

Motion Network System

motionCAT *series*

User's Manual

Motion by Control Automation Technology

Preliminary



Hivertec, inc.
<http://www.hivertec.co.jp/>

This manual is for the motionCAT series products listed below.

Category		Product name	Model number
Master board		PCI master board	HPCI-MNT520M
		Compact PCI master board	HCPCI-MNT720M
HMG type	Motion slave	HMG 1-axis motion slave	HMG-P1
		HMG 2-axes motion slave	HMG-P2
		HMG 3-axes motion slave	HMG-P3
		HMG 4-axes motion slave	HMG-P4
		HMG 5-axes motion slave	HMG-P5
		HMG 6-axes motion slave	HMG-P6
	DIO slave	HMG DIO 32 slave	HMG-D1
		HMG DIO 64 slave	HMG-D2
		HMG DIO 96 slave	HMG-D3
		HMG DIO 128 slave	HMG-D4
	Composite slave	HMG 1-axis + DIO 32 slave	HMG-P1D1
		HMG 1-axis + DIO 64 slave	HMG-P1D2
		HMG 1-axis + DIO 96 slave	HMG-P1D3
		HMG 2-axes + DIO 32 slave	HMG-P2D1
		HMG 2-axes + DIO 64 slave	HMG-P2D2
		HMG 3-axes + DIO 32 slave	HMG-P3D1

It is prohibited to reprint or copy all or any part of this manual or programs without prior written permission.
The contents of this manual are subject to change without notice to enable improvements to be made.
Please notify your sales representative for this product if you find any problems with the contents of this manual.

Motionnet is a registered trademark of Nippon Pulse Motor Co.,Ltd. Windows Vista, Windows 2000 Professional, Windows XP Home Edition, Windows XP Professional, Visual C++, and Visual Basic are registered trademarks of Microsoft Corporation in the U.S. and other countries. Other company and product names are trademarks or registered trademarks of the respective companies.

Hivertec Inc.
Mitsuseimei Shin-Ohashi Bldg.
1-8-11 Shin-Ohashi, Koto-Ku, Tokyo 135-0007, Japan
TEL +81-3-3846-3801
FAX +81-3-3846-3773
sales@hivertec.co.jp

Preliminary January 30, 2008 released
Copyright Hivertec Inc.

Extent of Warranty

1. The motionCAT series products warranty is valid for a period of three years from the date of purchase. If a defect is acknowledged by Hivertec within the period of warranty, Hivertec will repair or replace the product upon return of the product to Hivertec.
2. Hivertec is not responsible beyond the purchase price of the product for any damages or loss of profit, direct, indirect, or secondary, caused by application, delivery, or failure of a Hivertec product either within or outside of the period of warranty.

Limitations to Liability



1. Hivertec is not responsible for any damages resulting from product installation, connections, settings, or operation that do not follow the contents of this manual.
2. The motionCAT series products use semiconductor devices manufactured for general electronics equipment, such as machine tools, instrumentation, FA devices, OA devices, and communications equipment. They are not designed, conceived, approved for, or warranted for application in devices for which faulty operation or failure will have a direct affect on human life or result in personal injury or damage to property. The safety, quality, and performance of the motionCAT series products are not guaranteed explicitly or implicitly beyond those given in this manual or related catalogs.
3. Hivertec is not responsible for any damages resulting from modifications or repairs made to the products without the approval of Hivertec either within or outside of the period of warranty.
4. The contents of this manual do not guarantee or grant rights to patents, copyright, trademark rights, or any other rights to the intellectual property of Hivertec or any third party. Hivertec is not responsible for any problems that may occur concerning the rights to intellectual property of third parties resulting from the application of information provided in this manual.

Important Safety Instructions



Thank you for choosing the motionCAT series products.

This manual contains information that is important for the safe and reliable operation of the motionCAT products. Read this section and understand the information contained before attempting to use the products.




Furthermore, save this manual and store it in an easily accessible location near the installed motionCAT series products, so that it can be referenced when necessary.

Safety Precautions	
<p>Always read this manual and any attached documents completely before attempting to use the motionCAT series products. Be sure that you understand the information provided and are using the products correctly. Do not use the products before having a complete understanding of the products, product safety information, and precautions.</p> <p>In this manual, safety precautions are classified as either Warnings or Cautions.</p>	
 Warning	Indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury.
 Caution	Indicates a potentially hazardous situation which, if not avoided, may result in minor or moderate injury, or property damage.



1. Authorized For

 Caution	
	<p>The motionCAT series products and this manual are designed for those with the following knowledge.</p> <ul style="list-style-type: none">- A basic knowledge of installing and wiring expansion boards.- A basic knowledge of electronic control devices and personal computers.








2. Applicable Bus

 Warning	
	HPCI-MNT520M is a board applicable to PCI Local Bus Specification Rev.2.2. Do not use the board in an environment that does not support PCI Local Bus Specification Rev.2.2.
	HCPCI-MNT720M is a board applicable to CompactPCI Specification Revision 2.1. Do not use the board in an environment that does not support CompactPCI Specification Revision 2.1.

3. Environmental Conditions

 Warning	
	Store and use the motionCAT series products only under the following environmental conditions. <ul style="list-style-type: none">- Ambient operating temperature: 0 to 50 °C- Ambient operating humidity: 20% to 85% (with no condensation)- Ambient storage temperature: -15 to 75 °C- Ambient storage humidity: 10% to 90% (with no condensation)- Atmosphere: No corrosive gas, inflammable gas, oil mist, or dust- Altitude: 3,000 m above sea level max. (Lower upper level of temperature ranges by 2 °C for every 300 m.)

4. Compatible Cables and Communication System

 Warning	
	Do not connect a communication cable to a product incompatible with Motionnet.
	Use shielded LAN cables, CAT5e or CAT6, as communication cables. The use of a cable other than these may cause an operating error.
	The cable length between slaves or between the master and slave must be 60 cm or more. A too short cable may cause an operating error.
	The total transmission distance must be within 50 meters. A too long transmission distance may cause an operating error.
	Limit the number of modules per line to 32 or less. Connecting more than 32 modules may cause an operating error.
	Do not use duplicate module IDs (addresses) for modules on the same line. Duplicate IDs may cause an operating error or a failure.

5. Transportation and Installation



Warning



Touch a metal object to discharge static electricity from your body before touching a motionCAT series product. Static electricity may cause a failure of the product.



Do not wrap a motionCAT series product in any wrapping material that easily carries a static charge, such as bubble wrap. Static electricity may cause a failure of the product.



Do not touch the edge connector on a motionCAT series product. Contamination on the edge connector may cause an operating error.



Do not place any heavy object on a motionCAT series product. A heavy object may damage components on the product, possibly causing a failure of the product.



Set the jumpers (switches) on a motionCAT series product before installing the product in the computer. If the jumpers (switches) are set with the power supply turned ON, the settings may not be recognized correctly, possibly causing an operating error.



Set the jumpers (switches) on a motionCAT series product correctly. Incorrect settings may cause an operating error.



Always turn OFF the power supply to the computer and disconnect the power supply cable before installing a motionCAT series product. Installing the product without removing the power supply cable may cause a failure, or the device may exhibit an unexpected operation.



HCPCI-MNT720M does not provide hot-swap functionality.

Always turn OFF the power supply to the computer and disconnect the power supply cable before installing HCPCI-MNT720M in or removing it from the computer. Installing the product without removing the power supply cable may cause a failure, or the device may exhibit an unexpected operation.



When installing a motionCAT series product in the computer, be sure the board is straight to the connector in the computer and press in the board so that the gold-plated edge connector is deeply inserted into the PCI connector. If the board is inserted at an angle or not inserted deeply enough, poor contact may cause an operating error or a board failure.



When installing a motionCAT series product in the computer, use mounting brackets and mounting screws to secure it firmly. Improper securing of the board may cause an operating error.



Caution












Do not drop a motionCAT series product or handle it roughly. Vibration or shock may cause a failure.






Do not touch the solder surface of a motionCAT series product with your hands. Sharp points on the components may cause injury.



6. Wiring

 Warning	
	Turn OFF the power supply to the computer and disconnect the power supply cable before wiring connectors to exterior lines or before connecting or disconnecting such connectors. Performing this work without removing the power supply cable may cause a failure. Furthermore, the device may exhibit an unexpected operation.
	When wiring connectors to exterior lines, check connector signal tables carefully and wire all lines correctly. Incorrect wiring may cause a failure or burning.
	Always keep external power supply within ratings. Using a power supply that is not within ratings may cause a failure, burning, or operating error.
	Always keep circuits connected to the I/O circuits within the rated currents and voltages. Using circuits that are not within ratings may cause a failure, burning, or operating error.
	Use the recommended connectors for external lines. Using any other connector may cause an operating error because of faulty contact.
	Always lock connectors for external lines. A connector that is not locked may come loose, possibly causing an operating error because of faulty contact.
	Do not pull on or place heavy objects on cables for external wiring. The connector may come loose and faulty contact may cause an operating error.
	Separate cables for external wiring as far as possible from AC power cables, motor cables, or other cables that generate excessive noise. Noise may cause an operating error.

7. Trial Operation and Adjustment

 Warning	
	Always debug the program completely before using the motionCAT series products to drive devices. Any error in the program may cause unexpected operation.
	When using sample programs provided with the motionCAT series products to operate devices, always start at low speed and be sure that settings match the mechanical system before attempting operation. Operating with settings that do not match the mechanical system may result in unexpected operation.

8. Disposal

 Warning	
	Abide by all applicable laws and ordinances when disposing of the motionCAT series products.

[Manual Configuration]

The motionCAT series products come with the manuals shown below.

■ **motionCAT series User's Manual This manual**

This manual provides the following information.

- (1) Advantages, specifications, and configurations of motionCAT
- (2) motionCAT master board
- (3) motionCAT software (driver installation, sample programs, etc.)
- (4) Operation of motionCAT motion modules

■ **motionCAT series HMG Slave User's Manual**

This manual provides information mainly on HMG slave hardware.

- (1) Slave configuration/models
- (2) Slave installation and wiring
- (3) Slave specifications
- (4) External connection
- (5) Safety instructions
- (6) Others

[Glossary]

(1) Terms used in motionCAT Series products

Term	Explanation
System communication	Communication for acquiring connection status, device type, and I/O port configuration, etc. from each local device.
Cyclic communication	Cyclic communication with local devices. (A communication that is performed sequentially, beginning with the lowest numbered local device and when communication with the last numbered device finishes, started again from the lowest numbered device) It is used for inputting and outputting I/O port statuses.
Data communication	This communication is for transmitting data to/from motion devices (G9003) and the like. Data communication is enabled by sending a data transmission command from the CPU to interrupt the cyclic communication the center device is continuously performing. Data communication is impossible if cyclic communication is not started.
Center device (G9001A)	A device that communicates with local devices in the motionCAT system.
Local device	Generic term for DIO devices (G9002) and motion devices (G9003) that are controlled by the center device.
DIO device (G9002)	DIO device equipped with four I/O ports. It is controlled by the center device.
Motion device (G9003)	Motion device capable of controlling the motion of one axis. It is controlled by using various commands sent from the center device.
Line	Serial communication line of the center device. Communication line. Transmission line. System.
Line number	Identifier for the communication line of the center device that is mounted on the master board.
Local device information	Local device information on one line acquired by system communication.
Module ID	ID for identifying a module. ID range is 00h to 3Fh (0 to 63).
Master board	Master board where the center device is mounted. The communication of motionCAT system is controlled by this board.
Slave	A slave consists of a communication board and a number of modules.
Module	Circuit board for assembling a slave. There are DIO modules and motion modules.
DIO slave	A slave that consists of a communication board and DIO modules.
Motion slave	A slave that consists of a communication board and motion modules.
Composite slave	A slave that consists of a communication board and motion and DIO modules.
DIO module	Circuit board where a DIO device is mounted.
Motion module	Circuit board where a motion device is mounted.

Table 1. Glossary of terms used in motionCAT Series products

(2) Terms used in motion modules

Term	Explanation
Controlled axis	A motor (such as a servomotor or a stepping motor) being controlled by the motion module is called a controlled axis to distinguish it from other kinds of motors (such as a spindle rotation motor).
Continuous feed	Starting a motion by specifying a speed only, without specifying a stopping position. A stop command is used to stop the motion.
Positioning control	Control to move an object to a specified target position and stop it there, without defining the path leading to the target position.
Relative coordinates (Incremental)	The current position is regarded as the relative origin position and the target position is considered to be the coordinates in respect to that origin position. In the motion module, the feed coordinates are all given as relative coordinates.
Movement unit, speed unit	The unit of movement, or coordinates, is expressed in pulses. Speeds are expressed in pulses per second (pps).
Operation speed	A generic term that indicates a specified speed of positioning, origin returning, continuous feed, or other operation. The FH register (RFH) value multiplied by the speed multiplier makes the operation speed.
Base speed	The base speed is defined by the FL register (RFL) and the speed multiplier setting register (RMG). When an operation is started, its speed reaches the base speed directly from the stopped state. When being stopped, the operation is immediately stopped from the base speed. Acceleration is performed while the operation speed is above the base speed, and deceleration is stopped when the operation speed reaches the base speed. (during automatic acceleration/deceleration)
Auxiliary speed	This is used as the origin entry speed during some origin returning operations, or as the speed during backlash or slip operations. Figure 1 shows speed patterns.
	<p>Figure 1 Speed Patterns (Example: linear acceleration/deceleration, speed multiplier of 1)</p>
Axis sensors	<p>Figure 2 shows an example of a table mechanism directly moved by a motor. +ELS, -ELS, OLS, and DLS in the figure are called axis sensors.</p> <ul style="list-style-type: none"> +/-ELS A stroke end limit sensor (ELS) is provided on both ends of the direct movement axis. When the ELS in the direction of movement detects the carriage, the motion module immediately stops the command pulse output or performs deceleration stop. The axis can move only in the opposite direction when an ELS is detecting the carriage. The +/- corresponds to the direction of the coordinates. An ELS can also be used as the sensor origin position, depending on which origin returning method is selected. OLS This sensor functions as the sensor origin position or the deceleration sensor during a high-speed origin returning, depending on which origin returning method is selected. DLS A deceleration sensor. Upon detecting a DLS, the operation is decelerated toward the base speed. In motion modules, an OLS is often used as the deceleration sensor during an origin returning operation. Z-phase This is used when Z-phase signals of the encoder (ENC) are set as the origin position. Deceleration is started using an OLS, and the origin returning completion is brought about by the specified nth (n: 1 to 15) z-phase signal. Instead of the ENC Z-phase, a photo-interrupter placed on the rotation axis can also be used as Z-phase signals to generate one pulse per rotation. <p>Figure 2 Axis sensors (direct movement axis)</p>

(To be continued)

(Continued from previous page.)

Term	Explanation
Synchronous input/output for external devices	<p>There are LTCH signals, CLR signals, and CMP output in motion modules for synchronization with external parts.</p> <p>LTCH Count values can be latched by LTCH input, which can be enabled by the DIP switch on a motion module.</p> <p>CLR Count values can be cleared by CLR input, which can be enabled by the DIP switch on a motion module.</p> <p>CMP output The specified counter is compared with the set comparator data to produce an external output upon matching the comparator condition. The "synchronous output" function is also available to produce outputs at regular intervals by counting. (Comparator 3 only)</p>
Command pulses and motor driver	<p>Command pulses Command pulses are output to a motor driver. Command pulse-train signals are given to a servo driver or a stepping motor driver. One command pulse rotates the stepping motor one step angle. For servomotors, one command pulse rotates the positioning detection device (normally the encoder built into the motor) for one pulse. Therefore, each command pulse provides a position (travel distance), and the command pulse frequency gives the rotational speed.</p> <p>Driver Compatible drivers connected to the motion module are those supporting pulse-train input. Therefore, any servo motor, linear motor, or stepping motor can be controlled as long as the motor driver is able to handle pulse-train input.</p> <div data-bbox="491 824 1235 1037" style="text-align: center;"> <p>The diagram illustrates the connection between a control board, a motor driver, and a motor with an encoder. On the left, a 'Board' has two inverters. Three pulse trains are shown: one from the top inverter to the Motor driver, and two from the bottom inverter to the Motor driver. The Motor driver is a rectangular block with two input terminals. On the right, a 'Motor (Encoder)' is shown with two signal lines connecting it to the Motor driver: one line goes from the Motor to the driver, and the other from the driver to the Motor.</p> </div> <p style="text-align: center;">Figure 3 Command pulses and a motor/motor driver</p>
Speed override	By writing speed data in the operation speed FH register (RFH), the operation speed can be changed during operation.
Position override	<p>A position override refers to an operation to change the end position, after the start of the original positioning feed (e.g. travel distance: 10, 000), prior to reaching the original end position. There are two operation patterns:</p> <p>(1) Changing the original end position to a point further away.</p> <p>(2) Changing the original end position to a closer point.</p> <p>In pattern (2), operation stops at the time of the command, then reverses toward the new end position.</p>
Backlash and slip Operations	Backlash compensation inserts a preset amount of compensation immediately before a commanded movement each time the movement direction is changed. A slip operation inputs an amount of compensation regardless of the movement method. Slip operation is useful for friction drive systems, such as those that have feed slip.

Figure 2. Terms used in motion modules

Contents

1. Introduction	1
2. What Is motionCAT (Motion by Control Automation Technology)?	1
2.1 Advantages of motionCAT series products	1
2.2 Connection rules for motionCAT slaves	2
2.3 Communication Specifications	3
2.4 motionCAT series product configuration	3
2.5 Accompanying software	4
2.5.1 Accompanying software for Windows	4
2.6 Control overview	4
2.6.1 Control overview	4
(1) motionCAT configuration	4
(2) Module IDs	5
2.6.2 Communication overview	6
(1) Cyclic communication	6
(2) Data communication	8
2.6.3 Calculating the communication time	9
(1) Amount of time for a cyclic communication	9
(2) Amount of time for one data communication	9
(3) Total amount of time for a cyclic communication including data communications	10
2.6.4 Communication cable disconnection detection/monitoring function	10
3. Master Board Hardware	11
3.1 Model numbers	11
3.2 Block diagram	11
3.2.1 HPCI-MNT520M block diagram	11
3.2.2 HPCI-MNT720M block diagram	11
3.3 Port addresses and PCI configuration register	12
3.3.1 Board addresses	12
3.3.2 PCI configuration register	13
3.4 Board settings and connector positions	14
3.4.1 HPCI-MNT520M settings	14
3.4.2 HPCI-MNT720M settings	14
3.4.3 Master board settings	14
3.5 Connector signal tables	15
3.6 Generic input/output circuits	16
3.6.1 Input and output circuit types	16
3.6.2 External connection examples	16
3.7 Master Board Specifications	17
4. Slave Hardware Overview	18
4.1 Slave configuration	18
4.1.1 Slave and modules	18
4.2 Slave product lineup	18
4.3 Slave specifications	18
4.3.1 Communication board	18
4.3.2 Slave cabinet	19
4.4 Composite slave	20
4.4.1 Notes for the use of composite slaves	20
4.5 Motion slave specifications	21
4.5.1 Motion slave configuration	21
4.5.2 For HMG-Px 1-axis motion module specifications (x: 1 to 6)	22
4.6 DIO slave specifications	23
4.6.1 DIO slave configuration	23
4.6.2 HMG-DIOx module specifications (x: 1 to 4)	24
5. Center Device (G9001A)/ Option Ports	25
5.1 Center device (G9001A)	25

5.1.1	Local device information (DINFO).....	25
5.1.2	Cyclic Communication Error Flag (IOERR).....	26
5.1.3	Input-Port Change Flag Setting (IINT).....	27
5.1.4	Input-Port Change Flag (IRES).....	29
5.1.5	Port Data (DATA).....	31
5.1.6	Center Main Status (CMSTS).....	32
5.1.7	Center Interrupt Status (CISTS).....	33
5.1.8	Center Device commands (CCMD).....	34
	(1) Center Operation commands.....	34
	(2) Center Register Control commands.....	35
	(3) Reading and writing from/to the center register.....	36
5.1.9	Center device register.....	36
	(1) RENV0 register(Read/Write).....	36
	(2) RERCNT (Read only).....	37
	(3) RSYCNT (Read only).....	37
	(4) RDJADD (Read only).....	37
	(5) RVER (Read only).....	37
5.2	Option ports.....	37
	(1) Generic Input Port.....	38
	(2) Generic Output Setting Port.....	38
	(3) G9001 Status.....	38
	(4) Transmission Speed Checking register.....	38
	(5) PCI Interrupt Permission.....	39
	(6) PCI Interrupt Status Port.....	39
	(7) Board ID Checking Port.....	39
	(8) Board Code Checking Port.....	39
6.	Local Devices.....	40
6.1	DIO device (G9002).....	40
6.2	Motion device (G9003).....	40
	6.2.1 Advantages.....	40
	6.2.2 I/O ports.....	42
	6.2.3 Motion Main Status (MMSTS).....	42
	6.2.4 Generic Output Status Port (IOPIB).....	42
	6.2.5 Generic Output Port (IOPOB).....	43
	6.2.6 Motion operation commands, control commands, register control commands.....	43
	(1) Motion Operation commands.....	43
	(2) Motion Control commands.....	44
	(3) Motion Register Control commands.....	45
	6.2.7 Motion Device registers.....	47
	(1) RMV: Travel Distance register (28 bits).....	47
	(2) RFL: Base Speed register (17 bits).....	47
	(3) RFH: Operation Speed Setting register.....	47
	(4) RUR: Acceleration Rate register (16 bits).....	48
	(5) RDR: Deceleration Rate register (16 bits).....	48
	(6) RMG: Speed Multiplier Setting register (11 bits).....	48
	(7) RDP: Deceleration Starting Point register (24 bits).....	48
	(8) RMD: Operation Mode register (25 bits).....	49
	(9) RUS: Acceleration S-Curve register (16 bits).....	50
	(10) RDS: Deceleration S-Curve register (16 bits).....	50
	(11) RFA: Auxiliary Speed register (17 bits).....	50
	(12) RIRQ: Event Factor Setting register (13 bits).....	51
	(13) RIST: Event Status register (13 bits).....	51
	(14) REST: Error Status register (15 bits).....	52
	(15) RSTS: Expansion Status register (28 bits).....	53
	(16) RCTR1: Command Position Counter register (28 bits).....	53
	(17) RCTR2: Machine Position Counter register (28 bits).....	54
	(18) RCTR3: General-Purpose/Error Counter register (16 bits).....	54

(19) RCMP 1 to 3: Comparator 1 to 3 registers (28 bits).....	54
(20) RLTC1: Counter 1 Latch register (28 bits).....	54
(21) RLTC2: Counter 2 Latch register (28 bits).....	54
(22) RLTC3: Counter 3 Latch register (17 bits).....	55
(23) RPLS: Remaining Movement Pulse register (28 bits).....	55
(24) RSPD: Speed Monitoring Register (27 bits).....	55
(25) RSDC: Automatically-Calculated Deceleration Starting-Point Register.....	55
6.2.8 Motion Device Environment Setting Registers.....	56
(1) RENV1: Environment Setting 1 Register (30 bits).....	56
(2) RENV2: Environment Setting 2 Register (23 bits).....	58
(3) RENV3: Environment Setting 3 Register (31 bits).....	58
(4) RENV4: Environment Setting 4 Register (28 bits).....	60
(5) RENV5: Environment Setting 5 Register (32 bits).....	61
(6) RENV6: Environment Setting 6 Register (32 bits).....	61
7. Software.....	62
7.1 Software configuration.....	62
7.1.1 Configuration of the software for Windows.....	62
7.2 Installing and uninstalling the device driver for Windows.....	63
7.2.1 Installation on Windows Vista.....	63
7.2.2 Installation on Windows XP.....	63
7.2.3 Installation on Windows 2000.....	63
7.2.4 Uninstalling the device driver for Windows.....	63
7.3 Using two or more master boards.....	64
7.4 Preparation for application building.....	64
7.5 How to access the board.....	65
7.5.1 Data structures for board (device) recognition.....	65
7.5.2 Board access preparatory procedure and end processing.....	66
7.6 Driver functions.....	67
7.6.1 List of driver functions.....	67
7.6.2 Return values of driver functions.....	68
7.6.3 Details on driver functions.....	69
7.7 Library functions.....	79
7.7.1 List of library functions.....	79
7.7.2 Return values of library functions.....	80
7.7.3 Details on library functions.....	81
7.8 Sample software.....	96
7.8.1 A simplest process of starting a cyclic communication.....	96
7.8.2 Setting, checking, and clearing interrupt-on-input-changes.....	97
7.8.3 Checking and clearing a cyclic communication error.....	98
7.8.4 Data exchange with port data areas (port information, data device status, etc.).....	99
7.8.5 Data communication (Part1): Setting a value for a motion device register.....	100
7.8.6 Data communication (Part 2): Reading a motion device register.....	101
7.9 Accompanying software.....	102
7.9.1 Device driver for Windows.....	102
7.9.2 Sample programs for Windows.....	102
(1) Sample for DIO modules.....	102
(2) Sample for motion modules.....	104
7.9.3 "Quick Start" for Windows.....	109
(1) Selecting modules.....	109
(2) Operating motion modules.....	110
(3) Operating DIO modules.....	113
8. Motion Module Operation.....	114
8.1 Programming procedure.....	114
8.1.1 Master board initial settings.....	114
8.2 Motion module (device) initial settings.....	115
8.2.1 Input specifications settings of machine sensors.....	116
8.2.2 Servo interface settings.....	117

8.2.3	Origin-returning method settings.....	118
(1)	Origin sensor settings and origin-returning methods.....	119
8.2.4	Event factor settings.....	121
8.3	Motion module (device) operation	121
8.3.1	Operation modes	121
(1)	Continuous operation by command	122
(2)	Positioning operation	122
(3)	Origin-returning operation.....	123
(4)	ELS, SLS operation.....	124
(5)	Z-phase Operation.....	125
(6)	Manual pulsar operation.....	126
8.3.2	Speed pattern settings	129
8.3.3	Speed patterns.....	130
8.3.4	Speed Pattern Setting registers	131
(1)	RMV: Travel Distance register (28 bits)	131
(2)	RFL: Base Speed register (17 bits)	131
(3)	RFH: Operation Speed Setting register (17 bits).....	132
(4)	RUR: Acceleration Rate register (16 bits).....	132
(5)	RDR: Deceleration Rate register (16 bits).....	133
(6)	RMG: Speed Multiplier Setting register (11 bits).....	133
(7)	RDP: Deceleration Starting Point register (24 bits)	134
(8)	RUS: Acceleration S-Curve register (16 bits)	134
(9)	RDS: Deceleration S-Curve register (16 bits).....	135
(10)	RFA: Auxiliary Speed register (17 bits)	135
8.4	Status processing.....	136
8.5	Applications	138
8.5.1	Simultaneous start using STA input.....	138
8.5.2	Group start.....	138
8.5.3	Comparator settings.....	139
8.5.4	Soft-limit settings	141
8.5.5	Synchronous output settings.....	142
8.5.6	Starting of other motion modules in the same slave using Comparator 3	143

Contents of Tables and Figures

1.	Introduction		
2.	What Is motionCAT (Motion by Control Automation Technology)?		
	Table 2.	1-1	Transmission cycle 1
	Figure 2.	2-1	motionCAT slave connection rules 2
	Table 2.	3-1	Communication specifications 3
	Table 2.	4-1	motionCAT product configuration and model numbers 3
	Figure 2.	6-1	motionCAT configuration 4
	Figure 2.	6-2	HMG-type slave configuration 4
	Table 2.	6-1	Device types 4
	Figure 2.	6-4	Slave configuration and its part names as well as module IDs 5
	Figure 2.	6-5	One set in cyclic communication 6
	Table 2.	6-2	Data exchanged in cyclic communication 6
	Table 2.	6-3	Port data addresses 7
	Figure 2.	6-6	Data communication 8
	Figure 2.	6-7	No consecutive data communications 8
	Figure 2.	6-8	Bit configuration of a data/system transmission command 8
	Table 2.	6-9	Contents of data/system transmission command 8
	Figure 2.	6-9	Disconnection detection 10
3.	Master Board Hardware		
	Figure 3.	2-1	HPCI-MNT520M block diagram 11
	Figure 3.	2-2	HCPCI-MNT720M block diagram 11
	Table 3.	3-1	Board addresses 12
	Table 3.	3-2	PCI configuration register 13
	Table 3.	3-3	PCI address space 13
	Figure 3.	4-1	Positions of the connectors and switches on HPCI-MNT520M 14
	Figure 3.	4-2	Positions of the connectors and switches on HCPCI-MNT720M 14
	Figure 3.	4-3	Transmission speed switch 14
	Figure 3.	4-4	Board ID setting rotary switch 14
	Figure 3.	5-1	Connector positions on HPCI-MNT520M 15
	Figure 3.	5-2	Connector positions on HCPCI-MNT720M 15
	Table 3.	5-1	RJ45 (J1, J2) connector signals 15
	Table 3.	5-2	DIO pin assignment (J1) 15
	Figure 3.	6-3	Connection with a sink type 16
	Figure 3.	6-4	Connection example of an output circuit 16
	Table 3.	7-1	HPCI-MNT520M specifications 17
	Table 3.	7-2	HCPCI-MNT720M specifications 17
4.	Slave Hardware Overview		
	Table 4.	1-1	HMG slave configuration 18
	Table 4.	2-2	Slave product lineup 18
	Table 4.	3-1	Communication board specifications 19
	Figure 4.	3-1	Slave cabinets and dimensions 19
	Table 4.	4-1	Composite slave product name 20
	Figure 4.	4-1	Arranging composite slave modules 20
	Table 4.	5-1	Motion module model 21
	Figure 4.	5-1	Motion slave configuration 21
	Table 4.	5-2	HMG-Px motion module specifications 22
	Figure 4.	6-1	DIO module model 23
	Figure 4.	6-1	DIO slave configuration 23
	Table 4.	6-2	HMG-DIO specifications 24
5.	Center Device (G9001A)/ Option Ports		
	Table 5.	1-1	Local device information addresses 25
	Table 5.	1-2	Cyclic Communication Error Flag addresses 26
	Table 5.	1-3	Input-Port Change Flag Setting addresses 28

Table 5.	1-4	Input-Port Change Flag addresses	30
Table 5.	1-5	Port Data addresses	31
Figure 5.	1-1	Bit configuration of Center Main Status (CMSTS)	32
Table 5.	1-6	Contents of Center Main Status (MSTS)	32
Figure 5.	1-2	Bit configuration of Center Interrupt Status (CISTS)	33
Table 5.	1-7	Contents of Center Interrupt Status (CISTS)	33
Table 5.	1-8	Contents of Center Operation commands	35
Table 5.	1-9	Contents of Center Register Control commands	35
Figure 5.	1-3	Bit configuration of the RENV0 register	36
Table 5.	1-10	Contents of the RENV0 register	36
Figure 5.	1-4	RSYCNT	37
Figure 5.	1-5	RDJADD	37
Figure 5.	1-6	RVER	37
Figure 5.	2-1	Bit configuration of Generic Input Port	38
Figure 5.	2-1	Contents of Generic Input Port	38
Figure 5.	2-2	Bit configuration of Generic Output Setting and Output Status	38
Figure 5.	2-2	Contents of Generic Output Setting and Output Status Checking Port	38
Figure 5.	2-3	Bit configuration of G9001 Status Checking Port	38
Figure 5.	2-3	Contents of G9001 Status Checking	38
Figure 5.	2-4	Bit configuration of G9001 Transmission Speed Checking Port	38
Figure 5.	2-4	Contents of G9001 Transmission Speed Checking Port	38
Figure 5.	2-5	Bit configuration of PCI Interrupt Permission Setting Port	39
Table 5.	2-5	Contents of PCI Interrupt Permission Setting Port	39
Figure 5.	2-6	Bit configuration of PCI Interrupt Status Port	39
Figure 5.	2-6	Contents of PCI Interrupt Status Port	39
Figure 5.	2-7	Bit configuration of Board ID Checking Port	39
Table 5.	2-7	Contents of Board ID Checking Port	39
Table 5.	2-8	Contents of Board Code Checking Port	39
6. Local Devices			
Figure 6.	2-1	Motion module port assignments	42
Figure 6.	2-2	Bit configuration of Motion Main Status (MMSTS)	42
Table 6.	2-1	Contents of Motion Main Status (MMSTS)	42
Figure 6.	2-3	Bit configuration of Generic Output Status Port (IOPIB)	42
Figure 6.	2-2	Contents of Generic Output Status Port (IOPIB)	42
Figure 6.	2-4	Bit configuration of Generic Output Setting Port (IOPOB)	43
Table 6.	2-3	Contents of Generic Output Port (IOPOB)	43
Table 6.	2-4	Operation Commands	43
Table 6.	2-5	Speed Change commands	44
Table 6.	2-6	Stop commands	44
Table 6.	2-7	Stop commands	44
Table 6.	2-8	Motion Control commands	44
Table 6.	2-9	Motion Register Write commands	45
Table 6.	2-10	Motion Register Read commands	46
Figure 6.	2-5	RMV: Bit configuration of the Travel Distance register	47
Figure 6.	2-6	RFL: Bit configuration of the Base Speed register	47
Figure 6.	2-7	RFH: Bit configuration of the Operation Speed Setting register	47
Figure 6.	2-8	RUR: Bit configuration of the Acceleration Rate register	48
Figure 6.	2-9	RDR: Bit configuration of the Deceleration Rate register	48
Figure 6.	2-10	RMG: Bit configuration of the Speed Multiplier Setting register	48
Figure 6.	2-11	RDP: Bit configuration of the Deceleration Starting Point register	48
Figure 6.	2-12	RMD: Bit configuration of the Operation Mode register	49
Table 6.	2-11	RMD: Contents of the Operation Mode register	49
Figure 6.	2-13	RUS: Bit configuration of the Acceleration S-Curve register	50
Figure 6.	2-14	RDS: Bit configuration of the Deceleration S-Curve register	50
Figure 6.	2-15	RFA: Bit configuration of the Auxiliary Speed register	50
Figure 6.	2-16	Bit configuration of the Event Factor Setting register	51
Table 6.	2-12	RIRQ: Contents of the Event Factor Setting register	51

Figure 6.	2-17	Bit configuration of the Event Status register	51
Table 6.	2-13	RIST: Contents of the Event Status register	51
Figure 6.	2-18	Bit configuration of the Error Status register	52
Table 6.	2-14	REST: Contents of the Error Status register	52
Figure 6.	2-19	RSTS: Bit configuration of the Expansion Status register	53
Table 6.	2-15	RSTS: Contents of the Expansion Status register	53
Figure 6.	2-20	RCTR1: Bit configuration of the Command Position Counter register	53
Figure 6.	2-21	RCTR2: Bit configuration of the Machine Position Counter register	54
Figure 6.	2-22	RCTRz3: Bit configuration of the General-Purpose/Error Counter register	54
Figure 6.	2-23	RCMP 1 to 3: Bit configuration of Comparator 1 to 3 registers	54
Figure 6.	2-24	RLTC1: Bit configuration of the Counter 1 Latch register	54
Figure 6.	2-25	RLTC2: Bit configuration of the Counter 2 Latch register	54
Figure 6.	2-26	RLTC3: Bit configuration of the Counter 3 Latch register	55
Figure 6.	2-27	RPLS: Bit configuration of the Remaining Movement Pulse register	55
Figure 6.	2-28	RSPD: Bit configuration of the Speed Monitoring register	55
Table 6.	2-16	RSPD: Contents of the Speed Monitoring register	55
Figure 6.	2-29	RSDC: Bit configuration of the Automatically-Calculated Deceleration Starting-Point register	55
Figure 6.	2-30	RENV1: Bit configuration of the Environment Setting 1 register	56
Table 6.	2-17	RENV1: Contents of Environment Setting 1 register	56
Figure 6.	2-31	Bit configuration of the Environment Setting 2 register	58
Table 6.	2-18	RENV2: Contents of Environment Setting 2 Register	58
Figure 6.	2-32	RENV3: Bit configuration of the Environment Setting 3 register	58
Table 6.	2-19	RENV3: Contents of Environment Setting 3 Register	59
Figure 6.	2-33	RENV4: Bit configuration of the Environment Setting 4 register	60
Table 6.	2-20	RENV4: Contents of Environment Setting 4 Register	60
Figure 6.	2-34	RENV5: Bit configuration of the Environment Setting 5 register	61
Table 6.	2-21	RENV5: Contents of Environment Setting 5 Register	61
Figure 6.	2-35	RENV6: Bit configuration of the Environment Setting 6 register	61
Table 6.	2-22	RENV6: Contents of Environment Setting 6 register	61
Figure 6.	2-36	Reverse pulses for vibration suppression	61
7. Software			
Figure 7.	1-1	Software relationships	62
Figure 7.	3-1	Using more than one board	64
Table 7.	6-1	List of driver functions	67
Table 7.	6-2	Return values of driver functions	68
Table 7.	7-1	List of library functions	79
Table 7.	7-2	Return values of library functions	80
Figure 7.	7-1	Initial values of the motion module registers	82
Figure 7.	8-1	Interrupt-on-input-change setting data	97
Figure 7.	9-1	DIO module sample screen	103
Figure 7.	9-2	Module ID combo box	103
Figure 7.	9-3	DIO module operation screen	104
Table 7.	9-3	Initial values of the sample program G9003 registers	104
Figure 7.	9-4	Program selection screen	105
Figure 7.	9-5	Open/Close Device screen	105
Figure 7.	9-6	Return Origin operation sample screen	106
Figure 7.	9-7	Input polarity selection	106
Figure 7.	9-8	Operation speed setting	106
Figure 7.	9-9	Return Origin start buttons	107
Figure 7.	9-10	Continuous feed operation sample screen	107
Figure 7.	9-11	Positioning operation sample screen	108
Figure 7.	9-12	Multiple axes operation sample screen	108
Figure 7.	9-13	Start-up screen	109
Figure 7.	9-14	Screen after master board selection	109
Figure 7.	9-15	Module selection screen	110
Table 7.	9-4	Initial values of "Quick Start" G9003 registers	110
Figure 7.	9-16	Motion module operation screen	111

Figure 7.	9-17	Signal input statuses	111
Figure 7.	9-18	Parameter settings for operation	112
Figure 7.	9-19	Operation start buttons	112
Figure 7.	9-20	DIO module operation screen	113
8. Motion Module Operation			
Figure 8.	1-1	Basic programming procedure	114
Table 8.	2-1	Initial settings for motion devices	115
Table 8.	2-2	Settings of machine sensor's input specifications	116
Table 8.	2-3	Servo interface settings	117
Table 8.	2-4	Origin-returning method settings	118
Table 8.	2-5	Origin-returning methods	119
Table 8.	2-6	Origin-returning methods (by CTR2 reference)	120
Figure 8.	2-1	RIRQ: Bit configuration of the Event Factor Setting register	121
Table 8.	3-1	List of operation modes	121
Table 8.	3-2	Continuous operation by command	122
Table 8.	3-3	Positioning operation	123
Table 8.	3-4	Origin-returning operation settings and the related statuses	123
Table 8.	3-5	Origin-returning operations	124
Table 8.	3-6	ELS, SLS operation	125
Table 8.	3-7	Z-phase operation	125
Table 8.	3-8	Examples of relationships between FH speed [pps] and pulsar input frequency FP [pps]	127
Table 8.	3-9	Pulsar A/B-phase input settings	128
Table 8.	3-10	Manual pulsar operation	128
Table 8.	3-11	Speed pattern setting registers	129
Figure 8.	3-1	Sections defined by the registers in acceleration/deceleration operation	129
Table 8.	3-12	Speed patterns	130
Table 8.	3-13	Examples of speed multiplier settings	133
Figure 8.	4-4	Bit configuration of Motion Main Status (MMSTS)	136
Table 8.	4-1	Contents of Motion Main Status (MMSTS)	136
Figure 8.	4-2	RIST: Bit configuration of the Event Status register	136
Figure 8.	4-3	REST: Bit configuration of the Error Status register	137
Table 8.	4-2	REST: Contents of the Error Status register	137
Figure 8.	4-4	Status management example	137
Figure 8.	5-1	Setting example of the soft-limit	141
Figure 8.	5-2	Synchronous output	142
Figure 8.	5-3	Motion module top view	143

1. Introduction

This manual provides the following information regarding the motionCAT series products.

- (1) Advantages, specifications, and configurations of motionCAT
- (2) Master board
- (3) Slave overview
- (4) Center/local devices
- (5) Software (driver installation, sample programs, etc.)
- (6) Motion module operation

For information on the connections and settings of slaves offered as the motionCAT series products, refer to the corresponding slave manuals.

Our products usually come with the documents as follows:

- (1) "motionCAT series User's Manual" This manual
- (2) "motionCAT series HMG Slave User's Manual"
- (3) Accompanying software (CD) Compatible OSs:

*1. In this manual, the different editions of Windows Vista 32bit (x86) are collectively referred to as Windows Vista.

*2. In this manual, Windows XP Home Edition and Windows XP Professional are collectively referred to as Windows XP.

*3. In this manual, Windows 2000 Professional is referred to as Windows 2000.

2. What Is motionCAT (Motion by Control Automation Technology)?

MotionCAT is a motion network system that uses Motionnet[®].

It is capable of connecting any kinds of motor drivers supplied by various manufacturers.

Its communication master connects a number of slaves and LAN cables, controlling communications by the multi-dropped method.

* Motionnet is a registered trademark of Nippon Pulse Motor Co.,Ltd. MotionCAT uses the G9000 series manufactured by Nippon Pulse Motor, and controls the motors and I/Os by high-speed communications via LAN cables.

2.1 Advantages of motionCAT series products

- (1) Motion network system -> Reduced cost wiring Generic LAN cables are used.
- (2) 32 motors per one line, or 64 motors per two lines, can be connected for motor control.
- (3) 2048 I/Os (32 I/Os x 64 slaves) can be controlled.
- (4) A maximum total length of LAN cables per one line is 50 meters.
- (5) A motor control slave is connected to a driver that uses the pulse-train input method. Servomotors, stepping motors, etc. offered by any manufactures can be used.
- (6) Easy porting of software from the CPD board series.
- (7) Easy I/O control through the I/O port memory on the mother board automatically updated every transmission cycle.
- (8) Short transmission time, quick response -> 20Mbps

The following table shows a transmission cycle.

Connections per one line	8	16	32	Comment
Transmission cycle	0.12ms	0.24ms	0.49ms	Data communication time 19.3us/6 bytes (Writing to one register in a motion device)

Table 2. 1-1 Transmission cycle

2.2 Connection rules for motionCAT slaves

The following "Communication cable connection rules" must be observed in connecting the mother board.

Communication cable connection rules

1. Cables to be used Shielded LAN cables, CAT5e or CAT6
2. Total number of modules per line 32 modules or less
(In the example shown below, 2 modules for the #1 Slave, 1 module for the #2 Slave)
3. Total length of cables per line 50 m or less
4. Minimum length of cable between slaves .. 60 cm or more
5. Setting for the last module The communication settings switch (termination) must be set to ON.
(See the user's manual for each slave.)
6. Connections to in/out LAN connectors The two LAN connectors of a slave can be connected in any order. (No particular connection order)

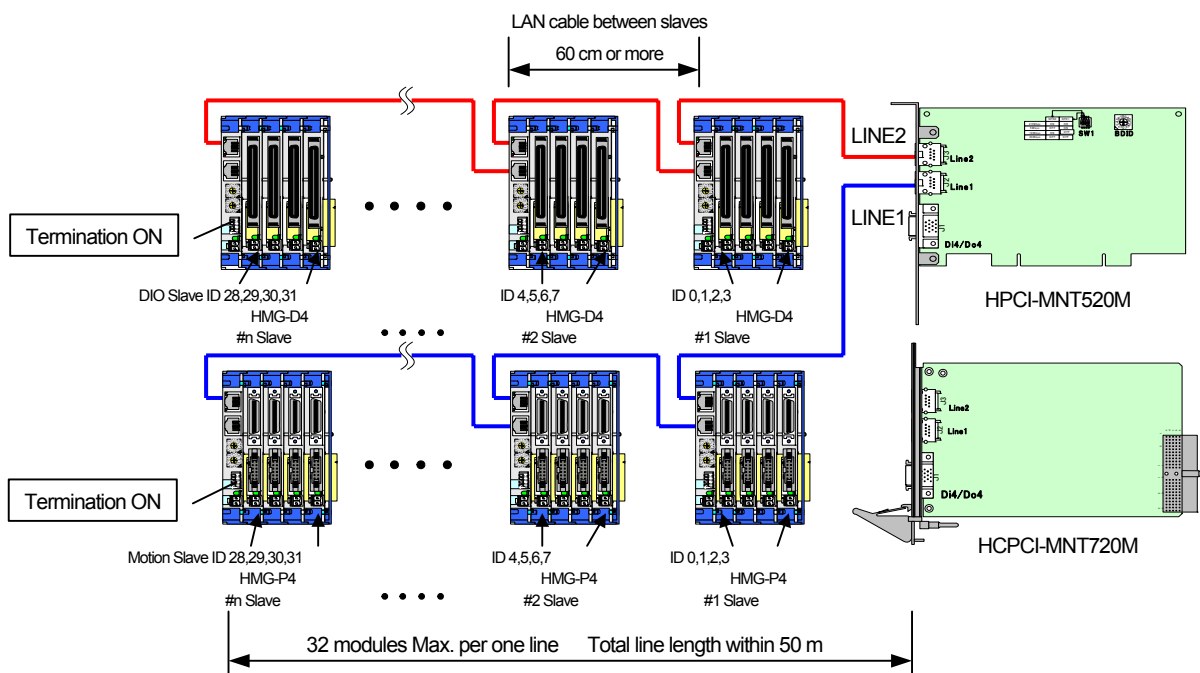


Figure 2. 2-1 motionCAT slave connection rules

2.3 Communication Specifications

Item	Specification
Reference clock	40 or 80 MHz (depends on system)
Transmission speed	2.5, 5, 10, or 20 Mbps
Communication code	NRZ
Communication protocol	Motionnet by Nippon Pulse Motor
Communication system	Half-duplex communication
Communication I/F	RS-485, pulse transformer
Connection method	Multi-dropped
Transmission distance	50 m
Number of local devices connected	32
Time per cyclic communication	At 20 Mbps: 8 local devices on one line 0.12 ms 16 local devices on one line.... 0.24 ms 32 local devices on one line.... 0.49 ms
Data communication time	Transmission of 3 words * (writing to one register in a motion device) 19.3μs
Transmission data size	1 to 3 words / frame
For communication cable disconnection	Motion stop upon detecting disconnection

* 1 word = 2 bytes = 16 bits

Table 2. 3-1 Communication specifications

2.4 motionCAT series product configuration

MotionCAT is capable of connecting a number of slaves to the master board by the multi-dropped method.

The master board provides two communication lines, allowing each to connect with a maximum of 32 modules.

A slave consists of one communication board and one to four (six) modules. The communication board offers communication functions. The modules are available as motion modules and DIO modules.

The following table shows our product configuration as well as model numbers.

Category	Product name	Model number	Comment
Master board	PCI master board	HPCI -MNT520M	
	Compact PCI master board	HCPCI-MNT720M	
Motion slave	HMG 1-axis motion slave (1 axis x 1)	HMG-P1	Pulse-train Positioning
	HMG 2-axes motion slave (1 axis x 2)	HMG-P2	
	HMG 3-axes motion slave (1 axis x 3)	HMG-P3	
	HMG 4-axes motion slave (1 axis x 4)	HMG-P4	
	HMG 5-axes motion slave (1 axis x 5)	HMG-P5	
	HMG 6-axes motion slave (1 axis x 6)	HMG-P6	
DIO slave	HMG DIO 32 slave (16-in/16-out x 1)	HMG-D1	Photo isolation Generic I/O
	HMG DIO 64 slave (16-in/16-out x 2)	HMG-D2	
	HMG DIO 96 slave (16-in/16-out x 3)	HMG-D3	
	HMG DIO 128 slave (16-in/16-out x 4)	HMG-D4	
Composite slave	HMG 1-axis + DIO 32 slave	HMG-P1D1	Combination of the slaves above
	HMG 1-axis + DIO 64 slave	HMG-P1D2	
	HMG 1-axis + DIO 96 slave	HMG-P1D3	
	HMG 2-axes + DIO 32 slave	HMG-P2D1	
	HMG 2-axes + DIO 64 slave	HMG-P2D2	
	HMG 3-axes + DIO 32 slave	HMG-P3D1	

Table 2. 4-1 motionCAT product configuration and model numbers

2.5 Accompanying software

The motionCAT series products come with the following pieces of software, which are prepared to help you to understand the information on the software aspect described in this user's manual.

2.5.1 Accompanying software for Windows

- (1) Device drivers For Windows Vista, XP, and 2000
- (2) Sample programs
 - For motion modules Sample for instructing how to use 1-axis motion modules with driver functions
 - For DIO modules Sample for instructing how to use DIO modules with driver functions
- (3) Quick start Software for test run

2.6 Control overview

2.6.1 Control overview

(1) motionCAT configuration

The motionCAT system consists of a master board and slaves.

One master board is equipped with two center devices (G9001A) to control two separate communication lines. Each slave consists of a communication board and one to six modules. The modules are available as motion modules and DIO modules.

Each motion module is equipped with one motion device (G9003) capable of controlling one axis.

Each DIO module is equipped with one DIO device (G9002) capable of controlling I/O(16-in/16-out).

Also, motion and DIO devices mounted on modules are collectively referred to as local devices.

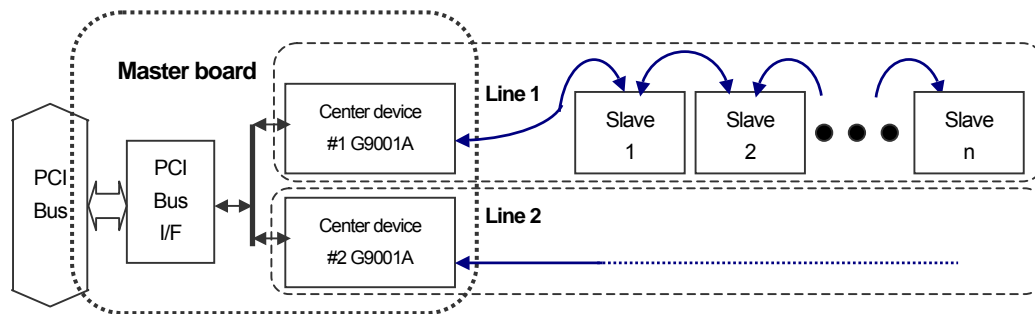


Figure 2. 6-1 motionCAT configuration

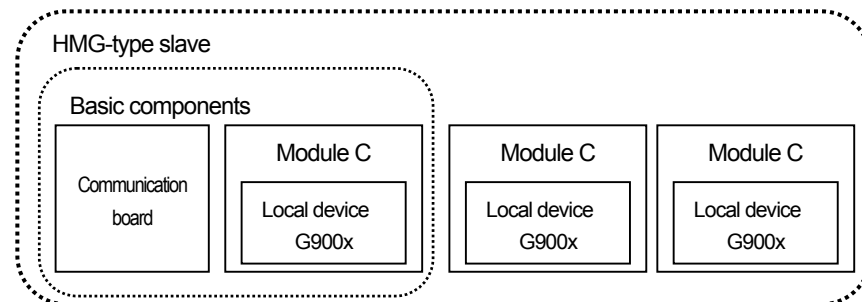


Figure 2. 6-2 HMG-type slave configuration

Device name		Model number	Function
Center device		G9001A	This device performs transmission control over all the local devices connected to one line.
Local device	Motion device	G9003	This device controls the motion of one axis.
	DIO device	G9002	With two sets of input and output ports, this device controls I/O(16-in/16-out).

Table 2. 6-1 Device types

(2) Module IDs

(1) A module ID (M ID) is an address (ID) to identify each module.

This ID is specified using the M ID decode switch on the communication board.

The 'M ID' for the base module is set with a value between HEX00 to 3C (0 to 60).

There must not be duplicate M IDs on the same LINE.

Note: Set the decode switch to the 'M ID' specified by the software functions.

(2) IDs of the succeeding modules

The modules of #2 onward are given "M IDs", or addresses, in increments of +1 according to the stacking order of those modules.

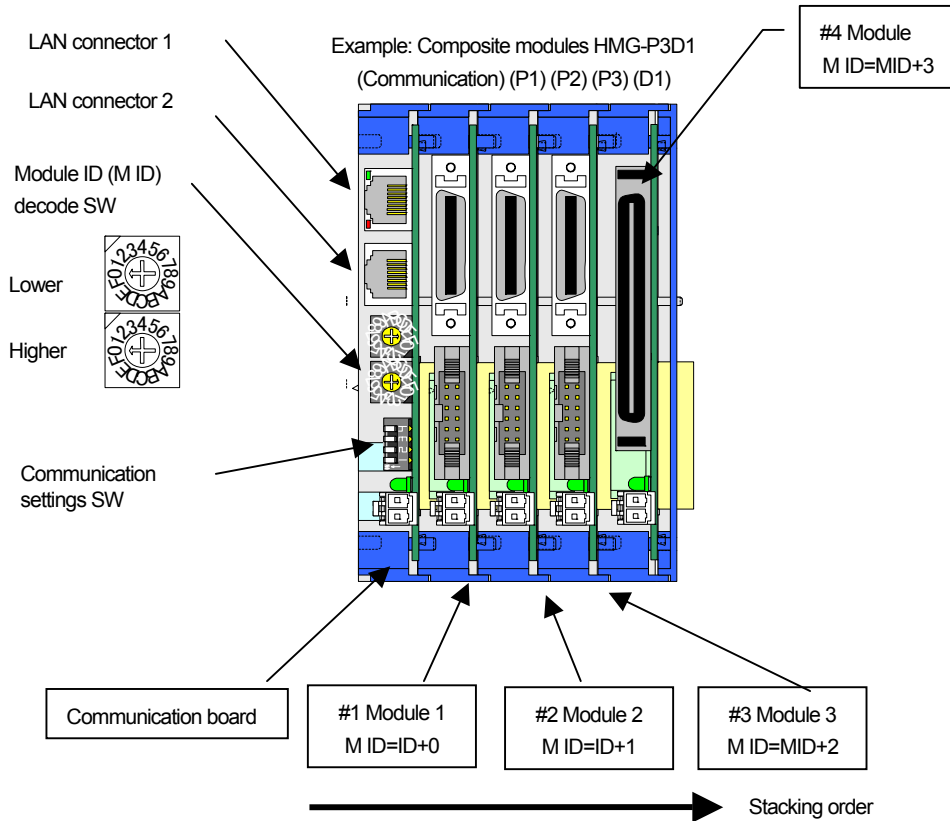


Figure 2. 6-4 Slave configuration and its part names as well as module IDs

2.6.2 Communication overview

MotionCAT uses Motionnet communications.

Motionnet communications include cyclic communications and data communications.

(1) Cyclic communication

In Motionnet communications, the center device controls the communication line.

Each local device can transmit only when allowed by the center device.

The center device communicates with each connected local device sequentially, beginning with that having the lowest numbered module ID. The center device starts by establishing a communication with the first local device. Over the communication line (one line), there is only one device that can receive this communication (there must not be duplicated module IDs).

The local device that received the communication is the next to have the right to use the communication line, which will be used for a communication from the local device to the center device. This constitutes one set of communications.

The center device establishes similar communications with all of the local devices in the order of module IDs.

This communication that is started again from the module with the lowest ID when the communication with the module with the last ID finishes is called cyclic communication.

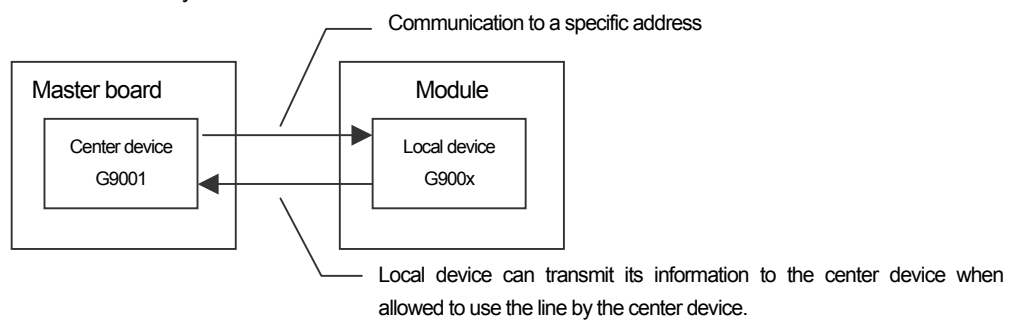


Figure 2. 6-5 One set in cyclic communication

[Data exchanged in cyclic communication]

Cyclic communication does not stop unless so specified by using a software command.

Size of data exchanged in cyclic communications is 4 bytes.

The contents of the data exchanged using this number of bytes is different depending on the local device.

Device name	Contents transmitted from the center device to the local device	Contents transmitted from the local device to the center device
G9002 (DIO device)	Output data to output port	Input data from input port
G9003 (Motion device)	Output data to generic output port	Input data from generic input port Motion device status

Table 2. 6-2 Data exchanged in cyclic communication

Communication time is explained later in "2.6.3 Calculating the communication time."

[Data acquisition using cyclic communication and module ID]

The master board is equipped with an I/O port memory (port data).

This memory is updated with the I/O port status of each module by the cyclic communication.

In this way, handling of I/O status is similar to accessing an ordinary memory.

The following table shows the I/O port memory address map.

