RS-M8196F
Motion Control Module

Function Reference
(Version 1.0)

ICP DAS CO., LTD.

Warranty

All products manufactured by ICPDAS Inc. are warranted against defective materials for a period of one year from the date of delivery to the original purchaser.

Warning

ICPDAS Inc. assumes no liability for damages consequent to the use of this product. ICPDAS Inc. reserves the right to change this manual at any time without notice. The information furnished by ICPDAS Inc. is believed to be accurate and reliable. However, no responsibility is assumed by ICPDAS Inc. for its use, or for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademark

The names used for identification only maybe registered trademarks of their respective companies.

License

The user can use, modify and backup this software on a single machine. The user may not reproduce, transfer or distribute this software, or any copy, in whole or in part.

Contents

1 Preface

The RS-M8196F is a remote serial 6-axis stepping/pulse-type servo motion controller which uses Modbus RTU as its communication protocol. The RS-M8196F is a slave in a Modbus RTU network and supports all standard Modbus function codes. Three serial interfaces are provided (RS232, RS485 and RS422) and the user can select any of the three serial interfaces for communication. The RS-M8196F can expand a PLC system by adding 6-axis motion control support.

The motion controller is suitable for general-purpose motion control applications. In additions to its wide speed range, this intelligent motion controller also has a variety of built-in motion control functions, such as 2- to 6-axis linear interpolation, 2- and 3-axis circular interpolation, 3-axis helical interpolation, T/S-curve acceleration/deceleration, and automatic home search, etc. In addition the RS-M8196F acts as an FRnet master and can control up to 16 remote DIO slaves (128 digital outputs and 128 digital inputs). FRnet is a two-wire serial bus and has a scan interval of 0.72 ms and it is specifically designed for easy and cost effective wiring. ICPDAS provides a large range of FRnet I/O terminal boards and modules.

Libraries are provided for the RS-M8196F which allows the user to remotely control the device without having to understand the Modbus communication. The libraries (DLL, Dynamic-link library) support all Windows operation systems currently on the market. Examples are provided to show how to use the libraries in the different integrated development environment (IDE), like Visual Studio, Delphi, Borland, LabVIEW, Visual Basic 6.0, etc.

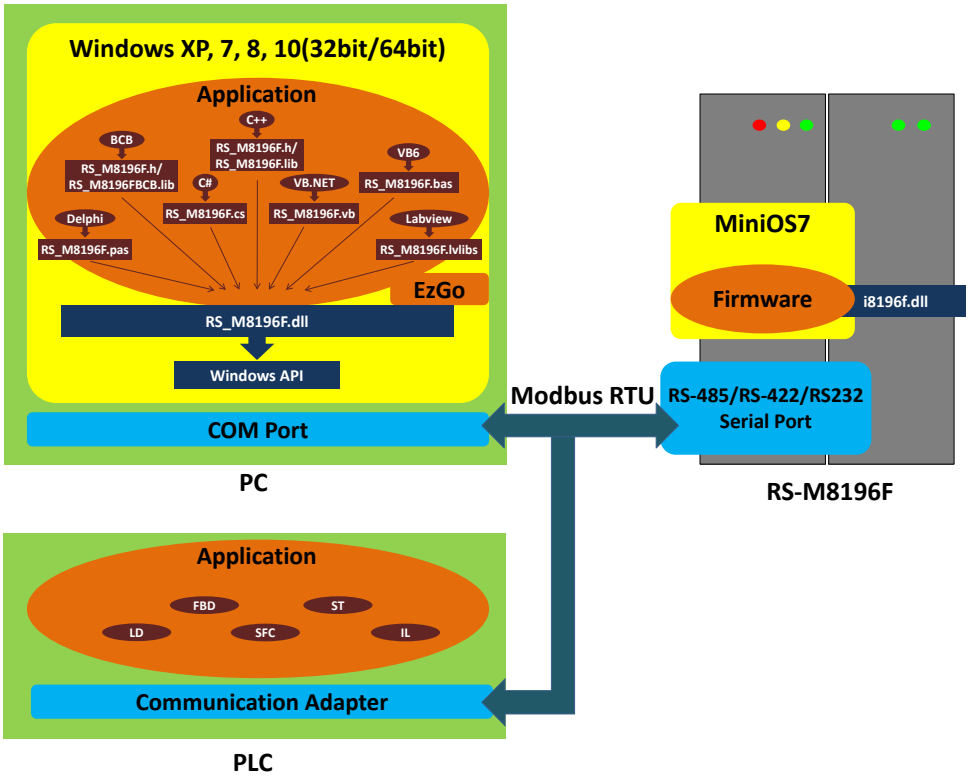


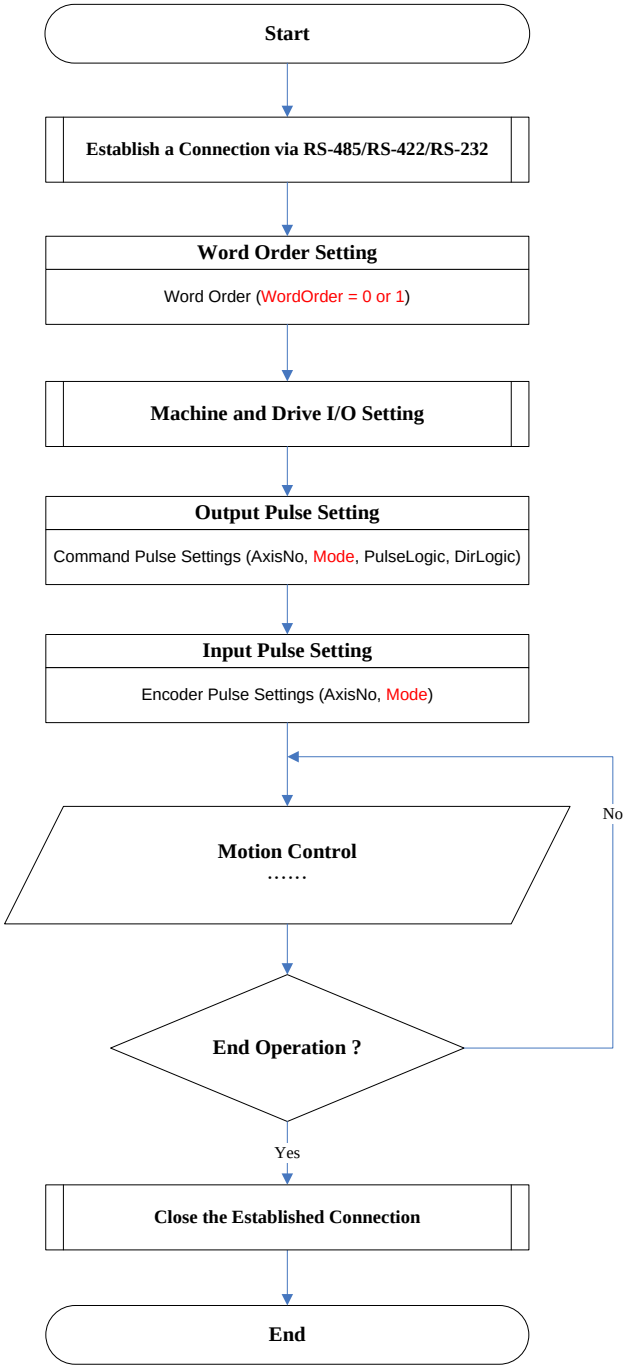
Figure 1.1 Software application

The following sections will explain how to use each functions.

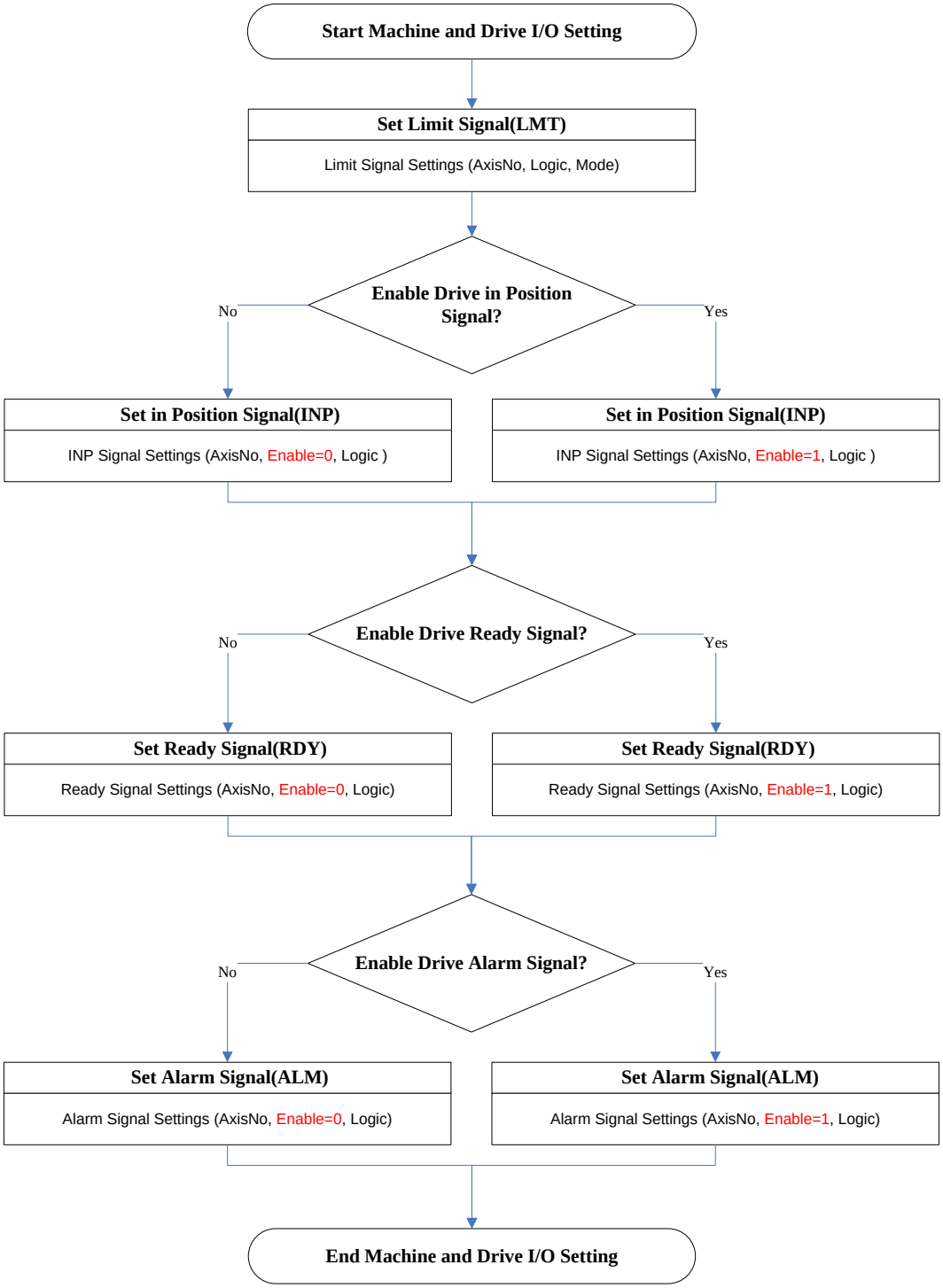
Section 2: System Setting Related Functions
Section 3: Motion Control Signal Related Functions
Section 4: Auto Homing Related Functions
Section 5: Motion Control Related Functions
Section 6: Other Application Functions
Section 7: GPIO and FRnet Related Functions
Section 8: Version Related Functions

1.1 Flow Chart

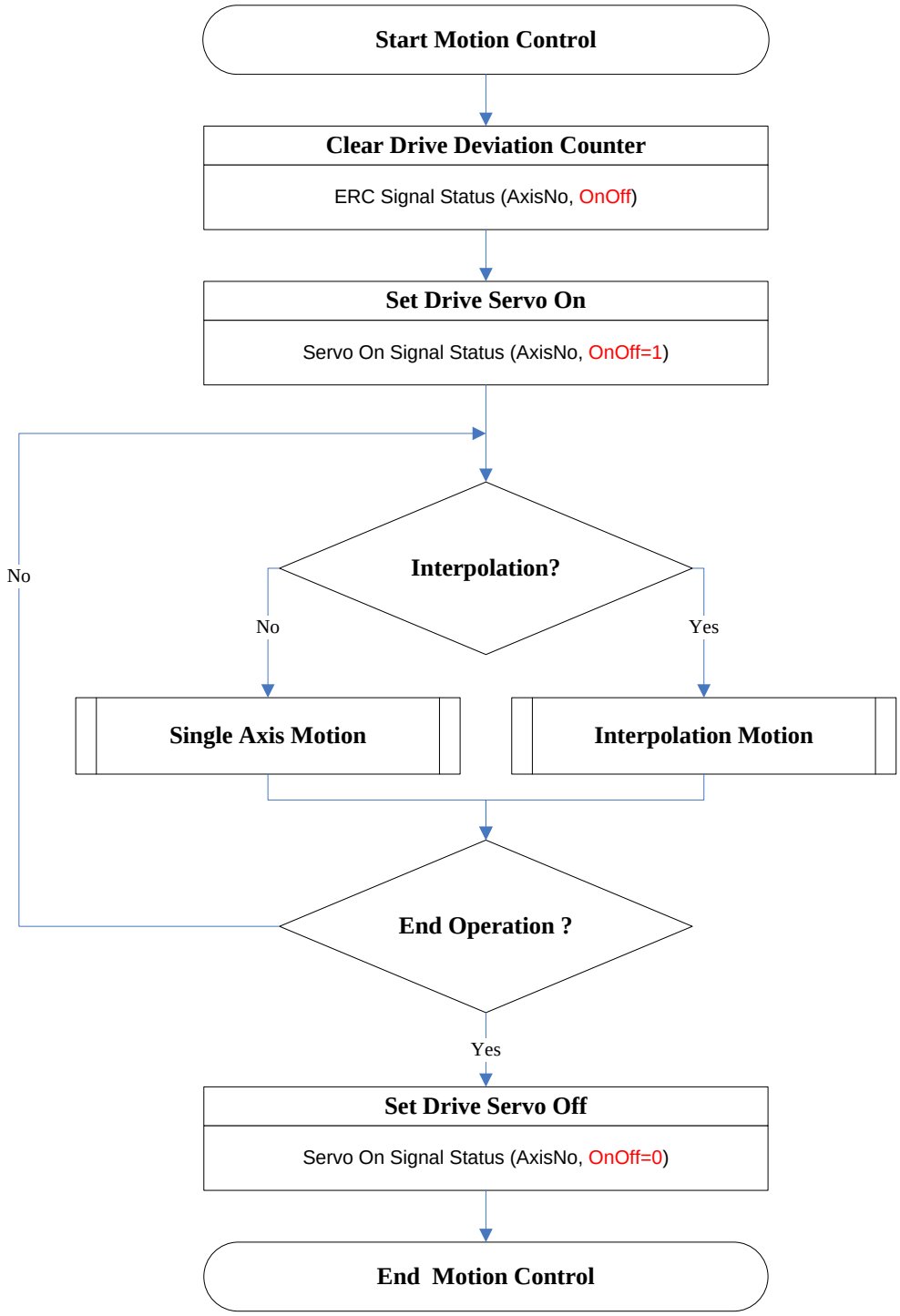
This section illustrates the basic function call sequence required for the initialization of the RS-M8196F and motion command execution.



