mpMeiPutForcePathEnd</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>mpMeiPutSpdOverride</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>mpMeiIncrementMove</td>
<td>×</td>
<td>•</td>
</tr>
<tr>
<td>Robot control API</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpExRcsIncrementMove</td>
<td>×</td>
<td>•</td>
</tr>
<tr>
<td>Servo monitor/control API</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpSvsGetVelTrqFb</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>mpSvsStartTrqLimit</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>mpSvsSetTrqLimit</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>mpSvsEndTrqLimit</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>mpSvsStartTrqCtrl</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>mpSvsSetTrqCtrl</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>mpSvsEndTrqCtrl</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>mpSvsForceInit</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Kinematics API</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpConvAxesToCartPos</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>mpConvCartPosToAxes</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>mpConvPulseToAngle</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>mpConvAngleToPulse</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>mpConvFBPulseToPulse</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>mpMakeFrame</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>mpInvFrame</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>mpRotFrame</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>mpMulFrame</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>mpZYXeulerToFrame</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>mpFrameToZYXeuler</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>mpCrossProduct</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>mpInnerProduct</td>
<td>•</td>
<td>•</td>
</tr>
</tbody>
</table>

### Alarm code

- **O**: Executable
- **X**: Inexecutable (API Error)

### Alarm meaning:

- 2060: Alarm or error occurrence
- 2080: Different mode
- 2010: Manipulator in operation
- 3450: Failed to turn on the servo power
mpCreateTask

Creates a task.

Syntax

```c
int mpCreateTask
(  
  int priority, /* priority of new task */
  int stackSize, /* size (bytes) of stack */
  FUNCPTTR 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 is the priority for a special task to receive the notification of the I/O control cycle.

- **MP_PRI_IP_CLK_TAKE**

  This is the priority for a special task to receive the notification of the 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.
2 Task Control API
mpCreateTask

[stack Size]
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 function 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 cycle</td>
<td>1</td>
</tr>
<tr>
<td>MP_PRI_IP_CLK_TAKE</td>
<td>Notification of interpolation cycle</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>

NOTE

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 function, 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 function releases the specified task from the suspended state.

- **Parameter**

  - **[tid]**
    Task ID

- **Return Value**

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

  Nonzero  MSG_Q_ID (Normal end)
  0  Error
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**

  0  Normal end
  -1  Error

- **Note**

  This function 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 function 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)**
  - Wait endlessly until the message arrives

**Return Value**

- **Value >= 0**
  - Data length copied to `buffer` (bytes)
- **-1**
  - Error

To retrieve the error cause, use `mpErrMsgQRcv()`.
2 Task Control API
mpMsgQReceive

Note

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

float mpGetRtc(void)

Return value: 0.001 or more
(unit: msec)
mpMsgQSend

Sends a message to the message queue.

- **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 function 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)**
    Wait endlessly until the message arrives

  - **[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.
Returns Value

0 Normal end
-1 Error
To retrieve the error cause, use mpErrMsgQSnd().

Note
The period of time per tick (msec) can be retrieved by the following function:
float mpGetRtc(void)
Return value: 0.001 or more
(unit: msec)
mpErrMsgQRcv

Retrieves an error when a message queue is received.

- **Syntax**
  
  int mpErrMsgQRcv(void)

- **Description**
  
  This function 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>
  
  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 function retrieves the cause of the error which occurs when mpMsgQSend() is executed. Only when retrieving it immediately after mpMsgQSend() 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 A timeout occurs when other than NO_WAIT is set to timeout of mpMsgQSend().</td>
</tr>
<tr>
<td>E_NONEXIST</td>
<td>0x00000002 The message queue was deleted</td>
</tr>
<tr>
<td>E_LIMIT_EXCEEDED</td>
<td>0x00000003 A timeout occurs when NO_WAIT is set to timeout of mpMsgQSend(), or, a value larger than maxMsgLength of mpMsgQCreate() is set to nBytes of mpMsgQSend().</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>
  int foo(void)
  {
    if (mpMsgQSend(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;
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);
}
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 function 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 function 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>Code</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 function 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) To retrieve the cause of the error, use mpErrSemTake().</td>
</tr>
</tbody>
</table>
mpSemGive

Gives a semaphore.

- **Syntax**

  STATUS mpSemGive
  (  
    SEM_ID semId /* semaphore ID to give */
  )

- **Parameter**

  [semId]
  The ID of the semaphore to be given

- **Return Value**

  0  Normal end
  -1  Error (when the semaphore ID is invalid)
mpErrSemTake

Retrieves an error when a semaphore is retrieved.

- **Syntax**
  
  ```c
  int mpErrSemTake(void)
  ```

- **Description**
  
  This function 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>Timeout</td>
</tr>
<tr>
<td>E_NONEXIST</td>
<td>The semaphore was deleted</td>
</tr>
<tr>
<td>E_OBJECTS_ID</td>
<td>Object ID error</td>
</tr>
<tr>
<td>E_OBJ_NO_METHOD</td>
<td>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;
      }
  }
  ```
default://case E_OBJ_NO_METHOD:
    // Some error processing is done.
    break;
    return (NG);
}
mpTaskDelay

Delays a task.

- **Syntax**

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

- **Description**

  This function 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 function:

  float mpGetRtc(void)

  Return value: 0.001 or more
  (unit: msec)
mpGetRtc

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

- **Syntax**
  
  float mpGetRtc(void)

- **Parameter**
  
  None

- **Return Value**

  The OS clock cycle (unit: msec)

  The return value for the mpGetRtc() is int type for the DX200 and FS100, however, for the YRC1000micro, the value is the float type. In the YRC1000micro, mpGetRtc() returns the value less than 1.
mpGetSysClkRate

Retrieves the OS clock frequency.

- **Syntax**
  ```c
  int mpGetSysClkRate(void)
  ```

- **Parameter**
  None

- **Return Value**
  The OS clock frequency [Hz]
mpClkAnnounce

Notifies the I/O or interpolation cycle event.

- **Syntax**

  ```c
  STATUS mpClkAnnounce
  (  
    int clk_id  
  )
  ```

- **Parameter**

  `[clk_id]`
  Sends one of the following values to `clk_id`
  - **MP_IO_CLK**
    The I/O control 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]
  ```c
  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);
  }

  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 APIs which can be used in the task that performs fixed-cycle processing by using mpClkAnnounce are restricted. For details, refer to chapter 15.5.2 "Task Priority and Usable API" in "YRC1000micro OPTIONS INSTRUCTIONS Programmer's Manual For New Language Environment MotoPlus (HW1484526)".
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 “Chap. 2.6 Processing Time Measurement” of “Chap. 2 Task” in “YRC1000micro OPTIONS INSTRUCTIONS Programmer’s Manual for New Language Environment MotoPlus (HW1484526)”. If the compatibility specification (S2C1116) in the processing time measurement is set, the operation in the processing time measurement is changed. For the specification, use the bit specification.

- **Parameter**

```plaintext
[maxLapTimeNum] Number of created lap time counter
```

Creates the specified number of the stopwatches to store the lap time.

- **Return Value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Error: The stopwatch cannot be created. (insufficient memory)</td>
</tr>
</tbody>
</table>

- **Note**

- **Parameter**

```plaintext
d0 bit: Acquisition time of mpStopWatchGetTime()
0: equivalent to 2 byte data, 1: equivalent to 4 byte data (default)
d1 bit: Measurement of the lap time in mpStopWatchStop(), Execute/Not execute
0: Execute the lap time measurement, 1: Not execute the lap time measurement (default)
```
2 Task Control API
mpStopWatchCreate

Make sure to call this function before using the stopwatch. After that, all of the stopwatch routines use the stopwatch object ID created in this function 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 function releases the memory area allocated for the stopwatch.
mpStopWatchStart

Starts the measurement with the stopwatch.

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

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

- **Return Value**
  ```
  None
  ```
mpStopWatchStop

Stops the measurement with 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 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 function clears the lap time counter of the stopwatch. If the stopwatch does not have any lap time counter, this function does not run.
mpStopWatchGetTime

Retrieves 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

Retrieves the number of the lap time counter.

- **Syntax**

  ```c
  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

Retrieves 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 retrieve the latest lap time and the oldest lap time, the following values are provided.

  - **MP_STOP_WATCH_GET_LAP_NEWEST**
    The value to retrieve the latest lap time
  
  - **MP_STOP_WATCH_GET_LAP_OLDEST**
    The value to retrieve 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

Retrieves the range of the current 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
mpUsrWdogCreate

Creates a user watchdog timer.

- **Syntax**
  
  ```
  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 or more 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:
      
      ```
      typedef void (*MP_USR_WDG_ROUTINE) (MP_WDG_HANDLE);
      ```

      Example of time-out routine:
      
      ```
      void fncwdRoutine0( MP_WDG_HANDLE handle )
      {
      ...
      }
      ```
3 User Watchdog API
mpUsrWdogCreate

Return Value

<table>
<thead>
<tr>
<th>Type</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><strong>ERROR</strong></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**

[handle]

Specify the handle value of the user watchdog timer to be deleted.

**Return Value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td>Normal end</td>
</tr>
</tbody>
</table>
| 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**
  
<table>
<thead>
<tr>
<th>Value</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 an invalid handle value is specified, and the like.</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td>Normal end</td>
</tr>
<tr>
<td>ERROR</td>
<td>Error</td>
</tr>
</tbody>
</table>

  In the case when an invalid handle value is specified, and the like.
mpSocket

Opens a socket.

- **Syntax**
  
  ```c
  LONG mpSocket(LONG domain, LONG type, LONG 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**
  
<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;=0</td>
<td>Socket descriptor</td>
</tr>
<tr>
<td>-1</td>
<td>Error</td>
</tr>
</tbody>
</table>
mpListen

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

- **Syntax**
  
  ```c
  LONG mpListen(LONG sockfd, LONG 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**
  
  - **[sockfd]**
    The descriptor which refers to the socket
  
  - **[backlog]**
    The number of the I/O data

- **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>
mpAccept

Accepts a connection from the client to the socket.

- **Syntax**

  ```c
  LONG mpAccept(LONG sockfd, struct sockaddr *addr, LONG *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 `sockfd`. 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**

  - **sockfd**
    
      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

LONG mpBind(LONG sockfd, const struct sockaddr *addr, LONG addrlen);

- 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

[sockfd]
The socket descriptor created by mpSocket

[addr]
The pointer of the sockaddr structure kept 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 the addr (unit: byte)

- Return Value

0 Normal end
-1 Error
mpConnect

Initiates a connection to the socket.

- **Syntax**

  ```c
  LONG connect(LONG sockfd, const struct sockaddr *addr, LONG addrlen);
  ```

- **Description**

  If `sockfd` is a socket of type SOCK_STREAM, this routine establishes a virtual circuit between `sockfd` and another socket specified by the name. If `sockfd` 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**

  - `[sockfd]`  
    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**

<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>
mpRecv

mpRecvFrom

Receives a message from the socket.

■ Syntax

LONG mpRecv(LONG sockfd, const void *buf, LONG len, LONG flags);
LONG mpRecvFrom(LONG sockfd, const void *buf, LONG len, LONG flags,
                struct sockaddr *src_addr, LONG *addrlen);

■ 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

[sockfd]
Socket descriptor

[buf]
The pointer to the buffer for the received data

[len]
The maximum allowable length of received data to be stored in buf (unit: byte)

[flags]
Set "0".

[src_addr]
in] The pointer to the sockaddr structure which stores the socket address

[addrlen]
[in] [out] At calling, this must be initialized with the structure size (in bytes)
specified by src_addr. 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**
  
  LONG mpSend(LONG sockfd, const void *buf, LONG len, LONG flags);
  
  LONG mpSendTo(LONG sockfd, const void *buf, size_t len, LONG flags, 
  const struct sockaddr *dest_addr, LONG addrlen);

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

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

- **Return Value**
  
  Value >= 0 The number of bytes sent (normal end)
  
  -1 Error
mpClose

Closes the socket descriptor.

■ Syntax

LONG mpClose(LONG fd);

■ Description

Closes the socket descriptor.

■ Parameter

[fd]
Socket 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</td>
</tr>
</tbody>
</table>

NOTE

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 YRC1000micro 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**

  ```
  ULONG     mpInetAddr(const CHAR *cp);
  CHAR *    mpInetNtoa(struct in_addr in);
  void      mpInetNtoaB(struct in_addr inetAddress, char *pString)
  ```

- **Description**

  mpInetAddr() converts the internet host address `cp` 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
  LONG mpGetsockname(LONG sockfd, struct sockaddr *addr, LONG *addrlen);
  ```

- **Description**
  
  The current socket address specified by `sockfd` is returned by storing in the buffer specified by `addr`. The `addrlen` parameter must be initialized to the data size (in bytes) specified by `addr`. When the parameter returns, the actual size (in bytes) of the socket address is stored in `addrlen`.

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

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

<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>
mpSetsockopt

Sets socket options. (Sets options for disconnection.)

- **Syntax**

  ```c
  STATUS mpSetsockopt
  (  
      int sockfd, /* target socket */
      int level, /* protocol level of option */
      int optname, /* option name */
      char * optval, /* pointer to option value */
      int optlen /* option length */
  )
  ```

- **Parameter**

  - `[sockfd]`
    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 `sockfd` 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`

- **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**

  0 Normal end

  -1 Error (Incorrect file descriptor, incorrect parameter specification)
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**

  - **[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.

  - **[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)];
  };
  ```

  ```
  #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)
  ```
4  Network API
mpSelect

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.

[pExceptFds]
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 &gt;= 0</td>
<td>The number of available sockets</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**
  
  void free(void *ptr);

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

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

- **Return Value**
  
  None
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 full pathname (drive name + file name) of the file to be created
    
    `<Ex.>` MPRAM1:0/TestFile.dat
    
    Usable drive names are as follows.
    
    - MP_SRAM_DEV_DOS (“MPSRAM1:0”)
    - MP_DRAM_DEV_DOS (“MPRAM1:0”)
    - MP_USB0_DEV_DOS (“MPUSB0”)

    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)
  
  - `-3` Specified drive name is invalid.
6 File Control API
mpCreate

### 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

  Usable drive names are as follows.
  - MP_SRAM_DEV_DOS ("MPSRAM1:0")
  - MP_DRAM_DEV_DOS ("MPRAM1:0")
  - MP_USB0_DEV_DOS ("MPUSB0")

- **[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)
- `-3` Specified drive name is invalid.
Note

The maximum number of file descriptors that can be used simultaneously is MP_FILES (16).
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

Usable drive names are as follows.

- MP_SRAM_DEV_DOS ("MPSRAM1:0")
- MP_DRAM_DEV_DOS ("MPRAM1:0")
- MP_USB0_DEV_DOS ("MPUSB0")

■ 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 (No specified file or descriptor)</td>
</tr>
<tr>
<td>-3</td>
<td>Specified drive name is invalid.</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.

Please check a drive name and a file name enough not to delete except the target file.
mpClose

Closes files.

**Syntax**

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

**Parameter**

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

**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.
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**

<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 (File is opened, no specified file)</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.
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 (Specified file descriptor is incorrect or file for reading is not opened.)

- **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`.

---

**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.
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 */
  int fd, /* file descriptor on which to write */
  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 (Specified file descriptor is incorrect, file for writing is not opened, or disk capacity is insufficient.)

- **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
- **[function]**
The request code depending on the device
- **[arg]**
The parameter corresponding to the request code

### Function

<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>FIONMOVE</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>FIWHERE</td>
<td>Returns the current byte position of the file.</td>
<td>position = mpIoctl (fd, FIWHERE, 0);</td>
</tr>
</tbody>
</table>

**[arg]**
The parameter corresponding to the request code
6 File Control API
mpioctl

Return Value

>=0 Normal end (The return value depending on the device)
-1 Error (Specified file descriptor is incorrect.)

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 (Specified file descriptor is incorrect.)

---

**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
{
    unsigned long st_dev; /* device ID number */
    unsigned long st_ino; /* file serial number */
    int st_mode; /* file mode (see below) */
    unsigned long st_nlink; /* number of links to file */
    unsigned short st_uid; /* user ID of file's owner */
    unsigned short st_gid; /* group ID of file's group */
    unsigned long st_rdev; /* device ID, only if special file */
    long long 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;
    unsigned char st_attrib; /* file attribute byte (dosFs only) */
    int reserved1; /* reserved for future use */
    int reserved2; /* reserved for future use */
};
```
6  File Control API

mpStat

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

0   Normal end

-1 Error (Incorrect specified file descriptor, I/F memory extension option is not used.)

-3 Invalid drive name is specified.

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.

NOTE
mpOpendir

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

■ Syntax

```c
DIR * mpOpendir /* return : A pointer to a directory descriptor, */
    (                      /*         or NULL if there is an error by opendir() */
    char * dirName   /* name of directory to open */
    );
```

```c
#define NAME_MAX 255 /* 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

[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, */
 struct dirent * d = NULL; /* or NULL if there is an end-of-directory */
( /* marker or error */
    DIR * pDir /* pointer to directory descriptor */
);
```

```c
#define NAME_MAX 255 /* 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

- **Return Value**

- `>0` Normal end (The address of the read directory entry)
- `0` Error (Specified file descriptor is incorrect.)

---

**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 255 /* 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 */) ;
```

```c
#define NAME_MAX 255 /* 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**

  `[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</td>
</tr>
</tbody>
</table>

  (Specified directory descriptor is incorrect.)

  - 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.
mpCreatedir

Creates a directory in the USB drive.

- **Syntax**

```c
int mpCreatedir
(const char *dirName/* name of directory to create */
);
```

- **Parameter**

**[dirName]**

A name of the directory to be created

Input the full pathname (up to 255 bytes) including the drive name ("MPUSB0").

<Ex.>

MPUSB0/directory → "directory" is created in the root of the USB drive 0.

MPUSB0/directory/sample → "sample" is created in "directory" of the USB drive 0.

- **Return Value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end (Directory creation completed)</td>
</tr>
<tr>
<td>-1</td>
<td>Error (The USB drive is not specified, The specified directory name has an error, or the specified directory already exists.)</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.

Any directory cannot be created in the SRAM drive or the DRAM drive. The directory which has multiple hierarchies cannot be created simultaneously.
mpRemovedir

Deletes the directory in the USB drive.

- **Syntax**

  ```c
  int mpRemovedir
  (const char *dirName/* name of directory to remove */
  );
  ```

- **Parameter**

  **[dirName]**
  A name of the directory to be deleted
  
  Input the full pathname (up to 255 bytes) including the drive name ("MPUSB0" or "MPUSB1").
  
  **<Ex.>**
  
  MPUSB0/directory/sample
  → "sample" in the "directory" of the USB drive 0 is deleted.
  
  MPUSB0/directory
  → "directory" of the USB drive 0 is deleted.

- **Return Value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end (Directory deletion completed)</td>
</tr>
<tr>
<td>-1</td>
<td>Error (The USB drive is not specified, The specified directory name has an error, the specified directory does not exist, or any file and/or any directory exist(s) in the specified directory.)</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.

The SRAM drive and the DRAM drive cannot be specified. Only when the targeted directory is empty, it can be deleted.
mpRenamedir

Renames the directory in the USB drive.

**Syntax**

```c
int mpRenamedir
(const char *srcDirName/* name of directory to rename (source) */
const char *destDirName/* name of directory to rename (destination) */
);
```

**Parameter**

- **[srcDirName]**
  A name of the directory (Change source)
- **[destDirName]**
  A name of the directory (Change destination)

Input the full pathname (up to 255 bytes) including the drive name ("MPUSB0").

<Ex.>

- **[srcDirName]** MPUSB0/directory
- **[destDirName]** MPUSB0/sample

→ "directory" of the USB drive 0 is changed to "sample".

**Return Value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end (Directory renaming completed)</td>
</tr>
<tr>
<td>-1</td>
<td>Error (The USB drive is not specified, The specified directory name has an error, the paths of the change source and the change destination are different, the directory of the change source does not exist, or the directory of the change destination already exists.)</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.

The SRAM drive and the DRAM drive cannot be specified.
7 Existing File Access API

mpLoadFile

Wants file data in the specified drive to the controller.

**Syntax**

```
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 drive that reads the written file data

  ```
  #define MP_DRV_ID_DRAM 1
  #define MP_DRV_ID_SRAM 2
  #define MP_DRV_ID_USB0 3
  #define MP_DRV_ID_USB1 4
  ```

- **[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.
- 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 specified drive.

Syntax

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 drive that stores the read file data

#define MP_DRV_ID_DRAM 1
#define MP_DRV_ID_SRAM 2
#define MP_DRV_ID_USB0 3
#define MP_DRV_ID_USB1 4

[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 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>
</table>
| 0x1086 or less      | 4230 or less   | Error related to the file For details, refer to the error numbers (decimal) in the “Chap. 9.1 Error Message” of “Chap. 9 Error” in “YRC1000micro MAINTENANCE MANUAL (RE-CHO-A115)”.
| 0x2060              | 8288           | Alarm or error status |
| 0x2090              | 8336           | Data being accessed by another function Retry later. |
| 0xffffffff         | -3             | Invalid drive name |
| 0xffffff             | 65535         | Other than the above |
| 0xffffffff         | -1             | Other than the above |
• FileControl API is executed only with the priority MP_PRIORITY_NORMAL. When a task with a higher priority than MP_PRIORITY_NORMAL calls FileControl API, and the task priority limitation is set to "Automatic Adjustment" (S2C1101 = 1), the system temporarily lowers the priority to MP_PRIORITY_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.
mpRefreshFileList

Updates a file list.

- **Syntax**

  ```c
  LONG mpRefreshFileList /* return : status */
  (   
      SHORT extension /* 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>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>

**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

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>

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.

NOTE
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**

- 0 Normal end
- -1 Error

**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.
mpFdWriteFile

Writes file data in the specified fd to the controller.

- **Syntax**
  
  ```c
  int mpFdWriteFile /* return : status */
  (  
    int fd, /* file descriptor to read from */
    MP_FILE_NAME_SEND_DATA *sData /* 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.
  
  - **[sData]**
    
    The pointer to the structure that stores the written file information
    (The following members are provided.)
    
    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.)
    
    #define TRANS_FILE_LEN (32 + 1 + 3)

- **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 “Chap. 9.1 Error Message” of “Chap. 9 Error” in “YRC1000micro MAINTENANCE MANUAL (RE-CHO-A115)”.</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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Retry later.</td>
</tr>
<tr>
<td>0xffffffff</td>
<td>65535</td>
<td>Other than the above</td>
</tr>
<tr>
<td>0xffffffff</td>
<td>-1</td>
<td>Other than the above</td>
</tr>
</tbody>
</table>
7 Existing File Access API

mpFdWriteFile

• 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 mpFdReadFile /* return : status */
(    int fd, /* file descriptor to read from */
    MP_FILE_NAME_SEND_DATA *sData /* 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.

- **[sData]**
  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.)

  ```
  #define TRANS_FILE_LEN    (32 + 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&lt;br&gt;For details, refer to the error numbers (decimal) in the “Chap. 9.1 Error Message” of “Chap. 9 Error” in “YRC1000micro MAINTENANCE MANUAL (RE-CHO-A115)”.</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&lt;br&gt;Retry later.</td>
</tr>
<tr>
<td>0xffff</td>
<td>65535</td>
<td>Other than the above</td>
</tr>
<tr>
<td>0xffffffff</td>
<td>-1</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.
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   *rData /* 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        uIsEndFlag;    /* Always 1 with the normal end */
USHORT        uListDataNum;  /* The number of read job names */
UCHAR         cListData[MP_LIST_DATA_SIZE]; // Work area used for reading */
CHAR               reserved[2];
} MP_GET_JOBLIST_RSP_DATA;
```

#define MP_LIST_DATA_SIZE    1000   /*Size of work area used for reading */

■ 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>

■ Note

- Job list format
7 Existing File Access API
mpFdGetJobList

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

**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.
8 System Monitor API

mpGetVarData

Retrieves variables B, I, D, R (byte, integer, double precision, real).
Specifies multiple variables.
If linear motion axes such as a servo truck are included, -1(error) is returned.

- Syntax

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

- Parameter

[sData]
The 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
    <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.

<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>

[NOTE]
If multiple variable data are retrieved by this function, 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)
8 System Monitor API
mpGetVarData

- 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**
This function takes a few msec to process, so be sure not to use it in the I/O control cycle task or in the interpolation cycle task. Is it is used, the process cannot be completed within the control cycle.
mpGetSVarInfo

Gets the value of the variable (S).

- Syntax

```c
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[MAX_SVAR_SIZE+1]; /*MAX_SVAR_SIZE (32
characters)+null character\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

<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>
• Since this function’s processing time is a few milliseconds, do not use this function in a task with the I/O control cycle or interpolation 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 function, 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 function cannot be executed because the system is busy. In this case, the error (-1) is returned. Retry this function 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]`
    Pointer to the 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 - 5127 | Universal input #00010 - #05127(4096)                   |
      | 10010 - 15127 | Universal output #10010 - #15127(4096)          |
      | 20010 - 21287 | External input #20010 - #21287(1024)                   |
      | 3030010 - 31287 | External output #30010 - #31287(1024)               |
      | 40010 - 42567 | Specific input #40010 - #42567(2048)                  |
      | 50010 - 55127 | Specific output #50010 - #55127(4096)                 |
      | 60010 - 60647 | Interface panel #60010 - #60647(512)                  |
      | 70010 - 79997 | Auxiliary relay #70010 - #79997(7992)                 |
      | 80010 - 85127 | Control state #80010 - #85127(4096)                   |
      | 87010 - 87207 | Pseudo input #87010 - #87207(160)                     |
      | 27010 - 29567 | Network input #27010 - #29567(2048)                   |
      | 37010 - 39567 | Network output #37010 - #39567(2048)                  |
      | 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`.