Address	Read (INP) port data (DATA)	Write (OUT) port data (DATA)
BAR2+100	Line 1 module ID 0 port 1, 0 data	Line 1 module ID 0 port 1, 0 setting
+102	Line 1 module ID 0 port 3, 2 data	Line 1 module ID 0 port 3, 2 setting
+104	Line 1 module ID 1 port 1, 0 data	Line 1 module ID 1 port 1, 0 setting
+106	Line 1 module ID 1 port 3, 2 data	Line 1 module ID 1 port 3, 2 setting
+108	Line 1 module ID 2 port 1, 0 data	Line 1 module ID 2 port 1, 0 setting
+10A	Line 1 module ID 2 port 3, 2 data	Line 1 module ID 3 port 3, 2 setting
,		
+1FC	Line 1 module ID 63 port 1, 0 data	Line 1 module ID 63 port 1, 0 setting
+1FE	Line 1 module ID 63 port 3, 2 data	Line 1 module ID 63 port 3, 2 setting
,		
+300	Line 2 module ID 0 port 1, 0 data	Line 2 module ID 0 port 1, 0 setting
+302	Line 2 module ID 0 port 3, 2 data	Line 2 module ID 0 port 3, 2 setting
+304	Line 2 module ID 1 port 1, 0 data	Line 2 module ID 1 port 1, 0 setting
+306	Line 2 module ID 1 port 3, 2 data	Line 2 module ID 1 port 3, 2 setting
+308	Line 2 module ID 2 port 1, 0 data	Line 2 module ID 2 port 1, 0 setting
+30A	Line 2 module ID 2 port 3, 2 data	Line 2 module ID 3 port 3, 2 setting
,		
+3FC	Line 2 module ID 63 port 1, 0 data	Line 2 module ID 63 port 1, 0 setting
+3FE	Line 2 module ID 63 port 3, 2 data	Line 2 module ID 63 port 3, 2 setting

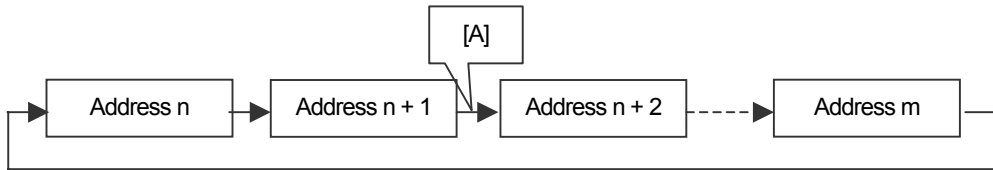
Table 2. 6-3 Port data addresses

(2) Data communication

Data communication provides commands to operate motion devices.

Data communication is performed by interrupting cyclic communication at any time (when a command is sent from the software). (see Note 1)

The following figure shows an example of cyclic communication sequence.



“Address n” represents the local device whose module ID is n. The arrows represent the sequence in cyclic communication. If a data transmission command is issued at [A]:

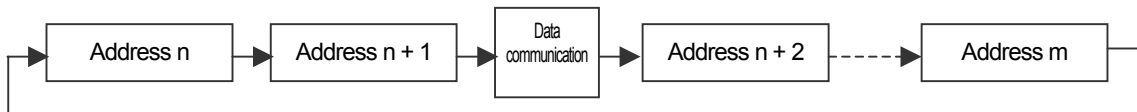


Figure 2. 6-6 Data communication

Although one cyclic communication is extended, the data exchange with the target device can be performed promptly. In data communication, like the case of cyclic communication, a response communication is transmitted to the center device from the local device that has received data.

Communication time is explained later in “2.6.3 Calculating the communication time.”

Note 1: Data communication is performed “at any time” as mentioned above, yet within the following constraint.

One cyclic communication must be inserted when performing consecutive data communications or retrying a data communication due to a communication error.

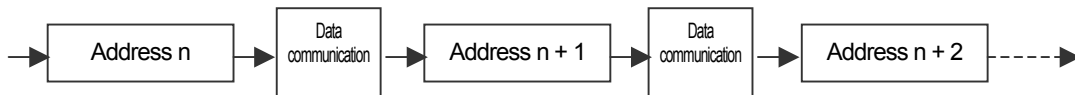


Figure 2. 6-7 No consecutive data communications

[Data transmission command and module ID specification]

The following figure and table show data transmission commands for specified module IDs.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	0	0	0	Module ID 00h to 3Fh					

Figure 2. 6-8 Bit configuration of a data/system transmission command

Command type	Command	Explanation
Data communication	0100 0000 00## #### (4000h to 403Fh)	The data in the sending FIFO is transmitted to a specified device. Response data is stored in the receiving FIFO.

Note. Bits indicated as “#”: The higher-order bits that specify a module ID are set sequentially from the top # bit.

Table 2. 6-9 Contents of data/system transmission command

2.6.3 Calculating the communication time

The communication time is classified into:

- (1) Amount of time for a cyclic communication
- (2) Amount of time for one data communication
- (3) Total amount of time for a cyclic communication including data communications

Under definitions as follows:

N: Number of connected local devices

B: Number of bytes of transmitted data (in case of two-byte data transmission: B = 2)

K: Transmission speed coefficient

Transmission speed (Mbps)	K
20	1
10	2
5	4
2.5	8

(1) Amount of time for a cyclic communication

Basic item	Time required (μ s)
Communication time for one local device (CT)	$7.7 \times K$

$$\text{Amount of time for a cyclic communication } (\mu\text{s}) = (\text{CT} + 7.4) \times N$$

Example: Where the transmission speed = 20 Mbps, The number of local devices = 30

$$\text{A cyclic communication time} = (7.7 \times 1 + 7.4) \times 30 = 453\mu\text{s}$$

(2) Amount of time for one data communication

Data communications are classified into the following two types:

- (1) When data is present in a response from the local device (variable-length data)
- (2) When data is absent in a response from the local device

Basic item	Time required (μ s)
Send time (ST)	$(B \times 0.6 + 3.25) \times K$
Respond time: with response data (JT)	$(B \times 0.6 + 5.65) \times K$
Respond time: without response data (JT)	$5.05 \times K$

$$\text{Amount of time for one data communication } (\mu\text{s}) = \text{ST} + \text{JT} + 7.4$$

Example 1: Time required for writing data to one register of a motion device, where the transmission speed = 20 Mbps:

A write command (2 bytes) and register data (4 bytes), totaling 6 bytes, are transmitted to write the data to the motion device register. There is no response data. (A response frame only)

$$\begin{aligned} \text{A data communication time} &= (6 \times 0.6 + 3.25) \times 1 + 5.05 \times 1 + 7.4 \\ &= 19.3\mu\text{s} \end{aligned}$$

Example 2: Time required for reading data from one register of a motion device, where the transmission speed = 20 Mbps:

A read command (2 bytes) is transmitted to write the data to the motion device register.

Response data consists of a read command (2 bytes) and register data (4 bytes), totaling 6 bytes.

$$\begin{aligned} \text{A data communication time} &= (2 \times 0.6 + 3.25) \times 1 + (6 \times 0.6 + 5.65) \times 1 + 7.4 \\ &= 21.1\mu\text{s} \end{aligned}$$

(3) Total amount of time for a cyclic communication including data communications

This is calculated by adding data communication times to a cyclic communication time.

Example 1: A total communication time required for a cyclic communication that includes data communications (reading from a motion device register) four times, each with 2-byte sending and 6-byte responding, where 32 local devices are connected with the transmission speed = 20 Mbps:

A total communication time

$$\begin{aligned} &= \text{a cyclic communication time} + (\text{a data communication time}) \times \text{the number of data communications} \\ &= (7.7 \times 1 + 7.4) \times 32 + \{(2 \times 0.6 + 3.25) \times 1 + (6 \times 0.6 + 5.65) \times 1 + 7.4\} \times 4 \\ &= 483.2 + 21.1 \times 4 \\ &= 567.6 \mu\text{s} \end{aligned}$$

Note 1: An actual communication time will be shorter than the above calculated value which includes allowance.

Note 2: If a communication error occurs, the actual communication time will be longer than the above calculated value due to a retry communication.

Note 3: The above communication time excludes delay time derived from the buses, OS, and application software.

2.6.4 Communication cable disconnection detection/monitoring function

This is a function to stop motion in the event of an accident in the Motion Network—LAN cable disconnection.

A disconnection is detected by each slave that monitors cyclic communications.

(The absence of a communication for 20 msec after the final communication is considered to be a disconnection, where the communication speed is 20 Mbps.) The slaves placed across the disconnected position from the master board, or those placed on the termination side, are cut off from communications upon a disconnection. Therefore, those slaves can detect the disconnection themselves, and reset the motion modules connected to the slaves.

On the other hand, the slaves placed on the master board side would not be reset when communications are maintained after the disconnection. So, the master's program monitors cyclic communications for an error (See 5.1.2 Cyclic Communication Error Flag (IOERR)), and stops cyclic communications upon error detection to reset those slaves as well.

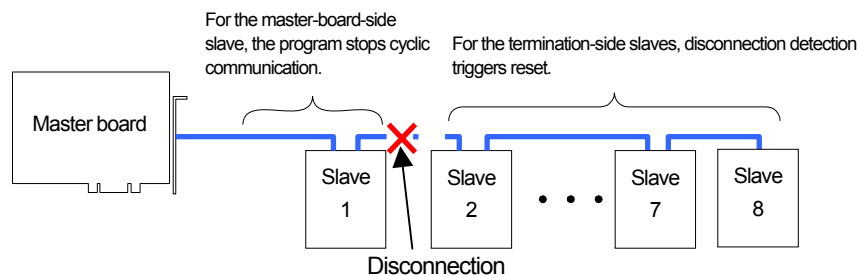


Figure 2. 6-9 Disconnection detection

3. Master Board Hardware

3.1 Model numbers

The MotionCAT master board is available in the following types:

- PCI Bus HPCI-MNT520M
- CompactPCI Bus HPCI-MNT720M

The difference between the specifications of these is found only in the bus.

3.2 Block diagram

3.2.1 HPCI-MNT520M block diagram

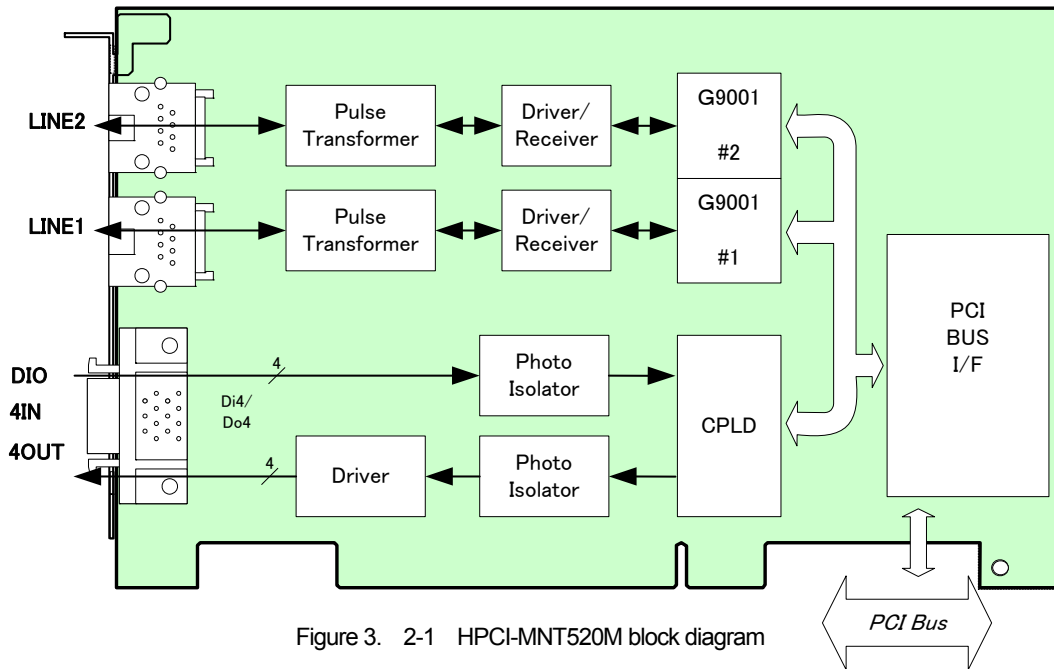


Figure 3. 2-1 HPCI-MNT520M block diagram

3.2.2 HPCI-MNT720M block diagram

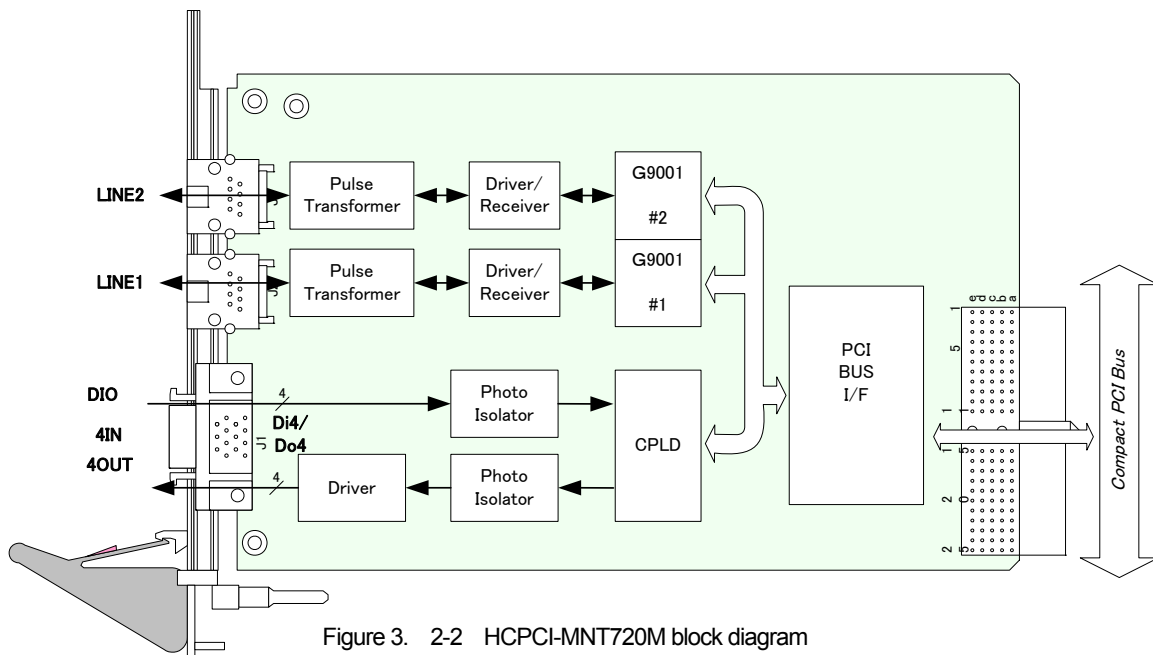


Figure 3. 2-2 HPCI-MNT720M block diagram

3.3 Port addresses and PCI configuration register

3.3.1 Board addresses

The same address maps are used for HPCI-MNT520M and HCPCI-MNT720M.
All the ports use memory maps.

Base	Category	Address	Read (INP)		Write (OUT)	
			Name	Contents	Name	Contents
BAR2	G9001A (#1) Line 1	+ 0	CMSTS	Center main status (15- 0)	CCMD	Command (15- 0)
		+ 2	CISTS	Center interrupt status (15- 0)	-	Not used (reserved)
		+ 4	CBUF	I/O buffer IN (15- 0)	CBUF	I/O buffer OUT (15- 0)
		+ 6	RFIFO	Data-receiving FIFO (15- 0)	SFIFO	Data-sending FIFO (15- 0)
		+ 8				
		'	-	Not used (reserved)	-	Not used (reserved)
		+77				
		+78	DINFO	Local device information (78-B7)	DINFO	Local device information (78-B7)
		+B8	IOERR	Cyclic-communication error flag (B8-BF)	IOERR	Cyclic communication error flag reset (B8-BF)
		+C0	IINT	Input-port change flag setting status (C0-DF)	IINT	Input-port change flag setting (C0-DF)
		+E0	IRES	Input-port change flag (E0-FF)	IRES	Input-port change flag reset (E0-FF)
	+100	DATA	Port data IN (100-1FF)	DATA	Port data OUT (100-1FF)	
	G9001A (#2) Line 2	+200	CMSTS	Center main status (15- 0)	CMD	Command (15- 0)
		+202	CISTS	Center interrupt status (15- 0)	-	Not used (reserved)
		+204	CBUF	I/O buffer IN (15- 0)	CBUF	I/O buffer OUT (15- 0)
		+206	RFIFO	Data-receiving FIFO (15- 0)	SFIFO	Data-sending FIFO (15- 0)
		+208				
		'	-	Not used (reserved)	-	Not used (reserved)
		+277				
		+278	DINFO	Local device information (278-2B7)	DINFO	Local device information (278-2B7)
		+2B8	IOERR	Cyclic-communication error flag (2B8-2BF)	IOERR	Cyclic communication error flag reset (2B8-2BF)
		+2C0	IINT	Input-port change flag setting status (2C0-2DF)	IINT	Input-port change flag setting (2C0-2DF)
+2E0		IRES	Input-port change flag (2E0-2FF)	IRES	Input-port change flag reset (2E0-2FF)	
+300	DATA	Port data IN (300-3FF)	DATA	Port data OUT (300-3FF)		
BAR3	Option ports	+0	DIN	Generic input status (J1: MDR14)	-	Not used (reserved)
		+2	DOUT	Generic output status	DOUT	Generic output status(J1: MDR14)
		+4	STS	G9001A status	-	Not used (reserved)
		+6	SPD	Transmission speed setting status	-	Not used (reserved)
		+8	INTE	PCI interrupt permission status	INTE	PCI interrupt permission
		+A	INTS	PCI bus interrupt status	-	Not used (reserved)
		+10	BID	Board ID setting status	-	Not used (reserved)
		+12	BCOD	Board Code 'MNT520__'	-	Not used (reserved)

BAR: base address

Table 3. 3-1 Board addresses

3.3.2 PCI configuration register

The contents of the PCI configuration register are the same in HPCI-MNT520M and HCPCI-MNT720M. The following table shows what the PCI configuration register contains.

31	24	23	16	15	8	7	0	Addresses
Device ID 1034h				Vendor ID 14a9h				00h
Device status				Device control				04h
Class code						Revision ID (00h)		08h
Base class (06h)		Sub class (80h)		Program interface				
Self test		Header type		Master latency timer		Cash line		0ch
Base address register	00000000h (Reserved)							10h
	PCI9030 (PCI configuration register)							14h
	G9001 base address (BAR2)							18h
	Option port base address (BAR3)							1ch
	00000000h (Reserved)							20h
	00000000h (Reserved)							24h
Card bus CIS pointer								28h
Sub system ID 1034h				Sub system vendor ID 14a9h				2ch
Reserved								30h ? fch

Table 3. 3-2 PCI configuration register

Base address	Target	Space	Bus width	Remarks
BAR2	#1G9001A	512 Byte	16bit	
BAR2+200h	#2G9001A	512 Byte	16bit	
BAR3	Option port	16 Byte	16bit	

Table 3. 3-3 PCI address space

BAR: base address

3.4 Board settings and connector positions

3.4.1 HPCI-MNT520M settings

HPCI-MNT520M features two switches to be configured and three connectors.

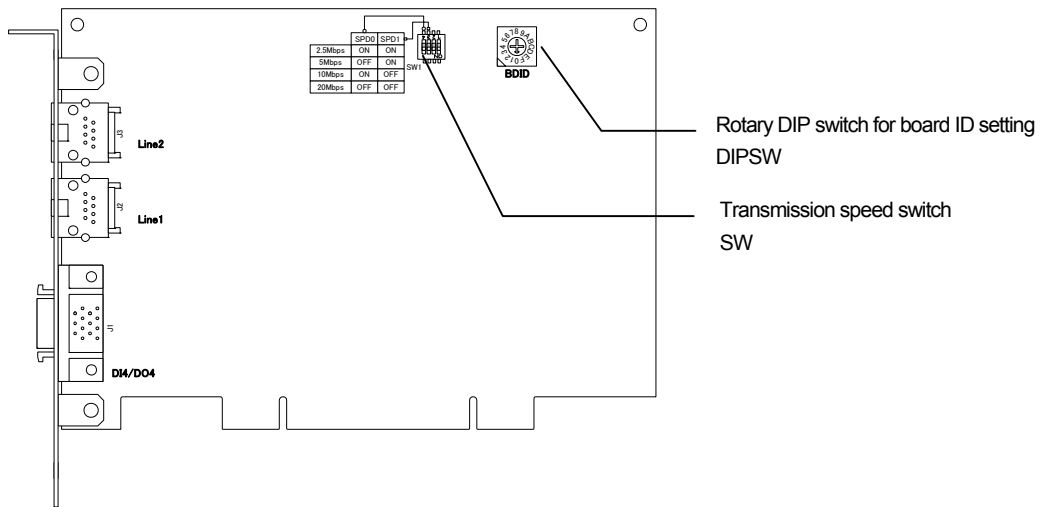


Figure 3. 4-1 Positions of the connectors and switches on HPCI-MNT520M

3.4.2 HCPCI-MNT720M settings

HCPCI-MNT720M features two switches to be configured and three connectors.

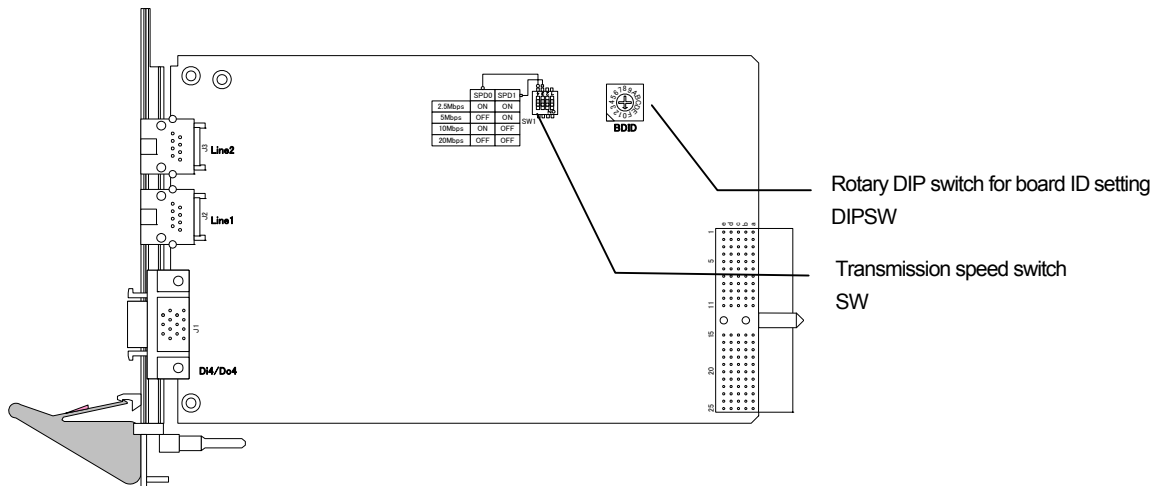


Figure 3. 4-2 Positions of the connectors and switches on HCPCI-MNT720M

3.4.3 Master board settings

The settings can be configured in the same way on HPCI-MNT520M and HCPCI-MNT720M.

Set the board ID and the transmission speed. Set the same transmission speed for the connected slaves, too.

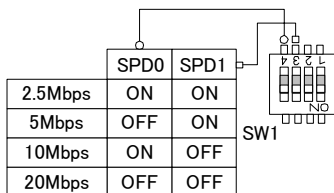


Figure 3. 4-3 Transmission speed switch



The ID range is 0h to Fh (0 to 15), which is used for identifying master boards. For how to use board IDs in the software, see "7.3 Using two or more master boards"

Figure 3. 4-4 Board ID setting rotary switch

3.5 Connector signal tables

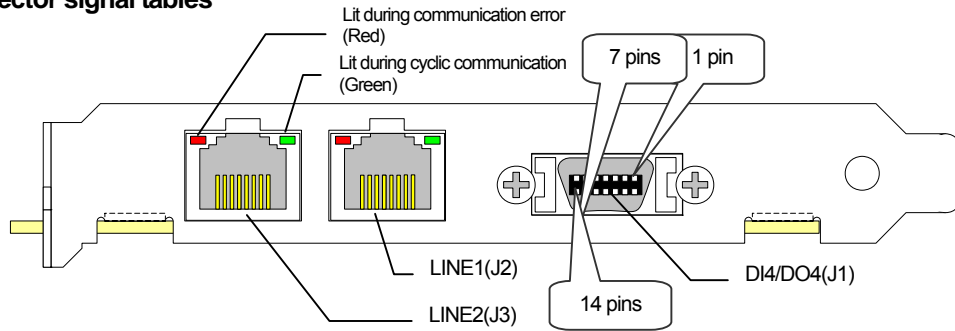


Figure 3. 5-1 Connector positions on HPCI-MNT520M

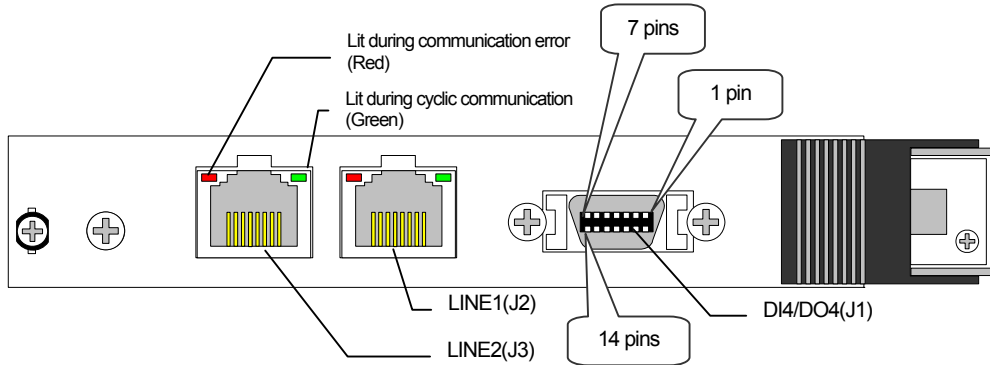


Figure 3. 5-2 Connector positions on HCPCI-MNT720M

(1) For Motionnet communication

J2(Line1) : Motionnet communication line 1

J3(Line2) : Motionnet communication line 2

Pin	Function
1	Frame to resistance for ground
2	Frame to resistance for ground
3	S +
4	Reserved
5	Reserved
6	S -
7	Frame to resistance for ground
8	Frame to resistance for ground

Table 3. 5-1 RJ45 (J1, J2) connector signals

(2) For generic input/output

Pin	Signal name	Pin	Signal name
1	EXTPOW (+24V input)	8	EXTPOW (+24V input)
2	IN1	9	IN2
3	IN3	10	IN4
4	EXTPOW (+24V input)	11	EXTPOW (+24V input)
5	OUT1	12	OUT2
6	OUT3	13	OUT4
7	EXTGND (+24V input)	14	EXTGND (+24V input)

Sumitomo 3M : 10214-52A2JL

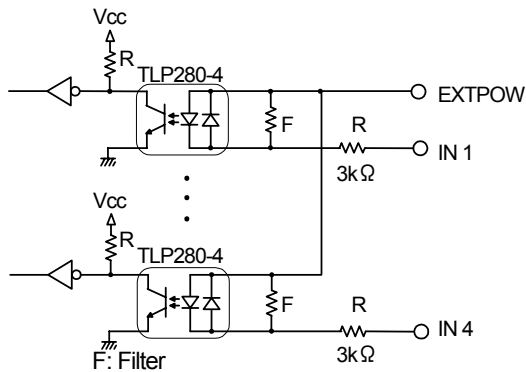
Recommended harness (cable side) Plug: (insulation displacement type)

Shell: (aluminum die casting)

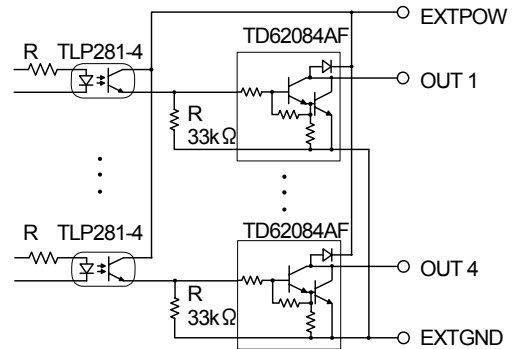
Table 3. 5-2 DIO pin assignment (J1)

3.6 Generic input/output circuits

3.6.1 Input and output circuit types



Rated input voltage: DC12V to DC24V
 Rated input current: 5mA/point
 Figure 3. 6-1 Input circuit



Rated load voltage: DC12V to DC24V
 Rated load current: 30mA or less/point
 Figure 3. 6-2 Output circuit

3.6.2 External connection examples

The following figures show typical connection examples of input/output circuits.

(1) Connection example of an input circuit

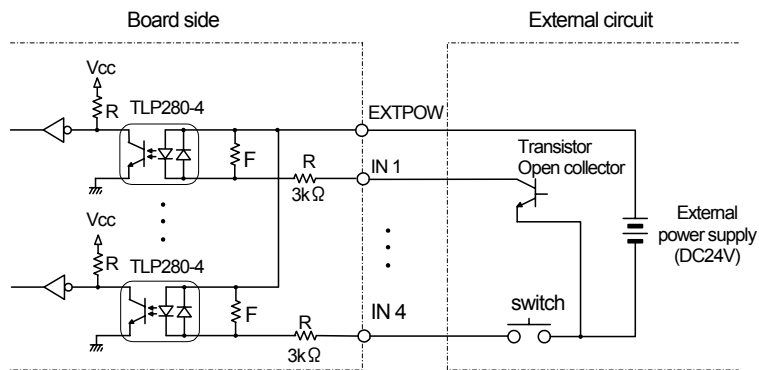


Figure 3. 6-3 Connection with a sink type

(2) HPCI-MNT720M Master board hardware specifications

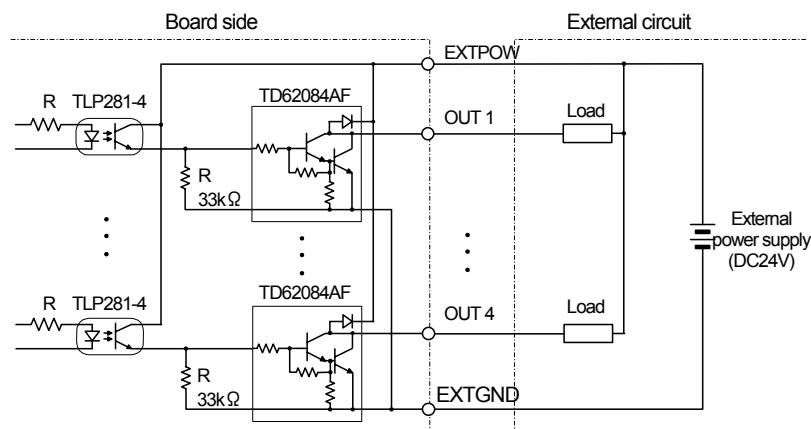


Figure 3. 6-4 Connection example of an output circuit

3.7 Master Board Specifications

(1) HPCI-MNT520M Master board hardware specifications

Category	Item	Specification	Remarks
[Basic specifications]	<ul style="list-style-type: none"> ■ Number of controlled lines ■ Transmission speed ■ Communication system ■ Communication I/F ■ Communication cable ■ Connection method 	Max. 2 lines Max. 32 modules per line (Total cable length per line: within 50 m) 20,10,5,2.5Mbps Half-duplex communication RS-485, pulse transformer Shielded LAN cable, CAT5e or CAT6 Multi-dropped	Control LSI: G9001A (by Nippon Pulse Motor)
[Bus specification]	<ul style="list-style-type: none"> ■ PCI Bus 	PCI Local Bus Specification Rev.2.2	
[Generic I/O]	<ul style="list-style-type: none"> ■ Photo isolation I/O ● Input port ● Output port 	4-in/4-out DIO port Input logic: '1' when input photo-coupler ON Rated input voltage: DC12V to DC24V Working input voltage range: DC12V to DC25V Rated input current: 5 mA/point Output logic: Output ON when written '1' Rated load voltage: DC5V to DC24V Load current: 80 mA or less/point	
[Environmental conditions]	<ul style="list-style-type: none"> ■ Supply power ■ Consumption current ■ Ambient temperature ■ Board dimensions 	+ 5V ± 5% 450mA MAX 0 to 50 °C (no condensation) Width 151 mm × Length 107 mm (short size)	

Table 3. 7-1 HPCI-MNT520M specifications

(2) HPCI-MNT720M Master board hardware specifications

Category	Item	Specification	Remarks
[Basic specifications]	<ul style="list-style-type: none"> ■ Number of controlled lines ■ Transmission speed ■ Communication system ■ Communication I/F ■ Communication cable ■ Connection method 	Max. 2 lines Max. 32 modules per line (Total cable length per line: within 50 m) 20,10,5,2.5Mbps Half-duplex communication RS-485, pulse transformer Shielded LAN cable, CAT5e or CAT6 Multi-dropped	Control LSI: G9001A (by Nippon Pulse Motor)
[Bus specification]	<ul style="list-style-type: none"> ■ PCI Bus 	PCI Local Bus Specification Rev.2.2	
[Generic I/O]	<ul style="list-style-type: none"> ■ Photo isolation I/O ● Input port ● Output port 	4-in/4-out DIO port Input logic: '1' when input photo-coupler ON Rated input voltage: DC12V to DC24V Working input voltage range: DC12V to DC25V Rated input current: 5 mA/point Output logic: Output ON when written '1' Rated load voltage: DC5V to DC24V Load current: 80 mA or less/point	
[Environmental conditions]	<ul style="list-style-type: none"> ■ Supply power ■ Consumption current ■ Ambient temperature ■ Board dimensions 	+ 5V ± 5% 500mA MAX 0 to 50 °C (no condensation) Width 160 mm × Length 100 mm (3U size)	

Table 3. 7-2 HPCI-MNT720M specifications

4. Slave Hardware Overview

This chapter describes an overview of slave hardware.
See "HMG Slave User's Manual" for details on slave hardware.

4.1 Slave configuration

4.1.1 Slave and modules

(2) An HMG consists of one communication board and one to six modules.

Component	Explanation	Remarks
Communication board (Functions specific to each slave)	This board provides Motionnet communication functions. Three functions configured by users are: (1) Two LAN connectors (2) Module ID decode switch [SW] (3) Communication settings switch (termination ON/OFF, communication cable disconnection detection enabled/disabled, two transmission speed settings)	
Module (Motion or DIO)	This module controls one-axis motion or I/O. The communication board and modules constitute a basic configuration.	

Table 4. 1-1 HMG slave configuration

4.2 Slave product lineup

The following table shows types of slaves available.

Category		Product name	Product name	Comment
HMG type	Motion slave	HMG 1-axis motion slave (1 axis x 1)	HMG-P1	Pulse-train Positioning
		HMG 2-axes motion slave (1 axis x 2)	HMG-P2	
		HMG 3-axes motion slave (1 axis x 3)	HMG-P3	
		HMG 4-axes motion slave (1 axis x 4)	HMG-P4	
		HMG 5-axes motion slave (1 axis x 5)	HMG-P5	
		HMG 6-axes motion slave (1 axis x 6)	HMG-P6	
	DIO slave	HMG DIO 32 slave (16-in/16-out x 1)	HMG-D1	Photo isolation Generic I/O
		HMG DIO 64 slave (16-in/16-out x 2)	HMG-D2	
		HMG DIO 96 slave (16-in/16-out x 3)	HMG-D3	
		HMG DIO 128 slave (16-in/16-out x 4)	HMG-D4	
	Composite slave	HMG 1-axis + DIO 32 slave	HMG-P1D1	Combination of the slaves above
		HMG 1-axis + DIO 64 slave	HMG-P1D2	
		HMG 1-axis + DIO 96 slave	HMG-P1D3	
		HMG 2-axes + DIO 32 slave	HMG-P2D1	
HMG 2-axes + DIO 64 slave		HMG-P2D2		
HMG 3-axes + DIO 32 slave		HMG-P3D1		

Table 4. 2-2 Slave product lineup

4.3 Slave specifications

A slave consists of a communication board and a cabinet, namely the communication board is attached to the slave cabinet.

4.3.1 Communication board

The roles of a communication board are as follows:

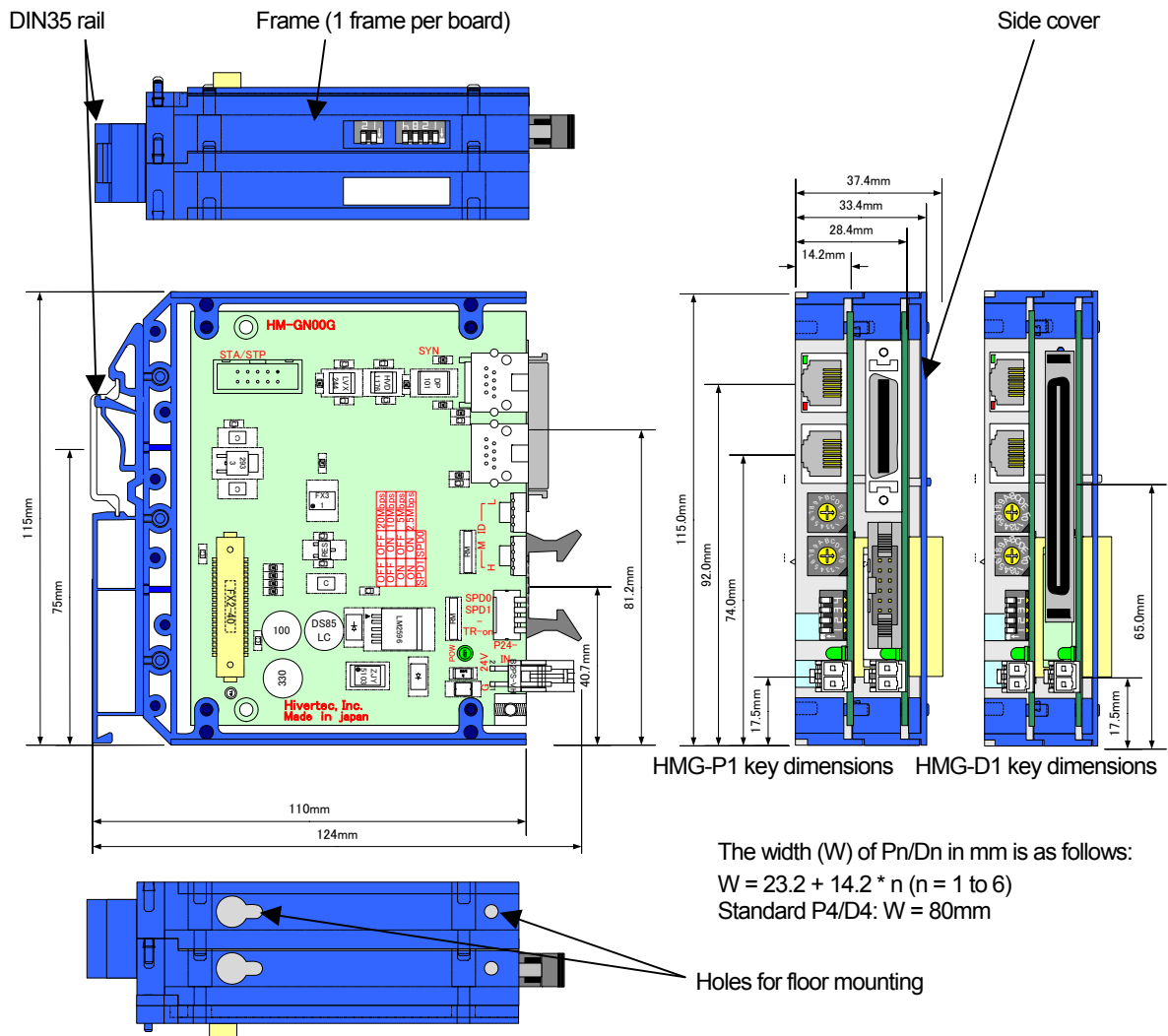
- (1) The communication board, with its driver and receiver, distributes transmission signals through LAN cables into the modules connected subsequently.
- (2) The board provides the connected modules with values set by the embedded module ID decode switch.
- (3) The board provides the communication settings switch to turn on/off the termination for the LAN cables. It also provides the connected modules with transmission speed set values.
- (4) The board distributes clock signals to the connected modules.
- (5) The board receives + 24 V supply power and provides the connected modules with control logic supply.

Category	Item	Specification	Remarks
Function	(1) Transmission standard	Based on Motionnet lines	
	(2) Connector	Cascade connection with two RJ45 connectors	
	(3) Transmission speed setting	20MHz, 10MHz, 5MHz, 2.5MHz (According to the master's transmission speed)	Communication setting switch
	(4) Termination	The termination SW of the last slave only set to ON	
	Power supply terminal	+24V, GND	
	Power distribution	Logic power supply is distributed to the modules connected subsequently. External +24V GND and logic GND shared.	Insulated from power supply terminal GND of each module.
Environmental conditions	Power ON indicator	ON when + 24V is applied to the power supply terminal (green light)	
	Communication status indicator	Indicated during cyclic communication (green light)	
	Communication cable disconnection detection	Motion module reset upon detecting cyclic communication failure	When communication settings switch (2nd) is ON
	Supply power	+12V to +24V (standard: +24V)	
	Logic power supply capacity	For max. four modules	
Ambient temperature	0 to 50 °C (no condensation)		
Board model number / dimensions	HM-GN00G D75mm × H105mm (RJ45 side)		

Table 4. 3-1 Communication board specifications

4.3.2 Slave cabinet

(1) Key dimensions and installation



The width (W) of Pn/Dn in mm is as follows:

$$W = 23.2 + 14.2 * n \quad (n = 1 \text{ to } 6)$$

Standard P4/D4: W = 80mm

Figure 4. 3-1 Slave cabinets and dimensions

4.4 Composite slave

A composite slave consists of motion and DIO modules. Available combinations are listed in the following table as shown earlier.

Composite slave product name		Part name	Number of modules	Module combination
1	HMG 1-axis +DIO 32 slave	HMG-P1D1	2	HM-P100C x1, HM-DIO32C x1
2	HMG 1-axis +DIO 64 slave	HMG-P1D2	3	HM-P100C x1, HM-DIO32C x2
3	HMG 1-axis +DIO 96 slave	HMG-P1D3	4	HM-P100C x1, HM-DIO32C x3
4	HMG 2-axis +DIO 32 slave	HMG-P2D1	3	HM-P100C x2, HM-DIO32C x1
5	HMG 2-axis +DIO 64 slave	HMG-P2D2	4	HM-P100C x2, HM-DIO32C x2
6	HMG 3-axis +DIO 32 slave	HMG-P3D1	4	HM-P100C x3, HM-DIO32C x1

Table 4. 4-1 Composite slave product name

4.4.1 Notes for the use of composite slaves

(1) How to arrange composite slave modules

Composite slave modules are arranged as illustrated in the figure below. Motion modules are followed by DIO modules (motion module -> motion module -> DIO module). Based on the ID specified in the decode switch, the module IDs are allocated sequentially.

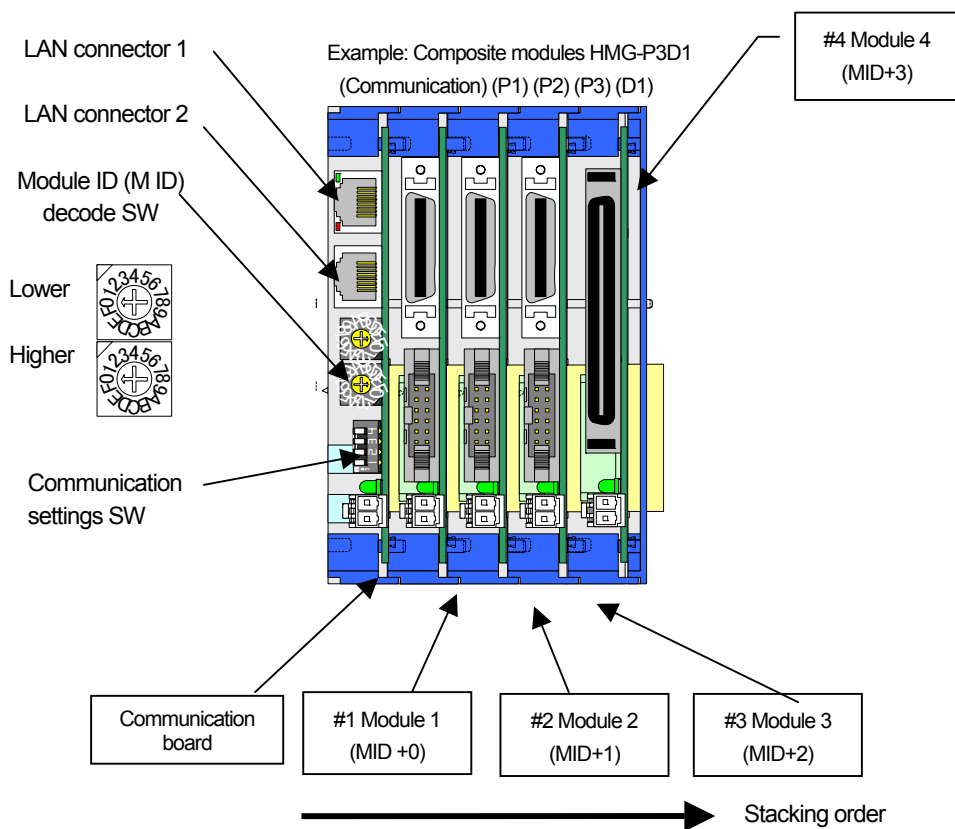


Figure 4. 4-1 Arranging composite slave modules

4.5 Motion slave specifications

A motion slave consists of a communication board and one to six 1-axis motion modules being stacked.

Module type	Model number	Name
Motion module	HM-P100C	1-axis motion module

Table 4. 5-1 Motion module model

See "HMG Slave User's Manual" for details on motion slaves.

4.5.1 Motion slave configuration

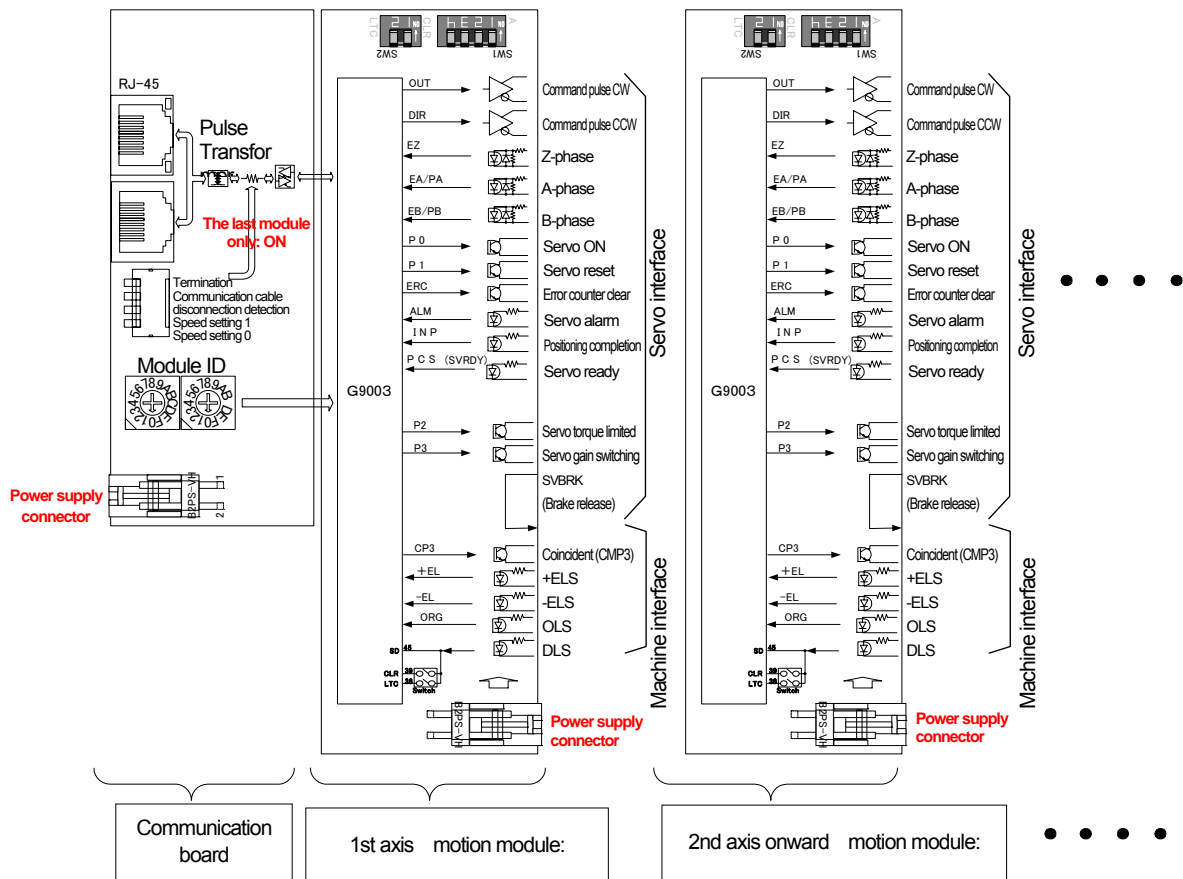


Figure 4. 5-1 Motion slave configuration

4.5.2 For HMG-Px 1-axis motion module specifications (x: 1 to 6)

Category	Item	Specification	Remarks	
Basic specifications	■ Axis control system Control method	One axis control per module Independent axis position control	Control LSI: G9001A (by Nippon Pulse Motor)	
	■ Position command			
	Command pulses	Positioning pulse-train command output	Output element: Differential driver	
	Position command range	-134,217,728 to 134,217,727 [pulses] [length 28-bit]		
	Command coordinate	Relative coordinate command		
	Command range for continuous feed	Unlimited		
	Position override	Possible	Change target-position before position completion	
Basic specifications	■ Speed control			
	Speed range	0.1 pps to 6.6 Mpps (x0.1 to x66.6) [Speed register range: 1 to 100000 (17-bit)] As for encoder speed: 4 Mcps Max. (when x4, differential input)	0.1 time mode: 0.1 to 10kpps 1time mode: 1 to 100kpps 10times mode: 10 to 1Mpps 50 times mode: 50 to 5Mpps	
	Speed override	Possible		
	■ Acceleration/deceleration control Automatic acceleration/deceleration system	S-curve acceleration/deceleration, partial s-curve acceleration/deceleration, and linear acceleration/deceleration (All with triangular drive peak prevention function) During automatic acceleration/deceleration system: Asymmetrically sloped acceleration/deceleration possible	Acceleration/deceleration range during the slope of acceleration/deceleration Linear acce/dece: 0.5ms to 261s S-curve acce/dece: 1ms to 522s	
Function specifications	■ Origin-returning control Origin-returning method Origin search Getting out of origin area	13 ways of returning to a sensor's origin, Z-phase origin, or ELS shared origin Available Available		
	■ Counter function	Counter 1: command position (command pulse count) [length 28-bit] Counter 2: machine position (encoder count) [length 28-bit] Counter 3: general-purpose/error counter [length 17-bit]		
	■ Comparators	Three comparators are available. These can be used for any counter above. CMP 3 has its output at the machine I/F connector.		
	■ Synchronizing signal output (CMP3 output)	With CMP3 and counter 3, CMP3 terminal output is possible at regular intervals.		
	■ Encoder input / pulsar input	Encoder input and pulsar input are directed to the same connector's terminal. Encoder output refers to differential output. (For input speed, see Basic specifications Speed range.)		
	■ Backlash compensation	Compensation pulses are inserted immediately before command operation for every turn of the movement direction. Counter during compensation can be set as either valid or invalid.		
	■ Vibration suppression at stop	Effective to suppress vibrations caused by stepping motor stop.		
	■ Machine interface ■ Servo interface	± ELS, D, DLS (All insulated with couplers), encoder A, B, Z-phase Command-pulse output (differential) Servo alarm, in-position, servo ready inputs Servo reset, servo ON, clear-servo-error-counter outputs (all insulated with couplers)		
Safety feature	■ Communication cable disconnection detection	Upon detection, motion control is reset (when power ON).	Communication settings switch (when the 2nd ON)	
Environmental conditions	■ Supply power	DC12V to 25V (power supply for external I/O provided separately)		
	■ Consumption current	Model number	Consumption current in the circuit, as a reference (Included in communication board 24 V power supply for calculating the amount of power supply)	Consumption current at the module 24V power supply terminal (Consumed at servo I/F and machine I/F, as a precondition)
		HMG-P1	0.53 [A]MAX	0.8 [A] MAX
		HMG-P2	0.74 [A]MAX	1.6 [A] MAX
		HMG-P3	0.96 [A]MAX	2.4 [A] MAX
		HMG-P4	1.16 [A]MAX	3.2 [A] MAX
		HMG-P5	1.37 [A]MAX	4.0 [A] MAX
HMG-P6	1.58 [A]MAX	4.8 [A] MAX		
■ Ambient temperature	0 to 50 °C (no condensation)			
■ Module dimensions	Motion module: P100C Width 85 mm × Length 105 mm			

Table 4. 5-2 HMG-Px motion module specifications

4.6 DIO slave specifications

A DIO slave consists of a communication board and one to four 16-in/16-out DIO modules being stacked.

Module type	Model number	Name
DIO module	hm-dio32c	DIO module

Figure 4. 6-1 DIO module model

See “HMG Slave User’s Manual” for details on DIO slaves.

4.6.1 DIO slave configuration

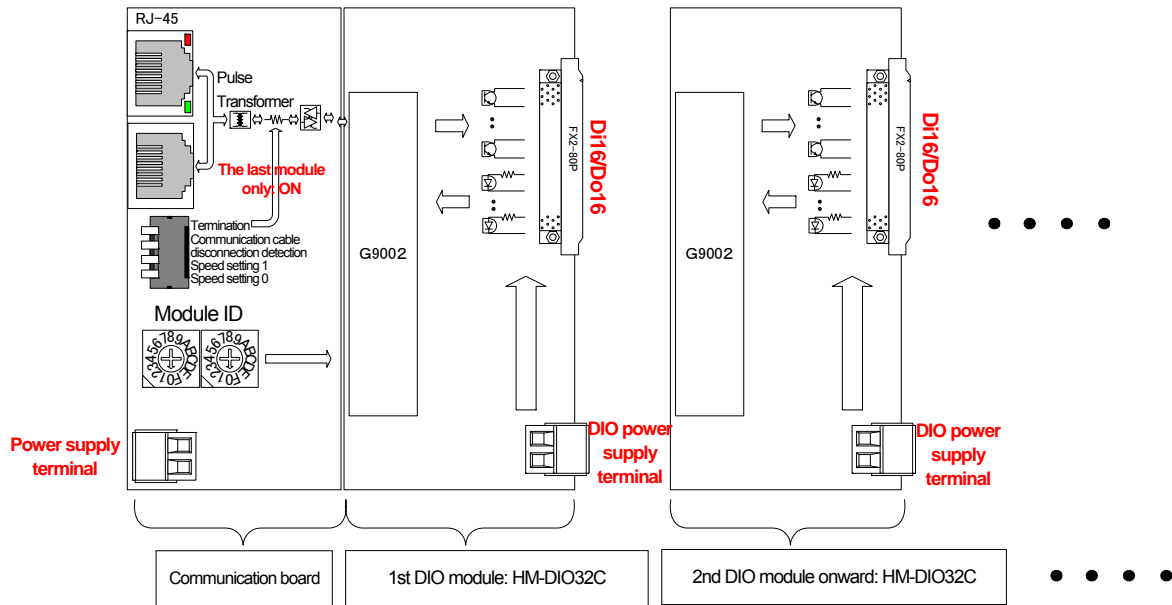


Figure 4. 6-1 DIO slave configuration

4.6.2 HMG-DIOx module specifications (x: 1 to 4)

Category	Item	Specification	Remarks	
Basic specifications	■ DI specifications	Input count	16	Control LSI: G9002 (by Nippon Pulse Motor)
		Input form	Photo-coupler isolation input (TLP280-4 equivalent used)	
		Rated input voltage	DC12V to DC24V	
		Employed input voltage range	DC10V to DC26V	
		Rated input current	8mA	
		Input resistance	3.0kΩ	
		Response time	1 msec or less	
		Filter	With about 1-msec integrating circuit	
	Input count/Common count	16/common		
	■ DO specifications	Output count	16	Control LSI: G9002 (by Nippon Pulse Motor)
		Output form	Photo-coupler isolation, Open collector / transistor array output (TD62084AP equivalent used)	
		Rated load voltage	DC12V to DC24V	
		Recommended load current	10 mA or less	
		Maximum load current	80 mA (8 units in total: 500 mA or less)	
Output ON residual voltage		0.5V or less (40 mA or less output current) 1.0V or less (80 mA or less output current)		
Response time		50μsec or less		
Input count/Common count		16/common		
Output logic	Output transistor ON when internal logic '1'			
Safety feature	■ Communication cable disconnection detection	Upon detection, the current output port status is maintained.		
Environmental conditions	■ Supply power	DC12V to DC24V		
	■ Consumption current	Model	Circuit consumption current	DO consumption current
		HMG-DIO1	560mA MAX	1.5A MAX
		HMG-DIO2	700mA MAX	3.0A MAX
		HMG-DIO3	830mA MAX	4.5A MAX
HMG-DIO4	960mA MAX	6.0A MAX		
■ Ambient temperature	0 to 50 °C (no condensation)			
■ Module dimensions	DIO module: DIO32C Width 85 mm × Length 105 mm			

Table 4. 6-2 HMG-DIO specifications

5. Center Device (G9001A)/ Option Ports

A master board is equipped with two center devices (G9001A) and a set of option ports to provide optional functions.

5.1 Center device (G9001A)

The center device (G9001A) communicates with local devices such as motion and DIO devices. This section describes statuses, commands, registers, etc. of the center device.

For information on management, see “7. Software” as well. Also, in the explanations of register bits in this section, “0” means that the value other than “0” must not be written, and that only “0” must be read.

5.1.1 Local device information (DINFO)

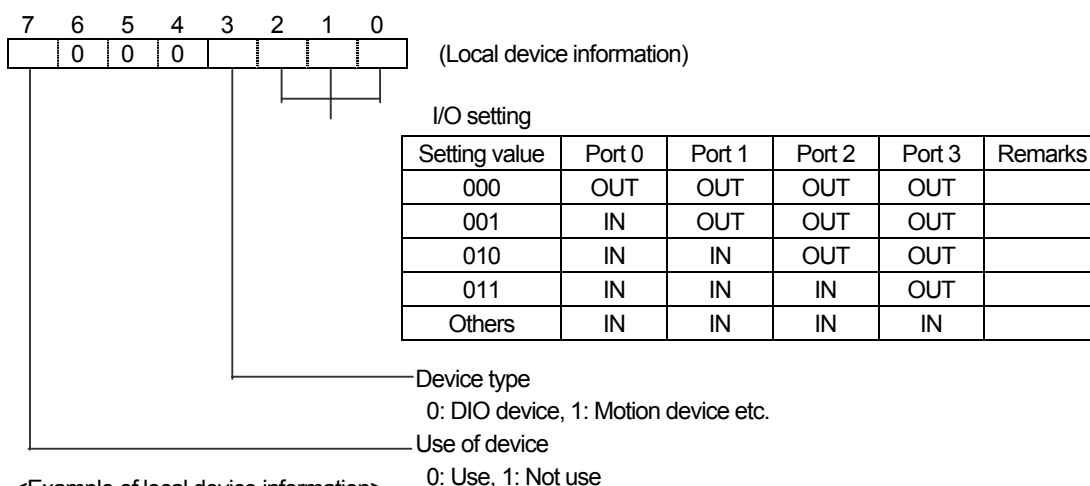
All the local devices (device number: 0 to 63) are polled sequentially through system communication. Responses from the local devices enable checking of the connection status, device type, and I/O port configuration, etc. of each local device so as to update “local device information.”

