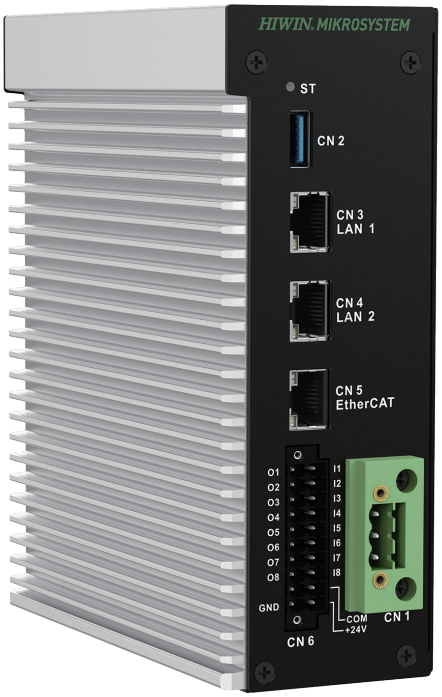
HIWIN® MIKROSYSTEM



HIMC Series

Modbus TCP User Guide

www.hiwinmikro.tw

MH02UE01-2510_V1.1

Revision History

The version of the guide is also indicated on the bottom of the front cover.

MH02UE01-2510_V1.1

Version

Release Date

Release Date	Version	Applicable Software Version	Revision Contents
Oct. 15th, 2025	1.1	iA Studio 3.3	1. Add the communication interface description of HIMC3's CN3 and CN4. 2. Update the representation of register addresses.
Feb. 28th, 2025	1.0	iA Studio 3.1	1. Update section 2.1 **Communication interface**. 2. Update section 3.2 **Function codes**. 3. Update section 3.4 **Data type**. 4. Update chapter 4 **Register Map**.
Jun. 30th, 2022	0.3	iA Studio 2.0	System date and time are added to the registers assigned for Controller Information.
Sep. 16th, 2020	0.2	iA Studio 1.3	Change unit system: m-rad-s → mm-deg-ms
Apr. 10th, 2018	0.1	iA Studio 1.0.2461.0	First edition.

Related Documents

Through related documents, users can quickly understand the positioning of this manual and the correlation between manuals and products. Go to HIWIN MIKROSYSTEM's official website → Download → Manual Overview for details (https://www.hiwinmikro.tw/Downloads/ManualOverview_EN.htm).

Table of Contents

1. Overview 1-1
   1.1 Introduction of HIMC Series Modbus TCP 1-2

2. Communication interface of HIMC series Modbus TCP 2-1
   2.1 Communication interface 2-2

3. Functions of HIMC series Modbus TCP 3-1
   3.1 Data storage 3-2
   3.2 Function codes 3-2
   3.3 Exception codes 3-3
   3.4 Data type 3-3

4. Register Map 4-1
   4.1 Coils (0X) 4-2
      4.1.1 Axis 4-3
      4.1.2 System Call 4-4
      4.1.3 HMPL Task 4-5
   4.2 Discrete Inputs (1X) 4-6
   4.3 Input Registers (3X) 4-7
      4.3.1 Axis 4-8
      4.3.2 Controller Information 4-10
      4.3.3 HMPL Task 4-11
   4.4 Holding Registers (4X) 4-12
      4.4.1 Axis 4-13
      4.4.2 GPIO 4-15
      4.4.3 Slave GPIO 4-16
      4.4.4 User Table 4-17
      4.4.5 User-defined Parameters 4-19
      4.4.6 Undefined Registers 4-19

1. Overview

1. Overview 1-1
1.1 Introduction of HIMC Series Modbus TCP 1-2

1.1 Introduction of HIMC Series Modbus TCP

HIWIN Motion Controller (HIMC[*1]) supports Modbus TCP communication protocol. Users are allowed to access HIMC via Modbus TCP by HMI (Human Machine Interface) or PC to read and write parameters of axis, system call, controller information, etc.

Note:

*1: It includes models named after HIMC series.

2. Communication interface of HIMC series Modbus TCP

2. Communication interface of HIMC series Modbus TCP 2-1
2.1 Communication interface 2-2

2.1 Communication interface

Connectors CN3 and CN4 are provided for communication with PC or HMI via Modbus TCP.