For the YRC1000micro, since the scanning time of the concurrent I/O is shorter than the I/O control cycle, the status changing of the signal may not be detected even if the priority is set to the notification task for the I/O control cycle. For example, in the case of monitoring the pulse signal which is changing instantly OFF-ON-OFF, ON may not be acquired depending on the executing timing of the application. Monitor the signal by adjusting the pulse range or by preparing another signal to latch the pulse signal.

**[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
      
      | Value                  | Description          |
      |------------------------|----------------------|
      | MP_RESTYPE_VAR_B       | Byte type            |
      | MP_RESTYPE_VAR_I       | Integer type         |
      | MP_RESTYPE_VAR_D       | Double-precision type|
      | MP_RESTYPE_VAR_R       | Real type            |
      | MP_RESTYPE_CIO         | I/O                  |

    - **<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`.

- **NOTE**

  For the YRC1000micro, since the scanning time of the concurrent I/O is shorter than the I/O control cycle, the status changing of the signal may not be detected even if the priority is set to the notification task for the I/O control cycle. For example, in the case of monitoring the pulse signal which is changing instantly OFF-ON-OFF, ON may not be acquired depending on the executing timing of the application. Monitor the signal by adjusting the pulse range or by preparing another signal to latch the pulse signal.
8 System Monitor API

**mpMonitor**

- **[rData]**
  Monitor information value

- **[infonum]**
  Number of monitor (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>
mpGetPosVarData

Retrieves the position-type variable.

**Syntax**

```
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: 
  ```
  typedef struct 
  { 
    USHORT   usType; 
    USHORT   usIndex; 
  } MP_VAR_INFO;
  ```

  **Member:** 
  ```
  <usType>
  Variable type
  ```
  ```
  Value | Description
  MP_RESTYPE_VAR_ROBOT | Robot
  MP_RESTYPE_VAR_BASE | Base
  MP_RESTYPE_VAR_STATION | Station
  ```

  **<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`.

  - If multiple position-type variables are retrieved by this function, these data are not synchronized. Thus, while retrieving multiple values, these may be changed by another application task or system task.
  - Since this function's processing time is a few milliseconds, do not use this API in a task with the I/O control cycle or interpolation cycle. If it is used in such a task, the process cannot be completed within the control cycle.
8 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></td>
<td>D07 - D06</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>Reserved by manufacturer</td>
<td></td>
</tr>
<tr>
<td>D16 - D21</td>
<td>Tool number (0 - 63)</td>
<td></td>
</tr>
<tr>
<td>D22 - D27</td>
<td>User coordinate number</td>
<td></td>
</tr>
<tr>
<td>D28 - D31</td>
<td>Reserved by manufacturer</td>
<td></td>
</tr>
<tr>
<td>rData[1]</td>
<td>(Extended attribute)</td>
<td></td>
</tr>
</tbody>
</table>

[n]m

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**

```
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:

```
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:

```c
#define STR_VAR_SIZE 32
#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 {
    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 63)

- `<uf_no>`
  User coordinate system file No. (0 to 62)

- `<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_ELBO</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

0 : Normal end  
Any of following negative values: : Abnormal end

<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>•</td>
</tr>
<tr>
<td></td>
<td>MP_P_VAR_BUFF bp;</td>
</tr>
<tr>
<td></td>
<td>•</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>Bit No.</td>
<td>d0 to d5</td>
<td>.dtype</td>
</tr>
<tr>
<td>Type</td>
<td>d08</td>
<td>d0</td>
</tr>
<tr>
<td></td>
<td>d09</td>
<td>.fig_ctrl</td>
</tr>
<tr>
<td></td>
<td>d10</td>
<td>d1</td>
</tr>
<tr>
<td></td>
<td>d11</td>
<td>d2</td>
</tr>
<tr>
<td>Upper elbow/Lower elbow</td>
<td>d12</td>
<td>d3</td>
</tr>
<tr>
<td>Flip/No flip</td>
<td></td>
<td>d4</td>
</tr>
<tr>
<td>R&lt;180/R&gt;=180</td>
<td></td>
<td>d5</td>
</tr>
<tr>
<td>T&lt;180/T&gt;=180</td>
<td></td>
<td>.tool_no</td>
</tr>
<tr>
<td>S&lt;180/S&gt;=180</td>
<td></td>
<td>.uf_no</td>
</tr>
<tr>
<td>Tool No.</td>
<td>d16 to d21</td>
<td></td>
</tr>
<tr>
<td>User coordinate No.</td>
<td>d22 to d27</td>
<td></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
          ```c
          typedef struct
          {
            SHORT     sIsAlarm;
            CHAR       reserved[2];
          } MP_ALARM_STATUS_RSP_DATA;
          ```
          - **Member:** `<sIsAlarm>`
            - Alarm status

<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>

- **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>
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;
    CHAR     reserved[2];
    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:  
  ```c
  #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

```c
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:

```c
typedef struct
{
    SHORT  sMode;
    SHORT  sRemote;
} 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>

<sRemote>

Remote mode

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Remote OFF</td>
</tr>
<tr>
<td>1</td>
<td>Remote ON</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:
            ```c
            typedef struct
            {
                SHORT  sCycle;
                CHAR   reserved[2];
            } MP_CYCLE_RSP_DATA;
            ```
          - Member: `<sCycle>` Cycle

- **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>
mpGetServoPower

Retrieves the ON/OFF status of the servo power.

- **Syntax**

```c
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

- MP_SERVO_POWER_RSP_DATA

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

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

  Member:  <sServoPower>
      Servo power

<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>

- **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>
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
[in] The data structure which receives the play status

Syntax: typedef struct
{  
SHORT sStart;
SHORT sHold;
} MP_PLAY_STATUS_RSP_DATA

Member: <sStart>
Operation status

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

<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>

■ 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

<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>
<tr>
<td>6</td>
<td>Subtask 6</td>
</tr>
<tr>
<td>7</td>
<td>Subtask 7</td>
</tr>
<tr>
<td>8</td>
<td>Subtask 8</td>
</tr>
<tr>
<td>9</td>
<td>Subtask 9</td>
</tr>
<tr>
<td>10</td>
<td>Subtask 10</td>
</tr>
<tr>
<td>11</td>
<td>Subtask 11</td>
</tr>
<tr>
<td>12</td>
<td>Subtask 12</td>
</tr>
<tr>
<td>13</td>
<td>Subtask 13</td>
</tr>
<tr>
<td>14</td>
<td>Subtask 14</td>
</tr>
<tr>
<td>15</td>
<td>Subtask 15</td>
</tr>
</tbody>
</table>
8  System Monitor API
mpGetMasterJob

[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:  #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)

■ Return Value

  0      Normal end
  -1     Error
8 System Monitor API

mpGetCurJob

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;
    ```

    Membership: `<sTaskNo>`

    Task number

    | Value | Description          |
    |-------|----------------------|
    | 0     | Master task          |
    | 1     | Subtask 1            |
    | 2     | Subtask 2            |
    | 3     | Subtask 3            |
    | 4     | Subtask 4            |
    | 5     | Subtask 5            |
    | 6     | Subtask 6            |
    | 7     | Subtask 7            |
    | 8     | Subtask 8            |
    | 9     | Subtask 9            |
    | 10    | Subtask 10           |
    | 11    | Subtask 11           |
    | 12    | Subtask 12           |
    | 13    | Subtask 13           |
    | 14    | Subtask 14           |
    | 15    | Subtask 15           |
mpGetCurJob

**[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;
```

Member:  
- `<usJobLine>`  
  Job line  
- `<usStep>`  
  Step  
- `<cJobName[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 “Chap. 4.2.2 Special Playback Operations” in “YRC1000micro OPERATOR’S MANUAL (RE-CSO-A058)”.

**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

  ```c
  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>D10</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: `#define MAX_JOB_NAME_LEN (33)`
  ```c
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
8  System Monitor API
mpGetJobDate

\<sStartMin>\>
  Minute
\<sStartSec>\>
  Second
\<lElapsedTime>\>
  Not used

- **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>
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>`
  Controls 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>R3 (Robot 3)</td>
</tr>
<tr>
<td>3</td>
<td>R4 (Robot 4)</td>
</tr>
<tr>
<td>4</td>
<td>R5 (Robot 5)</td>
</tr>
<tr>
<td>5</td>
<td>R6 (Robot 6)</td>
</tr>
<tr>
<td>6</td>
<td>R7 (Robot 7)</td>
</tr>
<tr>
<td>7</td>
<td>R8 (Robot 8)</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>B3 (Base 3)</td>
</tr>
<tr>
<td>11</td>
<td>B4 (Base 4)</td>
</tr>
<tr>
<td>12</td>
<td>B5 (Base 5)</td>
</tr>
<tr>
<td>13</td>
<td>B6 (Base 6)</td>
</tr>
<tr>
<td>14</td>
<td>B7 (Base 7)</td>
</tr>
<tr>
<td>15</td>
<td>B8 (Base 8)</td>
</tr>
</tbody>
</table>

**[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
System Monitor API

mpGetCartPos

Syntax:  #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 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>

For more information on the position data, refer to “Chap. 2 Manipulator Coordinate Systems and Operations” in “YRC1000micro OPERATOR’S MANUAL (RE-CSO-A058)“.

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:

```c
typedef struct
{
    SHORT  sRobotNo;
    SHORT  sFrame;
    SHORT  sToolNo;
    CHAR    reserved[2];
} MP_CARTPOS_EX_SEND_DATA;
```

Member: **<sRobotNo>**  
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>R3 (Robot 3)</td>
</tr>
<tr>
<td>3</td>
<td>R4 (Robot 4)</td>
</tr>
<tr>
<td>4</td>
<td>R5 (Robot 5)</td>
</tr>
<tr>
<td>5</td>
<td>R6 (Robot 6)</td>
</tr>
<tr>
<td>6</td>
<td>R7 (Robot 7)</td>
</tr>
<tr>
<td>7</td>
<td>R8 (Robot 8)</td>
</tr>
</tbody>
</table>

Member: **<sFrameNo>**  
Specified coordinate frame

<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, ..., 64</td>
<td>User coordinate number 1, 2,..., 63</td>
</tr>
</tbody>
</table>

Member: **<sToolNo>**  
Tool number(0 - 63)
[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</td>
</tr>
<tr>
<td></td>
<td>(micron in the case of traveling axis)</td>
</tr>
<tr>
<td>lPos[9]</td>
<td>2nd external axis pulse value</td>
</tr>
<tr>
<td></td>
<td>(micron in the case of traveling axis)</td>
</tr>
<tr>
<td>lPos[10]</td>
<td>3rd external axis pulse value</td>
</tr>
<tr>
<td></td>
<td>(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 “Chap. 2 Manipulator Coordinate Systems and Operations” in “YRC1000micro OPERATOR’S MANUAL (RE-CSO-A058)*.

■ 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>
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: typedef struct
        ```c
        {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>R3 (Robot 3)</td>
</tr>
<tr>
<td>3</td>
<td>R4 (Robot 4)</td>
</tr>
<tr>
<td>4</td>
<td>R5 (Robot 5)</td>
</tr>
<tr>
<td>5</td>
<td>R6 (Robot 6)</td>
</tr>
<tr>
<td>6</td>
<td>R7 (Robot 7)</td>
</tr>
<tr>
<td>7</td>
<td>R8 (Robot 8)</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>B3 (Base 3)</td>
</tr>
<tr>
<td>11</td>
<td>B4 (Base 4)</td>
</tr>
<tr>
<td>12</td>
<td>B5 (Base 5)</td>
</tr>
<tr>
<td>13</td>
<td>B6 (Base 6)</td>
</tr>
<tr>
<td>14</td>
<td>B7 (Base 7)</td>
</tr>
<tr>
<td>15</td>
<td>B8 (Base 8)</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>
<tr>
<td>19</td>
<td>S4 (Station 4)</td>
</tr>
<tr>
<td>20</td>
<td>S5 (Station 5)</td>
</tr>
<tr>
<td>21</td>
<td>S6 (Station 6)</td>
</tr>
</tbody>
</table>
### mpGetPulsePos

**Description**

The pointer to the data structure which receives the position coordinates in pulse

### Syntax

```c
#define MAX_PULSE_AXES (8)

typedef struct {
    LONG IPos[MAX_PULSE_AXES];
} MP_PULSE_POS_RSP_DATA;
```

### Member

- **<IPos[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>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>
</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>

### Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>S7 (Station 7)</td>
</tr>
<tr>
<td>23</td>
<td>S8 (Station 8)</td>
</tr>
<tr>
<td>24</td>
<td>S9 (Station 9)</td>
</tr>
<tr>
<td>25</td>
<td>S10 (Station 10)</td>
</tr>
<tr>
<td>26</td>
<td>S11 (Station 11)</td>
</tr>
<tr>
<td>27</td>
<td>S12 (Station 12)</td>
</tr>
<tr>
<td>28</td>
<td>S13 (Station 13)</td>
</tr>
<tr>
<td>29</td>
<td>S14 (Station 14)</td>
</tr>
<tr>
<td>30</td>
<td>S15 (Station 15)</td>
</tr>
<tr>
<td>31</td>
<td>S16 (Station 16)</td>
</tr>
<tr>
<td>32</td>
<td>S17 (Station 17)</td>
</tr>
<tr>
<td>33</td>
<td>S18 (Station 18)</td>
</tr>
<tr>
<td>34</td>
<td>S19 (Station 19)</td>
</tr>
<tr>
<td>35</td>
<td>S20 (Station 20)</td>
</tr>
<tr>
<td>36</td>
<td>S21 (Station 21)</td>
</tr>
<tr>
<td>37</td>
<td>S22 (Station 22)</td>
</tr>
<tr>
<td>38</td>
<td>S23 (Station 23)</td>
</tr>
<tr>
<td>39</td>
<td>S24 (Station 24)</td>
</tr>
</tbody>
</table>
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

  ```c
  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>R3 (Robot 3)</td>
</tr>
<tr>
<td>3</td>
<td>R4 (Robot 4)</td>
</tr>
<tr>
<td>4</td>
<td>R5 (Robot 5)</td>
</tr>
<tr>
<td>5</td>
<td>R6 (Robot 6)</td>
</tr>
<tr>
<td>6</td>
<td>R7 (Robot 7)</td>
</tr>
<tr>
<td>7</td>
<td>R8 (Robot 8)</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>B3 (Base 3)</td>
</tr>
<tr>
<td>11</td>
<td>B4 (Base 4)</td>
</tr>
<tr>
<td>12</td>
<td>B5 (Base 5)</td>
</tr>
<tr>
<td>13</td>
<td>B6 (Base 6)</td>
</tr>
<tr>
<td>14</td>
<td>B7 (Base 7)</td>
</tr>
<tr>
<td>15</td>
<td>B8 (Base 8)</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>
<tr>
<td>19</td>
<td>S4 (Station 4)</td>
</tr>
<tr>
<td>20</td>
<td>S5 (Station 5)</td>
</tr>
</tbody>
</table>
### mpGetRadPos

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

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

  _Syntax:_
  ```c
  #define MAX_PULSE_AXES (8)
  typedef struct
  {
    LONG lRadPos[MAX_PULSE_AXES];
  } MP_RAD_POS_RSP_DATA;
  ```

  _Member:_
  ```c
  <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>
8 System Monitor API
mpGetRadPos

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>
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>R3 (Robot 3)</td>
</tr>
<tr>
<td>3</td>
<td>R4 (Robot 4)</td>
</tr>
<tr>
<td>4</td>
<td>R5 (Robot 5)</td>
</tr>
<tr>
<td>5</td>
<td>R6 (Robot 6)</td>
</tr>
<tr>
<td>6</td>
<td>R7 (Robot 7)</td>
</tr>
<tr>
<td>7</td>
<td>R8 (Robot 8)</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>B3 (Base 3)</td>
</tr>
<tr>
<td>11</td>
<td>B4 (Base 4)</td>
</tr>
<tr>
<td>12</td>
<td>B5 (Base 5)</td>
</tr>
<tr>
<td>13</td>
<td>B6 (Base 6)</td>
</tr>
<tr>
<td>14</td>
<td>B7 (Base 7)</td>
</tr>
<tr>
<td>15</td>
<td>B8 (Base 8)</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>
<tr>
<td>19</td>
<td>S4 (Station 4)</td>
</tr>
<tr>
<td>20</td>
<td>S5 (Station 5)</td>
</tr>
</tbody>
</table>
### mpGetRadPosEx

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:**
```c
#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>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>S6 (Station 6)</td>
</tr>
<tr>
<td>22</td>
<td>S7 (Station 7)</td>
</tr>
<tr>
<td>23</td>
<td>S8 (Station 8)</td>
</tr>
<tr>
<td>24</td>
<td>S9 (Station 9)</td>
</tr>
<tr>
<td>25</td>
<td>S10 (Station 10)</td>
</tr>
<tr>
<td>26</td>
<td>S11 (Station 11)</td>
</tr>
<tr>
<td>27</td>
<td>S12 (Station 12)</td>
</tr>
<tr>
<td>28</td>
<td>S13 (Station 13)</td>
</tr>
<tr>
<td>29</td>
<td>S14 (Station 14)</td>
</tr>
<tr>
<td>30</td>
<td>S15 (Station 15)</td>
</tr>
<tr>
<td>31</td>
<td>S16 (Station 16)</td>
</tr>
<tr>
<td>32</td>
<td>S17 (Station 17)</td>
</tr>
<tr>
<td>33</td>
<td>S18 (Station 18)</td>
</tr>
<tr>
<td>34</td>
<td>S19 (Station 19)</td>
</tr>
<tr>
<td>35</td>
<td>S20 (Station 20)</td>
</tr>
<tr>
<td>36</td>
<td>S21 (Station 21)</td>
</tr>
<tr>
<td>37</td>
<td>S22 (Station 22)</td>
</tr>
<tr>
<td>38</td>
<td>S23 (Station 23)</td>
</tr>
<tr>
<td>39</td>
<td>S24 (Station 24)</td>
</tr>
</tbody>
</table>

[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:**
```c
#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)
8  System Monitor API
mpGetRadPosEx

<IRadUnit[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

    | Value | Description       |
    |-------|-------------------|
    | 0     | R1 (Robot 1)      |
    | 1     | R2 (Robot 2)      |
    | 2     | R3 (Robot 3)      |
    | 3     | R4 (Robot 4)      |
    | 4     | R5 (Robot 5)      |
    | 5     | R6 (Robot 6)      |
    | 6     | R7 (Robot 7)      |
    | 7     | R8 (Robot 8)      |
    | 8     | B1 (Base 1)       |
    | 9     | B2 (Base 2)       |
    | 10    | B3 (Base 3)       |
    | 11    | B4 (Base 4)       |
    | 12    | B5 (Base 5)       |
    | 13    | B6 (Base 6)       |
    | 14    | B7 (Base 7)       |
    | 15    | B8 (Base 8)       |
    | 16    | S1 (Station 1)    |
    | 17    | S2 (Station 2)    |
    | 18    | S3 (Station 3)    |
    | 19    | S4 (Station 4)    |
    | 20    | S5 (Station 5)    |
mpGetDegPos

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

### mpGetDegPos

<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>
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

- **MP_CTRL_GRP_SEND_DATA**

  [in] The data structure which specifies the control group

  Syntax: typedef struct

  ```c
  {  
      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>R3 (Robot 3)</td>
</tr>
<tr>
<td>3</td>
<td>R4 (Robot 4)</td>
</tr>
<tr>
<td>4</td>
<td>R5 (Robot 5)</td>
</tr>
<tr>
<td>5</td>
<td>R6 (Robot 6)</td>
</tr>
<tr>
<td>6</td>
<td>R7 (Robot 7)</td>
</tr>
<tr>
<td>7</td>
<td>R8 (Robot 8)</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>B3 (Base 3)</td>
</tr>
<tr>
<td>11</td>
<td>B4 (Base 4)</td>
</tr>
<tr>
<td>12</td>
<td>B5 (Base 5)</td>
</tr>
<tr>
<td>13</td>
<td>B6 (Base 6)</td>
</tr>
<tr>
<td>14</td>
<td>B7 (Base 7)</td>
</tr>
<tr>
<td>15</td>
<td>B8 (Base 8)</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>
<tr>
<td>19</td>
<td>S4 (Station 4)</td>
</tr>
<tr>
<td>20</td>
<td>S5 (Station 5)</td>
</tr>
</tbody>
</table>
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:  

Position coordinates in degrees (up to 8 arrays)

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>S6 (Station 6)</td>
</tr>
<tr>
<td>22</td>
<td>S7 (Station 7)</td>
</tr>
<tr>
<td>23</td>
<td>S8 (Station 8)</td>
</tr>
<tr>
<td>24</td>
<td>S9 (Station 9)</td>
</tr>
<tr>
<td>25</td>
<td>S10 (Station 10)</td>
</tr>
<tr>
<td>26</td>
<td>S11 (Station 11)</td>
</tr>
<tr>
<td>27</td>
<td>S12 (Station 12)</td>
</tr>
<tr>
<td>28</td>
<td>S13 (Station 13)</td>
</tr>
<tr>
<td>29</td>
<td>S14 (Station 14)</td>
</tr>
<tr>
<td>30</td>
<td>S15 (Station 15)</td>
</tr>
<tr>
<td>31</td>
<td>S16 (Station 16)</td>
</tr>
<tr>
<td>32</td>
<td>S17 (Station 17)</td>
</tr>
<tr>
<td>33</td>
<td>S18 (Station 18)</td>
</tr>
<tr>
<td>34</td>
<td>S19 (Station 19)</td>
</tr>
<tr>
<td>35</td>
<td>S20 (Station 20)</td>
</tr>
<tr>
<td>36</td>
<td>S21 (Station 21)</td>
</tr>
<tr>
<td>37</td>
<td>S22 (Station 22)</td>
</tr>
<tr>
<td>38</td>
<td>S23 (Station 23)</td>
</tr>
<tr>
<td>39</td>
<td>S24 (Station 24)</td>
</tr>
</tbody>
</table>

```

**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:  

Position coordinates in degrees (up to 8 arrays)

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

Member: <lDegUnit[MAX_PULSE_AXES]>
  Position coordinates in degrees (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
System Monitor API

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

    | Value | Description   |
    |-------|--------------|
    | 0     | R1 (Robot 1) |
    | 1     | R2 (Robot 2) |
    | 2     | R3 (Robot 3) |
    | 3     | R4 (Robot 4) |
    | 4     | R5 (Robot 5) |
    | 5     | R6 (Robot 6) |
    | 6     | R7 (Robot 7) |
    | 7     | R8 (Robot 8) |
    | 8     | B1 (Base 1)  |
    | 9     | B2 (Base 2)  |
    | 10    | B3 (Base 3)  |
    | 11    | B4 (Base 4)  |
    | 12    | B5 (Base 5)  |
    | 13    | B6 (Base 6)  |
    | 14    | B7 (Base 7)  |
    | 15    | B8 (Base 8)  |
    | 16    | S1 (Station 1)|
    | 17    | S2 (Station 2)|
System Monitor API

mpGetFBPulsePos

[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:  #define MAX_PULSE_AXES (8)

typedef struct
{
  LONG  IPos[MAX_PULSE_AXES];
} MP_FB_PULSE_POS_RSP_DATA;

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>S3 (Station 3)</td>
</tr>
<tr>
<td>19</td>
<td>S4 (Station 4)</td>
</tr>
<tr>
<td>20</td>
<td>S5 (Station 5)</td>
</tr>
<tr>
<td>21</td>
<td>S6 (Station 6)</td>
</tr>
<tr>
<td>22</td>
<td>S7 (Station 7)</td>
</tr>
<tr>
<td>23</td>
<td>S8 (Station 8)</td>
</tr>
<tr>
<td>24</td>
<td>S9 (Station 9)</td>
</tr>
<tr>
<td>25</td>
<td>S10 (Station 10)</td>
</tr>
<tr>
<td>26</td>
<td>S11 (Station 11)</td>
</tr>
<tr>
<td>27</td>
<td>S12 (Station 12)</td>
</tr>
<tr>
<td>28</td>
<td>S13 (Station 13)</td>
</tr>
<tr>
<td>29</td>
<td>S14 (Station 14)</td>
</tr>
<tr>
<td>30</td>
<td>S15 (Station 15)</td>
</tr>
<tr>
<td>31</td>
<td>S16 (Station 16)</td>
</tr>
<tr>
<td>32</td>
<td>S17 (Station 17)</td>
</tr>
<tr>
<td>33</td>
<td>S18 (Station 18)</td>
</tr>
<tr>
<td>34</td>
<td>S19 (Station 19)</td>
</tr>
<tr>
<td>35</td>
<td>S20 (Station 20)</td>
</tr>
<tr>
<td>36</td>
<td>S21 (Station 21)</td>
</tr>
<tr>
<td>37</td>
<td>S22 (Station 22)</td>
</tr>
<tr>
<td>38</td>
<td>S23 (Station 23)</td>
</tr>
<tr>
<td>39</td>
<td>S24 (Station 24)</td>
</tr>
</tbody>
</table>
Return Value

0  Normal end
-1  Error
mpGetFBPulsePosEx

Retrieves the feedback position (the motor-axis, the load-axis) in pulse counts.

- **Syntax**