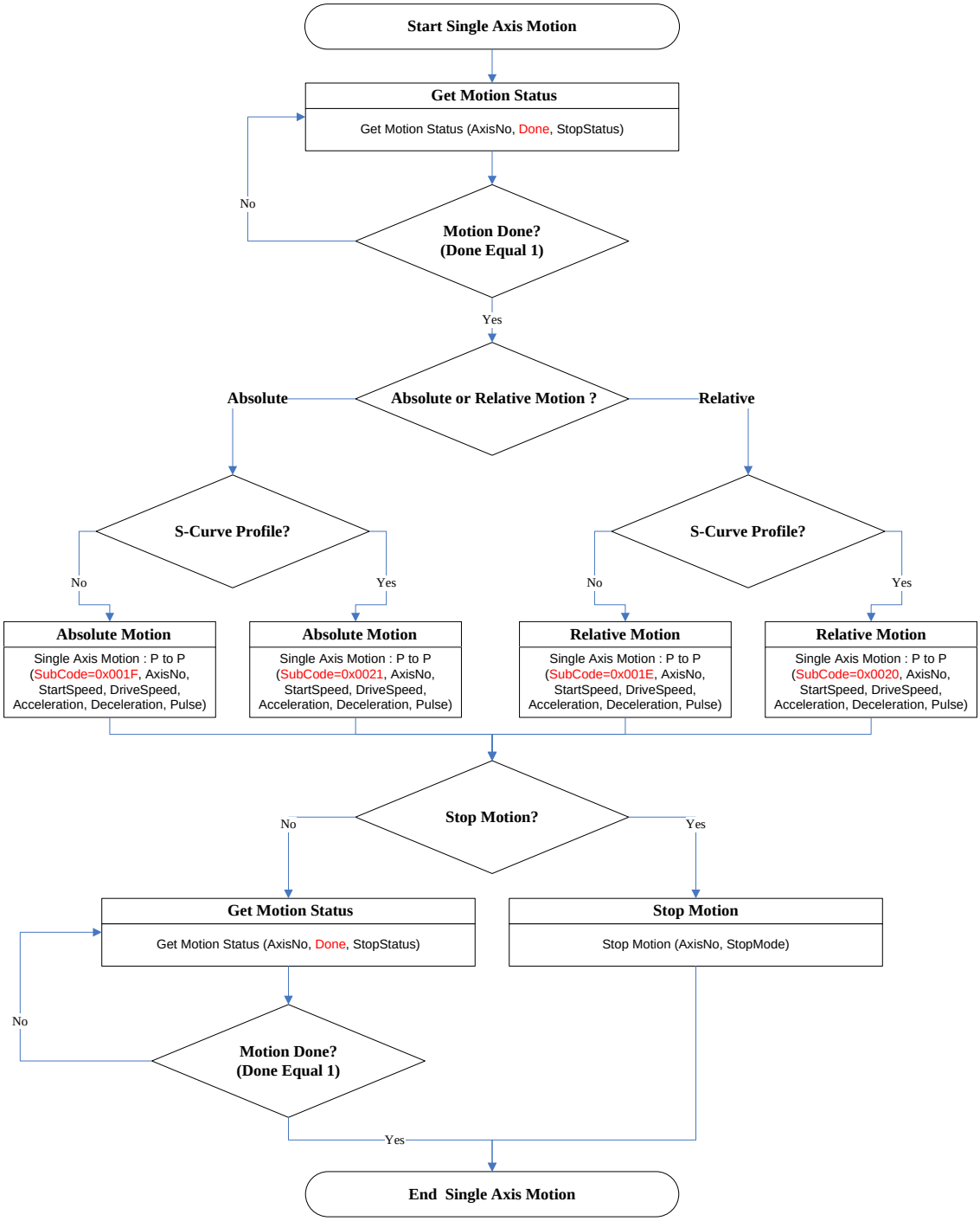
1.1.1 Machine and Drive I/O Setting



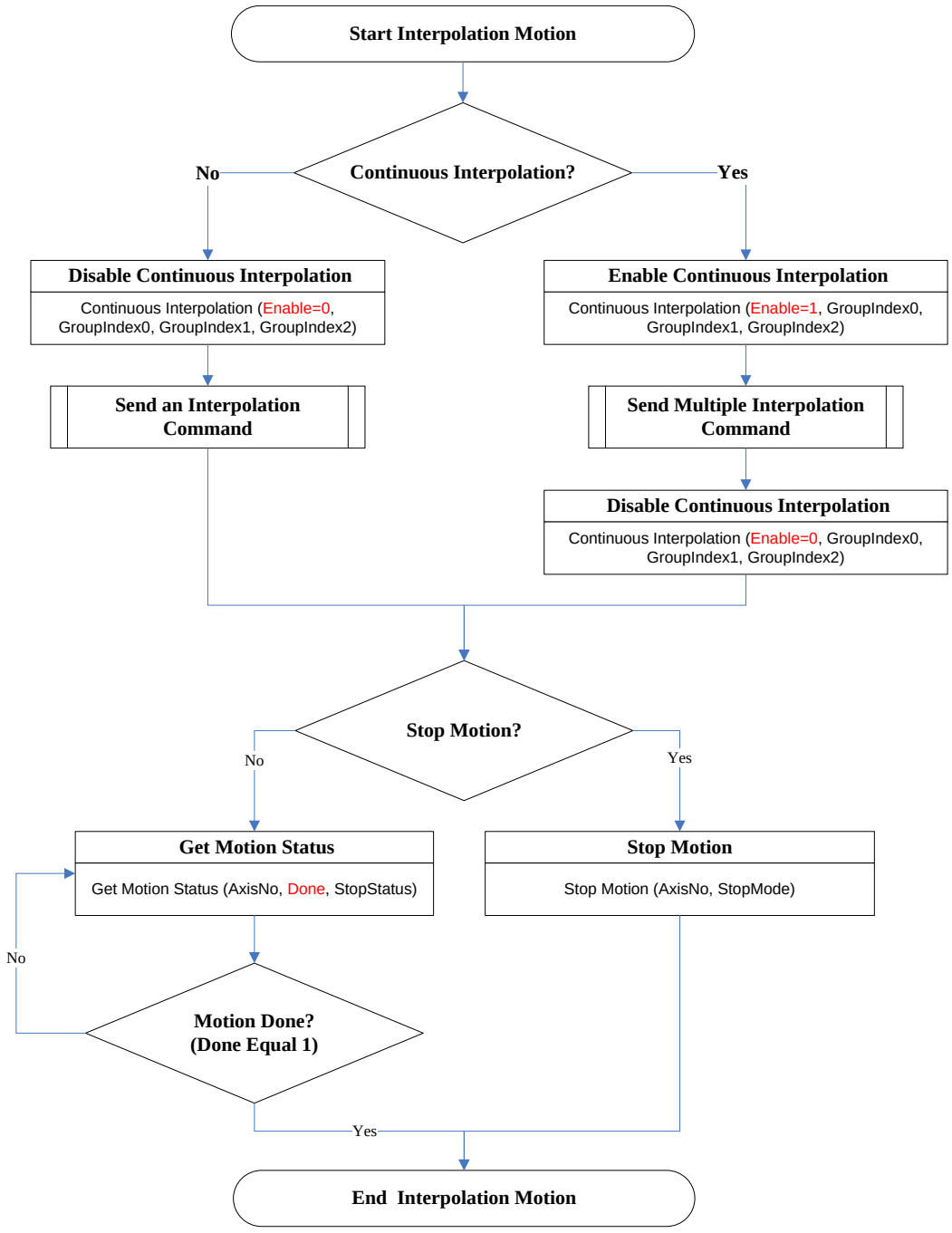
1.1.2 Motion Control



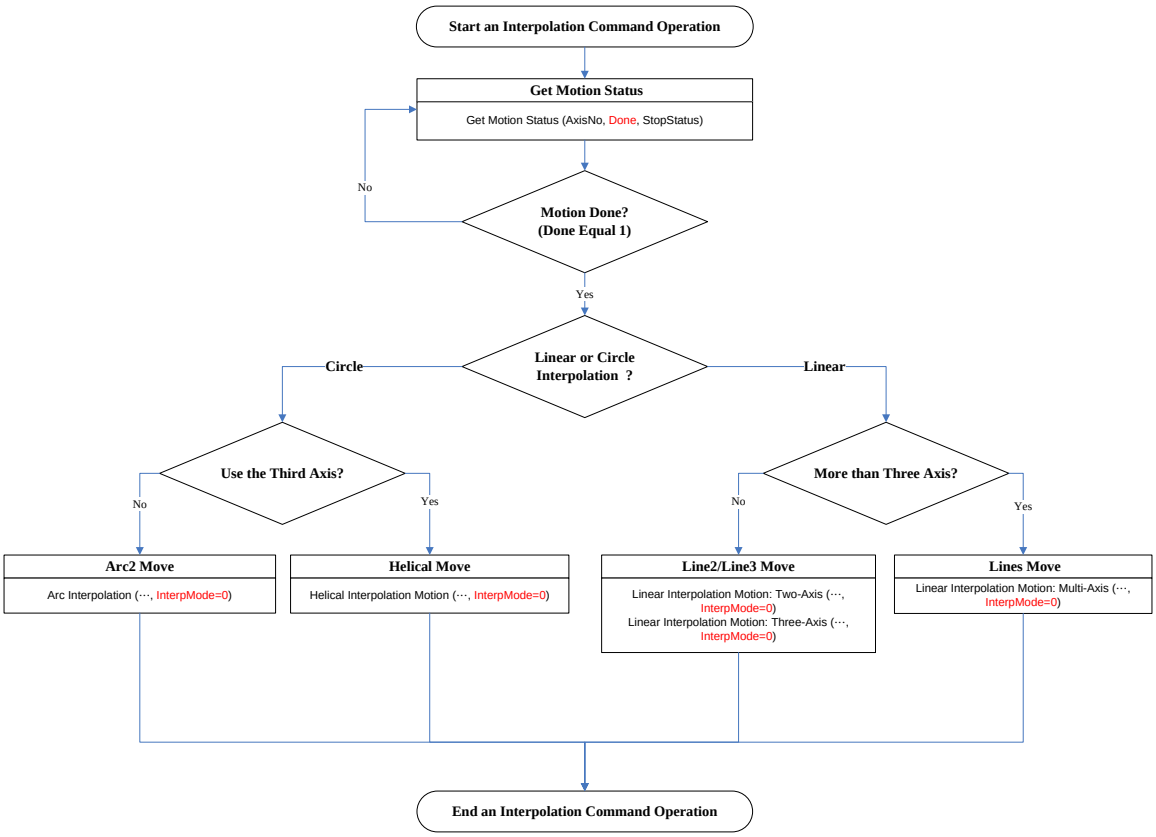
1.1.3 Single Axis Motion Control



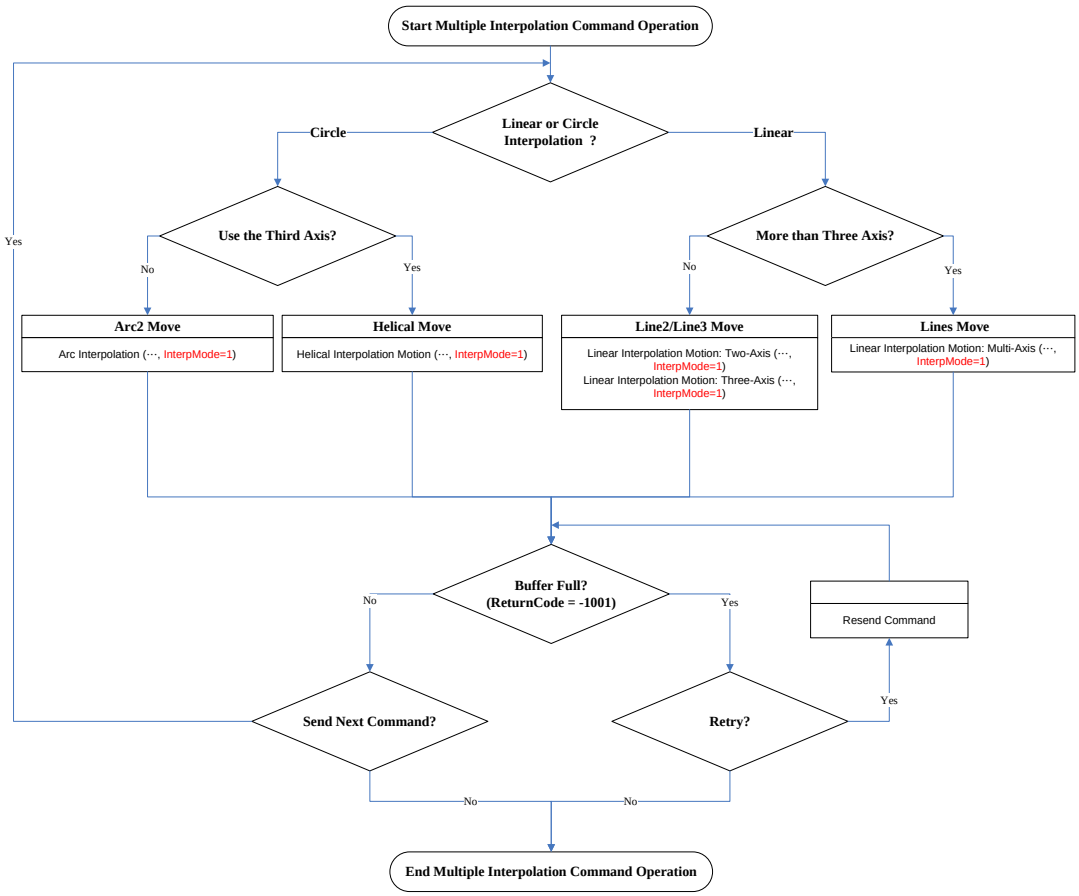
1.1.4 Interpolation Motion Control



1.1.5 General Interpolation Control



1.1.6 Multiple Interpolation Control



2 System Setting

Name	Type*	Function Code	Description
The 32 Bit Values Word Order Setting			
Word Order	R/W	0x03, 0x04, 0x06, 0x10	Set the order of the most and least significant WORD (MSW, LSW).

* R: read, W: write

2.1 The 32 Bit Values Word Order Setting

2.1.1 Word Order

Description:

Set the order of the most and least significant WORD (MSW, LSW). Since Modbus uses 16 bit registers to hold values, the 32 bit value must split between two registers.

Parameters:

Name	Type	Description	Address
WordOrder	R/W	0: MSW at the first Register 1: MSW at the second Register	Function Code: 0x03, 0x04, 0x06, 0x10 Starting Address: 9 (Base 0)

Sample:

Set Word Order

Request					
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**	**Register Value1**
1~247	0x10	9	1	2	WordOrder

Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x10	9	1

Get Word Order

Request			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x03	9	1

Response			
Slave Address	**Function Code**	**Byte Count**	**Register Value1**
1~247	0x03	2	WordOrder

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

3 Motion Control Signal

Name	Type*	Function Code	Description
Machine and Drive I/O Settings			
Servo On Signal Status	R/W	0x01, 0x02, 0x05, 0x0F	Set/Get the "SRV_ON" signal status.
ERC Signal Status	R/W	0x01, 0x02, 0x05, 0x0F	Set/Get the "ERC" signal status.
Alarm Reset Signal Status	R/W	0x01, 0x02, 0x05, 0x0F	Set/Get the "ALM_RST" signal status.
Alarm Signal Setting	W	SubCode	Set the "ALARM" signal operating mode. If function has been enabled and the signal is active then no command pulses will be outputted.
INP Signal Setting	W	SubCode	Set the "INP" signal operating mode. If function has been enabled, then controller waits until the signal has been triggered before continuing to execute next motion command.
Ready Signal Setting	W	SubCode	Set the "RDY" signal operating mode. If function has been enabled and the signal is inactive then no command pulses will be outputted.
Limit Signal Setting	W	SubCode	Set the "LMT" signal operating mode. If the "LMT+" signal is being triggered while the movement is in positive direction the motion stops according to the set stop mode. On the other hand the motion will stop when moving in negative direction and the "LMT-" signal is active.
I/O Status	R	0x04	Get the motion I/O status of axis.
Motion Control Pulse Counter Settings			
Command Pulse Setting	W	SubCode	Set the command pulse mode for each axis.
Encoder Pulse Setting	W	SubCode	Set the encoder pulse mode for each axis.
Command Pulse Counter	R/W	0x03,0x04,0x06, 0x10	Set/Get the command pulse counter value.

Encoder Pulse Counter	R/W	0x03,0x04,0x06, 0x10	Set/Get the encoder pulse counter value.
Enable Ring Counter	R/W	0x03,0x04,0x06, 0x10	Set the maximum ring counter value for both the encoder and command pulse counter. The function will be enabled when the value is set.
Disable Ring Counter	W	SubCode	Disable the ring counter function.

* R: read, W: write

3.1 Machine and Drive I/O Settings

3.1.1 Servo On Signal Status

Description:

Set/Get the "SRV_ON" signal status.

Parameters:

Name	Type	Description	Address
OnOff	R/W	Signal status 0: OFF 1: ON	Function Code: 0x01, 0x02, 0x05, 0x0F Starting Address:

Axis	Base 0	Base 1
Axis0	291	292
Axis1	292	293
Axis2	293	294
Axis3	294	295
Axis4	295	296
Axis5	296	297

Sample:

Set Signal Status

Request					
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**	**Value**
1~247	0x0F	291	6	1	OnOff

Bit	Status
Bit0	Axis0 Signal Status
Bit1	Axis1 Signal Status
Bit2	Axis2 Signal Status
Bit3	Axis3 Signal Status
Bit4	Axis4 Signal Status
Bit5	Axis5 Signal Status
Bit6, Bit7	Reserved

Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x0F	291	6

Get Signal Status

Request			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x01	291	6

Response			
Slave Address	**Function Code**	**Byte Count**	**Status**
1~247	0x01	1	OnOff

Bit	**Status**
Bit0	Axis0 Signal Status
Bit1	Axis1 Signal Status
Bit2	Axis2 Signal Status
Bit3	Axis3 Signal Status
Bit4	Axis4 Signal Status
Bit5	Axis5 Signal Status
Bit6, Bit7	Reserved

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

3.1.2 ERC Signal Status

Description:

Set/Get the "ERC" signal status.