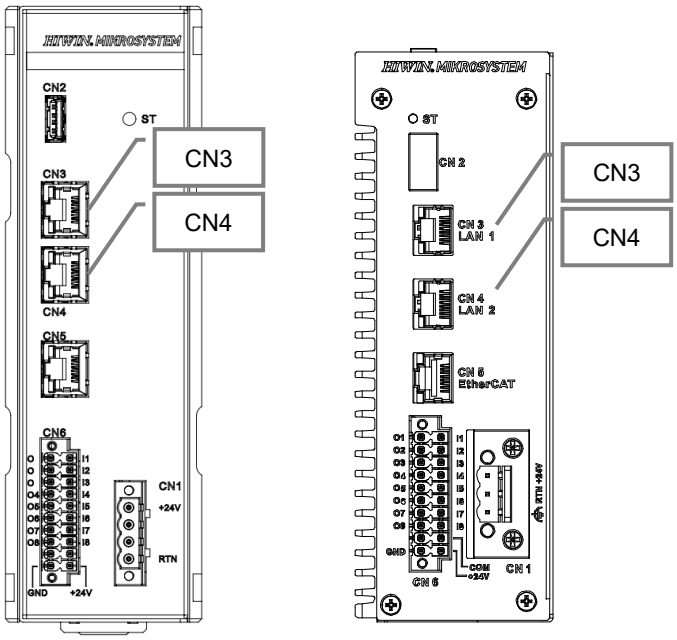


Figure 2.1.1

The default information of connectors CN3 and CN4 is as follows:

Table 2.1.1

Connector	CN3	CN4
IP Address	192.168.0.101 (Default)	169.254.188.20 (Fixed)
Port	502	

Note:

CN3's IP address can be set via iA Studio. Please refer to section 4.14 **IP Setting** in "iA Studio User Guide."

The simulator provided in HIMC can also be accessed via Modbus TCP. Information needed when accessing HIMC simulator is as follows:

Table 2.1.2

Simulator	
IP Address	127.0.0.1
Port	502

Note:

Before accessing HIMC simulator via Modbus TCP, please ensure the connection with HIMC simulator is established. For how to connect to HIMC simulator, please refer to section 2.1.3 **Connecting to the simulator** in “iA Studio User Guide.”

(This page is intentionally left blank.)

3. Functions of HIMC series Modbus TCP

3. Functions of HIMC series Modbus TCP 3-1
3.1 Data storage 3-2
3.2 Function codes 3-2
3.3 Exception codes 3-3
3.4 Data type 3-3

3.1 Data storage

Data in standard Modbus are stored in four different tables. Register assignment of HIMC is as follows:

Table 3.1.1

Table Name	Code	Coils/Register Address	Data Size	Attribute
Coils	0X	0_d ~ 65535_d (0000_h ~ $FFFF_h$)	1 bit	Read / Write
Discrete Inputs	1X		1 bit	Read-only
Input Registers	3X		16 bits	Read-only
Holding Registers	4X		16 bits	Read / Write

3.2 Function codes

Supported function codes in HIMC are as follows:

Table 3.2.1

Function Code	Description
1_d (01_h)	Read **Coils**.
2_d (02_h)	Read **Discrete Inputs**.
3_d (03_h)	Read **Holding Registers**.
4_d (04_h)	Read **Input Registers**.
5_d (05_h)	Write single **Coils**.
6_d (06_h)	Write single **Holding Registers**.
15_d ($0F_h$)	Write multiple **Coils**.
16_d (10_h)	Write multiple **Holding Registers**.
23_d (17_h)	Write and read multiple **Holding Registers**.

3.3 Exception codes

When a request is received but cannot be processed, an exception response with an exception code will be sent from HIMC. Supported exception codes in HIMC are as follows:

Table 3.3.1

Exception Code	Definition	Description
1_d (01_h)	Illegal function code	An unsupported function code is requested. For example, function code 20_d (14_h) is specified in a request.
2_d (02_h)	Illegal data address	The requested register is not allowable. For example, for a controller with 100 registers, the controller will reply exception code 02 if a request with starting address 96 and register length 5 is received.
3_d (03_h)	Illegal data value	The value specified in a request is not allowable. For example, the initial address of a parameter is 0 and it occupies two registers. If a request with starting address 1, or starting address 0 and register length 1 is received to read or write the parameter, the controller will reply exception code 03.