  ```c
  LONG mpGetFBPulsePosEx(
      MP_FB_PULSE_POS_EX_SEND_DATA* sData,
      /*the data structure which specifies the control group (FB expansion)*/
      MP_FB_PULSE_POS_RSP_DATA* rData
      /* the data structure which receives the feedback position coordinates in pulse */
  );
  ```

- **Description**

  mpGetPulsePos() retrieves the command value. mpGetFBPulsePos() retrieves the feedback pulse value of the motor-axis. When the motor-axis is specified by mpGetFBPulsePosEx(), the same value as that of mpGetFBPulsePos() is retrieved. When the load-axis is specified by mpGetFBPulsePosEx(), the operated pulse value, which the feedback pulse value of the motor-axis is converted to, is retrieved.

- **Parameter**

  **[sData]**

  [in] The pointer to the data structure which specifies the control group (FB expansion)

  - **MP_FB_PULSE_POS_EX_SEND_DATA**

    Syntax:
    ```c
    typedef struct{
        ULONG grp_kind; /* control group type (ROBOT/BASE/STATION) */
        ULONG grp_no;  /* control group number */
        UCHAR type;    /* data type */
        UCHAR reserved[3];
    } MP_FB_PULSE_POS_EX_SEND_DATA;
    ```

    Member:<grp_kind>

    control group type
    0: ROBOT
    1: BASE
    2: STATION

    Member:<grp_no>

    control group number (beginning with 0)
    when the control group type is ROBOT(0): 0 to 7
    when the control group type is BASE(1): 0 to 7
    when the control group type is STATION(2): 0 to 23
8 System Monitor API

mpGetFBPulsePosEx

#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 eight)

<table>
<thead>
<tr>
<th>Value</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>
</tbody>
</table>

■ Return Value

0: Normal end
-1: Error
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_CTRLGRP_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>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>R3 (Robot 3)</td>
</tr>
<tr>
<td>3</td>
<td>R4 (Robot 4)</td>
</tr>
<tr>
<td>4</td>
<td>R5 (Robot 5)</td>
</tr>
<tr>
<td>5</td>
<td>R6 (Robot 6)</td>
</tr>
<tr>
<td>6</td>
<td>R7 (Robot 7)</td>
</tr>
<tr>
<td>7</td>
<td>R8 (Robot 8)</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>B3 (Base 3)</td>
</tr>
<tr>
<td>11</td>
<td>B4 (Base 4)</td>
</tr>
<tr>
<td>12</td>
<td>B5 (Base 5)</td>
</tr>
<tr>
<td>13</td>
<td>B6 (Base 6)</td>
</tr>
<tr>
<td>14</td>
<td>B7 (Base 7)</td>
</tr>
<tr>
<td>15</td>
<td>B8 (Base 8)</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>
<tr>
<td>19</td>
<td>S4 (Station 4)</td>
</tr>
<tr>
<td>20</td>
<td>S5 (Station 5)</td>
</tr>
<tr>
<td>21</td>
<td>S6 (Station 6)</td>
</tr>
<tr>
<td>22</td>
<td>S7 (Station 7)</td>
</tr>
</tbody>
</table>
8 System Monitor API

mpGetServoSpeed

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>S8 (Station 8)</td>
</tr>
<tr>
<td>24</td>
<td>S9 (Station 9)</td>
</tr>
<tr>
<td>25</td>
<td>S10 (Station 10)</td>
</tr>
<tr>
<td>26</td>
<td>S11 (Station 11)</td>
</tr>
<tr>
<td>27</td>
<td>S12 (Station 12)</td>
</tr>
<tr>
<td>28</td>
<td>S13 (Station 13)</td>
</tr>
<tr>
<td>29</td>
<td>S14 (Station 14)</td>
</tr>
<tr>
<td>30</td>
<td>S15 (Station 15)</td>
</tr>
<tr>
<td>31</td>
<td>S16 (Station 16)</td>
</tr>
<tr>
<td>32</td>
<td>S17 (Station 17)</td>
</tr>
<tr>
<td>33</td>
<td>S18 (Station 18)</td>
</tr>
<tr>
<td>34</td>
<td>S19 (Station 19)</td>
</tr>
<tr>
<td>35</td>
<td>S20 (Station 20)</td>
</tr>
<tr>
<td>36</td>
<td>S21 (Station 21)</td>
</tr>
<tr>
<td>37</td>
<td>S22 (Station 22)</td>
</tr>
<tr>
<td>38</td>
<td>S23 (Station 23)</td>
</tr>
<tr>
<td>39</td>
<td>S24 (Station 24)</td>
</tr>
</tbody>
</table>

[rData]
[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 (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

<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>
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

  ```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>R3 (Robot 3)</td>
</tr>
<tr>
<td>3</td>
<td>R4 (Robot 4)</td>
</tr>
<tr>
<td>4</td>
<td>R5 (Robot 5)</td>
</tr>
<tr>
<td>5</td>
<td>R6 (Robot 6)</td>
</tr>
<tr>
<td>6</td>
<td>R7 (Robot 7)</td>
</tr>
<tr>
<td>7</td>
<td>R8 (Robot 8)</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>B3 (Base 3)</td>
</tr>
<tr>
<td>11</td>
<td>B4 (Base 4)</td>
</tr>
<tr>
<td>12</td>
<td>B5 (Base 5)</td>
</tr>
<tr>
<td>13</td>
<td>B6 (Base 6)</td>
</tr>
<tr>
<td>14</td>
<td>B7 (Base 7)</td>
</tr>
<tr>
<td>15</td>
<td>B8 (Base 8)</td>
</tr>
<tr>
<td>16</td>
<td>S1 (Station 1)</td>
</tr>
<tr>
<td>17</td>
<td>S2 (Station 2)</td>
</tr>
</tbody>
</table>
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:
```
#define MAX_PULSE_AXES (8)
```

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

Member: `<ISpeed[MAX_PULSE_AXES]>`

Speed (up to 8 arrays)

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

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
      - 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>R3 (Robot 3)</td>
</tr>
<tr>
<td>3</td>
<td>R4 (Robot 4)</td>
</tr>
<tr>
<td>4</td>
<td>R5 (Robot 5)</td>
</tr>
<tr>
<td>5</td>
<td>R6 (Robot 6)</td>
</tr>
<tr>
<td>6</td>
<td>R7 (Robot 7)</td>
</tr>
<tr>
<td>7</td>
<td>R8 (Robot 8)</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>B3 (Base 3)</td>
</tr>
<tr>
<td>11</td>
<td>B4 (Base 4)</td>
</tr>
<tr>
<td>12</td>
<td>B5 (Base 5)</td>
</tr>
<tr>
<td>13</td>
<td>B6 (Base 6)</td>
</tr>
<tr>
<td>14</td>
<td>B7 (Base 7)</td>
</tr>
<tr>
<td>15</td>
<td>B8 (Base 8)</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>
<tr>
<td>19</td>
<td>S4 (Station 4)</td>
</tr>
<tr>
<td>20</td>
<td>S5 (Station 5)</td>
</tr>
<tr>
<td>21</td>
<td>S6 (Station 6)</td>
</tr>
<tr>
<td>22</td>
<td>S7 (Station 7)</td>
</tr>
<tr>
<td>23</td>
<td>S8 (Station 8)</td>
</tr>
</tbody>
</table>
8 System Monitor API

mpGetTorque

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>S9 (Station 9)</td>
</tr>
<tr>
<td>25</td>
<td>S10 (Station 10)</td>
</tr>
<tr>
<td>26</td>
<td>S11 (Station 11)</td>
</tr>
<tr>
<td>27</td>
<td>S12 (Station 12)</td>
</tr>
<tr>
<td>28</td>
<td>S13 (Station 13)</td>
</tr>
<tr>
<td>29</td>
<td>S14 (Station 14)</td>
</tr>
<tr>
<td>30</td>
<td>S15 (Station 15)</td>
</tr>
<tr>
<td>31</td>
<td>S16 (Station 16)</td>
</tr>
<tr>
<td>32</td>
<td>S17 (Station 17)</td>
</tr>
<tr>
<td>33</td>
<td>S18 (Station 18)</td>
</tr>
<tr>
<td>34</td>
<td>S19 (Station 19)</td>
</tr>
<tr>
<td>35</td>
<td>S20 (Station 20)</td>
</tr>
<tr>
<td>36</td>
<td>S21 (Station 21)</td>
</tr>
<tr>
<td>37</td>
<td>S22 (Station 22)</td>
</tr>
<tr>
<td>38</td>
<td>S23 (Station 23)</td>
</tr>
<tr>
<td>39</td>
<td>S24 (Station 24)</td>
</tr>
</tbody>
</table>

**[rData]**

[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)

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

  Member: `<i>TorquePcnt[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>
8 System Monitor API
mpGetTorque

- Return Value
  0 Normal end
  -1 Error
mpGetTorqueEx

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

**Syntax**

```c
LONG mpGetTorqueEx (MP_CTRL_GRP_SEND_DATA* sData,
                    MP_TORQUE_EX_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

<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>R3 (Robot 3)</td>
</tr>
<tr>
<td>3</td>
<td>R4 (Robot 4)</td>
</tr>
<tr>
<td>4</td>
<td>R5 (Robot 5)</td>
</tr>
<tr>
<td>5</td>
<td>R6 (Robot 6)</td>
</tr>
<tr>
<td>6</td>
<td>R7 (Robot 7)</td>
</tr>
<tr>
<td>7</td>
<td>R8 (Robot 8)</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>B3 (Base 3)</td>
</tr>
<tr>
<td>11</td>
<td>B4 (Base 4)</td>
</tr>
<tr>
<td>12</td>
<td>B5 (Base 5)</td>
</tr>
<tr>
<td>13</td>
<td>B6 (Base 6)</td>
</tr>
<tr>
<td>14</td>
<td>B7 (Base 7)</td>
</tr>
<tr>
<td>15</td>
<td>B8 (Base 8)</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>
<tr>
<td>19</td>
<td>S4 (Station 4)</td>
</tr>
<tr>
<td>20</td>
<td>S5 (Station 5)</td>
</tr>
<tr>
<td>21</td>
<td>S6 (Station 6)</td>
</tr>
<tr>
<td>22</td>
<td>S7 (Station 7)</td>
</tr>
<tr>
<td>23</td>
<td>S8 (Station 8)</td>
</tr>
</tbody>
</table>
## System Monitor API
mpGetTorqueEx

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>S9 (Station 9)</td>
</tr>
<tr>
<td>25</td>
<td>S10 (Station 10)</td>
</tr>
<tr>
<td>26</td>
<td>S11 (Station 11)</td>
</tr>
<tr>
<td>27</td>
<td>S12 (Station 12)</td>
</tr>
<tr>
<td>28</td>
<td>S13 (Station 13)</td>
</tr>
<tr>
<td>29</td>
<td>S14 (Station 14)</td>
</tr>
<tr>
<td>30</td>
<td>S15 (Station 15)</td>
</tr>
<tr>
<td>31</td>
<td>S16 (Station 16)</td>
</tr>
<tr>
<td>32</td>
<td>S17 (Station 17)</td>
</tr>
<tr>
<td>33</td>
<td>S18 (Station 18)</td>
</tr>
<tr>
<td>34</td>
<td>S19 (Station 19)</td>
</tr>
<tr>
<td>35</td>
<td>S20 (Station 20)</td>
</tr>
<tr>
<td>36</td>
<td>S21 (Station 21)</td>
</tr>
<tr>
<td>37</td>
<td>S22 (Station 22)</td>
</tr>
<tr>
<td>38</td>
<td>S23 (Station 23)</td>
</tr>
<tr>
<td>39</td>
<td>S24 (Station 24)</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 iTorqueNm[MAX_PULSE_AXES];
} MP_TORQUE_EX_RSP_DATA;

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

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iTorqueNm[0]</td>
<td>In 1st axis (S) direction</td>
</tr>
<tr>
<td></td>
<td>(unit: Nm)</td>
</tr>
<tr>
<td>iTorqueNm[1]</td>
<td>In 2nd axis (L) direction</td>
</tr>
<tr>
<td></td>
<td>(unit: Nm)</td>
</tr>
<tr>
<td>iTorqueNm[2]</td>
<td>In 3rd axis (U) direction</td>
</tr>
<tr>
<td></td>
<td>(unit: Nm)</td>
</tr>
<tr>
<td>iTorqueNm[3]</td>
<td>In 4th axis (R) direction</td>
</tr>
<tr>
<td></td>
<td>(unit: Nm)</td>
</tr>
<tr>
<td>iTorqueNm[4]</td>
<td>In 5th axis (B) direction</td>
</tr>
<tr>
<td></td>
<td>(unit: Nm)</td>
</tr>
<tr>
<td>iTorqueNm[5]</td>
<td>In 6th axis (T) direction</td>
</tr>
<tr>
<td></td>
<td>(unit: Nm)</td>
</tr>
<tr>
<td>iTorqueNm[6]</td>
<td>In 7th axis (E) direction</td>
</tr>
<tr>
<td></td>
<td>(unit: Nm)</td>
</tr>
<tr>
<td>iTorqueNm[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:
    ```c
<sTimeType>
Type of system time
```

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_POWER_ON_TIME_ID</td>
<td>Control power time</td>
</tr>
<tr>
<td>MP_SERVO_ON_TIME_ID</td>
<td>Servo power time</td>
</tr>
<tr>
<td>MP_PLAYBACK_TIME_ID</td>
<td>Playback time</td>
</tr>
<tr>
<td>MP_MOVING_TIME_ID</td>
<td>Moving time</td>
</tr>
<tr>
<td>MPOPERATION_TIME_ID</td>
<td>Operating time</td>
</tr>
</tbody>
</table>

- **[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;
```
8 System Monitor API
mpGetSysTimes

Member: <sStartYear> Year
<sStartMonth> Month
<sStartDay> Day
<sStartHour> Hour
<sStartMin> Minute
<sStartSec> Second
<lElapsedTime> Elapsed time

Return Value

0  Normal end
-1  Error
mpGetSysVersionNo

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

Syntax

[rData]

LONG mpGetSysVersionNo

( MP_SYS_VERSION_NO_RSP_DATA* rData )

Parameters

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

MP_SYS_VERSION_NO_RSP_DATA

[out] The data structure which receives the system version

Syntax: 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: System version character length 22

Member: <sys_version>

System version

(e.g.) “YBS1.00.00A(JP/EN)-00”

Return Value

0 Normal end

-1 Error (Version number could not be retrieved)
mpGetJogSpeed

Retrieves the current jog speed.

■ Syntax

```c
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 function and the functions listed below. Thus, if this function is executed while one of the following functions is already being executed, the task may be suspended to wait for the function to be completed. (waiting for semaphore acquisition)

Therefore, DO NOT use the following functions 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(), mpManualMOV()
mpGetJogCoord

Retrieves the current operational coordinates.

- **Syntax**
  ```c
  LONG  mpGetJogCoord ( 
    MP_JOGCOORD_RSP_DATA*  rData 
  );
  ```

- **Parameter**
  - **[rData]**
    **[out]** The pointer to the data structure which receives the operational coordinates
    ```c
    • 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
    | Value | Meaning                      |
    |-------|-----------------------------|
    | 0     | Joint coordinates           |
    | 1     | Cartesian coordinates       |
    | 2     | Cylindrical coordinates     |
    | 3     | Tool coordinates            |
    | 4     | User coordinates            |
    | 5     | External reference point    |
    | 6     | Torch coordinates           |

- **Return Value**
  - 0  Normal end
  - -1 Error
Exclusive control by using semaphores is executed between this function and the functions listed below. Thus, if this function is executed while one of the following functions is already being executed, the task may be suspended to wait for the function to be completed. (waiting for semaphore acquisition)

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

- mpCancelError()
- mpResetAlarm()
- mpSetCycle()
- mpSetServoPower()
- mpSetMasterJob()
- mpSetCurJob()
- mpStartJob()
- mpHold()
- mpWaitForJobEnd()
- mpDeleteJob()
- mpGetJogSpeed()
- mpGetJogCoord()
- mplMOV()
- mpMOVJ()
- mpMOVL()
- mpPulseMOVJ()
- mpPulseMOVL()
- mpManualMOV()
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 functions or the servo control functions, the control group number must be specified by the parameter. To use one of those functions, retrieve the appropriate control group number by using this function (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_R5_GID</td>
<td>R5 (Robot 5)</td>
</tr>
<tr>
<td>MP_R6_GID</td>
<td>R6 (Robot 6)</td>
</tr>
<tr>
<td>MP_R7_GID</td>
<td>R7 (Robot 7)</td>
</tr>
<tr>
<td>MP_R8_GID</td>
<td>R8 (Robot 8)</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_B5_GID</td>
<td>B5 (Base 5)</td>
</tr>
<tr>
<td>MP_B6_GID</td>
<td>B6 (Base 6)</td>
</tr>
<tr>
<td>MP_B7_GID</td>
<td>B7 (Base 7)</td>
</tr>
<tr>
<td>MP_B8_GID</td>
<td>B8 (Base 8)</td>
</tr>
</tbody>
</table>
### Group ID Description

<table>
<thead>
<tr>
<th>Group ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>
<tr>
<td>MP_S4_GID</td>
<td>S4 (Station 4)</td>
</tr>
<tr>
<td>MP_S5_GID</td>
<td>S5 (Station 5)</td>
</tr>
<tr>
<td>MP_S6_GID</td>
<td>S6 (Station 6)</td>
</tr>
<tr>
<td>MP_S7_GID</td>
<td>S7 (Station 7)</td>
</tr>
<tr>
<td>MP_S8_GID</td>
<td>S8 (Station 8)</td>
</tr>
<tr>
<td>MP_S9_GID</td>
<td>S9 (Station 9)</td>
</tr>
<tr>
<td>MP_S10_GID</td>
<td>S10 (Station 10)</td>
</tr>
<tr>
<td>MP_S11_GID</td>
<td>S11 (Station 11)</td>
</tr>
<tr>
<td>MP_S12_GID</td>
<td>S12 (Station 12)</td>
</tr>
<tr>
<td>MP_S13_GID</td>
<td>S13 (Station 13)</td>
</tr>
<tr>
<td>MP_S14_GID</td>
<td>S14 (Station 14)</td>
</tr>
<tr>
<td>MP_S15_GID</td>
<td>S15 (Station 15)</td>
</tr>
<tr>
<td>MP_S16_GID</td>
<td>S16 (Station 16)</td>
</tr>
<tr>
<td>MP_S17_GID</td>
<td>S17 (Station 17)</td>
</tr>
<tr>
<td>MP_S18_GID</td>
<td>S18 (Station 18)</td>
</tr>
<tr>
<td>MP_S19_GID</td>
<td>S19 (Station 19)</td>
</tr>
<tr>
<td>MP_S20_GID</td>
<td>S20 (Station 20)</td>
</tr>
<tr>
<td>MP_S21_GID</td>
<td>S21 (Station 21)</td>
</tr>
<tr>
<td>MP_S22_GID</td>
<td>S22 (Station 22)</td>
</tr>
<tr>
<td>MP_S23_GID</td>
<td>S23 (Station 23)</td>
</tr>
<tr>
<td>MP_S24_GID</td>
<td>S24 (Station 24)</td>
</tr>
</tbody>
</table>

#### Return Value

- **Normal end**: Returns the control group number which begins with 0.
- **Error**: Error
mpGetToolData

Retrieves tools.

**Syntax**

```c
LONG mpGetToolData(
    SHORT sTool/No,
    MP_TOOL_RSP_DATA* rData
)
```

**Parameter**

- **[sTool/No]**
  - [in] Tool number
  - Set the tool number of the tool to be retrieved.

- **[rData]**
  - [out] The pointer to the data structure which receives tools.
    - MP_TOOL_RSP_DATA
  - [out] The data structure which receives tools

Syntax:

```c
typedef struct
{
    USHORT err_no; /* error number */
    CHAR reserved0[2];
    UCHAR name[MAX_TOOL_NAME]; /* tool name */
    LONG x; /* position of the tool tip, X-direction (0.001mm) */
    LONG y; /* position of the tool tip, Y-direction (0.001mm) */
    LONG z; /* position of the tool tip, Z-direction (0.001mm) */
    LONG rx; /* posture of the tool, X-direction (0.0001deg.) */
    LONG ry; /* posture of the tool, Y-direction (0.0001deg.) */
    LONG rz; /* posture of the tool, Z-direction (0.0001deg.) */
    LONG xg; /* center of gravity of the tool, X-direction (0.001mm) */
    LONG yg; /* center of gravity of the tool, Y-direction (0.001mm) */
    LONG zg; /* center of gravity of the tool, Z-direction (0.001mm) */
    LONG w; /* tool weight (0.001kg) */
    LONG ix; /* position of center of gravity on the tool, X-direction moment of inertia (0.001mm) */
    LONG iy; /* position of center of gravity on the tool, Y-direction moment of inertia (0.001mm) */
    LONG iz; /* position of center of gravity on the tool, Z-direction moment of inertia (0.001mm) */
    CHAR reserved1[8];
}MP_TOOL_RSP_DATA;
```

MAX_TOOL_NAME: The maximum number of characters of the tool name: 16
8  System Monitor API
mpGetToolData

Member:  <err_no>
        Error number

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>Normal end</td>
</tr>
<tr>
<td>0x3620</td>
<td>Error in the specified tool file number</td>
</tr>
</tbody>
</table>

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

Retrieves the date and time of the controller.

- **Syntax**

  ```c
  LONG mpGetCalendar(
  MP_CALENDAR_RSP_DATA* rData
  );
  ```

- **Parameter**

  - **[rData]**
    - [out] The pointer to the data structure which receives the calendar information

    ```c
    • MP_CALENDAR_RSP_DATA
      Syntax:
    
    typedef struct{
      USHORT usYear; /* Year */
      USHORT usMonth; /* Month */
      USHORT usDay; /* Day */
      USHORT usHour; /* Hour */
      USHORT usMin; /* Minute */
      USHORT usSec; /* Second */
      CHAR reserved[4];
    } MP_CALENDAR_RSP_DATA;
    ```

- **Return Value**

  - 0 Normal end
  - -1 Error
Retrieves the information of the move instruction (step) corresponding to the specified job name and the step number.