Parameters:

Name	Type	Description	Address
OnOff	R/W	Signal status 0: OFF 1: ON	Function Code: 0x01, 0x02, 0x05, 0x0F Starting Address:

Axis	Base 0	Base 1
Axis0	297	298
Axis1	298	299
Axis2	299	300
Axis3	300	301
Axis4	301	302
Axis5	302	303

Sample:

Set Signal Status

Request					
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**	**Value**
1~247	0x0F	297	6	1	OnOff

Bit	Status
Bit0	Axis0 Signal Status
Bit1	Axis1 Signal Status
Bit2	Axis2 Signal Status
Bit3	Axis3 Signal Status
Bit4	Axis4 Signal Status
Bit5	Axis5 Signal Status
Bit6, Bit7	Reserved

Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x0F	297	6

Get Signal Status

Request			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x01	297	6

Response			
Slave Address	**Function Code**	**Byte Count**	**Status**
1~247	0x01	1	OnOff

Bit	**Status**
Bit0	Axis0 Signal Status
Bit1	Axis1 Signal Status
Bit2	Axis2 Signal Status
Bit3	Axis3 Signal Status
Bit4	Axis4 Signal Status
Bit5	Axis5 Signal Status
Bit6, Bit7	Reserved

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

3.1.3 Alarm Reset Signal Status

Description:

Set/Get the "ALM_RST" signal status.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Address</th></tr>
<tr><td>OnOff</td><td>R/W</td><td>Signal status
0: OFF
1: ON</td><td>Function Code: 0x01, 0x02, 0x05, 0x0F

Starting Address:
<table>
<tr><th>Axis</th><th>Base 0</th><th>Base 1</th></tr>
<tr><td>Axis0</td><td>303</td><td>304</td></tr>
<tr><td>Axis1</td><td>304</td><td>305</td></tr>
<tr><td>Axis2</td><td>305</td><td>306</td></tr>
<tr><td>Axis3</td><td>306</td><td>307</td></tr>
<tr><td>Axis4</td><td>307</td><td>308</td></tr>
<tr><td>Axis5</td><td>308</td><td>309</td></tr>
</table></td></tr>
</table>

Sample:

Set Signal Status

<table>
<tr><th colspan="6">Request</th></tr>
<tr><th>Slave Address</th><th>Function Code</th><th>Starting Address</th><th>Quantity</th><th>Byte Count</th><th>Value</th></tr>
<tr><td>1~247</td><td>0x0F</td><td>303</td><td>6</td><td>1</td><td>OnOff
<table>
<tr><th>Bit</th><th>Status</th></tr>
<tr><td>Bit0</td><td>Axis0 Signal Status</td></tr>
<tr><td>Bit1</td><td>Axis1 Signal Status</td></tr>
<tr><td>Bit2</td><td>Axis2 Signal Status</td></tr>
<tr><td>Bit3</td><td>Axis3 Signal Status</td></tr>
<tr><td>Bit4</td><td>Axis4 Signal Status</td></tr>
<tr><td>Bit5</td><td>Axis5 Signal Status</td></tr>
<tr><td>Bit6, Bit7</td><td>Reserved</td></tr>
</table></td></tr>
</table>

<table>
<tr><th colspan="4">Response</th></tr>
<tr><th>Slave Address</th><th>Function Code</th><th>Starting Address</th><th>Quantity</th></tr>
<tr><td>1~247</td><td>0x0F</td><td>303</td><td>6</td></tr>
</table>

Get Signal Status

Request			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x01	303	6

Response			
Slave Address	**Function Code**	**Byte Count**	**Status**
1~247	0x01	1	OnOff

Bit	Status
Bit0	Axis0 Signal Status
Bit1	Axis1 Signal Status
Bit2	Axis2 Signal Status
Bit3	Axis3 Signal Status
Bit4	Axis4 Signal Status
Bit5	Axis5 Signal Status
Bit6, Bit7	Reserved

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

3.1.4 Alarm Signal Settings

Description:

Set the "ALARM" signal operating mode. If function has been enabled and the signal is active then no command pulses will be outputted.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Address</th></tr>
<tr><td>SubCode</td><td>W</td><td>0x000E</td><td rowspan="4">Function Code: 0x10

Starting Address: 1280 (Base 0)</td></tr>
<tr><td>AxisNo</td><td>W</td><td>Axis definition
<table>
<tr><th>Axis</th><th>Value</th></tr>
<tr><td>AXIS_0</td><td>0x01</td></tr>
<tr><td>AXIS_1</td><td>0x02</td></tr>
<tr><td>AXIS_2</td><td>0x04</td></tr>
<tr><td>AXIS_3</td><td>0x08</td></tr>
<tr><td>AXIS_4</td><td>0x10</td></tr>
<tr><td>AXIS_5</td><td>0x20</td></tr>
</table></td></tr>
<tr><td>Enable</td><td>W</td><td>0: Disable the signal
1: Enable the signal</td></tr>
<tr><td>Logic</td><td>W</td><td>Active logic level
0: Low active
1: High active</td></tr>
</table>

Sample:

Execute Command

Request				
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**
1~247	0x10	1280	4	8
Register Value1 (SubCode)	**Register Value2**	**Register Value3**	**Register Value4**	
0x000E	AxisNo	Enable	Logic	

Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x10	1280	4

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

3.1.5 INP Signal Settings

Description:

Set the "INP" signal operating mode. If function has been enabled, then controller waits until the signal has been triggered before continuing to execute next motion command.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Address</th></tr>
<tr><td>SubCode</td><td>W</td><td>0x000F</td><td rowspan="5">Function Code: 0x10

Starting Address: 1280 (Base 0)</td></tr>
<tr><td>AxisNo</td><td>W</td><td>Axis definition
<table>
<tr><th>Axis</th><th>Value</th></tr>
<tr><td>AXIS_0</td><td>0x01</td></tr>
<tr><td>AXIS_1</td><td>0x02</td></tr>
<tr><td>AXIS_2</td><td>0x04</td></tr>
<tr><td>AXIS_3</td><td>0x08</td></tr>
<tr><td>AXIS_4</td><td>0x10</td></tr>
<tr><td>AXIS_5</td><td>0x20</td></tr>
</table></td></tr>
<tr><td>Enable</td><td>W</td><td>0: Disable the signal
1: Enable the signal</td></tr>
<tr><td>Logic</td><td>W</td><td>Active logic level
0: Low active
1: High active</td></tr>
</table>

Sample:

Execute Command

Request				
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**
1~247	0x10	1280	4	8
Register Value1 (SubCode)	**Register Value2**	**Register Value3**	**Register Value4**	
0x000F	AxisNo	Enable	Logic	

Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x10	1280	4

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

3.1.6 Ready Signal Settings

Description:

Set the "RDY" signal operating mode. If function has been enabled and the signal is inactive then no command pulses will be outputted.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Address</th></tr>
<tr><td>SubCode</td><td>W</td><td>0x0010</td><td rowspan="4">Function Code: 0x10

Starting Address: 1280 (Base 0)</td></tr>
<tr><td>AxisNo</td><td>W</td><td>Axis definition
<table>
<tr><th>Axis</th><th>Value</th></tr>
<tr><td>AXIS_0</td><td>0x01</td></tr>
<tr><td>AXIS_1</td><td>0x02</td></tr>
<tr><td>AXIS_2</td><td>0x04</td></tr>
<tr><td>AXIS_3</td><td>0x08</td></tr>
<tr><td>AXIS_4</td><td>0x10</td></tr>
<tr><td>AXIS_5</td><td>0x20</td></tr>
</table></td></tr>
<tr><td>Enable</td><td>W</td><td>0: Disable the signal
1: Enable the signal</td></tr>
<tr><td>Logic</td><td>W</td><td>Active logic level
0: Low active
1: High active</td></tr>
</table>

Sample:

Execute Command

Request				
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**
1~247	0x10	1280	4	8
Register Value1 (SubCode)	**Register Value2**	**Register Value3**	**Register Value4**	
0x0010	AxisNo	Enable	Logic	

Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x10	1280	4

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

3.1.7 Limit Signal Settings

Description:

Set the "LMT" signal operating mode. If the "LMT+" signal is being triggered while the movement is in positive direction the motion stops according to the set stop mode. On the other hand the motion will stop when moving in negative direction and the "LMT-" signal is active.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Address</th></tr>
<tr><td>SubCode</td><td>W</td><td>0x0006</td><td rowspan="4">Function Code: 0x10

Starting Address: 1280 (Base 0)</td></tr>
<tr><td>AxisNo</td><td>W</td><td>Axis definition
<table>
<tr><th>Axis</th><th>Value</th></tr>
<tr><td>AXIS_0</td><td>0x01</td></tr>
<tr><td>AXIS_1</td><td>0x02</td></tr>
<tr><td>AXIS_2</td><td>0x04</td></tr>
<tr><td>AXIS_3</td><td>0x08</td></tr>
<tr><td>AXIS_4</td><td>0x10</td></tr>
<tr><td>AXIS_5</td><td>0x20</td></tr>
</table></td></tr>
<tr><td>Logic</td><td>W</td><td>Active logic level
0: Low active
1: High active</td></tr>
<tr><td>Mode</td><td>W</td><td>0: Disable
1: Decelerating stop
2: Sudden stop</td></tr>
</table>

Sample:

Execute Command

Request				
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**
1~247	0x10	1280	4	8
Register Value1 (SubCode)	**Register Value2**	**Register Value3**	**Register Value4**	
0x0006	AxisNo	Logic	Mode	

Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x10	1280	4

Exception Code:
If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

3.1.8 I/O Status

Description:

Get the motion I/O status of axis.

Parameters:

Name	Type	Description	Address
Status	R	Signal state, see Table 3.1 If bit is zero: the corresponding signal is OFF If bit is one: the corresponding signal is ON	Function Code: 0x04 Starting Address:

Axis	Base 0	Base 1
Axis0	36	37
Axis1	37	38
Axis2	38	39
Axis3	39	40
Axis4	40	41
Axis5	41	42

Table 3.1 I/O Signal State

Bit	Signal	Description
Bit 0	Reserved	Reserved
Bit 1	LMT+	Positive limit switch
Bit 2	LMT-	Negative limit switch
Bit 3	EMG	Emergency stop switch
Bit 4	ALARM	Servo drive alarm signal
Bit 5	HOME (ORG)	Home switch
Bit 6	SLD (NHOME)	Slow down switch
Bit 7	INP	Servo drive in-position signal
Bit 8	EZ	Servo drive Z phase (Index signal)
Bit 9	RDY	Servo drive ready signal
Bit 10	LTC	Latch input
Bit 11	Reserved	Reserved
Bit 12	Reserved	Reserved
Bit 13	SRV_ON	Signal for activating servo drive
Bit 14	ERC	Output signal for clearing position deflection of servo drive
Bit 15	ALM_RST	Signal for resetting servo drive alarm

Sample:

Get Status

Request			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x04	36	6

Response					
Slave Address	**Function Code**	**Byte Count**			
1~247	0x04	12			
Register Value1	**Register Value2**	**Register Value3**	**Register Value4**	**Register Value5**	**Register Value6**
Axis0 I/O Status	Axis1 I/O Status	Axis3 I/O Status	Axis3 I/O Status	Axis4 I/O Status	Axis5 I/O Status

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

3.2 Motion Control Pulse Counter Settings

3.2.1 Command Pulse Settings

Description:

Set the command pulse mode for each axis.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Address</th></tr>
<tr><td>SubCode</td><td>W</td><td>0x0004</td><td rowspan="6">Function Code: 0x10

Starting Address: 1280 (Base 0)</td></tr>
<tr><td>AxisNo</td><td>W</td><td>Axis definition
<table>
<tr><th>Axis</th><th>Value</th></tr>
<tr><td>AXIS_0</td><td>0x01</td></tr>
<tr><td>AXIS_1</td><td>0x02</td></tr>
<tr><td>AXIS_2</td><td>0x04</td></tr>
<tr><td>AXIS_3</td><td>0x08</td></tr>
<tr><td>AXIS_4</td><td>0x10</td></tr>
<tr><td>AXIS_5</td><td>0x20</td></tr>
</table></td></tr>
<tr><td>Mode</td><td>W</td><td>0: Pulse/Direction
1: CW/CCW
2: A/B Phase</td></tr>
<tr><td>PulseLogic</td><td>W</td><td>Active logic level of pulse signal (CW signal)
1: Low active
0: High active</td></tr>
<tr><td>DirLogic</td><td>W</td><td>Active logic level of direction signal (CCW signal)
1: Low active
0: High active

Note that in "CW/CCW" and "A/B Phase" mode this parameter is invalid</td></tr>
</table>

Sample:

Execute Command

Request				
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**
1~247	0x10	1280	5	10
Register Value1 (SubCode)	**Register Value2**	**Register Value3**	**Register Value4**	**Register Value5**
0x0004	AxisNo	Mode	PulseLogic	DirLogic

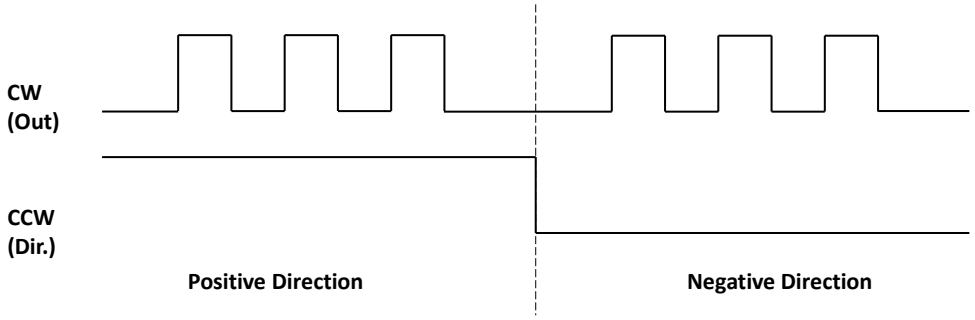
Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x10	1280	5

Exception Code:

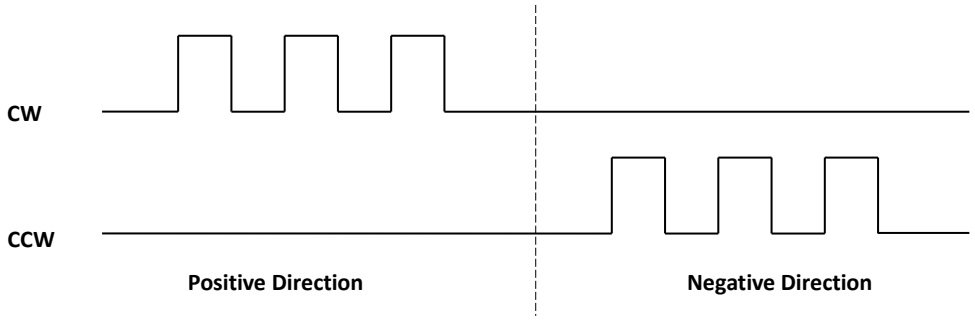
If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

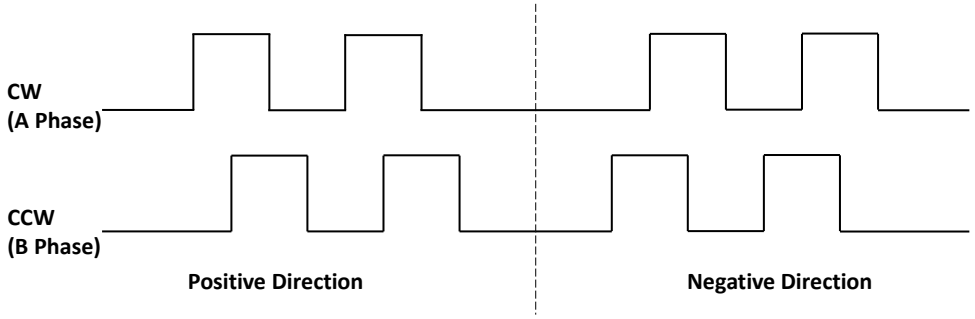
Remarks:



(a) Mode 0: Pulse/Direction



(b) Mode 1: CW/CCW



(c) Mode 2: AB Phase

Figure 3.1 Pulse Mode

3.2.2 Encoder Pulse Settings

Description:

Set the encoder pulse mode for each axis.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Address</th></tr>
<tr><td>SubCode</td><td>W</td><td>0x0005</td><td rowspan="3">Function Code: 0x10

Starting Address: 1280 (Base 0)</td></tr>
<tr><td>AxisNo</td><td>W</td><td>Axis definition
<table>
<tr><th>Axis</th><th>Value</th></tr>
<tr><td>AXIS_0</td><td>0x01</td></tr>
<tr><td>AXIS_1</td><td>0x02</td></tr>
<tr><td>AXIS_2</td><td>0x04</td></tr>
<tr><td>AXIS_3</td><td>0x08</td></tr>
<tr><td>AXIS_4</td><td>0x10</td></tr>
<tr><td>AXIS_5</td><td>0x20</td></tr>
</table></td></tr>
<tr><td>Mode</td><td>W</td><td>
<table>
<tr><th>Mode</th><th>Value</th></tr>
<tr><td>CW/CCW</td><td>1</td></tr>
<tr><td>A/B Phase</td><td>2</td></tr>
<tr><td>A/B Phase divide 2</td><td>3</td></tr>
<tr><td>A/B Phase divide 4</td><td>4</td></tr>
</table></td></tr>
</table>

Sample:

Execute Command

Request				
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**
1~247	0x10	1280	3	6
Register Value1 (SubCode)	**Register Value2**	**Register Value3**		
0x0005	AxisNo	Mode		

Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x10	1280	3

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

3.2.3 Command Pulse Counter

Description:

Set/Get the command pulse counter value.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Address</th></tr>
<tr><td>LogicPos</td><td>R/W</td><td>Value of command pulse counter</td><td>Function Code:
0x03,0x04,0x06,0x10

This parameter is 32 bit integer data, it takes two registers.

Starting Address:
<table>
<tr><th>Axis</th><th>Base 0</th><th>Base 1</th></tr>
<tr><td>Axis0</td><td>60,61</td><td>61,62</td></tr>
<tr><td>Axis1</td><td>62,63</td><td>63,64</td></tr>
<tr><td>Axis2</td><td>64,65</td><td>65,66</td></tr>
<tr><td>Axis3</td><td>66,67</td><td>67,68</td></tr>
<tr><td>Axis4</td><td>68,69</td><td>69,70</td></tr>
<tr><td>Axis5</td><td>70,71</td><td>71,72</td></tr>
</table>
</td></tr>
</table>

Sample:

Set Value

Request					
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**	
1~247	0x10	60	12	24	
Register Value1	**Register Value2**	**Register Value3**	**Register Value4**	**Register Value5**	**Register Value6**
Axis0 LogicPos		Axis1 LogicPos		Axis2 LogicPos	
LSW	MSW	LSW	MSW	LSW	MSW
Register Value7	**Register Value8**	**Register Value9**	**Register Value10**	**Register Value11**	**Register Value12**
Axis3 LogicPos		Axis4 LogicPos		Axis5 LogicPos	
LSW	MSW	LSW	MSW	LSW	MSW

Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x10	60	12

Get Value

Request			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x03	60	12

Response					
Slave Address	**Function Code**	**Byte Count**			
1~247	0x03	24			
Register Value1	**Register Value2**	**Register Value3**	**Register Value4**	**Register Value5**	**Register Value6**
Axis0 LogicPos		Axis1 LogicPos		Axis2 LogicPos	
LSW	MSW	LSW	LSW	MSW	LSW
Register Value7	**Register Value8**	**Register Value9**	**Register Value10**	**Register Value11**	**Register Value12**
Axis3 LogicPos		Axis4 LogicPos		Axis5 LogicPos	
LSW	MSW	LSW	LSW	MSW	LSW

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

3.2.4 Encoder Pulse Counter

Description:

Set/Get the encoder pulse counter value.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Address</th></tr>
<tr><td>EncPos</td><td>R/W</td><td>Value of encoder pulse counter</td><td>Function Code: 0x03,0x04,0x06,0x10

This parameter is 32 bit integer data, it takes two registers.