3.4 Data type

Parameters in HIMC are of different data types. Data types in HIMC are as follows:

Table 3.4.1

Data Type	Data Size	Range
int8_t	8 bits	-128 ~ 127
uint8_t	8 bits	0 ~ 255
int16_t	16 bits	-32,768 ~ 32,767
uint16_t	16 bits	0 ~ 65,535
int32	32 bits	-2,147,483,648 ~ 2,147,483,647
uint32	32 bits	0 ~ 4,294,967,295
int64	64 bits	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
uint64	64 bits	0 ~ 18,446,744,073,709,551,615
float	32 bits	3.4E +/- 38 (7-digit)

Parameter data are stored in respective registers according to data types. Users need to follow the instructions described below to read parameter data.

- int8_t, uint8_t, int16_t and uint16_t

  Parameters of data types int8_t, uint8_t, int16_t and uint16_t are used for digital inputs or outputs and controller status. The parameter data are stored as follows:

Number System	Value	Register N
dec	17_d	17_d
hex	0011_h	0011_h

- int32_t and uint32_t

  Parameters of data types int32_t and uint32_t are used for digital inputs or outputs and controller status. The parameter data are stored as follows:

Number System	Value	Register N (Start)	Register N+1 (End)
dec	2097169_d	17_d	32_d
hex	00200011_h	0011_h	0020_h

- float

  If parameter data types are float, the parameter data are stored as follows:

Number System	Value	Register N (Start)	Register N+1 (End)
float	0.85_f	Invalid	Invalid
hex	$3F59999A_h$	$999A_h$	$3F59_h$

- double

  If parameter data types are double, the parameter data are stored as follows:

Number System	Value	Register N (Start)	Register N+1	Register N+2	Register N+3 (End)
double	0.85_f	Invalid	Invalid	Invalid	Invalid
hex	$3FEB333333333333_h$	3333_h	3333_h	3333_h	$3FEB_h$

4. Register Map

4. Register Map 4-1
4.1 Coils (0X) 4-2
4.1.1 Axis 4-3
4.1.2 System Call 4-4
4.1.3 HMPL Task 4-5
4.2 Discrete Inputs (1X) 4-6
4.3 Input Registers (3X) 4-7
4.3.1 Axis 4-8
4.3.2 Controller Information 4-10
4.3.3 HMPL Task 4-11
4.4 Holding Registers (4X) 4-12
4.4.1 Axis 4-13
4.4.2 GPIO 4-15
4.4.3 Slave GPIO 4-16
4.4.4 User Table 4-17
4.4.5 User-defined Parameters 4-19
4.4.6 Undefined Registers 4-19

4.1 Coils (0X)

Registers for **Coils** are provided for HIMC to execute the axis command functions of each axis, system call, and HMPL Task. The default categories of **Coils** are defined in Table 4.1.1. Up to 16 axes of motion commands are supported.

Table 4.1.1

Category	Description
Axis	Execute the axis command functions of each axis such as enable the axis and clear errors.
System Call	Execute system call such as emergency stop, jog, and relative movement.
HMPL Task	Run or stop HMPL Task.

4.1.1 Axis

By accessing the registers assigned for **Axis** parameters, users can perform point-to-point motion, enable the axis, clear errors, and set the position as zero.

The registers assigned for **Axis** are defined as follows:

Table 4.1.1.1

Register Address*1	Parameter	Data Type	Attribute	Description
0_d (0000_h)	Select axis	bool	Read / Write	Set axis N as the selected axis or display if axis N is selected. Bit 0: 0: Cancel the selected axis. Bit 0: 1: Select axis. Note: When using **System Call** to perform motion control, motion control will only be performed on the selected axis.
1_d (0001_h)	P2P repeat	bool	Read / Write	Set repeated point-to-point motion on axis N. Or display if point-to-point motion is repeatedly performed on axis N. Bit 0: 0: Do not repeat point-to-point motion. Bit 0: 1: Repeat point-to-point motion. Note: Use **System Call** to perform point-to-point motion.
2_d (0002_h)	Axis enable/disable	bool	Read / Write	Enable or disable axis N. Or display if axis N is enabled or disabled. Bit 0: 0: Axis N is disabled. Bit 0: 1: Axis N is enabled.
3_d (0003_h)	Clear error stop	bool	Read / Write	Clear the fault status of axis N. Bit 0: 1: Clear fault status.
4_d (0004_h)	Set zero	bool	Read / Write	Set the current position of axis N as zero position. Bit 0: 1: Set current position as zero position.