**Syntax**

```c
LONG mpGetJobStepData(
    MP_JOB_STEP_NO_SEND_DATA* sData /* the data structure which specifies the job step number */
    MP_JOB_STEP_RSP_DATA* rData /* the data structure which receives the move instruction information */
);
```

**Parameter**

**[sData]**

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

- **MP_JOB_STEP_NO_SEND_DATA**
  
  Syntax:  
  ```
  typedef struct{
    USHORT usStep; /* step number */
    CHAR reserved0[2];
    CHAR cJobName[MAX_JOB_NAME_LEN]; /* job name*/
    CHAR reserved1[3];
  } MP_JOB_STEP_NO_SEND_DATA;
  #define MAX_JOB_NAME_LEN 33 /* up to 32 characters + null character \0 */
  ```

  Member:  
  - `<usStep>` step number (1 to 9998)
  - `<cJobName[MAX_JOB_NAME_LEN]>` job name (up to 32 characters)

**[rData]**

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

- **MP_JOB_STEP_RSP_DATA**
  
  Syntax:  
  ```
  typedef struct{
    USHORT err_no; /* error number */
    CHAR reserved0[2];
    ULONG attr; /* step attribute (future function) */
    UCHAR comment[STEP_COMM_SIZE+1]; /* comment */
    UCHAR reserved1[3];
    ULONG movCtrlDataNum; /* move instruction number in step */
    MP_MOV_CTRL_DATA movCtrlData[MAX_JOB_MOV_CTRL_DATA_NUM]; /* data structure for every move instruction*/
    LONG posLevel; /* setting value of positioning level (PL=) */
    LONG cornerRadius; /* setting value of corner radius (CR=) */
    LONG accValue; /* specified value of acceleration (ACC=) */
    LONG decValue; /* specified value of deceleration (DEC=) */
  } MP_JOB_STEP_RSP_DATA;
  ```

  Member:  
  - `<err_no>` error number
  - `<attr>` step attribute (future function)
  - `<comment[STEP_COMM_SIZE+1]>` comment
  - `<movCtrlDataNum>` move instruction number in step
  - `<movCtrlData[MAX_JOB_MOV_CTRL_DATA_NUM]>` data structure for every move instruction
  - `<posLevel>` setting value of positioning level
  - `<cornerRadius>` setting value of corner radius
  - `<accValue>` specified value of acceleration
  - `<decValue>` specified value of deceleration
8 System Monitor API

mpGetJobStepData

UCHAR reserved2[32];
} MP_JOB_STEP_RSP_DATA;
#define STEP_COMM_SIZE 32  /* up to 32 characters for a comment */
#define MAX_JOB_MOV_CTRL_DATA_NUM 4  /* maximum move instruction data number */

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>0x4040</td>
<td>Specified JOB not found</td>
</tr>
<tr>
<td>0x4130</td>
<td>No position data</td>
</tr>
<tr>
<td>0x5200</td>
<td>Out of data range</td>
</tr>
</tbody>
</table>

<attr>
step attribute

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0001</td>
<td>Edit lock of line</td>
<td>JOB_LINE_ATTR_EDIT_LOCK</td>
</tr>
<tr>
<td>0x0002</td>
<td>Change line to comment</td>
<td>JOB_LINE_ATTR_COMMENT</td>
</tr>
</tbody>
</table>

<comment[STEP_COMM_SIZE+1]>
comment (up to 32 characters)

<movCtrlDataNum>
move instruction number in step
(e.g.) MOVJ: movCtrlDataNum = 1
SMOVL+MOVJ: movCtrlDataNum = 2

<movCtrlData>
data structure for every move instruction

<posLevel>
setting value of positioning level (PL=)

<cornerRaduis>
setting value of conner radius (CR=)

<accValue>
specified value of acceleration (ACC=)

<decimal>
specified value of deceleration (DEC=)
"If the tag “PL=”, “CR=”, “ACC=” or “DEC=” is not in step,”-1” is set to posLevel, cornerRadius, accValue or decValue.
Also, in case a variable is used for the setting such as “PL=B000”, "-1" is set.

• MP_MOV_CTRL_DATA
data structure for every move instruction

Syntax:
typedef struct{
    UCHAR intpType;    /* interpolation type */
    UCHAR intpKind;    /* move instruction type */
    UCHAR speedType;   /* speed type */
    UCHAR reserved0;
    LONG speedValue;   /* speed setting value (VJ=,V=,VR=,VE=,VS=) */
    ULONG posNum;      /* position data number in move instruction */
} MP_MOV_CTRL_DATA;
8 System Monitor API

mpGetJobStepData

MP_JOB_POS_DATA posData[MAX_JOB_MOV_POS_NUM];
/* position data structure */

UCHAR reserved1[32];
}

#define MAX_JOB_MOV_POS_NUM 3 /* maximum position data number */

Member: <intpType>
interpolation type

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Joint interpolation</td>
<td>INTP_TYPE_JOINT</td>
</tr>
<tr>
<td>1</td>
<td>Linear interpolation</td>
<td>INTP_TYPE_LINEAR</td>
</tr>
<tr>
<td>2</td>
<td>Circular interpolation</td>
<td>INTP_TYPE_CIRCLE</td>
</tr>
<tr>
<td>3</td>
<td>Spline interpolation</td>
<td>INTP_TYPE_SPLINE</td>
</tr>
<tr>
<td>4</td>
<td>Reference point</td>
<td>INTP_TYPE_REF</td>
</tr>
</tbody>
</table>

<intpKind>
move instruction type

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal</td>
<td>INTP_KIND_NORMAL</td>
</tr>
<tr>
<td>1</td>
<td>External reference point</td>
<td>INTP_KIND_EI</td>
</tr>
<tr>
<td>2</td>
<td>Coordination</td>
<td>INTP_KIND_SLAVE</td>
</tr>
<tr>
<td>3</td>
<td>Increment value move</td>
<td>INTP_KIND_INC</td>
</tr>
<tr>
<td>4</td>
<td>Increment value move (coordination)</td>
<td>INTP_KIND_SLAVE_INC</td>
</tr>
<tr>
<td>5</td>
<td>Conveyor synchronization</td>
<td>INTP_KIND_SYNCHRO</td>
</tr>
<tr>
<td>6</td>
<td>SVSPOTMOV</td>
<td>INTP_KIND_SVSPOTMOV</td>
</tr>
<tr>
<td>7</td>
<td>SVDRESMOV</td>
<td>INTP_KIND_SVDRESMOV</td>
</tr>
<tr>
<td>255</td>
<td>Others</td>
<td>INTP_KIND_OTHER</td>
</tr>
</tbody>
</table>

Combination of <intpType> and <intpKind> (movCtrlData[0])
8 System Monitor API
mpGetJobStepData

Combination of <intpType> and <intpKind> (movCtrlData[1 to 3])

<table>
<thead>
<tr>
<th>intpType</th>
<th>intpKind</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>MOVJ</td>
</tr>
<tr>
<td>1</td>
<td>MOVL</td>
</tr>
<tr>
<td>2</td>
<td>MOVC</td>
</tr>
<tr>
<td>3</td>
<td>MOVS</td>
</tr>
<tr>
<td>4</td>
<td>REFP</td>
</tr>
</tbody>
</table>

*The interpolation of +SMOVX is the same as that of SMOVx instruction.

<speedType>

speed type
If there is no speed specified tag, speedType=0.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Joint speed (0.01[%])</td>
<td>SPEED_TYPE_VJ</td>
</tr>
<tr>
<td>1</td>
<td>Control point speed (0.1[mm/sec])</td>
<td>SPEED_TYPE_V</td>
</tr>
<tr>
<td>2</td>
<td>Position angular speed (0.1[mm/sec])</td>
<td>SPEED_TYPE_VR</td>
</tr>
<tr>
<td>3</td>
<td>External axis speed (0.1[mm/sec])</td>
<td>SPEED_TYPE_VE</td>
</tr>
<tr>
<td>4</td>
<td>Target speed (0.1[mm/sec])</td>
<td>SPEED_TYPE_VS</td>
</tr>
</tbody>
</table>

<speedValue>

speed setting value (VJ=, V=, VR=, VE=, VS=)
VJ= 0.01[%] unit
Other than VJ= 0.1[mm/sec] unit
*Regardless of the setting for speed data input form, the unit is [mm/sec].
If there is no speed specified tag, speedValue=-1.

<posNum>
registered number of the position data in the move instruction (each one for manipulator + base + station, three in total)

<posData>
position data structure

- MP_JOB_POS_DATA
the position data structure

(Syntax):

typedef struct{
CTRLG_T    ctrl_grp;    /* control group */
LONG       posType;     /* position data type */
CTRLG_T    varIndex;    /* position variable number */
ULONG      attr;        /* data attribute */
ULONG      attrExt;     /* data attribute(expansion) */
LONG       pos[MP_GRP_AXES_NUM];    /* position data */
} MP_JOB_POS_DATA;
#define MAX_GRP_AXES_NUM 8 /* maximum control axis number */
8 System Monitor API

mpGetJobStepData

<ctrl_grp>
control group

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 to 7</td>
<td>R1(Robot1) to R8(Robot8)</td>
</tr>
<tr>
<td>8 to 15</td>
<td>B1(Base1) to B8(Base8)</td>
</tr>
<tr>
<td>16 to 39</td>
<td>S1(Station1) to S24(Station24)</td>
</tr>
</tbody>
</table>

<posType>
position data type

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>position data</td>
<td>JOB_POS_TYPE_POS</td>
</tr>
<tr>
<td>1</td>
<td>position variable</td>
<td>JOB_POS_TYPE_VAR</td>
</tr>
<tr>
<td>2</td>
<td>local position variable</td>
<td>JOB_POS_TYPE_LOCAL_VAR</td>
</tr>
</tbody>
</table>

<varIndex>
position variable number
When posType=1(position variable) or 2(local position variable), the position variable number is stored.
When posType=0 (position data), 0 is set.

<attr><attrExt><pos>
data attribute, data attribute (expansion), position data
The data structure is the same as that of mpGetPosVarData().

<table>
<thead>
<tr>
<th>Member name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpGetPosVarData()</td>
<td>Bit No.</td>
</tr>
<tr>
<td>attr</td>
<td>rData[0]</td>
</tr>
<tr>
<td></td>
<td>D05-D00 Variable type</td>
</tr>
<tr>
<td></td>
<td>0: Pulse</td>
</tr>
<tr>
<td></td>
<td>16: Cartesian (base coordinates)</td>
</tr>
<tr>
<td></td>
<td>17: Cartesian (robot coordinates)</td>
</tr>
<tr>
<td></td>
<td>18: Cartesian (tool coordinates)</td>
</tr>
<tr>
<td></td>
<td>19: Cartesian (user coordinates)</td>
</tr>
<tr>
<td></td>
<td>20: Cartesian (master tool)</td>
</tr>
<tr>
<td></td>
<td>D07-D06 Reserved by manufacturer</td>
</tr>
<tr>
<td></td>
<td>D08 0:Front 1:Back</td>
</tr>
<tr>
<td></td>
<td>D09 0:Upper arm 1:Lower arm</td>
</tr>
<tr>
<td></td>
<td>D10 0:Flip 1:No flip</td>
</tr>
<tr>
<td></td>
<td>D11 0:R&lt;180deg 1:R&gt;=180deg</td>
</tr>
<tr>
<td></td>
<td>D12 0:T&lt;180deg 1:T&gt;=180deg</td>
</tr>
<tr>
<td></td>
<td>D13 0:S&lt;180deg 1:S&gt;=180deg</td>
</tr>
<tr>
<td></td>
<td>D14-D15 Reserved by manufacturer</td>
</tr>
<tr>
<td></td>
<td>D16-D21 Tool number(0-63)</td>
</tr>
<tr>
<td></td>
<td>D22-D27 User coordinate number</td>
</tr>
<tr>
<td></td>
<td>D28-D31 Reserved by manufacturer</td>
</tr>
<tr>
<td>attrExt</td>
<td>rData[1] (Expansion attribute)</td>
</tr>
<tr>
<td>pos[0]</td>
<td>rData[2] 1st axis(S) pulse value</td>
</tr>
<tr>
<td>pos[1]</td>
<td>rData[3] 2nd axis(L) pulse value</td>
</tr>
</tbody>
</table>
8 System Monitor API
mpGetJobStepData

<table>
<thead>
<tr>
<th>pos</th>
<th>rData</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>3rd axis(U) pulse value</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Z-axis coordinate(unit:micron)</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>4th axis(R) pulse value</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>Wrist angle Rx(unit:0.0001deg)</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>5th axis(B) pulse value</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>Wrist angle Ry(unit:0.0001deg)</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>6th axis(T) pulse value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Wrist angle Rz(unit:0.0001deg)</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>7th axis(E) pulse value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>angle Re(unit:0.0001deg)</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>8th axis pulse value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8th axis pulse value(micron in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the case of traveling axis)</td>
</tr>
</tbody>
</table>

- **Return Value**
  - 0: Normal end
  - -1: Error (For details, check err_no.)
mpGetToolNo

Retrieves the operation tool number for the specified control group.

- **Syntax**
  ```c
  LONG mpGetToolNo(
      ULONG  sRobotNo,
      MP_GET_TOOL_NO_RSP_DATA*  rData
  );
  ```

- **Parameter**
  - **[sRobotNo]**
    [in] The number of the manipulator which retrieves the tool number

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_R1_GID</td>
<td>R1</td>
</tr>
<tr>
<td>MP_R2_GID</td>
<td>R2</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>MP_R8_GID</td>
<td>R8</td>
</tr>
</tbody>
</table>

  - **[rData]**
    [out] The pointer to the data structure which receives the operation tool number

    ```c
    typedef struct{
        USHORT err_no; /* error number */
        SHORT sToolNo; /* operation tool number to be retrieved */
        CHAR reserved[4];
    }MP_GET_TOOL_NO_RSP_DATA;
    ```

- **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.
mpGetEncoderTemp

Retrieves the celsius temperature of each axes of encoder.

- Syntax
  
  ```c
  LONG mpGetEncoderTemp(
      MP_CTRL_GRP_SEND_DATA *sData,
      MP_ENCODER_TEMP_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

<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>R3 (Robot 3)</td>
</tr>
<tr>
<td>3</td>
<td>R4 (Robot 4)</td>
</tr>
<tr>
<td>4</td>
<td>R5 (Robot 5)</td>
</tr>
<tr>
<td>5</td>
<td>R6 (Robot 6)</td>
</tr>
<tr>
<td>6</td>
<td>R7 (Robot 7)</td>
</tr>
<tr>
<td>7</td>
<td>R8 (Robot 8)</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>B3 (Base 3)</td>
</tr>
<tr>
<td>11</td>
<td>B4 (Base 4)</td>
</tr>
<tr>
<td>12</td>
<td>B5 (Base 5)</td>
</tr>
<tr>
<td>13</td>
<td>B6 (Base 6)</td>
</tr>
<tr>
<td>14</td>
<td>B7 (Base 7)</td>
</tr>
<tr>
<td>15</td>
<td>B8 (Base 8)</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>
<tr>
<td>19</td>
<td>S4 (Station 4)</td>
</tr>
<tr>
<td>20</td>
<td>S5 (Station 5)</td>
</tr>
<tr>
<td>21</td>
<td>S6 (Station 6)</td>
</tr>
</tbody>
</table>
8 System Monitor API

mpGetEncoderTemp

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>S7 (Station 7)</td>
</tr>
<tr>
<td>23</td>
<td>S8 (Station 8)</td>
</tr>
<tr>
<td>24</td>
<td>S9 (Station 9)</td>
</tr>
<tr>
<td>25</td>
<td>S10 (Station 10)</td>
</tr>
<tr>
<td>26</td>
<td>S11 (Station 11)</td>
</tr>
<tr>
<td>27</td>
<td>S12 (Station 12)</td>
</tr>
<tr>
<td>28</td>
<td>S13 (Station 13)</td>
</tr>
<tr>
<td>29</td>
<td>S14 (Station 14)</td>
</tr>
<tr>
<td>30</td>
<td>S15 (Station 15)</td>
</tr>
<tr>
<td>31</td>
<td>S16 (Station 16)</td>
</tr>
<tr>
<td>32</td>
<td>S17 (Station 17)</td>
</tr>
<tr>
<td>33</td>
<td>S18 (Station 18)</td>
</tr>
<tr>
<td>34</td>
<td>S19 (Station 19)</td>
</tr>
<tr>
<td>35</td>
<td>S20 (Station 20)</td>
</tr>
<tr>
<td>36</td>
<td>S21 (Station 21)</td>
</tr>
<tr>
<td>37</td>
<td>S22 (Station 22)</td>
</tr>
<tr>
<td>38</td>
<td>S23 (Station 23)</td>
</tr>
<tr>
<td>39</td>
<td>S24 (Station 24)</td>
</tr>
</tbody>
</table>

**[rData]**

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

- MP_ENCODER_TEMP_RSP_DATA
  
  [out] The data structure which receives the temperature value

  **Syntax:**
  ```c
  #define MAX_PULSE_AXES (8)
  typedef struct
  {
    LONG lTemp[MAX_PULSE_AXES];
  } MP_ENCODER_TEMP_RSP_DATA;
  ```

  **Member:**
  ```c
  <lTemp[MAX_PULSE_AXES]>
  Temperature of encoder (up to 8 arrays)
  ```

**Array**

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

**Return Value**

- 0 : Normal end
- -1 : Error
8  System Monitor API
mpGetEncoderTemp

Note

This API can be used in the following software configuration.

- System software: YBS1.31.00A-00 or later
- MotoPlusIDE: Ver2.01 or later
mpPutVarData

Writes variables (B, I, D, R).

- **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: `typedef struct
        {
        USHORT   usType;
        USHORT   usIndex;
        LONG     ulValue;
        } MP_VAR_DATA;`
      Member: `<usType>`
        Variable type
      Value | Description
      -----------------  
      MP_RESTYPE_VAR_B | Byte type
      MP_RESTYPE_VAR_I | Integer type
      MP_RESTYPE_VAR_D | Double-precision type
      MP_RESTYPE_VAR_R | Real type
      
      `<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

0  Normal end

-1  Error

- NOTE -

- Immediately after the job starts, the system has access to the variable area, and the function to write to the variable area such as mpPutVarData(), mpPutSVarInfo() 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 function which writes variables, I/O, or registers and the following modification methods other than MotoPlus, thus the MotoPlus function 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.

  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 function cannot be executed because the system is busy. In this case, the error (-1) is returned. Retry this function after a while.
mpPutSVarInfo

Writes the variable (S).

**Syntax**

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

**Parameter**

- **[sData]**
  Pointer to the data area which specifies variable

```c
define struct 
{
    USHORT usType; /* Variable type (Only MP_RESTYPE_VAR_S is valid) */
    USHORT usIndex; /* Variable index */
    UCHAR ucValue[S_VAR_SIZE+1]; /*S_VAR_SIZE (32 characters)+null character\0*/
    CHAR reserved[3];
} MP_SVAR_SEND_INFO;
```

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`.

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

**Return Value**

- 0 Normal end
- -1 Error

**NOTE**

- Since this function’s processing time is a few milliseconds, do not use this function in a task with the I/O control cycle or interpolation 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 function cannot be executed because the system is busy. In this case, the error (-1) is returned. Retry this function after a while.
mpWriteIO

Writes I/O.

Syntax

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.

When writing multiple values simultaneously, this function 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 writing the values. Thus, writing many values may affect the robot motion. DO NOT specify a large value as num during the robot motion.

It may take time to reflect the written value contents. DO NOT read the same value by mpReadIO() or mpMonitor() soon after executing this API.

[num]

Number of the I/O data (up to 126)
9  System Control API

mpWriteIO

<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>

There is no interlock between the MotoPlus function 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.

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

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

  **Member:**
  - **<usType>**
    Variable type
    - **Value** | **Description**
      | MP_RESTYPE_VAR_ROBOT | Robot |
      | MP_RESTYPE_VAR_BASE | Base |
      | MP_RESTYPE_VAR_STATION | Station |
  - **<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`. 
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>D07 - D06</td>
<td>Reserved by manufacturer</td>
<td></td>
</tr>
<tr>
<td>D08</td>
<td>0: Front 1: Back</td>
<td></td>
</tr>
<tr>
<td>D09</td>
<td>0: Upper arm 1: Lower arm</td>
<td></td>
</tr>
<tr>
<td>D10</td>
<td>0: Flip 1: No flip</td>
<td></td>
</tr>
<tr>
<td>D11</td>
<td>0: R &lt; 180 deg 1: R &gt;= 180 deg</td>
<td></td>
</tr>
<tr>
<td>D12</td>
<td>0: T &lt; 180 deg 1: T &gt;= 180 deg</td>
<td></td>
</tr>
<tr>
<td>D13</td>
<td>0: S &lt; 180 deg 1: S &gt;= 180 deg</td>
<td></td>
</tr>
<tr>
<td>D14 - D15</td>
<td>Reserved by manufacturer</td>
<td></td>
</tr>
<tr>
<td>D16 - D21</td>
<td>Tool number (0 - 63)</td>
<td></td>
</tr>
<tr>
<td>D22 - D27</td>
<td>User coordinate number</td>
<td></td>
</tr>
<tr>
<td>D28 - D31</td>
<td>Reserved by manufacturer</td>
<td></td>
</tr>
<tr>
<td>rData[1]</td>
<td>(Extended attribute)</td>
<td></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 23)

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

**NOTE**
Immediately after the job starts, the system has access to the variable area, and the function to write to the variable area such as mpPutVarData(), mpPutSVarInfo(), 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 function which writes variables, I/O, or registers and the following modification methods other than MotoPlus, thus the MotoPlus function 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 function cannot be executed because the system is busy. In this case, the error (-1) is returned. Retry this function 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 write user variable in high speed.

- **Syntax**

  ```c
  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:

    ```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)
<val>
Variable data buffer (Output)

Syntax:  
#define STR_VAR_SIZE 32
#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 63)

<uf_no>
User coordinate system file No. (0 to 62)

<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_ELBO</td>
<td>0: Uper 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
9  System Control API
mpPutUserVars

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>Any of following negative values:</td>
<td>Abnormal end</td>
</tr>
<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></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[10]</td>
<td>Bit No. p, bp, ex</td>
</tr>
<tr>
<td>Type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Front/Back</td>
<td>rData[0]</td>
<td>d0 to d5 .dtype</td>
</tr>
<tr>
<td>Upper elbow</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lower elbow</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flip/No flip</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R&lt;180/R&gt;=180</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T&lt;180/T&gt;=180</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S&lt;180/S&gt;=180</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tool No.</td>
<td>d16 to d21 .tool_no</td>
<td></td>
</tr>
<tr>
<td>User coordinate No.</td>
<td>d22 to d27 .uf_no</td>
<td></td>
</tr>
<tr>
<td>Position data</td>
<td>rData[2 to 9]</td>
<td>.data[0] to .data[7]</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 function, all the jobs under execution are stopped.

■ Parameter

- `[alm_code]`  
  Alarm code: 8000 to 8999

- `[alm_msg]`  
  Alarm message (up to 32 characters)

- `[sub_code]`  
  Alarm sub code

■ Return Value

- 0 Normal end
- ERROR Error (Alarm code range exceeded)

NOTE:

- Executing this function 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 function cannot be read by an alarm-reading function "mpGetAlarmStatus()" or "mpGetAlarmCode()" immediately after executing this function. This is because there is a time lag between the request of generating the alarm by this function 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
      - **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       |

- **Return Value**

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

**NOTE**

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

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

- mpCancelError()
- mpResetAlarm()
- mpSetCycle()
- mpSetServoPower()
- mpSetMasterJob()
- mpSetCurJob()
- mpStartJob()
- mpHold()
- mpWaitForJobEnd()
- mpDeleteJob()
- mpGetJogSpeed()
- mpGetJogCoord()
- mplMOV()
- mpMOVJ()
- mpMOVL()
- mpPulseMOVJ()
- mpPulseMOVL()
- mpManualMOV()
mpResetAlarm

Resets the robot alarm.

- **Syntax**
  
  ```c
  LONG mpResetAlarm ( 
  MP_STD_RSP_DATA* rData
  );
  ```

- **Parameter**
  
  **[rData]**
  
  [out] The pointer to 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

<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**

  0    Normal end
  
  -1   Error

**NOTE**

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

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

- mpCancelError(), mpResetAlarm(), mpSetCycle(),
- mpSetServoPower(), mpSetMasterJob(), mpSetCurJob(),
- mpStartJob(), mpHold(), mpWaitForJobEnd(),
- mpDeleteJob(), mpGetJogSpeed(), mpGetJogCoord(),
- mplMOV(), mpMOVJ(), mpMOVL(), mpPulseMOVJ(),
- mpPulseMOVL(), mpManualMOV()
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
  
  Syntax: `typedef struct {
    SHORT   sCycle;
    CHAR    reserved[2];
  } MP_CYCLE_SEND_DATA;`
  
  Member: `<sCycle>`
  
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Step</td>
</tr>
<tr>
<td>2</td>
<td>1Cycle</td>
</tr>
<tr>
<td>3</td>
<td>Auto</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>`
  
<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>
</tbody>
</table>
9 System Control API
mpSetCycle

■ 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 function and the functions listed below. Thus, if this function is executed while one of the following functions is already being executed, the task may be suspended to wait for the function to be completed. (waiting for semaphore acquisition)

Therefore, DO NOT use the following functions 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(), mpManualMOV()
mpSetServoPower

Sets the ON/OFF of the servo power.