Starting Address:
<table>
<tr><th>Axis</th><th>Base 0</th><th>Base 1</th></tr>
<tr><td>Axis0</td><td>72,73</td><td>73,74</td></tr>
<tr><td>Axis1</td><td>74,75</td><td>75,76</td></tr>
<tr><td>Axis2</td><td>76,77</td><td>77,78</td></tr>
<tr><td>Axis3</td><td>78,79</td><td>79,80</td></tr>
<tr><td>Axis4</td><td>80,81</td><td>81,82</td></tr>
<tr><td>Axis5</td><td>82,83</td><td>83,84</td></tr>
</table>
</td></tr>
</table>

Sample:

Set Value

Request					
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**	
1~247	0x10	72	12	24	
Register Value1	**Register Value2**	**Register Value3**	**Register Value4**	**Register Value5**	**Register Value6**
Axis0 EncPos		Axis1 EncPos		Axis2 EncPos	
LSW	MSW	LSW	MSW	LSW	MSW
Register Value7	**Register Value8**	**Register Value9**	**Register Value10**	**Register Value11**	**Register Value12**
Axis3 EncPos		Axis4 EncPos		Axis5 EncPos	
LSW	MSW	LSW	MSW	LSW	MSW

Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x10	72	12

Get Value

Request			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x03	72	12

Response					
Slave Address	**Function Code**	**Byte Count**			
1~247	0x03	24			
Register Value1	**Register Value2**	**Register Value3**	**Register Value4**	**Register Value5**	**Register Value6**
Axis0 EncPos		Axis1 EncPos		Axis2 EncPos	
LSW	MSW	LSW	LSW	MSW	LSW
Register Value7	**Register Value8**	**Register Value9**	**Register Value10**	**Register Value11**	**Register Value12**
Axis3 EncPos		Axis4 EncPos		Axis5 EncPos	
LSW	MSW	LSW	LSW	MSW	LSW

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

3.2.5 Enable Ring Counter

Description:

Set the maximum ring counter value for both the encoder and command pulse counter. The function will be enabled when the value is set.

Parameters:

Name	Type	Description	Address
RingValue	R/W	The upper limit of the pulse counter value Range: 2 ~ 2147483647	Function Code: 0x03,0x04,0x06,0x10 This parameter is 32 bit integer data, it takes two registers. Starting Address:

Axis	Base 0	Base 1
Axis0	120,121	121,122
Axis1	122,123	123,124
Axis2	124,125	125,126
Axis3	126,127	127,128
Axis4	128,129	129,130
Axis5	130,131	131,132

Sample:

Set Value

Request					
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**	
1~247	0x10	120	12	24	
Register Value1	**Register Value2**	**Register Value3**	**Register Value4**	**Register Value5**	**Register Value6**
Axis0 RingValue		Axis1 RingValue		Axis2 RingValue	
LSW	MSW	LSW	MSW	LSW	MSW
Register Value7	**Register Value8**	**Register Value9**	**Register Value10**	**Register Value11**	**Register Value12**
Axis3 RingValue		Axis4 RingValue		Axis5 RingValue	
LSW	MSW	LSW	MSW	LSW	MSW

Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x10	120	12

Get Value

Request			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x03	120	12

Response					
Slave Address	**Function Code**	**Byte Count**			
1~247	0x03	24			
Register Value1	**Register Value2**	**Register Value3**	**Register Value4**	**Register Value5**	**Register Value6**
Axis0 RingValue		Axis1 RingValue		Axis2 RingValue	
LSW	MSW	LSW	LSW	MSW	LSW
Register Value7	**Register Value8**	**Register Value9**	**Register Value10**	**Register Value11**	**Register Value12**
Axis3 RingValue		Axis4 RingValue		Axis5 RingValue	
LSW	MSW	LSW	LSW	MSW	LSW

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

Remarks:

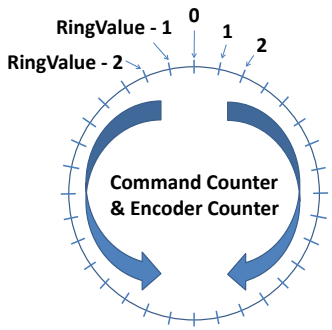


Figure 3.1 Ring Counter

3.2.6 Disable Ring Counter

Description:

Disable the ring counter function.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Address</th></tr>
<tr><td>SubCode</td><td>W</td><td>0x0016</td><td rowspan="2">Function Code: 0x10

Starting Address: 1280 (Base 0)</td></tr>
<tr><td>AxisNo</td><td>W</td><td>Axis definition
<table>
<tr><th>Axis</th><th>Value</th></tr>
<tr><td>AXIS_0</td><td>0x01</td></tr>
<tr><td>AXIS_1</td><td>0x02</td></tr>
<tr><td>AXIS_2</td><td>0x04</td></tr>
<tr><td>AXIS_3</td><td>0x08</td></tr>
<tr><td>AXIS_4</td><td>0x10</td></tr>
<tr><td>AXIS_5</td><td>0x20</td></tr>
</table></td></tr>
</table>

Sample:

Execute Command

Request				
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**
1~247	0x10	1280	2	4
Register Value1 (SubCode)	**Register Value2**			
0x0016	AxisNo			

Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x10	1280	2

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

4 Auto Homing

Name	Type*	Function Code	Description
Auto Homing Related Parameters			
Homing Settings	W	SubCode	Set auto homing search parameters.
Start Auto Homing			
Start Homing	W	SubCode	Start the homing procedure.

* R: read, W: write

4.1 Auto Homing Related Parameters

4.1.1 Homing Settings

Description:

Set auto homing search parameters.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Address</th></tr>
<tr><td>SubCode</td><td>W</td><td>0x0009</td><td rowspan="8">Function Code: 0x10

Starting Address: 1280 (Base 0)</td></tr>
<tr><td>AxisNo</td><td>W</td><td>Axis definition
<table>
<tr><th>Axis</th><th>Value</th></tr>
<tr><td>AXIS_0</td><td>0x01</td></tr>
<tr><td>AXIS_1</td><td>0x02</td></tr>
<tr><td>AXIS_2</td><td>0x04</td></tr>
<tr><td>AXIS_3</td><td>0x08</td></tr>
<tr><td>AXIS_4</td><td>0x10</td></tr>
<tr><td>AXIS_5</td><td>0x20</td></tr>
</table></td></tr>
<tr><td>HomeLogic</td><td>W</td><td>Active logic level of "HOME"
0: Low active
1: High active</td></tr>
<tr><td>SLDLogic</td><td>W</td><td>Active logic level of "SLD"
0: Low active
1: High active</td></tr>
<tr><td>EZLogic</td><td>W</td><td>Active logic level of "EZ"
0: Low active
1: High active</td></tr>
<tr><td>HomeSteps</td><td>W</td><td>Specify whether operation of each step id executed

If bit is zero: Non-execution
If bit is one: Execution

<table>
<tr><th>Bit</th><th>Step</th></tr>
<tr><td>Bit 1</td><td>Step1:High-speed near home search (use the SLD signal)</td></tr>
<tr><td>Bit 2</td><td>Reserved</td></tr>
<tr><td>Bit 3</td><td>Step2: Low-speed home search (use the HOME signal)</td></tr>
<tr><td>Bit 4</td><td>Reserved</td></tr>
<tr><td>Bit 5</td><td>Step3: Low-speed Z-</td></tr>
</table></td></tr>
</table>

			phase search (use the EZ signal)	
		Bit 6	Reserved	
		Bit 7	Step4: High-speed offset drive	
		Bit 8	Reserved	
Step4Offset	W	Offset of step4 using		

Sample:

Execute Command

Request					
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**	
1~247	0x10	1280	7	14	
Register Value1 (SubCode)	**Register Value2**	**Register Value3**	**Register Value4**	**Register Value5**	**Register Value6**
0x0009	AxisNo	HomeLogic	SLDLogic	EZLogic	HomeSteps
Register Value7					
Step4Offset					

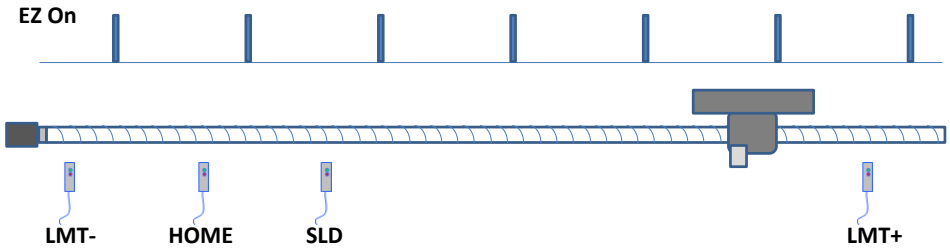
Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x10	1280	7

Exception Code:

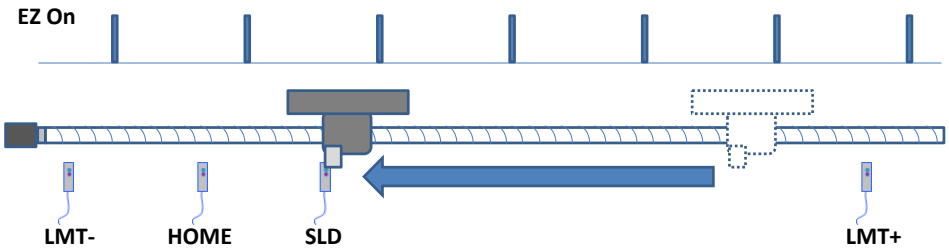
If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

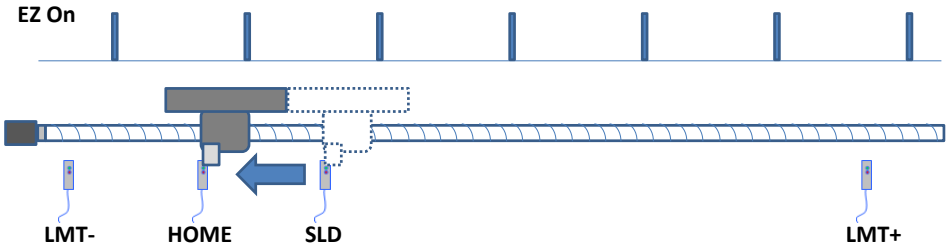
Remarks:



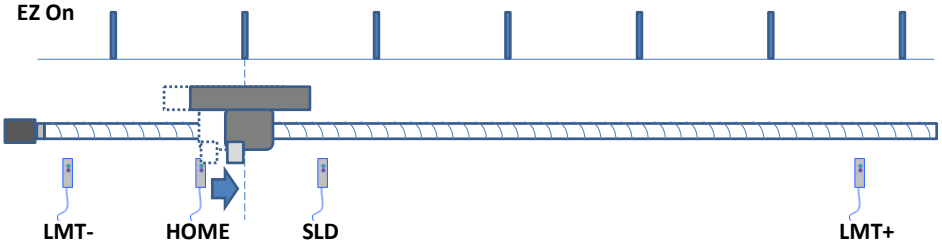
(a) Step 0: Wait auto homing command.



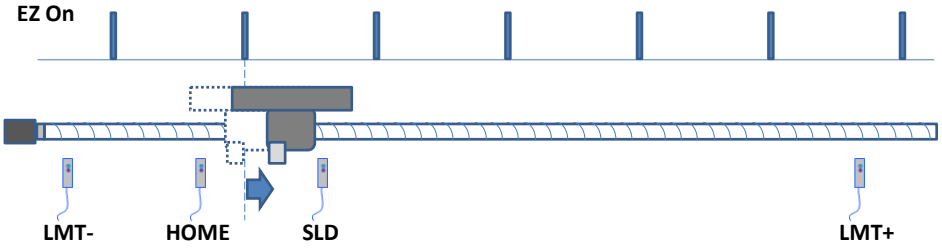
(b) Step 1: Search SLD signal (high speed).



(c) Step 2: Search HOME signal (low speed).



(d) Step 3: Search EZ signal (low speed).



(e) Step 4: Offset (high speed).

Figure 4.1 Four Step Homing

4.2 Start Auto Homing

4.2.1 Start Homing

Description:

Start the homing procedure.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Address</th></tr>
<tr><td>SubCode</td><td>W</td><td>0x000A</td><td rowspan="10">Function Code: 0x10

Starting Address: 1280 (Base 0)

The parameters marked with "*" are 32 bit integer data, they take two registers.</td></tr>
<tr><td>AxisNo</td><td>W</td><td>Axis definition
<table>
<tr><th>Axis</th><th>Value</th></tr>
<tr><td>AXIS_0</td><td>0x01</td></tr>
<tr><td>AXIS_1</td><td>0x02</td></tr>
<tr><td>AXIS_2</td><td>0x04</td></tr>
<tr><td>AXIS_3</td><td>0x08</td></tr>
<tr><td>AXIS_4</td><td>0x10</td></tr>
<tr><td>AXIS_5</td><td>0x20</td></tr>
</table></td></tr>
<tr><td>StartSpeed*</td><td>W</td><td>Start speed (PPS)</td></tr>
<tr><td>Acceleration*</td><td>W</td><td>Acceleration (PPS/Sec)</td></tr>
<tr><td>Deceleration*</td><td>W</td><td>Deceleration (PPS/Sec)</td></tr>
<tr><td>HighSpeed*</td><td>W</td><td>Near home search (step 1) and offset drive (step 4) speed (PPS)</td></tr>
<tr><td>LowSpeed*</td><td>W</td><td>Home search (step 2) and Z-phase (EZ) search (step 3) speed</td></tr>
<tr><td>HomingDir</td><td>W</td><td>Home search direction
0: Negative
1: Positive</td></tr>
</table>

Sample:

Execute Command

Request					
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**	
1~247	0x10	1280	13	26	
Register Value1 (SubCode)	**Register Value2**	**Register Value3**	**Register Value4**	**Register Value5**	**Register Value6**
0x000A	AxisNo	StartSpeed		Acceleration	
		LSW	MSW	LSW	MSW
Register Value7	**Register Value8**	**Register Value9**	**Register Value10**	**Register Value11**	**Register Value12**
Deceleration		HighSpeed		LowSpeed	
LSW	MSW	LSW	MSW	LSW	MSW
Register Value13					
HomingDir					

Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x10	1280	13

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

5 Motion Control

Name	Type*	Function Code	Description
Current Motion Status			
Motion Status	R	0x04	Get the current motion status of the axis.
Current Speed	R	0x04	Get the current speed of the axis.
Current Acceleration	R	0x04	Get the current acceleration of the axis.
Single Axis Motiom			
Single Axis Motion: P to P	W	SubCode	Begin a single axis move.
Single Axis Motion: Velocity	W	SubCode	Begin a single axis continuous command pulse moving.
Linear Interpolation Motion			
Linear Interpolation Motion: Two-Axis	W	SubCode	Begin a linear interpolation move for any two axes.
Linear Interpolation Motion: Three-Axis	W	SubCode	Begin a linear interpolation move for any three axes.
Linear Interpolation Motion: Multi-Axis	W	SubCode	Begin a linear interpolation move for any number of axes.
Arc Interpolation Motion			
Arc Interpolation Motion	W	SubCode	Begin a circular interpolation move for any two axes.
Helical Interpolation Motion			
Helical Interpolation Motion	W	SubCode	Begin a helical interpolation move for any three axes.
Continuous Interpolation Motion Setting			
Continuous Interpolation Motion	W	SubCode	Assigns axes to an interpolation group and sets the axes group to continuous interpolation mode.
Abort Motion			
Stop Motion	W	SubCode	Stop motion.
Enable Softlimit	W	SubCode	Enable software limit function.
Disable Softlimit	W	SubCode	Disable software limit function.
Synchronous Motion			
Hold Motion Commands	W	SubCode	Set the specified axes in holding mode.
Release Motion Commands	W	SubCode	Terminates the axes hold operation. The reserved motion commands are executed.

* R: read, W: write

5.1 Current Motion Status

5.1.1 Motion Status

Description:

Get the current motion status of the axis.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Address</th></tr>
<tr><td>Done</td><td>R</td><td>0: A motion command is being executed (axis is outputting command pulse)
1: Motion has finished</td><td>Function Code: 0x04

Starting Address:
<table>
<tr><th>Axis</th><th>Base 0</th><th>Base 1</th></tr>
<tr><td>Axis0</td><td>42</td><td>43</td></tr>
<tr><td>Axis1</td><td>43</td><td>44</td></tr>
<tr><td>Axis2</td><td>44</td><td>45</td></tr>
<tr><td>Axis3</td><td>45</td><td>46</td></tr>
<tr><td>Axis4</td><td>46</td><td>47</td></tr>
<tr><td>Axis5</td><td>47</td><td>48</td></tr>
</table></td></tr>
<tr><td>StopStatus</td><td>R</td><td>Indicates the cause of a motion stop:

0: Then motion command is still running
Other: See Table 5.1</td><td>Function Code: 0x04

Starting Address:
<table>
<tr><th>Axis</th><th>Base 0</th><th>Base 1</th></tr>
<tr><td>Axis0</td><td>48</td><td>49</td></tr>
<tr><td>Axis1</td><td>49</td><td>50</td></tr>
<tr><td>Axis2</td><td>50</td><td>51</td></tr>
<tr><td>Axis3</td><td>51</td><td>52</td></tr>
<tr><td>Axis4</td><td>52</td><td>53</td></tr>
<tr><td>Axis5</td><td>53</td><td>54</td></tr>
</table></td></tr>
</table>

Table5.1 Motion Stop Cause

Bit	Cause
Bit 0	Command has reached the target position
Bit 1	Auto homing has finished
Bit 2	Motion command has been interrupted by a stop command
Bit 3	The axis finished outputting command pulse and waits for the INP signal to trigger
Bit 4	The positive direction software limit has been activated
Bit 5	The negative direction software limit has been activated
Bit 6	The LMT+ signal has been activated

Bit 7	The LMT- signal has been activated
Bit 8	The ALM signal has been activated
Bit 9	The EMG signal has been activated
Bit 10 ~ Bit 15	Reserved

Sample:

Get Status

Request			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x04	42	6

Response					
Slave Address	**Function Code**	**Byte Count**			
1~247	0x04	12			
Register Value1	**Register Value2**	**Register Value3**	**Register Value4**	**Register Value5**	**Register Value6**
Axis0 Done	Axis1 Done	Axis2 Done	Axis3 Done	Axis4 Done	Axis5 Done

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

5.1.2 Current Speed

Description:

Get the current speed of the axis.