If you know “local device information” to be set in advance, it is possible to write the information through the application.

Use “mnt520_rLclInfo ()” to read local device information, and “mnt520_wLclInfo ()” to write data to local device information.

(For details on functions, see “7. Software.”)

Local device information is made up of 8 bits for each local device.



<Example of local device information>

82h for HM-DIO32C

8Bh for HM-P100C

Address	Read (INP)	Write (OUT)
BAR2 +78	Line 1 module ID=0 local device info	Line 1 module ID=0 local device info setting
+79	Line 1 module ID=1 local device info	Line 1 module ID=1 local device info setting
+80	Line 1 module ID=2 local device info	Line 1 module ID=2 local device info setting
-	-	-
+B5	Line 1 module ID=61 local device info	Line 1 module ID=61 local device info setting
+B6	Line 1 module ID=62 local device info	Line 1 module ID=62 local device info setting
+B7	Line 1 module ID=63 local device info	Line 1 module ID=63 local device info setting
,		
+278	Line 2 module ID=0 local device info	Line 2 module ID=0 local device info setting
+279	Line 2 module ID=1 local device info	Line 2 module ID=1 local device info setting
+280	Line 2 module ID=2 local device info	Line 2 module ID=2 local device info setting
-		
+2B5	Line 2 module ID=61 local device info	Line 2 module ID=61 local device info setting
+2B6	Line 2 module ID=62 local device info	Line 2 module ID=62 local device info setting
+2B7	Line 2 module ID=63 local device info	Line 2 module ID=63 local device info setting

Table 5. 1-1 Local device information addresses

5.1.2 Cyclic Communication Error Flag (IOERR)

Communication with the I/O ports of local devices are always carried out by cyclic communication.

If three consecutive communication errors occur for the local device of the same module ID, it is handled as a cyclic communication error. (A disconnection is also handled as a cyclic communication error.)

In response to an error, the corresponding particular bit of the Cyclic Communication Error Flag is turned to “1” according to the rules described below.

Checking this allows you to identify the local device that caused the error.

To turn the “1” bit caused by the error back to “0,” write “1” in that bit.

One way to reset the Flag is writing the read “Cyclic Communication Error Flag” data, as it is, to the Cyclic Communication Error Flag reset.

Use “mnt520_rLclCycErr()” to read a Cyclic Communication Error Flag, and “mnt520_wLclCycErr()” to write data to a Cyclic Communication Error Flag reset.

(For details on functions, see “7.6.3 Details on driver functions.”)

<Bit configuration>

Read 0: Absence of error 1: Presence of error

Write 0: Ignore 1: Reset Flag

Reading “BAR2 + 0B8h” allows you to check module IDs 15 to 0 on Line 1.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0

Reading “BAR2 + 2BAh” allows you to check module IDs 31 to 16 on Line 2.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID31	ID30	ID29	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16

As shown above, the lowest bit shows the error status of the local device with the lowest module ID.

The address of a Cyclic Communication Error Flag is determined in the following way. (all digits to the right of the decimal point discarded)

(1) For Line 1:

Address = BAR2 + 0B8h + (module ID/8)

(2) For Line 2:

Address = BAR2 + 2B8h + (module ID/8)

Address (HEX)	Read (INP)	Write (OUT)
BAR2 +B8	Line 1 module ID 15-0 Cyclic Communication Error Flag	Line 1 module ID 15-0 Cyclic Communication Error Flag reset
+BA	Line 1 module ID 31-16 Cyclic Communication Error Flag	Line 1 module ID 31-16 Cyclic Communication Error Flag reset
+BC	Line 1 module ID 47-32 Cyclic Communication Error Flag	Line 1 module ID 47-32 Cyclic Communication Error Flag reset
+BE	Line 1 module ID 63-48 Cyclic Communication Error Flag	Line 1 module ID 63-48 Cyclic Communication Error Flag reset
+2B8	Line 2 module ID 15-0 Cyclic Communication Error Flag	Line 2 module ID 15-0 Cyclic Communication Error Flag reset
+2BA	Line 2 module ID 31-16 Cyclic Communication Error Flag	Line 2 module ID 31-16 Cyclic Communication Error Flag reset
+2BC	Line 2 module ID 47-32 Cyclic Communication Error Flag	Line 2 module ID 47-32 Cyclic Communication Error Flag reset
+2BE	Line 2 module ID 63-48 Cyclic Communication Error Flag	Line 2 module ID 63-48 Cyclic Communication Error Flag reset

Table 5. 1-2 Cyclic Communication Error Flag addresses

5.1.3 Input-Port Change Flag Setting (IINT)

It is possible to automatically obtain port information regarding the connected DIO devices through cyclic communication.

Also, the center device regularly obtains device status information on the motion devices etc. through cyclic communication.

To detect such an input port change and a status change of motion devices, an Input-Port Change Flag is used.

To set an Input-Port Change Flag, turn to "1" the bit corresponding to the local device number whose status change you want to monitor.

Use "mnt520_rLclSetInt()" to read an Input-Port Change Flag Setting status.

Use "mnt520_wLclSetInt()" to write data to an Input-Port Change Flag Setting.

(For details on functions, see "8. Function References.")

<Bit configuration>

Read 0: Absence of the Input-Port Change Flag Setting 1: Presence of the Input-Port Change Flag Setting

Write 0: Do not monitor input port changes 1: Monitor input port changes

Reading "BAR2 + 0C0h" allows you to check the Input-Port Change Flag Setting status of the module IDs 3 to 0 on Line 1.

Writing to "BAR2 + 0C0h" allows you to set the Input-Port Change Flag for module IDs 3 to 0 on Line 1.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Module ID=3				Module ID=2				Module ID=1				Module ID=0			
Port 3	Port 2	Port 1	Port 0	Port 3	Port 2	Port 1	Port 0	Port 3	Port 2	Port 1	Port 0	Port 3	Port 2	Port 1	Port 0

Reading "BAR2 + 2C2h" allows you to check the Input-Port Change Flag Setting status of the module IDs 7 to 4 on Line 2.

Writing in "BAR2 + 2C2h" allows you to set the Input-Port Change Flag for module IDs 7 to 4 on Line 2.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Module ID=7				Module ID=6				Module ID=5				Module ID=4			
Port 3	Port 2	Port 1	Port 0	Port 3	Port 2	Port 1	Port 0	Port 3	Port 2	Port 1	Port 0	Port 3	Port 2	Port 1	Port 0

As shown above, one local device corresponds to four bits. The lowest four bits are used as the Input-Port Change Flag Setting area for the local device with the lowest module ID.

The lowest bit among those four bits corresponds to port 0; the other bits to the rest of the ports in order, up to port 3.

For other local devices, determine their addresses according to the rule shown below.

(all digits to the right of the decimal point discarded)

(1) For Line 1:

Address = 0C0h + (module ID/2)

(2) For Line 2:

Address = 2C0h + (module ID/2)

Address	Read (INP)	Write (OUT)
BAR2+C0	Line 1 module ID 3-0 Input-Port Change Flag Setting status	Line 1 module ID 3-0 Input-Port Change Flag Setting
+C2	Line 1 module ID 7-4 Input-Port Change Flag Setting status	Line 1 module ID 7-4 Input-Port Change Flag Setting
+C4	Line 1 module ID 11-8 Input-Port Change Flag Setting status	Line 1 module ID 11-8 Input-Port Change Flag Setting
+C6	Line 1 module ID 15-12 Input-Port Change Flag Setting status	Line 1 module ID 15-12 Input-Port Change Flag Setting
+C8	Line 1 module ID 19-16 Input-Port Change Flag Setting status	Line 1 module ID 19-16 Input-Port Change Flag Setting
+CA	Line 1 module ID 23-20 Input-Port Change Flag Setting status	Line 1 module ID 23-20 Input-Port Change Flag Setting
+CC	Line 1 module ID 27-24 Input-Port Change Flag Setting status	Line 1 module ID 27-24 Input-Port Change Flag Setting
+CE	Line 1 module ID 31-24 Input-Port Change Flag Setting status	Line 1 module ID 31-24 Input-Port Change Flag Setting
+D0	Line 1 module ID 35-32 Input-Port Change Flag Setting status	Line 1 module ID 35-32 Input-Port Change Flag Setting
+D2	Line 1 module ID 39-36 Input-Port Change Flag Setting status	Line 1 module ID 39-36 Input-Port Change Flag Setting
+D4	Line 1 module ID 43-40 Input-Port Change Flag Setting status	Line 1 module ID 43-40 Input-Port Change Flag Setting
+D6	Line 1 module ID 47-44 Input-Port Change Flag Setting status	Line 1 module ID 47-44 Input-Port Change Flag Setting
+D8	Line 1 module ID 51-48 Input-Port Change Flag Setting status	Line 1 module ID 51-48 Input-Port Change Flag Setting
+DA	Line 1 module ID 55-52 Input-Port Change Flag Setting status	Line 1 module ID 55-52 Input-Port Change Flag Setting
+DC	Line 1 module ID 59-56 Input-Port Change Flag Setting status	Line 1 module ID 59-56 Input-Port Change Flag Setting
+DE	Line 1 module ID 63-60 Input-Port Change Flag Setting status	Line 1 module ID 63-60 Input-Port Change Flag Setting
+2C0	Line 2 module ID 3-0 Input-Port Change Flag Setting status	Line 2 module ID 3-0 Input-Port Change Flag Setting
+2C2	Line 2 module ID 7-4 Input-Port Change Flag Setting status	Line 2 module ID 7-4 Input-Port Change Flag Setting
+2C4	Line 2 module ID 11-8 Input-Port Change Flag Setting status	Line 2 module ID 11-8 Input-Port Change Flag Setting
+2C6	Line 2 module ID 15-12 Input-Port Change Flag Setting status	Line 2 module ID 15-12 Input-Port Change Flag Setting
+2C8	Line 2 module ID 19-16 Input-Port Change Flag Setting status	Line 2 module ID 19-16 Input-Port Change Flag Setting
+2CA	Line 2 module ID 23-20 Input-Port Change Flag Setting status	Line 2 module ID 23-20 Input-Port Change Flag Setting
+2CC	Line 2 module ID 27-24 Input-Port Change Flag Setting status	Line 2 module ID 27-24 Input-Port Change Flag Setting
+2CE	Line 2 module ID 31-24 Input-Port Change Flag Setting status	Line 2 module ID 31-24 Input-Port Change Flag Setting
+2D0	Line 2 module ID 35-32 Input-Port Change Flag Setting status	Line 2 module ID 35-32 Input-Port Change Flag Setting
+2D2	Line 2 module ID 39-36 Input-Port Change Flag Setting status	Line 2 module ID 39-36 Input-Port Change Flag Setting
+2D4	Line 2 module ID 43-40 Input-Port Change Flag Setting status	Line 2 module ID 43-40 Input-Port Change Flag Setting
+2D6	Line 2 module ID 47-44 Input-Port Change Flag Setting status	Line 2 module ID 47-44 Input-Port Change Flag Setting
+2D8	Line 2 module ID 51-48 Input-Port Change Flag Setting status	Line 2 module ID 51-48 Input-Port Change Flag Setting
+2DA	Line 2 module ID 55-52 Input-Port Change Flag Setting status	Line 2 module ID 55-52 Input-Port Change Flag Setting
+2DC	Line 2 module ID 59-56 Input-Port Change Flag Setting status	Line 2 module ID 59-56 Input-Port Change Flag Setting
+2DE	Line 2 module ID 63-60 Input-Port Change Flag Setting status	Line 2 module ID 63-60 Input-Port Change Flag Setting

Table 5. 1-3 Input-Port Change Flag Setting addresses

5.1.4 Input-Port Change Flag (IRES)

If any change occurs in a port where an Input-Port Change Flag has been set, the corresponding bit for the Input-Port Change Flag is turned to "1."

Monitoring Input-Port Change Flags allows you to know in which local device and port number a status change occurs.

Use "mnt520_rLclInt()" to read an Input-Port Change Flag.

Use "mnt520_wLclInt()" to write data to an Input-Port Change Flag reset.

(For details on functions, see "8. Function References.")

<Bit configuration>

Read 0: Absence of an input port change 1: Presence of an input port change

Write 0: Ignore 1: Reset Input-Port Change Flag

Reading "BAR2 + 0E0h" allows reading of the Input-Port Change Flags for the module IDs 3 to 0 on Line 1.

Writing data to "BAR2 + 0E0h" allows resetting the Input-Port Change Flags for module IDs 3 to 0 on Line 1.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Module ID=3				Module ID=2				Module ID=1				Module ID=0			
Port 3	Port 2	Port 1	Port 0	Port 3	Port 2	Port 1	Port 0	Port 3	Port 2	Port 1	Port 0	Port 3	Port 2	Port 1	Port 0

Reading "BAR2 + 0E2h" allows reading of the Input-Port Change Flags for the module IDs 7 to 4 on Line 2.

Writing the value to "BAR2 + 0E2h" allows resetting the Input-Port Change Flags for module IDs 7 to 4 on Line 2.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Module ID=7				Module ID=6				Module ID=5				Module ID=4			
Port 3	Port 2	Port 1	Port 0	Port 3	Port 2	Port 1	Port 0	Port 3	Port 2	Port 1	Port 0	Port 3	Port 2	Port 1	Port 0

As shown above, one local device corresponds to four bits. The lowest four bits are used as the Input-Port Change Flag data area for the local device with the lowest module device ID.

The lowest bit among those four bits corresponds to port 0; the other bits to the rest of the ports in order, up to port 3.

For other local devices, determine their addresses according to the rule shown below.

(all digits to the right of the decimal point discarded)

(1) For Line 1:

Address = 0E0h + (module ID/2)

(2) For Line 2:

Address = 2E0h + (module ID/2)

One way to reset a Flag is writing the read "Input-Port Change Flag" data, as it is, to the Input-Port Change Flag reset.

Address	Read (INP)	Write (OUT)
BAR2 +E0	Line 1 Module ID 3-0 Input-Port Change Flag	Line 1 Module ID 3-0 Input-Port Change Flag reset
+E2	Line 1 Module ID 7-4 Input-Port Change Flag	Line 1 Module ID 7-4 Input-Port Change Flag reset
+E4	Line 1 Module ID 11-8 Input-Port Change Flag	Line 1 Module ID 11-8 Input-Port Change Flag reset
+E6	Line 1 Module ID 15-12 Input-Port Change Flag	Line 1 Module ID 15-12 Input-Port Change Flag reset
+E8	Line 1 Module ID 19-16 Input-Port Change Flag	Line 1 Module ID 19-16 Input-Port Change Flag reset
+EA	Line 1 Module ID 23-20 Input-Port Change Flag	Line 1 Module ID 23-20 Input-Port Change Flag reset
+EC	Line 1 Module ID 27-24 Input-Port Change Flag	Line 1 Module ID 27-24 Input-Port Change Flag reset
+EE	Line 1 Module ID 31-24 Input-Port Change Flag	Line 1 Module ID 31-24 Input-Port Change Flag reset
+F0	Line 1 Module ID 35-32 Input-Port Change Flag	Line 1 Module ID 35-32 Input-Port Change Flag reset
+F2	Line 1 Module ID 39-36 Input-Port Change Flag	Line 1 Module ID 39-36 Input-Port Change Flag reset
+F4	Line 1 Module ID 43-40 Input-Port Change Flag	Line 1 Module ID 43-40 Input-Port Change Flag reset
+F6	Line 1 Module ID 47-44 Input-Port Change Flag	Line 1 Module ID 47-44 Input-Port Change Flag reset
+F8	Line 1 Module ID 51-48 Input-Port Change Flag	Line 1 Module ID 51-48 Input-Port Change Flag reset
+FA	Line 1 Module ID 55-52 Input-Port Change Flag	Line 1 Module ID 55-52 Input-Port Change Flag reset
+FC	Line 1 Module ID 59-56 Input-Port Change Flag	Line 1 Module ID 59-56 Input-Port Change Flag reset
+FE	Line 1 Module ID 63-60 Input-Port Change Flag	Line 1 Module ID 63-60 Input-Port Change Flag reset
BAR2+2E0	Line 2 Module ID 3-0 Input-Port Change Flag	Line 2 Module ID 3-0 Input-Port Change Flag reset
+2E2	Line 2 Module ID 7-4 Input-Port Change Flag	Line 2 Module ID 7-4 Input-Port Change Flag reset
+2E4	Line 2 Module ID 11-8 Input-Port Change Flag	Line 2 Module ID 11-8 Input-Port Change Flag reset
+2E6	Line 2 Module ID 15-12 Input-Port Change Flag	Line 2 Module ID 15-12 Input-Port Change Flag reset
+2E8	Line 2 Module ID 19-16 Input-Port Change Flag	Line 2 Module ID 19-16 Input-Port Change Flag reset
+2EA	Line 2 Module ID 23-20 Input-Port Change Flag	Line 2 Module ID 23-20 Input-Port Change Flag reset
+2EC	Line 2 Module ID 27-24 Input-Port Change Flag	Line 2 Module ID 27-24 Input-Port Change Flag reset
+2EE	Line 2 Module ID 31-24 Input-Port Change Flag	Line 2 Module ID 31-24 Input-Port Change Flag reset
+2F0	Line 2 Module ID 35-32 Input-Port Change Flag	Line 2 Module ID 35-32 Input-Port Change Flag reset
+2F2	Line 2 Module ID 39-36 Input-Port Change Flag	Line 2 Module ID 39-36 Input-Port Change Flag reset
+2F4	Line 2 Module ID 43-40 Input-Port Change Flag	Line 2 Module ID 43-40 Input-Port Change Flag reset
+2F6	Line 2 Module ID 47-44 Input-Port Change Flag	Line 2 Module ID 47-44 Input-Port Change Flag reset
+2F8	Line 2 Module ID 51-48 Input-Port Change Flag	Line 2 Module ID 51-48 Input-Port Change Flag reset
+2FA	Line 2 Module ID 55-52 Input-Port Change Flag	Line 2 Module ID 55-52 Input-Port Change Flag reset
+2FC	Line 2 Module ID 59-56 Input-Port Change Flag	Line 2 Module ID 59-56 Input-Port Change Flag reset
+2FE	Line 2 Module ID 63-60 Input-Port Change Flag	Line 2 Module ID 63-60 Input-Port Change Flag reset

Table 5. 1-4 Input-Port Change Flag addresses

5.1.5 Port Data (DATA)

Reading from Port Data is used for checking data from the input ports of DIO devices, or the motion main statuses as well as the generic output-port statuses of motion devices.

Writing to Port Data is used for data setting for the output ports of DIO devices, or for the generic output ports of motion devices.

If the device on Line 1—whose module ID is 0 as an example—is a DIO device, reading “BAR2+100h” allows you to check the data from the generic ports.

In a similar manner, reading “BAR2+102h” allows you to check generic output statuses, and writing to it allows generic output setting.

To input/output data to/from DIO devices, use the following functions:

mnt520_rIoPortB() ... Reads 1 byte from the specified ports in a DIO device.

mnt520_wIoPortB() ... Writes 1 byte to the specified ports in a DIO device.

mnt520_rIoPortW() ... Reads 2 bytes from the specified ports in a DIO device.

mnt520_wIoPortW() ... Writes 2 bytes to the specified ports in a DIO device.

For motion devices, use the following functions:

mnt520_rPclMsts() Checks the motion main statuses.

mnt520_rPclPort() Reads the generic output port statuses.

mnt520_wPclPort() Sets data to the generic output ports.

The address map below shows module IDs and port numbers for access.

Address	Read (INP) Port Data (DATA)	Write (OUT) Port Data (DATA)
BAR2+100	Line 1 Module ID 0 port 1, 0 data	Line 1 Module ID 0 port 1, 0 setting
+102	Line 1 Module ID 0 port 3, 2 data	Line 1 Module ID 0 port 3, 2 setting
+104	Line 1 Module ID 1 port 1, 0 data	Line 1 Module ID 1 port 1, 0 setting
+106	Line 1 Module ID 1 port 3, 2 data	Line 1 Module ID 1 port 3, 2 setting
+108	Line 1 Module ID 2 port 1, 0 data	Line 1 Module ID 2 port 1, 0 setting
+10A	Line 1 Module ID 2 port 3, 2 data	Line 1 Module ID 3 port 3, 2 setting
,		
+1FC	Line 1 Module ID 63 port 1, 0 data	Line 1 Module ID 63 port 1, 0 setting
+1FE	Line 1 Module ID 63 port 3, 2 data	Line 1 Module ID 63 port 3, 2 setting
,		
+300	Line 2 Module ID 0 port 1, 0 data	Line 2 Module ID 0 port 1, 0 setting
+302	Line 2 Module ID 0 port 3, 2 data	Line 2 Module ID 0 port 3, 2 setting
+304	Line 2 Module ID 1 port 1, 0 data	Line 2 Module ID 1 port 1, 0 setting
+306	Line 2 Module ID 1 port 3, 2 data	Line 2 Module ID 1 port 3, 2 setting
+308	Line 2 Module ID 2 port 1, 0 data	Line 2 Module ID 2 port 1, 0 setting
+30A	Line 2 Module ID 2 port 3, 2 data	Line 2 Module ID 3 port 3, 2 setting
,		
+3FC	Line 2 Module ID 63 port 1, 0 data	Line 2 Module ID 63 port 1, 0 setting
+3FE	Line 2 Module ID 63 port 3, 2 data	Line 2 Module ID 63 port 3, 2 setting

Table 5. 1-5 Port Data addresses

5.1.6 Center Main Status (CMSTS)

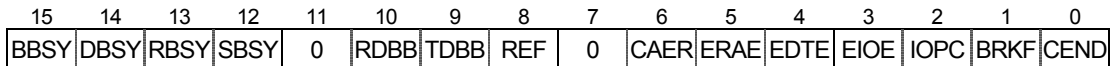


Figure 5. 1-1 Bit configuration of Center Main Status (CMSTS)

bit	Name	Description	Remarks
0	CEND	When writing to the data-sending FIFO is possible, CEND is turned to “1.” When it becomes possible to write the next data to the data-sending FIFO after a system or data communication is completed, CEND is turned to “1” and an interrupt signal (#INT) is output. How to clear this bit depends on the bit 9 status of the RENV0 register.	
1	BRKF	When receiving a brake frame, BRKF is turned to “1” and an interrupt signal is output. How to clear this bit depends on the bit 9 status of the RENV0 register.	
2	IOPC	When the status of an input port—whose “Interrupt-On-Input-Change Setting” has been set to “1”—is changed, IOPC is turned to “1” and an interrupt signal is output. That is a 256-bit OR signal of the Input-Port Change Flag. When all the bits are turned to “0”, the IOPC bit is returned to “0.”	
3	EIOE	When a cyclic communication error occurs, EIOE is turned to “1” and an interrupt signal is output. This is a 64-bit OR signal of the Cyclic Communication Error Flag. When all the bits are turned to “0”, the EIOE bit is returned to “0.”	
4	EDTE	When a data communication error occurs, EDTE is turned to “1” and an interrupt signal is output. How to clear this bit depends on the bit 9 status of the RENV0 register.	
5	ERAE Note 1	When an error occurs during receiving process by a local device, ERAE is turned to “1” and an interrupt signal is output. For the definition of errors, see “ERA” in “5.1.7 Center Interrupt Status.” ERA in Center Interrupt Status allows you to check in which local device the error has occurred and the contents of the error. How to clear this bit depends on the bit 9 status of the RENV0 register.	
6	CAER	This is for an application access error. When an improper access from the application occurs, such as writing a data transmission command with empty sending data, CAER is turned to “1” and an interrupt signal is output. For the definition of errors, see “CAE” in “5.1.7 Center Interrupt Status.” CAE allows you to check the contents of the error. How to clear this bit depends on the bit 9 status of the RENV0 register.	
7	(Undefined)	Always 0	
8	REF	“1” during the presence of unsent output-port data. When data is written to the output port area, REF is turned to “1.” Then, when a cyclic communication is performed for all the ports without more than one error, REF is returned to “0.”	
9	TDBB	“1” during the presence of send data in the data-sending FIFO. When data is written to the data-sending FIFO, TDBB is turned to “1.” Then, when a data transmission command or a sending-FIFO reset command is written, TDBB is returned to “0.”	
10	RDBB	“1” during the presence of received data in the data-receiving FIFO. When data from the center device is received, RDBB is turned to “1.” Then, when the application reads all the received data, RDBB is returned to “0.”	
11	(Undefined)	Always 0	
12	SBSY	Turns “1” while a cyclic communication is being started.	
13	RBSY	Turns “1” during reset process.	
14	DBSY	Turns “1” during system communication or data communication.	
15	BBSY	When the break communication command (0610h) is issued where RENV0 (8) is “1”, BBSY turns “1” and keeps it until the completion of the break communication. Other than this situation, BBSY remains “0.”	

Note 1. When errors occur in two or more local devices, ERA will contain information on the local device where the last error has occurred.

Table 5. 1-6 Contents of Center Main Status (MSTS)

5.1.7 Center Interrupt Status (CISTS)

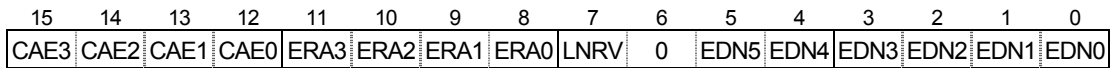


Figure 5. 1-2 Bit configuration of Center Interrupt Status (CISTS)

bit	Name	Description	Remarks										
5 to 0	EDN 5 to 0	Contains the module ID where an error—EDTE is 1 or ERAE is 1 in CMSTS—has occurred. The data is maintained until the next error.											
6	(Undefined)	Always 0											
7	LNRV	Turns “1” when data has not been received by the local device side. When data communication is ended by an error (EDTE = 1), LNRV is turned to “1” if the local device side has failed to receive data from the center device (no response from the local device side); and turned to “0” if the local device side has received data (namely when a communication sent from the local device side is crashed due to a communication failure, thus making it impossible for the center device to receive it successfully. Such a case requires checking to see whether data is received successfully by the local device side.) The LNRV value is maintained until the next error.											
11 to 8	ERA 3 to 0	ERA is used when the contents of a packet does not match the local device type despite successful data reception by the local device side. In such a case, code shown below is stored in the ERA bits and maintained until the next error. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="width: 15%;">Code</th> <th style="width: 85%;">Condition</th> </tr> </thead> <tbody> <tr> <td>0001</td> <td>When the I/O setting information in the “device information area” in the center device differs from the I/O port setting of the local device side.</td> </tr> <tr> <td>0010</td> <td>When a DIO device receives data communication.</td> </tr> <tr> <td>0011</td> <td>When a motion device receives a data communication whose size is above the receive buffer capacity (6 bytes) of the device. When another type of device receives data whose size is above the receive buffer capacity of the device.</td> </tr> </tbody> </table>	Code	Condition	0001	When the I/O setting information in the “device information area” in the center device differs from the I/O port setting of the local device side.	0010	When a DIO device receives data communication.	0011	When a motion device receives a data communication whose size is above the receive buffer capacity (6 bytes) of the device. When another type of device receives data whose size is above the receive buffer capacity of the device.			
Code	Condition												
0001	When the I/O setting information in the “device information area” in the center device differs from the I/O port setting of the local device side.												
0010	When a DIO device receives data communication.												
0011	When a motion device receives a data communication whose size is above the receive buffer capacity (6 bytes) of the device. When another type of device receives data whose size is above the receive buffer capacity of the device.												
15 to 12	CAE 3 to 0	Code shown below is stored in the CAE bits when the application performs an incorrect access to G9001A. The stored code will be maintained until the next error. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="width: 15%;">Code</th> <th style="width: 85%;">Condition</th> </tr> </thead> <tbody> <tr> <td>0001</td> <td>When the Cyclic Communication Start command is written without any local device to be used.</td> </tr> <tr> <td>0010</td> <td>When the Data Transmission Start command is written without send data set in the sending FIFO.</td> </tr> <tr> <td>0011</td> <td>When the following access is attempted where DBSY is 1: (1) Reading from the receiving FIFO, or writing to the sending FIFO. (2) Writing the System or Data Transmission Start command</td> </tr> <tr> <td>0100</td> <td>When data communication is performed toward a local device, for which the device information area (bit 7) is “0” (namely the device is set in unused status).</td> </tr> </tbody> </table>	Code	Condition	0001	When the Cyclic Communication Start command is written without any local device to be used.	0010	When the Data Transmission Start command is written without send data set in the sending FIFO.	0011	When the following access is attempted where DBSY is 1: (1) Reading from the receiving FIFO, or writing to the sending FIFO. (2) Writing the System or Data Transmission Start command	0100	When data communication is performed toward a local device, for which the device information area (bit 7) is “0” (namely the device is set in unused status).	
Code	Condition												
0001	When the Cyclic Communication Start command is written without any local device to be used.												
0010	When the Data Transmission Start command is written without send data set in the sending FIFO.												
0011	When the following access is attempted where DBSY is 1: (1) Reading from the receiving FIFO, or writing to the sending FIFO. (2) Writing the System or Data Transmission Start command												
0100	When data communication is performed toward a local device, for which the device information area (bit 7) is “0” (namely the device is set in unused status).												

Table 5. 1-7 Contents of Center Interrupt Status (CISTS)

5.1.8 Center Device commands (CCMD)

(1) Center Operation commands

Command type	Command	Description	Remarks																
Invalid command	0000 0000 0000 0000 (0000h)	Invalid command																	
Software reset command	0000 0001 0000 0000 (0100h)	Resets the center device (G9001A). Wait at least 100 μsec after issuing this command. Statuses after reset: Command 0000h Center main status 0000h Center interrupt status 0000h I/O buffer 0000h Data-sending FIFO Undefined Data-receiving FIFO Undefined Device information All 00h Cyclic communication error flag All 00h Input-port change flag setting All 00h Input-port change flag All 00h Port data All 00h Register etc. RVER=0001h All others 0000h																	
Sending FIFO reset command	0000 0010 0000 0000 (0200h)	Resets only the data-sending FIFO.																	
Receiving FIFO reset command	0000 0011 0000 0000 (0300h)	Resets only the data-receiving FIFO.																	
Clear-center-interrupt-status command	0000 0100 0\$\$\$ 00\$\$ (04xxh)	Bits 0, 1, 4, 5, and 6 of the command are used as follows: <table border="1" style="margin-left: 20px;"> <tr> <td style="text-align: center;">7</td> <td style="text-align: center;">6</td> <td style="text-align: center;">5</td> <td style="text-align: center;">4</td> <td style="text-align: center;">3</td> <td style="text-align: center;">2</td> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">CAER</td> <td style="text-align: center;">ERAE</td> <td style="text-align: center;">EDTE</td> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td style="text-align: center;">BRKF</td> <td style="text-align: center;">CEND</td> </tr> </table> Turning a bit to "1" causes the corresponding status to be cleared. However, the command is invalid if RENV0 (9) is "0."	7	6	5	4	3	2	1	0	0	CAER	ERAE	EDTE	0	0	BRKF	CEND	
7	6	5	4	3	2	1	0												
0	CAER	ERAE	EDTE	0	0	BRKF	CEND												
Clear-error-counter command	0000 0110 0000 0000 (0600h)	Clears the error count register.																	
Break communication command	0000 0110 0001 0000 (0610h)	When RENV0 (8) is set to "1" that disables auto-break function, this command allows you to issue a break communication at any timing. This command is invalid if RENV0 (8) is "0."																	
System communication command to all devices	0001 0000 0000 0000 (1000h)	This command polls all the devices (module IDs 0 to 63) sequentially to update the "local device information" area corresponding to each module ID. The "local device information" contains the following information: (1) Use of the local device: 0 for the absence of response, 1 for the presence of response (2) Type of the local device: 1 when the type is the motion device (3) Information on the I/O port configuration of the local device The lack of response to a system communication is not handled as an error. However, there may be a CRC mismatch error.																	
System communication to all local devices excluded from cyclic communication	0001 0001 0000 0000 (1100h)	This command sequentially polls all the local devices, for which the "use of the local device" bit of the "local device information" is 0. This enables updating the "local device information" area corresponding to each local device number. The updated contents are the same as those by the command 1000h. The lack of response to a system communication is not handled as an error. However, there may be a CRC mismatch error.																	
System communication to specified local devices	0001 0010 00## ##### (1200h to 123Fh)	This command polls only the specified local devices so as to update the "local device information" area corresponding to each module ID. The updated contents are the same as those by the command 1000h. The lack of response to a system communication is not handled as an error. However, CRC mismatch could cause an error.																	

(To be continued)

(Continued from previous page)

Command type	Command	Description	Remarks
Obtaining attribute information of specified local devices	0001 0011 00## ##### (1300h to 133Fh)	A response frame caused by polling contains local device attribute information. This command polls the specified local devices to copy the attribute information to the data-receiving FIFO. The "local device information" area is not changed. ■ The data-receiving FIFO contains the following information: Bit 4 to 0: (max data byte count)/8-1 Bit 7 to 5: Unused (undefined) Bit 15 to 8: Model code (DIO device: 01h, motion device: 81h) Bit 18 to 16: I/O port setting Bit 19: 0 fixed Bit 31 to 20: Motion device: Other local devices:	
Command transmission to the specified group	0010 0ggg cccc cccc (2*01h) (2*02h) (2*04h)	Transmits the command to the data devices belonging to the specified group. The setting "ggg = 000" indicates all the groups. cccc cccc 0000 0001(01h): Start 0000 0010(02h): Stop 0000 0100(04h): Software reset	
Starting cyclic communication	0011 0000 0000 0000 (3000h)	This command starts cyclic communication targeting the devices for which the "use of local device" bit of the "local device information" is 1.	
Quitting cyclic communication	0011 0001 0000 0000 (3100h)	Quits cyclic communication.	
Data communication	0100 0000 00## ##### (4000h to 403Fh)	Transmits data in the sending FIFO to the specified local device. The response data is stored in the receiving FIFO.	
Canceling data communication	0100 0001 0000 0000 (4100h)	Cancel data communication and resets the sending FIFO. This command is invalid after completing data transmission.	

Note 1. # bits: The higher-order bits of a module ID are set to the # bits sequentially from the top.
& bits: "0" is set for port 0 and 1, "1" for port 2 and 3.
\$ bits: Either value can be set.
* bits: Either value can be set.

Table 5. 1-8 Contents of Center Operation commands

(2) Center Register Control commands

The registers held by the center device (G9001A) are not assigned on the address map. Therefore, it is necessary to use Center Register Control commands to access the registers.

Command type	Command	Description	Remarks
RENVO Write command	0101 0101 0000 0000 (5500h)	When data is set to the I/O buffer and this command is issued, the value in the I/O buffer is copied to the RENVO register.	
RENVO Read command	0110 0101 0000 0000 (6500h)	Issuing this command causes the value in the RENVO register to be copied to the I/O buffer. Reading the I/O buffer enables the acquisition of the RENVO value.	
Error-Counter Read command	0110 0101 0000 0001 (6501h)	Issuing this command causes the value in the error counter register to be copied to the I/O buffer. Reading the I/O buffer enables the acquisition of the error counter value.	
Cyclic-Periodicity Read Register command	0110 0101 0000 0010 (6502h)	Issuing this command causes the value in the cyclic-periodicity register to be copied to the I/O buffer. Reading the I/O buffer enables the acquisition of the cyclic periodicity register value.	
Receiving-Address Read Register command	0110 0101 0000 0011 (6503h)	Issuing this command causes the value in the receiving-address register to be copied to the I/O buffer. Reading the I/O buffer enables the acquisition of the receiving-address register value.	
Version-Info Read Register command	0110 0101 0000 0100 (6504h)	Issuing this command causes the value in the version-info register to be copied to the I/O buffer. Reading the I/O buffer enables the acquisition of the version-info register value.	

Table 5. 1-9 Contents of Center Register Control commands

(3) Reading and writing from/to the center register

■ How to read

- (1) Write the desired register read command to the target center device (line number).
- (2) Read the I/O buffer of the target center device (line number).

<Example. Reading the error counter of #1G9001A (Line 1)>

```
DWORD hDev; // Master board handle (Already obtained in this example.)
WORD err_cnt; // Error counter
```

```
mnt520_wCenCmd( hDev, 0, 0x6501 ); // Write Error-Counter Read command.
mnt520_rCenBuf( hDev, 0, &err_cnt ); // Read I/O buffer
```

■ How to write

- (1) Write data to the I/O buffer of the target center device (line number).
- (2) Write the desired register writing command to the target center device (line number).

<Example. Writing data to RENV0 of #1G9001A (Line 1)>

```
DWORD hDev; // Master board handle (This has already been obtained in this example.)
```

```
mnt520_wCenBuf( hDev, 0, 0x0010 ); // Write to I/O buffer (auto-break function disabled).
mnt520_wCenCmd( hDev, 0, 0x5500 ); // Write RENV0 command.
```

5.1.9 Center device register

(1) RENV0 register(Read/Write)

This 16-bit register is used for configuring the center device environment. The status immediately after reset is "0000h."

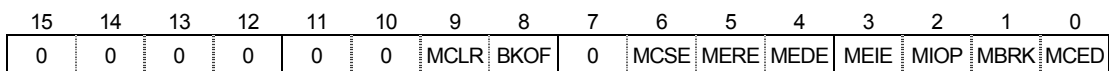


Figure 5. 1-3 Bit configuration of the RENV0 register

Name	Description	Remarks	Name
0	MCED	Masks CEND interrupt output by "1." The center interrupt status is changed.	Unnecessary interrupts are ignored.
1	MBRK	Masks BRKF interrupt output by "1." The center interrupt status is changed.	
2	MIOP	Masks IOPC interrupt output by "1." The center interrupt status is changed.	
3	MEIE	Masks EIOE interrupt output by "1." The center interrupt status is changed.	
4	MEDE	Masks EDTE interrupt output by "1." The center interrupt status is changed.	
5	MERE	Masks ERAE interrupt output by "1." The center interrupt status is changed.	
6	MCSE	Masks CAER interrupt output by "1." The center interrupt status is changed.	
7	(Undefined)	Always set "0."	
8	BKOF	Disables auto-break function by "1."	
9	MCLR	Defines how to clear the following status bits. 0: Cleared when status is read. (default) 1: Not cleared when status is read. To clear bits, use the command for clearing interrupt statuses (04xxh). Thus, it is possible to prevent bits from being cleared automatically on status reading.	
15 to 10	(Undefined)	Always set "0."	

Table 5. 1-10 Contents of the RENV0 register

(2) RERCNT (Read only)

This is a 16-bit register for counting error, which totals the number of such communication errors as no response and CRC errors.

When the error count reaches 65535 or more, the counter is stopped with the value 65535.

Issue the clear-counter command (0600h) to clear the counter.

Please note that no response to system communication is also counted as an error.

(The lack of response to system communication is substantially not an error.)

(3) RSYCNT (Read only)

This is a 16-bit register for measuring cyclic periodicity.

RSYCNT counts time in 1 μ s between changing points of MSYN (signal whose level is reversed by every cyclic period). Counting is carried out constantly, which allows referencing the count before the last MSYN changing point.

The upper limit to the count value is 65535 (65535 μ s), so the width of an MSYN signal beyond that limit cannot be measured.

* Since a calculated value of cyclic communication time includes allowance, the value measured by this register is usually smaller than the calculated value.

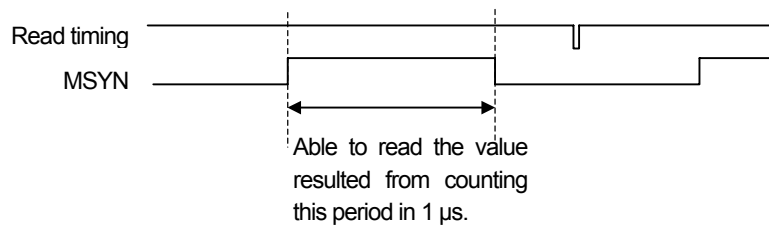


Figure 5. 1-4 RSYCNT

(4) RDJADD (Read only)

This register keeps the module ID in the last data communication received successfully.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	Module ID in the last data communication received successfully					

Figure 5. 1-5 RDJADD

(5) RVER (Read only)

This register is used to make a distinction between G9001 and G9001A on a software basis.

However, it is unnecessary to use this register since our motionCAT master boards are all equipped with the latest model, G9001A.

The values are always as shown below. (0001h).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 5. 1-6 RVER

5.2 Option ports

There is a set of option ports across the master board.

To read data from an option port, use;

`mnt520_rOptPortB()` ... For reading 1-byte data

`mnt520_rOptPortW()` ... For reading 2-byte data

To write data, use:

`mnt520_wOptPortB()` ... For writing 1-byte data

`mnt520_wOptPortW()` ... For writing 2-byte data

(1) Generic Input Port

(Base+00H:DIN) (Read only)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	IN4	IN3	IN2	IN1

Figure 5. 2-1 Bit configuration of Generic Input Port

bit	Abbreviation	Function	Remarks
0 to 3	IN 1 to 4	Generic input status "1" while ON	
4 to 15	(Undefined)	Always "0"	

Figure 5. 2-1 Contents of Generic Input Port

(2) Generic Output Setting Port

(Base+02H:DOUT) (Read/Write)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	OUT4	OUT3	OUT2	OUT1

Figure 5. 2-2 Bit configuration of Generic Output Setting and Output Status

bit	Abbreviation	Function	Remarks
0 to 3	OUT 1 to 4	Generic output setting: ON by "1", Generic output status: "1" while ON	
4~15	(Undefined)		Written values can be read.

Figure 5. 2-2 Contents of Generic Output Setting and Output Status Checking Port

(3) G9001 Status

(Base+04H:STS) (Read only)

15 to 12	11	10	9	8	7 to 4	3	2	1	0
0	2MSYN	2MERF	2MERR	2MCRY	0	1MSYN	1MERF	1MERR	1MCRY

Figure 5. 2-3 Bit configuration of G9001 Status Checking Port

bit	Abbreviation	Function	Remarks
0,8	xMCRY	1: When a signal from the signal line is received, specific period of time	
1,9	xMERR	1: When an abnormal frame or no response is received, specific period of time	
2,10	xMERF	1: When an error response frame is received	
3,11	xMSYN	Reversed by every cyclic period	

Figure 5. 2-3 Contents of G9001 Status Checking

(4) Transmission Speed Checking register

(Base+06H:SPD) (Read only)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	SPD0	SPD1	DSW2	DSW1

Figure 5. 2-4 Bit configuration of G9001 Transmission Speed Checking Port

bit	Abbreviation	Function	Remarks															
0	DSW1	Communication settings SW bit1 "1" while ON																
1	DSW2	Communication settings SW bit1 "1" while ON																
2,3	SPD1.0	Communication settings bit reading "1" while ON																
		<table border="1"> <thead> <tr> <th>SPD0</th> <th>SPD1</th> <th>Transmission speed</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>20Mbps</td> </tr> <tr> <td>1</td> <td>0</td> <td>10Mbps</td> </tr> <tr> <td>0</td> <td>1</td> <td>5Mbps</td> </tr> <tr> <td>1</td> <td>1</td> <td>2.5Mbps</td> </tr> </tbody> </table>	SPD0	SPD1	Transmission speed	0	0	20Mbps	1	0	10Mbps	0	1	5Mbps	1	1	2.5Mbps	
SPD0	SPD1	Transmission speed																
0	0	20Mbps																
1	0	10Mbps																
0	1	5Mbps																
1	1	2.5Mbps																

Figure 5. 2-4 Contents of G9001 Transmission Speed Checking Port

(5) PCI Interrupt Permission

(Base+08H:INTE) (Read/Write)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	INTE

Figure 5. 2-5 Bit configuration of PCI Interrupt Permission Setting Port

bit	Abbreviation	Function	Remarks
0	INTE	Permits interruption in PCI Bus. Permit by "1", forbid by "0." (Interruption factors depend on G9001A.)	Not supported by software for Windows.
1 to 15	(Undefined)		Written values can be read.

Table 5. 2-5 Contents of PCI Interrupt Permission Setting Port

(6) PCI Interrupt Status Port

(Base+0AH:INTS) (Read only)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	INTS

Figure 5. 2-6 Bit configuration of PCI Interrupt Status Port

bit	Abbreviation	Function	Remarks
0	INTS	"1" indicates the occurrence of an interruption in PCI Bus.	
1 to 15	(Undefined)	Always "0"	

Figure 5. 2-6 Contents of PCI Interrupt Status Port

(7) Board ID Checking Port

(Base+10H:ID) (Read only)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	BID4	BID3	BID2	BID1

Figure 5. 2-7 Bit configuration of Board ID Checking Port

bit	Abbreviation	Function	Remarks
0 to 3	BID	Able to read the status of the BID rotary switch on the board.	
4 to 15	(Undefined)	Always "0"	

Table 5. 2-7 Contents of Board ID Checking Port

(8) Board Code Checking Port

(Base+12H..17:BCOD) (Read only)

Address	Data	Contents	Remarks
+12H	4E4D H	N,M	
+14H	5254 H	52,T	
+16H	0000 H	_0	

Table 5. 2-8 Contents of Board Code Checking Port

6. Local Devices

6.1 DIO device (G9002)

A DIO device is an I/O device for the motionCAT system.

In our motionCAT series products, the DIO device is mounted on a DIO module. Its two input ports and two output ports can be controlled by the center device. (8 bits for each port)

6.2 Motion device (G9003)

A motion device (G9003) is an axis control device for the motionCAT system.

Various commands given via communications from the center device (G9001A) allows output of high-speed pulses to drive stepping and servo motors. Using constant speed, linear acceleration/deceleration, and S-curve acceleration/deceleration, it is possible to provide many different kinds of continuous, positioning, and origin-returning operations. Also, operation status can be checked through communication.

In the explanation of register bits in this section, "0" means that the value other than "0" must not be written, and that only "0" must be read.

6.2.1 Advantages

(1) Communication

The transmission speed is a maximum of 20 Mbps.

(2) Acceleration/deceleration control

This control is available in linear acceleration/deceleration and s-curve acceleration/deceleration.

Linear acceleration/deceleration can be included in the intermediate range of s-curve acceleration/deceleration. (s-curve range setting)

(3) Speed override

A change of speed is possible during movement in any motion mode.

(4) Position override

A change of a target position (travel distance) is possible during movement in the positioning mode.

If a newly set position has been passed, positioning is performed by moving in the opposite direction after deceleration stop (or immediate stop during constant-speed operation).

(5) Triangular drive peak prevention (FH correction) function

In the positioning mode, triangular drive peak prevention can be performed by automatically lowering the maximum speed when the travel distance is short.

(6) Various counter circuits

The following three counters are available.

Name	Bit length	Explanation	Count input
CTR1 Command position counter	28	Counter for command position control	Command-pulse
CTR2 Machine position counter	28	Counter for machine position control	Encoder (pulser) input Command-pulse
CTR3 General-purpose/error counter	16	Counter for deviation between command and machine positions Counter for comparator coincident output Counter for synchronous signal output	Command-pulse Encoder (pulser) input 1/4065 of reference clock Deviation between command-pulse and encoder (pulser) input

All the counters can be reset by command writing or CLR signal input.

Also, count data can be latched by command writing, LTCH input, or OLD input.

CTR3 provides ring count function that repeats specified count range.

(7) Comparator function

Three comparators are available for comparing set values with internal counter values.

Counters to be compared are selected from CTR1 (command position counter), CTR2 (machine position counter), and CTR3 (general-purpose/error counter).

- (8) Soft-limit function
Soft-limit setting is possible with CMP1, CMP2.
Immediate or deceleration stop is performed upon reaching a soft-limit range. And then, only a movement in the opposite direction is possible.
- (9) Backlash compensation
Backlash compensation function performs travel distance compensation each time the movement direction is changed.
- (10) Synchronous output function
This function can output pulse signals at specified regular intervals.
- (11) Vibration suppression function (for stepping motors)
With a control constant preset, two pulses in forward and reverse directions are added immediately before stop.
This allows suppression of vibration caused by the stop.
- (12) Manual pulsar input function
It is possible to directly operate the motor by inputting manual pulsar signals in the encoder input terminal of a module.
Input signals are quadrature-encoded signals (1, 2, 4 multiply) or up/down signals.
Command pulse output can be stopped when ELS or the soft-limit setting is valid. Movement in the opposite direction is possible.
- (13) Stepping motor's a loss of synchronism detection function
The deviation counter (CTR3) operates by command pulses and encoder signals (EA/EB). With a comparator, the counter can be used for the detection of stepping motor's a loss of synchronism and for positioning check.
- (14) Command pulse modes
Selectable modes are common pulse mode (command pulse train and sign) and 2-pulse mode (CW/CCW).
- (15) Idling pulse output function (for stepping motors)
It is possible to specify a pulse count of the starting pulse rate at acceleration start.
A runaway will be less likely to occur when initial speed is set high during the acceleration/deceleration control of the stepping motor.
- (16) Operation modes
Available basic operation modes are continuous, positioning, and origin-returning operation.
Optional setting of operation mode bits allows various operation.
<Operation mode examples>
1. Start/stop by commands
 2. Continuous operation, positioning operation by manual pulsars
 3. Origin-returning operation
 4. Positioning operation by commands
- (17) Various origin-returning sequences
<Origin-returning examples>
1. In constant-speed operation, stop by OLS signal ON. (sensor's origin)
 2. In acceleration/deceleration operation, decelerate by DLS signal ON, stop by OLS signal ON. (sensor's origin)
 3. In acceleration/deceleration operation, decelerate and stop by OLS signal ON, reverse and get out of OLS, then return to the OLS at constant speed. (sensor's origin)
 4. In constant-speed operation, stop by OLS signal ON followed by EX signal count. (sensor + Z-phase)
 5. In constant-speed operation, reverse by OLS signal ON, stop by EX signal count. (sensor + Z-phase)
 6. In acceleration/deceleration operation, decelerate by OLS signal ON, stop by EZ signal count. (sensor + Z-phase)
 7. In acceleration/deceleration operation, decelerate and stop by OLS signal ON, reverse, then stop by EZ count. (sensor + Z-phase)
 8. In constant-speed operation, stop by ELS signal ON. (normal stop) (ELS shared)
 9. In constant-speed operation, reverse by ELS signal ON, stop by EZ signal count. (ELS shared + Z-phase)
 10. In acceleration/deceleration operation, decelerate and stop by ELS signal ON, reverse, then stop by EZ signal count. (ELS shared + Z-phase)

6.2.2 I/O ports

A motionCAT motion module is equipped with four ports: the highest port 3 is for output, and port 2, 1, 0 for input. The figure below shows those ports from the top, namely Generic Output Setting, Generic Output Status, the higher-order and lower-order bits of Motion Main Statuses.

Port 3	Port 2	Port 1	Port 0
Generic Output Setting Port IOPOB	Generic Output Status Port IOPIB	Motion Main Status MMSTS (bit 15 to 8)	Motion Main Status MMSTS (bit 7 to 0)

Figure 6. 2-1 Motion module port assignments

6.2.3 Motion Main Status (MMSTS)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	GRP2	GRP1	GRP0	0	0	0	SBSY	0	0	0	0	SEVT	SERR	SEND	SINT

Figure 6. 2-2 Bit configuration of Motion Main Status (MMSTS)

bit	Abbreviation	Function	Remarks
0	SINT	Turns "1" when any of MMSTS bit 1, 2, 3 is "1."	
1	SEND	Turns "1" upon operation stop. SEND is reset when started, where bit 28 is "1" in RENV1. Also, the Interrupt Reset command (0008H) turns SEND to "1."	
2	SERR	Turns "1" upon the occurrence of stop by an error, a failure in position override, or an abnormal encoder signal. SERR turns "0" when REST is read.	
3	SEVT	Turns "1" upon the occurrence of the event set by RIRQ. SEVT turns "0" when RIST is read.	
7 to 4	(Undefined)	Always "0"	
8	SBSY	Turns "1" upon the start of pulse output, and "0" upon operation stop. (=BSY)	
11 to 9	(Undefined)	Always "0"	
14 to 12	GRP2 to GRP0	Group setting statuses Independent group :000, Group 1:001, Group 2:010 Group 3:011, Group 4:100, Group 5:101 Group 6:110, Group 7:111	
15	(Undefined)	Always "0"	

Table 6. 2-1 Contents of Motion Main Status (MMSTS)

Note. MMSTS is updated by cyclic communication. So there is a delay, as long as one cyclic communication period at the maximum, in reflecting the G9003 setting in MMSTS.

6.2.4 Generic Output Status Port (IOPIB)

This port becomes effective after the initialization of RENV2 register setting.

7	6	5	4	3	2	1	0
ELL	GRP2	GRP1	GRP0	SVGAIN	SVTL	SVRST	SVON

Figure 6. 2-3 Bit configuration of Generic Output Status Port (IOPIB)

bit	Abbreviation	Function	Remarks
0	SVON	Output status of servo ON (ON by "1")	
1	SVRST	Output status of servo reset (ON by "1")	
2	SVTL	Output status of servo torque control (ON by "1")	
3	SVGAIN	Output status of servo gain switching (ON by "1")	
4	GRP0	Group setting statuses Independent group :111, Group 1:110, Group 2:101 Group 3:100, Group 4:011, Group 5:010 Group 6:001, Group 7:000	
5	GRP1		
6	GRP2		
7	ELL	ELS polarity (0: NO, 1: NC)	

Figure 6. 2-2 Contents of Generic Output Status Port (IOPIB)

Note. IOPIB is updated by cyclic communication. So there is a delay, as long as one cyclic communication period at the maximum, in reflecting the G9003 setting in IOPIB.

6.2.5 Generic Output Port (IOPOB)

This port becomes effective after the initialization of RENV2 register setting.

7	6	5	4	3	2	1	0
ELL	GRP2	GRP1	GRP0	SVGAIN	SVTL	SVRST	SVON

Figure 6. 2-4 Bit configuration of Generic Output Setting Port (IOPOB)

bit	Abbreviation	Function	Remarks
0	SVON	Output of servo ON (ON by "1")	
1	SVRST	Output of servo reset (ON by "1")	
2	SVTL	Output of servo torque control (ON by "1")	
3	SVGAIN	Output of servo gain switching (ON by "1")	
4	GRP0	Group setting Independent group:111, Group 1:110, Group 2:101 Group 3:100, Group 4:011, Group 5:010 Group 6:001, Group 7:000	
5	GRP1		
6	GRP2		
7	ELL	ELS polarity (0: NO, 1: NC)	

Table 6. 2-3 Contents of Generic Output Port (IOPOB)

Note. The IOPOB setting is reflected in G9003 by cyclic communication. So there is a delay, as long as one cyclic communication period at the maximum, in reflecting the IOPOB setting in G9003.

6.2.6 Motion operation commands, control commands, register control commands

It is possible to control axes via data communication from the center device using commands described in this section.

Command types are those without data and those with data (register data).

If a data-less-type command is used for communication with data, the data is ignored.

If a data-attached-type command is used for communication with data exceeding the valid bit count, the excess bits are ignored.

Also, if a data-attached-type command is used for communication from the center device without data, no data is updated.

If data of 5 words or more (1 word is 2 bytes.) is transmitted, a local-side receiving-process error (ERAE) occurs on the center device side. The command is not updated in this case.

(1) Motion Operation commands

■ Start commands

(1) Start commands

Start operation when written during a stopped state.

(2) Residual Start commands

Start the operation of the remaining pulses at the positioning counter when written after a positioning operation is paused.

(3) Travel-Distance-Attached Start commands

Write a travel distance and a start command during a stopped state. The travel distance is valid for positioning operation.

If communication is performed without travel-distance setting, RMV register data (travel distance data) becomes 0 to provide an operation with 0 travel distance.

(4) Simultaneous Start command

Starts the operation when RMD bit 14 is "1."

	Command name	Abbreviation	1st word hex	2nd word hex	3rd word hex	Contents	Response
(1)	FL Constant-Speed Start	STAF _L	0050			FL Constant-Speed Start	Response frame only
	FH Constant-Speed Start	STAF _H	0051			FH Constant-Speed Start	Response frame only
	Acceleration Start	STAU _D	0053			Acceleration Start (acceleration → FH constant-speed → deceleration)	Response frame only
(2)	Residual FL Constant-Speed Start	CNTF _L	0054			Residual FL Constant-Speed Start	Response frame only
	Residual FH Constant-Speed Start	CNTF _H	0055			Residual FH Constant-Speed Start	Response frame only
	Residual Acceleration Start	CNTU _D	0057			Residual Acceleration Start	Response frame only
(3)	Travel-Distance-Attached FL Constant-Speed Start	RMSTF _L	0058	RMV lower data	RMV higher data	Written in RMV register + FL constant-speed start	Response frame only
	Travel-Distance-Attached FH Constant-Speed Start	RMSTF _H	0059	RMV lower data	RMV higher data	Written in RMV register + FH constant-speed start	Response frame only
	Travel-Distance-Attached Acceleration Start	RMSTU _D	005b	RMV lower data	RMV higher data	Written in RMV register + acceleration start	Response frame only
(4)	Simultaneous Start	CMSTA	0006			STA output (valid within the slave)	Response frame only
		SPSTA	002a			STA substitutional input (valid only within the own axis)	Response frame only

Table 6. 2-4 Operation Commands

■ Speed Change commands

Change the operation speed when written during operation. A command written during a stopped state is ignored.