Syntax

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Servo power ON</td>
</tr>
<tr>
<td>0</td>
<td>Servo power 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>
<tr>
<td>0x2060</td>
<td>In error/alarm status</td>
</tr>
<tr>
<td>0x3450</td>
<td>Failed (Unable to turn servo on)</td>
</tr>
</tbody>
</table>
9  System Control API
mpSetServoPower

- 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 function and the functions listed below. Thus, if this function is executed while one of the following functions is already being executed, the task may be suspended to wait for the function to be completed. (waiting for semaphore acquisition)

Therefore, DO NOT use the following functions 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(), mpManualMOV()
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:  
    ```
    typedef struct
    {
        SHORT sTaskNo;
        CHAR cJobName[MAX_JOB_NAME_LEN];
        CHAR reserved[5];
    } MP_MASTER_JOB_SEND_DATA;
    ```
    
    Member:  
    ```
    <sTaskNo>
    Task number
    ```
    
    | Value | Description |
    |-------|-------------|
    | 0     | Master task |
    | 1     | Subtask 1   |
    | 2     | Subtask 2   |
    | 3     | Subtask 3   |
    | 4     | Subtask 4   |
    | 5     | Subtask 5   |
    | 6     | Subtask 6   |
    | 7     | Subtask 7   |
    | 8     | Subtask 8   |
    | 9     | Subtask 9   |
    | 10    | Subtask 10  |
    | 11    | Subtask 11  |
    | 12    | Subtask 12  |
    | 13    | Subtask 13  |
    | 14    | Subtask 14  |
    | 15    | Subtask 15  |

- **[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>`

<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 function and the functions listed below. Thus, if this function is executed while one of the following functions is already being executed, the task may be suspended to wait for the function to be completed. (waiting for semaphore acquisition)

Therefore, DO NOT use the following functions 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(), mpManualMOV()

- 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>
### System Control API

#### mpSetCurJob

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>API execution processor received the command. Return value = 0 and err_no = 0 Normal end</td>
</tr>
<tr>
<td>-1</td>
<td>API execution processor does not respond.</td>
</tr>
<tr>
<td>0 and err_no != 0</td>
<td>Error (Check err_no.)</td>
</tr>
</tbody>
</table>

**NOTE**

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

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

- mpCancelError() |
- mpResetAlarm() |
- mpSetCycle() |
- mpSetServoPower() |
- mpSetMasterJob() |
- mpSetCurJob() |
- mpStartJob() |
- mpHold() |
- mpWaitForJobEnd() |
- mpDeleteJob() |
- mpGetJogSpeed() |
- mpGetJogCoord() |
- mplMOV() |
- mpMOVJ() |
- mpMOVL() |
- mpPulseMOVJ() |
- mpPulseMOVL() |
- mpManualMOV() |
mpStartJob

Starts a job.

- **Syntax**

  ```c
  LONG  mpStartJob (    
    MP_START_JOB_SEND_DATA*  sData,    
    MP_STD_RSP_DATA*        rData);  
  ```

- **Description**

  This function 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
          - Syntax: `#define MAX_JOB_NAME_LEN (33)`
          ```c
          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;
          ```
9 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.

Exclusive control by using semaphores is executed between this function and the functions listed below. Thus, if this function is executed while one of the following functions is already being executed, the task may be suspended to wait for the function to be completed. (waiting for semaphore acquisition)
Therefore, DO NOT use the following functions for the task which must be performed periodically with high priority.

mpCancelError(), mpResetAlarm(), mpSetCycle(), mpSetServoPower(), mpSetMasterJob(), mpSetCurJob(), mpStartJob(), mpHold(), mpWaitForJobEnd(), mpDeleteJob(), mpGetJogSpeed(), mpGetJogCoord(), mplMOV(), mpMOVJ(), mpMOVL(), mpPulseMOVJ(), mpPulseMOVL(), mpManualMOV()
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 function and the functions listed below. Thus, if this function is executed while one of the following functions is already being executed, the task may be suspended to wait for the function to be completed. (waiting for semaphore acquisition)

Therefore, DO NOT use the following functions 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()
- mpManualMOV()
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:

  ```c
  typedef struct {
    SHORT sTaskNo;
    SHORT sTime;
  } MP_WAIT_JOB_SEND_DATA;
  ```

  Member: 

<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>
<tr>
<td>6</td>
<td>Subtask 6</td>
</tr>
<tr>
<td>7</td>
<td>Subtask 7</td>
</tr>
<tr>
<td>8</td>
<td>Subtask 8</td>
</tr>
<tr>
<td>9</td>
<td>Subtask 9</td>
</tr>
<tr>
<td>10</td>
<td>Subtask 10</td>
</tr>
<tr>
<td>11</td>
<td>Subtask 11</td>
</tr>
<tr>
<td>12</td>
<td>Subtask 12</td>
</tr>
<tr>
<td>13</td>
<td>Subtask 13</td>
</tr>
<tr>
<td>14</td>
<td>Subtask 14</td>
</tr>
<tr>
<td>15</td>
<td>Subtask 15</td>
</tr>
</tbody>
</table>

**sTime**

Time (unit: sec, from 0 to 32767)
9 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 function and the functions listed below. Thus, if this function is executed while one of the following functions is already being executed, the task may be suspended to wait for the function to be completed. (waiting for semaphore acquisition)

Therefore, DO NOT use the following functions 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(), mpManualMOV()
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: #define MAX_JOB_NAME_LEN (33)
    ```c
    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: typedef struct  
      ```c
      {  
          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>
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 function and the functions listed below. Thus, if this function is executed while one of the following functions is already being executed, the task may be suspended to wait for the function to be completed. (waiting for semaphore acquisition)

Therefore, DO NOT use the following functions 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(), mpManualMOV()
mpSetJogCoord

Sets the operation coordinates for the specified control group.

- Syntax

  ```c
  LONG mpSetJogCoord(
      MP_JOGCOORD_SEND_DATA* sData,
      MP_STD_RSP_DATA* rData
  )
  ```

- Parameter

  [sData]

  [in] The pointer to the data structure which sends the operation coordinates

  - MP_JOGCOORD_SEND_DATA

  [in] The data structure which sends the operation coordinates

  Syntax:
  ```c
  typedef struct {
      ULONG sCtrlGrp; /* control group to set the operation coordinate*/
      SHORT sJogCoord; /* operation coordinate to be set */
      SHORT sUsrFrmNo; /* user coordinate number */
  } MP_JOGCOORD_SEND_DATA;
  ```

  Member:
  ```c
  <sCtrlGrp>
  The control groups which specifies the operation coordinates
  ```

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_R1_GID</td>
<td>R1</td>
</tr>
<tr>
<td>MP_R2_GID</td>
<td>R2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>MP_R8_GID</td>
<td>R8</td>
</tr>
<tr>
<td>MP_B1_GID</td>
<td>B1</td>
</tr>
<tr>
<td>MP_B2_GID</td>
<td>B2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>MP_B8_GID</td>
<td>B8</td>
</tr>
</tbody>
</table>

  [sJogCoord]

  The operation coordinates to be set.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_JJOINTCOORD</td>
<td>Joint coordinates</td>
</tr>
<tr>
<td>MP_RECTCOORD</td>
<td>Cartesian coordinates</td>
</tr>
<tr>
<td>MP_CYLINDCOORD</td>
<td>Cylindrical coordinates</td>
</tr>
<tr>
<td>MP_TOOLCOORD</td>
<td>Tool coordinates</td>
</tr>
<tr>
<td>MP_USERCOORD</td>
<td>User coordinates</td>
</tr>
<tr>
<td>MP_EXTREFCOORD</td>
<td>External reference point coordinates</td>
</tr>
</tbody>
</table>
<sUsrFrmNo>
The user coordinates number (1~63)
Sets the user coordinates number in case the operation coordinate is set as “User coordinates” or “External reference point coordinates”.

[rData]
[out] The pointer to the structure of the standard received data

• MP_STD_RSP_DATA
  [out] The structure of the standard received data
  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>Manipulator is in 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>0x3350</td>
<td>User coordinate is not registered</td>
</tr>
<tr>
<td>0x3360</td>
<td>Error in user coordinate file</td>
</tr>
<tr>
<td>0x3370</td>
<td>Wrong control group</td>
</tr>
<tr>
<td>0x3460</td>
<td>Coordinate system is not set</td>
</tr>
<tr>
<td>0x3600</td>
<td>User coordinate number is out of range</td>
</tr>
</tbody>
</table>

## Return Value
- 0 : Normal end
- None zero : Error
• Operation coordinates system needs to be specified to each control group.

• The initial value of the operation coordinate is set as “Link Coordinate” for every groups. When the control power of the YRC1000micro is turned ON, the operation coordinate system setting turns back to the initial value.

• In the case of the followings, error number:0x3460 (Coordinate system is not set) occurs and the the coordinate system cannot be changed.
  
  – The “external reference point” is specified while the “external reference point (optional)” is invalid.

  – In the “select cartesian/cylindrical” (=parameter: S2C196), which is to be specified in the teaching condition,
    The “cylindrical coordinate” is specified while the “cartesian” (S2C196=1) is selected.
    The “cartesian coordinate” is specified while the “cylindrical” (S2C196=0) is selected.

  – In the “coordinate switch prohibited” (=parameter: S2C197),
    The “tool coordinate” is specified while the “tool coordinate switch prohibited” (S2C197=1 or 3) is selected.
    The “user coordinate” is specified while the “user coordinate switch prohibited” (S2C197=2 or 3) is selected.

S2C197:coordinate switch prohibited

  0: user/tool coordinate switch is available.
  1: tool coordinate switch is prohibited.
  2: coordinate switch is prohibited.
  3: user/tool coordinate switch is available.
mpSetToolNo

Sets the jog tool number for the specified control group.

Syntax

```c
LONG mpSetToolNo(
    MP_SET_TOOL_NO_SEND_DATA* sData,
    MP_STD_RSP_DATA* rData
)
```

Parameter

<sData>
[in] The pointer to the data structure which sends jog tool numbers

- MP_SET_TOOL_NO_SEND_DATA

Syntax:
```c
typedef struct {
    ULONG sRobotNo; /* robot number to set the jog tool number*/
    SHORT sToolNo; /* jog tool number to be set */
    SHORT reserved[2];
} MP_SET_TOOL_NO_SEND_DATA;
```

Member:

<sRobotNo>
The control group which sets the operation coordinate

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_R1_GID</td>
<td>R1</td>
</tr>
<tr>
<td>MP_R2_GID</td>
<td>R2</td>
</tr>
<tr>
<td></td>
<td>:</td>
</tr>
<tr>
<td>MP_R8_GID</td>
<td>R8</td>
</tr>
</tbody>
</table>

<sToolNo>
The jog tool number to be set (0~63).

<rData>
[out] The pointer to the structure of the standard received data

- MP_STD_RSP_DATA

Syntax:
```c
typedef struct {
    USHORT err_no;
    CHAR reserved[2];
} MP_STD_RSP_DATA;
```
9 System Control API
mpSetToolNo

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>0x2060</td>
<td>In error/alarm status</td>
</tr>
<tr>
<td>0x2080</td>
<td>Wrong operation mode</td>
</tr>
<tr>
<td>0x3370</td>
<td>Wrong control group (Setting error in sRobotNo)</td>
</tr>
<tr>
<td>0x3470</td>
<td>Tool is not changed</td>
</tr>
</tbody>
</table>

- Jog tool number is memorized to each robot, so it needs setting each.
- Initial values are set as R1:tool#0, R2:tool#1, ... etc.
The tool number is memorized even if the control power is turned OFF.
- When "tool number switch: unvalid" (=S2C431: tool switch permit specification=0) is specified, error number: 0x3470 (Tool is not changed) appears, and tools are not switched.

Return Value
0: Normal end
None zero: Error

**NOTE**
- Jog tool number is memorized to each robot, so it needs setting each.
- Initial values are set as R1:tool#0, R2:tool#1, ... etc.
The tool number is memorized even if the control power is turned OFF.
- When "tool number switch: unvalid" (=S2C431: tool switch permit specification=0) is specified, error number: 0x3470 (Tool is not changed) appears, and tools are not switched.
mpSetJobPosData

Changes the teaching position data of the step number in the specified job name.

- Syntax
  
  ```c
  LONG mpSetJobPosData(
      MP_JOB_POS_SEND_DATA* sData
      /* the data structure which specifies the teaching position */
      MP_SET_JOB_POS_RSP_DATA* rData
      /* the data structure which receives the change result of teaching position data */
  );
  ```

- Parameter
  
  - **[sData]**
    
    [in] The pointer to the data structure which specifies the teaching position

  - **MP_JOB_POS_SEND_DATA**
    
    Syntax:
    ```c
typedef struct{
  USHORT usStep; /* step number */
  CHAR reserved0[2];
  CHAR cJobName[MAX_JOB_NAME_LEN]; /* job name*/
  CHAR reserved1[3];
  LONG posNum; /*teaching position data number */
  MP_JOB_POS_DATA posData[MAX_JOB_POS_NUM];
  /* teaching position data structure */
} MP_JOB_POS_SEND_DATA;
```

  Defined:
  ```
  #define MAX_JOB_NAME_LEN 33 /* up to 32 characters + null character \0' */
  #define MAX_JOB_POS_NUM 12 /* maximum teaching position data number */
  ```

  Member:
  - `<usStep>`
    
    step number (1 to 99988)
  - `<cJobName[MAX_JOB_NAME_LEN]>`
    
    job name (up to 32 characters)
  - `<posNum>`
    
    teaching position data number
  - `<posData>`
    
    teaching position data structure

  - **MP_JOB_POS_DATA**
    
    teaching position data structure

    Syntax:
    ```c
typedef struct{
  CTRLG_T ctrl_grp; /* control group */
  LONG posType; /* position data type */
  CTRLG_T varIndex; /* position variable number */
  ULONG attr; /* data attribute */
  ULONG attrExt; /* data attribute (expansion) */
} MP_JOB_POS_DATA;
```
9 System Control API
mpSetJobPosData

LONG pos[MP_GRP_AXES_NUM]; /* position data */
} MP_JOB_POS_DATA;
#define MAX_GRP_AXES_NUM 8 /* maximum control axis number*/

<ctrl_grp>
control group

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 to 7</td>
<td>R1(Robot1) to R8(Robot8)</td>
</tr>
<tr>
<td>8 to 15</td>
<td>B1(Base1) to B8(Base8)</td>
</tr>
<tr>
<td>16 to 39</td>
<td>S1(Station1) to S24(Station24)</td>
</tr>
</tbody>
</table>

<posType>
position data type
A local position variable cannot be specified.
When the permission to change the local variable (S2C1115=1) in
mpSetJobPosData(), the local variable can be changed.
When the prohibition to change the local variable (S2C1115=0) in
mpSetJobPosData(), the local variable cannot be changed

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Position data</td>
<td>JOB_POS_TYPE_POS</td>
</tr>
<tr>
<td>1</td>
<td>Position variable</td>
<td>JOB_POS_TYPE_VAR</td>
</tr>
</tbody>
</table>

<varIndex>
position variable number
When posType=1 (position variable), the index for the position variable
data to be changed is specified.
When posType=0(position data), the position variable number is not used.

<attr><attrExt><pos>
data attribute, data attribute (expansion), position data
The data structure is the same as that of mpPutPosVarData().

<table>
<thead>
<tr>
<th>Member name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpPutPosVarData()</td>
<td></td>
</tr>
<tr>
<td>attr rData[0]</td>
<td></td>
</tr>
<tr>
<td>Bit No.</td>
<td>Contents</td>
</tr>
<tr>
<td>D05-D00</td>
<td>Variable type</td>
</tr>
<tr>
<td>0</td>
<td>Pulse</td>
</tr>
<tr>
<td>16</td>
<td>Cartesian (base coordinates)</td>
</tr>
<tr>
<td>17</td>
<td>Cartesian (robot coordinates)</td>
</tr>
<tr>
<td>18</td>
<td>Cartesian (tool coordinates)</td>
</tr>
<tr>
<td>19</td>
<td>Cartesian(user coordinates)</td>
</tr>
<tr>
<td>20</td>
<td>Cartesian(master tool)</td>
</tr>
<tr>
<td>D07-D06</td>
<td>Reserved by manufacturer</td>
</tr>
<tr>
<td>D08</td>
<td>0:Front 1:Back</td>
</tr>
<tr>
<td>D09</td>
<td>0:Upper arm 1:Lower arm</td>
</tr>
<tr>
<td>D10</td>
<td>0:Flip 1:No flip</td>
</tr>
<tr>
<td>D11</td>
<td>0:R&lt;180deg 1:R&gt;=180deg</td>
</tr>
<tr>
<td>D12</td>
<td>0:T&lt;180deg 1:T&gt;=180deg</td>
</tr>
<tr>
<td>D13</td>
<td>0:S&lt;180deg 1:S&gt;=180deg</td>
</tr>
<tr>
<td>D14-D15</td>
<td>Reserved by manufacturer</td>
</tr>
</tbody>
</table>
mpSetJobPosData

Member name | Pulse type | Cartesian type |
---|---|---|
pos[0] | rData[2] | 1st axis(S) pulse value | X-axis coordinate(unit:micron) |
pos[3] | rData[5] | 4th axis(R) pulse value | Wrist angle Rx(unit:0.0001deg) |
pos[4] | rData[6] | 5th axis(B) pulse value | Wrist angle Ry(unit:0.0001deg) |
pos[5] | rData[7] | 6th axis(T) pulse value | Wrist angle Rz(unit:0.0001deg) |
pos[6] | rData[8] | 7th axis(E) pulse value | angle Re(unit:0.0001deg) |
pos[7] | rData[9] | 8th axis pulse value | 8th axis pulse value(micron in the case of traveling axis) |

Value | Description |
---|---|
0x0000 | Normal end |
0x2010 | Robot is in operation |
0x2060 | In error/alarm status |
0x2080 | Wrong operation mode |
0x3370 | Wrong control group |
0x3470 | Tool is not changed |
0x3600 | User coordinate number is out of range |
0x3620 | Error in specifying tool file number |
0x3630 | Error in data type |
0x4020 | Edit lock job |
0x4040 | Specified JOB not found |
0x4130 | No position data |
0x4140 | Wrong position data type |

[out] The pointer to the data structure which receives the change result of teaching position data

- **MP_SET_JOB_POS_RSP_DATA**

```c
typedef struct{
  USHORT  err_no;  /* error number */
  USHORT  err_sub;  /* error sub-number */
  USHORT  err_index;  /* error status index */
  CHAR    reserved[2];
} MP_SET_JOB_POS_RSP_DATA;
```

Member: `<err_no>`
error number
**mpSetJobPosData**

The index which has an error in posData (the teaching position data structure) is specified.

<table>
<thead>
<tr>
<th>Error number</th>
<th>Error sub-number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x3630</td>
<td>0x0010</td>
<td>Error in specifying data type</td>
</tr>
<tr>
<td></td>
<td>0x0020</td>
<td>Data type does not coincide between manipulator and base</td>
</tr>
<tr>
<td></td>
<td>0x0030</td>
<td>Error in specifying master tool coordinate</td>
</tr>
<tr>
<td>0x5200</td>
<td>0x0010</td>
<td>Error in specifying step number</td>
</tr>
<tr>
<td></td>
<td>0x0040</td>
<td>Error in specifying position data type</td>
</tr>
<tr>
<td></td>
<td>0x0100</td>
<td>Error in specifying position data (1st axis (S) pulse value)</td>
</tr>
<tr>
<td></td>
<td>0x0101</td>
<td>Error in specifying position data (2nd axis (L) pulse value)</td>
</tr>
<tr>
<td></td>
<td>0x0102</td>
<td>Error in specifying position data (3rd axis (U) pulse value)</td>
</tr>
<tr>
<td></td>
<td>0x0103</td>
<td>Error in specifying position data (4th axis (R) pulse value)</td>
</tr>
<tr>
<td></td>
<td>0x0104</td>
<td>Error in specifying position data (5th axis (B) pulse value)</td>
</tr>
<tr>
<td></td>
<td>0x0105</td>
<td>Error in specifying position data (6th axis (T) pulse value)</td>
</tr>
<tr>
<td></td>
<td>0x0106</td>
<td>Error in specifying position data (7th axis (E) pulse value)</td>
</tr>
<tr>
<td></td>
<td>0x0107</td>
<td>Error in specifying position data (8th axis pulse value)</td>
</tr>
<tr>
<td></td>
<td>0x0110</td>
<td>Error in specifying position data (Cartesian type)</td>
</tr>
<tr>
<td>0x5210</td>
<td>0x0020</td>
<td>Teaching position data number does not coincide</td>
</tr>
<tr>
<td></td>
<td>0x0030</td>
<td>Control group does not coincide</td>
</tr>
<tr>
<td></td>
<td>0x0040</td>
<td>Position data type does not coincide</td>
</tr>
<tr>
<td></td>
<td>0x0050</td>
<td>Position variable number does not coincide</td>
</tr>
<tr>
<td></td>
<td>0x0060</td>
<td>User coordinate number does not coincide</td>
</tr>
</tbody>
</table>

**Return Value**

0: Normal end

-1: Error (For details, check err_no.)
mpParallelShiftJob

Shifts the specified step by the same deviation and converts the job.

- **Syntax**
  ```c
  LONG mpParallelShiftJob(
    MP_PARALLEL_SHIFT_JOB_SEND_DATA* sData,
    /* the data structure which specifies the parallel shift job conversion */
    MP_STD_RSP_DATA* rData
    /* the data structure which receives the error information*/
  );
  ```

- **Parameter**
  
  **[sData]**
  
  [in] The pointer to the data structure which specifies the parallel shift job conversion

  - **MP_PARALLEL_SHIFT_JOB_SEND_DATA**
    
    Syntax:
    ```c
typedef struct{
  CHAR srcJobName[MAX_JOB_NAME_LEN]; /* source job name */
  CHAR reserved0[3];
  CHAR dstJobName[MAX_JOB_NAME_LEN]; /* destination job name*/
  CHAR reserved1[3];
  USHORT usStartStep; /* conversion start step */
  USHORT usEndStep; /* conversion end step */
  USHORT usCoord; /* conversion coordinate */
  USHORT usUsrFrmNo; /* user coordinate number */
  USHORT usStationUnit; /* station data unit */
  CHAR reserved2[2];
  LONG shiftNum; /* shift data number */
  MP_SHIFT_VALUE_DATA shiftData[MAX_SHIFT_VALUE_NUM]; /* shift data structure */
} MP_PARALLEL_SHIFT_JOB_SEND_DATA;
```  

  - **Member:**
    
    `<srcJobName[MAX_JOB_NAME_LEN]>`
    
    source job name (up to 32 characters)

    `<dstJobName[MAX_JOB_NAME_LEN]>`
    
    destination job name (up to 32 characters)

    `<usStartStep>`
    
    conversion start step number (1 to 9998)

    `<usEndStep>`
    
    conversion end step number (1 to 9998)

    `<usCoord>`
    
    conversion coordinate

    0: pulse

    16: base

    17: manipulator

    18: tool
9 System Control API
mpParallelShiftJob

19: user
20: master tool

<usUsrFrmNo>
user coordinate number (1 to 63)

<usStationUnit>
future function (Set 0.)

<shiftNum>
shift data number
Set the number of control groups in the source job.
only R1:  1
R1+B1:  2
R1+R2:  2

• MP_SHIFT_VALUE_DATA

typedef struct{
CTRLG_T  ctrl_grp; /* control group*/
LONG     val[MP_GRP_AXES_NUM]; /* shift data*/
} MP_SHIFT_VALUE_DATA;
#define MAX_GRP_AXES_NUM 8 /* maximum control axis number */

<ctrl_grp>
control group

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 to 7</td>
<td>R1(Robot1) to R8(Robot8)</td>
</tr>
<tr>
<td>8 to 15</td>
<td>B1(Base1) to B8(Base8)</td>
</tr>
<tr>
<td>16 to 39</td>
<td>S1(Station1) to S24(Station24)</td>
</tr>
</tbody>
</table>

<val>
shift data

<table>
<thead>
<tr>
<th>Shift data</th>
<th>Pulse type</th>
<th>Cartesian type</th>
</tr>
</thead>
<tbody>
<tr>
<td>val[0]</td>
<td>1st axis (S) pulse value</td>
<td>X-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>val[1]</td>
<td>2nd axis (L) pulse value</td>
<td>Y-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>val[2]</td>
<td>3rd axis (U) pulse value</td>
<td>Z-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>val[3]</td>
<td>4th axis (R) pulse value</td>
<td>Wrist angle Rx(unit:0.0001deg)</td>
</tr>
<tr>
<td>val[4]</td>
<td>5th axis (B) pulse value</td>
<td>Wrist angle Ry(unit:0.0001deg)</td>
</tr>
<tr>
<td>val[5]</td>
<td>6th axis (T) pulse value</td>
<td>Wrist angle Rz(unit:0.0001deg)</td>
</tr>
<tr>
<td>val[6]</td>
<td>7th axis (E) pulse value</td>
<td>angle Re(unit:0.0001deg)</td>
</tr>
<tr>
<td>val[7]</td>
<td>8th axis pulse value</td>
<td>8th axis pulse value (micron in the case of traveling axis)</td>
</tr>
</tbody>
</table>
mpParallelShiftJob

[rData]

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

- **MP_STD_RSP_DATA**

  Syntax:
  ```
  typedef struct{
      USHORT err_no; /* error number */
      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>0x2060</td>
<td>In error/alarm status</td>
</tr>
<tr>
<td>0x2080</td>
<td>Wrong operation mode</td>
</tr>
<tr>
<td>0x2090</td>
<td>Data being accessed by another function</td>
</tr>
<tr>
<td>0x3350</td>
<td>User coordinate is not registered</td>
</tr>
<tr>
<td>0x3370</td>
<td>Wrong control group</td>
</tr>
<tr>
<td>0x3600</td>
<td>User coordinate number is out of range</td>
</tr>
<tr>
<td>0x3630</td>
<td>Error in data type</td>
</tr>
<tr>
<td>0x4020</td>
<td>Edit lock job</td>
</tr>
<tr>
<td>0x4030</td>
<td>The JOB name is already registered in another task.</td>
</tr>
<tr>
<td>0x4031</td>
<td>Error in job name</td>
</tr>
<tr>
<td>0x4040</td>
<td>Specified JOB not found</td>
</tr>
<tr>
<td>0x5200</td>
<td>Out of data range</td>
</tr>
<tr>
<td>0x5210</td>
<td>Data does not coincide</td>
</tr>
<tr>
<td>0x6010</td>
<td>Lack of job capacity</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

- The parallel shift job conversion using the conversion reference point is not available.
- The shift amount specification of the parallel shift job conversion (parameter S2C652) is not available.
- The position type variable used in the job is not subject to the parallel shift job conversion.
- The following jobs are not converted.
  - A job without a group
  - A concurrent job (option)
• The conversion process may take time depending on the number of step registered in job.