Parameters:

Name	Type	Description	Address
Speed	R	Current Speed (PPS)	Function Code: 0x04 This parameter is 32 bit integer data, it takes two registers. Starting Address:

Axis	Base 0	Base 1
Axis0	84,85	85,86
Axis1	86,87	87,88
Axis2	88,89	89,90
Axis3	90,91	91,92
Axis4	92,93	93,94
Axis5	94,95	95,96

Sample:

Get Status

Request			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x04	84	12

Response					
Slave Address	**Function Code**	**Byte Count**			
1~247	0x04	24			
Register Value1	**Register Value2**	**Register Value3**	**Register Value4**	**Register Value5**	**Register Value6**
Axis0 Speed		Axis1 Speed		Axis2 Speed	
LSW	MSW	LSW	MSW	LSW	MSW
Register Value7	**Register Value8**	**Register Value9**	**Register Value10**	**Register Value11**	**Register Value12**
Axis3 Speed		Axis4 Speed		Axis5 Speed	
LSW	MSW	LSW	MSW	LSW	MSW

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

5.1.3 Current Acceleration

Description:

Get the current acceleration of the axis.

Parameters:

Name	Type	Description	Address
Acc	R	Current Acceleration (PPS/Sec)	Function Code: 0x04 This parameter is 32 bit integer data, it takes two registers. Starting Address:

Axis	Base 0	Base 1
Axis0	96,97	97,98
Axis1	98,99	99,100
Axis2	100,101	101,102
Axis3	102,103	103,104
Axis4	104,105	105,106
Axis5	106,107	107,108

Sample:

Get Status

Request			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x04	96	12

Response					
Slave Address	**Function Code**	**Byte Count**			
1~247	0x04	24			
Register Value1	**Register Value2**	**Register Value3**	**Register Value4**	**Register Value5**	**Register Value6**
Axis0 Acc		Axis1 Acc		Axis2 Acc	
LSW	MSW	LSW	MSW	LSW	MSW
Register Value7	**Register Value8**	**Register Value9**	**Register Value10**	**Register Value11**	**Register Value12**
Axis3 Acc		Axis4 Acc		Axis5 Acc	
LSW	MSW	LSW	MSW	LSW	MSW

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

5.2 Single Axis Motiom

5.2.1 Single Axis Motion: P to P

Description:

Begin a single axis move.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Address</th></tr>
<tr><td>SubCode</td><td>W</td><td><table><tr><th>Type</th><th>Value</th></tr><tr><td>Relative, T-Curve</td><td>0x001E</td></tr><tr><td>Absolute , T-Curve</td><td>0x001F</td></tr><tr><td>Relative, S-Curve</td><td>0x0020</td></tr><tr><td>Absolute , S-Curve</td><td>0x0021</td></tr></table></td><td rowspan="8">Function Code: 0x10

Starting Address: 1280 (Base 0)

The parameters marked with "*" are 32 bit integer data, they take two registers.</td></tr>
<tr><td>AxisNo</td><td>W</td><td>Axis definition<table><tr><th>Axis</th><th>Value</th></tr><tr><td>AXIS_0</td><td>0x01</td></tr><tr><td>AXIS_1</td><td>0x02</td></tr><tr><td>AXIS_2</td><td>0x04</td></tr><tr><td>AXIS_3</td><td>0x08</td></tr><tr><td>AXIS_4</td><td>0x10</td></tr><tr><td>AXIS_5</td><td>0x20</td></tr></table></td></tr>
<tr><td>StartSpeed*</td><td>W</td><td>Start speed(PPS)</td></tr>
<tr><td>DriveSpeed*</td><td>W</td><td>Drive speed (PPS)</td></tr>
<tr><td>EndSpeed*</td><td>W</td><td>End speed (PPS)</td></tr>
<tr><td>Acceleration*</td><td>W</td><td>Acceleration(PPS/Sec)</td></tr>
<tr><td>Deceleration*</td><td>W</td><td>Deceleration (PPS/Sec)</td></tr>
<tr><td>Pulse*</td><td>W</td><td>Position/Distance (Pulse)
> 0: driving in positive direction
< 0: driving in negative direction</td></tr>
</table>

Sample:

Execute Command

Request					
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**	
1~247	0x10	1280	14	28	
Register Value1 (SubCode)	**Register Value2**	**Register Value3**	**Register Value4**	**Register Value5**	**Register Value6**
0x001E ~ 0x0021	AxisNo	StartSpeed		DriveSpeed	
		LSW	MSW	LSW	MSW
Register Value7	**Register Value8**	**Register Value9**	**Register Value10**	**Register Value11**	**Register Value12**
EndSpeed		Acceleration		Deceleration	
LSW	MSW	LSW	MSW	LSW	MSW
Register Value13	**Register Value14**				
Pulse					
LSW	MSW				

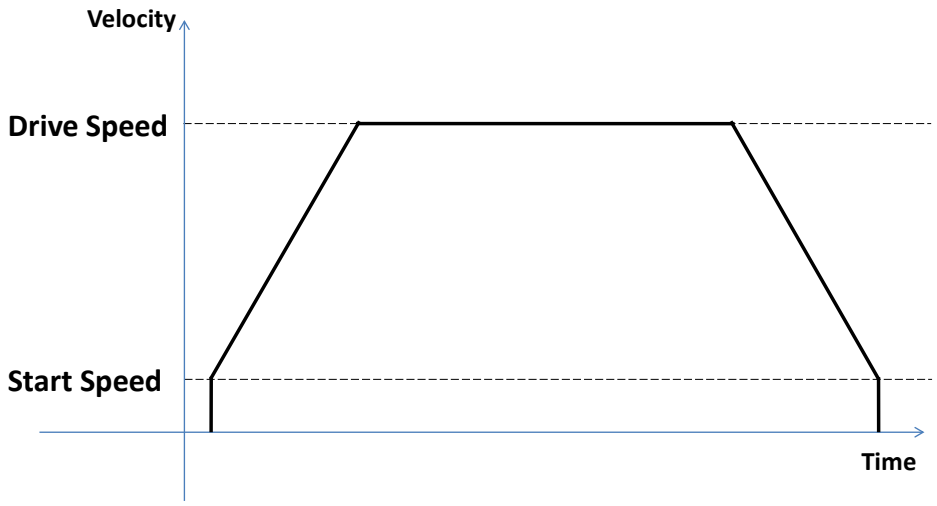
Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x10	1280	14

Exception Code:

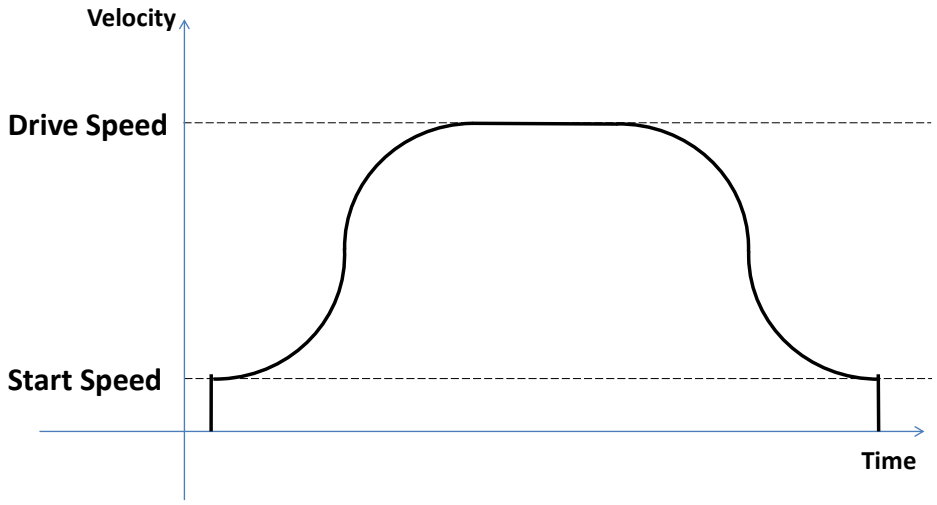
If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

Remarks:



(a) T-Curve Velocity Profile



(b) S-Curve Velocity Profile

Figure 5.1 Velocity Profile

5.2.2 Single Axis Motion: Velocity

Description:

Begin a single axis continuous command pulse move.

Parameters:

Name	Type	Description	Address
SubCode	W	0x001B	Function Code: 0x10 Starting Address: 1280 (Base 0) The parameters marked with "*" are 32 bit integer data, they take two registers.
AxisNo	W	Axis definition **Axis** / **Value** AXIS_0 0x01 AXIS_1 0x02 AXIS_2 0x04 AXIS_3 0x08 AXIS_4 0x10 AXIS_5 0x20	
StartSpeed*	W	Start speed (PPS)	
DriveSpeed*	W	Drive speed (PPS)	
Acceleration*	W	Acceleration (PPS/Sec)	
Direction	W	0: driving in negative direction 1: driving in positive direction	

Sample:

Execute Command

Request					
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**	
1~247	0x10	1280	9	18	
Register Value1 (SubCode)	**Register Value2**	**Register Value3**	**Register Value4**	**Register Value5**	**Register Value6**
0x001B	AxisNo	StartSpeed		DriveSpeed	
		LSW	MSW	LSW	MSW
Register Value7	**Register Value8**	**Register Value9**			
Acceleration		Direction			
LSW	MSW				

Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x10	1280	9

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

5.3 Linear Interpolation Motion

5.3.1 Linear Interpolation Motion: Two-Axis

Description:

Begin a linear interpolation move for any two axes.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Address</th></tr>
<tr><td>SubCode</td><td>W</td><td><table><tr><th>Type</th><th>Value</th></tr><tr><td>Relative, T-Curve</td><td>0x0022</td></tr><tr><td>Absolute , T-Curve</td><td>0x0023</td></tr><tr><td>Relative, S-Curve</td><td>0x0024</td></tr><tr><td>Absolute , S-Curve</td><td>0x0025</td></tr></table></td><td rowspan="11">Function Code: 0x10

Starting Address: 1280 (Base 0)

The parameters marked with "*" are 32 bit integer data, they take two registers.</td></tr>
<tr><td>MainAxis</td><td>W</td><td rowspan="2">Axis definition<table><tr><th>Axis</th><th>Value</th></tr><tr><td>AXIS_0</td><td>0x01</td></tr><tr><td>AXIS_1</td><td>0x02</td></tr><tr><td>AXIS_2</td><td>0x04</td></tr><tr><td>AXIS_3</td><td>0x08</td></tr><tr><td>AXIS_4</td><td>0x10</td></tr><tr><td>AXIS_5</td><td>0x20</td></tr></table></td></tr>
<tr><td>SlaveAxis</td><td>W</td></tr>
<tr><td>StartSpeed*</td><td>W</td><td>Start speed(PPS)</td></tr>
<tr><td>DriveSpeed*</td><td>W</td><td>Drive speed (PPS)</td></tr>
<tr><td>EndSpeed*</td><td>W</td><td>End speed (PPS)</td></tr>
<tr><td>Acceleration*</td><td>W</td><td>Acceleration(PPS/Sec)</td></tr>
<tr><td>Deceleration*</td><td>W</td><td>Deceleration (PPS/Sec)</td></tr>
<tr><td>MainAxisPulse*</td><td>W</td><td rowspan="2">Position/Distance (Pulse)
> 0: driving in positive direction
< 0: driving in negative direction</td></tr>
<tr><td>SlaveAxisPulse*</td><td>W</td></tr>
<tr><td>InterpMode</td><td>W</td><td>Command execution mode:

0: General
No command buffering takes place. A new command can only be executed if the previous command has finished

1: Continuous
Command buffering; up to 5000</td></tr>
</table>

		command can be stored. Use this mode to generate a continuous motion path.	

Sample:

Execute Command

Request					
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**	
1~247	0x10	1280	18	36	
Register Value1 (SubCode)	**Register Value2**	**Register Value3**	**Register Value4**	**Register Value5**	
0x0022 ~ 0x0025	MainAxis	SlaveAxis	StartSpeed		
			LSW	MSW	
Register Value6	**Register Value7**	**Register Value8**	**Register Value9**	**Register Value10**	**Register Value11**
DriveSpeed		EndSpeed		Acceleration	
LSW	MSW	LSW	MSW	LSW	MSW
Register Value12	**Register Value13**	**Register Value14**	**Register Value15**	**Register Value16**	**Register Value17**
Deceleration		MainAxisPulse		SlaveAxisPulse	
LSW	MSW	LSW	MSW	LSW	MSW
Register Value18					
InterpMode					

Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x10	1280	18

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

5.3.2 Linear Interpolation Motion: Three-Axis

Description:

Begin a linear interpolation move for any three axes.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Address</th></tr>
<tr><td>SubCode</td><td>W</td><td><table><tr><th>Type</th><th>Value</th></tr><tr><td>Relative, T-Curve</td><td>0x0026</td></tr><tr><td>Absolute , T-Curve</td><td>0x0027</td></tr><tr><td>Relative, S-Curve</td><td>0x0028</td></tr><tr><td>Absolute , S-Curve</td><td>0x0029</td></tr></table></td><td rowspan="15">Function Code: 0x10

Starting Address: 1280 (Base 0)

The parameters marked with "*" are 32 bit integer data, they take two registers.</td></tr>
<tr><td>MainAxis</td><td>W</td><td rowspan="3">Axis definition<table><tr><th>Axis</th><th>Value</th></tr><tr><td>AXIS_0</td><td>0x01</td></tr><tr><td>AXIS_1</td><td>0x02</td></tr><tr><td>AXIS_2</td><td>0x04</td></tr><tr><td>AXIS_3</td><td>0x08</td></tr><tr><td>AXIS_4</td><td>0x10</td></tr><tr><td>AXIS_5</td><td>0x20</td></tr></table></td></tr>
<tr><td>SecondAxis</td><td>W</td></tr>
<tr><td>ThirdAxis</td><td>W</td></tr>
<tr><td>StartSpeed*</td><td>W</td><td>Start speed (PPS)</td></tr>
<tr><td>DriveSpeed*</td><td>W</td><td>Drive speed (PPS)</td></tr>
<tr><td>EndSpeed*</td><td>W</td><td>End speed (PPS)</td></tr>
<tr><td>Acceleration*</td><td>W</td><td>Acceleration (PPS/Sec)</td></tr>
<tr><td>Deceleration*</td><td>W</td><td>Deceleration (PPS/Sec)</td></tr>
<tr><td>MainAxisPulse*</td><td>W</td><td rowspan="3">Position/Distance (Pulse)
> 0: driving in positive direction
< 0: driving in negative direction</td></tr>
<tr><td>SecondAxisPulse*</td><td>W</td></tr>
<tr><td>ThirdAxisPulse*</td><td>W</td></tr>
<tr><td>InterpMode</td><td>W</td><td>Command execution mode:

0: General
No command buffering takes place. A new command can only be executed if the previous command has finished

1: Continuous</td></tr>
</table>

		Command buffering; up to 5000 command can be stored. Use this mode to generate a continuous motion path.	

Sample:

Execute Command

Request					
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**	
1~247	0x10	1280	21	42	
Register Value1 (SubCode)	**Register Value2**	**Register Value3**	**Register Value4**	**Register Value5**	**Register Value6**
0x0026 ~ 0x0029	MainAxis	SecondAxis	ThirdAxis	StartSpeed	
				LSW	MSW
Register Value7	**Register Value8**	**Register Value9**	**Register Value10**	**Register Value11**	**Register Value12**
DriveSpeed		EndSpeed		Acceleration	
LSW	MSW	LSW	MSW	LSW	MSW
Register Value13	**Register Value14**	**Register Value15**	**Register Value16**	**Register Value17**	**Register Value18**
Deceleration		MainAxisPulse		SecondAxisPulse	
LSW	MSW	LSW	MSW	LSW	MSW
Register Value19	**Register Value20**	**Register Value21**			
ThirdAxisPulse		InterpMode			
LSW	MSW				

Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x10	1280	21

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

5.3.3 Linear Interpolation Motion: Multi-Axis

Description:

Begin a linear interpolation move for any number of axes.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Address</th></tr>
<tr><td>SubCode</td><td>W</td><td><table><tr><th>Type</th><th>Value</th></tr><tr><td>Relative</td><td>0x002A</td></tr><tr><td>Absolute</td><td>0x002B</td></tr></table></td><td rowspan="15">Function Code: 0x10

Starting Address: 1280 (Base 0)

The parameters marked with "*" are 32 bit integer data, they take two registers.</td></tr>
<tr><td>BitMultiAxes</td><td>W</td><td>Axis definition (multiple choice)<table><tr><th>Bit</th><th>Axis</th></tr><tr><td>Bit 0</td><td>AXIS_0</td></tr><tr><td>Bit 1</td><td>AXIS_1</td></tr><tr><td>Bit 2</td><td>AXIS_2</td></tr><tr><td>Bit 3</td><td>AXIS_3</td></tr><tr><td>Bit 4</td><td>AXIS_4</td></tr><tr><td>Bit 5</td><td>AXIS_5</td></tr><tr><td>Bit 6 ~ 15</td><td>Reserved</td></tr></table></td></tr>
<tr><td>AccDecMode</td><td>W</td><td><table><tr><th>Curve</th><th>Value</th></tr><tr><td>T-Curve</td><td>0x6565</td></tr><tr><td>S-Curve</td><td>0x6666</td></tr></table></td></tr>
<tr><td>StartSpeed*</td><td>W</td><td>Start speed(PPS)</td></tr>
<tr><td>DriveSpeed*</td><td>W</td><td>Drive speed (PPS)</td></tr>
<tr><td>EndSpeed*</td><td>W</td><td>End speed (PPS)</td></tr>
<tr><td>Acceleration*</td><td>W</td><td>Acceleration(PPS/Sec)</td></tr>
<tr><td>Deceleration*</td><td>W</td><td>Deceleration (PPS/Sec)</td></tr>
<tr><td>Pulse[0]*</td><td>W</td><td rowspan="6">Each element contains the relative distance to move for the corresponding axis</td></tr>
<tr><td>Pulse[1]*</td><td>W</td></tr>
<tr><td>Pulse[2]*</td><td>W</td></tr>
<tr><td>Pulse[3]*</td><td>W</td></tr>
<tr><td>Pulse[4]*</td><td>W</td></tr>
<tr><td>Pulse[5]*</td><td>W</td></tr>
<tr><td>InterpMode</td><td>W</td><td>Command execution mode:

0: General
No command buffering takes place. A new command can only be executed if the previous command has finished

1: Continuous</td></tr>
</table>

		Command buffering; up to 5000 command can be stored. Use this mode to generate a continuous motion path.	