Note:

*1: The register address of the parameters of each axis: Register address + 16 * N (N_{max} = 15)

4.1.2 System Call

By accessing the registers assigned for **System Call** parameters, users can perform motion control on the axis such as emergency stop, jog, and relative movement.

The registers assigned for **System Call** are as follows:

Table 4.1.2.1

Register Address	Parameter	Data Type	Attribute	Description*1
2304_d (0900_h)	Emergency stop	bool	Read / Write	Perform emergency stop on all axes and disable all axes. Bit 0: 1: Emergency stop.
2305_d (0901_h)	Stop all	bool	Read / Write	Stop motions on all axes. Bit 0: 1: Stop motions on all axes.
2306_d (0902_h)	Stop	bool	Read / Write	Stop motion on the selected axis. Bit 0: 1: Stop motion on the selected axis.
2307_d (0903_h)	Jog+	bool	Read / Write	Perform jog in positive direction on the selected axis. Bit 0: 1: Jog+.
2308_d (0904_h)	Jog-	bool	Read / Write	Perform jog in negative direction on the selected axis. Bit 0: 1: Jog-.
2309_d (0905_h)	Move relative	bool	Read / Write	Perform relative movement on the selected axis. Bit 0: 1: Perform relative movement.
2310_d (0906_h)	P2P P1	bool	Read / Write	Move the selected axis to position 1. Bit 0: 1: Move the selected axis to position 1.
2311_d (0907_h)	P2P P2	bool	Read / Write	Move the selected axis to position 2. Bit 0: 1: Move the selected axis to position 2.
2312_d (0908_h)	Home	bool	Read / Write	Perform homing on the selected axes. Bit 0: 1: Perform homing.
2313_d (0909_h)	Save User Table	bool	Read / Write	Save User Table. Bit 0: 1: Save User Table.

Note:

*1: To perform motion control on the selected axis, please select the axis and set the related motion parameters first.

4.1.3 HMPL Task

By accessing the registers assigned for **HMPL Task** parameters, users can run or stop **HMPL Task**. Up to 64 HMPL Tasks (0~63) are supported.

The registers assigned for **HMPL Task** are as follows:

Table 4.1.3.1

Register Address*1	Parameter	Data Type	Attribute	Description
2336_d (0920_h)	Task start/stop	bool	Read / Write	Run or stop Task. Bit 0: 0: Stop Task. Bit 0: 1: Run Task.

Note:

*1: The register address of the parameters of each Task: Register address + N (N_{max} = 63)

4.2 Discrete Inputs (1X)

Registers for **Discrete Inputs** are not defined. Users are allowed to use these registers freely.

4.3 Input Registers (3X)

Registers for **Input Registers** are provided for HIMC to monitor the status of each axis, the information of controller, and the status of HMPL Task. The default categories of **Input Registers** are defined in Table 4.3.1. Up to 16 axes of motion commands are supported.

Table 4.3.1

Category	Description
Axis	Monitor the status of each axis such as motion status, position feedback, and error code.
Controller Information	Monitor the information of controller such as controller status, system date and time.
HMPL Task	Monitor the status of HMPL Task such as running, debug mode, and paused.

4.3.1 Axis

By accessing the registers assigned for **Axis** parameters, users can monitor axis' motion status, position feedback, error code, etc.

The registers assigned for **Axis** are defined as follows:

Table 4.3.1.1

Register Address*1	Parameter	Data Type	Attribute	Description	Unit*2
0_d (0000_h) 1_d (0001_h)	Motion status	uint32_t	Read-only	Displays the motion status of axis N. Bit 0: The axis is enabled. Bit 1: The axis is moving. Bit 2: The axis is in-position. Bit 3: The axis is synchronized. Bit 4: The axis is grouped.	-
2_d (0002_h) 3_d (0003_h)	Fault status	uint32_t	Read-only	Displays the fault status of axis N. Bit 0: Error stop. Bit 1: Drive fault. Bit 2: Position error is too large. Bit 3: Right hardware limit is reached. Bit 4: Left hardware limit is reached. Bit 5: Right software limit is reached. Bit 6: Left software limit is reached.	-
4_d (0004_h) 5_d (0005_h)	Position feedback	float	Read-only	Display the position feedback of axis N.	mm or deg
6_d (0006_h) 7_d (0007_h)	Velocity feedback	float	Read-only	Display the velocity feedback of axis N.	mm/s or deg/s
8_d (0008_h) 9_d (0009_h)	Acceleration feedback	float	Read-only	Display the acceleration feedback of axis N.	mm/s^2 or deg/s^2
10_d ($000A_h$) 11_d ($000B_h$)	Jerk	float	Read-only	Display the jerk of axis N.	mm/s^3 or deg/s^3
12_d ($000C_h$) 13_d ($000D_h$)	CoE error code	int32_t	Read-only	Display the error code of CoE drive of axis N.	-
14_d ($000E_h$) 15_d ($000F_h$)	Axis error code	int32_t	Read-only	Display the last error code of axis N.	-

Note:

*1: The register address of the parameters of each axis: Register address + 30 * N (N_{max} = 15)

*2: The unit, linear unit (mm) or rotary unit (deg), is decided according to the setting in iA Studio.

4.3.2 Controller Information

By accessing the registers assigned for **Controller Information** parameters, users can monitor the information of controller such as controller status, system date and time.

The registers assigned for **Controller Information** are defined as follows:

Table 4.3.2.1

<table>
<tr><th>Register Address</th><th>Parameter</th><th>Data Type</th><th>Attribute</th><th>Description</th></tr>
<tr><td>4096d
(1000h)</td><td rowspan="2">System status*1</td><td rowspan="2">uint32_t</td><td rowspan="2">Read-only</td><td rowspan="2">Display system status. The status of each value is as follows:<table><tr><td>0</td><td>Initializing.</td></tr><tr><td>1</td><td>Busy.</td></tr><tr><td>2</td><td>Synchronous:
Controller is ready to perform motion control.</td></tr><tr><td>3</td><td>Asynchronous:
Controller is not ready to perform motion control.</td></tr><tr><td>4</td><td>An error occurs in the controller.</td></tr></table></td></tr>
<tr><td>4097d
(1001h)</td></tr>
<tr><td>4098d
(1002h)</td><td rowspan="2">Error code*2</td><td rowspan="2">uint32_t</td><td rowspan="2">Read-only</td><td rowspan="2">Display the latest HIMC error code.</td></tr>
<tr><td>4099d
(1003h)</td></tr>
<tr><td>4100d
(1004h)</td><td rowspan="3">System date</td><td rowspan="3">uint16_t</td><td rowspan="3">Read-only</td><td>Display system date (year).</td></tr>
<tr><td>4101d
(1005h)</td><td>Display system date (month).</td></tr>
<tr><td>4102d
(1006h)</td><td>Display system date (day).</td></tr>
<tr><td>4103d
(1007h)</td><td rowspan="3">System time</td><td rowspan="3">uint16_t</td><td rowspan="3">Read-only</td><td>Display system time (hour).</td></tr>
<tr><td>4104d
(1008h)</td><td>Display system time (minute).</td></tr>
<tr><td>4105d
(1009h)</td><td>Display system time (second).</td></tr>
</table>

Note:

*1: For related information of system status, please refer to section 1.5 **Main screen** in “iA Studio User Guide.” For how the LED indicator reacts to each status, please refer to section 2.4 **LED indicator** in “HIMC Installation Guide” or section 2.6 **LED indicator** in “HIMC3 Installation Guide.”

*2: The error code is stored in decimal format. Please convert to hexadecimal format to search for its description in chapter 5 **Appendix** in “iA Studio User Guide.”

4.3.3 HMPL Task

By accessing the registers assigned for **HMPL Task** parameters, users can monitor the status of **HMPL Task**. Up to 64 HMPL Tasks (0~63) are supported.