Command name	Abbreviation	1st word hex	2nd word hex	3rd word hex	Contents	Response
Instant-Speed-Change To FL Constant	FCHGL	0040			Changes speed instantly to FL constant speed.	Response frame only
Instant-Speed-Change To FH Constant	FCHGH	0041			Changes speed instantly to FH constant speed.	Response frame only
Deceleration To FL Speed	FSCHL	0042			Decelerates speed down to FL speed.	Response frame only
Acceleration To FH Speed	FSCHH	0043			Accelerates speed up to FH speed.	Response frame only

Table 6. 2-5 Speed Change commands

■ Stop commands

Stop operation when written.

Command name	Abbreviation	1st word hex	2nd word hex	3rd word hex	Contents	Response
Immediate Stop	STOP	0049			Immediate Stop	Response frame only
Deceleration Stop	SDSTP	004A			Deceleration Stop	Response frame only

Table 6. 2-6 Stop commands

Command name	Abbreviation	1st word hex	2nd word hex	3rd word hex	Contents	Response
Simultaneous Stop (STP output)	CMSTP	0007			Simultaneous Stop (STP output)	Response frame only

Table 6. 2-7 Stop commands

(2) Motion Control commands

These commands perform various controls such as counter reset.

Command name	Abbreviation	1st word hex	2nd word hex	3rd word hex	Contents	Response
NOP command	NOP	0000			Invalid command	Response frame only
SEND Interrupt Reset	INTRS	0008			Resets MMSTS bit 0 (SEND).	Response frame only
Software Reset	SRST	0004			Software reset Motion device (G9003) registers and commands are reset (except communication-related items)	Response frame only
Counter Reset	CUN1R	0020			Resets CTR1 (command position).	Response frame only
	CUN2R	0021			Resets CTR2 (machine position).	Response frame only
	CUN3R	0022			Resets CTR3 (general-purpose/error).	Response frame only
Clear-Servo-Error-Counter Output Control	SVCTR CLOUT	0024			Outputs the SVCTRCL signal.	Response frame only
	SVCTR CLRST	0025			Resets the SVCTRCL signal.	Response frame only
PCS command	PCSON	0028			Substitutes PCS signal input. This command performs the same process as PCS input ON when PCS input is enabled in the RMD register (operation mode).	Response frame only
Counter Latch	LTCH	0029			Substitutes LTCH signal input.	Response frame only

Table 6. 2-8 Motion Control commands

(3) Motion Register Control commands

■ Motion Register Write commands

Command name	Abbreviation	1st word hex	2nd word hex	3rd word hex	Response
RMV Override Write	WRMVOR	0080	Lower data	Higher data	Response frame only
RMV Write	WRMV	0090	Lower data	Higher data	Response frame only
RFL Write	WRFL	0091	Lower data	Higher data	Response frame only
RFH Write	WRFH	0092	Lower data	Higher data	Response frame only
RUR Write	WRUR	0093	Data		Response frame only
RDR Write	WRDR	0094	Data		Response frame only
RMG Write	WRMG	0095	Data		Response frame only
RDP Write	WRDP	0096	Lower data	Higher data	Response frame only
RMD Write	WRMD	0097	Lower data	Higher data	Response frame only
RUS Write	WRUS	0099	Data		Response frame only
RDS Write	WRDS	009A	Data		Response frame only
RFA Write	WRFA	009B	Lower data	Higher data	Response frame only
					Response frame only
RENV1 Write	WRENV1	009C	Lower data	Higher data	Response frame only
RENV2 Write	WRENV2	009D	Lower data	Higher data	Response frame only
RENV3 Write	WRENV3	009E	Lower data	Higher data	Response frame only
RENV4 Write	WRENV4	009F	Lower data	Higher data	Response frame only
RENV5 Write	WRENV5	00A0	Lower data	Higher data	Response frame only
RENV6 Write	WRENV6	00A1	Lower data	Higher data	Response frame only
					Response frame only
RCTR1 Write	WRCTR1	00A3	Lower data	Higher data	Response frame only
RCTR2 Write	WRCTR2	00A4	Lower data	Higher data	Response frame only
RCTR3 Write	WRCTR3	00A5	Data		Response frame only
					Response frame only
RCMP1 Write	WRCMP1	00A7	Lower data	Higher data	Response frame only
RCMP2 Write	WRCMP2	00A8	Lower data	Higher data	Response frame only
RCMP3 Write	WRCMP3	00A9	Lower data	Higher data	Response frame only
					Response frame only
RIRQ Write	WRIRQ	00AC	Lower data	Higher data	Response frame only

Table 6. 2-9 Motion Register Write commands

■ Motion Register Read commands and response data

Command name	Abbreviation	Command	Response data				
		1st word hex	Response data type	1st word hex	2nd word hex	3rd word hex	Valid bit count
RMV Read	RRMV	00D0	ARMV Data	00D0	Lower data	Higher data	28
RFL Read	RRFL	00D1	ARFL Data	00D1	Lower data	Higher data	17
RFH Read	RRFH	00D2	ARFH Data	00D2	Lower data	Higher data	17
RUR Read	RRUR	00D3	ARUR Data	00D3	Data	0	16
RDR Read	RRDR	00D4	ARDR Data	00D4	Data	0	16
RMG Read	RRMG	00D5	ARMG Data	00D5	Data	0	11
RDP Read	RRDP	00D6	ARDP Data	00D6	Lower data	Higher data	24
RMD Read	RRMD	00D7	ARMD Data	00D7	Lower data	Higher data	25
RUS Read	RRUS	00D9	ARUS Data	00D9	Data	0	16
RDS Read	RRDS	00DA	ARDS Data	00DA	Data	0	16
RFA Read	RRFA	00DB	ARFA Data	00DB	Lower data	Higher data	17
RENV1 Read	RRENV1	00DC	ARENV1 Data	00DC	Lower data	Higher data	30
RENV2 Read	RRENV2	00DD	ARENV2 Data	00DD	Lower data	Higher data	23
RENV3 Read	RRENV3	00DE	ARENV3 Data	00DE	Lower data	Higher data	31
RENV4 Read	RRENV4	00DF	ARENV4 Data	00DF	Lower data	Higher data	28
RENV5 Read	RRENV5	00E0	ARENV5 Data	00E0	Lower data	Higher data	32
RENV6 Read	RRENV6	00E1	ARENV6 Data	00E1	Lower data	Higher data	32
RCTR1 Read	RRCTR1	00E3	ARCTR1 Data	00E3	Lower data	Higher data	28
RCTR2 Read	RRCTR2	00E4	ARCTR2 Data	00E4	Lower data	Higher data	28
RCTR3 Read	RRCTR3	00E5	ARCTR3 Data	00E5	Data	Sign extension	16
RCMP1 Read	RRCMP1	00E7	ARCMP1 Data	00E7	Lower data	Higher data	28
RCMP2 Read	RRCMP2	00E8	ARCMP2 Data	00E8	Lower data	Higher data	28
				00E9	Lower data	Higher data	28
RCMP3 Read	RRCMP3	00E9	ARCMP3 Data				
RIRQ Read	RRIRQ	00EC	ARIRQ Data	00EC	Data	0	15
RLTC1 Read	RRLTC1	00ED	ARLTC1 Data	00ED	Lower data	Higher data	28
RLTC2 Read	RRLTC2	00EE	ARLTC2 Data	00EE	Lower data	Higher data	28
RLTC3 Read	RRLTC3	00EF	ARLTC3 Data	00EF	Lower data	Higher data	17
RSTS Read	RRSTS	00F1	ARSTS Data	00F1	Lower data	Higher data	23
REST Read	RREST	00F2	AREST Data	00F2	Data	0	15
RIST Read	RRIST	00F3	ARIST Data	00F3	Data	0	15
RPLS Read	RRPLS	00F4	ARPLS Data	00F4	Lower data	Higher data	28
RSPD Read	RRSPD	00F5	ARSPD Data	00F5	Lower data	Higher data	27
RSDC Read	RRSDC	00F6	ARSDC Data	00F6	Lower data	Higher data	24

Table 6. 2-10 Motion Register Read commands

Note 1. If a Register Read command is transmitted with four-word data attached, no response will be returned, resulting in a data communication error (EDTE) on the center device side.

6.2.7 Motion Device registers

The initial values of all the registers are "0." If the same values are needed as last time, rewriting is unnecessary.

<Explanations on symbols>

The bits indicated as "*" are ignored while written, and turned to "0" while read.

The bits indicated as "&" are ignored while written, and turned to the same value as the highest-order bit among those indicated as a blank while read.

(sign extension)

(1) RMV: Travel Distance register (28 bits)

This register is used to specify the travel distance in positioning operation.

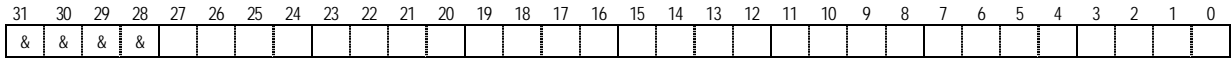


Figure 6. 2-5 RMV: Bit configuration of the Travel Distance register

Settings differ depending on the operation mode. The setting range is from -134,217,728 to +134,217,727.

A change in the RMV register during operation allows position override.

(2) RFL: Base Speed register (17 bits)

This register is used to specify the base speed (initial speed, stop speed) in operation that includes acceleration/deceleration.

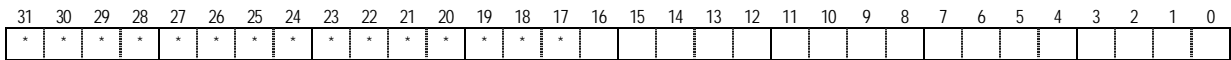


Figure 6. 2-6 RFL: Bit configuration of the Base Speed register

The speed in FL constant-speed operation and the base speed in acceleration/deceleration operation can be specified in the range of 1 to 100,000 (186A0h). A value between 100,000 and 131,071 (186A0h to 1FFFFh) is always handled as 100,000.

The actual speed will be the value calculated with the RMG (Speed Multiplier Setting register) value.

$$\text{Base speed [pps]} = \text{RFL} \times \frac{200}{(\text{RMG} + 1)}$$

(3) RFH: Operation Speed Setting register

This register sets the operation speed.

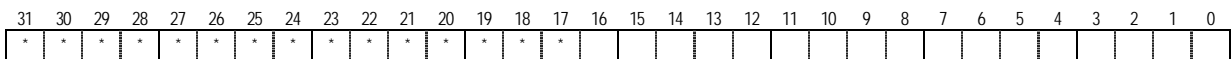


Figure 6. 2-7 RFH: Bit configuration of the Operation Speed Setting register

A change in the RFH register during operation allows speed override.

The speed in FH constant-speed and acceleration/deceleration operations can be specified in the range of 1 to 100,000 (186A0h). A value between 100,000 and 131,071 (186A0h to 1FFFFh) is always handled as 100,000.

For acceleration/deceleration operation, set a value larger than the one in RFL.

The actual speed will be the value calculated with the RMG (Speed Multiplier Setting register) value.

$$\text{Operation speed [pps]} = \text{RFH} \times \frac{200}{(\text{RMG} + 1)}$$

(4) RUR: Acceleration Rate register (16 bits)

This register sets the acceleration rate.

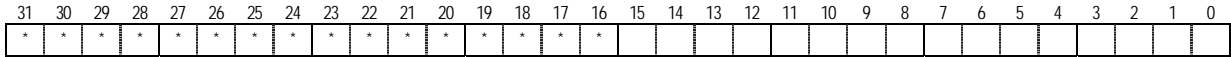


Figure 6. 2-8 RUR: Bit configuration of the Acceleration Rate register

Set the acceleration characteristic in acceleration/deceleration operation in the range of 1 to 65,535 (FFFFh).

(5) RDR: Deceleration Rate register (16 bits)

This register sets the deceleration rate.

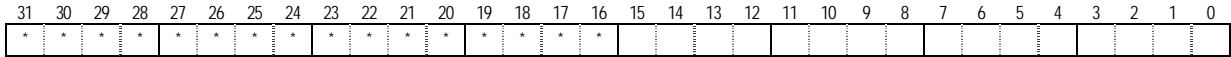


Figure 6. 2-9 RDR: Bit configuration of the Deceleration Rate register

Set the deceleration characteristic in acceleration/deceleration operation usually in the range of 1 to 65,535 (FFFFh). Even when the deceleration starting point is set to auto (MSDP (bit12) is 0 in RMD register), the set value in the RDR register is used as the deceleration rate. If RDR is set "0", the deceleration rate will be the value set in RUR.

(6) RMG: Speed Multiplier Setting register (11 bits)

This register sets the speed multiplier.

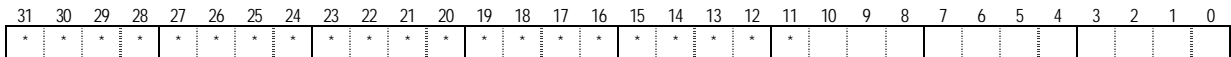


Figure 6. 2-10 RMG: Bit configuration of the Speed Multiplier Setting register

Set the relation between the speed and the set values of RFL, RFH, and RFA in the range of 2 to 2,047 (07FFh). The higher multiplier gives the speed configurable in rougher pitch. The operation speed [PPS] is the product of the speed multiplier and the value set in the speed register.

(7) RDP: Deceleration Starting Point register (24 bits)

This register is used to set a starting point of deceleration in positioning operation.

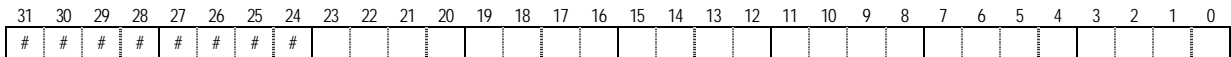


Figure 6. 2-11 RDP: Bit configuration of the Deceleration Starting Point register

Set a value to determine a deceleration starting point in acceleration/deceleration as well as positioning operations. The bits indicated as “#” are ignored while written, and set according to the MSDP (bit 12) setting in the RMD register while read.

MSDP	Setting	Bit #
0	Provides an offset to the automatically set value. A positive value causes deceleration to start early, resulting in a longer base speed (FL speed) section. A negative value causes deceleration to start later, not reaching the base speed (FL speed).	Same as bit 23 (sign extension)
1	Starts deceleration when the remaining travel distance gets lower than the set value.	0

(8) RMD: Operation Mode register (25 bits)

This register sets the operation mode.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSPE	MSY	MPCS	MSDP	METM	MSMD	MINP	MSDE	0	MOD						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	MIOR	MFH	MUB	MMPH	MPH	MINT	MMSK	MADJ	MSPO

Figure 6. 2-12 RMD: Bit configuration of the Operation Mode register

bit	Abbreviation	Contents	
6 to 0	MOD	Sets the operation mode.	
		MOD hex Operation mode	
		00	Continuous operation in (+) direction by command control
		08	Continuous operation in (-) direction by command control
		10	Origin-returning operation in (+) direction
		18	Origin-returning operation in (-) direction
		12	Getting-out-of-origin operation in (+) direction
		1A	Getting-out-of-origin operation in (-) direction
		15	Origin search operation in (+) direction
		1D	Origin search operation in (-) direction
		20	Operation to +EL or +SL
		28	Operation to -EL or -SL
		22	Getting-out-of -EL or -SL operation
		2A	Getting-out-of +EL or +SL operation
		24	Operation in (+) direction until EZ count completion
		2C	Operation in (-) direction until EZ count completion
		41	Positioning operation (set relative target position)
		44	Return to zero-point of command position (CTR1)
		45	Return to zero-point of machine position (CTR2)
		46	One-pulse operation in (+) direction
4E	One-pulse operation in (-) direction		
47	Timer operation		
01	Continuous operation by pulsar (PA/PB) input		
51	Positioning operation by pulsar (PA/PB) input		
54	Return to zero-point of command position by pulsar (PA/PB) input		
55	Return to zero-point of machine position by pulsar (PA/PB) input		
7	Undefined	(Always set 0.)	
8	MSDE	0: Disables DLS input. (Checking by RSTS possible.) 1: Decelerates (deceleration stop) operation by DLS input ON.	
9	MINP	0: Disables operation-completion delay by INPOS input. (Checking by RSTS possible.) 1: Completes operation by INPOS input ON.	
10	MSMD	Sets the acceleration/deceleration characteristic in acceleration/deceleration operation. (0: linear acceleration/deceleration 1: S-curve acceleration/deceleration)	
11	METM	Sets the operation completion timing. (0: cycle completion 1: pulse completion) Select pulse completion when using the vibration suppression function.	
12	MSDP	Sets a deceleration starting point in acceleration/deceleration operation. Valid in positioning operation. (0: auto setting 1: manual setting) (Note 1)	
13	MPCS	Enables or disables PCS input. 1: Performs pulse count management from PCS (SVRDY) input ON in positioning operation. Set "0" when using the SVRDY signal as servo ready.	
14	MSY	0: Starts immediately. 1: Starts by STA input or the Simultaneous Start command.	
15	MSPE	1: Stops by STP input or the Simultaneous Stop command. The stop method is set by bit 19 in RENV1.	
16	MSPO	1: Outputs the STP signal automatically upon abnormal stop.	
17	MADJ	Sets the FH correction function.	
18	MMSK	1: Masks pulse output.	
19	MINT	1: Masks interrupt output (SINT). (Error status and event status are changed.)	
23 to 20	-	(Reserved: Always set 0.)	
24	MIOR	Sets the method for monitoring the output setting bit of generic I/O ports. MIOR is used to prevent the interrupt-on-input-change operation by the center device when an output-port change occurs. 0: Able to read from port 2 the status of the output setting bit. 1: The bit corresponding to port 2 is "0" regardless of the status of the output setting bit.	
31 to 25	Undefined	(Always set 0.)	

Table 6. 2-11 RMD: Contents of the Operation Mode register

Note 1. Deceleration down to FL is impossible when the auto setting of Deceleration Starting Point is: (deceleration time) > (acceleration time x 2).
If the deceleration time exceeds double the acceleration time, set the Deceleration Starting Point to the manual setting.

(9) RUS: Acceleration S-Curve register (16 bits)

This register specifies the S-curve section in S-curve acceleration.

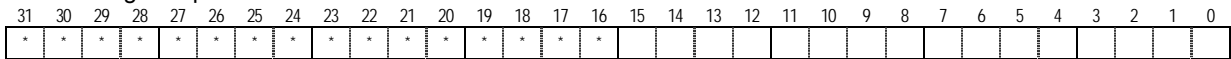


Figure 6. 2-13 RUS: Bit configuration of the Acceleration S-Curve register

Set the S-curve section in S-curve acceleration in the range of 1 to 50,000 (C350h).

A value between 50,000 and 65,535 (C350h to FFFFh) is always handled as 50,000.

The S-curve acceleration section, S_{SU} , will be the value calculated with the RMG (Speed Multiplier Setting register) value.

When “0” is set, the internal arithmetic will use “(RFH-RFL)/2” as a substitute to give an S-curve acceleration operation without a linear acceleration section.

When a value larger than “(RFH-RFL)/2” is set, the operation will not reach the maximum acceleration, resulting in acceleration time different from the one by the formula. Therefore, use a value less than “(RFH-RFL)/2.”

(10) RDS: Deceleration S-Curve register (16 bits)

This register specifies the S-curve section in S-curve deceleration.

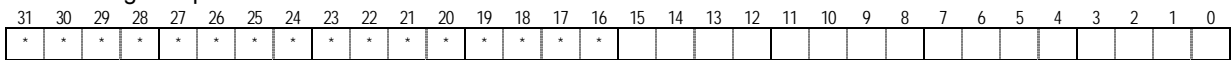


Figure 6. 2-14 RDS: Bit configuration of the Deceleration S-Curve register

Set the S-curve section in S-curve deceleration in the range of 1 to 50,000 (C350h).

The S-curve deceleration section, S_{SD} , will be the value calculated with the RMG (Speed Multiplier Setting register) value.

When “0” is set, the internal arithmetic will use “(RFH-RFL)/2” as a substitute to give an S-curve deceleration operation without a linear deceleration section.

When a value larger than “(RFH-RFL)/2” is set, the operation will not reach the maximum deceleration, resulting in deceleration time different from the one by the formula. Therefore, use a value less than “(RFH-RFL)/2.”

(11) RFA: Auxiliary Speed register (17 bits)

This register sets the reverse constant speed in origin-returning operation or the constant speed in backlash compensation.

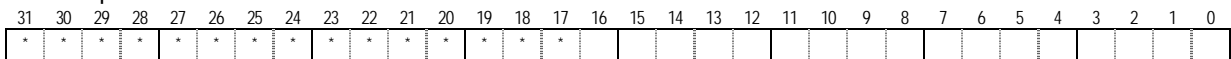


Figure 6. 2-15 RFA: Bit configuration of the Auxiliary Speed register

RFA is used as the reverse constant speed in origin-returning operation.

Or it is used to set the travel-distance compensation speed (FA speed) in backlash, in the range of 1 to 100,000 (186A0h).

A value between 100,000 and 131,071(186A0h and 1FFFFh) is always handled as 100,000.

The actual operation speed will be the value calculated with the RMG (Speed Multiplier Setting register) value.

(12) RIRQ: Event Factor Setting register (13 bits)

This register specifies the event factor. Set to “1” the bit corresponding to the event factor you want.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	IRNP	IRNA	IRSA	IRSD	IROL	IRLT	IRCL	IRC3	IRC2	IRC1	IRDE	IRDS	IRUE	IRUS	IREN
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 6. 2-16 Bit configuration of the Event Factor Setting register

bit	Abbreviation	Contents
0	IREN	When operation stops normally
1	IRUS	When acceleration begins
2	IRUE	When acceleration finishes
3	IRDS	When deceleration begins
4	IRDE	When deceleration finishes
5	IRC1	When matching the Comparator-1 condition
6	IRC2	When matching the Comparator-2 condition
7	IRC3	When matching the Comparator-3 condition
8	IRCL	When resetting the count value by CLR signal input
9	IRLT	When latching the count value by LTCH input
10	IROL	When latching the count value by OLS input
11	IRSD	When DLS input is turned ON
12	IRSA	When STA input is turned ON
13	IRNA	When the Group is begun
14	IRNP	When the Group is stopped
31 to 15	Undefined	(Always set “0.”)

Table 6. 2-12 RIRQ: Contents of the Event Factor Setting register

(13) RIST: Event Status register (13 bits)

This register is used to check the event factor. (Read only)

The corresponding bit is turned to “1” upon the occurrence of an event.

RIST is reset when read.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	ISNP	ISNA	ISSA	ISSD	ISOL	ISLT	ISCL	ISC3	ISC2	ISC1	ISDE	ISDS	ISUE	ISUS	ISEN
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 6. 2-17 Bit configuration of the Event Status register

bit	Abbreviation	Contents
0	ISEN	When operation stops normally
1	ISUS	When acceleration begins
2	ISUE	When acceleration finishes
3	ISDS	When deceleration begins
4	ISDE	When deceleration finishes
5	ISC1	When matching the Comparator-1 condition
6	ISC2	When matching the Comparator-2 condition
7	ISC3	When matching the Comparator-3 condition
8	ISCL	When resetting the count value by CLR signal input
9	ISLT	When latching the count value by LTCH input
10	ISOL	When latching the count value by OLS input
11	ISSD	When DLS input is turned ON
12	ISSA	When STA input is turned ON
13	ISNA	When the Group is begun
14	ISNP	When the Group is stopped
31 to 15	Undefined	(Always set “0.”)

Table 6. 2-13 RIST: Contents of the Event Status register

(14) REST: Error Status register (15 bits)

This register is used to check the error factor. (Read only)

The corresponding bit is turned to "1" upon the occurrence of an error.

REST is reset when reading.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	ESPE	ESEE	ESOR	0	ESNT	ESPO	ESSD	ESEM	ESSP	ESAL	ESML	ESPL	ESC3	ESC2	ESC1
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 6. 2-18 Bit configuration of the Error Status register

bit	Abbreviation	Contents	
0	ESC1	When stopped due to Comparator-1 condition matching (+SLS)	
1	ESC2	When stopped due to Comparator-2 condition matching (-SLS)	
2	ESC3	When stopped due to Comparator-3 condition matching	
3	ESPL	When stopped due to +ELS input ON	
4	ESML	When stopped due to -ELS input ON	
5	ESAL	When stopped due to SVALM input ON	
6	ESSP	When stopped due to STP input	
7	ESEM	(Reserved: undefined)	
8	ESSD	When deceleration stop occurs due to DLS input ON	
9	ESPO	When the overflow of the manual-pulsar buffer counter occurs	
10	ESNT	When stopped due to a communication error	
11	Undefined	Undefined (Always "0.")	
12	ESOR	When positioning override operation fails	
13	ESEE	When an encoder input error occurs	Note: Operation is not stopped.
14	ESPE	When a pulsar input error occurs	
31 to 15	Undefined	(Always set "0.")	

Table 6. 2-14 REST: Contents of the Error Status register

(15) RSTS: Expansion Status register (28 bits)

This register is used to check the Expansion Status. (Read only)

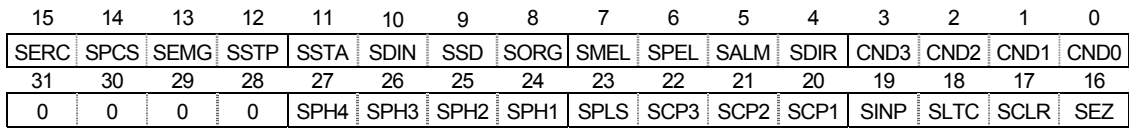


Figure 6. 2-19 RSTS: Bit configuration of the Expansion Status register

bit	Abbreviation	Contents	
3 to 0	CND3 to 0	Indicates operation status.	
		0000: During a stop	1000: During acceleration
		0001: Waiting for STA input	1001: During FH constant-speed operation
		0010: Waiting for SVCTRCL timer completion	1010: During deceleration
		0011: Waiting for direction-change timer completion	1011: Waiting for INPOS
		0100: During backlash compensation	1111: Others (during start control)
		0101: Waiting for manual pulsar input	1100: Undefined
		0110: During FA constant-speed operation	1101: Undefined
		0111: During FL constant-speed operation	1110: Undefined
4	SDIR	Operation direction (0: + direction 1: - direction)	
5	SALM	"1" when the ALM input signal is in ON state.	
6	SPEL	"1" when the +EL input signal is in ON state.	
7	SMEL	"1" when the -EL input signal is in ON state.	
8	SORG	"1" when the ORG input signal is in ON state.	
9	SSD	"1" when the DLS input signal is in ON state. The DLS input polarity (RENV1 bit 6) and the latch function (RENV1 bit 5) are reflected.	
10	SDIN	"1" when the DLS input terminal is High. (Input terminal status)	
11	SSTA	Unused	
12	SSTP	Unused	
13	SEMG	Unused	
14	SPCS	"1" when the SVRDY (PCS) input signal is in ON state.	
15	SERC	"1" when the SVCTRCL output signal is in ON state.	
16	SEZ	"1" when the encoder Z-phase input signal is in ON state.	
17	SCLR	"1" when the CLR input signal is in ON state.	
18	SLTC	"1" when the LTCH input signal is in ON state.	
19	SINP	"1" when the INPOS input signal is in ON state.	
20	SCP1	"1" when the CMP 1 comparison condition is matched.	
21	SCP2	"1" when the CMP 2 comparison condition is matched.	
22	SCP3	"1" when the CMP 3 comparison condition is matched.	
23	SPLS	"1" when pulsar output (±) is in ON state. Note 1	
24	SPH1	Unused	
25	SPH2	Unused	
26	SPH3	Unused	
27	SPH4	Unused	
31~28	Undefined	(Always "0.")	

Table 6. 2-15 RSTS: Contents of the Expansion Status register

Note 1. Logical OR output of the CW/CCW signal

(16) RCTR1: Command Position Counter register (28 bits)

This register provides CTR1 (command position counter).

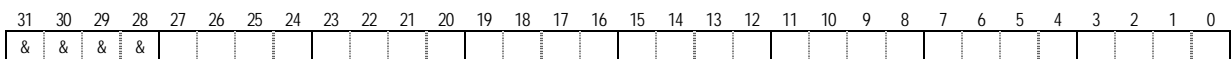


Figure 6. 2-20 RCTR1: Bit configuration of the Command Position Counter register

This register is for counting command pulses.
Its values range from -134,217,728 to +134,217,727. (signed 28 bits)

(17) RCTR2: Machine Position Counter register (28 bits)

This register provides CTR2 (machine position counter).

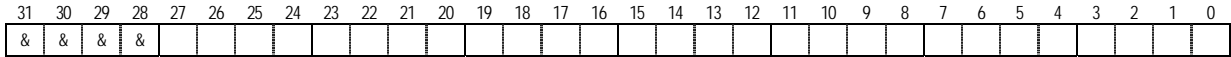


Figure 6. 2-21 RCTR2: Bit configuration of the Machine Position Counter register

This register is used for counting encoder signals (EA/EB input), pulsar signals (PA/PB input), and command pulses. Its values range from -134,217,728 to +134,217,727. (signed 28 bits)

(18) RCTR3: General-Purpose/Error Counter register (16 bits)

This register provides CTR3 (general-purpose/error counter).

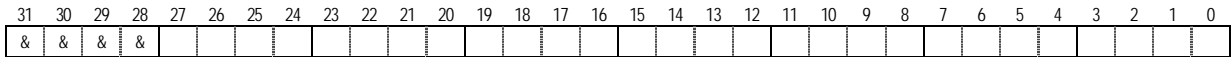


Figure 6. 2-22 RCTRz3: Bit configuration of the General-Purpose/Error Counter register

This register is used for counting the differences between command pulses and encoder/pulsar signals. RCTR3 also serves as a general-purpose counter. Its values range from -32,768 to +32,767. RCTR3 does not count values beyond the valid range, giving the maximum value instead.

(19) RCMP 1 to 3: Comparator 1 to 3 registers (28 bits)

These registers set comparison data for Comparator 1 to 3.

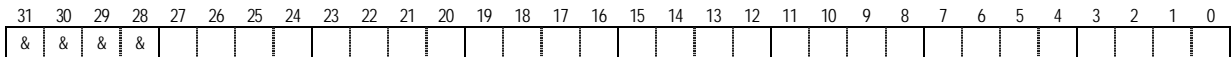


Figure 6. 2-23 RCMP 1 to 3: Bit configuration of Comparator 1 to 3 registers

Their values range from -134,217,728 to +134,217,727. (signed 28 bits)

(20) RLTC1: Counter 1 Latch register (28 bits)

This register latches CTR1 (command position) data. (Read only)

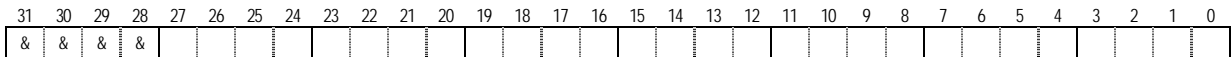


Figure 6. 2-24 RLTC1: Bit configuration of the Counter 1 Latch register

The contents of CTR1 are copied to RLTC1 by LTCH, OLS input, or the LTCH command. The data ranges from -134,217,728 to +134,217,727. (signed 28 bits)

(21) RLTC2: Counter 2 Latch register (28 bits)

This register latches CTR2 (machine position) data. (Read only)

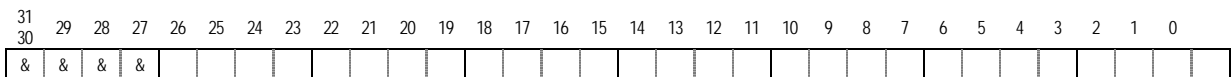


Figure 6. 2-25 RLTC2: Bit configuration of the Counter 2 Latch register

The contents of CTR2 are copied to RLTC2 by LTCH, OLS input, or the LTCH command. The data ranges from -134,217,728 to +134,217,727. (signed 28 bits)

6.2.8 Motion Device Environment Setting Registers

(1) RENV1: Environment Setting 1 Register (30 bits)

This is a register for Environment Setting 1, mainly to define the input/output terminal specifications.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERCL	EPW2	EPW1	EPW0	EROR	EROE	ALML	ALMM	ORGL	SDL	SDLT	SDM	ELM	PMD2	PMD1	PMD0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	PDTC	SEDR	SEDM	DTMF	FLTR	PCSL	LTCL	INPL	CLR1	CLR0	STPM	STAM	ETW1	ETW0

Figure 6. 2-30 RENV1: Bit configuration of the Environment Setting 1 register


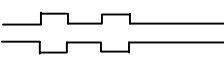

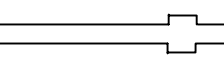

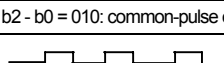

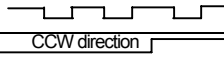
bit	Abbreviation	Contents							
2 to 0	PMD0 to PMD2	Signal terminal name	b2 - b0 = 100: 2-pulse output form	Command output					
				CW pulse output					
				CCW pulse output					
		Signal terminal name	b2 - b0 = 010: common-pulse output form	CW pulse output					
				Pulse train					
				Direction output					
		Setting value	'0'	'1'					
3	ELM	Process performed upon ELS input ON (Note 1)	Immediate stop	Deceleration stop					
4	SDM	Process performed upon DLS input ON (Note 2)	Deceleration only	Deceleration stop					
5	SDLT	DLS-input latch function (Note 2) To be ON when DLS signal width is short. Latch signals are reset when DLS input is OFF at startup. Latch signals are also reset by "SDLT = 0."	Disabled	Enabled					
6	SDL	DLS signal input polarity	0: NC	1: NO					
7	ORGL	OLS signal input polarity	0: NC	1: NO					
8	ALMM	Process performed upon SVALM input ON	Immediate stop	Deceleration stop					
9	ALML	SVALM signal input polarity	0: NC	1: NO					
10	EROE	On immediate stop by ±ELS, SVALM input, the error counter is cleared; the (SVCTRCL) signal is automatically output. However, no output occurs on deceleration stop. The signal is also output if immediate stop is performed where MOD is "010 X000" in the RMD register (operation up to ELS) and ELS normal stop is set.	Auto-output disabled	Auto-output enabled					
11	EROR	Auto-output of the clear-error-counter (SVCTRCL) signal at origin-returning completion	Auto-output disabled	Auto-output enabled					
14 to 12	EPW2 to 0	Output pulse width of the clear-error-counter (SVCTRCL) signal							
		000	001	010	011	100	101	110	111
		12μs	102μs	409μs	1.6ms	13ms	52ms	104ms	Level output
15	ERCL	Output logic setting for the clear-error-counter (SVCTRCL) signal (0: negative logic 1: positive logic)							
17 to 16	ETW1 to 0	Delay time after SVCTRCL signal output		00	01	10	11		
				0μs	12μs	1.6ms	104ms		
18	STAM	STA input method		0: Level trigger		1: Edge trigger			
19	STPM	STP input stop (bit 15 is 1 in RMD) type setting		0: Immediate stop		1: Deceleration stop			
21 to 20	CLR1 to 0	CLR input type	00	01	10		11		
			Falling edge	Rising edge	L level		H level		
22	INPL	INPOS signal input polarity		0: NC		1: NO			
23	LTCL	LTC signal input type		0: Falling		1: Rising			
24	PCSL	SVRDY (PCS) signal input polarity		0: NC		1: NO			
25	FLTR	Unused (Always set "0.")							
26	DTMF	Direction change timer (0.2ms) function		0: ON		1: OFF			
27	SEDM	SEND output forbidden (Motion main status is changed.)		0: Permitted		1: Forbidden			
28	SEDR	1: SEND is reset at startup.							
29	PDTC	Unused (Always set "0.")							
31, 30	Undefined	(Always set "0.")							

Table 6. 2-17 RENV1: Contents of Environment Setting 1 register

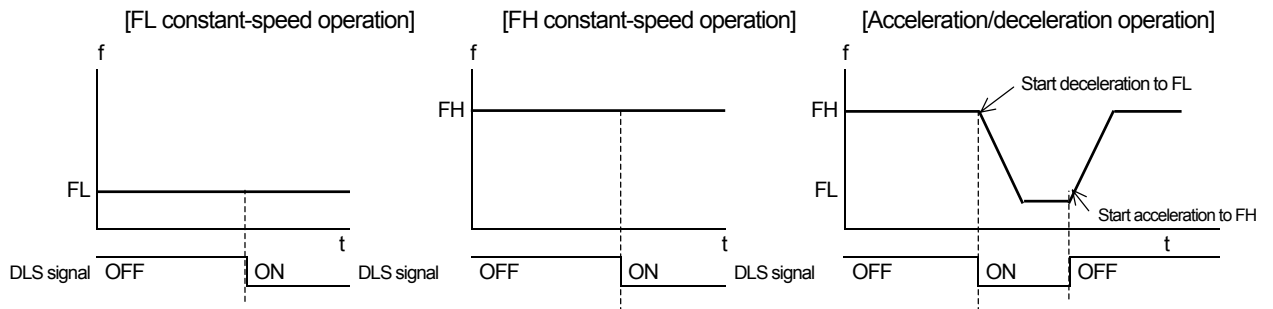
Note 1. If the process performed upon ELS input ON is set to "Deceleration stop" (ELM = 1), operation starts to be decelerated by ELS input ON, and stops beyond the ELS position. Be careful, therefore, not to cause a collision with the machines.

Note 2. If MSDE is 0 in the RMD (Operation Mode) register, DLS signals are ignored.

With DLS signal input enabled, DLS signal ON during operation causes (1) deceleration, (2) latch/deceleration, (3) deceleration stop, or (4) latch/ deceleration stop, depending on the SDM (bit4) and SDLT (bit5) settings.

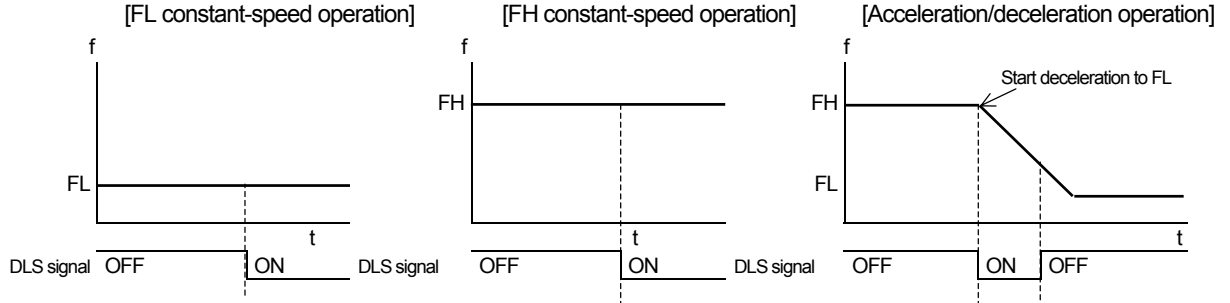
(1) Deceleration <SDM (bit4) = 0, SDLT (bit5) = 0 in RENV1 register>

- In constant-speed operation, DLS signals are ignored.
- In acceleration/deceleration operation, DLS signal ON causes deceleration down to the FL speed. After or during deceleration, the operation is accelerated up to FH upon DLS signal OFF.
- If an Acceleration Start command is written during DLS signal ON, operation is performed at FL speed, and accelerated up to FH when DLS signal is turned OFF.



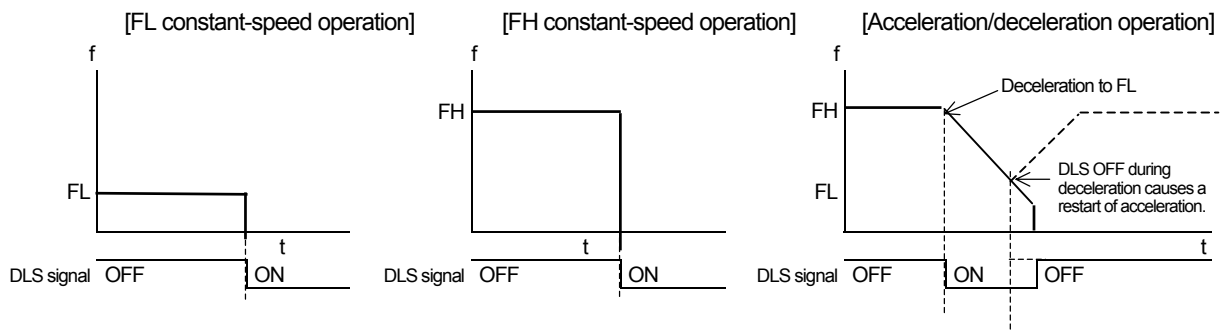
(2) Latch/deceleration <SDM (bit4) = 0, SDLT (bit5) = 1 in RENV1 register>

- In constant-speed operation, DLS signals are ignored.
- In acceleration/deceleration operation, DLS signal ON causes deceleration down to the FL speed. Even when the DLS signal is turned OFF after or during deceleration, the operation is maintained at FL speed instead of being accelerated toward the FH speed.
- If an Acceleration Start command is written during DLS signal ON, operation is performed at FL speed and not accelerated toward the FH speed even when the DLS signal is turned OFF.



(3) Deceleration stop <SDM (bit4) = 1, SDLT (bit5) = 0 in RENV1 register>

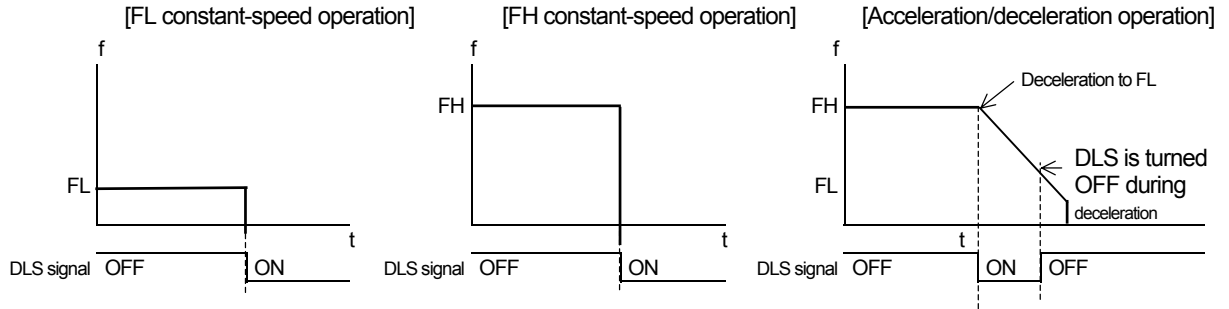
- DLS signal ON during constant-speed operation causes operation to stop.
- In acceleration/deceleration operation, DLS signal ON causes operation to decelerate down to the FL speed then stop. DLS signal OFF during deceleration causes acceleration up to FH.
- If a Start command is written during DLS signal ON, operation is not started, thus completed.
- An event occurs when operation stops.



(4) Latch/ deceleration stop <SDM (bit4) = 1, SDLT (bit5) = 1 in RENV1 register>

DLS signal ON during constant-speed operation causes operation to stop.

- In acceleration/deceleration operation, DLS signal ON causes operation to decelerate down to the FL speed then stop.
- DLS signal OFF during deceleration does not cause acceleration.
- If a Start command is written during DLS signal ON, operation is not started, thus completed.
- An event occurs when operation stops.



(2) RENV2: Environment Setting 2 Register (23 bits)

This is a register for Environment Setting 2 to define the specifications of generic ports, encoder input, and pulsar input.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIM1	PIM0	PINF	EZL	EDIR	EIM1	EIM0	EINF	1	1	1	1	1	1	1	1
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	IDL2	IDL1	IDL0	0	POFF	EOFF	PDIR

Figure 6. 2-31 Bit configuration of the Environment Setting 2 register

bit	Abbreviation	Contents
7 to 0	POM	Set all to "1." (board specifications)
8	EINF	(Reserved: Always set "0.")
10 to 9	EIM1 to 0	Encoder A/B-phase input specification b10,9: 00, 01, 10, 11 Multiplier: x1, x2, x4, UP/DOWN
11	EDIR	Encoder input count polarity (0: Phase A lead 1: Phase B lead)
12	EZL	Encoder Z-phase signal input polarity (0: Falling edge 1: Rising edge)
13	PINF	(Reserved: Always set "0.")
15 to 14	PIM1 to 0	Pulsar A/B-phase input specification b10,9: 00, 01, 10, 11 Multiplier: x1, x2, x4, UP/DOWN
16	PDIR	Pulsar input count polarity (0: Phase A lead 1: Phase B lead)
17	EOFF	Encoder input mask
18	POFF	Pulsar input mask
19	Undefined	Undefined (Always set "0.")
22 to 20	IDL2 to 0	Idling pulse count setting (0 to 7 pulses)
31 to 23	Undefined	(Always set "0.")

Table 6. 2-18 RENV2: Contents of Environment Setting 2 Register

(3) RENV3: Environment Setting 3 Register (31 bits)

This is a register for Environment Setting 3 mainly to define the origin-returning operation method and the counter operation specification.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	BSYC	CI32	CI31	CI30	CI21	CI20	EZD3	EZD2	EZD1	EZD0	ORM3	ORM2	ORM1	ORM0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	CU3H	CU2H	CU1H	0	CU3B	CU2B	CU1B	0	CU3R	CU2R	CU1R	0	CU3C	CU2C	CU1C

Figure 6. 2-32 RENV3: Bit configuration of the Environment Setting 3 register

(To be continued)

(Continued from previous page)

bit	Abbreviation	Contents			
3 to 0	ORM3 to ORM0	Specifies the origin-returning method.			
		ORG mode	ORM3 to 0	Origin-returning details	
		0	0000	[Sensor origin returning (OLS ON detection)] Constant-speed return: Immediate stop and completion by OLS ON. High-speed return: Deceleration stop and completion by OLS ON (DLS deceleration, OLS origin also possible)	
		1	0001	[Sensor origin returning (OLS detection, followed by getting out of origin, reentry, and completion)] Constant-speed return: Immediate stop by OLS ON, followed by getting out of origin in reverse direction, and completion by next OLS detection. High-speed return: Deceleration stop by OLS ON, followed by getting out of origin in reverse direction, and completion by next OLS detection.	
		2	0010	[Sensor + encoder origin returning (OLS ON detection)] Constant-speed return: OLS ON detection, then immediate stop by Z-phase, and completion. High-speed return: Deceleration by OLS ON, then immediate stop by Z-phase, and completion.	
		3	0011	[Sensor + encoder origin returning (OLS ON detection)] Constant-speed return: OLS ON detection, then immediate stop by Z-phase, and completion. High-speed return: OLS ON detection, then deceleration stop by Z-phase, and completion.	
		4	0100	[Sensor reversal + encoder origin (OLS ON reversal)] Constant-speed return: Immediate stop by OLS ON, then constant-speed operation in reverse direction, and completion by Z-phase detection. High-speed return: Deceleration stop by OLS ON, then constant-speed operation in reverse direction, and completion by Z-phase detection.	
		5	0101	[Sensor high-speed reversal + encoder origin (OLS ON reversal)] Constant-speed return: Immediate stop by OLS ON, then constant-speed operation in reverse direction, and completion by Z-phase detection. High-speed return: Deceleration stop by OLS ON, then high-speed operation in reverse direction, and deceleration stop by Z-phase toward completion.	
		6	0110	[ELS shared sensor origin] Constant-speed return: Immediate stop by ELS ON, followed by getting out of origin at constant speed in reverse direction, and completion by ELS OFF detection. High-speed return: Deceleration stop by ELS ON, followed by getting out of origin at constant speed in reverse direction, and completion by ELS OFF detection.	
		7	0111	[ELS reversal at low-speed + encoder origin] Constant-speed return: Immediate stop by ELS ON, then constant-speed operation in reverse direction, and completion by Z-phase detection. High-speed return: Deceleration stop by ELS ON, then constant-speed operation in reverse direction, and completion by Z-phase detection.	
		8	1000	[ELS reversal at constant-speed + encoder origin] Constant-speed return: Immediate stop by ELS ON, then constant-speed operation in reverse direction, and completion by Z-phase detection. High-speed return: Deceleration stop by ELS ON, then high-speed operation in reverse direction, and deceleration stop by Z-phase toward completion.	
The following are origin-returning methods using CTR2 clear [CTR2 (machine counter) = 0].					
9 to 12	1001 to 1100				
7 to 4	EZD3 to 0	Sets the EZ count value for origin-returning. 0000 (1st) to 1111 (16th)			
9 to 8	CI21 to 20	CTR2 (machine position) count input method	b9,8	00	Encoder (pulsar) input
				01	Command pulse
				10	Encoder (pulsar) input
				000	Output pulse
12 to 10	CI32 to 30	CTR3 (general-purpose/error) count input method	b12-10	001	Encoder (pulsar) input
				010	Encoder (pulsar) input
				011	1/4096 clock of the internal reference clock (40 MHz)
				100	Output pulse and encoder (pulsar) input (Error count)
				101	Output pulse and encoder (pulsar) input (Error count)
13	BSYC	Uses CTR3 when "1: during operation (BSY = L)" is set.			
15,14	Undefined	(Always set "0.")			
16	CU1C	When CLR input OFF → ON	(Command position)	0: Do not reset 1: Reset	
17	CU2C		(Machine position)		
18	CU3C		(General-purpose/error)		
19	Undefined	(Always set "0.")			
20	CU1R	When origin-returning completed	(Command position)	0: Do not reset 1: Reset	
21	CU2R		(Machine position)		
22	CU3R		(General-purpose/error)		
23	Undefined	Undefined (Always set "0.")			
24	CU1B	When backlash compensated	(Command position)	0: Do not use counter 1: Use counter	
25	CU2B		(Machine position)		
26	CU3B		(General-purpose/error)		
27	Undefined	Undefined (Always set "0.")			
28	CU1H	(Command position)	0: Do not use counter 1: Use counter		
29	CU2H	(Machine position)			
30	CU3H	(General-purpose/error)			
31	Undefined	(Always set "0.")			

Table 6. 2-19 RENV3: Contents of Environment Setting 3 Register

(4) RENV4: Environment Setting 4 Register (28 bits)

This is a register for Environment Setting 4 to configure Comparator 1 to 4.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	C2D1	C2D0	C2S2	C2S1	C2S0	C2C1	C2C0	0	C1D1	C1D0	C1S2	C1S1	C1S0	C1C1	C1C0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	LTOF	LTFD	LTM1	LTM0	C3D1	C3D0	C3S3	C3S2	C3S1	C3S0	C3C1	C3C0

Figure 6. 2-33 RENV4: Bit configuration of the Environment Setting 4 register

bit	Abbreviation	Contents					
1,0	C1C1 to 0	Comparator 1: Comparison counter used (Note 1)	b1,0 = 00	b1,0 = 01	b1,0 = 10	b1,0 = 11	
			CTR1(command position)	CTR2(machine position)	CTR3(general-purpose/error)	Comparison-condition mismatch	
4 to 2	C1S2 to 0	Comparator 1: Comparison method (Note 2)	b4-2 = 000	b4-2 = 001	b4-2 = 010	b4-2 = 011	
			Always comparison-condition mismatch	RCMP1 data = Comparison CTR			
				Any count direction	While count increased	While count decreased	
			b4-2 = 100	b4-2 = 101	b4-2 = 110	b4-2 = 111	
		RCMP1 > Comparison CTR	RCMP1 < Comparison CTR	As +SLS (RCMP1<CTR1)	Always comparison-condition mismatch		
6,5	C1D1 to 0	Comparator 1: Process performed upon condition match		b6,5 = 00	b6,5 = 01	b6,5 = 10	
				No process	Immediate stop	Deceleration stop	
7	Undefined	(Always set "0.")					
9,8	C2C1 to 0	Comparator 2: Comparison counter used (Note 2)	b9,8 = 00	b9,8 = 01	b9,8 = 10	b9,8 = 11	
			CTR1 (command position)	CTR2 (machine position)	CTR3 (general-purpose/error)	Comparison-condition mismatch	
12 to 10	C2S2 to 0	Comparator 2: Comparison method (Note 2)	b12-10 = 000	b12-10 = 001	b12-10 = 010	b12-10 = 011	
			Always comparison-condition mismatch	RCMP2 data = Comparison CTR			
				Any count direction	While count increased	While count decreased	
			b12-10 = 100	b12-10 = 101	b12-10 = 110	b12-10 = 111	
		RCMP2 > Comparison CTR	RCMP2 < Comparison CTR	As +SLS (RCMP2>CTR1)	Always comparison-condition mismatch		
14,13	C2D1 to 0	Comparator 2: Process performed upon condition match		b14,13 = 00	b14,13 = 01	b14,13 = 10	
				No process	Immediate stop	Deceleration stop	
15	Undefined	(Always set "0.")					
17,16	C3C1 to 0	Comparator 3: Comparison counter used (Note 3)	b17,16 = 00	b17,16 = 01	b17,16 = 10	b17,16 = 11	
			CTR1 (command position)	CTR2 (machine position)	CTR3 (general-purpose/error)	Comparison-condition mismatch	
21 to 18	C3S3 to 0	Comparator 3: Comparison method (Note 3)	b21-18 = 0001	b21-18 = 0010	b21-18 = 0011	b21-18 = 0011	
			RCMP3 data = Comparison CTR				
				Any count direction	While count increased	While count decreased	
			b21-18 = 0100	b21-18 = 0101	b21-18 = 0110	b21-18 = 0111	
				RCMP3 > Comparison CTR	RCMP3 < Comparison CTR	Setting forbidden	
			b21-18 = 1000	b21-18 = 1001	b21-18 = 1010	b21-18 = 1010	
			Use as synchronous signal output (Note 3)				
	Any count direction	While count increased	While count decreased				
Others: Always comparison-condition mismatch							
23,22	C3D1 to 0	Comparator 3: Process performed upon condition match		b23,22 = 00	b23,22 = 01	b23,22 = 10	
				No process	Immediate stop	Deceleration stop	
25,24	LTM1 to 0	LTM1 to 0 Counter (CTR 1 to 3) latch timing	b25,24 = 00	b25,24 = 01	b25,24 = 10	b25,24 = 11	
			Latch OFF → ON	OLS input	CMP2 condition match	CMP3 condition match	
26	LTFD	1: Latch current speed data instead of CTR3					
27	LTOF	1: Hardware latch disabled (only software latch enabled)					
31 to 28	Undefined	(Always set "0.")					

Table 6. 2-20 RENV4: Contents of Environment Setting 4 Register

- Note 1. When you select CTR3 (set as Error) as a comparison counter, the absolute value of the count value is compared with the comparator data. (Absolute value range: 0 to 32,767)
- Note 2. When you set "C1S2 to 0 = 110 (+ Soft-limit)" or "C2S2 to 0 = 110 (- soft-limit)", select CTR1 (command position) as a comparison counter. A process performed by the soft-limit is a stop regardless of the settings in "C1D1 to 0" and "C2D1 to 0."
- Note 3. When you set "C3S3 to 0 = 1000 to 1010 (synchronous signal output)", select CTR3 (set as General-Purpose) as a comparison counter. The other counters cannot be selected. Use a positive value as the comparator setting value.

(5) RENV5: Environment Setting 5 Register (32 bits)

This is a register for Environment Setting 5, mainly to specify travel-distance compensation data.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSTP	0	0	ADJ	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PMG4	PMG3	PMG2	PMG1	PMG0	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0

Figure 6. 2-34 RENV5: Bit configuration of the Environment Setting 5 register

bit	Abbreviation	Contents
11 to 0	BR11 to 0	Backlash compensation value setting [Setting range: 0 to 4,095]
12	ADJ	Travel-distance compensation method (0: compensation function OFF, 1: backlash compensation)
14,13	Undefined	(Always set "0.")
15	PSTP	(Always set "0.")
26 to 16	PD10 to 0	Division ratio setting in pulsar input The division ratio is (set value)/2048. [Setting range: 0 to 2,047] With "0", the frequency-division circuit is OFF. (= 2048/2048)
31 to 27	PMG4 to 0	Multiplier setting in pulsar input Multiplier is: x (set value + 1). [Setting range: 0 to 31]

Table 6. 2-21 RENV5: Contents of Environment Setting 5 Register

(6) RENV6: Environment Setting 6 Register (32 bits)

This is a register for Environment Setting 6 to specify time for the vibration suppression function.

This function is ON when both RT and FT values are other than 0.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0

Figure 6. 2-35 RENV6: Bit configuration of the Environment Setting 6 register

bit	Abbreviation	Contents
15 to 0	RT15 to 0	Specifies RT time shown in the figure below. [Setting range: 0 to 65,535] Unit: 25nsec x 64 (1.6μs)
31 to 16	FT15 to 0	Specifies FT time shown in the figure below. [Setting range: 0 to 65,535] Unit: 25nsec x 64 (1.6μs)

Table 6. 2-22 RENV6: Contents of Environment Setting 6 register

The dashed lines in the figure indicate pulses added by the vibration suppression function.

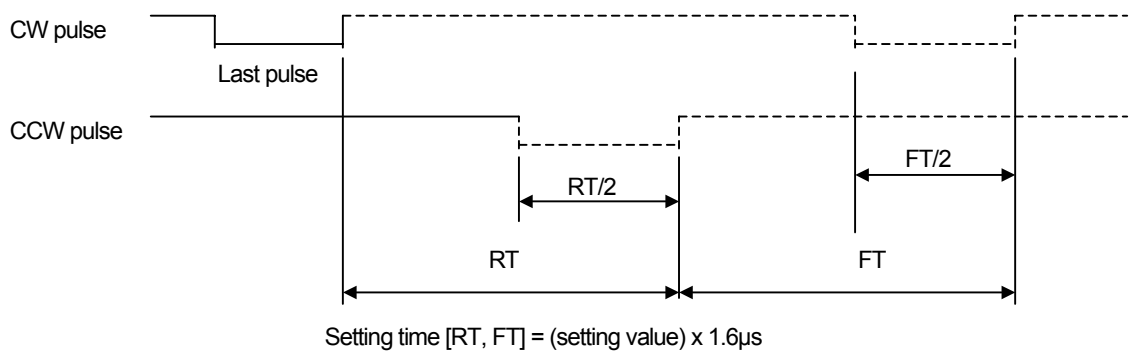


Figure 6. 2-36 Reverse pulses for vibration suppression

7. Software

This chapter describes the following points about the accompanying software.

The same software is used in HPCI-MNT520M and MCPCI-MNT720M.

■ For Windows

- ◇How to install and uninstall the driver
- ◇Access to the board, and board IDs
- ◇How to use driver functions
- ◇Driver functions reference
- ◇Library functions reference

7.1 Software configuration

7.1.1 Configuration of the software for Windows

Our software is available in three types: library functions, driver functions, and a device driver.

The device driver is software to input and output data for motionCAT master board HPCI-MNT520M or HPCI-MNT720M.