Sample:

Execute Command

Request					
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**	
1~247	0x10	1280	26	52	
Register Value1 (SubCode)	**Register Value2**	**Register Value3**	**Register Value4**	**Register Value5**	
0x002A ~ 0x002B	BitMultiAxes	AccDecMode	StartSpeed		
			LSW	MSW	
Register Value6	**Register Value7**	**Register Value8**	**Register Value9**	**Register Value10**	**Register Value11**
DriveSpeed		EndSpeed		Acceleration	
LSW	MSW	LSW	MSW	LSW	MSW
Register Value12	**Register Value13**	**Register Value14**	**Register Value15**	**Register Value16**	**Register Value17**
Deceleration		Pulse[0]		Pulse[1]	
LSW	MSW	LSW	MSW	LSW	MSW
Register Value18	**Register Value19**	**Register Value20**	**Register Value21**	**Register Value22**	**Register Value23**
Pulse[2]		Pulse[3]		Pulse[4]	
LSW	MSW	LSW	MSW	LSW	MSW
Register Value24	**Register Value25**	**Register Value26**			
Pulse[5]		InterpMode			
LSW	MSW				

Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x10	1280	26

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

5.4 Arc Interpolation Motion

5.4.1 Arc Interpolation Motion

Description:

Begin a circular interpolation move for any two axes.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Address</th></tr>
<tr><td>SubCode</td><td>W</td><td><table><tr><th>Type</th><th>Value</th></tr><tr><td>Relative</td><td>0x002C</td></tr><tr><td>Absolute</td><td>0x002D</td></tr></table></td><td rowspan="12">Function Code:
0x10

Starting Address:
1280 (Base 0)

The parameters marked with "*" are 32 bit integer data, they take two registers.</td></tr>
<tr><td>MainAxis</td><td>W</td><td rowspan="2">Axis definition<table><tr><th>Axis</th><th>Value</th></tr><tr><td>AXIS_0</td><td>0x01</td></tr><tr><td>AXIS_1</td><td>0x02</td></tr><tr><td>AXIS_2</td><td>0x04</td></tr><tr><td>AXIS_3</td><td>0x08</td></tr><tr><td>AXIS_4</td><td>0x10</td></tr><tr><td>AXIS_5</td><td>0x20</td></tr></table></td></tr>
<tr><td>SlaveAxis</td><td>W</td></tr>
<tr><td>StartSpeed*</td><td>W</td><td>Start speed (PPS)</td></tr>
<tr><td>DriveSpeed*</td><td>W</td><td>Drive speed (PPS)</td></tr>
<tr><td>EndSpeed*</td><td>W</td><td>End speed (PPS)</td></tr>
<tr><td>Acceleration*</td><td>W</td><td>Acceleration (PPS/Sec)</td></tr>
<tr><td>Deceleration*</td><td>W</td><td>Deceleration (PPS/Sec)</td></tr>
<tr><td>ArcDirection</td><td>W</td><td>Direction of rotation
0: CW
1: CCW</td></tr>
<tr><td>MainAxisCenterPoint*</td><td>W</td><td>Center point of the main axis (Pulse)</td></tr>
<tr><td>SlaveAxisCenterPoint*</td><td>W</td><td>Center point of the slave axis (Pulse)</td></tr>
<tr><td>MainAxisFinishPoint*</td><td>W</td><td>End point of the main axis (Pulse)</td></tr>
<tr><td>SlaveAxisFinishPoint*</td><td>W</td><td>End point of the slave axis (Pulse)</td></tr>
<tr><td>InterpMode</td><td>W</td><td>Command execution mode:

0: General
No command buffering takes place. A new command can only be executed if the previous command has finished</td></tr>
</table>

		1: Continuous Command buffering; up to 5000 command can be stored. Use this mode to generate a continuous motion path.	

Sample:

Execute Command

Request					
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**	
1~247	0x10	1280	23	46	
Register Value1 (SubCode)	**Register Value2**	**Register Value3**	**Register Value4**	**Register Value5**	
0x002C ~ 0x002D	MainAxis	SlaveAxis	StartSpeed		
			LSW	MSW	
Register Value6	**Register Value7**	**Register Value8**	**Register Value9**	**Register Value10**	**Register Value11**
DriveSpeed		EndSpeed		Acceleration	
LSW	MSW	LSW	MSW	LSW	MSW
Register Value12	**Register Value13**	**Register Value14**	**Register Value15**	**Register Value16**	
Deceleration		ArcDirection	MainAxisCenterPoint		
LSW	MSW		LSW	MSW	
Register Value17	**Register Value18**	**Register Value19**	**Register Value20**	**Register Value21**	**Register Value22**
SlaveAxisCenterPoint		MainAxisFinishPoint		SlaveAxisFinishPoint	
LSW	MSW	LSW	MSW	LSW	MSW
Register Value23					
InterpMode					

Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x10	1280	23

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

Remarks:

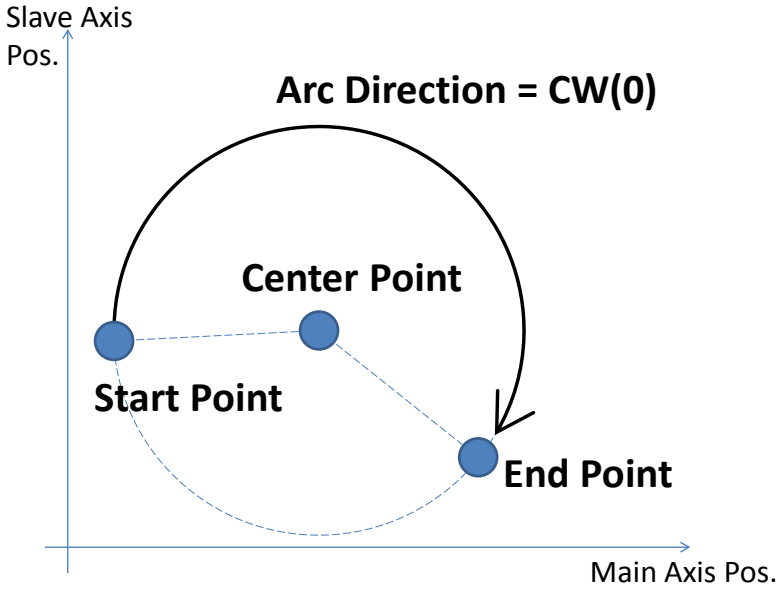


Figure 5.2 Circular Interpolation

5.5 Helical Interpolation Motion

5.5.1 Helical Interpolation Motion

Description:

Begin a helical interpolation move for any three axes.

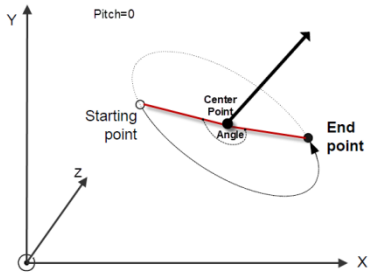
Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Address</th></tr>
<tr><td>SubCode</td><td>W</td><td><table><tr><th>Type</th><th>Value</th></tr><tr><td>Relative</td><td>0x002E</td></tr><tr><td>Absolute</td><td>0x002F</td></tr></table></td><td rowspan="14">Function Code: 0x10

Starting Address: 1280 (Base 0)

The parameters marked with "*" are 32 bit integer data, and marked with "**" are 32 bit floating point data. They take two registers.</td></tr>
<tr><td>Axis0</td><td>W</td><td rowspan="3">Axis definition<table><tr><th>Axis</th><th>Value</th></tr><tr><td>AXIS_0</td><td>0x01</td></tr><tr><td>AXIS_1</td><td>0x02</td></tr><tr><td>AXIS_2</td><td>0x04</td></tr><tr><td>AXIS_3</td><td>0x08</td></tr><tr><td>AXIS_4</td><td>0x10</td></tr><tr><td>AXIS_5</td><td>0x20</td></tr></table></td></tr>
<tr><td>Axis1</td><td>W</td></tr>
<tr><td>Axis2</td><td>W</td></tr>
<tr><td>StartSpeed*</td><td>W</td><td>Start speed (PPS)</td></tr>
<tr><td>DriveSpeed*</td><td>W</td><td>Drive speed (PPS)</td></tr>
<tr><td>EndSpeed*</td><td>W</td><td>End speed (PPS)</td></tr>
<tr><td>Acceleration*</td><td>W</td><td>Acceleration (PPS/Sec)</td></tr>
<tr><td>Deceleration*</td><td>W</td><td>Deceleration (PPS/Sec)</td></tr>
<tr><td>CenterPoint0*</td><td>W</td><td>Center point of the first axis (Pulse)</td></tr>
<tr><td>CenterPoint1*</td><td>W</td><td>Center point of the second axis (Pulse)</td></tr>
<tr><td>CenterPoint2*</td><td>W</td><td>Center point of the third axis (Pulse)</td></tr>
<tr><td>NormalDir0*</td><td>W</td><td rowspan="3">The parameters describe a three dimensional rotation vector (helical axis).

The rotation vector determines:
• The positive direction of the translational movement (pitch direction).
• The positive direction of rotation. The positive direction of rotation is being determined by the right-hand rule: the right thumb points</td></tr>
<tr><td>NormalDir1*</td><td>W</td></tr>
<tr><td>NormalDir2*</td><td>W</td></tr>
</table>

		along the positive direction of the rotation axis and the curl of your fingers represents the direction of rotation 	
Angle**	W	Rotation angle in degrees (360 indicates one full revolution, 720 will result in two full revolution, etc.) > 0: positive direction rotation vector (right-hand rule) < 0: negative direction of the rotation vector	
Pitch**	W	The relative distance to move along the helical axis after each full revolution. If this parameter is zero a three dimensional arc will be executed 	
InterpMode	W	Command execution mode: 0: General No command buffering takes place. A new command can only be executed if the previous command has finished 1: Continuous Command buffering; up to 5000 command can be stored. Use this mode to generate a continuous motion path.	

Sample:

Execute Command

Request					
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**	
1~247	0x10	1280	31	62	
Register Value1 (SubCode)	**Register Value2**	**Register Value3**	**Register Value4**	**Register Value5**	**Register Value6**
0x002E ~ 0x002F	Axis0	Axis1	Axis2	StartSpeed	
				LSW	MSW
Register Value7	**Register Value8**	**Register Value9**	**Register Value10**	**Register Value11**	**Register Value12**
DriveSpeed		EndSpeed		Acceleration	
LSW	MSW	LSW	MSW	LSW	MSW
Register Value13	**Register Value14**	**Register Value15**	**Register Value16**	**Register Value17**	**Register Value18**
Deceleration		CenterPoint0		CenterPoint1	
LSW	MSW	LSW	MSW	LSW	MSW
Register Value19	**Register Value20**	**Register Value21**	**Register Value22**	**Register Value23**	**Register Value24**
CenterPoint2		NormalDir0		NormalDir1	
LSW	MSW	LSW	MSW	LSW	MSW
Register Value25	**Register Value26**	**Register Value27**	**Register Value28**	**Register Value29**	**Register Value30**
NormalDir2		Angle		Pitch	
LSW	MSW	LSW	MSW	LSW	MSW
Register Value31					
InterpMode					

Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x10	1280	31

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

5.6 Continuous Interpolation Motion Setting

5.6.1 Continuous Interpolation Motion

Description:

Assigns axes to an interpolation group and sets the axes group to continuous interpolation mode. Once the group has switch to continuous mode, all the arriving commands are being treated as continuous interpolation commands.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Address</th></tr>
<tr><td>SubCode</td><td>W</td><td>0x0018</td><td rowspan="5">Function Code: 0x10

Starting Address:
1280 (Base 0)</td></tr>
<tr><td>Enable</td><td>W</td><td>0: Disable
1: Enable</td></tr>
<tr><td>GroupIndex0</td><td>W</td><td rowspan="3">Select the axes which belong to the interpolation group
<table>
<tr><th>Axis</th><th>Value</th></tr>
<tr><td>AXIS_0</td><td>0x01</td></tr>
<tr><td>AXIS_1</td><td>0x02</td></tr>
<tr><td>AXIS_2</td><td>0x04</td></tr>
<tr><td>AXIS_3</td><td>0x08</td></tr>
<tr><td>AXIS_4</td><td>0x10</td></tr>
<tr><td>AXIS_5</td><td>0x20</td></tr>
</table></td></tr>
<tr><td>GroupIndex1</td><td>W</td></tr>
<tr><td>GroupIndex2</td><td>W</td></tr>
</table>

Sample:

Execute Command

Request				
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**
1~247	0x10	1280	5	10
Register Value1 (SubCode)	**Register Value2**	**Register Value3**	**Register Value4**	**Register Value5**
0x0018	Enable	GroupIndex0	GroupIndex1	GroupIndex2

Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x10	1280	5

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

5.7 Abort Motion

5.7.1 Stop Motion

Description:

Stop motion.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Address</th></tr>
<tr><td>SubCode</td><td>W</td><td>0x0037</td><td rowspan="3">Function Code: 0x10

Starting Address: 1280 (Base 0)</td></tr>
<tr><td>AxisNo</td><td>W</td><td>Axis definition
<table>
<tr><th>Axis</th><th>Value</th></tr>
<tr><td>AXIS_0</td><td>0x01</td></tr>
<tr><td>AXIS_1</td><td>0x02</td></tr>
<tr><td>AXIS_2</td><td>0x04</td></tr>
<tr><td>AXIS_3</td><td>0x08</td></tr>
<tr><td>AXIS_4</td><td>0x10</td></tr>
<tr><td>AXIS_5</td><td>0x20</td></tr>
</table></td></tr>
<tr><td>StopMode</td><td>W</td><td>1: Decelerating stop
2: Sudden stop</td></tr>
</table>

Sample:

Execute Command

Request				
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**
1~247	0x10	1280	3	6
Register Value1 (SubCode)	**Register Value2**	**Register Value3**		
0x0037	AxisNo	StopMode		

Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x10	1280	3

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

5.7.2 Enable Softlimit

Description:

Enable software limit function.

Parameters:

Name	Type	Description	Address
SubCode	W	0x0007	Function Code: 0x10 Starting Address: 1280 (Base 0) The parameters marked with "*" are 32 bit integer data, they take two registers.
AxisNo	W	Axis definition **Axis** / **Value** AXIS_0 0x01 AXIS_1 0x02 AXIS_2 0x04 AXIS_3 0x08 AXIS_4 0x10 AXIS_5 0x20	
StopMode	W	1: Decelerating stop 2: Sudden stop	
RefSource	W	Reference source 1: Command pulse counter 2: Encoder pulse counter	
LimitPositive*	W	Positive direction soft limit	
LimitNegative*	W	Negative direction soft limit	

Sample:

Execute Command

Request					
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**	
1~247	0x10	1280	8	16	
Register Value1 (SubCode)	**Register Value2**	**Register Value3**	**Register Value4**	**Register Value5**	**Register Value6**
0x0007	AxisNo	StopMode	RefSource	LimitPositive	
				LSW	MSW
Register Value7	**Register Value8**				
LimitNegative					
LSW	MSW				

Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x10	1280	8

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

5.7.3 Disable Softlimit

Description:

Disable software limit function.

Parameters:

Name	Type	Description	Address
SubCode	W	0x0008	Function Code: 0x10 Starting Address: 1280 (Base 0)
AxisNo	W	Axis definition **Axis** / **Value** AXIS_0 0x01 AXIS_1 0x02 AXIS_2 0x04 AXIS_3 0x08 AXIS_4 0x10 AXIS_5 0x20	

Sample:

Execute Command

Request				
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**
1~247	0x10	1280	2	4
Register Value1 (SubCode)	**Register Value2**			
0x0008	AxisNo			

Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x10	1280	2

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

5.8 Synchronous Motion

5.8.1 Hold Motion Commands

Description:

Set the specified axes in holding mode. The execution of the next motion command will be retained until the release command is executed.