• The following operation cannot be performed in the conversion process.
  – Job execution
  – Displaying the job window (When the conversion process is executed while the job window is displayed, the job window is closed and a blank window is displayed.)

• The following sub menus are concealed.(job contents, select job, create new job, call master job, supervisory and select micro job)

• If the operation mode is changed in the conversion process, the process is canceled. (The conversion cannot be executed in the play mode. In addition, the contents in the middle of conversion are abandoned.)
mpApplicationInfoNotify

Notify MotoPlus application information to the controller.
The notified application information is displayed on the version information
window of the programming pendant.

Syntax

```c
int mpApplicationInfoNotify(
    MP_APPINFO_SEND_DATA* sData,
    MP_STD_RSP_DATA* rData);
```

Parameter

**[sData]**

[in] The pointer to the data structure which sends application information

- **MP_APPINFO_SEND_DATA**

[in] The data structure which sends application information

Syntax:

```c
#define MP_MAX_APP_NAME 32  // Max length of characters for application name
#define MP_MAX_APP_VERSION 16  // Max length of characters for version number
#define MP_MAX_APP_COMMENT 32  // Max length of characters for comment

typedef struct
{
    CHAR     AppName[MP_MAX_APP_NAME+4];  /* Name of Application */
    CHAR     Version[MP_MAX_APP_VERSION+4];   /* VersionNo. */
    CHAR     Comment[MP_MAX_APP_COMMENT+4];   /* Comment */
    CHAR     reserved[36];
} Member:

<AppName> Application name Character strings of max 32 characters
-Version> Version number Character strings of max 16 characters
-<Comment> Comment (free space) Character strings of max 32 characters
```

**[rData]**

[out] The pointer to the structure of the standard received data

- **MP_STD_RSP_DATA**

[out] The structure of the standard received data

Member:  <err_no>

Error number

Normal end 0x0000
Error  0x0001 The number of characters is exceeded.
       0x0002 Application name is not input.
9 System Control API
mpApplicationInfoNotify

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

- **Note**
  Only half-width numbers, letters, and symbols can be displayed.
10 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**

<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>

Negative value means failure.
mpMotStop

Stops the operation (target execution).

- **Syntax**

  ```
  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**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;= 0</td>
<td>Success</td>
</tr>
<tr>
<td>&lt; 0</td>
<td>Failure</td>
</tr>
</tbody>
</table>
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

- `>= 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>Blt</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_INTPTYPE 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_INTPTYPE;
```

```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 ex1, ex2;
} 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></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO_WAIT</td>
<td>Doesn’t wait</td>
</tr>
<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.
10 Motion Control API
mpMotTargetReceive

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).

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO_WAIT</td>
<td>End immediately even if the report is not received</td>
</tr>
<tr>
<td>WAIT_FOREVER</td>
<td>Keep waiting without timeout</td>
</tr>
</tbody>
</table>

$options$
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()`.

    | Value | Meaning         |
    |-------|-----------------|
    | 0     | The first control group |
    | 1     | The second control group |
    | 2     | The third control group |
    | 3     | The fourth control group |

  - **[type]**
    
    Specify the type of the coordinate system.

    | Value            | Meaning                  | Auxiliary Data                                    |
    |------------------|--------------------------|--------------------------------------------------|
    | MP_PULSE_TYPE    | Pulse                    | Not used                                         |
    | MP_ANGLE_TYPE    | Angle                    | Not used                                         |
    | MP_BASE_TYPE     | Base coordinate system   | Not used                                         |
    | MP_ROBOT_TYPE    | Robot System             | Not used                                         |
    | MP_USER_TYPE     | User coordinate system   | User coordinate system number 0~15(#01~#16)     |
### aux

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

<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;
} 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

<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>
mpMotSetTool

specifies the tool number for the target.

■ Syntax

```c
int mpMotSetTool
(int grpNo, /* control group number */
 int toolNo /* tool number */
);
```

■ Description

Set the tool number specified in `toolNo` to the control group specified in `grpNo`. 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 `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>

- **[toolNo]**
  specify the tool number.

■ Return Value

<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_TOOL_OUTRNG</td>
<td>Specified tool number is out of range.</td>
<td>Specify the correct tool number.</td>
</tr>
<tr>
<td>E_MP_MOT_TOOL_SET_INHIBIT</td>
<td>Unable to change tool number because “tool number change: prohibited” is specified.</td>
<td>Specify “Tool number change: permitted”.</td>
</tr>
</tbody>
</table>
10  Motion Control API

mpMotSetOrigin

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 (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.
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>

### NOTICE

- 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.

  ```c
  mpMotSetSync(grpNo, aux, options);
  ```

```
```
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>

NOTICE

- 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>

**NOTICE**

- 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.
mpMotSetConfig

Specify the robot configuration of the specified control group.

- Syntax

```c
int mpMotSetConfig
    ( int grpNo, /**< Control group number */
    ULONG config /**< Robot configuration */
)
```

- Description

This routine specifies the robot configuration of the control group specified in `grpNo`. In `grpNo`, 0 indicates the first control group in the system, 1 indicates the second control group. Subsequent groups follow the same format. In a system with only a manipulator (R1), 0 indicates R1.

- 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>
```

- `[config]`

Specifies the robot configuration by bit units in the following table.

```
<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>
```
## 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>The specified control group does not exist.</td>
<td>Specify the correct control group.</td>
</tr>
</tbody>
</table>
mpMotSetAccuracy

Specifies the positioning level of the specified control group.

- **Syntax**

  ```c
  int mpMotSetAccuracy(
      int grpNo, /* Control group number */
      long acr /* Positioning level */
  )
  ```

- **Description**

  This routine specifies the positioning level of the control group specified in `grpNo`.

  In `grpNo`, 0 indicates the first control group in the system, 1 indicates the second control group. Subsequent groups follow the same format.

  In a system with only a manipulator (R1), 0 indicates R1.

- **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>

  **[acr]**

  Specify the positioning level in [μm] units. If `acr` is the negative number, the positioning level is set as invalid.
10 Motion Control API
mpMotSetAccuracy

- **Return Value**

  >= 0  
  < 0  

  Success  
  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>The specified control group does not exist.</td>
<td>Specify the correct control group.</td>
</tr>
</tbody>
</table>
mpMotSetAccel

Sets the target acceleration.

Syntax

```c
int mpMotSetAccel
    (    int grpNo, /* Control group number */
         long accel /* Acceleration adjustment ratio [0.01%] */
    )
```

Description

Set the ratio specified in `accel` to the control group specified in `grpNo`.

This routine reduces the amount of acceleration in the specified ratio.

The set details are reflected in all targets that are sent after this routine is called.

Acceleration/deceleration adjustment ratio

The acceleration/deceleration adjustment ratio (accel/decel) reduces the amount of acceleration/deceleration in the specified ratio.

Using this function can reduce the load inertia on the tool and the workpiece.
10 Motion Control API

mpMotSetAccel

Parameter

[grpNo]
Specify the target control group.
Specify the control group number retrieved by the mpCtrlGrpld2GrpNo().

<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>

[accel]
Specify the acceleration adjustment ratio in 0.01% units.
Specify the acceleration adjustment ratio in the range of 2000 (20%) to 10000 (100%).

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>The specified control group does not exist.</td>
<td>Specify the correct control group.</td>
</tr>
<tr>
<td>E_MP_MOT_ACC_OUTRNG</td>
<td>The specified acceleration adjustment ratio is out of range.</td>
<td>Specify the correct acceleration.</td>
</tr>
</tbody>
</table>

Note

This API can be used in the following software configuration.

- System software: YBS1.31.00A-00 or later
- MotoPlusIDE: Ver2.01 or later
mpMotSetDecel

Sets the target deceleration.

- Syntax

```c
int mpMotSetDecel
(
    int grpNo, /* Control group number */
    long decel /* Deceleration adjustment ratio [0.01%] */
)
```

- Description

Set the deceleration adjustment ratio in `decel` to the control group specified in `grpNo`.

This routine reduces the amount of deceleration in the specified ratio.

The set details are reflected in all targets that are sent after this routine is called.

---

**Acceleration/deceleration adjustment ratio**

The acceleration/deceleration adjustment ratio (accel/decel) reduces the amount of acceleration/deceleration in the specified ratio.

Using this function can reduce the load inertia on the tool and the workpiece.
10 Motion Control API

mpMotSetDecel

- **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>

  **[decel]**
  Specify the deceleration adjustment ratio in 0.01% units.
  Specify the deceleration adjustment ratio in the range of 2000 (20%) to 10000 (100%).

- **Return Value**

  
<table>
<thead>
<tr>
<th>Return Value</th>
<th>Meaning</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;= 0</td>
<td>Success</td>
<td></td>
</tr>
<tr>
<td>&lt; 0</td>
<td>Failure</td>
<td></td>
</tr>
</tbody>
</table>

  Negative value means failure.

  **[E_MP_MOT_GRP_NOT_EXIST]**
  The specified control group does not exist.
  Specify the correct control group.

  **[E_MP_MOT_DEC_OUTRNG]**
  The specified deceleration adjustment ratio is out of range.
  Specify the correct deceleration.

- **Note**

  This API can be used in the following software configuration.

  - System software: YBS1.31.00A-00 or later
  - MotoPlusIDE: Ver2.01 or later
11 Motion Monitor/Control API

mplMOV

Moves the current position of the manipulator with the increment value of linear motion.

- Syntax

```c
LONG  mplMOV ( 
    MP_IMOV_SEND_DATA*   sData,
    MP_STD_RSP_DATA*   rData 
 );
```

- Parameter

[sData]

[in] The pointer to the data structure which transmits the information of increment move

- MP_IMOV_SEND_DATA

[in] The data structure which transmits the information of increment move

```c
#define MAX_NO_OF_AXES (12)

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>R3 (Robot 3)</td>
</tr>
<tr>
<td>3</td>
<td>R4 (Robot 4)</td>
</tr>
<tr>
<td>4</td>
<td>R5 (Robot 5)</td>
</tr>
<tr>
<td>5</td>
<td>R6 (Robot 6)</td>
</tr>
<tr>
<td>6</td>
<td>R7 (Robot 7)</td>
</tr>
<tr>
<td>7</td>
<td>R8 (Robot 8)</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>B3 (Base 3)</td>
</tr>
<tr>
<td>11</td>
<td>B4 (Base 4)</td>
</tr>
<tr>
<td>12</td>
<td>B5 (Base 5)</td>
</tr>
</tbody>
</table>
### Value | Description
--- | ---
13 | B6 (Base 6)
14 | B7 (Base 7)
15 | B8 (Base 8)
16 | S1 (Station 1)
17 | S2 (Station 2)
18 | S3 (Station 3)
19 | S4 (Station 4)
20 | S5 (Station 5)
21 | S6 (Station 6)
22 | S7 (Station 7)
23 | S8 (Station 8)
24 | S9 (Station 9)
25 | S10 (Station 10)
26 | S11 (Station 11)
27 | S12 (Station 12)
28 | S13 (Station 13)
29 | S14 (Station 14)
30 | S15 (Station 15)
31 | S16 (Station 16)
32 | S17 (Station 17)
33 | S18 (Station 18)
34 | S19 (Station 19)
35 | S20 (Station 20)
36 | S21 (Station 21)
37 | S22 (Station 22)
38 | S23 (Station 23)
39 | S24 (Station 24)

**<lSpeed>**

Motion speed
0.1 to S1C*G17 (linear speed for registration) [mm/s]
0.1 to S1C*G25 (position angle speed) [deg/s]
For more information on each parameter, refer to "Chap. 8.2.0.4 Linear Speed for Registration" and "Chap. 8.2.0.5 Position Angle Speed" in "YRC1000micro OPERATOR'S MANUAL (RE-CSO-A058)".

**<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, ...64</td>
<td>User coordinate 1, 2, ...63</td>
</tr>
</tbody>
</table>

### <sToolNo>

**Tool number (0 to 63)**

### <lPos[MAX_NO_OF_AXES]>

**Specified position (up to 12 arrays)**

#### <<Robot axis>>

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPos[0]</td>
<td>X-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>IPos[1]</td>
<td>Y-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>IPos[2]</td>
<td>Z-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>IPos[3]</td>
<td>Wrist angle Rx (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>IPos[4]</td>
<td>Wrist angle Ry (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>IPos[5]</td>
<td>Wrist angle Rz (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>IPos[6]</td>
<td>Elbow angle E (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>IPos[7]</td>
<td>8th axis pulse value (micron in the case of traveling axis)</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>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].
Motion Monitor/Control API
mplMOV

[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>Home position is not registered</td>
</tr>
<tr>
<td>0x3050</td>
<td>Out of range (ABSO data)</td>
</tr>
<tr>
<td>0x3061</td>
<td>Each axis operation range limit function is temporarily disable ¹)</td>
</tr>
<tr>
<td>0x3062</td>
<td>Robot operation range limit function is temporarily disable ¹)</td>
</tr>
<tr>
<td>0x3063</td>
<td>Tool angle monitoring function is temporarily disable ¹)</td>
</tr>
<tr>
<td>0x3064</td>
<td>Tool switch monitoring function is temporarily disable ¹)</td>
</tr>
<tr>
<td>0x3065</td>
<td>External axis coasting value setting is not done ¹)</td>
</tr>
<tr>
<td>0x3350</td>
<td>User coordinate is not registered</td>
</tr>
<tr>
<td>0x3370</td>
<td>Wrong control group</td>
</tr>
</tbody>
</table>

¹ Only when the safety function (option) is valid

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 mplMOV(), mpMOVJ(), mpMOVL(), mpPulseMOVJ(), and mpPulseMOVL(), mpManualMOV() 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()
- mpManualMOV()
mpMOVJ

Moves the manipulator to the specified position with the joint motion.

- **Syntax**

  ```c
  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: &lt;sCtrlGrp&gt;

    Control group

    | Value | Description   |
    |-------|--------------|
    | 0     | R1 (Robot 1) |
    | 1     | R2 (Robot 2) |
    | 2     | R3 (Robot 3) |
    | 3     | R4 (Robot 4) |
    | 4     | R5 (Robot 5) |
    | 5     | R6 (Robot 6) |
    | 6     | R7 (Robot 7) |
    | 7     | R8 (Robot 8) |
    | 8     | B1 (Base 1)  |
    | 9     | B2 (Base 2)  |
    | 10    | B3 (Base 3)  |
    | 11    | B4 (Base 4)  |
    | 12    | B5 (Base 5)  |
    | 13    | B6 (Base 6)  |
    | 14    | B7 (Base 7)  |
Motion Monitor/Control API

<\(\text{Speed}\)>
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, ..., 64</td>
<td>User coordinate 1, 2, ..., 63</td>
</tr>
</tbody>
</table>

<sConfig>
Configuration

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D00 0: Front 1: Back</td>
<td></td>
</tr>
<tr>
<td>D01 0: Upper arm 1: Lower arm</td>
<td></td>
</tr>
<tr>
<td>D02 0: Flip 1: No flip</td>
<td></td>
</tr>
<tr>
<td>D03 0: R &lt; 180 1: R &gt;= 180</td>
<td></td>
</tr>
<tr>
<td>D04 0: T &lt; 180 1: T &gt;= 180</td>
<td></td>
</tr>
<tr>
<td>D05 0: S &lt; 180 1: S &gt;= 180</td>
<td></td>
</tr>
<tr>
<td>D06 - D15 Reserved by manufacturer</td>
<td></td>
</tr>
</tbody>
</table>
<sToolNo>
Tool number (0 to 63)

<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>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>

11 Motion Monitor/Control API

mpMOVJ

[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>Home position is not registered</td>
</tr>
<tr>
<td>0x3050</td>
<td>Out of range (ABSO data)</td>
</tr>
<tr>
<td>0x3061</td>
<td>Each axis operation range limit function is temporarily disable¹</td>
</tr>
<tr>
<td>0x3062</td>
<td>Robot operation range limit function is temporarily disable¹</td>
</tr>
<tr>
<td>0x3063</td>
<td>Tool angle monitoring function is temporarily disable¹</td>
</tr>
<tr>
<td>0x3064</td>
<td>Tool switch monitoring function is temporarily disable¹</td>
</tr>
<tr>
<td>0x3065</td>
<td>External axis coasting value setting is not done¹</td>
</tr>
<tr>
<td>0x3350</td>
<td>User coordinate is not registered</td>
</tr>
<tr>
<td>0x3370</td>
<td>Wrong control group</td>
</tr>
</tbody>
</table>

1 Only when the safety function (option) is valid

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(), mpManualMOV() 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(), mpManualMOV()
Moves the manipulator to the specified position with linear motion.

### 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

<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>R3 (Robot 3)</td>
</tr>
<tr>
<td>3</td>
<td>R4 (Robot 4)</td>
</tr>
<tr>
<td>4</td>
<td>R5 (Robot 5)</td>
</tr>
<tr>
<td>5</td>
<td>R6 (Robot 6)</td>
</tr>
<tr>
<td>6</td>
<td>R7 (Robot 7)</td>
</tr>
<tr>
<td>7</td>
<td>R8 (Robot 8)</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>B3 (Base 3)</td>
</tr>
<tr>
<td>11</td>
<td>B4 (Base 4)</td>
</tr>
<tr>
<td>12</td>
<td>B5 (Base 5)</td>
</tr>
<tr>
<td>13</td>
<td>B6 (Base 6)</td>
</tr>
<tr>
<td>14</td>
<td>B7 (Base 7)</td>
</tr>
</tbody>
</table>
### mpMOVL

#### <lSpeed>
Motion speed
- 0.1 to S1C*G17 (linear speed for registration) [mm/s]
- 0.1 to S1C*G25 (position angle speed) [deg/s]
For more information on each parameter, refer to “Chap. 8.2.0.4 Linear Speed for Registration” and “Chap. 8.2.0.5 Position Angle Speed” in “YRC1000micro OPERATOR’S MANUAL (RE-CSO-A058)”.

#### <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>

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>B8 (Base 8)</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>
<tr>
<td>19</td>
<td>S4 (Station 4)</td>
</tr>
<tr>
<td>20</td>
<td>S5 (Station 5)</td>
</tr>
<tr>
<td>21</td>
<td>S6 (Station 6)</td>
</tr>
<tr>
<td>22</td>
<td>S7 (Station 7)</td>
</tr>
<tr>
<td>23</td>
<td>S8 (Station 8)</td>
</tr>
<tr>
<td>24</td>
<td>S9 (Station 9)</td>
</tr>
<tr>
<td>25</td>
<td>S10 (Station 10)</td>
</tr>
<tr>
<td>26</td>
<td>S11 (Station 11)</td>
</tr>
<tr>
<td>27</td>
<td>S12 (Station 12)</td>
</tr>
<tr>
<td>28</td>
<td>S13 (Station 13)</td>
</tr>
<tr>
<td>29</td>
<td>S14 (Station 14)</td>
</tr>
<tr>
<td>30</td>
<td>S15 (Station 15)</td>
</tr>
<tr>
<td>31</td>
<td>S16 (Station 16)</td>
</tr>
<tr>
<td>32</td>
<td>S17 (Station 17)</td>
</tr>
<tr>
<td>33</td>
<td>S18 (Station 18)</td>
</tr>
<tr>
<td>34</td>
<td>S19 (Station 19)</td>
</tr>
<tr>
<td>35</td>
<td>S20 (Station 20)</td>
</tr>
<tr>
<td>36</td>
<td>S21 (Station 21)</td>
</tr>
<tr>
<td>37</td>
<td>S22 (Station 22)</td>
</tr>
<tr>
<td>38</td>
<td>S23 (Station 23)</td>
</tr>
<tr>
<td>39</td>
<td>S24 (Station 24)</td>
</tr>
</tbody>
</table>
### 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, ... 64</td>
<td>User coordinate 1, 2, ... 63</td>
</tr>
</tbody>
</table>
11 Motion Monitor/Control API

**mpMOVL**

### <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 to 63)**

### <lPos[MAX_NO_OF AXES]>

<< Robot axis >>

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPos[0]</td>
<td>X-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>IPos[1]</td>
<td>Y-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>IPos[2]</td>
<td>Z-axis coordinate (unit: micron)</td>
</tr>
<tr>
<td>IPos[3]</td>
<td>Wrist angle Rx (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>IPos[4]</td>
<td>Wrist angle Ry (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>IPos[5]</td>
<td>Wrist angle Rz (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>IPos[6]</td>
<td>Elbow angle E (unit: 0.0001 deg)</td>
</tr>
<tr>
<td>IPos[7]</td>
<td>Reserved</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>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**
  
  [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>0x0210</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>Home position is not registered</td>
</tr>
<tr>
<td>0x3050</td>
<td>Out of range (ABSO data)</td>
</tr>
<tr>
<td>0x3061</td>
<td>Each axis operation range limit function is temporarily disable&lt;sup&gt;1)&lt;/sup&gt;</td>
</tr>
<tr>
<td>0x3062</td>
<td>Robot operation range limit function is temporarily disable&lt;sup&gt;1)&lt;/sup&gt;</td>
</tr>
<tr>
<td>0x3063</td>
<td>Tool angle monitoring function is temporarily disable&lt;sup&gt;1)&lt;/sup&gt;</td>
</tr>
<tr>
<td>0x3064</td>
<td>Tool switch monitoring function is temporarily disable&lt;sup&gt;1)&lt;/sup&gt;</td>
</tr>
<tr>
<td>0x3065</td>
<td>External axis coasting value setting is not done&lt;sup&gt;1)&lt;/sup&gt;</td>
</tr>
<tr>
<td>0x3070</td>
<td>User coordinate is not registered</td>
</tr>
<tr>
<td>0x3370</td>
<td>Wrong control group</td>
</tr>
</tbody>
</table>

<sup>1</sup> Only when the safety function (option) is valid

### Return Value

- **0** API execution processor received the command.
  
  - Return value = 0 and err_no = 0  Normal end
  - Return value = 0 and err_no f = 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(), mpManualMOV() 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()
- mpManualMOV()
mpPulseMOVJ

Moves the manipulator to the specified pulse position with joint motion.