The driver functions are “device driver interface library,” namely input/output functions to serve as a bridge between the application and the device driver. The functions are provided as a DLL of Win32 API functions.

The library functions, which are made up of driver functions, control basic operations including the initialization of motion module HMG-Px, origin-returning, and positioning. The source files of those library functions are provided to allow you to modify the functions as needed. The library functions for respective development languages are based on the same kind of specifications.

(1) Device driver for Windows

- ◇For Windows Vista/XP and Windows 2000 ... hm520wdm.sys

(2) Device driver interface DLL for Windows

Various functions included in the device driver interface DLL are referred to as “driver functions.

- ◇For Windows Vista/XP and Windows 2000 ... himnt520.dll

(3) Library functions for Windows

- ◇Library function source file for VC++ ... hpxl1a.c
Library function header file for VC++ ... hpxl1a.h
- ◇Library function source file for VB6.0 ... hpxl1a.bas
- ◇Library function source file for VB.NET ... hpxl1a.vb

The relationships between the application program and those pieces of software are shown below.

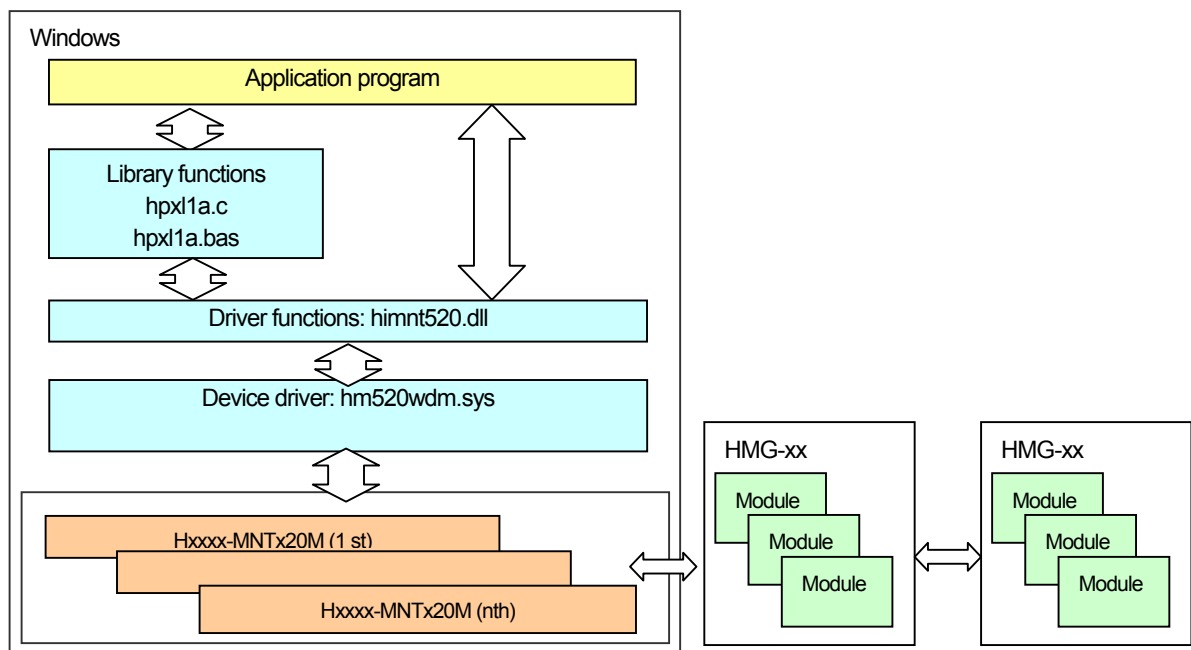


Figure 7. 1-1 Software relationships

7.2 Installing and uninstalling the device driver for Windows

7.2.1 Installation on Windows Vista

- (1) After making sure that the PC main unit is turned off, connect an MNT to the PCI (CompactPCI) bus slot of the PC main unit. Then, turn on the PC main unit to start Windows.
- (2) When Windows is started, the system detects the MNT and automatically displays a screen for installing the needed device driver. Insert the accompanying CD into the CD drive.
- (3) The system requests you to insert the disk. Click [Next].
- (4) A warning message appears, saying "Windows can't verify the publisher of this driver software." Select [Install this driver software anyway] to continue with the installation.
Follow the directions afterward given by the system to complete the installation.

7.2.2 Installation on Windows XP

- (1) After making sure that the PC main unit is turned off, connect an MNT to the PCI (CompactPCI) bus slot of the PC main unit. Then, turn on the PC main unit to start Windows.
- (2) When Windows is started, the system detects the MNT and automatically displays a screen for installing the needed device driver. Insert the accompanying CD into the CD drive.
- (3) Check [Install the software automatically (Recommended)].
- (4) A warning "has not passed Windows Logo testing to verify its compatibility with Windows XP" appears, but select [Continue Anyway] to continue with the installation. Follow the directions afterward given by the system to complete the installation.

7.2.3 Installation on Windows 2000

- (1) After making sure that the PC main unit is turned off, connect an MNT to the PCI (CompactPCI) bus slot of the PC main unit. Then, turn on the PC main unit to start Windows.
- (2) When Windows is started, the system detects the MNT and automatically displays a screen for installing the needed device driver.
- (3) The system requests you to specify the installation source directory. Insert the accompanying CD into the CD drive.
- (4) Check [Specify a location].
- (5) Specify "D:¥WDM" (if the CD drive is drive D).
Follow the directions afterward given by the system to complete the installation.

7.2.4 Uninstalling the device driver for Windows

- (1) Insert the accompanying CD into the CD drive.
- (2) Start Explorer to execute D:¥mn520uin.exe (if the CD drive is drive D).

7.3 Using two or more master boards

This section describes how to install more than one MNT boards in a computer, with each board having each external connection.

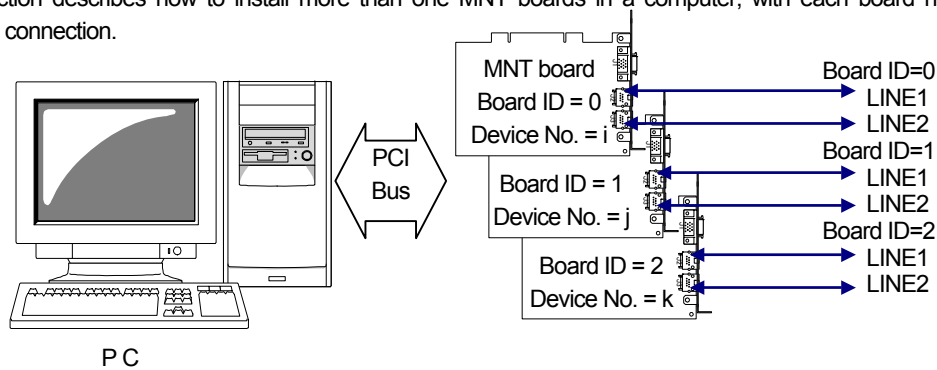


Figure 7. 3-1 Using more than one board

(1) Board device number

In the PCI Bus system, the BIOS manages board addresses. The BIOS gives numbers to the boards according to the slot in which each board is placed. Those numbers are called device numbers.

(2) Using board IDs

Since device numbers are allocated by the BIOS, it may be difficult to recognize the relationships between the boards and the slots from outside. To help you relate the individual boards to the software, the MNT board is equipped with "the board ID setting rotary switch." The board ID range is 0h to Fh (0 to 15), allowing the use of up to 16 MNT boards.

See "3.4.3 Master board setting" for how to set board IDs.

7.4 Preparation for application building

(1) Application building by Visual C++ (6.0 or above)

Add the following files to the project.

■Files to be added to the project

- ◇For library functions ... **hpx11a.c** ... Library function source file for VC++ applications
Library function code is written in the C language.
- ◇For driver functions ... **himnt520.lib** ... Driver function import library

■Include files

- ◇For library functions ... **hpx11a.h** ... Library function header file for VC++ applications
"Included" in the "hpx11a.c" file
(It is possible to build an application without library functions.)
- ◇For driver functions ... **himnt520.h** ... Header file for combining driver functions
"Included" in the "hpx11a.h" file

(2) Application building by Visual Basic (6.0)

Add the following files to the project.

- ◇For library functions ... **hpx11a.bas** ... Library standard module file for VB 6.0 applications
This file contains the descriptions of the library function definitions and code.
(It is possible to build an application without library functions.)
- ◇For driver functions ... **himnt520.bas** ... DLL function definition standard module file for driver interfaces
This file contains the descriptions of external function declarations as well as user defined declarations.

(3) Application building by Visual Basic .NET

Add the following files to the project.

- ◇For library functions ... **hpx11a.vb** ... Library source file for VB.NET applications
This file contains the descriptions of the library function definitions and code.
(It is possible to build an application without library functions.)
- ◇For driver functions ... **himnt520.vb** ... DLL function definition source file for driver interfaces
This file contains the descriptions of external function declarations as well as user defined declarations.

7.5 How to access the board

The driver and library functions allow the control of more than one master device.

To access the MNT you want to control, open this MNT first then obtain the device handle. The hardware resources (*) necessary for opening MNTs are referred to as "PCI device information."

* Hardware resources Board IDs, board base addresses, interrupt numbers, etc.

7.5.1 Data structures for board (device) recognition

The following data structures (32 bytes) are available for recognizing boards.

(1) Windows version of the data structures for board (device) recognition

[C language: Visual C++]

```
typedef struct _HMNT520INFO {           // Device information
    DWORD      dwBusNumber;             // Bus number
    DWORD      dwDeviceNumber;         // Device number
    DWORD      dwBaseAddress2;         // Base address 2
    DWORD      dwBaseAddress3;         // Base address 3
    DWORD      dwBaseAddress4;         // Base address 4
    DWORD      dwIrqNo;                 // IRQ number
    DWORD      dwNumber;                // Management number
    DWORD      dwBoardID;               // Board ID (0 to 15)
} HMNT520INFO, //PHMNT520INFO;
```

[Visual Basic]

Public Type HMNT520INFO

dwBusNumber	As Long	' Bus number
dwDeviceNumber	As Long	' Device number
dwBaseAddress2	As Long	' Base address 2
dwBaseAddress3	As Long	' Base address 3
dwBaseAddress4	As Long	' Base address 4
dwIrqNo	As Long	' IRQ number
dwNumber	As Long	' Management number
dwBoardID	As Long	' Board ID (0 to 15)

End Type

[Visual Basic.NET]

Public Structure HMNT520INFO

Dim dwBusNumber	As Integer	' Bus number
Dim dwDeviceNumber	As Integer	' Device number
Dim dwBaseAddress2	As Integer	' Base address 2
Dim dwBaseAddress3	As Integer	' Base address 3
Dim dwBaseAddress4	As Integer	' Base address 4
Dim dwIrqNo	As Integer	' IRQ number
Dim dwNumber	As Integer	' Management number
Dim dwBoardID	As Integer	' Board ID (0 to 15)

End Structure

7.5.2 Board access preparatory procedure and end processing

■ Preparatory Procedure

(1) Opening devices board by board

First, obtain the number of MNTs, and prepare the same number of data structures for board recognition. Next, obtain the MNT device information, and pass the device information on the MNT you want to control to the device opening function. Then, the MNT is opened.

Also, the device opening function returns the device handle value for accessing the MNT.

When using two or more MNTs, perform the above process for each MNT.

- ◆ `mnt520_GetMstBrdCount()` ... Gets the number of MNTs.
- ◆ `mnt520_GetMstBrdInfo()` ... Gets MNT's PCI device information.
- ◆ `mnt520_OpenMstBrd()` ... Performs open processing.

(2) Initial setting

After opening the MNTs successfully, initiate system communication to automatically set local device information. Set initial values of port data to begin cyclic communication. Since data communication becomes also possible once cyclic communication has been started, initialize the local devices.

- ◆ `mnt520_rSysLclInfo()` ... Gets local device (DIO, motion) information.
- ◆ `mnt520_wCenCmd()` ... Writes a command to the center device.
- ◆ `mnt520_rCenMsts()` ... Reads the center main status from the center device.
- ◆ `mnt520_wIoPortB()` ... Writes to the DIO device port.
- ◆ `mnt520_wIoPortW()` ... Writes to the DIO device port.
- ◆ `mnt520_wLclSetInt()` ... Sets the Interrupt-On-Port-Input-Change for the local device port.
- ◆ `mnt520_wPclPortB()` ... Writes to the motion device port.
- ◆ `mnt520_wPclSetInt()` ... Sets the Interrupt-On-Port-Input-Change for the motion device port.
- ◆ `mnt520_wPclReg()` ... Writes to the motion device register.

(3) Operation

After the initial setting, control the local devices through data and cyclic communications.

- ◆ `mnt520_rIoPortB()` ... Reads from a DIO device port.
- ◆ `mnt520_wIoPortB()` ... Writes to a DIO device port.
- ◆ `mnt520_rIoPortW()` ... Reads from a DIO device port.
- ◆ `mnt520_wIoPortW()` ... Writes to a DIO device port.
- ◆ `mnt520_rLclInt()` ... Reads the Interrupt-On-Port-Input-Change of a local device port.
- ◆ `mnt520_wLclInt()` ... Resets the Interrupt-On-Input-Change of the a local device port.
- ◆ `mnt520_rPclPort()` ... Reads from a motion device port
- ◆ `mnt520_wPclPort()` ... Writes to a motion device port
- ◆ `mnt520_rPclMsts()` ... Reads the Motion Main Status of a motion module
- ◆ `mnt520_wPclCmd()` ... Writes a command to a motion device
- ◆ `mnt520_rPclReg()` ... Reads from a motion device register
- ◆ `mnt520_wPclReg()` ... Writes to a motion device register

■ End processing

(4) Closing all the opened devices

To quit the application after the end of all the operations, perform ending process on a system basis, and then execute the "closing process" for the opened devices.

If there are two or more MNTs, perform this process for each MNT.

- ◆ `mnt520_CloseMstBrd()` ... Performs closing process for MNT520.

7.6 Driver functions

Driver functions are the “device driver interface library”, namely input/output functions to serve as a bridge between the application and the device driver. The functions are provided as a DLL of Win32 API functions.

7.6.1 List of driver functions

No	Function name	Role	Remarks
1	mnt520_GetMstBrdCount()	Gets the number of MNTs.	MNT520
2	mnt520_GetMstBrdInfo()	Gets MNTPCI device information.	
3	mnt520_OpenMstBrd()	Opens an MNT.	
4	mnt520_CloseMstBrd()	Closes an MNT.	
5	mnt520_rCenMsts()	Reads the Center Main Status.	Center device's control functions
6	mnt520_wCenCmd()	Writes the Center Device command.	
7	mnt520_rCenIsts()	Reads the Center Interrupt Status.	
8	mnt520_rCenBuf()	Reads from center device's I/O buffer.	
	mnt520_wCenBuf()	Writes to center device's I/O buffer.	
9	mnt520_rCenRFIFo()	Reads from center device's receiving FIFO.	
	mnt520_wCenSFIFo()	Writes to center device's sending FIFO.	
10	mnt520_rLclCycErr()	Reads the Cyclic Communication Error Flag.	Cyclic communication
	mnt520_wLclCycErr()	Resets the Cyclic Communication Error Flag.	
11	mnt520_rLclInfo()	Reads local deice (DIO, motion) information	System communication
	mnt520_wLclInfo()	Writes local deice (DIO, motion) information	
12	mnt520_rLclSetInt()	Reads local device's Input-Port Change Flag Setting Status.	Cyclic communication
	mnt520_wLclSetInt()	Writes local device's Input-Port Change Flag Setting.	
13	mnt520_rLclInt()	Reads local device's Input-Port Change Flag.	
	mnt520_wLclInt()	Resets local device's Input-Port Change Flag.	
14	mnt520_rIoPortB()	Reads 1-byte from DIO device's specified port	
	mnt520_wIoPortB()	Writes 1-byte to DIO device's specified port	
	mnt520_rIoPortW()	Reads 2-byte from DIO device's specified port	
	mnt520_wIoPortW()	Writes 2-byte to DIO device's specified port	
15	mnt520_rPclPort()	Reads the output status of motion device's Generic Output Port.	
	mnt520_wPclPort()	Writes the output status of motion device's Generic Output Port.	
16	mnt520_rPclMsts()	Reads the Motion Main Status.	Data communication
17	mnt520_wPclCmd()	Writes the Motion Device Control command	
18	mnt520_rPclReg()	Reads from a motion device register	
	mnt520_wPclReg()	Writes to a motion device register	
19	mnt520_rOptPortB()	Reads 1-byte from the option port	MNT520
	mnt520_rOptPortW()	Reads 2-byte from the option port	
	mnt520_wOptPortB()	Writes 1-byte to the option port	
	mnt520_wOptPortW()	Writes 2-byte to the option port	
	mnt520_rCenPortW()	Reads 2-byte from center device's specified port	
	mnt520_wCenPortW()	Writes 2-byte to center device's specified port	

Table 7. 6-1 List of driver functions

7.6.2 Return values of driver functions

If a return value of a function is abnormal (value other than "0") in using library functions, handle the abnormal event appropriately as shown below.

No	Return value		Abnormal event and check item
	Name	Hex	
1	NO_ERROR	00000000	Normal No abnormal event occurred
2	NOT_FOUND	00000001	Absence of the device driver ⊙The device driver has not been installed. ⊙The device driver is not located in the correct directly.
3	ALREADY_OPENED	00000002	Attempted to open a device that is already open. ⊙The Open command has been sent to an already opened device. ◇Use opened devices until they are closed. (Multiple opening forbidden) ⊙When using two or more boards, check to see that the device information for the opened devices has been updated.
4	INSUFFICIENT_MEMORY	00000004	Insufficient memory space for storing device information. ⊙Insufficient memory space for the application. ◇The main memory space in the computer is insufficient. ⊙Insufficient system resources (memory for the OS) ◇Too many applications have been started. ◇Too many windows are opened at the same time.
5	INVALID_HANDLE	00000008	An invalid device handle was specified. ⊙The specified handle is not the one obtained when the device was opened. ⊙The specified device has already been closed.
6	NOT_READY	00000010	Device's I/O ports cannot be used. ⊙There is no I/O port inside the device (board). (The device may be faulty. Contact us.)
7	ILLEGAL_DEVICE	00000020	Board's individual information is invalid. ⊙The device information is proper, but it does not match the board function. (The device may be faulty. Contact us.)
8	ILLEGAL_PARAM	00000100	The value of the function argument is invalid. ◇Check the values of other arguments.
9	CYC_COM_ERROR	00010000	Cyclic communication error ⊙The power to a slave is off. ⊙A communication cable is disconnected. ⊙A communication cable is under the influence of noise.
10	CYC_COM_STOP	00020000	Cyclic communication is stopped. ⊙A data communication was attempted when a cyclic communication was not enabled.
11	DATA_COM_ERROR	00040000	Data communication error
12	COM_OTHER_ERROR	00080000	Other communication errors ⊙Check the Center Interrupt Status for details.
13	OTHER_ERROR	80000000	Other errors

Table 7. 6-2 Return values of driver functions

Please note that errors originating from devices are not included in the error information above. Identify the cause of each error clearly and take proper measures accordingly.

7.6.3 Details on driver functions

This section provides descriptions in VC++ format. For VB, VB.NET, their corresponding data types are shown in the table below.

- Data types of arguments, hexadecimal notation

Language	VC++ language	VB	VB.NET
Data type	8bit	BYTE	Byte
	16bit	WORD	Integer
	32bit	DWORD	Long
Examples of data	0x0000	&H0	&H0
	0x1000	&H1000	&H1000

(1) mnt520_GetMstBrdCount () Get the number of MNTs.		
Role		Obtains the number of MNTs currently installed in the system.
Win VC++	Format	DWORD WINAPI mnt520_GetMstBrdCount(DWORD* count);
	Arguments	DWORD* count ... Address where the number of MNTs obtained is stored
	Return values	'0' indicates normal processing. For values other than '0', see "7.6.2 Return values of driver functions."
	Example	DWORD count; // The number of MNTs DWORD ret; // Return value ret = mnt520_GetMstBrdCount(&count);
(2) mnt520_GetMstBrdInfo () Gets the PCI device information of MNTs.		
Role		Obtains the PCI device information of MNTs currently installed in the system.
Win VC++	Format	DWORD WINAPI mnt520_GetMstBrdInfo (DWORD* count, HMNT520INFO* hInfo);
	Arguments	DWORD* count ... Address where the number of MNTs obtained is stored HMNT520INFO* hInfo ... The initial address of the PCI device information for the MNT obtained.
	Return values	'0' indicates normal processing. For values other than '0', see "7.6.2 Return values of driver functions."
	Example	It is assumed that two pieces of MNT520 are installed in the computer. DWORD count=2; // Two MNTs DWORD ret; // Return value HMNT520INFO hMntInfo[2]; // PCI device information (for two pieces) ret = mnt520_GetMstBrdInfo(&count,&hMntInfo[0]);
(3) mnt520_OpenMstBrd () Opens MNTs.		
Role		Opens an MNT and obtains the device handle of this MNT.
Win VC++	Format	DWORD WINAPI mnt520_OpenMstBrd (DWORD* hDev, HMNT520INFO* hInfo);
	Arguments	DWORD* hDev ... Address where the device handle obtained is stored HMNT520INFO* hInfo ... The initial address of the PCI device information for the MNT to be opened.
	Return values	'0' indicates normal processing. For values other than '0', see "7.6.2 Return values of driver functions."
	Example	DWORD ret; // Return value DWORD hDev; // Device handle HMNT520INFO hMntInfo[2]; // PCI device information (for two pieces) // Open 1st MNT. ret = mnt520_OpenMstBrd(&hDev,&hMntInfo[0]);
(4) mnt520_CloseMstBrd () Closes an MNT.		
Role		Closes an MNT.
Win VC++	Format	DWORD WINAPI mnt520_CloseMstBrd(DWORD hDev);
	Arguments	DWORD hDev ... Device handle
	Return values	'0' indicates normal processing. For values other than '0', see "7.6.2 Return values of driver functions."
	Example	DWORD ret; // Return value DWORD hDev; // Device handle ret = mnt520_CloseMstBrd(hDev);

(5) mnt520_rCenMsts () Reads the Center Main Status.		
Role		Reads the Center Main Status
Win VC++	Format	DWORD WINAPI mnt520_rCenMsts (DWORD hDev,WORD wLine,WORD* wSts);
	Arguments	DWORD hDev ... Device handle WORD wLine ... Line number (0: Line 1, 1:Line 2) WORD* wSts ... Center Main Staaus
	Return values	'0' indicates normal processing. For values other than '0', see "7.6.2 Return values of driver functions."
	Example	DWORD ret; // Return value DWORD hDev; // Device handle WORD sts; // Center Main Status // Read Center Main Status of Line 1. ret = mnt520_rCenMsts(hDev,0,&sts);

(6) mnt520_wCenCmd () Writes a command to the center device.		
Role		Writes a command to the center device.
Win VC++	Format	DWORD WINAPI mnt520_wCenCmd (DWORD hDev,WORD wLine,WORD wCmd);
	Arguments	DWORD hDev ... Device handle WORD wLine ... Line number (0: Line 1, 1:Line 2) WORD wCmd ... Center Device command code
	Return values	'0' indicates normal processing. For values other than '0', see "7.6.2 Return values of driver functions."
	Example	DWORD ret; // Return value DWORD hDev; // Device handle // Start cyclic communication by the center device on Line 1. ret = mnt520_rCenCmd(hDev,0,0x3000);

(7) mnt520_wCenIsts () Reads the Center Interrupt Status.		
Role		Reads the Center Interrupt Status.
Win VC++	Format	DWORD WINAPI mnt520_rCenIsts (DWORD hDev,WORD wLine,WORD* wSts);
	Arguments	DWORD hDev ... Device handle WORD wLine ... Line number (0: Line 1, 1:Line 2) WORD* wSts ... Center Interrupt Status
	Return values	'0' indicates normal processing. For values other than '0', see "7.6.2 Return values of driver functions."
	Example	DWORD ret; // Return value DWORD hDev; // Device handle WORD sts; // Center Interrupt Status // Read the Center Interrupt Status on Line 1. ret = mnt520_rCenIsts(hDev,0,&sts);

(8) mnt520_rCenBuf ()		Reads from the I/O buffer of the center device.
mnt520_wCenBuf ()		Writes to the I/O buffer of the center device
Role		Reads or writes data from/to the I/O buffer of the center device.
Win VC++	Format	DWORD WINAPI mnt520_rCenBuf (DWORD hDev,WORD wLine,WORD* wInBuf); DWORD WINAPI mnt520_wCenBuf (DWORD hDev,WORD wLine,WORD wOutBuf);
	Arguments	DWORD hDev ... Device handle WORD wLine ... Line number (0: Line 1, 1:Line 2) WORD* wInBuf ... Data to be read from center device's I/O buffer WORD wOutBuf ... Data to be written to center device's I/O buffer
	Return values	'0' indicates normal processing. For values other than '0', see "7.6.2 Return values of driver functions."
	Example	DWORD ret; // Return value DWORD hDev; // Device handle WORD wBuf; // Data to be read from center device's I/O buffer // Read from center device's I/O buffer on Line 1. ret = mnt520_rCenBuf(hDev,0,&wBuf);

(9) mnt520_rCenRFifo ()		Reads from the receiving FIFO of the center device.
mnt520_wCenSFifo ()		Writes to the sending FIFO of the center device
Role		Reads data from the receiving FIFO of the center device. Writes data to the sending FIFO of the center device.
Win VC++	Format	DWORD WINAPI mnt520_rCenRFifo (DWORD hDev,WORD wLine,WORD* wReceive); DWORD WINAPI mnt520_wCenSFifo (DWORD hDev,WORD wLine,WORD wSend);
	Arguments	DWORD hDev ... Device handle WORD wLine ... Line number (0: Line 1, 1:Line 2) WORD* wReceive ... Data to be read from center device's receiving FIFO WORD wSend ... Data to be written to center device's sending FIFO
	Return values	'0' indicates normal processing. For values other than '0', see "7.6.2 Return values of driver functions."
	Example	DWORD ret; // Return value DWORD hDev; // Device handle WORD wData; // Data to be read from the receiving FIFO // Read from center device's receiving FIFO on Line 1. ret = mnt520_rCenRFifo(hDev,0,&wData);

(10) mnt520_rLclCycErr () Reads a Cyclic Communication Error Flag. mnt520_wLclCycErr () Resets a Cyclic Communication Error Flag.		
Role	Reads a Cyclic Communication Error Flag. Resets a Cyclic Communication Error Flag.	
Win VC++	Format	DWORD WINAPI mnt520_rLclCycErr (DWORD hDev,WORD wLine,WORD wMidPm,WORD* wFlag); DWORD WINAPI mnt520_wLclCycErr (DWORD hDev,WORD wLine,WORD wMidPm,WORD wFlag);
	Arguments	DWORD hDev ... Device handle WORD wLine ... Line number (0: Line 1, 1:Line 2) WORD wMidPm ... Parameter to specify module IDs (MIDs) (0: MIDs = 0 to 15, 1: MIDs = 16 to 31, 2: MIDs = 32 to 47, 3: MID = 49 to 64) WORD* wFlag ... Error flag WORD wFlag ... Error flag
	Return values	'0' indicates normal processing. For values other than '0', see "7.6.2 Return values of driver functions."
	Example	DWORD ret; // Return value DWORD hDev; // Device handle WORD wFlag; // Cyclic Communication Error Flag // Read the Cyclic Communication Error Flag for MIDs 0 to 15 on Line 1. ret = mnt520_rLclCycErr(hDev,0,0,&wFlag);
	Remarks	Write the Error Flag data that has been read to reset the Error Flag.

(11) mnt520_rLclInfo () Reads local device information. mnt520_wLclInfo () Writes local device information.		
Role	Reads information on a local device connected to the MNT. Write information on a local device connected to the MNT.	
Win VC++	Format	DWORD WINAPI mnt520_rLclInfo (DWORD hDev,WORD wLine,WORD wMid,BYTE* byData); DWORD WINAPI mnt520_wLclInfo (DWORD hDev,WORD wLine,WORD wMid,BYTE byData);
	Arguments	DWORD hDev ... Device handle WORD wLine ... Line number (0: Line 1, 1:Line 2) WORD wMid ... Module ID (0: MID = 0, ... , 63: MID = 63) BYTE* byData ... Local device information BYTE byData ... Local device information
	Return values	'0' indicates normal processing. For values other than '0', see "7.6.2 Return values of driver functions."
	Example	DWORD ret; // Return value DWORD hDev; // Device handle BYTE byData; // Local device information // Read the information on the local device whose ID is 0 on Line 1 ret = mnt520_rLclInfo(hDev,0,0,&byData);
	Remarks	Although the module ID can be specified in the range of 0 to 63, one Line actually allows connecting a maximum of 32 modules.

(12) mnt520_rLclSetInt () Reads Interrupt-On-Input-Change Setting Statuses of local device's ports. mnt520_wLclSetInt () Writes Interrupt-On-Input-Change Setting for local device's ports.																																																				
Role	Reads the settings for issuing an interrupt when a change of input occurs at specified local device's ports. Writes settings for issuing an interrupt when a change of input occurs at specified local device's ports.																																																			
Win VC++	Format	DWORD WINAPI mnt520_rLclSetInt (DWORD hDev,WORD wLine,WORD wMidPm,WORD* wData); DWORD WINAPI mnt520_wLclSetInt (DWORD hDev,WORD wLine,WORD wMidPm,WORD wData);																																																		
	Arguments	DWORD hDev ... Device handle WORD wLine ... Line number (0: Line 1, 1:Line 2) WORD wMidPm ... Parameter to specify module IDs (MIDs) (0: MIDs = 0 to 3, 1: MIDs = 4 to 7 2:MIDs = 8 to 11, 3: MIDs = 12 to 15, 4:MIDs = 16 to 19, 5: MIDs = 20 to 23, 6:MIDs = 24 to 27, 7: MIDs = 28 to 31, 8:MIDs = 32 to 35, 9: MIDs = 36 to 39, 10:MIDs = 40 to 43,11: MIDs = 44 to 47, 12:MIDs = 48 to 51,13: MIDs = 52 to 55, 14:MIDs = 56 to 59,15: MIDs = 60 to 63) WORD* wData ... Interrupt-On-Input-Change Setting Statuses of four local devices (16 ports) WORD wData ... Interrupt-On-Input-Change Setting data for four local devices (16 ports)																																																		
	Return values	'0' indicates normal processing. For values other than '0', see "7.6.2 Return values of driver functions."																																																		
	Example	DWORD ret; // Return value DWORD hDev; // Device handle // Set Interrupt-On-Input-Change Setting to monitor Port 0 of Device 0 on Line 1. ret = mnt520_wLclSetInt(hDev,0,0,0x0001);																																																		
	Remarks	1. Interrupt-On-Input-Change Setting data Example. DIO device, wMidPm = 0 <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td></td> <td colspan="4" style="text-align: center;">Device 3</td> <td colspan="4" style="text-align: center;">Device 2</td> <td colspan="4" style="text-align: center;">Device 1</td> <td colspan="4" style="text-align: center;">Device 0</td> </tr> <tr> <td style="text-align: right;">bit</td> <td>15</td><td>14</td><td>13</td><td>12</td> <td>11</td><td>10</td><td>9</td><td>8</td> <td>7</td><td>6</td><td>5</td><td>4</td> <td>3</td><td>2</td><td>1</td><td>0</td> </tr> <tr> <td></td> <td></td><td></td><td></td><td></td> <td></td><td></td><td></td><td></td> <td></td><td></td><td></td><td></td> <td></td><td></td><td></td><td></td> </tr> </table> <div style="margin-left: 200px;"> Port 3 → Port 2 → Port 1 → Port 0 → </div> 2. Although the module ID can be specified in the range of 0 to 63, one Line actually allows connecting a maximum of 32 modules.		Device 3				Device 2				Device 1				Device 0				bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																
	Device 3				Device 2				Device 1				Device 0																																							
bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																				

(13) mnt520_rLclnt () mnt520_wLclnt ()		Reads an Input-Port Change Flag for local devices Resets an Input-Port Change Flag for local devices																																																		
Role		Reads an Input-Port Change Flag for specified local devices. Resets an Input-Port Change Flag for specified local devices.																																																		
Win VC++	Format	DWORD WINAPI mnt520_rLclnt (DWORD hDev,WORD wLine,WORD wMidPm,WORD* wFlag); DWORD WINAPI mnt520_wLclnt (DWORD hDev,WORD wLine,WORD wMidPm,WORD wFlag);																																																		
	Arguments	DWORD hDev ... Device handle WORD wLine ... Line number (0: Line 1, 1:Line 2) WORD wMidPm ... Parameter to specify module IDs (MIDs) (0: MIDs = 0 to 3, 1: MIDs = 4 to 7 2:MIDs = 8 to 11, 3: MIDs = 12 to 15, 4:MIDs = 16 to 19, 5: MIDs = 20 to 23, 6:MIDs = 24 to 27, 7: MIDs = 28 to 31, 8:MIDs = 32 to 35, 9: MIDs = 36 to 39, 10:MIDs = 40 to 43, 11: MIDs = 44 to 47, 12:MIDs = 48 to 51, 13: MIDs = 52 to 55, 14:MIDs = 56 to 59, 15: MIDs = 60 to 63) WORD* wFlag ... Input-Port Change Flag for four local devices (16 ports) WORD wFlag ... Input-Port Change Flag for four local devices (16 ports)																																																		
	Return values	'0' indicates normal processing. For values other than '0', see "7.6.2 Return values of driver functions."																																																		
	Example	DWORD ret; // Return value DWORD hDev; // Device handle WORD wFlag; // Input Change Flag BYTE byData; // Input port data // Set a Flag to monitor input changes at Ports 0 to 3 of Devices 0 to 3 on Line 1. ret = mnt520_rLclnt(hDev,0,0,&wFlag); if(wFlag & 0x0001){ // An input change occurred at Port 0 of Device 0. // Read Port 0 of Device 0. ret = mnt520_rIoPortB(hDev,0,0,0,&byData); ret = mnt520_wLclnt(hDev,0,0,wFlag); // Reset the Flag. }																																																		
	Remarks	1. To reset the Input-Port Change Flag, write the Input-Port Change Flag data that has been read. 2. Input-Port Change Flag Example: DIO device, wMidPm = 0 <table border="1" style="margin-left: 40px;"> <tr> <td></td> <td colspan="4">Device 3</td> <td colspan="4">Device 2</td> <td colspan="4">Device 1</td> <td colspan="4">Device 0</td> </tr> <tr> <td>bit</td> <td>15</td><td>14</td><td>13</td><td>12</td> <td>11</td><td>10</td><td>9</td><td>8</td> <td>7</td><td>6</td><td>5</td><td>4</td> <td>3</td><td>2</td><td>1</td><td>0</td> </tr> <tr> <td></td> <td></td><td></td><td></td><td></td> <td></td><td></td><td></td><td></td> <td></td><td></td><td></td><td></td> <td></td><td></td><td></td><td></td> </tr> </table> <div style="margin-left: 40px; margin-top: 10px;"> Port 3 Port 2 Port 1 Port 0 </div> 3. Although the module ID can be specified in the range of 0 to 63, one Line actually allows connecting a maximum of 32 modules.		Device 3				Device 2				Device 1				Device 0				bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																
	Device 3				Device 2				Device 1				Device 0																																							
bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																				

(14) mnt520_rloPortB ()		Reads 1-byte from a DIO device port
mnt520_rloPortW ()		Reads 2-byte from a DIO device port
mnt520_wloPortB ()		Writes 1-byte to a DIO device port
mnt520_wloPortW ()		Writes 2-byte to a DIO device port
Role		Reads data from a specified port of a DIO device. Writes data to a specified port of a DIO device.
Win VC++	Format	DWORD WINAPI mnt520_rloPortB (DWORD hDev,WORD wLine,WORD wMid,WORD wPrtNo, BYTE* byData); DWORD WINAPI mnt520_rloPortW (DWORD hDev,WORD wLine,WORD wMid,WORD wPrtNo, WORD* wData); DWORD WINAPI mnt520_wloPortB (DWORD hDev,WORD wLine,WORD wMid,WORD wPrtNo, BYTE byData); DWORD WINAPI mnt520_wloPortW (DWORD hDev,WORD wLine,WORD wMid,WORD wPrtNo, WORD wData);
	Arguments	DWORD hDev ... Device handle WORD wLine ... Line number (0: Line 1, 1:Line 2) WORD wMid ... Module ID (0: MID = 0, ... , 63: MID = 63) WORD wPrtNo ... Port number (0: Port 0, ... , 3: Port 3) BYTE* byData ... Address where the read port data (1 byte) is stored BYTE byData ... Port data to be written (1 byte) WORD* wData ... Address where the read port data (2 bytes) is stored WORD wData ... Port data to be written (2 bytes)
	Return values	'0' indicates normal processing. For values other than '0', see "7.6.2 Return values of driver functions."
	Example	DWORD ret; // Return value DWORD hDev; // Device handle BYTE byData; // Input port data // Read from Port 0 of Device 0. ret = mnt520_rloPortB(hDev,0,0,0,&byData);
	Remarks	Although the module ID can be specified in the range of 0 to 63, one Line actually allows connecting a maximum of 32 modules.

(15) mnt520_rPclPort ()		Reads 1-byte from a motion device I/O port.
mnt520_wPclPort ()		Writes 1-byte to a motion device output port.
Role		Reads data from the generic I/O port of a specified motion device. Writes data to the output port of a specified motion device.
Win VC++	Format	DWORD WINAPI mnt520_rPclPort (DWORD hDev,WORD wLine,WORD wMid,BYTE* byData); DWORD WINAPI mnt520_wPclPort (DWORD hDev,WORD wLine,WORD wMid,BYTE* byData);
	Arguments	DWORD hDev ... Device handle WORD wLine ... Line number (0: Line 1, 1:Line 2) WORD wMid ... Module ID (0: MID = 0, ... , 63: MID = 63) BYTE* byData. ... Address where data (1 byte) read from the I/O port is stored BYTE byData ... Data (1 byte) to be written to the output port.
	Return values	'0' indicates normal processing. For values other than '0', see "7.6.2 Return values of driver functions."
	Example	DWORD ret; // Return value DWORD hDev; // Device handle BYTE byDat // Input port data // Write to the output port of Device 0. (generic output) ret = mnt520_wPclPort(hDev,0,0,0x01);
	Remarks	Although the module ID can be specified in the range of 0 to 63, one Line actually allows connecting a maximum of 32 modules.

(16) mnt520_rPclMsts ()		Reads a Motion Main Status.
Role		Reads the Motion Main Status of a specified motion device.
Win VC++	Format	DWORD WINAPI mnt520_rPclMsts (DWORD hDev,WORD wLine,WORD wMid,WORD* wSts);
	Arguments	DWORD hDev ... Device handle WORD wLine ... Line number (0: Line 1, 1:Line 2) WORD wMid ... Module ID (0: MID = 0, ... , 63: MID = 63) WORD* wSts ... Address where a Motion Main Status is stored.
	Return values	'0' indicates normal processing. For values other than '0', see "7.6.2 Return values of driver functions."
	Example	DWORD ret; // Return value DWORD hDev; // Device handle WORD wSts; // Motion Main Status // Read the Motion Main Status of Device 0. ret = mnt520_rPclMsts(hDev,0,0,&wSts);
	Remarks	Although the module ID can be specified in the range of 0 to 63, one Line actually allows connecting a maximum of 32 modules.

(17) mnt520_wPclCmd ()		Writes a motion device control command.
Role		Writes a control command to a specified motion device.
Win VC++	Format	DWORD WINAPI mnt520_wPclCmd (DWORD hDev,WORD wLine,WORD wMid,WORD wCmd);
	Arguments	DWORD hDev ... Device handle WORD wLine ... Line number (0: Line 1, 1:Line 2) WORD wMid ... Module ID (0: MID = 0, ... , 63: MID = 63) WORD wCmd ... Control command code
	Return values	'0' indicates normal processing. For values other than '0', see "7.6.2 Return values of driver functions."
	Example	DWORD ret; // Return value DWORD hDev; // Device handle // Write the control command to Device 0 on Line 1. ret = mnt520_wPclCmd(hDev,0,0,0x0049); // mmediate stop
	Remarks	Although the module ID can be specified in the range of 0 to 63, one Line actually allows connecting a maximum of 32 modules. Note: To use this function, set RENVO (9) to "0." (CEND, BRKF, EDTE, ERAE, and CAER are automatically cleared when read.)

(18) mnt520_rPclReg ()		Reads from a motion device register.
mnt520_wPclReg ()		Writes to a motion device register.
Role		Reads data from a register of a specified motion device. Writes data to a register of a specified motion device.
Win VC++	Format	DWORD WINAPI mnt520_rPclReg (DWORD hDev,WORD wLine,WORD wMid,WORD wCmd, DWORD* dwReg); DWORD WINAPI mnt520_wPclCmd (DWORD hDev,WORD wLine,WORD wMid,WORD wCmd, DWORD dwReg);
	Arguments	DWORD hDev ... Device handle WORD wLine ... Line number (0: Line 1, 1:Line 2) WORD wMid ... Module ID (0: MID = 0, ... , 63: MID = 63) WORD wCmd ... Register control command code DWORD* dwReg ... Address where data read from a register is stored DWORD dwReg ... Register data to be written to a register
	Return values	'0' indicates normal processing. For values other than '0', see "7.6.2 Return values of driver functions."
	Example	DWORD ret; // Return value DWORD hDev; // Device handle // Write to the RMV register of Device 0 on Line 1. ret = mnt520_wPclReg(hDev,0,0,0x90,1000);
	Remarks	Although the module ID can be specified in the range of 0 to 63, one Line actually allows connecting a maximum of 32 modules. Note: To use this function, set RENVO (9) to "0." (CEND, BRKF, EDTE, ERAE, and CAER are automatically cleared when read.)

(19) mnt520_rOptPortB ()		Reads 1-byte from an option port
mnt520_rOptPortW ()		Reads 2-byte from an option port
mnt520_wOptPortB ()		Writes 1-byte to an option port
mnt520_wOptPortW ()		Writes 2-byte to an option port
Role		Reads data from an option port of a specified master board. Writes data to an option port of a specified master board.
Win VC++	Format	DWORD WINAPI mnt520_rOptPortB (DWORD hDev,WORD wAdrs,BYTE* byData); DWORD WINAPI mnt520_rOptPortW (DWORD hDev,WORD wAdrs,WORD* wData); DWORD WINAPI mnt520_wOptPortB (DWORD hDev,WORD wAdrs,BYTE byData); DWORD WINAPI mnt520_wOptPortW (DWORD hDev,WORD wAdrs,WORD wData);
	Arguments	DWORD hDev ... Device handle WORD wAdrs ... Board address of the option port BYTE* byData ... Data to be read (1 byte) WORD* wData ... Data to be read (2 bytes) BYTE byData ... Data to be written (1 byte) WORD wData ... Data to be written (2 bytes)
	Return values	'0' indicates normal processing. For values other than '0', see "7.6.2 Return values of driver functions."
	Example	DWORD ret; // Return value DWORD hDev; // Device handle // Write to the option port. ret = mnt520_wOptPortB(hDev,0x0010,1); // OUT1
	Remarks	

(20) mnt520_rCenPortW ()		Reads 2-byte from a specified port in the center device
mnt520_wCenPortW ()		Writes 2-byte to a specified port in the center device
Role		Reads data from a specified port in the center device Writes data to a specified port in the center device
Win VC++	Format	DWORD WINAPI mnt520_rCenPortW (DWORD hDev,WORD wAdrs,WORD* wData); DWORD WINAPI mnt520_wCenPortW (DWORD hDev,WORD wAdrs,WORD wData);
	Arguments	DWORD hDev ... Device handle WORD wAdrs ... Board address on the center device WORD* wData ... Data to be read (2 bytes) WORD wData ... Data to be written (2 bytes)
	Return values	'0' indicates normal processing. For values other than '0', see "7.6.2 Return values of driver functions."
	Example	DWORD ret; // Return value DWORD hDev; // Device handle // Write data to the sending FIFO of the Line 1 center device. ret = mnt520_wCenPortW(hDev,0x0006,0x0090);
	Remarks	

7.7 Library functions

Library functions—made up of driver functions—control basic operations such as the initialization, origin-returning, and positioning operations of a motion module HMG-Px.

7.7.1 List of library functions

No		Function name	Role	Remarks
1	Device related	hpx_GetDevInfo	Gets device information and the number of master board devices.	
2		hpx_DevOpen	Opens a master board and initializes motion modules.	
3		hpx_DevClose	Closes a master board.	
4	Initial settings	hpx_SetOrgMode	Sets the origin-returning mode.	
5		hpx_SetEls	Sets ELS input.	
6		hpx_SetOls	Sets OLS input.	
7		hpx_SetDls	Sets DLS input.	
8		hpx_SetLtc	Sets Ltch input.	
9		hpx_SetClr	Sets CLR input.	
10		hpx_SetCmp	Sets the comparator condition.	
11		hpx_SetEz	Sets the encoder Z-phase	
12		hpx_SetInpos	Sets INPOS input.	
13		hpx_SetSvAlm	Sets SVALM input.	
14		hpx_SetSvCtrCl	Sets SVCTRCL output.	
15		hpx_SetSvRdy	Sets SVRDY input.	
16		hpx_SetCmdPulse	Sets the output form of command pulses.	
17		hpx_SetAccProfile	Sets S-curve or linear acceleration/deceleration.	
18	hpx_SetAutoDec	Sets auto or manual calculation for a deceleration starting-point.		
19	hpx_SetSls	Sets soft-limit.		
20	hpx_SetCtr3	Sets the input source for Counter 3.		
21	hpx_SetFhAdj	Turns ON/OFF the FH correction function.		
22	Status reading	hpx_ReadMainSts	Reads the main status.	
23		hpx_ReadErrorSts	Reads the error status.	
24		hpx_ReadEventSts	Reads the event status.	
25		hpx_ReadExSts	Reads the expansion status.	
26		hpx_ReadOutp	Reads the output port status.	
27		hpx_ReadSpd	Reads the command speed.	
28		hpx_ReadCtr	Reads a counter.	
29	Operation settings	hpx_SetFLSpd	Base speed	
30		hpx_SetAuxSpd	Sets the auxiliary speed	
31		hpx_SetAccRate	Sets the acceleration rate.	
32		hpx_SetDecRate	Sets the deceleration rate.	
33		hpx_SetMult	Sets the multiplier value.	
34		hpx_SetEventMask	Sets the event-mask.	
35		hpx_SetDecPoint	Sets a deceleration starting-point.	
36		hpx_WritOpeMode	Sets the operation mode.	
37		hpx_WritSta	Sets the operation on STA input.	
38		hpx_WritStp	Sets the operation on STA input, and STP auto-output.	
39		hpx_WritFHSpd	Sets the operation speed.	
40		hpx_WritPos	Sets the travel distance.	
41	hpx_WritCtr	Presets the counter.		
42	Operation control command	hpx_DecStop	Deceleration stop	
43		hpx_QuickStop	Immediate stop	
44		hpx_EmgStop	Emergency stop	
45		hpx_AccStart	Acceleration start	
46		hpx_CnstStartFH	FH constant-speed start	
47		hpx_CnstStartFL	FL constant-speed start	
48		hpx_MvAccStart	Travel distance setting + acceleration start	
49		hpx_MvCnstStartFH	Travel distance setting + FH constant-speed start	
50		hpx_MvCnstStartFL	Travel distance setting + FL constant-speed start	
51		hpx_SetGroup	Sets the group.	
52		hpx_GrpStop	Stops the group.	
53		hpx_GrpAccStart	Group acceleration start	
54		hpx_GrpCnstStartFH	Group FH constant-speed start	
55		hpx_GrpCnstStartFL	Group FL constant-speed start	
56		hpx_SvOn	Servo ON	
57		hpx_SvOff	Servo OFF	
58		hpx_SvResetOn	Servo reset ON	
59		hpx_SvResetOff	Servo reset OFF	
60		hpx_SvTlOn	Servo torque limit ON	
61	hpx_SvTlOff	Servo torque limit OFF		
62	hpx_SvGainOn	Servo gain switching ON		
63	hpx_SvGainOff	Servo gain switching OFF		
64	hpx_PMOOn	Stepping motor excitation ON		
65	hpx_PMOOff	Stepping motor excitation OFF		
66	Acceleration/deceleration rate calculation	hpx_CalAccRate	Calculates the acceleration/deceleration rate.	
67	Deceleration starting-point calculation	hpx_CalDecPoint	Calculates the deceleration starting-point.	

Table 7. 7-1 List of library functions

7.7.2 Return values of library functions

If a return value of a function is abnormal (value other than "0") in using library functions, handle the abnormal event appropriately as shown below. In case there are two or more causes of an error, the ORed value may be returned. Please note that errors originating from devices are not included in the error information below. Identify the cause of each error clearly and take proper measures accordingly.

No	Return value		Abnormal event and check item
	Name	Hex	
1	NO_ERROR	00000000	Normal No abnormal event occurred
2	NOT_FOUND	00000001	Absence of the device driver ⊙The device driver has not been installed. ⊙The device driver is not located in the correct directly.
3	ALREADY_OPENED	00000002	Attempted to open a device that is already open. ⊙The Open command has been sent to an already opened device. ◇Use opened devices until they are closed. (Multiple opening forbidden) ⊙When using two or more boards, check to see that the device information for the opened devices has been updated.
4	INSUFFICIENT_MEMORY	00000004	Insufficient memory space for storing device information. ⊙Insufficient memory space for the application. ◇The main memory space in the computer is insufficient. ⊙Insufficient system resources (memory for the OS) ◇Too many applications have been started. ◇Too many windows are opened at the same time.
5	INVALID_HANDLE	00000008	An invalid device handle was specified. ⊙The specified handle is not the one obtained when the device was opened. ⊙The specified device has already been closed.
6	NOT_READY	00000010	Device's I/O ports cannot be used. ⊙There is no I/O port inside the device (board). (The device may be faulty. Contact us.)
7	ILLEGAL_DEVICE	00000020	Board's individual information is invalid. ⊙The device information is proper, but it does not match the board function. (The device may be faulty. Contact us.)
8	ILLEGAL_PARAM	00000100	The value of the function argument is invalid. ◇Check the values of other arguments.
9	CYC_COM_ERROR	00010000	Cyclic communication error ⊙The power to a slave is off. ⊙A communication cable is disconnected. ⊙A communication cable is under the influence of noise.
10	CYC_COM_STOP	00020000	Cyclic communication is stopped. ⊙A data communication was attempted when a cyclic communication was not enabled.
11	DATA_COM_ERROR	00040000	Data communication error
12	COM_OTHER ERROR	00080000	Other communication errors ⊙Check the Center Interrupt Status for details.
13	MODULE_OVERCOUNT	00100000	More than 32 modules are connected to one line. ⊙The number of modules connected to one line must be 32 or less.
14	NO_LOCAL	00200000	There is no module. Check the connections of the modules, condition of the power supply system, the cables, etc.
15	ILL_ACCESS_COM	00400000	An incorrect access attempted during communication
16	SEND_DATA_NONUSE	00800000	A data communication attempted to a device in the unused status.
17	LINE1_ERROR	10000000	The occurrence of an error on Line 1 ⊙An error occurred on Line 1. Check the connections of the modules, condition of the power supply system, the cables, etc. on Line 1.
18	LINE2_ERROR	20000000	The occurrence of an error on Line 2 ⊙An error occurred on Line 2. Check the connections of the modules, condition of the power supply system, the cables, etc. on Line 2.
19	OTHER_ERROR	80000000	Other errors

Table 7. 7-2 Return values of library functions

7.7.3 Details on library functions

This section provides descriptions in VC++ format. For VB, VB.NET, their corresponding data types are shown in the table below.

■ Data types of arguments, hexadecimal notation

Language	VC++ language	VB	VB.NET
Data type	8 bit	BYTE	Byte
	16 bit	WORD	Short
	32 bit	DWORD	Integer
Examples of data	0x0000	&H0	&H0
	0x1000	&H1000	&H1000

■ Device related

(1) `hpx_GetDevInfo ()` Gets the number of boards, device information.

Role	Gets the number of master boards connected to the computer, and the device information.
Format	<code>DWORD hpx_GetDevInfo(DWORD* HpcNum, HMNT520INFO* MntInfo);</code>
Arguments	<ul style="list-style-type: none"> ◆ <code>DWORD* MntNum</code> ... The number of master boards ◆ <code>HMNT520INFO* MntInfo</code> ... The structure storing the device information on the master boards
Remarks	<p><Example></p> <pre>DWORD count; DWORD ret; HMNT520INFO MntInfo[2]; ret = hpx_GetDevInfo(&count, &MntInfo[0]);</pre>

(2) `hpx_DevOpen ()` Opens a device, initializes registers and option ports.

Role	Opens the master board that holds the specified device information, and gets the device handle for distinguishing the board from the other. The obtained device handle will be used to access that particular master board. Also, the function initializes (*1) the motion modules connected to the opened master board.
Format	<code>DWORD hpx_DevOpen(DWORD* hDevID, HMNT520INFO* MntInfo);</code>
Arguments	<ul style="list-style-type: none"> ◆ <code>DWORD* hDevID</code> ... Device handle ◆ <code>HMNT520INFO* MntInfo</code> ... Address of the area that stores the information on the device to be opened
Remarks	<p><Example></p> <p>This example is based on the following assumption: Two master boards are installed in the computer. HMNT520INFO-type array MntInfo[2] is set as a device-information storage area which has already contained the device information on all the boards obtained by the <code>hpx_GetDevInfo</code> function.</p> <pre>DWORD ret; //Return value DWORD hDevID[2]; //Area for storing the device handle ret = hpx_DevOpen(hDevID[0], &MntInfo[0]); //1st device information ret = hpx_DevOpen(hDevID[1], &MntInfo[1]); //2nd device information</pre>

[Initial values of motion module registers (*1)]

Register	Contents	Register initial value	Remarks
RFL	Base speed	200	200pps
RFH	Operation speed	2000	2000pps
RUR	Acceleration rate	1387	Acceleration time from 200ppc to 2,000pps: About 500msec (in linear acceleration/deceleration)
RMG	Speed multiplier	199	x1
RFA	Auxiliary speed	200	200pps
RENV1	Environment setting 1	0x01434004	Command pulse output form: CW/CCW; SVCTRCL automatic output disabled; SVCTRCL output pulse width:13ms; DLS, OLS, SVALM: NC; Immediate stop upon ELS, SVALM input; DLS latch disabled; SVRDY, INPOS: NO; LATC, CLR: falling edge
RENV2	Environment setting 2	0x000004FF	Servo I/F output setting, encoder input multiplier x4
RENV3	Environment setting 3	0x00700002	Origin-returning mode 2 (OLS + Z) Counters 1 to 3 cleared upon origin-returning completion
RIRQ	Event-mask setting	0x00001	When stopped normally

Initial value "0" for the registers other than those listed above. (See "6. Local Devices" for details on registers.)

*1. The initial values of the registers are directly given in the function.

Figure 7. 7-1 Initial values of the motion module registers

(3) `hpx_DevClose ()` Closes a Device.

Role	Closes the master board specified by a device handle. This device handle will be invalid after the process.
Format	DWORD <code>hpx_DevClose(DWORD <i>hDevID</i>);</code>
Arguments	◆DWORD* <i>hDevID</i> ... Device handle
Remarks	The device closing process disables only the device handle, not turning off the pulse output and servo. Therefore, perform the end processing for the motion modules before closing a Device.

■Initial settings

(4) `hpx_SetOrgMode ()` Sets the origin-returning mode.

Role	Sets the origin-returning mode for the motion module whose module ID, Line number, and master board are specified by arguments.
Format	DWORD <code>hpx_SetOrgMode(DWORD <i>hDevID</i>, WORD <i>wLine</i>, WORD <i>wMid</i>, WORD <i>wMode</i>);</code>
Arguments	◆DWORD* <i>hDevID</i> ... Device handle ◆WORD <i>wLine</i> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <i>wMid</i> ... M_ID setting [0: Module ID = 0, ..., 63: Module ID = 63] ◆WORD <i>wMode</i> ... Origin-returning mode [0: ORGmode0, 1: ORGmode1, ..., 12: ORGmode12] (See "Table 8. 2-5 Origin-returning method" and "Table 8. 2-6 Origin-returning method (CTR2 reference method).")
Remarks	

(5) `hpx_SetEls ()` Sets an ELS.

Role	Sets the input polarity and the stop method of the ELS for the motion module whose module ID, Line number, and master board are specified by arguments.
Format	DWORD <code>hpx_SetEls(DWORD <i>hDevID</i>, WORD <i>wLine</i>, WORD <i>wMid</i>, WORD <i>wPol</i>, WORD <i>wStop</i>);</code>
Arguments	◆DWORD <i>hDevID</i> ... Device handle ◆WORD <i>wLine</i> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <i>wMid</i> ... M_ID setting [0: Module ID = 0, ..., 63: Module ID = 63] ◆WORD <i>wPol</i> ... Polarity [0: NC, 1: NO] ◆WORD <i>wStop</i> ... Stop method [0: Immediate stop, 1: Deceleration stop]
Remarks	

(6) `hpx_SetOls ()` Sets an OLS.

Role	Sets the input polarity of the OLS for the motion module whose module ID, Line number, and master board are specified by arguments.
Format	<code>DWORD hpx_SetOls(DWORD hDevID, WORD wLine, WORD wMid, WORD wPol);</code>
Arguments	<ul style="list-style-type: none"> ◆<code>DWORD hDevID</code> ... Device handle ◆<code>WORD wLine</code> ... Line number setting [0: Line number 1, 1: Line number 2] ◆<code>WORD wMid</code> ... M_ID setting [0: Module ID = 0, ..., 63: Module ID = 63] ◆<code>WORD wPol</code> ... Polarity [0: NC, 1: NO]
Remarks	

(7) `hpx_SetDls ()` Sets a DLS.