The registers assigned for **HMPL Task** are as follows:

Table 4.3.3.1

<table>
<tr><th>Register Address*1</th><th>Parameter</th><th>Data Type</th><th>Attribute</th><th>Description</th></tr>
<tr><td>4112_d (1010_h)</td><td rowspan="2">Task status</td><td rowspan="2">Int32_t</td><td rowspan="2">Read-only</td><td rowspan="2">Display the status of Task.
<table>
<tr><td>Bit 0</td><td>Task is imported to RAM.</td></tr>
<tr><td>Bit 1</td><td>Task is running.</td></tr>
<tr><td>Bit 2</td><td>Task is running in debug mode.</td></tr>
<tr><td>Bit 3</td><td>Task is paused.</td></tr>
<tr><td>Bit 4</td><td>Error occurs when running Task.</td></tr>
<tr><td>Bit 5</td><td>Task has been modified.</td></tr>
<tr><td>Bit 6</td><td>Error occurs when importing Task.</td></tr>
</table></td></tr>
<tr><td>4113_d (1011_h)</td></tr>
</table>

Note:

*1: The register address of the parameters of each Task: Register address + N * 2 (N_{max} = 63)

4.4 Holding Registers (4X)

Registers for **Holding Registers** are provided for HIMC to set motion parameters of each axis, access controller's and slave's IO, access controller's User Table, and set user-defined parameters. The default categories of **Holding Registers** are defined in Table 4.4.1. Up to 16 axes of motion commands, IO of 32 slaves, 128 indexes in User Table are supported. Memory section $4X28672_d$ ~ $4X40959_d$ are reserved for users to freely assign the desired parameters.

Table 4.4.1

Category	Description
Axis	Monitor status and set parameters of each axis.
GPIO	Control HIMC's general-purpose inputs and outputs (GPIO).
Slave GPIO	Control slave's general-purpose inputs and outputs (GPIO).
User Table	Access HIMC's User Table.
User-defined Parameters	Some registers are reserved for user-defined parameters. Users can assign desired parameters to registers.
Undefined Registers	Some registers are not defined yet. Users are allowed to use these registers freely.

Note:

For assigning desired parameters to registers, please refer to section 4.12 **Modbus Manager** in "iA Studio User Guide."

4.4.1 Axis

By accessing the registers assigned for **Axis** parameters, users can monitor or set axis' motion parameters, position feedback, error code, etc.

The registers assigned for **Axis** are defined as follows:

Table 4.4.1.1

Register Address*1	Parameter	Data Type	Attribute	Description	Unit*2
20480_d (5000_h)	Max. profile velocity	float	Read / Write	Set or display the max. velocity of axis N.	mm/s or deg/s
20481_d (5001_h)					
20482_d (5002_h)	Max. profile acceleration	float	Read / Write	Set or display the max. acceleration of axis N.	mm/s^2 or deg/s^2
20483_d (5003_h)					
20484_d (5004_h)	Max. profile deceleration	float	Read / Write	Set or display the max. deceleration of axis N.	mm/s^2 or deg/s^2
20485_d (5005_h)					
20486_d (5006_h)	Smooth time	float	Read / Write	Set or display the smooth time of axis N.	ms
20487_d (5007_h)					
20488_d (5008_h)	P2P dwell time	float	Read / Write	Set or display the dwell time of axis N.	ms
20489_d (5009_h)					
20490_d ($500A_h$)	P2P position 1	float	Read / Write	Set or display position 1 of axis N. Note: Position 1 of point-to-point motion.	mm or deg
20491_d ($500B_h$)					
20492_d ($500C_h$)	P2P position 2	float	Read / Write	Set or display position 2 of axis N. Note: Position 2 of point-to-point motion.	mm or deg
20493_d ($500D_h$)					
20494_d ($500E_h$)	Relative distance	float	Read / Write	Set or display the relative distance of axis N. Note: Relative distance is the moving distance when performing relative movement. Use **System Call** to perform relative movement.	mm or deg
20495_d ($500F_h$)					
20496_d (5010_h)	Home method	int16_t	Read / Write	Set or display the homing method of homing procedure of axis N.	-
20497_d (5011_h)	Home fast speed	float	Read / Write	Set or display the fast homing velocity of homing procedure of axis N.	mm/s or deg/s
20498_d (5012_h)					
20499_d (5013_h)	Home slow speed	float	Read / Write	Set or display the slow homing velocity of homing procedure of axis N.	mm/s or deg/s
20500_d (5014_h)					

Register Address*1	Parameter	Data Type	Attribute	Description	Unit*2
20501_d (5015_h)	Home acceleration	float	Read / Write	Set or display the homing acceleration of homing procedure of axis N.	mm/s^2 or deg/s^2
20502_d (5016_h)					
20503_d (5017_h)	Home offset	float	Read / Write	Set or display the home offset of homing procedure of axis N.	mm or deg
20504_d (5018_h)					
20505_d (5019_h)	Home timeout	int32_t	Read / Write	Set or display the timeout of homing procedure of axis N.	ms
20506_d ($501A_h$)					

Note:

*1: The register address of the parameters of each axis: Register address + 30 * N (N_{max} = 15)

*2: The unit, linear unit (mm) or rotary unit (deg), is decided according to the setting in iA Studio.