Parameters:

Name	Type	Description	Address
SubCode	W	0x0035	Function Code: 0x10 Starting Address: 1280 (Base 0)
BitMultiAxes	W	Axis definition (multiple choice) **Axis** / **Value** AXIS_0 / 0x01 AXIS_1 / 0x02 AXIS_2 / 0x04 AXIS_3 / 0x08 AXIS_4 / 0x10 AXIS_5 / 0x20	

Sample:

Execute Command

Request				
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**
1~247	0x10	1280	2	4
Register Value1 (SubCode)	**Register Value2**			
0x0035	BitMultiAxes			

Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x10	1280	2

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

5.8.2 Release Motion Commands

Description:

Terminates the axes hold operation. The reserved motion commands are executed.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Address</th></tr>
<tr><td>SubCode</td><td>W</td><td>0x0036</td><td rowspan="2">Function Code:
0x10

Starting Address:
1280 (Base 0)</td></tr>
<tr><td>BitMultiAxes</td><td>W</td><td>Axis definition (multiple choice)
<table>
<tr><th>Axis</th><th>Value</th></tr>
<tr><td>AXIS_0</td><td>0x01</td></tr>
<tr><td>AXIS_1</td><td>0x02</td></tr>
<tr><td>AXIS_2</td><td>0x04</td></tr>
<tr><td>AXIS_3</td><td>0x08</td></tr>
<tr><td>AXIS_4</td><td>0x10</td></tr>
<tr><td>AXIS_5</td><td>0x20</td></tr>
</table></td></tr>
</table>

Sample:

Execute Command

Request				
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**
1~247	0x10	1280	2	4
Register Value1 (SubCode)	**Register Value2**			
0x0036	BitMultiAxes			

Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x10	1280	2

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

6 Other Application

Name	Type*	Function Code	Description
Compare Trigger Operation			
Compare Trigger Settings	W	SubCode	Configure the compare trigger function. The compare function outputs a signal ("CMP" signal) when the compare condition has been met.
Latch Operation			
Latch Signal Setting	W	SubCode	Configure the position latch function. When the "LTC" signal is active the encoder pulse counter value can be latched in the same time instantly by hardware.
Latch Data	R	0x04	Get latched encoder pulse counter data.

* R: read, W: write

6.1 Compare Trigger Operation

6.1.1 Compare Trigger Settings

Description:

Configure the compare trigger function. The compare function outputs a signal ("CMP" signal) when the compare condition has been met.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Address</th></tr>
<tr><td>SubCode</td><td>W</td><td>0x0017</td><td rowspan="11">Function Code: 0x10

Starting Address: 1280 (Base 0)

The parameters marked with "*" are 32 bit integer data, they take two registers.</td></tr>
<tr><td>AxisNo</td><td>W</td><td>Axis definition
<table>
<tr><th>Axis</th><th>Value</th></tr>
<tr><td>AXIS_0</td><td>0x01</td></tr>
<tr><td>AXIS_1</td><td>0x02</td></tr>
<tr><td>AXIS_2</td><td>0x04</td></tr>
<tr><td>AXIS_3</td><td>0x08</td></tr>
<tr><td>AXIS_4</td><td>0x10</td></tr>
<tr><td>AXIS_5</td><td>0x20</td></tr>
</table></td></tr>
<tr><td>CmpTrigEnable</td><td>W</td><td>Enable compare trigger function
0: Disable
1: Enable</td></tr>
<tr><td>OutputLogic</td><td>W</td><td>Active logic level of "CMP"
0: Low active
1: High active</td></tr>
<tr><td>PulseWidth</td><td>W</td><td>Pulse width of trigger signals, see Table 6.1</td></tr>
<tr><td>MoveDir</td><td>W</td><td>Axis moving direction
0: Negative
1: Positive</td></tr>
<tr><td>CmpIncEnable</td><td>W</td><td>Enable auto-increment compare function

0: Disable (triggers only one output signal)
1: Enable (set the compare trigger to continuously trigger a output signal at equidistant position)</td></tr>
<tr><td>ConstPitch</td><td>W</td><td>The auto-increment distance</td></tr>
<tr><td>CmpData*</td><td>W</td><td>The first position at which the compare function will trigger an output signal. (If "CmpIncEnable" is disabled then one output signal will be triggered at the "CmpData" position)</td></tr>
</table>

Sample:

Execute Command

Request					
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**	
1~247	0x10	1280	10	20	
Register Value1 (SubCode)	**Register Value2**	**Register Value3**	**Register Value4**	**Register Value5**	**Register Value6**
0x0017	AxisNo	CmpTrigEnable	OutputLogic	PulseWidth	MoveDir
Register Value7	**Register Value8**	**Register Value9**	**Register Value10**		
CmpIncEnable	ConstPitch	CmpData			
		LSW	MSW		

Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x10	1280	10

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

Remarks:

Table 6.1 Pulse Width Setting

Pulse Width	**Minimum Trigger Period ***	**Value**
160ns~320ns	640ns	0x01
320ns~640ns	1.28us	0x02
640ns~1.28us	2.56us	0x03
1.28us~2.56us	5.12us	0x04
2.56us~5.12us	10.24us	0x05
5.12us~10.24us	20.48us	0x06
10.24us~20.48us	40.96us	0x07
20.48us~40.96us	81.92us	0x08
40.96us~81.92us	163.84us	0x09
81.92us~163.84us	327.68us	0x0A
163.84us~327.68us	655.36us	0x0B
327.68us~655.36us	1.31072ms	0x0C
655.36us~1.31072ms	2.62144ms	0x0D
1.31072ms~2.62144ms	5.24288ms	0x0E
2.62144ms~5.24288ms	10.48576ms	0x0F

* Please make sure the trigger signal period is bigger than minimum period.

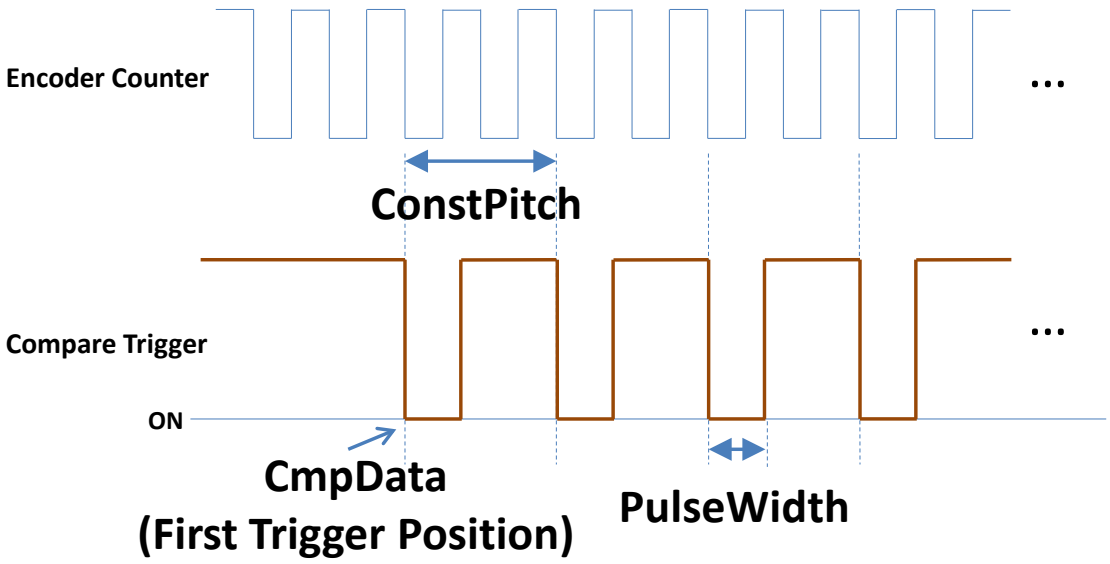


Figure 6.1 Auto-increment Compare Function

6.2 Latch Operation

6.2.1 Latch Signal Settings

Description:

Configure the position latch function. When the "LTC" signal is active the encoder pulse counter value can be latched in the same time instantly by hardware.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Address</th></tr>
<tr><td>SubCode</td><td>W</td><td>0x0038</td><td rowspan="4">Function Code: 0x10

Starting Address: 1280
(Base 0)</td></tr>
<tr><td>AxisNo</td><td>W</td><td>Axis definition
<table>
<tr><th>Axis</th><th>Value</th></tr>
<tr><td>AXIS_0</td><td>0x01</td></tr>
<tr><td>AXIS_1</td><td>0x02</td></tr>
<tr><td>AXIS_2</td><td>0x04</td></tr>
<tr><td>AXIS_3</td><td>0x08</td></tr>
<tr><td>AXIS_4</td><td>0x10</td></tr>
<tr><td>AXIS_5</td><td>0x20</td></tr>
</table></td></tr>
<tr><td>Enable</td><td>W</td><td>0: Disable the signal
1: Enable the signal</td></tr>
<tr><td>Logic</td><td>W</td><td>Active logic level
0: Low active
1: High active</td></tr>
</table>

Sample:

Execute Command

Request				
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**
1~247	0x10	1280	4	8
Register Value1 (SubCode)	**Register Value2**	**Register Value3**	**Register Value4**	
0x0038	AxisNo	Enable	Logic	

Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x10	1280	4

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

6.2.2 Latch Data

Description:

Get latched encoder pulse counter data.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Address</th></tr>
<tr><td>LatchData</td><td>R</td><td>Latched data</td><td>Function Code: 0x04

This parameter is 32 bit integer data, it takes two registers.

Starting Address:
<table>
<tr><th>Axis</th><th>Base 0</th><th>Base 1</th></tr>
<tr><td>Axis0</td><td>108,109</td><td>109,110</td></tr>
<tr><td>Axis1</td><td>110,111</td><td>111,112</td></tr>
<tr><td>Axis2</td><td>112,113</td><td>113,114</td></tr>
<tr><td>Axis3</td><td>114,115</td><td>115,116</td></tr>
<tr><td>Axis4</td><td>116,117</td><td>117,118</td></tr>
<tr><td>Axis5</td><td>118,119</td><td>119,120</td></tr>
</table></td></tr>
</table>

Sample:

Get Value

Request			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x04	108	12

Response					
Slave Address	**Function Code**	**Byte Count**			
1~247	0x04	24			
Register Value1	**Register Value2**	**Register Value3**	**Register Value4**	**Register Value5**	**Register Value6**
Axis0 LatchData		Axis1 LatchData		Axis2 LatchData	
LSW	MSW	LSW	MSW	LSW	MSW
Register Value7	**Register Value8**	**Register Value9**	**Register Value10**	**Register Value11**	**Register Value12**
Axis3 LatchData		Axis4 LatchData		Axis5 LatchData	
LSW	MSW	LSW	MSW	LSW	MSW

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

7 GPIO and FRnet

Name	Type*	Function Code		Description
General Purpose I/O				
General DO	R/W	OnOff	0x01, 0x02, 0x05, 0x0F	Set/Get the state of general purpose digital output channel.
		MultiGdoStatus	0x03, 0x04, 0x06, 0x10	
General DI	R	OnOff	0x02	Get the state of general purpose digital input channel.
		MultiGdiStatus	0x04	
FRnet I/O				
FRnet DO	R/W	OnOff	0x01, 0x02, 0x05, 0x0F	Set/Get the state of a remote FRnet DO module.
		DOStatus	0x03, 0x04, 0x06, 0x10	
FRnet DI	R	OnOff	0x02	Get the state of a remote FRnet DI module.
		DIStatus	0x04	
		DIModules	0x04	

* R: read, W: write

7.1 General Purpose I/O

7.1.1 General DO

Description:

Set/Get the state of general purpose digital output channel.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Address</th></tr>
<tr><td>OnOff</td><td>R/W</td><td>DO channel state
0: OFF
1: ON</td><td>Function Code: 0x01, 0x02, 0x05, 0x0F

Starting Address:
<table><tr><th>Channel</th><th>Base 0</th><th>Base 1</th></tr><tr><td>GDO_0</td><td>264</td><td>265</td></tr><tr><td>GDO _1</td><td>265</td><td>266</td></tr><tr><td>GDO _2</td><td>266</td><td>267</td></tr></table></td></tr>
<tr><td>MultiGdoStatus</td><td>R/W</td><td>Each bit represents a DO channel state
<table><tr><th>Bit</th><th>Channel</th></tr><tr><td>Bit 0</td><td>GDO_0</td></tr><tr><td>Bit 1</td><td>GDO_1</td></tr><tr><td>Bit 2</td><td>GDO_2</td></tr><tr><td>Bit 3~15</td><td>Reserved</td></tr></table>If bit is zero: the corresponding signal is OFF
If bit is one: the corresponding signal is ON</td><td>Function Code: 0x03, 0x04, 0x06, 0x10

Starting Address: 27 (Base 0)</td></tr>
</table>

Sample:

Set Status

Request					
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**	**Register Value1**
1~247	0x10	27	1	2	MultiGdoStatus

Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x10	27	1

Get Status

Request			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x03	27	1

Response			
Slave Address	**Function Code**	**Byte Count**	**Register Value1**
1~247	0x03	2	MultiGdoStatus

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

7.1.2 General DI

Description:

Get the state of general purpose digital input channel.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Address</th></tr>
<tr><td>OnOff</td><td>R</td><td>DI channel state
0: OFF
1: ON</td><td>Function Code: 0x02

Starting Address:
<table>
<tr><th>Channel</th><th>Base 0</th><th>Base 1</th></tr>
<tr><td>GDI_0</td><td>267</td><td>268</td></tr>
<tr><td>GDI _1</td><td>268</td><td>269</td></tr>
<tr><td>GDI _2</td><td>269</td><td>270</td></tr>
<tr><td>GDI _3</td><td>270</td><td>271</td></tr>
<tr><td>GDI _4</td><td>271</td><td>272</td></tr>
<tr><td>GDI _5</td><td>272</td><td>273</td></tr>
<tr><td>GDI _6</td><td>273</td><td>274</td></tr>
<tr><td>GDI _7</td><td>274</td><td>275</td></tr>
<tr><td>GDI _8</td><td>275</td><td>276</td></tr>
<tr><td>GDI _9</td><td>276</td><td>277</td></tr>
<tr><td>GDI _10</td><td>277</td><td>278</td></tr>
<tr><td>GDI _11</td><td>278</td><td>279</td></tr>
</table></td></tr>
<tr><td>MultiGdiStatus</td><td>R</td><td>Each bit represents a DI channel state
<table>
<tr><th>Bit</th><th>Channel</th></tr>
<tr><td>Bit 0</td><td>GDI_0</td></tr>
<tr><td>Bit 1</td><td>GDI_1</td></tr>
<tr><td>Bit 2</td><td>GDI_2</td></tr>
<tr><td>Bit 3</td><td>GDI_3</td></tr>
<tr><td>Bit 4</td><td>GDI_4</td></tr>
<tr><td>Bit 5</td><td>GDI_5</td></tr>
<tr><td>Bit 6</td><td>GDI_6</td></tr>
<tr><td>Bit 7</td><td>GDI_7</td></tr>
<tr><td>Bit 8</td><td>GDI_8</td></tr>
<tr><td>Bit 9</td><td>GDI_9</td></tr>
<tr><td>Bit 10</td><td>GDI_10</td></tr>
<tr><td>Bit 11</td><td>GDI_11</td></tr>
<tr><td>Bit 12~15</td><td>Reserved</td></tr>
</table>
If bit is zero: the corresponding signal is OFF
If bit is one: the corresponding signal is ON</td><td>Function Code: 0x04

Starting Address: 28 (Base 0)</td></tr>
</table>

Sample:

Get Status

Request			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x04	28	1

Response			
Slave Address	**Function Code**	**Byte Count**	**Register Value1**
1~247	0x04	2	MultiGdiStatus

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

7.2 FRnet I/O

7.2.1 FRnet DO

Description:

Set/Get the state of a remote FRnet DO module.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Address</th></tr>
<tr><td>OnOff</td><td>R/W</td><td>DO channel state
0: OFF
1: ON</td><td>Function Code: 0x01, 0x02, 0x05, 0x0F

Starting Address:
<table>
<tr><th>Channel</th><th>Base 0</th><th>Base 1</th></tr>
<tr><td>Group0 CH0~15</td><td>0~15</td><td>1~16</td></tr>
<tr><td>Group1 CH0~15</td><td>16~31</td><td>17~32</td></tr>
<tr><td>Group2 CH0~15</td><td>32~47</td><td>33~48</td></tr>
<tr><td>Group3 CH0~15</td><td>48~63</td><td>49~64</td></tr>
<tr><td>Group4 CH0~15</td><td>64~79</td><td>65~80</td></tr>
<tr><td>Group5 CH0~15</td><td>80~95</td><td>81~96</td></tr>
<tr><td>Group6 CH0~15</td><td>96~111</td><td>97~112</td></tr>
<tr><td>Group7 CH0~15</td><td>112~127</td><td>113~128</td></tr>
</table></td></tr>
<tr><td>DOStatus</td><td>R/W</td><td>Each bit represents a DO channel state
<table>
<tr><th>Bit</th><th>Channel</th></tr>
<tr><td>Bit 0</td><td>CH_0</td></tr>
<tr><td>Bit 1</td><td>CH_1</td></tr>
<tr><td>…</td><td>…</td></tr>
<tr><td>Bit 14</td><td>CH_14</td></tr>
<tr><td>Bit 15</td><td>CH_15</td></tr>
</table>
If bit is zero: the corresponding signal is OFF
If bit is one: the corresponding signal is ON</td><td>Function Code: 0x03, 0x04, 0x06, 0x10

Starting Address:
<table>
<tr><th>Channel</th><th>Base 0</th><th>Base 1</th></tr>
<tr><td>Group0</td><td>11</td><td>12</td></tr>
<tr><td>Group1</td><td>12</td><td>13</td></tr>
<tr><td>Group2</td><td>13</td><td>14</td></tr>
<tr><td>Group3</td><td>14</td><td>15</td></tr>
<tr><td>Group4</td><td>15</td><td>16</td></tr>
<tr><td>Group5</td><td>16</td><td>17</td></tr>
<tr><td>Group6</td><td>17</td><td>18</td></tr>
<tr><td>Group7</td><td>18</td><td>19</td></tr>
</table></td></tr>
</table>

Sample:

Set Status

Request					
Slave Address	**Function Code**	**Starting Address**	**Quantity**	**Byte Count**	
1~247	0x10	11	8	16	
Register Value1	**Register Value2**	**Register Value3**	**Register Value4**	**Register Value5**	**Register Value6**
Group0 DOStatus	Group1 DOStatus	Group2 DOStatus	Group3 DOStatus	Group4 DOStatus	Group5 DOStatus
Register Value7	**Register Value8**				
Group6 DOStatus	Group7 DOStatus				