Role	Sets the input polarity of the DLS for the motion module whose module ID, Line number, and master board are specified by arguments.
Format	<code>DWORD hpx_SetDls(DWORD hDevID, WORD wLine, WORD wMid, WORD wEnbl, WORD wPol, WORD wStop, WORD wLtc);</code>
Arguments	<ul style="list-style-type: none"> ◆<code>DWORD hDevID</code> ... Device handle ◆<code>WORD wLine</code> ... Line number setting [0: Line number 1, 1: Line number 2] ◆<code>WORD wMid</code> ... M_ID setting [0: Module ID = 0, ..., 63: Module ID = 63] ◆<code>WORD wEnbl</code> ... DLS enabled/disabled [0: Disabled, 1: Enabled] ◆<code>WORD wPol</code> ... Polarity [0: NC, 1: NO] ◆<code>WORD wStop</code> ... Operation upon DLS input [0: Deceleration only, 1: Deceleration stop] ◆<code>WORD wLtc</code> ... Latch upon DLS input [0: Latch disabled, 1: Latch enabled]
Remarks	When <code>wEnbl</code> is 0 (DLS disabled), <code>wStop</code> and <code>wLtc</code> are invalid.

(8) `hpx_SetLtc ()` Sets LATIC input.

Role	Sets the LATIC input polarity for the motion module whose module ID, Line number, and master board are specified by arguments.
Format	<code>DWORD hpx_SetLtc(DWORD hDevID, WORD wLine, WORD wMid, WORD wPol);</code>
Arguments	<ul style="list-style-type: none"> ◆<code>DWORD hDevID</code> ... Device handle ◆<code>WORD wLine</code> ... Line number setting [0: Line number 1, 1: Line number 2] ◆<code>WORD wMid</code> ... M_ID setting [0: Module ID = 0, ..., 63: Module ID = 63] ◆<code>WORD wPol</code> ... Polarity [0: Falling edge, 1: Rising edge]
Remarks	

(9) `hpx_SetClr ()` Sets CTR input.

Role	Sets the CLR input setting and selects the counter to be cleared for the motion module whose module ID, Line number, and master board are specified by arguments.
Format	<code>DWORD hpx_SetClr(DWORD hDevID, WORD wLine, WORD wMid, WORD wPol, WORD wCtr);</code>
Arguments	<ul style="list-style-type: none"> ◆<code>DWORD hDevID</code> ... Device handle ◆<code>WORD wLine</code> ... Line number setting [0: Line number 1, 1: Line number 2] ◆<code>WORD wMid</code> ... M_ID setting [0: Module ID = 0, ..., 63: Module ID = 63] ◆<code>WORD wPol</code> ... Input setting [0: Falling edge, 1: Rising edge, 2: L level, 3: H level] ◆<code>WORD wCtr</code> ... Counter to be cleared [0x0001:CTR1, 0x0002:CTR2, 0x0004:CTR3] (Selected by the ORed value)
Remarks	

(10) `hpx_SetCmp ()` Sets the comparator condition.

Role	Sets the comparator condition for the motion module whose module ID, Line number, and master board are specified by arguments.
Format	DWORD <code>hpx_SetCmp</code> (DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i> , WORD <i>wCtr</i> , WORD <i>wCon</i> , WORD <i>wIndex</i> , long <i>ICmp</i>);
Arguments	<ul style="list-style-type: none"> ◆DWORD <i>hDevID</i> ... Device handle ◆WORD <i>wLine</i> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <i>wMid</i> ... M_ID setting [0: Module ID = 0, ..., 63: Module ID = 63] ◆WORD <i>wCtr</i> ... Counter compared by the comparator [0:CTR1, 1:CTR2, 2:CTR3] ◆WORD <i>wCon</i> ... Comparator condition [0: Always comparison-condition mismatch 1: Cmp = Comparison counter (Any count direction) 2: Cmp = Comparison counter (While count increased) 3: Cmp = Comparison counter (While count decreased) 4: Cmp > Comparison counter 5: Cmp < Comparison counter] ◆WORD <i>wIndex</i> ... Synchronous output enabled/disabled [0: Disabled, 1: Enabled] ◆long <i>ICmp</i> ... Comparison data value for the comparator [-134,217,728 to +134,217,727]
Remarks	When the synchronous output is enabled (<i>wIndex</i> = 1), it is impossible to set “ <i>wCon</i> = 4, 5.”

(11) `hpx_SetEz ()` Sets the encoder Z-phase.

Role	Sets the input processing of the encoder Z-phase for the motion module whose module ID, Line number, and master board are specified by arguments.
Format	DWORD <code>hpx_SetEz</code> (DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i> , WORD <i>wCount</i> , WORD <i>wPol</i>);
Arguments	<ul style="list-style-type: none"> ◆DWORD <i>hDevID</i> ... Device handle ◆WORD <i>wLine</i> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <i>wMid</i> ... M_ID setting [0: Module ID = 0, ..., 63: Module ID = 63] ◆WORD <i>wCount</i> ... Z-phase count in origin-returning [0:1st, ..., 15: 16th] ◆WORD <i>wPol</i> ... Polarity [0: Falling edge, 1: Rising edge]
Remarks	

(12) `hpx_SetInpos ()` Sets INPOS.

Role	Sets the input-signal processing method of INPOS for the motion module whose module ID, Line number, and master board are specified by arguments.
Format	DWORD <code>hpx_SetInpos</code> (DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i> , WORD <i>wEnbl</i> , WORD <i>wPol</i>);
Arguments	<ul style="list-style-type: none"> ◆DWORD <i>hDevID</i> ... Device handle ◆WORD <i>wLine</i> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <i>wMid</i> ... M_ID setting [0: Module ID = 0, ..., 63: Module ID = 63] ◆WORD <i>wEnbl</i> ... INPOS control enabled/disabled [0: Disabled, 1: Enabled] ◆WORD <i>wPol</i> ... Polarity [0: NC, 1: NO]
Remarks	

(13) `hpx_SetSvAlm ()` Sets SVALM.

Role	Sets the input polarity and the stop method of SVALM for the motion module whose module ID, Line number, and master board are specified by arguments.
Format	DWORD <code>hpx_SetOls</code> (DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i> , WORD <i>wPol</i>);
Arguments	<ul style="list-style-type: none"> ◆DWORD <i>hDevID</i> ... Device handle ◆WORD <i>wLine</i> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <i>wMid</i> ... M_ID setting [0: Module ID = 0, ..., 63: Module ID = 63] ◆WORD <i>wPol</i> ... Polarity [0: NC, 1: NO] ◆WORD <i>wStop</i> ... Stop method [0: Immediate stop, 1: Deceleration stop]
Remarks	

(14) `hpx_SetSvCtrCI()` Sets the clear-error-counter output.

Role	ets the clear-error-counter output for the motion module whose module ID, Line number, and master board are specified by arguments.
Format	DWORD <code>hpx_SetSvCtrCI(</code> DWORD <code>hDevID,</code> WORD <code>wLine,</code> WORD <code>wMid,</code> WORD <code>wEnbl</code>);
Arguments	<ul style="list-style-type: none"> ◆DWORD <code>hDevID</code> ... Device handle ◆WORD <code>wLine</code> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <code>wMid</code> ... M_ID setting [0: Module ID = 0, ..., 63: Module ID = 63] ◆WORD <code>wEnbl</code> ... Automatic output setting [0: Disabled, 1: When origin completed, 2: When abnormally stopped, 3: When origin completed/ abnormally stopped]
Remarks	

(15) `hpx_SetSvRdy()` Sets the servo ready input.

Role	Sets the input polarity of the SVRDY for the motion module whose module ID, Line number, and master board are specified by arguments.
Format	DWORD <code>hpx_SetSvRdy(</code> DWORD <code>hDevID,</code> WORD <code>wLine,</code> WORD <code>wMid,</code> WORD <code>wPol</code>);
Arguments	<ul style="list-style-type: none"> ◆DWORD <code>hDevID</code> ... Device handle ◆WORD <code>wLine</code> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <code>wMid</code> ... M_ID setting [0: Module ID = 0, ..., 63: Module ID = 63] ◆WORD <code>wPol</code> ... Polarity [0: NC, 1: NO]
Remarks	

(16) `hpx_SetCmdPulse()` Sets the output form of command pulses

Role	Sets the output form of command pulses for the motion module whose module ID, Line number, and master board are specified by arguments.
Format	DWORD <code>hpx_SetCmdPulse(</code> DWORD <code>hDevID,</code> WORD <code>wLine,</code> WORD <code>wMid,</code> WORD <code>wCmdpls</code>);
Arguments	<ul style="list-style-type: none"> ◆DWORD <code>hDevID</code> ... Device handle ◆WORD <code>wLine</code> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <code>wMid</code> ... M_ID SETTING [0: MODULE ID = 0, ..., 63: MODULE ID = 63] ◆WORD <code>wCmdpls</code> ... Output form of command pulses [0: 2-pulse mode, 1: common pulse mode]
Remarks	

(17) `hpx_SetAccProfile()` Sets S-curve or linear acceleration/deceleration.

Role	Sets S-curve or linear acceleration/deceleration for the motion module whose module ID, Line number, and master board are specified by arguments.
Format	DWORD <code>hpx_SetAccProfile(</code> DWORD <code>hDevID,</code> WORD <code>wLine,</code> WORD <code>wMid,</code> WORD <code>wPro</code>);
Arguments	<ul style="list-style-type: none"> ◆DWORD <code>hDevID</code> ... Device handle ◆WORD <code>wLine</code> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <code>wMid</code> ... M_ID setting [0: Module ID = 0, ..., 63: Module ID = 63] ◆WORD <code>wPro</code> ... Acceleration/deceleration type [0: linear, 1: S-curve]
Remarks	

(18) `hpx_SetAutoDec()` Sets auto or manual calculation for a deceleration starting-point.

Role	Sets auto or manual calculation for a deceleration starting-point for the motion module whose module ID, Line number, and master board are specified by arguments.
Format	DWORD <code>hpx_SetAutoDec(</code> DWORD <code>hDevID,</code> WORD <code>wLine,</code> WORD <code>wMid,</code> WORD <code>wAuto</code>);
Arguments	<ul style="list-style-type: none"> ◆DWORD <code>hDevID</code> ... Device handle ◆WORD <code>wLine</code> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <code>wMid</code> ... M_ID SETTING [0: MODULE ID = 0, ..., 63: MODULE ID = 63] ◆WORD <code>wAuto</code> ... Method to set a deceleration starting-point [0: Auto, 1: Manual]
Remarks	

(19) `hpx_SetSls ()` Sets soft-limit.

Role	Sets the soft-limit capability for the motion module whose module ID, Line number, and master board are specified by arguments.
Format	DWORD <code>hpx_SetSls(</code> DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i> , long <i>IPsl</i> , long <i>ImSl</i> , WORD <i>wEnbl</i> , WORD <i>wStop</i>);
Arguments	<ul style="list-style-type: none"> ◆DWORD <i>hDevID</i> ... Device handle ◆WORD <i>wLine</i> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <i>wMid</i> ... M_ID SETTING [0: MODULE ID = 0, ..., 63: MODULE ID = 63] ◆long <i>IPsl</i> ... + SLS [Pulse count] ◆long <i>ImSl</i> ... - SLS [Pulse count] ◆WORD <i>wEnbl</i> ... Enabled/disabled [0: Disabled, 1: Enabled] ◆WORD <i>wStop</i> ... Stop method [0: Immediate stop, 1: Deceleration stop]
Remarks	<ol style="list-style-type: none"> 1. With soft-limit enabled, be sure to specify "psls" larger than "msls." 2. With soft-limit disabled, set as follows: msls = psls = 0, enable = 0 3. If SLS is ON while a start command is written, starting in the SLS ON direction is impossible, but the reverse direction is possible.

(20) `hpx_SetCtr3 ()` Sets the input source for Counter 3.

Role	Sets the input source for Counter 3 for the motion module whose module ID, Line number, and master board are specified by arguments.
Format	DWORD <code>hpx_SetCtr3(</code> DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i> , WORD <i>wSrc</i>);
Arguments	<ul style="list-style-type: none"> ◆DWORD <i>hDevID</i> ... Device handle ◆WORD <i>wLine</i> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <i>wMid</i> ... M_ID SETTING [0: MODULE ID = 0, ..., 63: MODULE ID = 63] ◆WORD <i>wSrc</i> ... Input source [0: Command pulses, 1: Encoder input, 2: Reserved, 3: Reserved, 4: Error count using encoder inputs and command pulses]
Remarks	

(21) `hpx_SetFHAdj ()` Turns ON/OFF the FH correction function.

Role	Turns ON/OFF the FH correction (triangular drive peak prevention) function for the motion module whose module ID, Line number, and master board are specified by arguments.
Format	DWORD <code>hpx_SetFHAdj(</code> DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i> , WORD <i>wEnbl</i>);
Arguments	<ul style="list-style-type: none"> ◆DWORD <i>hDevID</i> ... Device handle ◆WORD <i>wLine</i> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <i>wMid</i> ... M_ID SETTING [0: MODULE ID = 0, ..., 63: MODULE ID = 63] ◆WORD <i>wEnbl</i> ... FH correction setting [0:OFF,1:ON]
Remarks	

■Status reading

(22) `hpx_ReadMainSts ()` Reads the main status.

(23) `hpx_ReadErrorSts ()` Reads the error status.

(24) `hpx_ReadEventSts ()` Reads the event status.

(25) `hpx_ReadExSts ()` Reads the expansion status.

Role	Reads the main status, error status, event status, or expansion status of the motion module whose module ID, Line number, and master board are specified by arguments. Then, the read status data is stored in the specified area. For status data, see “6.2.7 Motion Device registers.”
Format	DWORD <code>hpx_ReadMainSts</code> (DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i> , WORD* <i>wSts</i>); DWORD <code>hpx_ReadErrorSts</code> (DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i> , DWORD* <i>dwSts</i>); DWORD <code>hpx_ReadEventSts</code> (DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i> , DWORD* <i>dwSts</i>); DWORD <code>hpx_ReadExSts</code> (DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i> , DWORD* <i>dwSts</i>);
Arguments	◆DWORD <i>hDevID</i> ... Device handle ◆WORD <i>wLine</i> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <i>wMid</i> ... M_ID SETTING [0: MODULE ID = 0, ..., 63: MODULE ID = 63] ◆WORD* <i>wSts</i> ... Address of the area that stores the main status. ◆DWORD* <i>dwSts</i> ... Address of the area that stores the error, event, or expansion status.
Remarks	As for <code>hpx_ReadErrorSts()</code> and <code>hpx_ReadEventSts()</code> , the Interrupt Reset command (0008h) is issued after status reading. (Resetting bit 1 of MMSTS)

(26) `hpx_ReadOutp ()` Reads the output port status.

Role	Reads the output port status of the motion module whose module ID, Line number, and master board are specified by arguments. Then, the read status data is stored in the specified area.
Format	DWORD <code>hpx_ReadOutp</code> (DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i> , BYTE* <i>byData</i>);
Arguments	◆DWORD <i>hDevID</i> ... Device handle ◆WORD <i>wLine</i> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <i>wMid</i> ... M_ID SETTING [0: MODULE ID = 0, ..., 63: MODULE ID = 63] ◆BYTE* <i>byData</i> ... Address of the area that stores the read data.
Remarks	

(27) `hpx_ReadSpd ()` Reads the current speed.

Role	Reads the command speed set for the motion module whose module ID, Line number, and master board are specified by arguments. Then, the read data is stored in the specified area. For the conversion into the actual command speed, the read data is to be multiplied by the speed multiplier.
Format	DWORD <code>hpx_ReadSpd</code> (DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i> , DWORD* <i>dwSpd</i>);
Arguments	◆DWORD <i>hDevID</i> ... Device handle ◆WORD <i>wLine</i> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <i>wMid</i> ... M_ID SETTING [0: MODULE ID = 0, ..., 63: MODULE ID = 63] ◆WORD* <i>dwSpd</i> ... Address of the area that stores the read data.
Remarks	

(28) `hpx_ReadCtr ()` Reads a counter.

Role	Reads the selected counter set for the motion module whose module ID, Line number, and master board are specified by arguments. Then, the read data is stored in the specified area.
Format	DWORD <code>hpx_ReadCtr</code> (DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i> , WORD <i>wCtr</i> , long* <i>IValue</i>);
Arguments	◆DWORD <i>hDevID</i> ... Device handle ◆WORD <i>wLine</i> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <i>wMid</i> ... M_ID SETTING [0: MODULE ID = 0, ..., 63: MODULE ID = 63] ◆WORD <i>wCtr</i> ... Counter to be selected [1: Counter 1, 2: Counter 2, 3: Counter 3] ◆long* <i>IValue</i> ... Address of the area that stores the read data.
Remarks	

■ Operation settings

(29) `hpx_SetFLSpd ()` Sets the base speed

(30) `hpx_SetAuxSpd ()` Sets the auxiliary speed

Role	Sets the base or auxiliary speed for the motion module whose module ID, Line number, and master board are specified by arguments. Base speed...The starting speed in acceleration/deceleration operation. (accelerated from this speed; decelerated to this speed then stopped) Auxiliary speed...Used as the origin entry speed during some origin returning operations To get a setting value, divide an actual speed (pps) by the speed multiplier.
Format	DWORD <code>hpx_SetFLSpd (DWORD <i>hDevID</i>, WORD <i>wLine</i>, WORD <i>wMid</i>, DWORD <i>dwSpd</i>);</code> DWORD <code>hpx_SetAuxSpd(DWORD <i>hDevID</i>, WORD <i>wLine</i>, WORD <i>wMid</i>, DWORD <i>dwSpd</i>);</code>
Arguments	◆DWORD <i>hDevID</i> ... Device handle ◆WORD <i>wLine</i> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <i>wMid</i> ... M_ID SETTING [0: MODULE ID = 0, ..., 63: MODULE ID = 63] ◆DWORD <i>dwSpd</i> ... The value of the base speed (FL) or auxiliary speed (FA) register [1 to 100,000, where FL(FA) < FH (operation speed)]
Remarks	

(31) `hpx_SetAccRate ()` Sets the acceleration rate.

(32) `hpx_SetDecRate ()` Sets the deceleration rate.

Role	Sets the acceleration (deceleration) rate for the motion module whose module ID, Line number, and master board are specified by arguments.
Format	DWORD <code>hpx_SetAccRate(DWORD <i>hDevID</i>, WORD <i>wLine</i>, WORD <i>wMid</i>, DWORD <i>dwRur</i>);</code> DWORD <code>hpx_SetDecRate(DWORD <i>hDevID</i>, WORD <i>wLine</i>, WORD <i>wMid</i>, DWORD <i>dwRdr</i>);</code>
Arguments	◆DWORD <i>hDevID</i> ... Device handle ◆WORD <i>wLine</i> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <i>wMid</i> ... M_ID SETTING [0: MODULE ID = 0, ..., 63: MODULE ID = 63] ◆DWORD <i>dwRur</i> ... Acceleration rate [1 to 65535] ◆DWORD <i>dwRdr</i> ... Deceleration rate [0 to 65535] (*1)
Remarks	*1. Relationship between acceleration (deceleration) rate and acceleration/deceleration time RFH, RFL, RUR (RDR) (1) In linier acceleration/deceleration: Acceleration/deceleration time (sec) = (2) In S-curve acceleration/deceleration: Acceleration/deceleration time (sec) = If the deceleration rate is 0: Deceleration rate = Acceleration rate

(33) `hpx_SetMult ()` Sets the multiplier value (MG).

Role	Sets the multiplier value (MG) (*1) for the motion module whose module ID, Line number, and master board are specified by arguments.
Format	DWORD <code>hpx_SetMult(DWORD <i>hDevID</i>, WORD <i>wLine</i>, WORD <i>wMid</i>, DWORD <i>dwRmg</i>);</code>
Arguments	◆DWORD <i>hDevID</i> ... Device handle ◆WORD <i>wLine</i> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <i>wMid</i> ... M_ID SETTING [0: MODULE ID = 0, ..., 63: MODULE ID = 63] ◆DWORD <i>dwRmg</i> ... The value of the speed multiplier setting register [1 to 7ffh] (*1)
Remarks	*1. Relationship between speed and speed multiplier, and between the multiplier setting value (MG) and speed multiplier (RFx is the value of each speed register (RFH, RFL, or RFA)) Speed (pps) = RFx × multiplier Multiplier setting value (MG) = (200/speed multiplier) - 1

[Multiplier setting example]

Setting value	Multiplier	Output speed range (pps)	Setting value	Multiplier	Output speed range (pps)
1999(0x7cf)	0.1	0.1 to 10,000	39(0x027)	5	5 to 500,000
999(0x3e7)	0.2	0.2 to 20,000	19(0x013)	10	10 to 1,000,000
399(0x18f)	0.5	0.5 to 50,000	9(0x009)	20	20 to 2,000,000
199(0x0c7)	1	1 to 100,000	3(0x003)	50	50 to 5,000,000
99(0x063)	2	2 to 200,000	2(0x002)	66.6	100 to 6,666,666

(34) `hpx_SetEventMask ()` Sets the event-mask.

Role	Sets the event-mask for the motion module whose module ID, Line number, and master board are specified by arguments.
Format	DWORD <code>hpx_SetEventMask(DWORD <i>hDevID</i>, WORD <i>wLine</i>, WORD <i>wMid</i>, DWORD <i>dwMsk</i>);</code>
Arguments	<ul style="list-style-type: none"> ◆DWORD <i>hDevID</i>... Device handle ◆WORD <i>wLine</i> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <i>wMid</i> ... M_ID SETTING [0: MODULE ID = 0, ..., 63: MODULE ID = 63] ◆DWORD <i>dwMsk</i> ... Event-mask data Giving ORed data of values listed below allows setting of two or more event reports.
Remarks	

[Event-mask setting values]

Setting	Event in event status report	Setting	Event in event status report
0x00001	When operation stops normally	0x00080	When matching the Comparator-3 condition
0x00002	When acceleration begins	0x00100	When clearing the count value by CLR input
0x00004	When acceleration finishes	0x00200	When latching the count value by LTC input
0x00008	When deceleration begins	0x00400	When latching the count value by OLS input
0x00010	When deceleration finishes	0x00800	When DLS input is turned ON
0x00020	When matching the Comparator-1 condition	0x01000	When STA input is turned ON
0x00040	When matching the Comparator-2 condition		

(35) `hpx_SetDecPoint ()` Sets a deceleration starting-point.

Role	Sets a deceleration starting-point (in manual calculation setting) or a deceleration starting-point offset value (in auto calculation setting) for the motion module whose module ID, Line number, and master board are specified by arguments. With the manual calculation setting for a deceleration starting-point, a setting value used is the remaining travel distance. As for calculation of a proper deceleration starting-point in the manual setting, see "8.3.4 Speed Pattern Setting Registers, (7) RDP: Deceleration Starting-Point Register." With the auto setting, a setting value is an offset to the automatically calculated value. Deceleration is started earlier when a positive value is set, and later when a negative value is set.
Format	DWORD <code>hpx_SetDecPoint(DWORD <i>hDevID</i>, WORD <i>wLine</i>, WORD <i>wMid</i>, long <i>IRdp</i>);</code>
Arguments	<ul style="list-style-type: none"> ◆DWORD <i>hDevID</i> ... Device handle ◆WORD <i>wLine</i> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <i>wMid</i> ... M_ID SETTING [0: MODULE ID = 0, ..., 63: MODULE ID = 63] ◆long <i>IRdp</i> ... Deceleration starting-point pulse count
Remarks	

■ Operation management settings

(36) `hpx_WriteOpMode ()` Sets the operation mode.

Role	Sets the operation mode for the motion module whose module ID, Line number, and master board are specified by arguments.
Format	DWORD <code>hpx_WriteOpMode(DWORD <i>hDevID</i>, WORD <i>wLine</i>, WORD <i>wMid</i>, WORD <i>wMode</i>);</code>
Arguments	<ul style="list-style-type: none"> ◆DWORD <i>hDevID</i> ... Device handle ◆WORD <i>wLine</i> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <i>wMid</i> ... M_ID SETTING [0: MODULE ID = 0, ..., 63: MODULE ID = 63] ◆WORD <i>wMode</i> ... Operation mode (See the table below.)
Remarks	

[Operation mode]

mode(Hex)	Operation mode	Remarks	
0 x 00	Continuous operation in (+) direction by command control		
0 x 08	Continuous operation in (-) direction by command control		
0 x 01	Continuous operation by pulsar input		
0 x 10	Origin-returning operation in (+) direction		
0 x 18	Origin-returning operation in (-) direction		
0 x 12	Getting-out-of-origin operation in (+) direction		
0 x 1a	Getting-out-of-origin operation in (-) direction		
0 x 15	Origin search operation in (+) direction		A take-out movelength is set by the <code>hpx_WritePos()</code> function.
0 x 1d	Origin search operation in (-) direction		
0 x 20	Operation to (+) EL or (+) SL		
0 x 28	Operation to (-) EL or (-) SL		
0 x 22	Getting-out-of (+) EL or (+) SL operation		
0 x 2a	Getting-out-of (-) EL or (-) SL operation		
0 x 24	Operation in (+) direction until EZ count completion		
0 x 2c	Operation in (-) direction until EZ count completion		
0 x 41	Positioning operation		
0 x 42	PCS positioning operation (operation created by library functions)		
0 x 44	Return to zero-point of command position		
0 x 45	Return to zero-point of machine position		
0 x 46	One-pulse operation in (+) direction		
0 x 4e	One-pulse operation in (-) direction		
0 x 47	Timer operation	A travel distance (timer time) is set by the <code>hpx_WritePos()</code> function.	
0 x 51	Continuous operation by pulsar input	A travel distance is set by the <code>hpx_WritePos()</code> function.	
0 x 54	Return to zero-point of command position by pulsar input		
0 x 55	Return to zero-point of machine position by pulsar input		

(37) `hpx_WriteSta ()` Sets the operation on STA input.

Role	Sets the operation on STA input for the motion module whose module ID, Line number, and master board are specified by arguments.
Format	DWORD <code>hpx_WriteSta(DWORD <i>hDevID</i>, WORD <i>wLine</i>, WORD <i>wMid</i>, WORD <i>wEnbl</i>, WORD <i>wTrg</i>);</code>
Arguments	<ul style="list-style-type: none"> ◆DWORD <i>hDevID</i> ... Device handle ◆WORD <i>wLine</i> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <i>wMid</i> ... M_ID SETTING [0: MODULE ID = 0, ..., 63: MODULE ID = 63] ◆WORD <i>wEnbl</i> ... Enabled/disabled setting [0: Disabled, 1: Enabled] ◆WORD <i>wTrg</i> ... Trigger setting [0: Level trigger, 1: Edge trigger]
Remarks	Specify this setting in advance when synchronizing modules.

(38) `hpx_WritStp ()` Sets the operation on STP input, and STP auto-output.

Role	Sets the operation on STP input and STP auto-output on abnormal stop for the motion module whose module ID, Line number, and master board are specified by arguments.
Format	DWORD <code>hpx_WritSta(</code> DWORD <code>hDevID</code> , WORD <code>wLine</code> , WORD <code>wMid</code> , WORD <code>wEnbl</code> , WORD <code>wStop</code> , WORD <code>wOut</code>);
Arguments	<ul style="list-style-type: none"> ◆DWORD <code>hDevID</code> ... Device handle ◆WORD <code>wLine</code> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <code>wMid</code> ... M_ID SETTING [0: MODULE ID = 0, ..., 63: MODULE ID = 63] ◆WORD <code>wEnbl</code> ... Enabled/disabled setting [0: Disabled, 1: Enabled] ◆WORD <code>wStop</code> ... Method to stop on STP input [0: Immediate stop, 1: Deceleration stop] ◆WORD <code>wOut</code> ... STP output on abnormal stop [0: Disabled, 1:Enabled]
Remarks	Use this setting when you want to stop the other modules on the same slave when a module is abnormally terminated.

(39) `hpx_WritFHSpd ()` Sets the operation speed.

Role	Sets the operation speed (given by dividing an actual speed by the multiplier) for the motion module whose module ID, Line number, and master board are specified by arguments.
Format	DWORD <code>hpx_WritFHSpd(</code> DWORD <code>hDevID</code> , WORD <code>wLine</code> , WORD <code>wMid</code> , DWORD <code>dwSpd</code>);
Arguments	<ul style="list-style-type: none"> ◆DWORD <code>hDevID</code> ... Device handle ◆WORD <code>wLine</code> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <code>wMid</code> ... M_ID SETTING [0: MODULE ID = 0, ..., 63: MODULE ID = 63] ◆DWORD <code>dwSpd</code> ... FH (operation speed) register value [1 to 100,000]
Remarks	

(40) `hpx_WritPos ()` Sets the travel distance in position operation.

Role	Sets the travel distance in position operation for the motion module whose module ID, Line number, and master board are specified by arguments.
Format	DWORD <code>hpx_WritPos(</code> DWORD <code>hDevID</code> , WORD <code>wLine</code> , WORD <code>wMid</code> , long <code>IDst</code>);
Arguments	<ul style="list-style-type: none"> ◆DWORD <code>hDevID</code> ... Device handle ◆WORD <code>wLine</code> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <code>wMid</code> ... M_ID SETTING [0: MODULE ID = 0, ..., 63: MODULE ID = 63] ◆long <code>IDst</code> ... Travel distance [-134,217,728 to 134,217,727 (28 bits)]
Remarks	

(41) `hpx_WritCtr ()` Presets the counter

Role	Presets the counter designated for the motion module whose module ID, Line number, and master board are specified by arguments.
Format	DWORD <code>hpx_WritCtr(</code> DWORD <code>hDevID</code> , WORD <code>wLine</code> , WORD <code>wMid</code> , long <code>IPre</code> , WORD <code>wCtr</code>);
Arguments	<ul style="list-style-type: none"> ◆DWORD <code>hDevID</code> ... Device handle ◆WORD <code>wLine</code> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <code>wMid</code> ... M_ID SETTING [0: MODULE ID = 0, ..., 63: MODULE ID = 63] ◆long <code>IPre</code> ... Preset value [-134,217,728 to 134,217,727 (28 bits)] ◆WORD <code>wCtr</code> ... Counter to be selected [1: Counter 1, 2: Counter 2, 3: Counter 3]
Remarks	

■ Operation control command

(42) hpx_DecStop () Deceleration stop

(43) hpx_QuickStop () Immediate stop

(44) hpx_EmgStop () Emergency stop

Role	Performs a deceleration stop, immediate stop, or emergency stop of the motion module whose module ID, Line number, and master board are specified by arguments.
Format	DWORD hpx_DecStop (DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i>); DWORD hpx_QuickStop(DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i>); DWORD hpx_EmgStop (DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i>);
Arguments	◆DWORD <i>hDevID</i> ... Device handle ◆WORD <i>wLine</i> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <i>wMid</i> ... M_ID SETTING [0: MODULE ID = 0, ..., 63: MODULE ID = 63]
Remarks	

(45) hpx_AccStart () Acceleration start

(46) hpx_CnstStartFH () FH constant-speed start

(47) hpx_CnstStartFL () FL constant-speed start

Role	Performs an acceleration, FH constant-speed, or FL constant-speed start of the motion module whose module ID, Line number, and master board are specified by arguments.
Format	DWORD hpx_AccStart (DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i>); DWORD hpx_CnstStartFH (DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i>); DWORD hpx_CnstStartFL (DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i>);
Arguments	◆DWORD <i>hDevID</i> ... Device handle ◆WORD <i>wLine</i> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <i>wMid</i> ... M_ID SETTING [0: MODULE ID = 0, ..., 63: MODULE ID = 63]
Remarks	

(48) hpx_MvAccStart () Travel distance setting + acceleration start

(49) hpx_MvCnstStartFH () Travel distance setting + FH constant-speed start

(50) hpx_MvCnstStartFL () Travel distance setting + FL constant-speed start

Role	Sets the travel distance for the motion module whose module ID, Line number, and master board are specified by arguments, and then performs an acceleration, FH constant-speed, or FL constant-speed start.
Format	DWORD hpx_MvAccStart (DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i> , long <i>IDst</i>); DWORD hpx_MvCnstStartFH (DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i> , long <i>IDst</i>); DWORD hpx_MvCnstStartFL (DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i> , long <i>IDst</i>);
Arguments	◆DWORD <i>hDevID</i> ... Device handle ◆WORD <i>wLine</i> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <i>wMid</i> ... M_ID SETTING [0: MODULE ID = 0, ..., 63: MODULE ID = 63] ◆long <i>IDst</i> ... Signed travel distance
Remarks	In using hpx_AccStart(), hpx_CnstStartFH(), or hpx_CnstStartFL(), two data communications are needed: (1) setting the travel distance (data communication), and then (2) issuing a command (data communication). In using hpx_MvAccStart(), hpx_MvCnstStartFH(), or hpx_MvCnstStartFL(), only one data communication is needed to handle both the travel distance setting and the issuing of a command.

(51) `hpx_SetGroup ()` Sets the group.

Role	Assigns the motion module—whose module ID, Line number, and master board are specified by arguments—to the designated group. The motion modules belonging to the same group can be controlled by deceleration stop, immediate stop, acceleration start, FH constant-speed start, FL constant-speed start with library functions.
Format	<code>DWORD hpx_GrDecStop (DWORD hDevID, WORD wLine, WORD wMid, WORD wGroup);</code> <code>DWORD hpx_GrQuickStop(DWORD hDevID, WORD wLine, WORD wMid, WORD wGroup);</code>
Arguments	◆ <code>DWORD hDevID</code> ... Device handle ◆ <code>WORD wLine</code> ... Line number setting [0: Line number 1, 1: Line number 2] ◆ <code>WORD wMid</code> ... M_ID SETTING [0: MODULE ID = 0, ..., 63: MODULE ID = 63] ◆ <code>WORD wGroup</code> ... Group setting [1: Group 1, ..., 7: Group 7]
Remarks	

(52) `hpx_GrpStop ()` Stops grouped modules.

Role	Performs a deceleration or immediate stop—depending on the setting of operation on STP input—of the motion modules whose group, Line number, and master board are specified by arguments.
Format	<code>DWORD hpx_GrpStop (DWORD hDevID, WORD wLine, WORD wGroup);</code>
Arguments	◆ <code>DWORD hDevID</code> ... Device handle ◆ <code>WORD wLine</code> ... Line number setting [0: Line number 1, 1: Line number 2] ◆ <code>WORD wGroup</code> ... Group setting [0: All Groups, 1: Group 1, ..., 7: Group 7]
Remarks	It is necessary to specify the settings of groups and of operation on STP input in advance. If there is no module in the specified group, no operation will be performed with the return value "0" (not error).

(53) `hpx_GrpAccStart ()` Group acceleration start

(54) `hpx_GrpCnstStartFH ()` Group FH constant-speed start

(55) `hpx_GrpCnstStartFL ()` Group FL constant-speed start

Role	Performs an acceleration, FH constant-speed, or FL constant-speed start of the motion modules whose group, Line number, and master board are specified by arguments.
Format	<code>DWORD hpx_GrpAccStart (DWORD hDevID, WORD wLine, WORD wGroup);</code> <code>DWORD hpx_GrpCnstStartFH (DWORD hDevID, WORD wLine, WORD wGroup);</code> <code>DWORD hpx_GrpCnstStartFL (DWORD hDevID, WORD wLine, WORD wGroup);</code>
Arguments	◆ <code>DWORD hDevID</code> ... Device handle ◆ <code>WORD wLine</code> ... Line number setting [0: Line number 1, 1: Line number 2] ◆ <code>WORD wGroup</code> ... Group setting [0: All Groups, 1: Group 1, ..., 7: Group 7]
Remarks	It is necessary to specify the settings of groups and of operation on STA input in advance. If there is no module in the specified group, no operation will be performed with the return value "0" (not error).

- (56) hpx_SvOn () Servo ON
- (57) hpx_SvOff () Servo OFF
- (58) hpx_SvResetOn () Servo reset ON
- (59) hpx_SvResetOff () Servo reset OFF
- (60) hpx_SvTIOOn () Servo torque limit ON
- (61) hpx_SvTIOOff () Servo torque limit OFF
- (62) hpx_SvGainOn () Servo gain switching ON
- (63) hpx_SvGainOff () Servo gain switching OFF
- (64) hpx_PMON () Stepping motor excitation ON
- (65) hpx_PMOFF () Stepping motor excitation OFF

Role	Turns ON or OFF the servo operation, servo reset, servo torque limit, servo gain switching, or stepping motor excitation output for the motion module whose module ID, Line number, and master board are specified by arguments.
Format	DWORD hpx_SvOn (DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i>); DWORD hpx_SvOff (DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i>); DWORD hpx_SvResetOn (DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i>); DWORD hpx_SvResetOff (DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i>); DWORD hpx_SvTIOOn (DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i>); DWORD hpx_SvTIOOff (DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i>); DWORD hpx_SvGainOn (DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i>); DWORD hpx_SvGainOff (DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i>); DWORD hpx_PMON (DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i>); DWORD hpx_PMOFF (DWORD <i>hDevID</i> , WORD <i>wLine</i> , WORD <i>wMid</i>);
Arguments	◆DWORD <i>hDevID</i> ... Device handle ◆WORD <i>wLine</i> ... Line number setting [0: Line number 1, 1: Line number 2] ◆WORD <i>wMid</i> ... M_ID SETTING [0: MODULE ID = 0, ..., 63: MODULE ID = 63]
Remarks	

■Acceleration/deceleration rate calculation

- (66) hpx_CalAccRate () Calculates the acceleration/deceleration rate.

Role	Calculates the acceleration/deceleration rate based on the acceleration/deceleration time, RFH, RFL, etc.
Format	DWORD hpx_CalAccRate(DWORD* <i>dwRur</i> , DWORD <i>dwtim</i> , DWORD <i>dwRfh</i> , DWORD <i>dwRfl</i> , WORD <i>wPro</i> , DWORD <i>dwRs</i>);
Arguments	◆DWORD* <i>dwRur</i> ... Pointer for the acceleration/deceleration rate (RUR) calculation result (acceleration/deceleration rate [1 to 65,535]) ◆DWORD <i>dwTim</i> ... Acceleration/deceleration time (msec) ◆DWORD <i>dwRfh</i> ... RFG register value [1 to 100,000] ◆DWORD <i>dwRfl</i> ... RFL register value [1 to 100,000] ◆WORD <i>wPro</i> ... Acceleration/deceleration type [0: Linear, 1:S-curve] ◆DWORD <i>dwRs</i> ... RUS (RDS) register value (S-curve speed section in S-curve acceleration/deceleration)
Remarks	

■Deceleration starting-point calculation

(66) `hpx_CalDecPoint ()` Calculates the deceleration starting-point.

Role	Calculates the optimum value for a deceleration starting-point in the manual setting. In positioning operation, setting a value larger than that optimum value results in a longer operation at the base speed. Conversely, setting a smaller value in the register causes the operation to stop before reaching the base speed.
Format	DWORD <code>hpx_CalDecPoint(DWORD* <i>dwRdp</i>, DWORD <i>dwRfh</i>, DWORD <i>dwRfl</i>, DWORD <i>dwRmg</i>, DWORD <i>dwRdr</i>, WORD <i>wPro</i>, DWORD <i>dwRds</i>);</code>
Arguments	<ul style="list-style-type: none"> ◆DWORD* <i>dwRdp</i> ... Pointer for the deceleration starting-point (pulse) calculation result (deceleration starting point [0 to 16,777,215]) ◆DWORD <i>dwRfh</i> ... RFH register value [1 to 100,000] ◆DWORD <i>dwRfl</i> ... RFL register value [1 to 100,000] ◆DWORD <i>dwRmg</i> ... RMG register value [2 to 2,047] ◆DWORD <i>dwRdr</i> ... RDR register value (deceleration rate [1 to 65,535]) ◆WORD <i>wPro</i> ... Acceleration/deceleration type [0: Linear, 1:S-curve] ◆DWORD <i>dwRds</i> ... RDS register value (S-curve speed section in S-curve deceleration [0 to 50,000])
Remarks	

7.8 Sample software

This section describes sample software in the C language. Appropriate variables are used in the examples as needed. It is assumed that all the slaves to be used have been turned on and, of course, the connections to serial lines and the termination settings have been performed properly.

Note. Error handlings are omitted from the examples in the following unless otherwise noted. Please keep in mind that creating actual software will surely require error handlings.
It is recommended that error checking is performed when a status is read.

7.8.1 A simplest process of starting a cyclic communication

A simplest example is initiating a cyclic communication after issuing a system communication command to automatically get the information on the local devices connected to the communication line.

```
mnt520_wCenCmd( hDev, 0, 0x1000 ); // Start a system communication for Line 1.  
// Monitor the center main status on Line 1.  
mnt520_rCenMsts( hDev, 0, &sts );  
while( sts & 1 ) {  
    mnt520_rCenMsts( hDev, 0, &sts );  
}  
mnt520_wCenCmd( hDev, 0, 0x3000 ); // Start a cyclic communication for Line 1.
```


7.8.3 Checking and clearing a cyclic communication error

If the local device of the same module ID causes errors (such as no response) three times in cyclic communication, bit 3 of the center main status (CMSTS) is turned to “1”, and the bit for the corresponding cyclic communication error flag is turned to “1.”

<Example>

```
DWORD hDev;          // Master board handle (Assuming that it has already been obtained.)
WORD  wSts, wFlag;

// The modules on Line 1 are controlled in the following.
mnt520_rCenMsts( hDev, 0, &wSts );          // Read the center main status.
if( wSts & 0x8 ){                          // A cyclic communication error occurs.
    mnt520_rLclCycErr( hDev, 0, 0, &wFlag ); // Read the cyclic communication error flag for local
                                                devices 0 to 15.
    if( wFlag != 0 ){                      // A cyclic communication error occurs at local devices 0
                                                to 15.
        // The wFlag bits show which device caused the error.
        // Depending on that, error handling is performed accordingly.
        if( wFlag & 0x0001 ){              // A cyclic communication error occurred at local device 0.
            // Error handling.
        } else if( wFlag & 0x0002 ){      // A cyclic communication error occurred at local device 1.
            // Error handling.
        } else if( wFlag & 0x0004 ){      // A cyclic communication error occurred at local device 2.
            .
            .
            .
        }
        mnt520_wLclCycErr( hDev, 0, 0, wFlag ); // Reset the cyclic communication error flag for local
                                                devices 0 to 15.
        break;
    }
    mnt520_rLclCycErr( hDev, 0, 1, &wFlag ); // Read the cyclic communication error flag for local
                                                devices 16 to 31.
    .
    .
    .
    mnt520_rLclCycErr( hDev, 0, 2, &wFlag ); // Read the cyclic communication error flag for local
                                                devices 32 to 47.
    .
    .
    .
    mnt520_rLclCycErr( hDev, 0, 3, &wFlag ); // Read the cyclic communication error flag for local
                                                devices 48 to 63.
    .
    .
    .
}
```

7.8.4 Data exchange with port data areas (port information, data device status, etc.)

This section shows how to exchange data with the I/O ports of a DIO device (G9002) and how to get the motion main status (MMSTS) of a motion device. Assume that the following two local devices are used.

Device type	Item	Item info	Output data
DIO device	Module ID	2	-
	Port 0	Input	-
	Port 1	Input	-
	Port 2	Output	55h
	Port 3	Output	aah
Motion device	Module ID	5	-

Note. The port area configuration of the motion device is as follows. (always fixed)

Port number	Mode	Contents
Port 0	Input	Lower 8 bits of the motion main status (MMSTS)
Port 1	Input	Higher 8 bits of the motion main status (MMSTS)
Port 2	Input	Output status of the generic output port (IOPIB)
Port 3	Output	Output from the generic output port (IOPIB)

<Example>

```

DWORD hDev;      // Master board handle (Assuming that it has already been obtained.)
WORD  wSts;      // Motion main status of the motion device
BYTE  byData;    // Input port data

// Control the modules on Line 1 in the following.
mnt520_rIoPortB( hDev, 0, 1, 0, &byData );    // Read port 0 of module ID 2.
mnt520_rIoPortB( hDev, 0, 1, 1, &byData );    // Read port 1 of module ID 2.

mnt520_wIoPortB( hDev, 0, 1, 2, 0x55 );      // Output by port 2 of module ID 2.
mnt520_wIoPortB( hDev, 0, 1, 3, 0xaa );      // Output by port 3 of module ID 2.

mnt520_wPclwCmd( hDev, 0, 4, 0x0053 );      // Acceleration start of module ID 5.
mnt520_rPclMsts( hDev, 0, 4, &wSts );      // Read the motion main status of module ID 5.

```

7.8.5 Data communication (Part1): Setting a value for a motion device register

The "mnt520_wPclReg()" function is normally used to write data to a motion device register. In this example, the following functions are used to show how to perform data communication.

Reading the center main status "mnt520_rCenMsts()",
Writing a center device command "mnt520_wCenCmd()",
Writing to center device's sending FIFO "mnt520_wCenSFifo"

<Example>

```
// Writing data "10000 (2710h)" to the RMV register of the motion device with module ID 0 on Line 1.
DWORD hDev;            // Master board handle (Assuming that it has already been obtained.)
WORD sts;             // Center main status

mnt520_wCenCmd ( hDev, 0, 0x0200 );            // Reset the sending FIFO.
mnt520_wCenSFifo( hDev, 0, 0x4000 );           // Write the register write command to the sending FIFO.
mnt520_wCenSFifo( hDev, 0, 0x2710 );           // Write lower words data (bits 15 to 0) to the sending FIFO.
mnt520_wCenSFifo( hDev, 0, 0x0000 );           // Write higher words data (bits 31 to 16) to the sending FIFO.
mnt520_wCenCmd ( hDev, 0, 0x4000 );           // Write the transmission command to module ID 0.

// Check data communication completion.
while( 1 ){
    mnt520_rCenMsts( hDev, 0, &sts );        // Read the center main status.
    if( sts & 1 ) break;                      // Data communication completed.
}
```

Note. If a cyclic or data communication error occurs while reading the center main status, take appropriate measures.

7.8.6 Data communication (Part 2): Reading a motion device register

The "mnt520_rPclReg()" function is normally used to write data to a motion device register.

In this example, the following functions are used to show how to perform data communication.

Reading the center main status	"mnt520_rCenMsts()",
Writing a center device command	"mnt520_wCenCmd()",
Writing to center device's sending FIFO	"mnt520_wCenSFifo()",
Reading from center device's receiving FIFO	"mnt520_rCenRFifo()"

<Example>

// Writing data "10000 (2710h)" to the RMV register of the motion device with module ID 1 on Line 1.

```
DWORD hDev;      // Master board handle (Assuming that it has already been obtained.)
WORD sts;        // Center main status
WORD cmd;        // Response data (1st word)
union {
    WORD reg[2];
    DWORD dwreg;
} read;          // Response data (2nd and 3rd words)
long ctr1;       // RCTR1 data

mnt520_wCenCmd ( hDev, 0, 0x0300 );      // Reset the receiving FIFO.
mnt520_wCenCmd ( hDev, 0, 0x0200 );      // Reset the sending FIFO.
mnt520_wCenSFifo( hDev, 0, 0x00e3 );     // Write the RCTR1 register read command to the sending FIFO.
mnt520_wCenCmd ( hDev, 0, 0x4001 );     // Write the transmission command to module ID 0.

// Check data communication completion.
while( 1 ) {
    mnt520_rCenMsts( hDev, 0, &sts );    // Read the center main status.
    if( sts & 1 ) break;                 // Data communication completed.
}

// Reading of the receiving FIFO is performed three times since response data by the motion device (G9003)
// consists of three words.
mnt520_rCenRFifo( hDev, 0, &tmp );      // Read from the receiving FIFO. (for reading RCTR1 register read
// command)
mnt520_rCenRFifo( hDev, 0, &read.reg[0] ); // Read from the receiving FIFO. (RCTR1 lower word data)
mnt520_rCenRFifo( hDev, 0, &read.reg[1] ); // Read from the receiving FIFO. (RCTR1 higher word data)
ctr1 = (long)read.dwreg;
```

7.9 Accompanying software

The motionCAT series products come with the following pieces of software, which are prepared to help you to understand the information on the software aspect described in this user's manual.

[Accompanying software for Windows]

- (1) Device drivers ... For Windows Vista, XP, and 2000
- (2) Sample programs
 - For motion modules ... Sample for instructing how to use 1-axis motion modules with driver functions
 - For DIO modules ... Sample for instructing how to use DIO modules with driver functions
- (3) Quick start ... Software for test run

7.9.1 Device driver for Windows

- (1) Device driver for Windows ...hm520wdm.sys
- (2) DLL for the device driver interface for Windows ...himnt520.dll
 - Functions included in the DLL for the device driver interface are referred to as “driver functions.”

7.9.2 Sample programs for Windows

The accompanying software includes samples for motion modules (VC, VB, VB.NET) and for DIO modules (VC, VB, VB.NET).

(1) Sample for DIO modules

This section describes sample programs for DIO modules.

(1) Executing the sample program

To use the sample program, copy it to your hard disk.

The executable file (sd32000.exe) for the sample can be executed by “double-clicking” the mouse button.

[Note on sample program execution]

- Do not use duplicate module IDs on the same Line when using two or more modules (including motion modules).

(2) Operating the sample program

The initialization process is partly fixed in the sample source program.

To execute the program with different initialization settings, modify the source program and rebuild it.

(1) Operation screen 1 (module selection)

After the sample program is successfully started up, a screen shown below appears.

Select the Board ID of the master board to be used.

Select the Line number to be used.

Statuses of the generic inputs of the master board.
White: absence of input,
Green: presence of input

Generic outputs of the master board and their statuses
Output is reversed by clicking operation.
Gray: absence of output,
Green: presence of output

This indicates the PCI device information of the selected master board.

Figure 7. 9-1 DIO module sample screen

Select the Board ID and the Line number of the master board to be used.

Next, click the [Start System Communication] button.

Now, the module ID combo box contains the DIO module IDs on the Line specified to perform system communication.

Module ID combo box
This let you select a module ID to be used.

Figure 7. 9-2 Module ID combo box

(2) Starting cyclic communication

Specify a module ID and click [Start Cyclic Communication]. Then, cyclic communication is started for all the available modules, and you can operate the specified DIO module.

By clicking [Stop Cyclic Communication], cyclic communication for all the modules is stopped, and the display goes back to the operation screen 1.

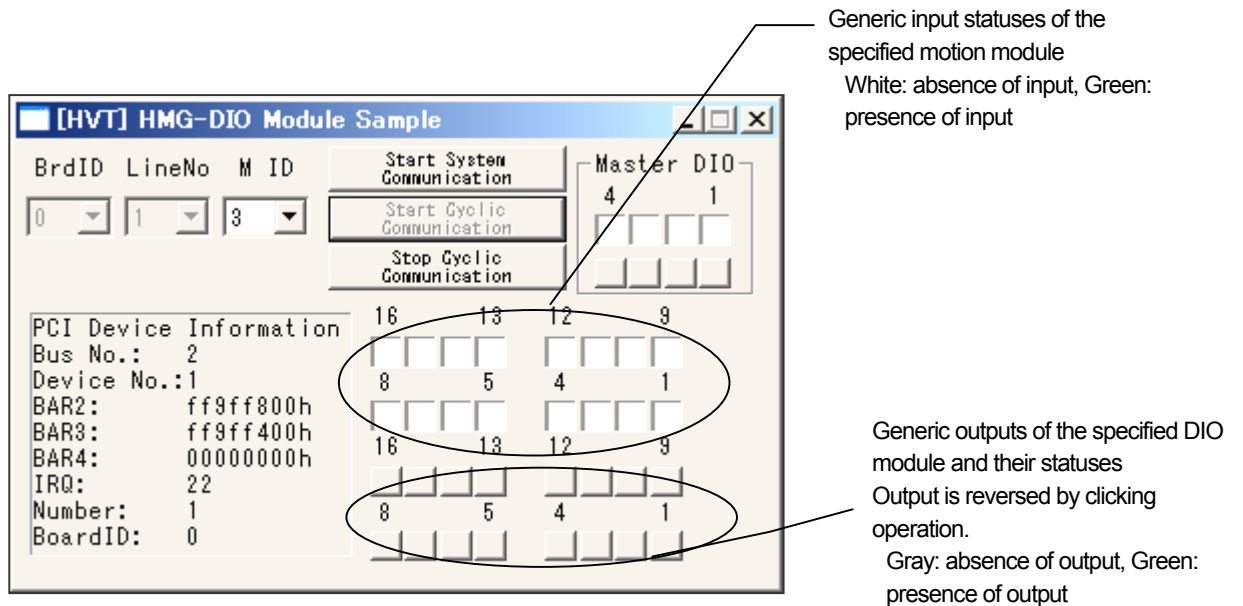


Figure 7. 9-3 DIO module operation screen

(2) Sample for motion modules

This section describes sample programs for motion modules.

(1) Executing the sample program

To use the sample program, copy it to your hard disk.

The executable file (sp10000.exe) for the sample can be executed by “double-clicking” the mouse button.

[Note on sample program execution]

- Do not use duplicate module IDs on the same Line when using two or more modules (including DIO modules).

(2) Operating the sample program

The initialization process for each axis is partly fixed in the sample source program.

To execute the program with different initialization settings, modify the source program and rebuild it.

Register	Contents	Register initial value	Remarks
RFL	Base speed	200	200pps
RFH	Operation speed	2000	2000pps
RUR	Acceleration rate	1387	Acceleration time from 200ppc to 2,000pps: About 500msec (in linear acceleration/deceleration time)
RMG	Speed multiplier	199	x1
RFA	Auxiliary speed	200	200pps
RENV1	Environment setting 1	0x00434004	Command pulse output form: CW/CCW; SVCTRCL automatic output disabled; SVCTRCL output pulse width:13ms; DLS, OLS, SVALM: NC; Immediate stop upon ELS, SVALM input; DLS latch disabled; SVRDY, INPOS: NO; LATC, CLR: falling edge
RENV2	Environment setting 2	0x000004FF	Servo I/F output setting, encoder input multiplier x4
RENV3	Environment setting 3	0x00700002	Origin-returning mode 2 (OLS + Z) Counters 1 to 3 cleared upon origin-returning completion
RIRQ	Event-mask setting	0x00001	When stopped normally

Initial value “0” for the registers other than those listed above. (See “6. Local Devices” for details on registers.)

Table 7. 9-3 Initial values of the sample program G9003 registers

(1) Program selection screen

After the sample program is successfully started up, a screen shown below appears.
Select a program here.

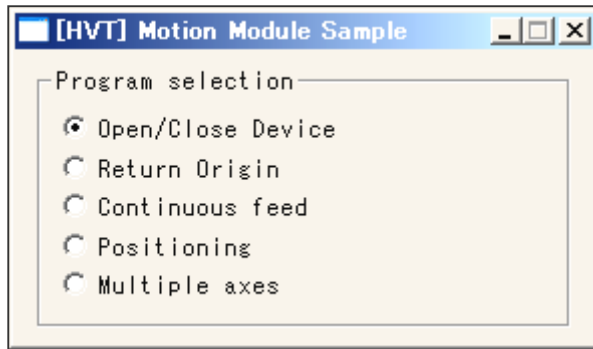


Figure 7. 9-4 Program selection screen

(2) Device opening/closing sample

Click the [Open/Close Device] radio button. Then, the following screen is displayed.
This shows a sample of device opening/closing.

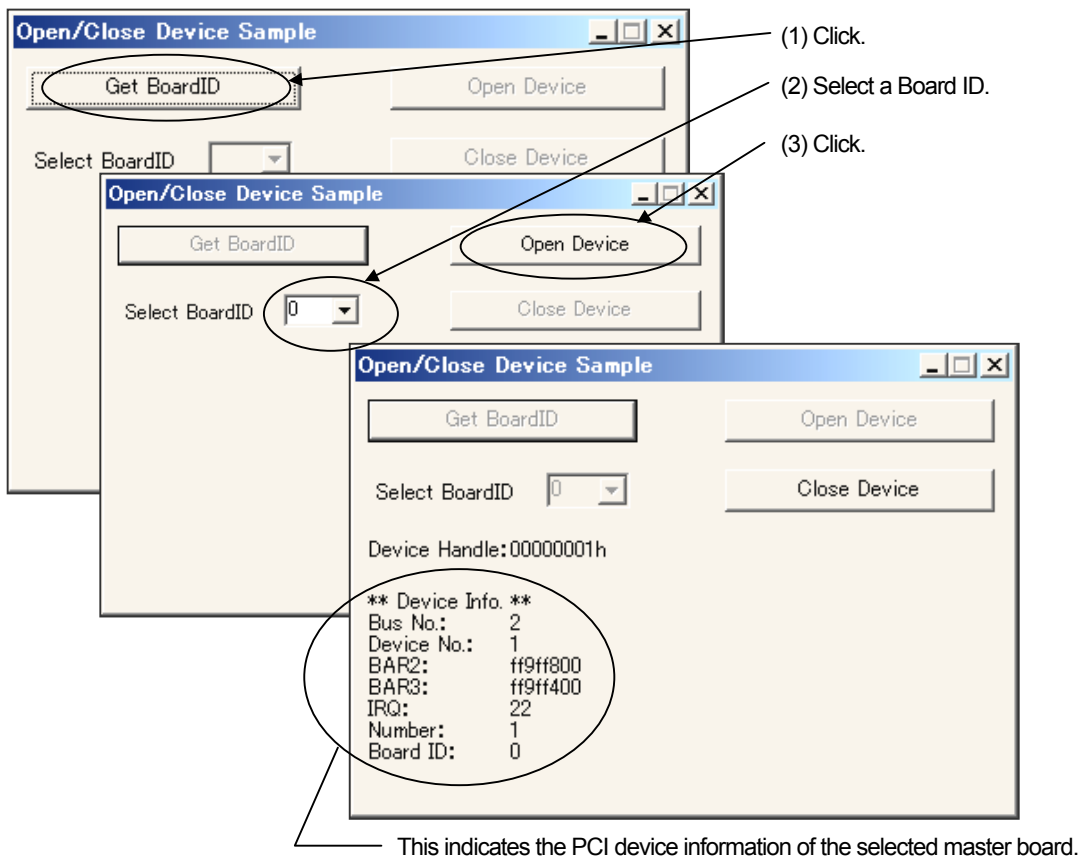


Figure 7. 9-5 Open/Close Device screen

(3) Origin-returning operation sample

Click the [Return Origin] radio button. Then, the following screen is displayed.

This screen allows you to set and perform origin-returning operation.