4.4.2 GPIO

By accessing the registers assigned for **GPIO** parameters, users can monitor or set HIMC's general-purpose inputs and general-purpose outputs. 8 general-purpose inputs (GPI1~GPI8) and 8 general-purpose outputs (GPO1~GPO8) are provided.

The registers assigned for **GPIO** are as follows:

Table 4.4.2.1

Register Address	Parameter	Data Type	Attribute	Description
24576_d (6000_h)	GPI	int32_t	Read-only	Display the status of GPI (1~8). Bit 0: 0: GPI1 is OFF. Bit 0: 1: GPI1 is ON. ⋮ ⋮ Bit 7: 0: GPI8 is OFF. Bit 7: 1: GPI8 is ON.
24577_d (6001_h)				
24578_d (6002_h)	GPO	int32_t	Read / Write	Set GPO (1~8) or display the status of GPO (1~8). Bit 0: 0: GPO1 is OFF. Bit 0: 1: GPO1 is ON. ⋮ ⋮ Bit 7: 0: GPO8 is OFF. Bit 7: 1: GPO8 is ON.
24579_d (6003_h)				

4.4.3 Slave GPIO

By accessing the registers assigned for **Slave GPIO** parameters, users can monitor or set slave's general-purpose inputs and general-purpose outputs. Up to 256 general-purpose inputs (GPI1~GPI256) and 256 general-purpose outputs (GPO1~GPO256) are provided.

The registers assigned for **Slave GPIO** are as follows:

Table 4.4.3.1

Register Address	Parameter	Data Type	Attribute	Description
24592_d (6010_h)	Select **Slave**	uint16_t	Read / Write	Select **Slave** of each **Slave GPIO**. The value of **Slave** is 0~31.

Note:

Before using **Slave GPIO**, users need to set the parameter above to obtain the GPIO of the corresponding **Slave**.

Table 4.4.3.2

Register Address*1	Parameter	Data Type	Attribute	Description
24608_d (6020_h) 24609_d (6021_h)	**Slave GPI**: Channel*2 1~32	uint32_t	Read-only	Display the status of GPI (1~32). Bit 0: 0: GPI1 is OFF. Bit 0: 1: GPI1 is ON. ⋮ ⋮ Bit 32: 0: GPI32 is OFF. Bit 32: 1: GPI32 is ON.
24672_d (6060_h) 24673_d (6061_h)	**Slave GPO**: Channel 1~32	uint32_t	Read / Write	Set GPO (1~32) or display the status of GPO (1~32). Bit 0: 0: GPO1 is OFF. Bit 0: 1: GPO1 is ON. ⋮ ⋮ Bit 32: 0: GPO32 is OFF. Bit 32: 1: GPO32 is ON.

Note:

*1: The register address of the parameters of each channel section: Register address + 2 * N (N_{max} = 7)

*2: **Slave GPI** and **Slave GPO** each have a maximum of 256 channels, and every 32 channels are divided into 8 sections (Channel 1~32, 33~64, ..., 225~256).

4.4.4 User Table

By accessing the registers assigned for **User Table** parameters, users can read or write the index values in **User Table**[*1]. Modbus interface only provides 128 indexes; users can adjust the index area via $4X24735_d$.

Note:

*1: **User Table** is stored in the memory of HIMC.

Table 4.4.4.1

Register Address[*1]	Parameter	Data Type	Attribute	Description
24735_d ($609F_h$)	Selected Table Region Index	uint16_t	Read / Write	Set or display the Nth index area.

Note:

*1: N indicates the index area from index N * 128 to index (N + 1) * 128 - 1.