**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

- **MP_PMOVJ_SEND_DATA**

  [in] The data structure which transmits the PMOVJ information

  Syntax: 
  ```
  typedef struct
  {
    ULONG    sCtrlGrp;
    LONG     lSpeed;
    SHORT    sToolNo;
    CHAR      reserved 0[2];
    LONG     lPos[MAX_NO_OF_AXES];
    CHAR      reserved 1[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>R3 (Robot 3)</td>
</tr>
<tr>
<td>3</td>
<td>R4 (Robot 4)</td>
</tr>
<tr>
<td>4</td>
<td>R5 (Robot 5)</td>
</tr>
<tr>
<td>5</td>
<td>R6 (Robot 6)</td>
</tr>
<tr>
<td>6</td>
<td>R7 (Robot 7)</td>
</tr>
<tr>
<td>7</td>
<td>R8 (Robot 8)</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>B3 (Base 3)</td>
</tr>
<tr>
<td>11</td>
<td>B4 (Base 4)</td>
</tr>
<tr>
<td>12</td>
<td>B5 (Base 5)</td>
</tr>
<tr>
<td>13</td>
<td>B6 (Base 6)</td>
</tr>
<tr>
<td>14</td>
<td>B7 (Base 7)</td>
</tr>
<tr>
<td>15</td>
<td>B8 (Base 8)</td>
</tr>
</tbody>
</table>
11 Motion Monitor/Control API
mpPulseMOVJ

- **<lSpeed>**
  Motion speed (1 - 10000 representing 0.01 to 100.0 %)

- **<sToolNo>**
  Tool number (0 to 63)

- **<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>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>
<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>Not in use</td>
</tr>
<tr>
<td>lPos[7]</td>
<td>Not in use</td>
</tr>
<tr>
<td>lPos[8]</td>
<td>1st external axis pulse value</td>
</tr>
<tr>
<td>lPos[9]</td>
<td>2nd external axis pulse value</td>
</tr>
<tr>
<td>lPos[10]</td>
<td>3rd external axis pulse value</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

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>Home position is not registered</td>
</tr>
<tr>
<td>0x3050</td>
<td>Out of range (ABSO data)</td>
</tr>
</tbody>
</table>
| 0x3061 | Each axis operation range limit function is temporarily disable | 1
| 0x3062 | Robot operation range limit function is temporarily disable | 1
| 0x3063 | Tool angle monitoring function is temporarily disable | 1
| 0x3064 | Tool switch monitoring function is temporarily disable | 1
| 0x3065 | External axis coasting value setting is not done | 1
| 0x3350 | User coordinate is not registered               |
| 0x3370 | Wrong control group                             |

1 Only when the safety function (option) is valid
11 Motion Monitor/Control API

mpPulseMOVJ

## 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(), mpManualMOV() 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.

### 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(), mpManualMOV()
mpPulseMOVL

Moves the manipulator to the specified pulse position with linear motion.

- **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:  
      ```c
      <sCtrlGrp>
        Control group
      ```

      | Value | Description          |
      |-------|----------------------|
      | 0     | R1 (Robot 1)         |
      | 1     | R2 (Robot 2)         |
      | 2     | R3 (Robot 3)         |
      | 3     | R4 (Robot 4)         |
      | 4     | R5 (Robot 5)         |
      | 5     | R6 (Robot 6)         |
      | 6     | R7 (Robot 7)         |
      | 7     | R8 (Robot 8)         |
      | 8     | B1 (Base 1)          |
      | 9     | B2 (Base 2)          |
      | 10    | B3 (Base 3)          |
      | 11    | B4 (Base 4)          |
      | 12    | B5 (Base 5)          |
      | 13    | B6 (Base 6)          |
      | 14    | B7 (Base 7)          |
      | 15    | B8 (Base 8)          |
### Motion Monitor/Control API

#### mpPulseMOVL

<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>
<tr>
<td>19</td>
<td>S4 (Station 4)</td>
</tr>
<tr>
<td>20</td>
<td>S5 (Station 5)</td>
</tr>
<tr>
<td>21</td>
<td>S6 (Station 6)</td>
</tr>
<tr>
<td>22</td>
<td>S7 (Station 7)</td>
</tr>
<tr>
<td>23</td>
<td>S8 (Station 8)</td>
</tr>
<tr>
<td>24</td>
<td>S9 (Station 9)</td>
</tr>
<tr>
<td>25</td>
<td>S10 (Station 10)</td>
</tr>
<tr>
<td>26</td>
<td>S11 (Station 11)</td>
</tr>
<tr>
<td>27</td>
<td>S12 (Station 12)</td>
</tr>
<tr>
<td>28</td>
<td>S13 (Station 13)</td>
</tr>
<tr>
<td>29</td>
<td>S14 (Station 14)</td>
</tr>
<tr>
<td>30</td>
<td>S15 (Station 15)</td>
</tr>
<tr>
<td>31</td>
<td>S16 (Station 16)</td>
</tr>
<tr>
<td>32</td>
<td>S17 (Station 17)</td>
</tr>
<tr>
<td>33</td>
<td>S18 (Station 18)</td>
</tr>
<tr>
<td>34</td>
<td>S19 (Station 19)</td>
</tr>
<tr>
<td>35</td>
<td>S20 (Station 20)</td>
</tr>
<tr>
<td>36</td>
<td>S21 (Station 21)</td>
</tr>
<tr>
<td>37</td>
<td>S22 (Station 22)</td>
</tr>
<tr>
<td>38</td>
<td>S23 (Station 23)</td>
</tr>
<tr>
<td>39</td>
<td>S24 (Station 24)</td>
</tr>
</tbody>
</table>

#### <Ispeed>

Motion speed

- 0.1 to $S1C^*G17$ (linear speed for registration) [mm/s]
- 0.1 to $S1C^*G25$ (position angle speed) [deg/s]

For more information on each parameter, refer to "Chap. 8.2.0.4 Linear Speed for Registration" and "Chap. 8.2.0.5 Position Angle Speed" in "YRC1000micro OPERATOR'S MANUAL (RE-CSO-A058)".

#### <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>

#### <sToolNo>

Tool number (0 to 63)
<< lPos[MAX_NO_OF_AXES] >>

<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 (T) pulse value</td>
</tr>
<tr>
<td>lPos[5]</td>
<td>6th axis (E) pulse value</td>
</tr>
<tr>
<td>lPos[6]</td>
<td>7th axis (B) pulse value</td>
</tr>
<tr>
<td>lPos[7]</td>
<td>8th 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>


<< 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>Not in use</td>
</tr>
<tr>
<td>lPos[7]</td>
<td>Not in use</td>
</tr>
<tr>
<td>lPos[8]</td>
<td>1st external axis pulse value</td>
</tr>
<tr>
<td>lPos[9]</td>
<td>2nd external axis pulse value</td>
</tr>
<tr>
<td>lPos[10]</td>
<td>3rd external axis pulse value</td>
</tr>
</tbody>
</table>

* Set “0” for data lPos[0] to lPos[5].
[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>Home position is not registered</td>
</tr>
<tr>
<td>0x3050</td>
<td>Out of range (ABSO data)</td>
</tr>
</tbody>
</table>
| 0x3061     | Each axis operation range limit function is temporarily disable
| 0x3062     | Robot operation range limit function is temporarily disable
| 0x3063     | Tool angle monitoring function is temporarily disable
| 0x3064     | Tool switch monitoring function is temporarily disable
| 0x3065     | External axis coasting value setting is not done
| 0x3350     | User coordinate is not registered                |
| 0x3370     | Wrong control group                              |

1 Only when the safety function (option) is valid

**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(), mpManualMOV() 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(), mpManualMOV()
mpManualMOV

Moves the manipulator to the specified target position in the teach mode.

**WARNING**

- mpManualMOV() enables the jog operation by the external device, instead of the programming pendant.
- DO NOT use this API for the controller to which the programming pendant is connected.
- By the operation of this API, the manipulator may move without operator’s intention and this may result in injury.

**Syntax**

```c
LONG  mpManualMOV ( 
    MP_POS_DATA* src_p,   /* Target position */
    SHORT Interpol[MP_GRP_NUM], /* Interpolation method */
    SHORT sSpeedLvl,   /* Speed level */
    MP_STD_RSP_DATA* rData); 
```

**Parameter**

*src_p* [in] The pointer to MP_POS_DATA

- **MP_POS_DATA**
  
  Syntax:
  
  ```c
typedef struct {
    CTRLG_T ctrl_grp;    /* Control group to be controlled*/
    CTRLG_T m_ctrl_grp;  /* Not used in this API */
    CTRLG_T s_ctrl_grp;  /* Not used in this API */
    MP_GRP_POS_INFO grp_pos_info[MP_GRP_NUM];
} MP_POS_DATA;
```

- **MP_GRP_POS_INFO**
  
  ```c
typedef struct {
    MP_POS_TAG pos_tag;  /* Correction amount data*/
    long pos[MP_GRP_AXES_NUM];
} MP_GRP_POS_INFO;
```

- **MP_POS_TAG**
  
  ```c
typedef struct {
    UCHAR data[8];     /* Data attribution, detail is as follows */
    CTRLG_T ei_ctrl_grp;    /* Not used in this API */
} MP_POS_TAG;
```

MP_GRP_AXES_NUM: Maximum axis number of the control group: 8
MP_GRP_NUM: Number of the control group: 32
MP_POS_DATA.ctrl_grp
Specify target control groups in bits.
Retrieve the control group number of the target group by
mpCtrlGrpId2GrpNo().
Turn ON the bit which corresponds to the retrieved group number.

MP_POS_DATA.grp_pos_info[*]
Specify the data related to the target position of each group.
As the array number, specify the control group number retrieved by
mpCtrlGrpId2GrpNo().
(e.g.) In the case of a system “R1+B1+S1”: the group number retrieved by
mpCtrlGrpId2GrpNo() is “R1=0, B1=1, S1=2”.

To operate R1 and B1,
MP_POS_DATA.ctrl_grp=0x03(0000_0011)
Specify the R1 target position to MP_POS_DATA.grp_pos_info[0]
Specify the B1 target position to MP_POS_DATA.grp_pos_info[1]

To operate S1,
MP_POS_DATA.ctrl_grp=0x04(0000_0100)
Specify the S1 target position to MP_POS_DATA.grp_pos_info[2]

MP_GRP_POS_INFO[*].MP_POS_TAG.data[0]: axis configuration data
Specify the axis configuration of the target control group as bits.
When the standard type manipulator composed of 6-axes (S, L, U, R,
B, T-axes) is used, the axis configuration data is
MP_POS_TAG.data[0]=0x3f(0011_1111).

MP_GRP_POS_INFO[*].MP_POS_TAG.data[2]: Tool file number
Specify the tool file number (0~63) of the manipulator position.

MP_GRP_POS_INFO[*].MP_POS_TAG.data[3]: Data type
Specify the data type of MP_GRP_POS_INFO.pos

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_PULSE_DTYPE</td>
<td>Pulse data</td>
</tr>
<tr>
<td>MP_ANGLE_DTYPE</td>
<td>Angle data</td>
</tr>
</tbody>
</table>
The valid data type of the base axis to specify is only "pulse data" and "base coordinate system data". For the station axis, only "pulse data" is valid. If other axes are specified, error #3620 (Error type is wrong) occurs.

```
MP_GRP_POS_INFO[*].MP_POS_TAG.data[4]: User coordinate number (1~63)
Specify the user coordinate number in case of the data type is the user coordinate number.
MP_GRP_POS_INFO[*].pos: Target position data
```

<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 angle [0.0001deg]</td>
<td>fig_ctrl (specify type)</td>
</tr>
</tbody>
</table>

1. As for the cartesian type, each value (e.g. x, y, z) and the type can be specified by casting MP_GRP_POS_INFO.pos with the following structure.

```c
typedef struct {
    LONG     x;
    LONG     y;
    LONG     z;
    LONG     rx;
    LONG     ry;
    LONG     rz;
    LONG     exp;
    UCHAR    reserved[3];
    UCHAR    fig_ctrl;    /* type specification */
} MP_F_COORD;
```
The position type variable data can be retrieved by `mpGetPosVarData()`. If the variable is a cartesian value, its figure specification can be also retrieved.

Set the state of D8~D13 (rData[0], retrieved by `mpGetPosVarData()`) to the fig_ctrl (the aforesaid MP_F_COORD type).

In addition, whether controlling the operation by referring to the given “form specification” or operating the manipulator by giving priority to the current figure is decided according to “S2C430: Relative job operation method specification”.

S2C430: Specify the relative job operation method
- 0: Previous step regarded (give priority to the current manipulator figure)
- 1: Form regarded (give priority to the given figure specification)

**[sInterpol[*]]**

[in] Specify the interpolation method

Specify the interpolation method of a manipulator to move to the target position. The specification of the “linear interpolation” to the base axis and the station axis is invalid.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_JOINT_INTERPOLATE</td>
<td>Link interpolation</td>
</tr>
<tr>
<td>MP_LINEAR_INTERPOLATE</td>
<td>Linear interpolation</td>
</tr>
</tbody>
</table>

**[sSpeedLv]**

[in] Specify the speed level

Specify the operation speed level which corresponds to the controller’s manual speed level.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_MANUALSPD_LOW</td>
<td>Low speed</td>
</tr>
<tr>
<td>MP_MANUALSPD_MID</td>
<td>Middle speed</td>
</tr>
<tr>
<td>MP_MANUALSPD_HIGH</td>
<td>High speed</td>
</tr>
<tr>
<td>MP_MANUALSPD_MAX</td>
<td>Maximum speed</td>
</tr>
</tbody>
</table>

**[rData]**

[out] The pointer to the data structure which receives the standard data
- `MP_STD_RSP_DATA`
[out] The data structure which receives the standard 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>0x3350</td>
<td>User coordinate is not registered</td>
</tr>
<tr>
<td>0x3360</td>
<td>Error in user coordinate file</td>
</tr>
<tr>
<td>0x3370</td>
<td>Wrong control group</td>
</tr>
<tr>
<td>0x3600</td>
<td>User coordinate is out of range.</td>
</tr>
<tr>
<td>0x3610</td>
<td>Error in axis construction data</td>
</tr>
<tr>
<td>0x3620</td>
<td>Error in specifying tool file number</td>
</tr>
<tr>
<td>0x3630</td>
<td>Error in data type</td>
</tr>
<tr>
<td>0x3640</td>
<td>Error in interpolation method</td>
</tr>
<tr>
<td>0x3650</td>
<td>Error in speed level specification</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.
• By this API, only one group is available to be operated. (e.g. The operation of only R1, B1, or S1 is possible. The simultaneous operation of R1 and S1 is impossible.) However, a simultaneous operation of a pair manipulators and a base axis is possible. (e.g. It is possible to set the target position of R1 and B1, and operate them at the same time.)

• Two or more move instructions such as mpIMOV(), mpMOVJ(), mpMOVL(), mpPulseMOVJ(), and mpPulseMOVL(), mpManualMOV() 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(), mpManualMOV()
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>
12 Sensor Control API

mpReceiveSkillCommand

[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>1 SKILLSEND</td>
</tr>
<tr>
<td>MP_SKILL_END</td>
<td>MP_SKILL_SEND</td>
<td>2 SKILL command force-quit</td>
</tr>
</tbody>
</table>

SYS2MP_SENS_MSG Structure

- main_comm

  (1) MP_SKILL_COMM
  This is sent when the SKILLSND command is executed or force-quit.
  Refer to sub_comm 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_comm

  (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 cmd. 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.

  - “0” is set when SKILLSND is executed at the JOB of R1 and sub_comm is MP_SKILL_SEND. And, “1” is set when it is executed at the JOB of R2.

  However, “0” is set when sub_comm is MP_SKILL_END.

- **cmd**
The pointer of the command (character string) specified by the SKILLSND command is stored.

  - “NULL” is set when sub_comm is MP_SKILL_END.

- **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>
SKILLSND instruction, as a control group, can be executed only by the JOB to which a manipulator or an external axis is defined. In case executing SKILLSND by a JOB to which a robot is not defined (without a control group), an alarm “4475: CANNOT EXECUTE JOB(NO ROBOT)” occurs. Also, when it is executed by a concurrent JOB, “4524:CANNOT EXECUTE INST(COUCUR JOB)” alarm occurs.

The communication area with the MotoPolus application which sends a command from SKILLSND is automatically selected to MP_SL_ID1 to the Robot 1 JOB and MP_SL_ID2 to the Robot 2 JOB.

For the coordinated JOB, the communication area is determined by the slave side manipulator.

- JOB: R1+R2:R1(R1:master):
  - MP_SL_ID2 is selected because R2 is the slave.
- JOB: R1+R2:R2(R2:master):
  - MP_SL_ID1 is selected because R1 is the slave.

For the exclusive job to the external axis, the communication area is not automatically selected. SLn tag should be put to SKILLSEND, and this tag selects the communication area.

- SKILLSEND SL5 "command": MP_SL_ID1 is selected.
- SKILLSEND SL6 "command": MP_SL_ID2 is selected.
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

```c
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;
```
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)
{
    SYS2MPSENS_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
\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;
    }
}
```

```c
// 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.

  - d31 to d16
    - reserved (always zero)

  - d0: Master Task
  - d1: Sub 1 Task
  - ... (not shown)
  - d15: Sub 15 Task

Return Value

- 0: Normal end
- ERROR: Error
mpMeiGetInterpolation

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>...</td>
<td>...</td>
</tr>
<tr>
<td>15</td>
<td>Sub task 15</td>
</tr>
</tbody>
</table>

- Return Value

One of the following values

<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>
<tr>
<td>-1</td>
<td>Error</td>
</tr>
</tbody>
</table>
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  |
    | ...   | ...         |
    | 15    | Sub task 15 |
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
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
    typedef struct {
        CTRLG_T ctrl_grp; /* Target control group for increment value move */
        CTRLG_T m_ctrl_grp; /* It's not used in this API(Used in mpMeiIncrementMove*/
        CTRLG_T s_ctrl_grp; /* It's not used in this API(Used in mpMeiIncrementMove*/
        MP_GRP_POS_INFO grp_pos_info[MP_GRP_NUM];
    } MP_POS_DATA;
typedef struct {
    MP_POS_TAG pos_tag;
    long pos[MP_GRP_AXES_NUM]; /* Details of data attribute are shown below */
} MP_GRP_POS_INFO;

typedef struct {
    UCHAR data[8]; /* Correction amount data */
    CTRLG_T ei_ctrl_grp; /* For EIMOV robot control group specification */
} MP_POS_TAG;

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

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_FRBOT_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 62: 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>

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.

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;
```
mpMeiPutCorrPath

  // Corrects in base coordinates

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);  // 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()`.

```
```
12  Sensor Control API
mpMeiputForcePathEnd

- 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(1000~15000) */
  )

  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 10.00% to 150.00% (minimum resolution: 1/100 [%], valid integer value: 1,000 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>
12 Sensor Control API
mpMeiPutSpdOverride

[ctrl_grp]
Control group
Specify the target group in bits.
Retrieve the control group number of the target group by
mpCtrlGrpld2GrpNo().
Specify OFF (all bits turned OFF) to make all the speed override amount
from MotoPlus invalid.

[src_p[MP_GRP_NUM]]
Speed override amount (1,000 to 15,000, unit: 1/100 [%])

■ 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>
mpMeiIncrementMove

Sets an increment value move per each interpolation cycle to the manipulator which is executing WAIT instruction. The coordinated and synchronized operation functions of the salve side over the master side control group operation is equipped.

- **Syntax**

```c
int mpMeiIncrementMove
(
    int sl_id,  /* Specifies the communication area with the system */
    MP_POS_DATA *src_p  /* Pointer to the structure that sets the data for path correction*/
)
```

- **Description**

This routine sets the increment move value to src_p->ctrl_grp indicating control group manipulators. This API adds the specified increment move value to the manipulator’s present position. By this operation, the manipulator moves for increment value move like JOG operation. To execute this operation, SKILLSND instruction had been executed and WAIT instruction is being executed by the JOB execution. In case JOB execution is stopped by the HOLD operation or change of the mode, this correcting operation also stops. And then, the increment value move is resumed when the JOB is re-stared.

SKILLSND instruction should be executed again in case the cursor of the JOB is moved or a JOB is newly selected.

In the following JOB example, by the execution of “INCMOVE”, 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 “mpMeiIncrementMove()”. 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 mpMeiIncrementMove(). (mpMeiIncrementMove() API returns as E_MP_MOVE_BUSY (-9) to its return value.)

For example, when setting 4msec as its interpolation cycle and gives 1mm per interpolation cycle in Y-direction, the manipulator moves 250mm in 1 second.

```c
<JOB example>
:

MOVL  C00000  V=100.0
SKILLSND  “INCMOVE”
WAIT  IN#(1) = ON  // Increment value move is possible
        only while this WAIT instruction is being executed.
MOVL  C00001  V=100.0
```

12 Sensor Control API
mpMeiIncrementMove

HW1484530 320/384
This specification is available not only for the manipulator but also to the external axis (station axis). When executing the increment value move using this API from an exclusive JOB to the external axis, specification of communication area indicating tag “SLn (SL5 or SL6)” is required. This is because the external axis has no specified communication area whereas Robot 1 is specified with communication area 1 and Robot 2 is specified with communication area 2 naturally.

As for the tag which indicates the communication area to SKILLSND instruction, SL5 indicates the communication area 1 and SL6 indicates the communication area 2.

In the system with one manipulator and one external axis, the exclusive JOB to the external axis specifies SL6 as its communication area because, to avoid the area to overlap, the communication area 1 is used when SKILLSND instruction is executed at Robot 1 side.

```<JOB example>
:
MOV L  EC00000  V=100.0
SKILLSND  SL6  “INCMOVE”
WAIT IN#(1) = ON           // Increment value move is possible
    only while this WAIT instruction is being executed.
```

### Parameter [sl_id]

Specifies the communication area with the system. The following values can be specified.

- Specify MP_SL_ID1 to the Robot 1 (R1) and MP_SL_ID2 to the Robot 1 (R2).
- When it is a coordinated JOB, specification of the area depends on the slave side manipulator.
- When it is an exclusive JOB to the external axis, specify MP_SL_ID1 when SL5 is specified by SKILLSND in the JOB and specify MP_SL_ID2 when SL6 is specified.

<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>
Increased move value information

typedef struct {
CTRLG_T ctrl_grp;  /* Object control group for 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) working operation */
MP_GRP_POS_INFO grp_pos_info[MP_GRP_NUM];
} MP_POS_DATA;

• ctrl_grp
  Control group of the object operation

• m_ctrl_grp
  Master side control group when the synchronized operation is executed. It should be the same control group as (ctrlGRP). Set “0” when the synchronized operation is not executed.

• s_ctrl_grp
  Slave side control group which is synchronized with the master side control group when the synchronized operation is executed. Set “0” when the synchronized operation is not executed.

< Errors>

Followings are the functional errors when setting the control group information.

1.
  – When the specified m_ctrl_grp is not “0” and the specified m_ctrl_grp is not found (does not exist) in the system configuration.
  – When s_ctrl_grp is not “0” and the specified m_ctrl_grp is “0”.
  – When more than one control groups are specified.
  – When more than one control groups are specified as a master.
  → E_MP_M_CTRL_GRP (-12)
    Master side control group specifying error

2.
  – When the specified m_ctrl_grp is not “0” and the specified s_ctrl_grp is not found (does not exist or “0” is set) in the system configuration.
  → E_MP_S_CTRL_GRP (-13)
    Slave side control group specifying error

3. When overlapped control groups exist in the specified m_ctrl_grp and s_ctrl_grp.
  → E_MP_SYNC_MOVE_CTRL_GRP (-11)
    Control group specifying error in the coordinated operation
12 Sensor Control API
mpMeIncrementMove

4. When “0” is set to either specified m_ctrl_grp or s_ctrl_grp.
   →E_MP_M_CTRL_GRP (-12)
   Master side control group specifying error
   →E_MP_S_CTRL_GRP (-13)
   Slave side control group specifying error

5. When R2 is operated to start in the MotoPlus application in which
   MP_SL_ID1 received the SKILLSND instruction executed from R1
   JOB but the JOB of R2 is already executed.
   →E_MP_USING_CTRL_GRP (-15)
   A specified control group JOB is being executed.