[Operation screen]

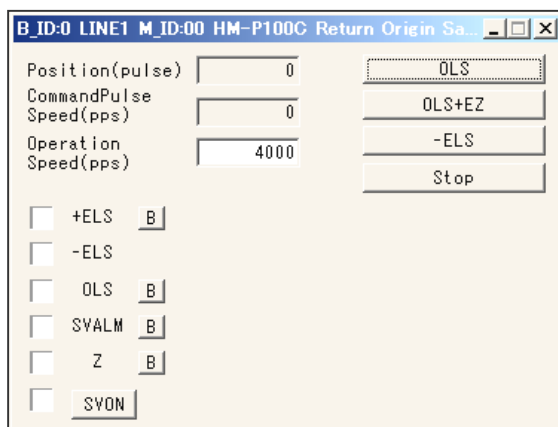


Figure 7. 9-6 Return Origin sample screen

[Executing origin-returning operation]

The available origin-returning methods are as follows:

- OLS origin-returning ... Origin-returning operation 1: Deceleration stop upon OLS detection, followed by getting out of OLS in reverse direction, reentry, and completion.
- OLS + Z-phase origin-returning operation ... Origin-returning operation 2: Deceleration upon OLS detection, then completion by encoder Z-phase detection.
- ELS shared origin-returning operation ... Origin-returning operation 6: Deceleration stop upon OLS detection, operation in reverse direction, and completion by ELS OFF.

For details on origin-returning operation, see “8.2.3 (1) Origin sensor settings and origin-returning methods.”

■ Polarity selection

When a sensor is input, the color of the corresponding indicator is turned to: Red for + ELS, - ELS, SVALM; Green for OLS, Z.

When SVON is output, the indicator is turned to Green.

By clicking the [SVON] button, you can turn on or off the servo.

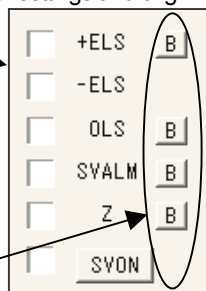


Figure 7. 9-7 Input polarity selection

Input polarity button: Switches between the input polarities.

It also shows the currently selected polarity.

A: Normal open setting

B: Normal close setting

Note 1. With the presence of + ELS, - ELS, or SVALM input, operation cannot be started.

Check the status of each sensor before starting operation.

* It is assumed that appropriate connections have been made for SVON.

Note 2. N.O. (Normally Opened) refers to the setting by which a flow of current through the terminal causes ON (detection).

N.C. (Normally Closed) refers to the setting by which the termination of the current flowing through the terminal causes ON (detection).

■ Operation speed setting

The operation speed can be specified in the range of 1 to 100,000 (pps).

The default value is 4,000 (pps), so adjust the value as appropriate.

Since the default base speed is 200 (pps), adjusting the operation speed to a value below 200 (pps) will cause acceleration toward 200 (pps) where OLS ON is supposed to give deceleration.

To avoid this, modify the sample source program so that the base speed is set to an appropriate value.

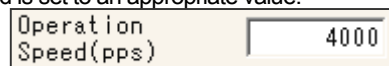


Figure 7. 9-8 Operation speed setting

■ Starting origin-returning operation

After selecting the polarity and setting the operation speed, click the [OLS],[OLS+EZ] , or [-ELS] button to execute the selected origin-returning operation.

To stop the operation in progress, click [Stop] .

The [Position] area shows the command pulse counter.

The [Speed] area shows the speed (pps) of currently output pulses.

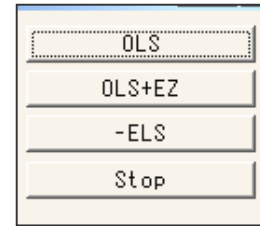


Figure 7. 9-9 Origin-returning operation start buttons

Note 1. Since the detection of OLS is edge detection, OLS is not detected if it is ON at startup of operation. In that case, perform continuous feed operation until OLS is turned OFF before starting origin-returning operation.

Note 2. In ELS shared origin-returning operation, ELS ON causes deceleration stop. Perform operation at a speed which makes the deceleration distance be less than the length between the position of ELS turned ON and the mechanical edge of the axis.

(4) Continuous feed operation sample

This option provides acceleration/deceleration continuous feed operation.

[Operation screen]

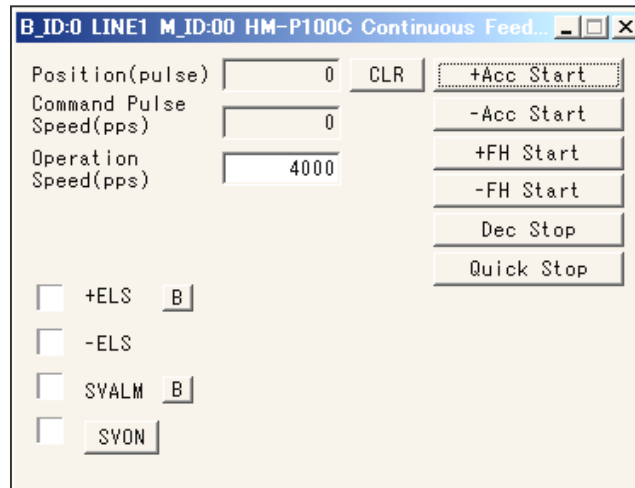


Figure 7. 9-10 Continuous feed operation sample screen

Check the connection status of each sensor before starting operation as is the case for origin-returning operation. Specify the operation speed in the range of 1 to 100,000 (pps). Then, click the [+Acc Start],[-Acc Start], [+FH Start], or [-FH Start] button to start the selected operation.

To deceleration stop the operation, click [Dec Stop]. To quick stop the operation, click [Quick Stop].

To reset the counter, click [CLR].

(5) Positioning operation sample

This option provides acceleration/deceleration positioning operation.

[Operation screen]

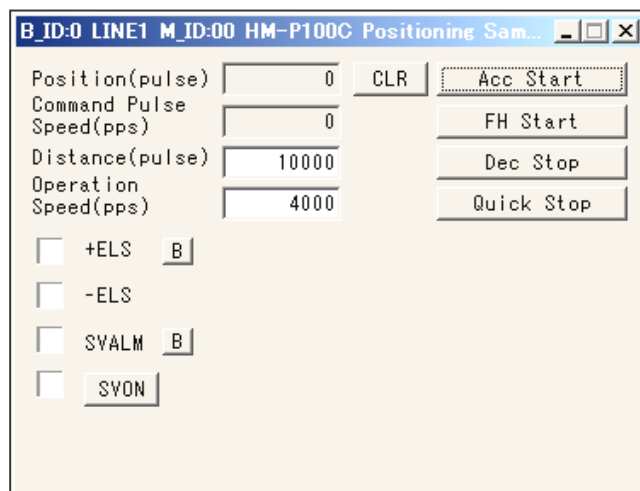


Figure 7. 9-11 Positioning operation sample screen

Check the connection status of each sensor before starting operation as is the case for origin-returning operation. Specify the operation speed in the range of 1 to 100,000 (pps). Set the travel distance (signed). Then, click the [Acc Start] button to start acceleration/deceleration positioning operation.

To deceleration stop the operation, click [Dec Stop]. To quick stop the operation, click [Quick Stop].

To reset the counter, click [CLR].

(6) Multiple axes operation sample

This option provides positioning operation of two specified motion modules on the same Line.

[Operation screen]

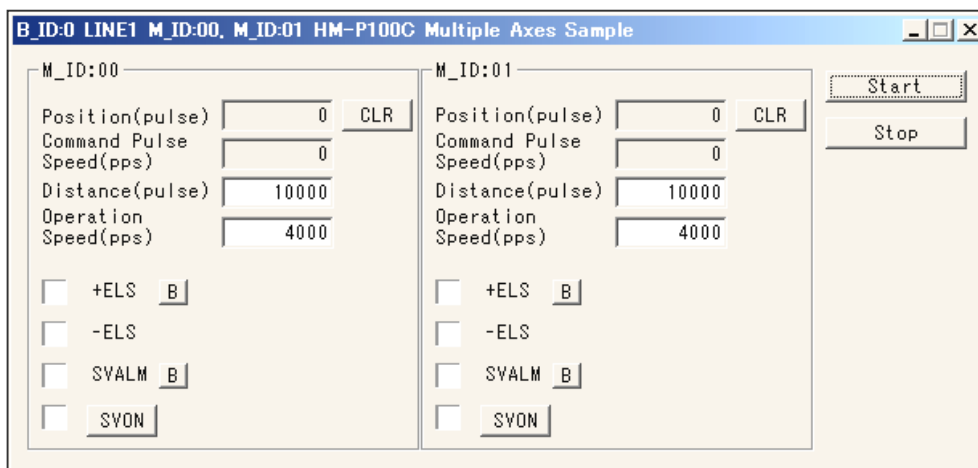


Figure 7. 9-12 Multiple axes operation sample screen

Check the connection status of each sensor before starting operation as is the case for origin-returning operation. Specify the operation speed in the range of 1 to 100,000 (pps). Set the travel distance (signed). Then, click the [Start] button to start acceleration/deceleration positioning operation of the two specified axes.

To stop the operation of the two axes in progress, click [Stop].

Note 1. This sample provides positioning operation for multiple axes independently. Since it is not interpolation operation, the two axes do not stop simultaneously when one axis is halted abnormally due to the servo alarm etc.

7.9.3 “Quick Start” for Windows

The “Quick Start” program is software for trial operation or operation check for the motionCAT system. Execute “%test%\Release\%mnt5200.exe” in the accompanying software disk.

[Note on “Quick Start” operation]

- Do not use duplicate module IDs on the same Line when operating more than one module.
- If there are duplicate board IDs in using more than one master board, board selection with board IDs is impossible.

(1) Selecting modules

Select a master board to be used with the board ID or the device number.

Once the master board is selected, a system communication is started to get the information on all the modules connected to the specified master board. Now, select a module to be used by specifying its module ID.

If a motion module is selected, the motion module operation screen is displayed. If a DIO module is selected, the DIO module operation screen is displayed.

This program allows you to use more than one master board, and control all the modules connected to each master board.

However, operation screens for both motion and DIO modules can be opened up to four at the same time.

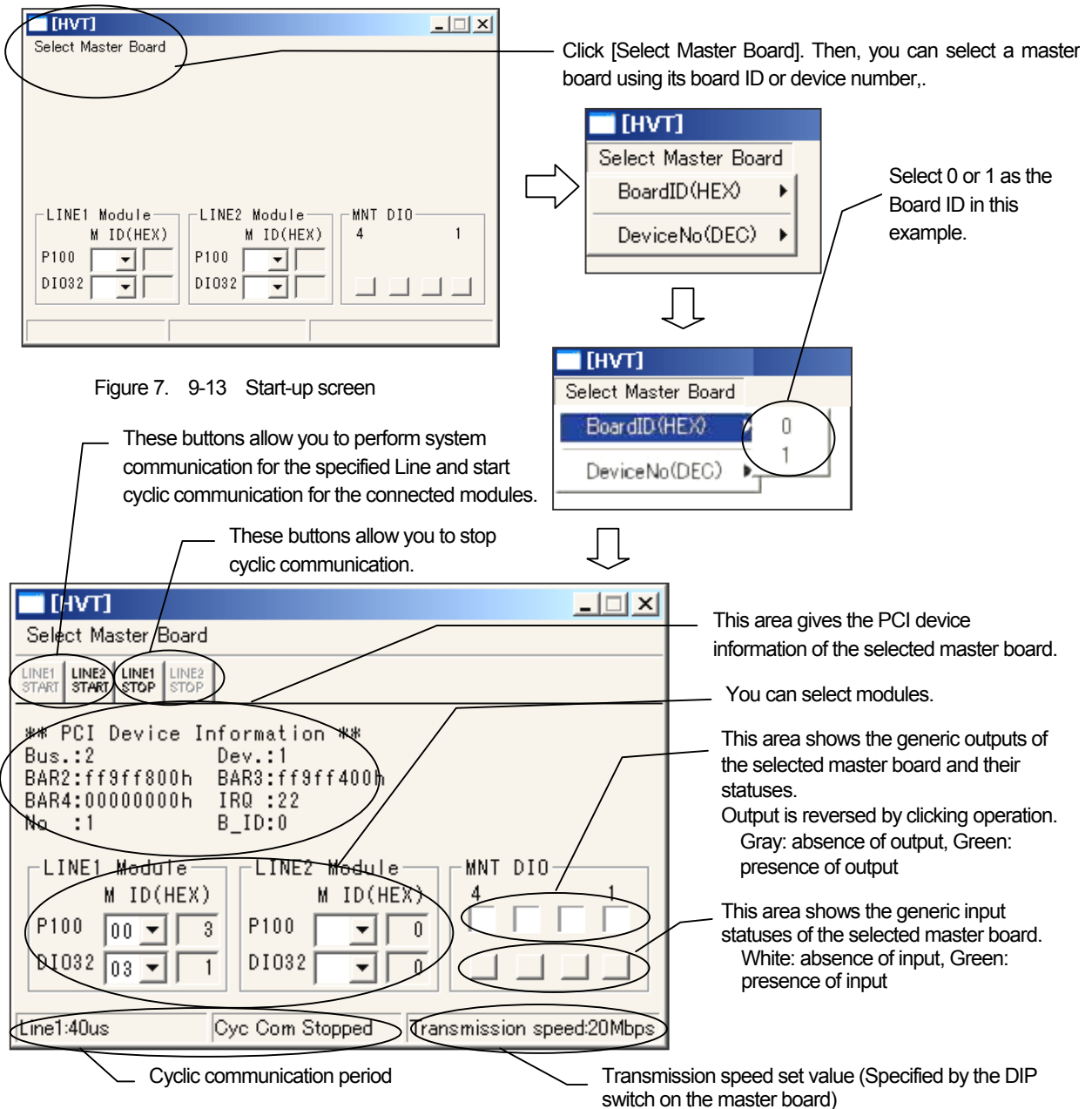


Figure 7. 9-14 Screen after master board selection

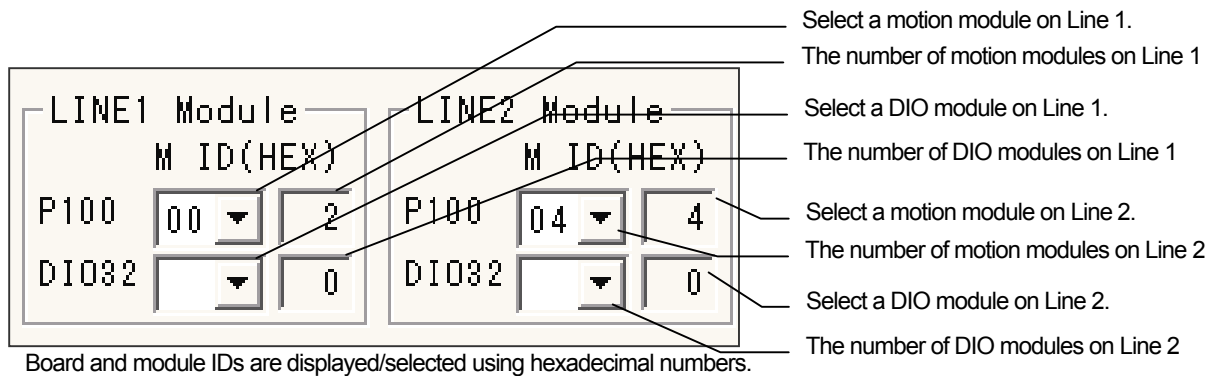


Figure 7. 9-15 Module selection screen

(2) Operating motion modules

■ Functions

The motion module operation screen in "Quick start" provides the following functions.

- (1) 1-axis positioning
- (2) ± Continuous feed
- (3) ± Origin-returning
- (4) Servo I/F output (SVON, SVRST, SVTL, SVGAIN)
- (5) Command counter (CTR 1)/ machine counter (CTR2) display
- (6) Command speed display
- (7) Machine sensor status display (±ELS, OLS, DLS)
- (8) Servo I/F input status display (SVALM, INPOS, SVRDY, Z-phase)
- (9) Servo I/F output status display (SVON, SVRST, SVTL, SVGAIN)
- (10) Servo I/F output (SVON, SVRST, SVTL, SVGAIN)
- (11) Parameter settings

Positioning operation's travel distance, operation speed, base speed, auxiliary speed, speed multiplier, acceleration time, deceleration time, CMP3 comparison data, origin-returning mode

■ Register's initial values in motion modules

The initialization process for each axis is partly fixed in the source program of "Quick Start."

To execute the program with different initialization settings, modify the source program and rebuild it.

Register	Contents	Initial value	Remarks
RMV	Travel distance	10000	10000 pulses
RFL	Base speed	200	200pps
RFH	Operation speed	2000	2000pps
RUR	Acceleration rate	1387	Linear acceleration time from 200ppc to 2,000pps: about 500msec, Acceleration distance: about 550 pulses
RMG	Speed multiplier	199	x1
RFA	Auxiliary speed	200	200pps
RMD	Operation mode	0x00000030	Continuous operation in (+) direction, DLS and INPOS enabled, linear acceleration/deceleration, deceleration starting point auto calculation, FH correction ON
RENV1	Environment setting 1	0x00034004	Input polarity of DLS, OLS, SVALM, INPOS, SVRDY: NC; DLS latch disabled; Immediate stop upon ELS, SVALM input; SVCTRCL auto output upon origin-returning completion disabled; SVCTRCL auto output on abnormal stop disabled; SVCTRCL output pulse width:13 msec; Pulse output delay time after SVCTRCL output: 104 msec
RENV2	Environment setting 2	0x000004FF	Encoder input x4 count
RENV3	Environment setting 3	0x00700001	Origin-returning mode 1 CTR reset upon origin-returning
RENV4	Environment setting 4	0x00040000	Comparator output where RCMP3 = RCTR1
RCMP3	CMP3 comparison data	10000	
RIRQ	Event setting	0x00000001	When stopped normally

Initial value "0" for the registers other than those listed above.

Table 7. 9-4 Initial values of "Quick Start" G9003 registers

■ Motion module operation screen

When a motion module is selected, the operation screen is displayed as shown below.

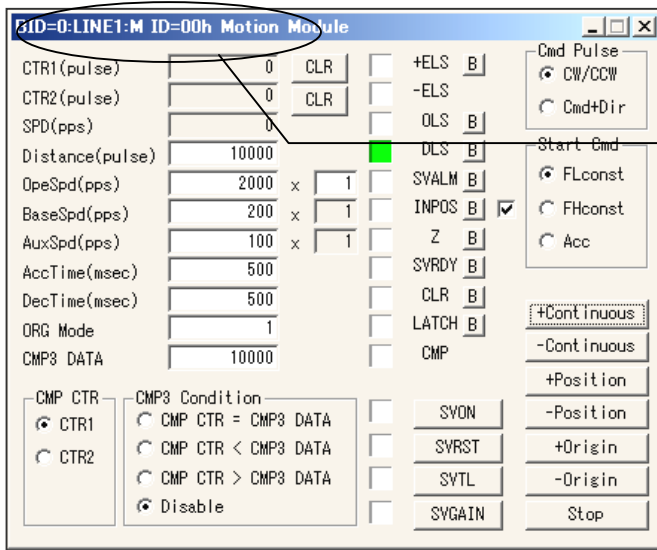


Figure 7. 9-16 Motion module operation screen

This title bar shows which module is focused on.

In this example:

Master board whose board ID is 0,
Connected to Line 1,
Module whose ID is 00h.

While cyclic communication is stopped, the message "Cyclic Communication Stopped" appears.

■ Signal input polarity settings and input status indicators

The indicators for input/output statuses are updated in a cycle of about 110 msec by level trigger. Check to see that each terminal is properly connected.

As for unconnected terminals, set their input polarities to "N.O."

Set the input polarities of the terminals to be used.

To enable INPOS, check the INPOS enabled/disabled checkbox. Also, check the connections of servo I/F outputs.

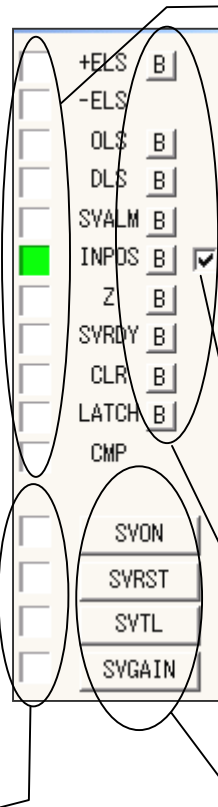


Figure 7. 9-17 Signal input statuses

SSVON : SSVON output ON/OFF
White: absence of output, Green: presence of output

SSVRST : SSVRST output ON/OFF
White: absence of output, Green: presence of output

SSVTL : SSVTL output ON/OFF
White: absence of output, Green: presence of output

SSVGAIN : SSVGAIN output ON/OFF
White: absence of output, Green: presence of output

+ ELS : + ELS input status
White: absence of input, Red: presence of input

- ELS : - ELS input status
White: absence of input, Red: presence of input

OLS : OLS input status
White: absence of input, Green: presence of input

DLS : DLS input status
White: absence of input, Green: presence of input

SVALM : SVALM input status
White: absence of input, Red: presence of input

INPOS : INPOS input status
White: absence of input, Green: presence of input

EZ : EZ input status
White: absence of input, Green: presence of input

SVRDY : SVRDY input status
White: absence of input, Green: presence of input

CLRY : CLRY input status
White: absence of input, Green: presence of input

LATCH : LATCH input status
White: absence of input, Green: presence of input

CMP : CMP condition match status
White: condition mismatch, Green: condition

INPOS enabled/disabled check box
Checked: enabled, Unchecked: disabled

Sensor input statuses and input polarity switching
B N.C. setting (Normally Closed)
A N.O. setting (Normally Opened)
Switch between N.C./N.O. by clicking.

SVON : SVON output ON/OFF

SVRST : SVRST output ON/OFF

SVTL : SVTL output ON/OFF

SVGAIN : SVGAIN output ON/OFF

Parameter settings for operation

For your safety, check operation at a low speed when running the system for the first time.

CTR1 (pulse)	0	CLR	Command position.....	CTR1 value: counting command pulse outputs
CTR2 (pulse)	0	CLR	Machine position.....	CTR2 value: counting encoder F/B by multiplier x4
SPD(pps)	0		Command speed.....	Command pulse speed (unit: pps)
Distance(pulse)	10000		Travel distance.....	Travel distance in positioning operation (unsigned) (unit: pulse)
OpeSpd(pps)	2000	x	Operation speed.....	Value set in RFH Operation speed = RFH × (speed multiplier)
BaseSpd(pps)	200	x	Base speed.....	Value set in RFL Base speed = RFL × (speed multiplier)
AuxSpd(pps)	100	x	Auxiliary speed.....	Used for origin modes 1, 4, 6, 7 Value set in RFA Auxiliary speed = RFA × (speed multiplier)
AccTime(msec)	500		Speed multiplier.....	A speed register (RFH, RFL, RFA) value multiplied by the speed multiplier gives the actual speed.
DecTime(msec)	500		Acceleration time.....	Time spent for acceleration from the base to operation speed in acceleration/deceleration start.
ORG Mode	1		Deceleration time.....	Time spent for deceleration from the operation to base speed in acceleration/deceleration start.
CMP3 DATA	10000		Origin-returning mode.....	Specifies the origin-returning mode.
CMP CTR <input checked="" type="radio"/> CTR1 <input type="radio"/> CTR2			CMP3 comparison data.....	Sets comparator comparison data.
CMP3 Condition <input type="radio"/> CMP CTR = CMP3 DATA <input type="radio"/> CMP CTR < CMP3 DATA <input type="radio"/> CMP CTR > CMP3 DATA <input checked="" type="radio"/> Disable			CMP3 comparison condition...	Sets the comparator comparison condition.
			Comparison counter selection.	Sets CTR1 (command position) or CTR2 (machine position) as the comparison counter for the comparator.

Figure 7. 9-18 Parameter settings for operation

Operation start buttons

Cmd Pulse <input checked="" type="radio"/> CW/CCW <input type="radio"/> Cmd+Dir	Command-pulse output form selection.....	Selects the output form of command pulses. The setting here must be fitting to the pulse input form given by the connected driver.
Start Cmd <input checked="" type="radio"/> FLconst <input type="radio"/> FHconst <input type="radio"/> Acc	Start command selection.....	Selects FL constant-speed, FH constant-speed, high-speed start (with acceleration/deceleration)
+Continuous	+ Continuous feed.....	Movement in (+) direction at the specified speed. Stopped by the [Stop] button or by +ELS, SVALM input.
-Continuous	- Continuous feed.....	Movement in (-) direction at the specified speed. Stopped by the [Stop] button or by -ELS, SVALM input.
+Position	+ Positioning operation.....	Positioning in (+) direction at the specified speed and travel distance. Stopped by the [Stop] button or by +ELS, SVALM input.
-Position	- Positioning operation.....	Positioning in (-) direction at the specified speed and travel distance. Stopped by the [Stop] button or by -ELS, SVALM input.
+Origin	+ direction origin-returning.....	Origin-returning in (+) direction at the specified speed and travel distance. Stopped by origin-returning completion, the [Stop] button, or +ELS, SVALM input. SVCTRCL output and CTR1, CTR2 reset upon origin-returning completion.
-Origin	- direction origin-returning.....	Origin-returning in (-) direction at the specified speed and travel distance. Stopped by origin-returning completion, the [Stop] button, or -ELS, SVALM input. SVCTRCL output and CTR1, CTR2 reset upon origin-returning completion.
Stop	Stop.....	Deceleration stop of acceleration/deceleration operation. Immediate stop for FL constant or FH constant operation.

Figure 7. 9-19 Operation start buttons

(3) Operating DIO modules

(1) Functions

1. Displays the input statuses of the specified DIO module (IN1 to IN16).
2. Displays the output statuses of the specified DIO module (OUT1 to OUT16).
3. Turns ON/OFF the outputs of the specified DIO module.

(2) DIO module operation screen

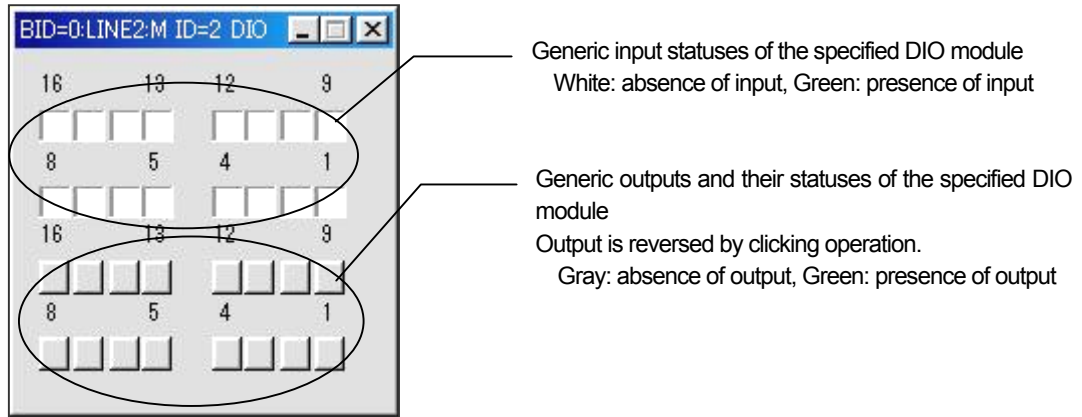


Figure 7. 9-20 DIO module operation screen

8. Motion Module Operation

This chapter provides the following information on motion modules:

- ◇ Programming procedure
- ◇ Initial settings
- ◇ Origin sensor settings and origin-returning methods
- ◇ Speed pattern settings
- ◇ Operation modes
- ◇ Statuses, registers, option ports, commands

8.1 Programming procedure

The basic programming procedure is outlined in the right flowchart. The following information is given according to this procedure.

8.1.1 Master board initial settings

The program begins with the initialization of the center devices, followed by the acquisition of the local device information via system communication, and the start of cyclic communication. Only then, data communication is possible for the motion devices.

(1) Opening devices board by board

First, get the number of MNTs and prepare the same number of data structures for board recognition.

Next, get the device information on the MNTs, and pass target MNT's device information to the device opening function.

Then, the target MNT is opened.

Also, the device opening function returns a device handle value for accessing the MNT.

This process needs to be performed device by device when using two or more MNTs.

1. Get the number of MNTs.
mnt520_GetMstBrdCount()
2. Get the PCI device information on MNTs.
mnt520_GetMstBrdInfo()
3. Open the MNT.
mnt520_OpenMstBrd()

(2) Center device initial settings

After opening the MNTs successfully, start system communication to set local device information automatically.

Specify the initial values of port data and start cyclic communication.

Once cyclic communication is started, data communication becomes possible. Now, initialize the local devices.

1. Get local device (DIO, motion) information.
mnt520_rSysLclInfo()
2. Write a command to the center device.
mnt520_wCenCmd()
3. Read the center main status.
mnt520_rCenMsts()
4. Set the interrupt-on-input-change setting for local device ports.
mnt520_wLclSetInt()
5. Set the interrupt-on-input-change setting for motion device ports.
mnt520_wPclSetInt()

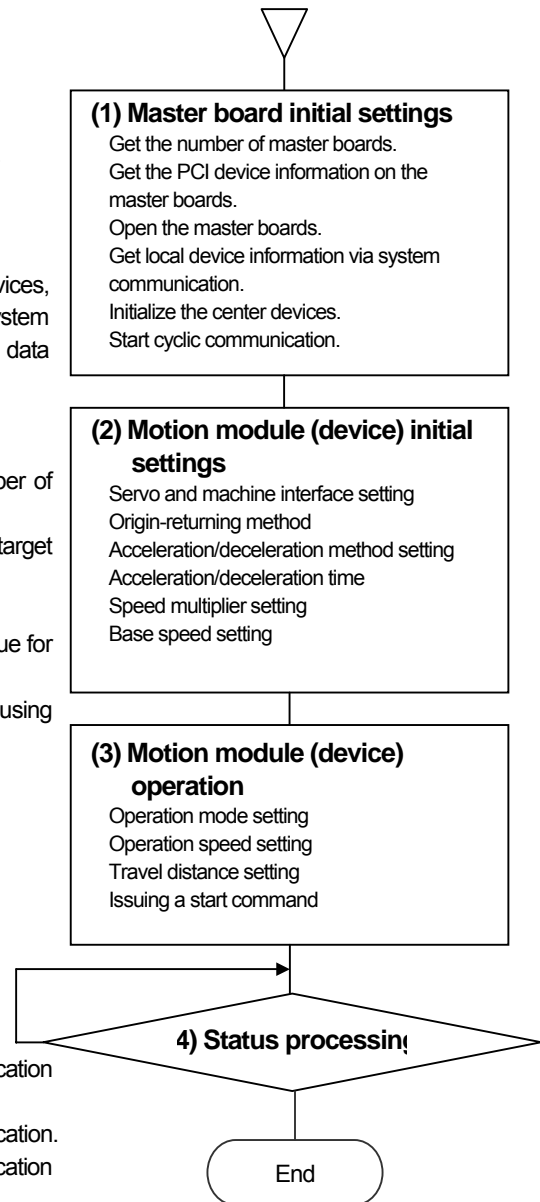


Figure 8. 1-1 Basic programming procedure

8.2 Motion module (device) initial settings

After the program is started, the center devices are initialized. Then, cyclic communication is started, which causes data communication to be enabled. Now, the next step is performing the initial settings for each motion device.

The items to be configured are the environment setting registers (RENV1 to RENV6), the operation mode register (partly), the event factor setting register, and the generic output ports.

With the following functions, set machine sensor's input specifications, the servo interfaces, the origin-returning method, and the event factor.

◆mnt520_wPclReg() ... Write to a motion device registers.

◆mnt520_wPclPortB() ... Write to a motion device port.

This section explains about various initial settings as classified below.

1	Machine sensor's input specifications	(1) ± ELS input (2) DLS input (3) OLS input
2	Servo interface settings	(1) Command pulse output form (2) Encoder A/B-phase input form (3) Servo alarm input (4) In-position input (5) Servo ready input (6) Clear-servo-error-counter output (7) Servo on, servo reset, servo torque, servo gain outputs
3	Origin-returning method settings	(1) Origin-returning method (2) Encoder Z-phase input specifications
4	Event factor settings	(1) Event factor settings

Table 8. 2-1 Initial settings for motion devices

8.2.1 Input specifications settings of machine sensors

The table below shows items related to the initialization of machine sensor's input specifications.

No.	Item	Signal name	Settings	
1	ELS input specifications	±ELS	1. Input polarity	IOPOB:ELL(b7)
			Enabled after setting RENV2, b7 to 0 (RENV2, b7 to 0 setting: 11111111)	
			NC (detected upon coupler current OFF)	1
			NO (detected upon coupler current ON)	0
			NO when ELS is unused (IOPOB:ELL=0), terminal disconnected	
			2. Operation upon ELS ON	RENV1:b3
2	DLS input specifications	DLS	1. DLS enabled/disabled	RMD:b8
			DLS input ignored (input status checking by RSTS possible)	0
			Deceleration (deceleration stop) upon DLS ON	1
			2. Input polarity	RENV1:b6
			NC (detected upon coupler current OFF)	0
			NO (detected upon coupler current ON)	1
			NO when DLS is unused (RENV1: b6 = 1, RMD: b8 = 0), terminal disconnected	
			3. Operation upon DLS ON	RENV1:b4
			Deceleration only	0
			Deceleration stop	1
			4. DLS input latch (used when DLS signal width is short)	RENV1:b5
			Latch disabled	0
Latch enabled	1			
3	OLS input polarity	OLS	1. Input polarity	RENV1:b7
			NC (detected upon coupler current OFF)	0
			NO (detected upon coupler current ON)	1
			NO when OLS is unused (RENV1: b7 = 1), terminal disconnected	

Table 8. 2-2 Settings of machine sensor's input specifications

8.2.2 Servo interface settings

The table below shows items related to the initialization of servo interfaces.

No.	Item	Signal name	Settings	
1	Command pulse output form	CWP ,CWN CCWP,CCWN	1. Command pulse output form	RENV1:b0-b2
			2-pulse mode (CW/CCW)	100
			Common command pulse mode (common pulse + DIR signal)	000
2	Encoder A/B-phase input form	AP,AN BP,BN	1. Multiplier setting	RENV2:b10,9
			x1	00
			x2	01
			x4	10
			Up/down	11
			2. Input forbidden (Forbidden until origin-returning completion in using some servo driver)	RENV2:b18,17
3	Servo alarm input specifications	SVALM	1. Input polarity	RENV1:b9
			NC (detected upon coupler current OFF)	0
			NO (detected upon coupler current ON)	1
			2. Operation upon input	RENV1:b8
			Immediate stop	0
			Deceleration stop	1
4	In-position input specifications	INPOS	1. INPOS control enabled/disabled	RMD:b9
			INPOS not applied to operation completion status	0
			Operation completion upon INPOS input ON	1
			2. Input polarity	RENV1:b22
			NC (detected upon coupler current OFF)	0
			NO (detected upon coupler current ON)	1
5	Servo ready input specifications	SVRDY	NO when SVALM is unused (RENV1: b9 = 1), terminal disconnected	
			1. Input polarity	RENV1:b24
			NC (detected upon coupler current OFF)	0
			NO (detected upon coupler current ON)	1
			NO when SVRDY is unused (RENV1: b24 = 1), terminal disconnected	
			PCS function unavailable when used as servo ready input. Servo ready input unavailable when PCS function is used.	
6	Clear-servo-error-counter output specifications	SVCTRCL	1. Auto output upon origin-returning completion	RENV1:b11
			Auto output disabled	0
			Auto output enabled	1
			2. Auto output upon immediate stop by ±ELS, SVALM input	RENV1:b10
			Auto output disabled	0
			Auto output enabled	1
			3. Output pulse width setting	RENV1:b14-12
13 msec (recommended)	100			
7	Servo on, servo reset, servo torque, servo gain output specifications	SVON SVRST SVTL SVGAIN	1. Control by generic output ports Enabled after setting RENV2, b7-0 (RENV2, b7 to 0 setting: 11111111)	IOPOB:b3-0
			SVON	Output ON when b0 = 1
			SVRST	Output ON when b1 = 1
			SVTL	Output ON when b2 = 1
			SVGAIN	Output ON when b3 = 1

Table 8. 2-3 Servo interface settings

8.2.3 Origin-returning method settings

The following origin-related items need to be initialized for a mechanical system using an origin.

No.	Item	Settings
1	Origin-returning method (ORGmode)	ORGmode (b3 to 0) of RENV3: 0000-1100 See "(1) Origin sensor settings and origin-returning methods" on the next page.
2	OLS input polarity	ORGL (b7) of RENV1 0: NO, 1: NC
3	Encoder Z-phase signal input polarity	EZL (b12) of RENV2 0: Falling edge, 1: Rising edge
4	Encoder Z-phase count setting	EZD 3 to 0 (b7 to 4) of RENV3 0000 (1st) to 1111 (16th) (count - 1)
5	± ELS input polarity	ELL (b7) of IOPOB 0: NO, 1: NC
6	Stop upon ± ELS ON	ELM (b3) of RENV1 0: Immediate stop, 1: Deceleration stop
7	Auxiliary speed setting (used in ORGmode = 1, 4, 6, 7)	RFA,RMG
8	Base speed setting	RFL,RMG
9	Operation speed setting	RFH,RMG
10	Setting for origin-returning completion	CU3R-CU1R (b22-20) of RENV3 CU1R (b20) = 1: Resets CTR1 (command position). CU2R (b21) = 1: Resets CTR2 (machine position). CU3R (b22) = 1: Resets CTR3 (general-purpose/error).
11	Setting for SVCTRCL signal auto output	EROR (b11) of RENV1 0: SVCTRCL signal not output when origin-returning completed 1: SVCTRCL signal automatically output when origin-returning completed

Table 8. 2-4 Origin-returning method settings

(1) Origin sensor settings and origin-returning methods

No	Origin-returning method	Constant-speed origin returning (FH constant-speed start)	High-speed origin returning (FH acceleration start)
0	OLS origin returning RENV3 ORGmode=0 (0000)	<p>Origin returning completion on OLS detection</p>	<p>Decelerates on OLS detection, finishes on reaching base speed (FL). Origin entry speed = FL speed</p>
1	OLS reversing and origin returning RENV3 ORGmode=1 (0001)	<p>Immediately stops on OLS detection, gets out of origin in reverse direction at FA speed, finishes on next OLS direction. Origin entry speed =FA speed</p>	<p>Decelerates on OLS detection and stops, gets out of origin in reverse direction at base speed, finishes on next OLS direction. Origin entry speed =FA speed</p>
2	OLS + ENC origin returning RENV3 ORGmode=2 (0010)	<p>After OLS detection, finishes on nth Z-phase detection.</p>	<p>Decelerates on OLS detection, finishes on nth Z-phase detection. * Finishes if last Z-phase occurs during deceleration.) Origin entry speed = FL speed</p>
3	OLS + ENC origin returning RENV3 ORGmode=3 (0011)		<p>After OLS detection, decelerates on nth Z-phase detection, finishes at base speed. Origin entry speed = FL speed</p>
4	OLS reversing + ENC origin returning RENV3 ORGmode=4 (0100)	<p>Immediately stops on OLS detection, reverses at FA speed, finishes on nth Z-phase detection. Origin entry speed =FA speed</p>	<p>Decelerates on OLS detection, reverses at FA speed on reaching base speed, finishes on nth Z-phase detection. Origin entry speed =FA speed</p>
5	OLS reversing + ENC origin returning RENV3 ORGmode=5 (0101)	<p>Immediately stops on OLS detection, reverses at base speed, finishes on nth Z-phase detection. Origin entry speed = FL speed</p>	<p>Decelerates on OLS detection, accelerates in reverse direction on reaching base speed, decelerates on nth Z-phase detection, finishes on reaching base speed. Origin entry speed = FL speed</p>
6	ELS-shared origin returning RENV3 ORGmode=6 (0110)	<p>Immediately stops on ELS detection, reverses at FA speed, finishes on ELS detection. Origin entry speed =FA speed</p>	<p>Decelerates on ELS detection and stops, reverses at FA speed, finishes on getting out of ELS. * ELS length > Deceleration distance)</p>
7	ELS reversing constant speed + ENC origin returning RENV3 ORGmode=7 (0111)	<p>Immediately stops on ELS detection, reverses at FA speed, finishes on nth Z-phase detection. Origin entry speed =FA speed</p>	<p>Decelerates on ELS detection and stops, reverses at FA speed, finishes on nth Z-phase detection. * ELS length > Deceleration distance</p>
8	ELS reversing constant speed + ENC origin returning RENV3 ORGmode=8 (1000)	<p>Immediately stops on ELS detection, reverses at base speed, finishes on nth Z-phase detection. Origin entry speed = FL speed</p>	<p>Decelerates on ELS detection, accelerates in reverse direction on reaching base speed, decelerates on nth Z-phase detection, finishes at base speed. * ELS length > Deceleration distance</p>

Table 8. 2-5 Origin-returning methods

No	Origin-returning method	High-speed origin returning (FH acceleration start)	
9	Origin-returning method ORGmode=9 (1001)		<p>Decelerates—and CTR2 cleared—on OLS detection, stops and reverses, finishes on reaching CTR2 zero-point. * CTR2 count input: command pulse (recommended) (RENV3 b9-b8='01')</p>
10	ENC reversing + CTR2 origin returning ORGmode=10 (1010)		<p>Decelerates—and CTR2 cleared—on nth Z-phase after OLS detection, stops and reverses, finishes on reaching CTR2 zero-point. * CTR2 count input: encoder (recommended) (RENV3 b9-b8='00')</p>
11	ENC reversing + CTR2 origin returning ORGmode=11 (1011)		<p>Decelerates on OLS detection, stops and reverses, decelerates—and CTR2 cleared—on nth Z-phase, stops and reverses, finishes on reaching CTR2 zero-point. * CTR2 count input: encoder (recommended) (RENV3 b9-b8='00')</p>
12	ELS reversing + CTR2 origin returning ORGmode=12 (1100)		<p>Decelerates on ELS detection, stops and reverses, decelerates—and CTR2 cleared—on nth Z-phase, stops and reverses, finishes on reaching CTR2 zero-point. * CTR2 count input: encoder (recommended) (RENV3 b9-b8='00')</p>

Table 8. 2-6 Origin-returning methods (by CTR2 reference)

Origin initialization needs to be performed for a mechanical system using an origin. As for the speed, acceleration/deceleration in origin returning, see the next section, “8.3.2 Speed pattern settings.”

- (1) Origin configuration ... Select an origin configuration from four largely-categorized patterns shown below. (Totally 13 patterns are available.)
 (1) Sensor origin (OLS), (2) Sensor origin and encoder Z-phase,
 (3) ELS shared origin, (4) ELS shared origin and encoder Z-phase
 (2) Origin-returning speed ... Constant-speed return, high-speed return (speed with acceleration/deceleration)

As for origin-returning methods, select from “Table 8. 2-5 Origin-returning methods” and “Table 8. 2-6 Origin-returning methods (by CTR2 reference).”

- Set ORGmode (bits 3 to 0) of RENV3 to the ORGmode value for the operation-setting method selected from “Table 8. 2-5 Origin-returning methods” and “Table 8. 2-6 Origin-returning methods (by CTR2 reference).”
- It is possible to clear the counter selected from CTR1 to CTR4 at the desired time during origin-returning operation.
 RENV3 bit20 = '1' ... CTR1 (command position counter), bit21 = '1' ... CTR2 (machine position counter)
 bit22 = '1' ... CTR3 (general-purpose/error counter)

Note: Timing for clearing is indicated by the symbol “●” in the figures.

- As for the origin returning methods No. 9 to 12 in “Table 8. 2-6 Origin-returning methods (by CTR2 reference)”, CTR2 is cleared upon OLS detection while origin-returning is in progress, and positioning is performed with the travel distance derived from the sign inversion of the CTR2 value at the point of deceleration completion. Therefore, the CTR2 value may not be “0” when the origin-returning operation is completed.

8.2.4 Event factor settings

Specify RIRQ (event factor setting register) in the initial setting.

If an event is caused by the factor specified in RIRQ, bit 3 of MMSTS is turned to "1."

To check the event factor then, read RIST.

Normally, set only bit 0 to "1." (Reporting normal end)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	IRSA	IRSD	IROL	IRLT	IRCL	IRC3	IRC2	IRC1	IRDE	IRDS	IRUE	IRUS	IREN
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8. 2-1 RIRQ: Bit configuration of the Event Factor Setting register

8.3 Motion module (device) operation

To start an operation after the initial setting process, write a start command according to the operation mode.

The operation mode can be set by specifying MOD (operation mode) of RMD (operation mode register).

Set the desired speed pattern by specifying RFL (base speed), RFH (operation speed), RUR/RDR (acceleration /deceleration rate), RMG (speed multiplier), RDP (deceleration starting point), RUS/RDS (acceleration/deceleration S-curve section), and RFA (auxiliary speed). If needed, write a travel distance (relative coordinate value) to RMV (travel distance register).

Lastly, write a start command (such as acceleration or constant-speed start) to start an operation.

8.3.1 Operation modes

Specify the operation mode (MOD) before issuing a start command. The MOD setting can be classified largely into 6 modes. MOD is specified with bits 6 to 0 of RMD (operation mode register).

No	Category	MOD	Operation mode
1	Continuous operation by command	00h	Continuous operation in (+) direction
		08h	Continuous operation in (-) direction
2	Positioning operation	41h	Positioning operation
		44h	Return to zero-point of command position
		45h	Return to zero-point of machine position
		46h	One-pulse operation in (+) direction
		4Eh	One-pulse operation in (-) direction
		47h	Timer operation
		3	Origin-returning operation
18h	Origin-returning operation in (-) direction		
12h	Getting-out-of-origin operation in (+) direction		
1Ah	Getting-out-of-origin operation in (-) direction		
15h	Origin search in (+) direction		
1Dh	Origin search in (-) direction		
4	Operation to ELS, SLS	20h	Operation to +ELS or +SLS
		28h	Operation to -ELS or -SLS
		22h	Getting-out-of +ELS or +SLS operation
		2Ah	Getting-out-of -ELS or -SLS operation
5	Z-phase Operation	24h	Operation in (+) direction until Z-phase count completion
		2Ch	Operation in (-) direction until Z-phase count completion
6	Manual pulsar operation	01h	Continuous operation by pulsar input
		51h	Positioning operation by pulsar input
		54h	Return to zero-point of command position by pulsar input
		55h	Return to zero-point of machine position by pulsar input

Table 8. 3-1 List of operation modes

(1) Continuous operation by command

Operation is started by issuing a start command in this mode, and continued until the issuance of a stop command.

(+) direction operation MOD=00h (-) direction operation MOD=08h	Movement	1. Mode setting Set MOD to "00h" for (+) direction, "08h" for (-) direction. 2. Speed setting Specify the operation speed in the RFH (operation speed) register. 3. Start command Write "0053h" for acceleration or "0051h" for FH constant-speed. 4. Stop command Write "004Ah" for deceleration stop or "0049h" for immediate stop.
	ELS	1. When moving in +/- direction, operation stops upon +/- ELS detection, and then reverses by a start command. 2. Specify bit 3 of RENV1 to select the stop method used when ELS is detected.

Table 8. 3-2 Continuous operation by command

(2) Positioning operation

ELS, DLS, and SLS provide the same operation in the positioning patterns except timer operation.

Positioning MOD=41h	Issuing a start command in this mode provides a movement with the travel distance specified by RMV (travel distance register). The direction is determined by the sign of RMV. (0 < RMV: + direction, 0 > RMV: - direction)	
	Movement	1. Mode setting Set MOD to "41h." 2. Travel distance pulse count Specify a relative travel distance in RMV. The direction is determined by the sign: [-134, 217, 728 < travel distance < +134, 217, 727 (relative position feed)] When started with the travel distance set to 0, operation is immediately completed without pulse output. 3. Speed setting Specify the operation speed in RFH (operation speed setting register). 4. Start command Write "0053h" for acceleration, "0051h" for FH constant-speed. 5. Stop command To stop operation in progress, write "004Ah" for deceleration stop, "0049h" for immediate stop.
	ELS	1. When moving in +/- direction, operation stops upon +/- ELS detection, and then reverses by a start command. 2. Specify bit 3 of RENV1 to select the stop method used when ELS is detected.
Return to zero-point of command position MOD=44h	Issuing a start command in this mode provides a movement until CTR1 (command position counter) reaches 0.	
	Movement	1. Mode setting Set MOD to "44h." 2. Speed setting Specify the operation speed in RFH (operation speed setting register). 3. Start command Write "0053h" for acceleration, "0051h" for FH constant-speed. 4. Stop command To stop operation in progress, write "004Ah" for deceleration stop, "0049h" for immediate stop.
	ELS	Same as above
Return to zero-point of machine position MOD=45h	Issuing a start command in this mode provides a movement until CTR2 (machine position counter) reaches 0.	
	Movement	1. Mode setting Set MOD to "45h." 2. Speed setting Specify the operation speed in RFH (operation speed setting register). 3. Start command Write "0053h" for acceleration, "0051h" for FH constant-speed. 4. Stop command To stop operation in progress, write "004Ah" for deceleration stop, "0049h" for immediate stop.
	ELS	Same as above
One-pulse operation in (+) direction MOD = 46h in (-) direction MOD = 4Eh	When a start command is issued without the RMV (travel distance register) setting, a 1-pulse movement is given and completed.	
	Movement	1. Mode setting Set MOD to "46h" for (+) 1 pulse, "4Eh" for (-) 1 pulse operation. 2. Speed setting Specify the operation speed in RFH (operation speed setting register). 3. Start command Write "0051h" for FH constant-speed.
	ELS	Same as above

(To be continued)

(Continued from previous page)

Timer operation MOD=47h	This mode uses internal operation time as a timer. RMV (travel distance register) serves as a timer counter. RMV setting range [1 to 134,217,727: positive value] The product of RMV pulse count, RFH (operation speed setting register), and RMG (speed multiplier setting register) is the time counted. When the counter reaches 0, the operation is completed. (Example: Time is 100ms where RMV = 100 [pulses], RFH = 1000, RMG = 199.)	
	Movement	1. Mode setting Set MOD to "47h." 2. Timer speed setting Specify the timer speed (pps) in RFH and RMG. (Example: RFH = 1000, RMG = 199 (1000 pps = 1 ms)) 3. Travel distance pulse count Specify the number of pulses in RMV. 4. Start command Write "0051h" (FH constant-speed). 5. Stop command To stop operation in progress, write "0049h" for immediate stop.
	Note	1. Set bit 11 of the RMD register to 0. (Operation completion timing: METM) 2. No command pulse output. CTR1 (command position) counting disabled. 3. ±ELS, DLS, SLS, and INPOS ignored. 4. SVALM signal enabled during timer operation. 5. Disabled functions are backlash compensation, vibration suppression, and timer function when direction changed.

Table 8. 3-3 Positioning operation

(3) Origin-returning operation

The origin-returning operation uses OLS, EX, and ±ELS inputs depending on the operation mode.

As for OLS input signals, specify the logic setting in the RENV1 (environment setting 1) register, and monitor the terminal status by the RSTS (expansion status) register.

Select the encoder Z-phase input polarity in RENV2 (environment setting 2). Specify the encoder Z-phase input count, which determines origin-returning completion, in RENV3 (environment setting 3). Monitor the input status by the RSTS register.

Specify the ±ELS input polarity in the generic output port (IOPOB). Select the operation mode given when ±ELS input is ON (immediate stop/ deceleration stop) in RENV1. Monitor the input status by RSTS (expansion status).

To execute the origin-returning operation, select its mode such as origin-returning, getting-out-of-origin, or origin search in MOD (operation mode) of RMD; and then write a start command.

<Origin-returning operation settings and the related statuses>

No.	Item	Registers to be set
1	Origin-returning method (ORGmode)	ORGmode (b3 to 0) of RENV3: 0000-1100 See "8.2.3 Origin-returning method settings."
2	OLS input polarity	ORGL (b7) of RENV1 0: NO, 1: NC
3	Reading OLS input status	SORG (b8) of RSTS 0: OFF, 1: ON
4	Encoder Z-phase signal input polarity	EZL (b12) of RENV2 0: Falling edge, 1: Rising edge
5	Encoder Z-phase count setting	EZD 3 to 0 (b7 to 4) of RENV3 0000 (1st) to 1111 (16th) (count - 1)
6	Reading encoder Z-phase input status	SEZ (b16) of RSTS 0: OFF, 1: ON
7	± ELS input polarity	ELL (b7) of IOPOB 0: NO, 1: NC
8	Stop upon ± ELS ON	ELM (b3) of RENV1 0: Immediate stop, 1: Deceleration stop
9	Reading ± ELS input status	SPEL (b6) and SMEL (b7) of RSTS 0: OFF, 1: ON
10	Auxiliary speed setting (used in ORGmode = 1, 4, 6, 7)	RFA, RMG
11	Base speed setting	RFL, RMG
12	Operation speed setting	RFH, RMG
13	Setting for origin-returning completion	CU3R-CU1R (b22-20) of RENV3 CU1R (bit 20) = 1: Resets CTR1 (command position). CU2R (bit 21) = 1: Resets CTR2 (machine position). CU3R (bit 22) = 1: Resets CTR3 (general-purpose/error).
14	Setting for SVCTRCL signal auto output	EROR (b11) of RENV1 0: SVCTRCL signal not output when origin-returning completed 1: SVCTRCL signal automatically output when origin-returning completed

Table 8. 3-4 Origin-returning operation settings and the related statuses

Origin-returning operation in (+) direction MOD = 10h in (-) direction MOD = 18h	Movement	<p>After a start command is written, this mode continues to provide a movement until the origin-returning completion condition is fulfilled. When origin-returning is completed, the counter is reset automatically and SVCTRCL (clear-error-counter) signal output is possible. Set the basic origin-returning method, and enable/disable the counter reset in RENV3. Enable/disable the SVCTRCL signal auto-output in RENV1.</p> <p>1. Mode setting Set MOD to "10h" for (+) direction, "18h" for (-) direction. 2. Speed setting Specify the operation speed in RFH (operation speed setting register). 3. Start command Write "0053h" for acceleration, "0051h" for FH constant-speed. 4. Stop command To stop operation in progress, write "004Ah" for deceleration stop, "0049h" for immediate stop.</p>
	ELS	Unless the ELS shared origin is set, a normal ELS operation is performed upon ELS detection during origin returning. (immediate stop/ deceleration stop)
Getting-out-of-origin operation in (+) direction MOD = 12h in (-) direction MOD = 1Ah	Movement	<p>After a start command is written, this mode continues to provide a movement until getting out of the origin (OLS ON state). Before operation start... OLS = OFF: If a start command is executed, movement will immediately stop. (stop normally). OLS = ON : Operation stops one pulse after OLS turns OFF.</p> <p>1. Mode setting Set MOD to "12h" for (+) direction, "1Ah" for (-) direction. 2. Speed setting Specify the operation speed in RFH (operation speed setting register). 3. Start command Write "0051h" for FH constant-speed. 4. Stop command To stop operation in progress, write "0049h" for immediate stop.</p>
	ELS	A normal ELS operation is performed upon ELS detection while getting out of the origin. (immediate stop/ deceleration stop)
Origin search in (+) direction MOD = 15h in (-) direction MOD = 1Dh	Movement	<p>This mode provides additional functions to origin-returning operation. The following operation blocks are offered: 1. "Origin-returning operation (operation in "ORGmode = 0", movement from OLS OFF to OLS ON)" in the direction opposite to the one specified. 2. "Getting-out-of-origin operation by positioning operation" in the direction opposite to the one specified. 3. "Origin-returning operation (movement in RENV3 ORGmode setting)" in the specified direction. There are three positional relationships between the current position and OLS. 1. OLS is OFF: 3 2. OLS is ON: 2 --> 3 3. OLS and ELS: After reversed upon ELS ON, 1 --> 2 --> 3</p> <p>1. Mode setting Set MOD to "15h" for (+) direction, "1Dh" for (-) direction. 2. Speed setting Specify the operation speed in RFH (operation speed setting register). 3. Start command Write "0053h" for acceleration, "0051h" for FH constant-speed. 4. Stop command To stop operation in progress, write "004Ah" for deceleration stop, "0049h" for immediate stop.</p>
	Note	<p>1. Origin-returning operation is based on the "RENV3: ORGmode" setting. 2. The travel distance for the getting-out-of-origin operation by positioning operation is the value set in RMV. This value is used repeatedly until getting out of the origin. Use a positive value (1 to 134,217,727) for setting RMV.</p>

Table 8. 3-5 Origin-returning operations

(4) ELS, SLS operation

This mode, available in two kinds of operations, provides operation until reaching the set limit.

Select the \pm ELS input polarity at the generic output port (IOPOB), and specify the operation pattern to be performed upon input ON (immediate stop/ deceleration stop) in RENV. Monitor the input status by RSTS (expansion status).

For the SLS (soft-limit) settings, see "Soft-limit function."

Operation to +ELS or +SLS MOD = 20h to -ELS or -SLS MOD = 28h	Movement	If an operation command is issued while ELS and SLS are OFF: This mode provides a movement until ELS or SLS is ON and finishes it normally. If an operation command is issued while ELS or SLS is ON: This mode finishes the operation normally without providing a movement.
		1. Mode setting Set MOD to "20h" for (+) direction, "28h" for (-) direction. 2. Speed setting Specify the operation speed in RFH (operation speed setting register). 3. Start command Write "0053h" for acceleration, "0051h" for FH constant-speed. 4. Stop command To stop operation in progress, write "004Ah" for deceleration stop, "0049h" for immediate stop.
	ELS	The pattern of stop upon ELS detection is based on the ELM (bit 3) setting in RENV1: immediate stop (0)/ deceleration stop (1).
Getting-out-of +ELS or +SLS MOD = 22h -ELS or -SLS MOD = 2Ah	Movement	If an operation command is issued while ELS or SLS is ON: This mode provides a movement until ELS or SLS is OFF and finishes it normally. If an operation command is issued while ELS and SLS are ON: This mode finishes the operation normally without providing a movement.
		1. Mode setting Set MOD to "22h" for (+) direction, "2Ah" for (-) direction. 2. Speed setting Specify the operation speed in RFH (operation speed setting register). 3. Start command Write "0051h" for FH constant-speed. 4. Stop command To stop operation in progress, write "0049h" for immediate stop.

Table 8. 3-6 ELS, SLS operation

(5) Z-phase Operation

This mode provides a movement until the completion of counting encoder Z-phase signals the number of times set in EZD of RENV3 plus 1.