■ When the data type is **float**, the assigned registers for **User Table** are as follows:

Table 4.4.4.2

Register Address[*1]	Parameter	Data Type	Attribute	Description
24736_d ($60A0_h$)	index	float	Read / Write	Set or display the index value in **User Table**.
24737_d ($60A1_h$)				

Note:

*1: Register address of index N = Register address + N * 2 (N_{max} = 127)

■ When the data type is **double**[*1], the assigned registers for **User Table** are as follows:

Table 4.4.4.3

Register Address[*2]	Parameter	Data Type	Attribute	Description
24736_d ($60A0_h$)	index	double	Read / Write	Set or display the index value in **User Table**.
24737_d ($60A1_h$)				
24738_d ($60A2_h$)				
24739_d ($60A3_h$)				

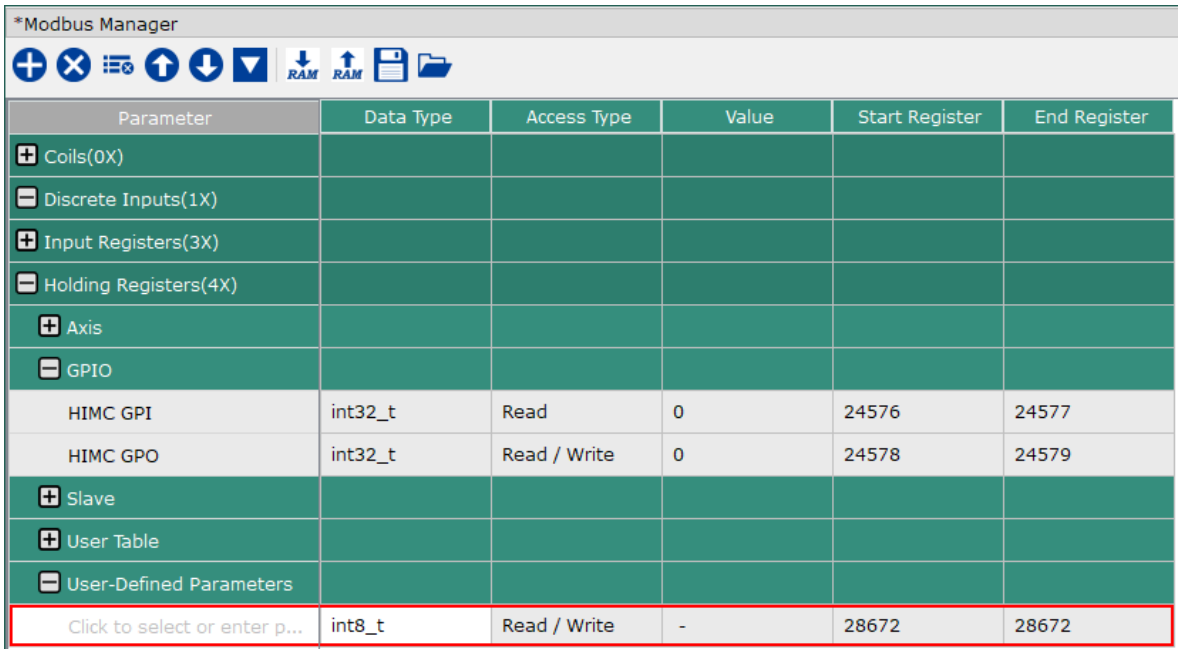
Note:

*1: Switch it by the internal variables `modbus_is_support_double` and `modbus_is_default_double`.

*2: Register address of index N = Register address + N * 4 (N_{max} = 127)

4.4.5 User-defined Parameters

By accessing the registers assigned for **User-defined Parameters**, users can read or write the user-defined parameters. The range of register address is from $4X28672_d$ ~ $4X40959_d$. Users need to define the desired parameters in iA Studio first.



*Modbus Manager

Parameter	Data Type	Access Type	Value	Start Register	End Register
Coils(0X)					
Discrete Inputs(1X)					
Input Registers(3X)					
Holding Registers(4X)					
Axis					
GPIO					
HIMC GPI	int32_t	Read	0	24576	24577
HIMC GPO	int32_t	Read / Write	0	24578	24579
Slave					
User Table					
User-Defined Parameters					
Click to select or enter p...	int8_t	Read / Write	-	28672	28672

Figure 4.4.5.1 Modbus Manager

Note:

For how to set user-defined parameters, please refer to section 4.12 **Modbus Manager** in “iA Studio User Guide.”

4.4.6 Undefined Registers

Register addresses from $4X40960_d$ to $4X65535_d$ are not defined. Users are allowed to use these registers freely.