R1 and R2 system control group setting and behavior
<setting example>

<table>
<thead>
<tr>
<th>m_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>
</tbody>
</table>
| 1          | 2          | 1          | Error
   E_MP_SYNC_MOVE_CTRL_GRP (-11)
   Control group specifying error in the coordinated operation |
| 1          | 0          | 1          | Error
   E_MP_M_CTRL_GRP (-12)
   Master side control group specifying error |
| 1          | 1          | 0          | Error
   E_MP_S_CTRL_GRP (-13)
   Slave side control group specifying error |
| 1          | 1          | 1          | Error
   E_MP_SYNC_MOVE_CTRL_GRP (-11)
   Control group specifying error in the coordinated operation |
| 1          | 0          | 2          | Error
   E_MP_M_CTRL_GRP (-12)
   Master side control group specifying error |
| 1          | 2          | 0          | Error
   E_MP_S_CTRL_GRP (-13)
   Slave side control group specifying error |
| 2          | 0          | 0          | R2 independent operation |
| 2          | 1          | 2          | Error
   E_MP_SYNC_MOVE_CTRL_GRP (-11)
   Control group specifying error in the coordinated operation |
| 2          | 2          | 1          | Operate R1 and R2 synchronizes |
| 2          | 0          | 1          | Error
   E_MP_M_CTRL_GRP (-12)
   Master side control group specifying error |
| 2          | 1          | 0          | Error
   E_MP_S_CTRL_GRP (-13)
   Slave side control group specifying error |
| 2          | 0          | 2          | Error
   E_MP_M_CTRL_GRP (-12)
   Master side control group specifying error |
typedef struct {
    MP_POS_TAG pos_tag;
    long pos[MP_GRP_AXES_NUM];
    /* For the property and details of the data, see below*/
} MP_GRP_POS_INFO;

typedef struct {
    UCHAR data[8]; /* Correction amount data */
    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 (32)

MP_POS_TAG.data[0]: Information of axis configuration

```
  d7  d6  d5  d4  d3  d2  d1  d0
  +-------------------
 |                  |
 |                  |
 |                  |
 | 1st Axis         |
 |                  |
 |                  |
 |                  |
 | 8th Axis         |
```

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_INC_PULSE_DTYPE</td>
<td>Pulse data (correction of the present position (INC operation))</td>
</tr>
<tr>
<td>MP_INC_ANGLE_DTYPE</td>
<td>Angle data (correction of the present position (INC operation))</td>
</tr>
<tr>
<td>MP_INC_BF_DTYPE</td>
<td>Base coordinate system data (correction of the present position (INC operation))</td>
</tr>
<tr>
<td>MP_INC_RF_DTYPE</td>
<td>Robot coordinate system data (correction of the present position (INC operation))</td>
</tr>
<tr>
<td>MP_INC_TF_DTYPE</td>
<td>Tool coordinate system data (correction of the present position (INC operation))</td>
</tr>
<tr>
<td>MP_INC_UF_DTYPE</td>
<td>User coordinate system data (correction of the present position (INC operation))</td>
</tr>
</tbody>
</table>

MP_POS_TAG.data[4]: User coordinate number
(0 to 62: valid only when the user coordinate is specified)

MP_POS_TAG.data[1], [5] to [7]: For the future extension

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>Wrist angle Rx [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>
12 Sensor Control API

mpMeiIncrementMove

- Return Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_MP_ARG_OBJ_ID (-1)</td>
<td>sl_id error</td>
</tr>
<tr>
<td>E_MP_ACCESS_LIMIT (-2)</td>
<td>Access limited</td>
</tr>
<tr>
<td>E_MP_CTRL_GRP (-3)</td>
<td>Control group error</td>
</tr>
<tr>
<td>E_MP_AXES_CONFIG (-4)</td>
<td>Axis configuration information error</td>
</tr>
<tr>
<td>E_MP_TOOL_NO (-5)</td>
<td>Tool number error</td>
</tr>
<tr>
<td>E_MP_POS_DTYPE (-6)</td>
<td>Data type error</td>
</tr>
<tr>
<td>E_MP_USER_FRAME (-7)</td>
<td>User coordinate setting error</td>
</tr>
<tr>
<td>E_MP_JOG_BUSY (-8)</td>
<td>JOG is in operation</td>
</tr>
<tr>
<td>E_MP_MOVE_BUSY(-9)</td>
<td>Move command is being executed</td>
</tr>
<tr>
<td>E_MP_NOT_UNDER_PLAYBACK-10</td>
<td>Not being playback operation</td>
</tr>
<tr>
<td>E_MP_SYNC_MOVE_CTRL_GRP-11</td>
<td>Control group error in coordinated operation</td>
</tr>
<tr>
<td>E_MP_M_CTRL_GRP (-12)</td>
<td>Master side control group specifying error</td>
</tr>
<tr>
<td>E_MP_S_CTRL_GRP(-13)</td>
<td>Slave side control group specifying error</td>
</tr>
<tr>
<td>E_MP_INACT_SKILLCMD (-14)</td>
<td>Calling API when SKILLSND instruction is not executed</td>
</tr>
<tr>
<td>E_MP_USING_CTRL_GRP (-15)</td>
<td>Specified control group JOB is being executed</td>
</tr>
<tr>
<td>E_MP_ROB_USE_SL_ID (-16)</td>
<td>Wrong SL specification is used in SKILLSND</td>
</tr>
</tbody>
</table>

In the environment using the independent control function, when a specified control group which executes the increment value move by this API in a job local task, the control group can execute the increment value move only in the same local task until the job local task completes. When the control group tries to execute the increment value move in the different job local tasks, this API returns "-3: Control group error", and does not execute the increment value move.

For example, when R1 executes the increment value move by this API in the master local task and then R1 executes the increment value move by this API in the sub 1 local task, this API returns "-3: Control group error".

When this API is executed in the environment using the independent control function, create a job so as to make the control group and the job local task to execute the increment value move the same.

For example, when executing the increment value move using this API in the dual-robot system, create a job so as to make the control group and the job local task to execute the increment value move the same by executing the increment value move respectively by R1 to execute the WAIT command only in the sub 1 local task and by R2 in the sub 2.
Sample code

Followings are the sample code of the increment value move operation. When incomove_req is set, the manipulator moves 1mm to Y-direction per interpolation cycle by the task received by the command.

The increment value move process is FOEVER (exists in the infinite loop). And this process should be executed in coordination with the interpolation cycle, thus mpClkAnnounce(MP_INTERPOLATION_CLK) is indicated to the head of the loop.

```c
int incmove_req;
void pathCorrectionTask(void)
{

MP_POS_DATA corpath_src_p;

// Initialization of the correction data
memset(&corpath_src_p, CLEAR, sizeof(MP_POS_DATA));
corpath_src_p.ctrl_grp = 1; // Robot1
corpath_src_p.grp_pos_info[0].pos_tag.data[0] = 0x3f;
    //Specification of the available axis (for 6 axes)
    //Increment value move operation on the base coordinate system
corpath_src_p.grp_pos_info[0].pos[0] = 0; //X
corpath_src_p.grp_pos_info[0].pos[1] = 0; //Y
corpath_src_p.grp_pos_info[0].pos[2] = 0; //Z
corpath_src_p.grp_pos_info[0].pos[3] = 0; //Rx
corpath_src_p.grp_pos_info[0].pos[4] = 0; //Ry
corpath_src_p.grp_pos_info[0].pos[5] = 0; //Rz
```

NOTE

The correction data received by the system after calling of this API is valid only once and for all. This means that the calling task for this API should be periodical (For example, it starts in the interpolation cycle.). Consequently, the program structure which can periodically call this API for the increment value move is realized.
FOEVER
{
    mpClkAnnounce(MP_INTERPOLATION_CLK);
    //Execute following process per interpolation cycle

    if (incmove_req == ON )
    //Correction required? It is set by the task received by the command
    {
        //PutCorrPath (Correct 1[mm] in Y-direction per interpolation cycle
        corrpath_src_p.grp_pos_info[0].pos[1] = 1000;
        mpMeiIncrementMove (MP_SL_ID1, &corrpath_src_p);
    }
}
}
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>

: 

MOV L 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.

MOV L 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_grp
  Slave side control group for coordinated (synchronized) operation.
  Set “0” in case synchronized operation is not executed.
NOTICE

< Errors>
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.
   →E_MXRCS_M_CTRL_GRP (-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_CTRL_GRP (-12)

3. When overlapped control groups exist in the specified “m_ctrl_grp” and “s_ctrl_grp”.
   →E_MXRCS_SYNC_MOVE_CTRL_GRP (-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>0</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_CTRL_GRP(-10)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Control group specification error in the coordinated operation</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Error: E_MXRCS_M_CTRL_GRP(-11)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Master side control group specification error</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Error: E_EXRCS_S_CTRL_GRP(-12)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Slave side control group specification error</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Error: E_EXRCS_SYNCMOVE_CTRL_GRP(-10)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Control group specification error in the coordinated operation</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>Error: E_MXRCS_M_CTRL_GRP(-11)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Master side control group specification error</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>Error: E_EXRCS_S_CTRL_GRP(-12)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>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_CTRL_GRP(-10)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>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_MXRCS_M_CTRL_GRP(-11)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Master side control group specification error</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>Error: E_EXRCS_S_CTRL_GRP(-12)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Slave side control group specification error</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
<td>Error: E_MXRCS_M_CTRL_GRP(-11)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Master side control group specification error</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>Error: E_EXRCS_S_CTRL_GRP(-12)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Slave side control group specification error</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>Error: E_EXRCS_SYNC_MOVE_CTRL_GRP(-10)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Control group specification error in the coordinated operation</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>Error: E_MXRCS_M_CTRL_GRP(-11)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>real value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_INC_PULSE_DTYPE</td>
<td>0x80</td>
<td>Increment value specification by pulse data</td>
</tr>
<tr>
<td>MP_INC_ANGLE_DTYPE</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[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>
</tbody>
</table>
### 13 Robot Motion Control API

**mpExRcsIncrementMove**

<table>
<thead>
<tr>
<th>Array</th>
<th>Pulse type</th>
<th>Angle type</th>
<th>Cartesian type</th>
</tr>
</thead>
<tbody>
<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

<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 mpExRcsIncrementMove() is executing.</td>
</tr>
</tbody>
</table>

**NOTE**

If this API is executed in the state that the job which executes the increment value move is not executed, it returns “OK(0): Normal end”, and doesn’t execute the increment value move.

(e.g.) When the job execution is stopped during the manipulator’s increment value move by this API because of the emergency stop, hold ON, or mode switch, the increment value move of the manipulator also stops. All the while this API returns “OK(0): Normal end” as the return value. When the JOB is started again and executes the WAIT instruction, the manipulator starts the increment value move again while this API keeps returning “OK(0): Normal end”.
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), \texttt{mpClkAnnounce(MP\_INTERP\_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 corpath_src_p;

    //
    memset(&corpath_src_p, CLEAR, sizeof(MP\_EXPOS\_DATA));
```

**Note**

In the environment using the independent control function, when a specified control group which executes the increment value move by this API in a job local task, the control group can execute the increment value move only in the same local task until the job local task completes.

When the control group tries to execute the increment value move in the different job local tasks, this API returns "-3: Error in control group setting", and does not execute the increment value move.

For example, when R1 executes the increment value move by this API in the master local task and then R1 executes the increment value move by this API in the sub 1 local task, this API returns "-3: Error in control group setting".

When this API is executed in the environment using the independent control function, create a job so as to make the control group and the job local task to execute the increment value move the same.

For example, when executing the increment value move using this API in the dual-robot system, create a job so as to make the control group and the job local task to execute the increment value move the same by executing the increment value move respectively by R1 to execute the WAIT command only in the sub 1 local task and by R2 in the sub 2.

The correction data that the system received by calling this API is valid only once. So normally the task which calls this API needs to be periodical (For example, it starts in the interpolation cycle.). That realizes the program structure which periodically calls this API when the increment value move is needed.

**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.
corrpah_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;

    // Increment value move operation on the base coordinate system

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);
    }
}
14 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() and mpGetTorque() 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() and mpGetTorque(). Regarding the data of torque, the direction (sign) may be different from that of mpGetTorque().

  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

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>dst_vel[x][y]</code></td>
<td>The feedback speed of the y-th axis of the x-th control group. Use the value retrieved by <code>mpCtrlGrpId2GrpNo()</code> as the control group number.</td>
</tr>
</tbody>
</table>
mpSvsGetVelTrqFb

[dst_trq]

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>

<data>

- **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];`

Array Description

<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 speed. 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(). For mpGetTorque(), 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() 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 “YRC1000micro OPTIONS INSTRUCTIONS Programmer’s Manual For New Language Environment MotoPlus (HW1484526)”, 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
  - Endless function
  - Motor gun pressure instructions

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

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_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>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 “YRC1000micro OPTIONS INSTRUCTIONS Programmer’s Manual For New Language Environment MotoPlus (HW1484526)”, 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().

- **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().
mpSvsSetTrqLimit

**Axes configuration information**
Specify the target axis in bits.
Specify the control group number of the target group retrieved by `mpCtrlGrpId2GrpNo()` as the array number.

```
[axes_config[]]
```

**[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>
### mpSvsSetTrqLimit

- **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
  - **Member: `<min[]>`**
    - Torque lower limit value

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>data[x].max[y]</code></td>
<td>x: control group number, y: axis number. The torque upper limit value of the y-th axis of the x-th control group. Use the value retrieved by <code>mpCtrlGrpId2GrpNo()</code> as the control group number.</td>
</tr>
<tr>
<td><code>data[x].min[y]</code></td>
<td>x: control group number, y: axis number. The torque lower limit value of the y-th axis of the x-th control group. Use the value retrieved by <code>mpCtrlGrpId2GrpNo()</code> as the control group number.</td>
</tr>
</tbody>
</table>

**NOTE**

- 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()`. For `mpGetTorque()`, 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()` is the reverse of the direction of the speed and the torque for this API.
Return Value

0 Normal end
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_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_NON_CTRL_GRP -7</td>
<td>No group is specified.</td>
</tr>
<tr>
<td>E_SVS_CTRL_GRP_NOT_EXIST -8</td>
<td>Non-existent group is specified.</td>
</tr>
<tr>
<td>E_SVS_NOT_TRQ_LMT_AXIS -18</td>
<td>An invalid axis is specified.</td>
</tr>
<tr>
<td>E_SVS_TRQ_LMT_LARGER_THAN_ULT -20</td>
<td>A value larger than the upper limit is set to the lower limit.</td>
</tr>
<tr>
<td>E_SVS_TRQ_LMT_RANGE_MAX -21</td>
<td>The value set to the upper limit exceeds the allowable range.</td>
</tr>
<tr>
<td>E_SVS_TRQ_LMT_RANGE_MIN -22</td>
<td>The value set to the lower limit exceeds the allowable range.</td>
</tr>
<tr>
<td>E_SVS_RTP_NOT_SUPPORTED -25</td>
<td>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 “YRC1000micro OPTIONS INSTRUCTIONS Programmer’s Manual For New Language Environment MotoPlus (HW1484526)”, 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.

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().
14 Servo Monitor/Control API

mpSvsEndTrqLimit

- Return Value

  0 Normal end
  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_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>The servo-control part was not able to end the torque limit mode.</td>
</tr>
<tr>
<td>E_SVS_AX_COMM_FAILURE -13</td>
<td>Re-creation of the current position is failed.</td>
</tr>
<tr>
<td>E_SVS_TRQ_LMT_CANCEL -17</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 “YRC1000micro OPTIONS INSTRUCTIONS Programmer’s Manual For New Language Environment MotoPlus (HW1484526)”, and be sure to confirm safety before using this API.

Syntax

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
  - Endless function
  - Motor gun pressure instructions

Note that the actual torque control is started after executing this API and then mpSvsSetTrqCtrl().
14 Servo Monitor/Control API
mpSvsStartTrqCtrl

**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().

  <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

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_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>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 The servo-control part was not able to move to the torque control mode.</td>
</tr>
<tr>
<td>E_SVS_AX_TRQ_LMT_SET</td>
<td>-16</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 "YRC1000micro OPTIONS INSTRUCTIONS Programmer's Manual For New Language Environment MotoPlus (HW1484526)", 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 *arg_ctl_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>
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.</td>
</tr>
<tr>
<td>x: control group number</td>
<td>Use the value retrieved by mpCtrlGrpId2GrpNo() as the control group number.</td>
</tr>
<tr>
<td>y: axis number</td>
<td></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(). For mpGetTorque(), 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() 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>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_NON_CTRL_GRP -7</td>
<td>No group is specified.</td>
</tr>
<tr>
<td>E_SVS_CTRL_GRP_NOT_EXIST -8</td>
<td>Non-existent group is specified.</td>
</tr>
<tr>
<td>E_SVS_NOT_TRQ_CTL_AXIS -19</td>
<td>An invalid axis is specified.</td>
</tr>
<tr>
<td>E_SVS_TRQ_CTL_RANGE -23</td>
<td>The value set as the torque command value exceeds the allowable range.</td>
</tr>
<tr>
<td>E_SVS_RTP_NOT_SUPPORTED -25</td>
<td>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 “YRC1000micro OPTIONS INSTRUCTIONS Programmer’s Manual For New Language Environment MotoPlus (HW1484526)”, 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.

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 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 control is not 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 end 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_AX_TRQ_LMT_CANCEL</td>
<td>-17 The servo-control part was not able to end the torque control mode.</td>
</tr>
<tr>
<td>E_SVS_PR_MAKE_FAILURE</td>
<td>-24 Re-creation of the current position is failed.</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>

0 Normal end

Less than 0 Error

One of the following values is returned according to the error content.
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
  | OK          | Normal end |
  | ERROR       | Error      |
  |             | In the case when it was impossible to end the servo control in execution, and the like. |
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>...</td>
<td>...</td>
</tr>
<tr>
<td>31</td>
<td>Thirty-second 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>63</td>
<td>Tool number 63</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;
  } 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>
15 Kinematics API
mpConvAxesToCartPos

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>
<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],
      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>...</td>
<td>...</td>
</tr>
<tr>
<td>31</td>
<td>Thirty-second control group</td>
</tr>
</tbody>
</table>
Kinematics API

mpConvCartPosToAxes

[coord]
Pointer to the cartesian coordinate position structure (input)

- MP_COORD
  Cartesian coordinate position structure
  Syntax: typedef struct
  {
    long  x, y, z;
    long  ry, rz;
    long  ex1, ex2;
  }    MP_COORD;

[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>63</td>
<td>Tool number 63</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</td>
</tr>
<tr>
<td>D01</td>
<td>0:Upper arm</td>
</tr>
<tr>
<td>D02</td>
<td>0:Flip</td>
</tr>
<tr>
<td>D03</td>
<td>0:R&lt;180</td>
</tr>
<tr>
<td>D04</td>
<td>0:T&lt;180</td>
</tr>
<tr>
<td>D05</td>
<td>0:S&lt;180</td>
</tr>
<tr>
<td>D06 - D31</td>
<td>Reserved by manufacturer</td>
</tr>
</tbody>
</table>
15 Kinematics API

mpConvCartPosToAxes

[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
  {
    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 (-1)</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 (-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>
<tr>
<td>E_KINEMA_KINEMA_TYPE_ERR</td>
<td>Inverse kinematics wrong</td>
</tr>
<tr>
<td>E_KINEMA_INTERPOLATION_INVALID (-6)</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().

    | Value | Description               |
    |-------|---------------------------|
    | 0     | First control group       |
    | 1     | Second control group      |
    | ...   | ...                       |
    | 31    | Thirty-second control group|

  - **[pulse]**
    Pulse position
    Syntax: `#define MP_GRP_AXES_NUM (8)`

    | Value | Description                   |
    |-------|------------------------------|
    | pulse[0] | 1st axis (S) pulse value    |
    | pulse[1] | 2nd axis (L) pulse value     |
    | pulse[2] | 3rd axis (U) pulse value     |
    | pulse[3] | 4th axis (R) pulse value     |
    | pulse[4] | 5th axis (B) pulse value     |
    | pulse[5] | 6th axis (T) pulse value     |
    | pulse[6] | 7th axis (E) pulse value     |
    | pulse[7] | 8th axis (W) pulse value     |
15 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**

<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>

**NOTE**
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>...</td>
<td>...</td>
</tr>
<tr>
<td>31</td>
<td>Thirty-second 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**

<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.
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>...</td>
<td>...</td>
</tr>
<tr>
<td>31</td>
<td>Thirty-second 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>
15 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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>One of the following negative values</td>
<td>Error</td>
</tr>
</tbody>
</table>

- E_KINEMA_FAILURE (-1) Failed in conversion
- E_KINEMA_CONV_IMPOSSIBLE (-2) Conversion impossible
- E_KINEMA_GRP_OUT_RANGE (-3) Specified control group out of range

NOTE
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;

<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</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

[param_name]
Pointer to the homogeneous transformation matrix structure (input)

- MP_FRAME
  homogeneous transformation matrix structure

Syntax: typedef struct
```c
{
    double nx, ny, nz;
    double ox, oy, oz;
    double ax, ay, az;
    double ox, py, pz;
} MP_FRAME;
```
Kinematics API

mplInvFrame

[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 (-1)</td>
<td>Failed in conversion</td>
</tr>
</tbody>
</table>

0 Normal end
One of the following negative values 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
    Syntax: typedef struct
    ```c
    {
        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)
**[vector]**

Pointer to the position vector structure (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>

**[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 (-1)</td>
<td>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)

  ```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>
mpMulFrame

[frame2]
Pointer to the homogeneous transformation matrix structure (input)

[prod]
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>
mpZYXeulerToFrame

Converts an optional XYZ eulerian expression cartesian value to a homogeneous transformation matrix.

- **Syntax**

  ```c
  int mpZYXeulerToFrame(
    MP_COORD* coord,
    MP_FRAME* frame
  );
  ```

- **Parameter**

  `[coord]`

  Pointer to the cartesian coordinate position structure (input)

  - MPCOORD
    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</td>
<td>(-1) 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>
15 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;
  } 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>

- 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>
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;
    ```

- **[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>0</td>
<td>Normal end</td>
</tr>
</tbody>
</table>

One of the following negative values

<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>

Member | Description
---|---------------------------|
X      | X-component value (unit: micron)
Y      | Y-component value (unit: micron)
Z      | Z-component value (unit: micron)
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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>One of the following negative values</td>
<td>Error</td>
</tr>
</tbody>
</table>

<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>
### 16 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 LOAD ERROR</td>
<td>1</td>
<td>The number of loaded files exceeds the limit.</td>
</tr>
<tr>
<td></td>
<td></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 (MotoPlus)</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>4479</td>
<td>MOTOPLUS MM TASK WATCHDOG ERROR</td>
<td></td>
<td>The man-machine task does not run for 3 seconds or more.</td>
</tr>
</tbody>
</table>
YRC1000micro OPTIONS
INSTRUCTIONS
(API Function Specifications)

HEAD OFFICE
2-1 Kurosakishiroishi, Yahatanishi-ku, Kitakyushu 806-0004, Japan
Phone +81-93-645-7703 Fax +81-93-645-7802

YASKAWA America Inc. (Motoman Robotics Division)
100 Automation Way, Miamisburg, OH 45342, U.S.A.
Phone +1-937-847-6200 Fax +1-937-847-6277

YASKAWA Europe GmbH (Robotics Division)
Yaskawastrasse 1, 85391 Allershausen, Germany
Phone +49-8166-90-100 Fax +49-8166-90-103

YASKAWA Nordic AB
Verkstadsgatan 2, Box 504 ,SE-385 25 Torsas, Sweden
Phone +46-480-417-900 Fax +46-486-414-10

YASKAWA Electric (China) Co., Ltd.
22F, One Corporate Avenue, No.222, Hubin Road, Huangpu District, Shanghai 200021, China
Phone +86-21-5385-2200 Fax +86-21-5385-3299

YASKAWA SHOUGANG ROBOT Co. Ltd.
No7 Yongchang North Road, Beijing E&T Development Area, China 100176
Phone +86-10-6788-2858 Fax +86-10-6788-2878

YASKAWA India Private Ltd. (Robotics Division)
#426, Udyog Vihar, Phase- IV, Gurgaon, Haryana, India
Phone +91-124-475-8500 Fax +91-124-475-8542

YASKAWA Electric Korea Corporation
35F, Three IFC, 10 Gukjegeumyung-ro, Yeongdeungpo-gu, Seoul, Korea 07326
Phone +82-2-784-7844 Fax +82-2-784-8495

YASKAWA Electric Taiwan Corporation
12F, No.207, Sec. 3, Beishin Rd., Shindian District, New Taipei City 23143, Taiwan
Phone +886-2-8913-1333 Fax +886-2-8913-1513

YASKAWA Electric (Singapore) PTE Ltd.
151 Lorong Chuan, #04-02A, New Tech Park, Singapore 556741
Phone +65-6282-3003 Fax +65-6289-3003

YASKAWA Electric (Thailand) Co., Ltd.
59,1st-5th Floor, Flourish Building, Soi Ratchadapisek 18,Ratchadapisek Road,
Huaykwang, Bangkok 10310, THAILAND
Phone +66-2-017-0099 Fax +66-2-017-0199

PT. YASKAWA Electric Indonesia
Secure Building-Gedung B Lantai Dasar & Lantai 1 Jl. Raya Protokol Halim Perdanakusuma,
Jakarta 13610, Indonesia
Phone +62-21-2982-6470 Fax +62-21-2982-6741

Specifications are subject to change without notice
for ongoing product modifications and improvements.

YASKAWA

YASKAWA ELECTRIC CORPORATION

© Printed in Japan August 2018 17-07

MANUAL NO. HW1484530 ◆ 384/384