After a start command is written, the Z-phase operation mode performs an immediate stop upon the completion of counting encoder Z-phase signals the set number of times.

Specify the input polarity of EZ input signals in RENV2 (environment setting 2), encoder Z-phase signal count in RENV3 (environment setting 3). Monitor the encoder Z-phase input status by the RSTS (expansion status) register.

In (+) direction MOD = 24h In (-) direction MOD = 2ch	Movement	Z-phase count setting range is 1 to 16 (EZD set value: 0 to 15). Be sure to select "constant-speed" operation.
		1. Mode setting Set MOD to "24h" for (+) direction, "2Ch" for (-) direction. 2. Speed setting Specify the operation speed in RFH (operation speed setting register). 3. Start command Write "0051h" for FH constant-speed. 4. Stop command To stop operation in progress, write "0049h" for immediate stop.

Table 8. 3-7 Z-phase operation

(6) Manual pulsar operation

This mode provides an operation by pulsar input.

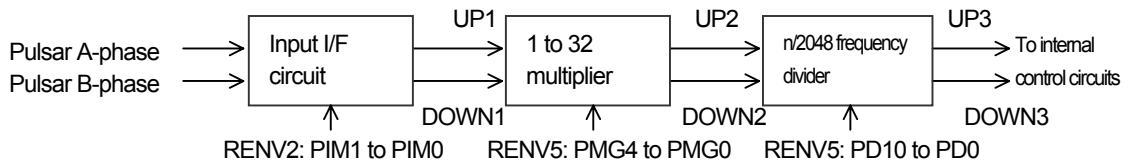
Manual pulsar operation is available in such modes as continuous operation, positioning operation, return to zero-point of synchronous command position, return to zero-point of synchronous machine position by pulsar input. Issuing a start command in any of the above modes enables pulsar signal input, which causes the command output of pulses. The pulsar input shares the same input terminal with the encoder input. While the pulsar input is used, the encoder input cannot be set.

With the input interface circuit for pulsar signals, pulsar A/B-phase input is available in four types which are set in the RENV2 (environment setting 2) register.

- ◆ Quadrature-encoded signals (1, 2, 4 multiplier) input
- ◆ 2-pulse input (up/down)

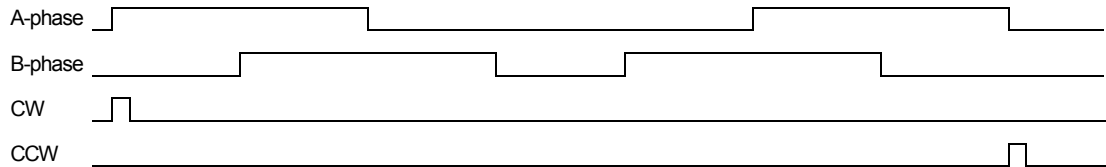
In addition to 1 to 4 multiplier mentioned above, a 1 to 32-multiplier circuit and (1 to 2048)/2048 frequency divider are implemented.

Specify PMG4 to 0 of RENV5 for the 1 to 32 multiplier setting, and PD10 to 0 of RENV5 for n/2048 frequency division setting.

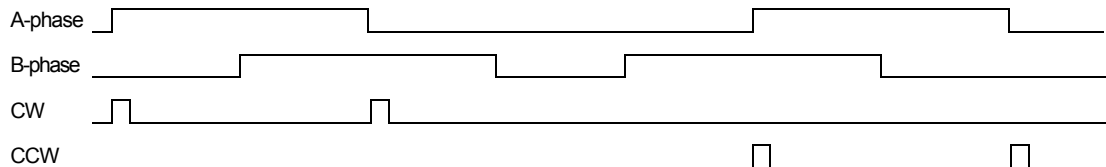


■ The following figures show CW, CCW signals according to the PIM1 and PIM0 settings of RENV2.

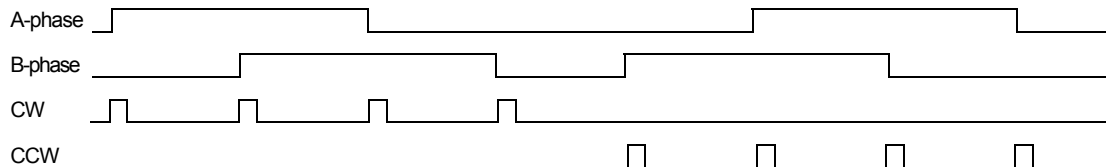
(1) Quadrature-encoded signal input: x1 multiplier (PIM = 00)



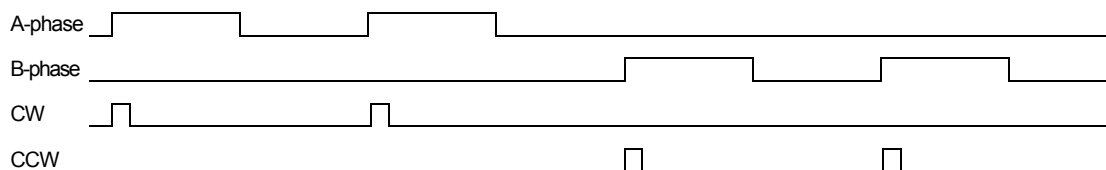
(2) Quadrature-encoded signal input: x2 multiplier (PIM = 01)



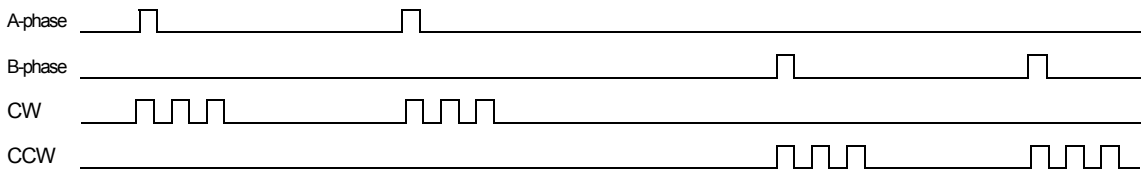
(3) Quadrature-encoded signal input: x4 multiplier (PIM = 10)



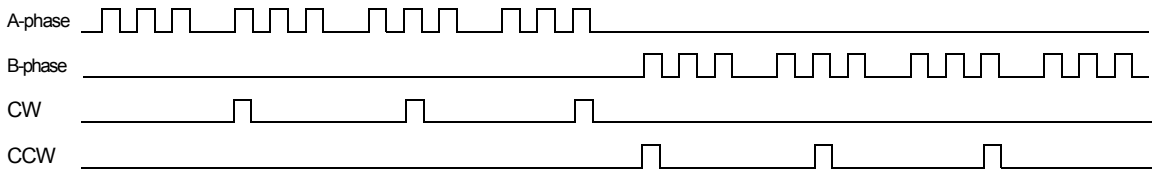
(4) 2-pulse input (PIM = 11)



■ The following figure shows an operation when the 1 to 32-multiplier circuit is set to 3 multiplier (PMG = 2 in RENV5).



■ The following figure shows an operation when the division ratio of the n/2048 frequency divider is set to 512/2048 (PD = 512 in RENV5).



Use the FH constant-speed start command (0051h) to start the pulsar input mode.

Pulsar inputs cause internal pulses to be output at FH speed with periodic interruptions.

Therefore, there is a difference in timing between pulsar input and output pulses, as long as an internal pulse period at the maximum.

The maximum input frequency of pulsar signals is limited by the FH speed.

An error occurs if pulsar signal inputs from the encoder terminal are changed simultaneously, or if there is an overflow of the input buffer counter (16 bits) due to exceeding input frequencies.

The cause of an error can be checked by REST (error status).

“FP” below represents the pulsar input frequency.

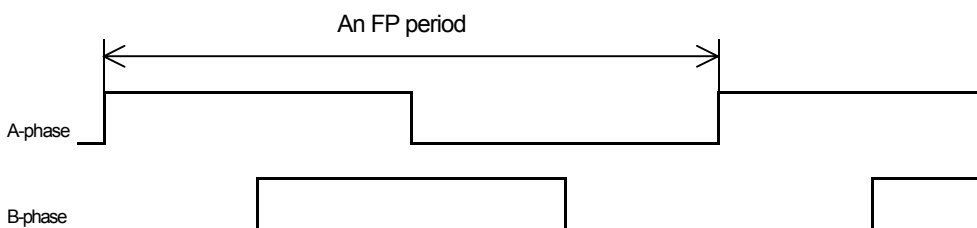
$FP < (\text{Set speed}) / (\text{Input I/F multiplier value}) / (\text{PMG set value} + 1) / (\text{PD set value} / 2048)$ where the PD set value $\neq 0$

$FP < (\text{Set speed}) / (\text{Input I/F multiplier value}) / (\text{PMG set value} + 1)$ where the PD set value = 0

<Examples of relationships between FH speed [pps] and pulsar input frequency FP [pps]>

Pulsar A/B-phase input setting	PMG (multiplier) value RENV5: PMG4 to 0 0	PD (division ratio) value RENV5: PD10 to 0	Range of use
Quadrature-encoded signals: x1 (RENV2:PIM=00)	0 (x1)	0 (0000h)	FP < FH
	0 (x1)	1024 (0400h)	FP < FH×2
	2 (x3)	0 (0000h)	FP < FH/3
Quadrature-encoded signals: x2 (RENV2:PIM=01)	0 (x1)	0 (0000h)	FP < FH/2
	0 (x1)	1024 (0400h)	FP < FH
	2 (x3)	0 (0000h)	FP < FH/6
Quadrature-encoded signals: x4 (RENV2:PIM=10)	0 (x1)	0 (0000h)	FP < FH/4
	0 (x1)	1024 (0400h)	FP < FH/2
	2 (x3)	0 (0000h)	FP < FH/6
2-pulse input (RENV2:PIM=11)	0 (x1)	0 (0000h)	FP < FH
	0 (x1)	1024 (0400h)	FP < FH×2
	2 (x3)	0 (0000h)	FP < FH/3

Table 8. 3-8 Examples of relationships between FH speed [pps] and pulsar input frequency FP [pps]



Note. If pulsar A/B-phase input frequencies fluctuate, set the “FP period” above to the shortest period rather than the average one.

<Pulsar A/B-phase input settings>

Item	Register: bits	Setting value				
		b15,14	00	01	10	11
Input mode	RENV2:PIM1,0 (b15,14)	Multiplier	x1	x2	x4	Up/down
Count polarity	RENV2:PDIR (b16)	0: Phase A lead, 1: Phase B lead				
Input mask	RENV2:POFF (b18)	0: Input allowed, 1: Input forbidden				
Operation status reading	RSTS :CND (b3-0)	0101: Waiting pulsar A/B-phase input				
In-phase input error	REST :ESPE (b14)	1: Pulsar A/B in-phase input error occurred				
Input buffer overflow	REST :ESPO (b9)	1: Buffer overflow occurred				

Table 8. 3-9 Pulsar A/B-phase input settings

Start command	Issue the FH constant-speed command, "0051h." Specify the maximum pulsar input frequency in RFH and RMG.	
ELS	Pulsar output is stopped upon ELS detection, but operation in the reverse direction is possible. The error status is not changed when stopped by ELS input.	
Mode canceling	Issue the immediate stop command, "0049h."	
Pulsar continuous operation MOD=01h	Movement	This mode provides a continuous operation by pulsar input. Command pulses are output by handle feed (normal handle feed). When pulsar signals are input after the issuance of the start command, command pulses are output in synchronization with pulsar signals. The direction is determined by the pulsar signal input and the pulsar input count polarity setting.
		<ol style="list-style-type: none"> 1. Mode setting Set MOD to "01h." 2. Maximum pulsar input frequency setting Specify the maximum pulsar input frequency in RFH and RMG. 3. Start command Write "0051h" for FH constant-speed.
Pulsar positioning operation MOD=51h	Movement	This mode provides a positioning operation in synchronization with pulsar input signals (direction ignored), and finishes the operation upon reaching the target position. In the positioning operation, the RMV set value is used as relative position data. Pulsar input after the completion will be ignored. If started with RMV set to "0", the operation is immediately stopped without pulse output.
		<ol style="list-style-type: none"> 1. Mode setting Set MOD to "51h." 2. Travel distance pulse count Specify the relative travel distance in RMV. Determine the direction with the sign. 3. Maximum pulsar input frequency setting Specify the maximum pulsar input frequency in RFH and RMG. 4. Start command Write "0051h" for FH constant-speed.
Return to zero-point of command position by pulsar input MOD=54h	Movement	This mode provides a movement in synchronization with pulsar input signals (direction ignored) until the command position reaches "CTR1 = 0." The direction and the travel distance depend on the command position (CTR1) at the time of the operation command issuance.
		<ol style="list-style-type: none"> 1. Mode setting Set MOD to "54h." 2. Maximum pulsar input frequency setting Specify the maximum pulsar input frequency in RFH and RMG. 3. Start command Write "0051h" for FH constant-speed.
Return to zero-point of machine position by pulsar input MOD=55h	Movement	This mode provides a movement in synchronization with pulsar input signals (direction ignored) until the machine position reaches "CTR2 = 0." The direction and the travel distance depend on the machine position (CTR2) at the time of the operation command issuance.
		<ol style="list-style-type: none"> 1. Mode setting Set MOD to "55h." 2. Maximum pulsar input frequency setting Specify the maximum pulsar input frequency in RFH and RMG. 3. Start command Write "0051h" for FH constant-speed.
Note	<ol style="list-style-type: none"> 1. The backlash compensation function is supported during the pulsar input mode, unless the pulsar input is reversed during backlash compensation process. 2. It is possible to perform an immediate stop during the multiplier operation, with the settings of PIM1, 0 and PMG4 to 0, by writing the immediate stop command "0049h." In this case, the total output pulse count may not be an integral multiple of the multiplier value. 	

Table 8. 3-10 Manual pulsar operation

8.3.2 Speed pattern settings

The speed patterns are set by partly specifying the registers listed in the table below and RMD (operation mode register).

Specify—normally during the initial setting—the acceleration/deceleration pattern (linear/S-curve: specified in bit 10 of RMD) and the acceleration/deceleration speed (acceleration/deceleration rate) in the corresponding registers.

Also, specify RMG for the speed multiplier, RFL for the base speed, and RFA for auxiliary speed (origin entry speed during origin returning, etc.) in advance.

The operation speed is not set in the initial setting process. Rather, when issuing a start command for starting an operation, specify the speed in RFH (operation speed setting register).

The set values in those registers do not have to be rewritten for each operation unless there is a need for the change of values.

Please note that “0” is invalid in some of the registers.

Register	Setting	Bit length	Setting range	R/W
RMV	Travel distance	28	-134,217,728 (08000000h) to 134,217,727 (07FFFFFFh)	R/W
RFL	Base speed	17	1 to 100,000 (000186A0h) (Note 2)	R/W
RFH	Operation speed	17	1 to 100,000 (000186A0h) (Note 2)	R/W
RUR	Acceleration rate	16	1 to 65,535 (0000FFFFh)	R/W
RDR	Deceleration rate (Note 1)	16	0 to 65,535 (0000FFFFh)	R/W
RMG	Speed multiplier	11	2 to 2,047 (000007FFh)	R/W
RDP	Deceleration starting point	24	0 to 16,777,215 (00FFFFFFh)	R/W
RUS	S-curve section in acceleration	16	0 to 50,000 (0000C350h) (Note 3)	R/W
RDS	S-curve section in deceleration	16	0 to 50,000 (0000C350h) (Note 3)	R/W
RFA	Auxiliary speed	17	1 to 100,000 (000186A0h) (Note 2)	R/W

Table 8. 3-11 Speed pattern setting registers

Note 1. With RDR set to 0, the deceleration rate will be the value set in RUR.

Note 2. A set value in the range of 186A0h to 1FFFFh (100000 to 131071) is always handled as 186A0h (100000).

Note 3. A set value in the range of 0C350h to 0FFFFh (50000 to 65535) is always handled as 0C350h (50000).

[Sections defined by the register data in acceleration/deceleration operation]

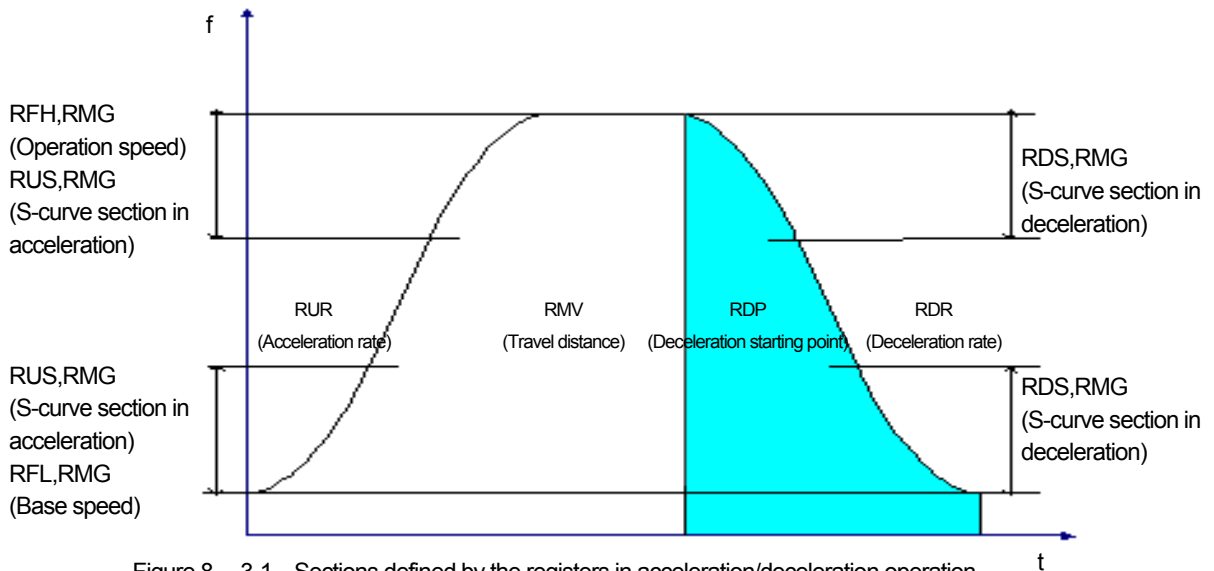


Figure 8. 3-1 Sections defined by the registers in acceleration/deceleration operation

8.3.3 Speed patterns

The speed pattern varies depending on the start and stop commands selected.

	Speed pattern	Continuous mode	Positioning operation mode
Acceleration/deceleration operation		<ol style="list-style-type: none"> 1. Write the acceleration start command (0053h). 2. Write the deceleration stop command (004Ah). 	<ol style="list-style-type: none"> 1. Write the acceleration start command (0053h). 2. Reached to the deceleration starting point. Or write the deceleration stop command (004Ah). When bit 12 (MSDP) is 1 in RMD (operation mode register) for setting deceleration starting point manually, where RDP (deceleration starting point register) is 0: Immediate stop.
		Immediate stop when the immediate stop command (0049h) is written.	
FH constant-speed operation		<ol style="list-style-type: none"> 1. Write the FH constant-speed start command (0051h). 2. Write the immediate stop command (0049h). 	<ol style="list-style-type: none"> 1. Write the FH constant-speed start command (0051h). 2. The positioning counter is 0. Or write the immediate stop (0049h) command.
		Deceleration stop when the deceleration stop command (004Ah) is written.	
FL constant-speed operation		<ol style="list-style-type: none"> 1. Write the FL constant-speed start command (0050h). 2. Write the immediate stop (0049h)/ deceleration stop (004Ah) command. 	<ol style="list-style-type: none"> 1. Write the FL constant-speed start command (0050h). 2. The positioning counter is 0. Or write the immediate stop (0049h)/ deceleration stop (004Ah) command.

Table 8. 3-12 Speed patterns

8.3.4 Speed Pattern Setting registers

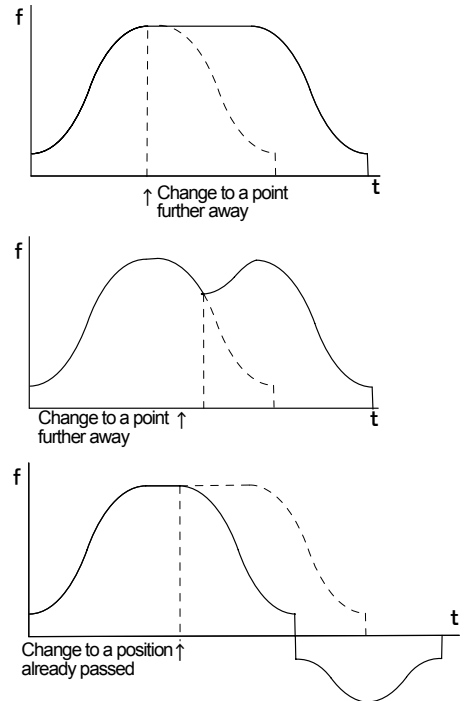
(1) RMV: Travel Distance register (28 bits)

This register is used to specify the travel distance in positioning operation. Settings differ depending on the operation mode. The setting range is from -134,217,728 to +134,217,727. A change in the RMV register during operation allows position override.

<Travel distance override>

The target position can be overridden by overwriting the RMV register. The target position is changed with reference to the start position.

- (1) If new data changes the original target position to a position further away during acceleration or constant-speed operation, the operation will be performed using the current speed pattern, and positioning will be completed at the new data position (new RMV value).
- (2) If new data changes the original target position to a position further away during deceleration, the movement will be reaccelerated from the current position up to the operation speed. And then, positioning will be completed at the new data position (new RMV value).
The reacceleration curve will be the same as the one given normally by the setting, "RFL = Fu", where Fu is the speed at the time of the data change.
- (3) If the position defined by the new data has already been passed, or if the data is changed during deceleration to make the new target position closer to the current position than the original target, the direction will be reversed after deceleration stop. And then, positioning will be completed at the new data position (new RMV value).



Acceleration/deceleration operation is used only when acceleration start is performed.

The target position data (RMV register value) can be overwritten any number of times until the end of the positioning operation.

Note 1. The position override is valid only when there is a movement (during FL/FH constant-speed, acceleration/deceleration, backlash operation).

An override immediately before the movement completion may not be applied.

If the override is ignored, SERR (bit2) of MMSTS is turned to 1, and the error cause can be checked through the REST register.

This error occurs when the override write command (0080h) is given to the RMV register during a stopped state. Therefore, a position override error occurs if 0080h is given to RMV during a stopped state before the operation startup.

(2) RFL: Base Speed register (17 bits)

This register is used to specify the base speed (stop speed) in operation that includes acceleration/deceleration.

The speed in FL constant-speed operation and the base speed in acceleration/deceleration operation can be specified in the range of 1 to 100,000 (186A0h). A value between 100,000 and 131,071 (186A0h to 1FFFFh) is always handled as 100,000.

The actual speed will be the value calculated with the RMG (Speed Multiplier Setting register) value.

$$\text{Base speed [pps]} = \text{RFL} \times \frac{200}{(\text{RMG} + 1)}$$

(3) RFH: Operation Speed Setting register (17 bits)

This register sets the operation speed.

A change in the RFH register during operation allows speed override.

The speed in FH constant-speed and acceleration/deceleration operations can be specified in the range of 1 to 100,000 (186A0h). A value between 100,000 and 131,071 (186A0h to 1FFFFh) is always handled as 100,000.

For acceleration/deceleration operation, set a value larger than the one in RFL (base speed).

The actual operation speed will be the value calculated with the RMG (Speed Multiplier Setting register) value.

$$\text{Operation speed [pps]} = \text{RFH} \times \frac{200}{(\text{RMG} + 1)}$$

(4) RUR: Acceleration Rate register (16 bits)

This register sets the acceleration rate.

Set the acceleration characteristic in acceleration/deceleration operation in the range of 1 to 65,535 (FFFFh).

The relationship between the set value and the acceleration time is shown by the following formulas:

1. Linear acceleration (Bit 10 (MSMD) = 0 in RMD register)

$$\text{Acceleration time [sec]} = \frac{(\text{RFH} - \text{RFL}) \times (\text{RUR} + 1)}{5,000,000}$$

2. S-curve acceleration without linear section (Bit 10 (MSMD) = 1 in RMD register and RUS = 0)

$$\text{Acceleration time [sec]} = \frac{(\text{RFH} - \text{RFL}) \times (\text{RUR} + 1)}{2,500,000}$$

3. S-curve acceleration including linear section (Bit 10 (MSMD) = 1 in RMD register and RUS > 0)

$$\text{Acceleration time [sec]} = \frac{(\text{RFH} - \text{RFL} + 2 \times \text{RUS}) \times (\text{RUR} + 1)}{5,000,000}$$

<Example of acceleration rate calculation>

Assume that the settings in the example are: linear acceleration (bit 10 (MSMD) of RMD register = 0), base speed = 10 pps, operation speed = 110kpps, acceleration time = 300 ms.

(1) Set the multiplier to "x2" to output 110 kpps:

$$\text{RMG} = 99(63\text{h}).$$

(2) To give the operation speed of 110 kpps with the multiplier set to the x2 mode:

$$\text{RFH} = 55000(\text{D6D8h})$$

(3) To give the start speed of 10 pps with the multiplier set to the x2 mode:

$$\text{RFL} = 5(0005\text{h})$$

(4) To give the acceleration time 300 ms, set RUR to 26.275 based on the formula that defines the relationship between the acceleration time and the RUR set value.

$$\text{Acceleration time [sec]} = \frac{(\text{RFH} - \text{RFL}) \times (\text{RUR} + 1)}{5,000,000}$$

$$0.3 \text{ [sec]} = \frac{(55000 - 5) \times (\text{RUR} + 1)}{5,000,000}$$

This leads to...

$$\text{RUR} = \frac{5,000,000 \times 0.3}{(55000 - 5)} - 1 = 26.275$$

Note that RUR must be set to "26" or "27" since RUR can accept only integers.

This means that the actual acceleration time will be either 297 ms (RUR = 26) or 308 ms (RUR = 27).

(5) RDR: Deceleration Rate register (16 bits)

This register sets the deceleration rate.

Set the deceleration characteristic in acceleration/deceleration operation in the range of 1 to 65,535 (FFFFh).

However, if RDR is set to 0, the deceleration rate will be the value set in RUR.

The relationship between the set value and the deceleration time is shown by the following formulas

1. Linear deceleration (Bit 10 (MSMD) = 0 in RMD register)

$$\text{Deceleration time [sec]} = \frac{(\text{RFH} - \text{RFL}) \times (\text{RDR} + 1)}{5,000,000}$$

2. S-curve deceleration without linear section (Bit 10 (MSMD) = 1 in RMD register and RDS = 0)

$$\text{Deceleration time [sec]} = \frac{(\text{RFH} - \text{RFL}) \times (\text{RDR} + 1)}{2,500,000}$$

3. S-curve deceleration including linear section (Bit 10 (MSMD) = 1 in RMD register and RDS > 0)

$$\text{Deceleration time [sec]} = \frac{(\text{RFH} - \text{RFL} + 2 \times \text{RDS}) \times (\text{RDR} + 1)}{5,000,000}$$

When the deceleration starting point is set to auto (bit 12 (MSDP) = 0 in RMD register):

Specify RDR to give “(deceleration time) ≤ (acceleration time x 2).”

If RDR has been specified to give “(deceleration time) > (acceleration time x 2)”, the operation may be decelerated and stopped before reaching the base speed. In such a case, set the deceleration starting point to the manual setting (bit12 (MSDP) = 1 in RMD register).

(6) RMG: Speed Multiplier Setting register (11 bits)

This register sets the speed multiplier.

Set the relationship between the speed and the set values of RFL, RFH, and RFA in the range of 2 to 2,047 (07FFh).

The higher multiplier gives the speed configurable in rougher pitch.

The relationship between the set value and the speed multiplier is shown by the following formula:

$$\text{Speed multiplier [x]} = \frac{200}{(\text{RMG} + 1)}$$

<Examples of speed multiplier settings>

Set value	Multiplier	Speed range (pps)	Set value	Multiplier	Speed range (pps)
1999 (7CFh)	0.1	0.1 to 10,000.0	39 (027h)	5	5 to 500,000
999 (3E7h)	0.2	0.2 to 20,000.0	19 (013h)	10	10 to 1,000,000
399 (18Fh)	0.5	0.5 to 50,000.0	9 (009h)	20	20 to 2,000,000
199 (0C7h)	1	1 to 100,000	3 (003h)	50	50 to 5,000,000
99 (063h)	2	2 to 200,000	2 (002h)	66.6	66.6... to 6,666,666.6...

Table 8. 3-13 Examples of speed multiplier settings

(7) RDP: Deceleration Starting Point register (24 bits)

This register is used to set a starting point of acceleration/deceleration as well as positioning operations. The value set in RDP is used in different ways depending on the deceleration starting-point setting method selected in MSDP (bit 12) in the RMD register.

(l) For the manual setting (bit12 (MSDP) = 1 in the RMD register)

Specify the pulse count for the deceleration starting point in the range of 0 to 16,777,215 (00FFFFFFh).

Deceleration will start when: (The value of the positioning counter) \leq (RDP set value).

If the set value for the deceleration starting point is smaller than the optimum value, the speed at the end of the deceleration will be faster than the FL speed.

Conversely, if it is larger than the optimum value, the movement will be the FL constant-speed operation after the end of the deceleration.

The optimum value for a deceleration starting point is shown by the following formulas:

(1) Linear deceleration (bit10 (MSMD) = 0 in the RMD register)

$$\text{Optimum value of deceleration starting point [pulses]} = \frac{(\text{RFH}^2 - \text{RFL}^2) \times (\text{RDR} + 1)}{(\text{RMG} + 1) \times 50,000}$$

However, if a triangular drive is performed without changing the value in the RFH register while the FH correction function is disabled (bit17 (MADJ) = 1 in the RMD register), the optimum value is derived from the formula below.

(For idling control, subtract the idling pulse count from the set value in the RMV register, and then assign the result value to RMV shown below. The idling pulse count to be used here is "1 to 6" when IDL of RENV2 is "0 to 7.")

$$\text{Optimum value of deceleration starting point [pulses]} = \frac{\text{RMV} \times (\text{RDR} + 1)}{\text{RUR} + \text{RDR} + 2}$$

(2) S-curve deceleration without linear section (Bit 10 (MSMD) of RMD register = 0, and RDS register = 0)

$$\text{Optimum value of deceleration starting point [pulses]} = \frac{(\text{RFH}^2 - \text{RFL}^2) \times (\text{RDR} + 1) \times 2}{(\text{RMG} + 1) \times 50,000}$$

(3) S-curve deceleration including linear section (Bit 10 (MSMD) of RMD register > 0, and RDS register > 0)

$$\text{Optimum value of deceleration starting point [pulses]} = \frac{(\text{RFH} + \text{RFL}) \times (\text{RFH} - \text{RFL} + 2 \times \text{RDS}) \times (\text{RDR} + 1)}{(\text{RMG} + 1) \times 50,000}$$

(8) RUS: Acceleration S-Curve register (16 bits)

This register specifies the S-curve section in S-curve acceleration.

Set the S-curve section in S-curve acceleration in the range of 1 to 50,000 (C350h).

A value between 50,000 and 65,535 (C350h to FFFFh) is always handled as 50,000.

The S-curve acceleration section, S_{SU} , will be the value calculated with the RMG (Speed Multiplier Setting register) value.

$$S_{SU}[\text{pps}] = \text{RUS} \times \frac{200}{(\text{RMG} + 1)}$$

This means that S-curve acceleration operations are given during the periods from "the base speed" to "the base speed + S_{SU} " and from "the operation speed - S_{SU} " to "the operation speed." And a linear acceleration operation is given in the in-between section. However, when "0" is set, the internal arithmetic will apply "(RFH-RFL)/2" as a substitute so as to give an S-curve acceleration operation without a linear acceleration section.

When the minimum value "1" is set, the result will be almost the same as a linear acceleration operation.

When a value larger than "(RFH-RFL)/2" is set, the operation will not reach the maximum acceleration, resulting in acceleration time different from the one by the formula. Therefore, use a value less than "(RFH-RFL)/2."

(9) RDS: Deceleration S-Curve register (16 bits)

This register specifies the S-curve section in S-curve deceleration.

Set the S-curve section in S-curve deceleration in the range of 1 to 50,000 (C350h).

The S-curve deceleration section, S_{SD} , will be the value calculated with the RMG (Speed Multiplier Setting register) value.

$$S_{SD}[\text{pps}] = \text{RDS} \times \frac{200}{(\text{RMG} + 1)}$$

This means that S-curve deceleration operations are given during the periods from “the operation speed” to “the operation speed - S_{SD} ” and from “the base speed + S_{SD} ” to “the operation speed.” And a linear deceleration operation is given in the in-between section.

However, when “0” is set, the internal arithmetic will apply “(RFH-RFL)/2” as a substitute so as to give an S-curve deceleration operation without a linear deceleration section.

When the minimum value “1” is set, the result will be almost the same as a linear deceleration operation.

When a value larger than “(RFH-RFL)/2” is set, the operation will not reach the maximum deceleration, resulting in deceleration time different from the one by the formula. Therefore, use a value less than “(RFH-RFL)/2.”

(10) RFA: Auxiliary Speed register (17 bits)

This register sets the reverse constant speed in origin-returning operation or the constant speed in backlash compensation.

RFA is used as the reverse constant speed in origin-returning operation.

Or it is used to set the travel-distance compensation speed (FA speed) in backlash, in the range of 1 to 100,000 (186A0h).

A value between 100,000 and 131,071(186A0h and 1FFFFh) is always handled as 100,000.

The actual operation speed will be the value calculated with the RMG (Speed Multiplier Setting register) value.

$$\text{Auxiliary speed} [\text{pps}] = \text{RFA} \times \frac{200}{(\text{RMG} + 1)}$$

8.4 Status processing

Operation completion can be recognized by reading statuses through polling.

Polling allows reading of the motion main status (MMSTS), event status (RIST), and error status (REST) to detect events including operation completion and errors. Then, the next operation will be determined accordingly.

In such an operation as positioning, the operation completion is detected through those events in each motion device.

If the operation completion or an error is detected in MMSTS, RIST or REST is read to finish the process.

The RIST setting is changed upon the occurrence of an event report factor set in RIRQ. Normally RIRQ has been preset in the initial setting.

(1) Motion Main Status (MMSTS)

First, set bit 0 of RIRQ to "1" in the initial setting. (reporting normal end)

Statuses are always monitored through the MMSTSs of the target motion devices.

The MMSTSs are polled after the issuance of a start command.

Bit 1 is monitored, and changed from "0" to "1" upon motion completion.

At this point, bit 3 is turned to "1" for normal completion, or bit 2 is turned to "1" for abnormal completion.

If bit 2 is "1", the error status is read and the interrupt reset command is issued.

If bit 3 is "1", the event status is read and the interrupt reset command is issued.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	GRP2	GRP1	GRP0	0	0	0	SBSY	0	0	0	0	SEVT	SERR	SEND	SINT

Figure 8. 4-4 Bit configuration of Motion Main Status (MMSTS)

bit	Abbreviation	Function	Remarks
0	SINT	Turns "1" when any of MMSTS bit 1, 2, 3 is "1."	
1	SEND	Turns "1" upon operation stop. SEND is reset when started, where bit 28 is "1" in RENV1. Or, SEND is turned to "1" by the Interrupt Reset command (0008h).	
2	SERR	Turns "1" upon the occurrence of stop by an error, a failure in position override, or an abnormal encoder signal. SERR turns "0" when REST is read	
3	SEVT	Turns "1" upon the occurrence of the event set by RIRQ. SEVT turns "0" when RIST is read.	
7 to 4	(Undefined)	Always "0"	
8	SBSY	Turns "1" upon the start of pulse output, and "0" upon operation stop. (=BSY)	
11 to 9	(Undefined)	Always "0"	
14 to 12	GRP2 to GRP0	Group setting statuses Independent group: 000: 000, Group 1: 001, Group 2: 010 Group 3: 011, Group 4: 100, Group 5: 101 Group 6: 110, Group 7: 111	
15	(Undefined)	Always "0"	

Table 8. 4-1 Contents of Motion Main Status (MMSTS)

* The change of bit 3 means movement completion if bit 0 is set to 1 (normal end) in RIRQ (event factor setting register).

(2) Event Status (RIST)

RIST must be read every time "bit 3 (SEVT) = 1" is detected in MMSTS.

Preset bit 0 of RIRQ (event factor setting register) of the Event Status to "1" (normal end). (This is for basic operation.)

RIST is automatically reset to "0" after read, and bit 3 of MMSTS is also turned to "0."

Note. MMSTS is updated by cyclic communication. So there is a delay, as long as one cyclic communication period at the maximum, in reflecting the G9003 setting in MMSTS.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	ISNP	ISNA	ISSA	ISSD	ISOL	ISLT	ISCL	ISC3	ISC2	ISC1	ISDE	ISDS	ISUE	ISUS	ISEN
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8. 4-2 RIST: Bit configuration of the Event Status register

b0 (ISEN): "1" = Normal end

(3) Error Status (REST)

REST must be read every time "bit 4 (SERR) = 1" is detected in MSTs.

The corresponding bit is turned to "1" upon the occurrence of an error. The Error Status cannot be masked.

REST is automatically reset to "0" after read, and bit 2 of MMSTS is also turned to "0."

Note. MMSTS is updated by cyclic communication. So there is a delay, as long as one cyclic communication period at the maximum, in reflecting the G9003 setting in MMSTS.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	ESPE	ESEE	ESOR	0	ESNT	ESPO	ESSD	ESEM	ESSP	ESAL	ESML	ESPL	ESC3	ESC2	ESC1
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8. 4-3 REST: Bit configuration of the Error Status register

bit	Abbreviation	Contents	
0	ESC1	When stopped due to Comparator-1 condition matching (+SLS)	
1	ESC2	When stopped due to Comparator-2 condition matching (-SLS)	
2	ESC3	When stopped due to Comparator-3 condition matching	
3	ESPL	When stopped due to +ELS input ON	
4	ESML	When stopped due to -ELS input ON	
5	ESAL	When stopped due to SVALM input ON	
6	ESSP	(Reserved)	
7	ESEM	(Reserved)	
8	ESSD	When deceleration stop occurs due to DLS input ON	
9	ESPO	When the overflow of the manual-pulsar buffer counter occurs	
10	ESNT	When stopped due to a communication error	
11	Undefined	Undefined (Always "0.")	
12	ESOR	When positioning override operation fails	
13	ESEE	When an encoder input error occurs	Note: Operation is not stopped.
14	ESPE	When a pulsar input error occurs	
31 to 15	Undefined	(Always "0.")	

Table 8. 4-2 REST: Contents of the Error Status register

<Status management example (when only bit 0 of RIRQ is set to 1 (reporting normal end))>

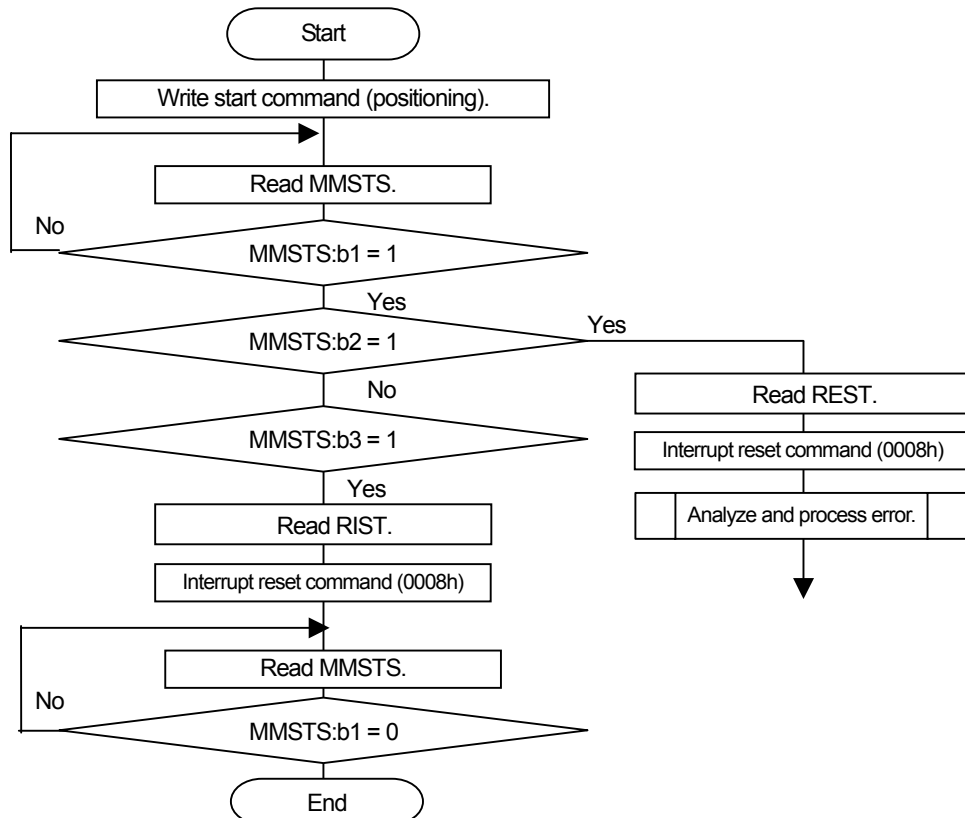


Figure 8. 4-4 Status management example

8.5 Applications

8.5.1 Simultaneous start using STA input

STA input allows a simultaneous start of motion modules in the slave.

Preset MSY (bit 14) of the RMD (operation mode) to “1” (waiting for STA input) for the motion modules to be started simultaneously. Use the simultaneous start command to start an operation.

By specifying the RIRQ (event factor setting) register, a simultaneous start (STA input ON) can cause the event. Reading the RIST register then allows you to check the event factor.

Also, reading the RSTS (expansion status) register allows you to monitor the operation status (waiting for STA input) as well as the STA input status.

<How to perform simultaneous start>

After setting MSY (bit 14) of the RMD register to “1” for the motion modules to be started simultaneously, write a start command to cause the “waiting for STA input” status.

Then, write the simultaneous start command (0006h) to the target motion modules to start them simultaneously.

To cancel the “waiting for STA input” status, write the immediate stop command (0049h).

Set the “waiting for STA input” status. 1: Started by STA input	<Set MSY (bit 14) of RMD.>
Read the SAT signal. 0: STA signal OFF 1: STA signal ON	<SSTA (bit 11) of RSTS>
Read the operation status. 0001: Waiting for STA input	<CND (bits 3 to 0) of RSTS>
Set the event factor. 1: Event occurs upon STA input ON.	<Set IRSA (bit 12) of RIRQ.>
Read the event factor. 1: When STA signal is ON.	<ISSA (bit 12) of RIST>
Simultaneous start command	<CMSTA: operation command (0006h)>
Command for simultaneous start only within the own axis Performs a process equivalent to the one by STA signal input only for the own axis.	<SPSTA: operation command (002Ah)>

8.5.2 Group start

It is possible to perform a group start by presetting the group for motion modules connected to the same Line.

Set the group settings for the motion modules to be started as a group in advance. Also, set MSY (bit 14) to “1” (waiting for STA input) in RMD (operation mode). Then, issue a group start command to execute a group start.

By specifying the RIRQ (event factor setting) register, a group start can cause the event. Reading the RIST register then allows you to check the event factor.

Also, reading the RSTS (expansion status) register allows you to monitor the operation status (waiting for STA input).

<How to perform group start>

Set the group settings for the motion modules to be started as a group. Next, set MSY (bit 14) to “1” in the RMD register for the motion modules for group starting.

Following this, write a start command to cause the “waiting for STA input” status.

And then, issue a group start command (2x01h, where x = group number) to the center device connected to the group.

To cancel the “waiting for STA input” status, write the immediate stop command (0049h).

Group setting	<Set GRP2 to GRP0 (bits 6 to 4) of IOPOB.> 000: Group 7, 001: Group 6, 010: Group 5, 011: Group 4, 100: Group 3, 101: Group 2, 110: Group 1, 111: Independent
Group setting status	<Set GRP2 to GRP0 (bits 14 to 12) of IOPOB.> 000: Independent, 001: Group 1, 010: Group 2, 011: Group 3, 100: Group 4, 101: Group 5, 110: Group 6, 111: Group 7
“Waiting for STA input” setting	<Set MSY (bit 14) of RMD.> 01: Started by STA input (or by a group start command)
Read the operation status	<CND (bits 3 to 0) of RSTS> 0001: Waiting for STA input
Set the event factor.	<Set IRNA (bit 12) of RIRQ.> 1: Event occurs upon group start.
Read the event factor.	<ISNA (bit 12) of RIST> 1: When group start is performed.
Group start command	<CCMD> 2x01, where x = specified group (0: All groups, 1: Group 1, ..., 7: Group 7)

8.5.3 Comparator settings

A motion module is equipped with three 28-bit comparators. The values set in the RCMP 1 to 3 registers are compared with the counter value. And a comparison counter can be selected from CTR 1 to 3 for each comparator. There are various comparison methods and three kinds of processing to be used upon condition matching. Specify these comparator settings in the RENV4 (environment setting 4) register.

Using the comparator function allows:

- (1) External output of comparison results given by Comparator 3.
- (2) Immediate/deceleration operation stop.
- (3) Soft-limit function using Comparators 1 and 2.
- (4) Stepping motor's a loss of synchronism detection function using CTR3 (error) and a comparator.
- (5) Synchronous output using CTR3 (general-purpose) and Comparator 3.
- (6) Starting of other modules in the same slave using Comparator 3.

(1) Comparison data

Comparison data can be selected from the table below for each comparator.

Comparison data	Comparator 1		Comparator 2		Comparator 3	
		C1C1~0		C2C1~0		C3C1~0
CTR1 (Command position)	O	"00"	O	"00"	O	"00"
CTR2 (machine position)	O	"01"	O	"01"	O	"01"
CTR3 (general-purpose/error)	O	"10"	O	"10"	O	"10"
Mainly used for	+SLS		-SLS		loss of synchronism detection, synchronous output	

1. The symbol "O" indicates comparison possible.
- 2."+ SLS" and "-SLS" indicate soft-limit.
3. When you select CTR3 (set as Error) as a comparison counter, the absolute value of the count value is compared with the comparator data. (Absolute value range: 0 to 32,767)
To select comparison data, specify C1C1 to 0 (bits1,0), C2C1 to 0 (bits 9,8), and C3C1 to 0 (bit17,16) of RENV4.

(2) Comparison methods

The comparison method can be selected from the table below for each comparator.

Comparison method	Comparator 1		Comparator 2		Comparator 3	
	C1S2~0		C2S2~0		C3S3~0	
Comparator = Comparison counter (Any count direction)	O	"001"	O	"001"	O	"0001"
Comparator = Comparison counter (Only while count increased)	O	"010"	O	"010"	O	"0010"
Comparator = Comparison counter (Only while count decreased)	O	"011"	O	"011"	O	"0011"
Comparator > Comparison counter	O	"100"	O	"100"	O	"0100"
Comparator < Comparison counter	O	"101"	O	"101"	O	"0101"
Used as soft-limit	O	"110"	O	"110"		
Synchronous output (Any count direction)					O	"1000"
Synchronous output (Only while count increased)					O	"1001"
Synchronous output (Only while count decreased)					O	"1010"

- The symbol "O" indicates comparison possible, blanks impossible.
- It is not allowed to set "C3S3 to 0 = 0111" for Comparator 3. The other set values are always for comparison-condition mismatch.
- When using "C3S3 to 0 = 1000 to 1010" for Comparator 3 (synchronous output), select the comparison counter CTR3 (general-purpose/error). The other counters cannot be used. Also, set the comparator setting value to a positive value.
- When the soft-limit is selected, Comparator 1 serves as the (+) limit value using the "Comparator < Comparison counter" comparison method. Also, Comparator 2 serves as the (-) limit value using the "Comparator > Comparison counter" comparison method. Use CTR1 (command position) as the comparison counter.
- To select the comparison method, specify C1S2 to 0 (bits 4 to 2), C2S2 to 0 (bits 12 to 10), and C3S3 to 0 (bits 21 to 18) of RENV4.

(3) Process performed upon comparator condition match

Process performed upon condition match can be selected from the table below.

Process performed upon condition match	Comparator 1		Comparator 2		Comparator 3	
	C1D1 to 0		C2D1 to 0		C3D1 to 0	
No process	"00"		"00"		"00"	
Immediate stop	"01"		"01"		"01"	
Deceleration stop	"10"		"10"		"10"	
Comparison-condition mismatch	"11"		"11"		"11"	

For selecting the process to be performed, specify C1D1 to 0 (bits 6, 5), C2D1 to 0 (bits 14, 13), and C3D1 to 0 (bits 23, 22) of RENV4.

Set the event factor. <Set IRC3 to 1 (bits 7 to 5) of RIRQ.> IRC1(bit5) = The event occurs upon Comparator 1 condition match. IRC2(bit6) = The event occurs upon Comparator 2 condition match. IRC3(bit7) = The event occurs upon Comparator 3 condition match.
Read the event factor. <ISC3 to 1 (bits 7 to 5) of RIST> IRC1(bit5) = When Comparator 1 condition matched. IRC2(bit6) = When Comparator 2 condition matched. IRC3(bit7) = When Comparator 3 condition matched.
Read the status of the comparator condition match. <SCP3 to 1 (bits 22 to 20) of RSTS> SCP1(bit20) = When Comparator 1 condition matched. SCP2(bit21) = When Comparator 2 condition matched. SCP3(bit22) = When Comparator 3 condition matched.
Read the error interrupt factor. <ESC3 to 1 (bits 2 to 0) of REST> ESC1(bit0) = When stopped by Comparator 1 condition match. (+SLS) ESC2(bit1) = When stopped by Comparator 2 condition match. (-SLS) ESC3(bit2) = When stopped by Comparator 3 condition match.

8.5.4 Soft-limit settings

The soft-limit function is available using Comparators 1 and 2.

Select CTR1 (command position) as the comparison counter for the Comparators.

Comparator 1 is used as the (+) limit value, and Comparator 2 as the (-) limit value in order to perform stop control based on the comparison results and operation directions.

The soft-limit operation allows:

- (1) Immediate stop of pulse output.
- (2) Deceleration stop of pulse output.

In using the soft-limit function, if deceleration stop is selected as the process performed upon comparator condition match (C1D, C2D), deceleration stop will be executed upon reaching the soft-limit during a high-speed start (command 0052h). If another process is selected for condition match or a constant-speed start is being performed, immediate stop will be the result.

Also, if the soft-limit is ON while a start command is written, it is impossible to start a movement in the soft-limit ON direction; but possible in the reverse direction.

[Setting example]

RENV4=00003838h : Use Comparator 1 as the (+) soft-limit, and Comparator 2 as the (-) soft-limit.

Select immediate stop as the process to be performed upon soft-limit ON.

RCMP1 = (+) limit value

RCMP2 = (-) limit value

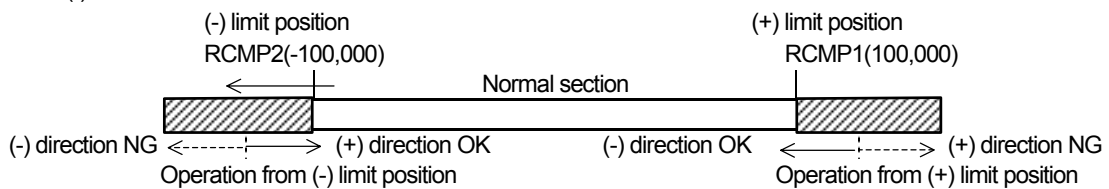


Figure 8. 5-1 Setting example of the soft-limit

Set the comparison method for Comparator 1.	<Set C1S2 to 0 (bits 4 to 2) of RENV4.>
001: RCMP1 data = Comparison counter	(Any count direction)
010: Same as above	(While count increased)
011: Same as above	(While count decreased)
100: RCMP1 data > Comparison counter	
101: RCMP1 data < Comparison counter	
110: Used as (+) soft-limit (RCMP1 < CTR1)	
Others: Always comparison-condition mismatch	
Process performed upon Comparator 1 condition match	<Set C1D1 to 0 (bits 6, 5) of RENV4.>
01: Immediate stop	
10: Deceleration stop	
Set the comparison method for Comparator 2.	<Set C2S2 to 0 (bits 12 to 10) in RENV4.>
001: RCMP2 data = Comparison counter	(Any count direction)
010: Same as above	(While count increased)
011: Same as above	(While count decreased)
100: RCMP2 data > Comparison counter	
101: RCMP2 data < Comparison counter	
110: Used as (-) soft-limit (RCMP2 > CTR1)	
Others: Always comparison-condition mismatch	
Process performed upon Comparator 2 condition match	<Set C2D1 to 0 (bits 14, 13) of RENV4.>
01: Immediate stop	
10: Deceleration stop	

Note: The bold-faced types indicate the settings used in the example in the figure above.

8.5.5 Synchronous output settings

It is possible to output signals to the CMP3 terminal of the machine interface connector at regular intervals using Comparator 3 and CTR3 (general-purpose/error counter) set for general purpose.

The synchronous output function can be used by setting “C3C1 to C3C0 = 10 (CTR3)” and “C3S3 to C3S0 = 1000 to 1010 (Synchronous output).”

The CTR3 count range is from 0 to the RCMP3 set value (max. 32,767). Decreasing the count from 0 leads to the RCMP3 set value, and increasing the count from the RCMP3 set value leads to 0.

CTR3 input can be specified in CI32 to CI30 of RENV3.

Set CTR3 (error) input. <Set CI32 to 30 (bits 12 to 10) of RENV3.> 000: Command pulse 001: ENC input 011: 1/4096 clock of the internal reference clock (CLK = 40 MHz) 100: Error counting by command pulse and ENC input
Selects the comparison counter for Comparator 3 <Set C3C1 to 0 (bits 17, 16) of RENV4.> 00: CTR1 (command position) 10: CTR3 (general-purpose/error) 01: CTR2 (machine position) 11: Always comparison-condition mismatch
Set the comparison method for Comparator 3. <Set C3S3 to 0 (bits 21 to 18) of RENV4.> 0001: RCMP3 data = Comparison counter (Any count direction) 0010: Same as above (While count increased) 0011: Same as above (While count decreased) 0100: RCMP3 data > Comparison counter 0101: RCMP3 data < Comparison counter 0111: Setting forbidden 1000: Used for synchronous signal output (Any count direction) 1001: Same as above (While count increased) 1010: Same as above (While count decreased) Others: Always comparison-condition mismatch

Note: The bold-faced types indicate the settings configurable in synchronous output.

[Output example]

Synchronous signals are output by output pulses regardless of the operation direction. The count range is from 0 to 4.

Set values: RENV3=00000000h, RENV4=00220000h, RCMP4 = 4

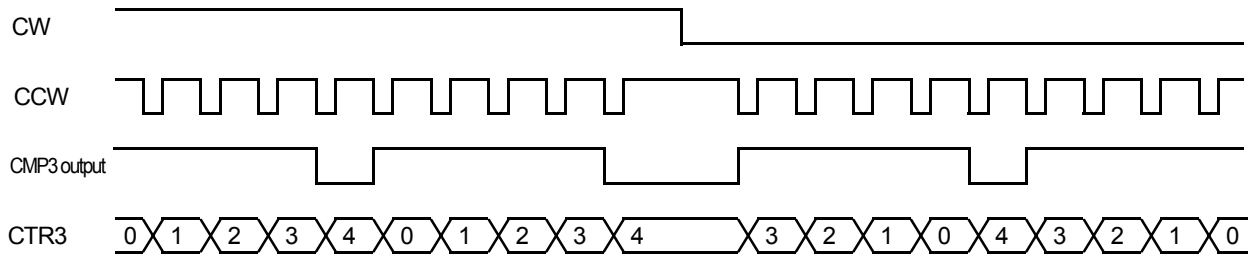


Figure 8. 5-2 Synchronous output

8.5.6 Starting of other motion modules in the same slave using Comparator 3

It is possible to start other motion modules in the same slave when the comparison condition is matched using Comparator 3 and CTR3 (general-purpose/error counter) set for general purpose.

[Setting the base motion module]

- Switch settings for the motion module
Encoder type selection Turn on SW1 to 4.

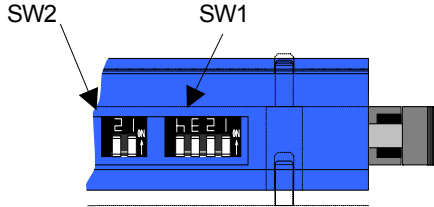


Figure 8. 5-3 Motion module top view

Set CTR3 (error) input.	<Set CI32 to 30 (bits 12 to 10) of RENV3.>
000: Command pulse	
001: ENC input	
011: 1/4096 clock of the internal reference clock (CLK = 40 MHz)	
100: Error counting by command pulse and ENC input	
Selects the comparison counter for Comparator 3 <Set C3C1 to 0 (bits 17, 16) of RENV4.>	
00: CTR1 (command position)	10: CTR3 (general-purpose/error)
01: CTR2 (machine position)	11: Always comparison-condition mismatch
Set the comparison method for Comparator 3.	<Set C3S3 to 0 (bits 21 to 18) of RENV4.>
0001: RCMP3 data = Comparison counter	(Any count direction)
0010: Same as above	(While count increased)
0011: Same as above	(While count decreased)
0100: RCMP3 data > Comparison counter	
0101: RCMP3 data < Comparison counter	
0111: Setting forbidden	
1000: Used for synchronous signal output	(Any count direction)
1001: Same as above	(While count increased)
1010: Same as above	(While count decreased)
Others: Always comparison-condition mismatch	

[Setting the motion modules to be started]

Set the "waiting for STA input" setting	<Set MSY (bit 14) of RMD.>
1: Started by STA input	

Note. The bold-faced types indicate the settings configurable for starting the other motion modules in the same slave upon comparator condition match.

<Steps to start the other motion modules in the same slave upon comparator condition match>

(It is assumed that the standard initialization settings, speed, travel distance, etc. are preset.)

- (1) Set the comparator condition for the "base motion module."
- (2) Set MSY (bit14) of the RMD register to "1" for the "motion modules to be started."
- (3) Write a start command to the "motion modules to be started" to turn them into the "waiting for STA" status.
- (4) Start the "base motion module."
- (5) Then, the "motion modules to be started" will be started up when the comparator condition is matched in the "base motion module."

To cancel the "waiting for STA input" status, write the immediate stop command (0049h).