Response			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x10	11	8

Get Status

Request			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x03	11	8

Response					
Slave Address	**Function Code**	**Byte Count**	**Register Value1**	**Register Value2**	**Register Value3**
1~247	0x03	16	Group0 DOStatus	Group1 DOStatus	Group2 DOStatus
Register Value4	**Register Value5**	**Register Value6**	**Register Value7**	**Register Value8**	
Group3 DOStatus	Group4 DOStatus	Group5 DOStatus	Group6 DOStatus	Group7 DOStatus	

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

7.2.2 FRnet DI

Description:

Get the state of a remote FRnet DI module.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Address</th></tr>
<tr><td>OnOff</td><td>R</td><td>DI channel state
0: OFF
1: ON</td><td>Function Code: 0x02

Starting Address:
<table>
<tr><th>Channel</th><th>Base 0</th><th>Base 1</th></tr>
<tr><td>Group8 CH0~15</td><td>128~143</td><td>129~144</td></tr>
<tr><td>Group9 CH0~15</td><td>144~159</td><td>145~160</td></tr>
<tr><td>Group10 CH0~15</td><td>160~175</td><td>161~176</td></tr>
<tr><td>Group11 CH0~15</td><td>176~191</td><td>177~192</td></tr>
<tr><td>Group12 CH0~15</td><td>192~207</td><td>193~208</td></tr>
<tr><td>Group13 CH0~15</td><td>208~223</td><td>209~224</td></tr>
<tr><td>Group14 CH0~15</td><td>224~239</td><td>225~240</td></tr>
<tr><td>Group15 CH0~15</td><td>240~255</td><td>241~256</td></tr>
</table></td></tr>
<tr><td>DIStatus</td><td>R</td><td>Each bit represents a DI channel state
<table>
<tr><th>Bit</th><th>Channel</th></tr>
<tr><td>Bit 0</td><td>CH_0</td></tr>
<tr><td>Bit 1</td><td>CH_1</td></tr>
<tr><td>…</td><td>…</td></tr>
<tr><td>Bit 14</td><td>CH_14</td></tr>
<tr><td>Bit 15</td><td>CH_15</td></tr>
</table>
If bit is zero: the corresponding signal is OFF
If bit is one: the corresponding signal is ON</td><td>Function Code: 0x04

Starting Address:
<table>
<tr><th>Group</th><th>Base 0</th><th>Base 1</th></tr>
<tr><td>Group8</td><td>19</td><td>20</td></tr>
<tr><td>Group9</td><td>20</td><td>21</td></tr>
<tr><td>Group10</td><td>21</td><td>22</td></tr>
<tr><td>Group11</td><td>22</td><td>23</td></tr>
<tr><td>Group12</td><td>23</td><td>24</td></tr>
<tr><td>Group13</td><td>24</td><td>25</td></tr>
<tr><td>Group14</td><td>25</td><td>26</td></tr>
<tr><td>Group15</td><td>26</td><td>27</td></tr>
</table></td></tr>
<tr><td>DIModules</td><td>R</td><td>Each bit represents an FRnet DI module</td><td>Function Code: 0x04

Starting Address:10 (Base 0)</td></tr>
</table>

		<table><tr><th>Bit</th><th>Group</th></tr><tr><td>Bit 0</td><td>Group8</td></tr><tr><td>Bit 1</td><td>Group9</td></tr><tr><td>Bit 2</td><td>Group10</td></tr><tr><td>Bit 3</td><td>Group11</td></tr><tr><td>Bit 4</td><td>Group12</td></tr><tr><td>Bit 5</td><td>Group13</td></tr><tr><td>Bit 6</td><td>Group14</td></tr><tr><td>Bit 7</td><td>Group15</td></tr><tr><td>Bit 8 ~ 15</td><td>Reserved</td></tr></table> If bit is one: the corresponding FRnet DI module is connected	

Sample:

Get Active DI Module Number

Request			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x04	10	1

Response			
Slave Address	**Function Code**	**Byte Count**	**Register Value1**
1~247	0x04	2	DIModules

Get Status

Request			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x04	19	8

Response					
Slave Address	**Function Code**	**Byte Count**	**Register Value1**	**Register Value2**	**Register Value3**
1~247	0x04	16	Group8 DIStatus	Group9 DIStatus	Group10 DIStatus
Register Value4	**Register Value5**	**Register Value6**	**Register Value7**	**Register Value8**	
Group11 DIStatus	Group12 DIStatus	Group13 DIStatus	Group14 DIStatus	Group15 DIStatus	

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

8 Version

Name	Type*	Function Code	Description
RS-M8196F Version			
Firmware Version	R	0x04	Get RS-M8196F firmware version.
I-8196F Version	R	0x04	Get I-8196F versions.

* R: read, W: write

8.1 RS-M8196F Version

8.1.1 Firmware Version

Description:

Get RS-M8196F firmware version.

Parameters:

Name	Type	Description	Address
Ver	R	Firmware version Format: 0xXXYY XX: Major version number YY: Minor version number Example: 0x0100 indicates the version number is 1.0	Function Code: 0x04 Starting Address: 0 (Base 0)

Sample:

Get Version

Request			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x04	0	1

Response			
Slave Address	**Function Code**	**Byte Count**	**Register Value1**
1~247	0x04	2	Ver

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

8.2 I-8196F Version

8.2.1 I-8196F Version

Description:

Get I-8196F versions.

Parameters:

Name	Type	Description	Address
FWVer	R	Firmware version	Function Code: 0x04 Starting Address: 2 (Base 0)
DSPVer	R	DSP firmware version	Function Code: 0x04 Starting Address: 3 (Base 0)
PCBVer	R	PCB version	Function Code: 0x04 Starting Address: 4 (Base 0)
PLDVer	R	PLD version	Function Code: 0x04 Starting Address: 5 (Base 0)
FPGAVer	R	FPGA version	Function Code: 0x04 Starting Address: 6 (Base 0)

Sample:

Get Version

Request			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x04	2	5

Response			
Slave Address	**Function Code**	**Byte Count**	**Register Value1**
1~247	0x04	10	FWVer
Register Value2	**Register Value3**	**Register Value4**	**Register Value5**
DSPVer	PCBVer	PLDVer	FPGAVer

Exception Code:

If you receive an exception code, see appendix "Error Codes".

Error		
Slave Address	**Function Code**	**Exception Code**
1~247	Function Code + 0x80	01~04

9 Appendix

9.1 Modbus Address

9.1.1 Coils: Function Code 0x01, 0x05, 0x0F

Address*	Variable	Comment
0~15	FRnet DO Group 0 Channel 0 ~15	FRnet DO Group Output Signal Status
16~31	FRnet DO Group 1 Channel 0 ~15	
32~47	FRnet DO Group 2 Channel 0 ~15	
48~63	FRnet DO Group 3 Channel 0 ~15	
64~79	FRnet DO Group 4 Channel 0 ~15	
80~95	FRnet DO Group 5 Channel 0 ~15	
96~111	FRnet DO Group 6 Channel 0 ~15	
112~127	FRnet DO Group 7 Channel 0 ~15	
128~263	Reserved	
264~266	General DO 0 ~ 2	General DO Signal Status
267~290	Reserved	
291~296	Axis 0~5 Servo On	Servo On Signal Status
297~302	Axis 0~5 ERC	ERC Signal Status
303~308	Axis 0~5 Alarm Reset	Alarm Reset Signal Status

*Base 0

9.1.2 Discrete Inputs: Function Code 0x02

Address*	Variable	Comment
0~15	FRnet DO Group 0 Channel 0 ~15	FRnet DO Group Output Signal Status
16~31	FRnet DO Group 1 Channel 0 ~15	
32~47	FRnet DO Group 2 Channel 0 ~15	
48~63	FRnet DO Group 3 Channel 0 ~15	
64~79	FRnet DO Group 4 Channel 0 ~15	
80~95	FRnet DO Group 5 Channel 0 ~15	
96~111	FRnet DO Group 6 Channel 0 ~15	
112~127	FRnet DO Group 7 Channel 0 ~15	
128~143	FRnet DI Group 8 Channel 0 ~15	FRnet DI Group Input Signal Status
144~159	FRnet DI Group 9 Channel 0 ~15	
160~175	FRnet DI Group 10 Channel 0 ~15	
176~191	FRnet DI Group 11 Channel 0 ~15	
192~207	FRnet DI Group 12 Channel 0 ~15	
208~223	FRnet DI Group 13 Channel 0 ~15	
224~239	FRnet DI Group 14 Channel 0 ~15	
240~255	FRnet DI Group 15 Channel 0 ~15	
264~266	General DO 0 ~ 2	General DO Signal Status
267~278	General DI 0 ~ 2	General DI Signal Status
279~290	Reserved	
291~296	Axis 0~5 Servo On	Servo On Signal Status
297~302	Axis 0~5 ERC	ERC Signal Status
303~308	Axis 0~5 Alarm Reset	Alarm Reset Signal Status

*Base 0

9.1.3 Holding Registers: Function Code 0x03, 0x06, 0x10

Address*	Variable	Comment
0~8	Reserved	
9	Word Order	Word Order Setting
10	Reserved	
11	FRnet DO Group 0	FRnet DO Group Output Signal Status
12	FRnet DO Group 1	
13	FRnet DO Group 2	
14	FRnet DO Group 3	
15	FRnet DO Group 4	
16	FRnet DO Group 5	
17	FRnet DO Group 6	
18	FRnet DO Group 7	
19~26	Reserved	
27	General DO	General DO Signal Status
28~59	Reserved	
60~61	Axis 0 Command Pulse Counter	Command Pulse Counter
62~63	Axis 1 Command Pulse Counter	
64~65	Axis 2 Command Pulse Counter	
66~67	Axis 3 Command Pulse Counter	
68~69	Axis 4 Command Pulse Counter	
70~71	Axis 5 Command Pulse Counter	
72~73	Axis 0 Encoder Pulse Counter	Encoder Pulse Counter
74~75	Axis 1 Encoder Pulse Counter	
76~77	Axis 2 Encoder Pulse Counter	
78~79	Axis 3 Encoder Pulse Counter	
80~81	Axis 4 Encoder Pulse Counter	
82~83	Axis 5 Encoder Pulse Counter	
84~119	Reserved	
120~121	Axis 0 Ring Counter	Ring Counter Setting

	Setting	
122~123	Axis 1 Ring Counter Setting	
124~125	Axis 2 Ring Counter Setting	
126~127	Axis 3 Ring Counter Setting	
128~129	Axis 4 Ring Counter Setting	
130~131	Axis 5 Ring Counter Setting	

*Base 0

9.1.4 Input Registers: Function Code 0x04

Address*	Variable	Comment
0	Version	RS-M8196F Firmware Version
1	Card Status	i-8196F Information
2	Firmware Version	
3	DSP Version	
4	PCB Version	
5	PLD Version	
6	FPGA Version	
7	Reserved	
8	Function Return Code	i-8196F API Return Code
9	Word Order	Word Order Setting
10	FRnet DI Modules	Exist FRnet DI Group Number
11	FRnet DO Group 0	FRnet DO Group Output Signal Status
12	FRnet DO Group 1	
13	FRnet DO Group 2	
14	FRnet DO Group 3	
15	FRnet DO Group 4	
16	FRnet DO Group 5	
17	FRnet DO Group 6	
18	FRnet DO Group 7	
19	FRnet DI Group 8	FRnet DI Group Input Signal Status
20	FRnet DI Group 9	
21	FRnet DI Group 10	
22	FRnet DI Group 11	
23	FRnet DI Group 12	
24	FRnet DI Group 13	
25	FRnet DI Group 14	
26	FRnet DI Group 15	
27	General DO	General DO Signal Status
28	General DI	General DI Signal Status
29~35	Reserved	
36	Axis 0 I/O Status	I/O Status
37	Axis 1 I/O Status	
38	Axis 2 I/O Status	
39	Axis 3 I/O Status	
40	Axis 4 I/O Status	
41	Axis 5 I/O Status	
42	Axis 0 Motion Done	Motion Status
43	Axis 1 Motion Done	
44	Axis 2 Motion Done	
45	Axis 3 Motion Done	

46	Axis 4 Motion Done	
47	Axis 5 Motion Done	
48	Axis 0 Stop Status	
49	Axis 1 Stop Status	
50	Axis 2 Stop Status	
51	Axis 3 Stop Status	
52	Axis 4 Stop Status	
53	Axis 5 Stop Status	
54~59	Reserved	
60~61	Axis 0 Command Pulse Counter	Command Pulse Counter
62~63	Axis 1 Command Pulse Counter	
64~65	Axis 2 Command Pulse Counter	
66~67	Axis 3 Command Pulse Counter	
68~69	Axis 4 Command Pulse Counter	
70~71	Axis 5 Command Pulse Counter	
72~73	Axis 0 Encoder Pulse Counter	Encoder Pulse Counter
74~75	Axis 1 Encoder Pulse Counter	
76~77	Axis 2 Encoder Pulse Counter	
78~79	Axis 3 Encoder Pulse Counter	
80~81	Axis 4 Encoder Pulse Counter	
82~83	Axis 5 Encoder Pulse Counter	
84~85	Axis 0 Current Speed	Current Speed
86~87	Axis 1 Current Speed	
88~89	Axis 2 Current Speed	
90~91	Axis 3 Current Speed	
92~93	Axis 4 Current Speed	
94~95	Axis 5 Current Speed	
96~97	Axis 0 Current Acceleration	Current Acceleration
98~99	Axis 1 Current Acceleration	
100~101	Axis 2 Current Acceleration	

102~103	Axis 3 Current Acceleration	
104~105	Axis 4 Current Acceleration	
106~107	Axis 5 Current Acceleration	
108~109	Axis 0 Latch Data	Latch Data
110~111	Axis 1 Latch Data	
112~113	Axis 2 Latch Data	
114~115	Axis 3 Latch Data	
116~117	Axis 4 Latch Data	
118~119	Axis 5 Latch Data	
120~121	Axis 0 Ring Counter Setting	Ring Counter Setting
122~123	Axis 1 Ring Counter Setting	
124~125	Axis 2 Ring Counter Setting	
126~127	Axis 3 Ring Counter Setting	
128~129	Axis 4 Ring Counter Setting	
130~131	Axis 5 Ring Counter Setting	

*Base 0

9.2 Error Codes

When you receive an exception code, you can use input register (function code 04, address 8) to read the function return code.

Get Function Return Code

Request			
Slave Address	**Function Code**	**Starting Address**	**Quantity**
1~247	0x04	8	1

Response			
Slave Address	**Function Code**	**Byte Count**	**Register Value1**
1~247	0x04	2	ReturnCode

Table 9.1 Function Error Codes

Code	Description
0	No error
-5121	Modbus RTU packet transmission error occurred. Make sure that the: (a) Communication is normal. (b) RS-M8196F firmware version supports this function.
-5122	
-5123	
1	Reserved
2	No I-8196F card detected in the slot.
3	Specified slot is not being supported.
4	An invalid parameter value has been passed to a function.
5	An invalid value has been passed to the driving speed parameter.
6	An invalid value has been passed to the start speed parameter.
7	An invalid value has been passed to the end speed parameter.
8	Reserved
9	An invalid value has been passed to the “move direction” parameter.
10	Internal error
11	Internal error
12	Internal error
13	Internal error
14	Internal error
15	Internal error
16	Internal error
17	Internal error
18	• Some parameters cannot be set if a motion command is still executing. • If the axis is in not in a continuous interpolation mode then the next motion command cannot be sent while the previous motion command is still executing.
19	The I-8196F card has not been registered and initialized yet.
20	An invalid value has been passed to the axis parameter.
21	Internal error.
22	Reserved
23	A continuous interpolation command has been sent while the axes

	are not in continuous interpolation mode.
24	Reserved
25	Reserved
26	Invalid address has been passed to the FRnet DI module address parameter.
27	Invalid value has been passed to the FRnet DO module address parameter.
28	Reserved
29	Reserved
30	Reserved
31	The called function cannot be called when the axes are in continuous interpolation mode.
32	Reserved
33	Internal error
34	Internal error
35	Emergency stop has been activated. No new motion commands can be executed while emergency stop is active.
36	A "non-continuous" interpolation command has been sent while the axes are "continuous interpolation" mode.
37	Reserved
38	Reserved
39	Reserved
40	Reserved
41	Reserved
42	Reserved
43	The value passed to the ring counter parameter is not being supported.
44	Compare trigger function is in use, can't use ring counter mode.
45	Internal error
46	Internal error
47	Resetting the DSP failed.
48	The DDA has already been set.
49	The value passed to the set position counter parameter exceeds the valid range. Valid range:-2147483646~ 2147483646.
50	The axes are in ring counter mode. Some functions are not supported in ring counter mode .
-32767	I8196_ERR_DSP_SYSTEM_ERROR
-1001	I8196_ERR_DSP_MOTIONPATH_FULL
-1002	I8196_ERR_DSP_REMAIN_STILL
-1003	I8196_ERR_DSP_DIMENSION_OUT_RANGE
-1004	I8196_ERR_DSP_SPEED_VALUE
-1005	I8196_ERR_DSP_SMALLSHAPE_ERROR
-1006	I8196_ERR_DSP_SAWSHAPE_ERROR
-1007	I8196_ERR_DSP_AXIS_INUSE
-1008	I8196_ERR_DSP_NORMALVECTOR
-1009	I8196_ERR_DSP_CIRCLE_RADIUS
-1011	I8196_ERR_DSP_MECH_PARA_VALUE
-1100	I8196_ERR_DSP_MOTIONPATH_ALREADY_FREE
-1201	I8196_ERR_DSP_EXCEED_SPEED_LIMIT
-1202	I8196_ERR_DSP_EXCEED_VRING_LIMIT
-1203	I8196_ERR_DSP_EXCEED_MAX_POSTION
-2001	I8196_ERR_DSP_NO_COMMAND
-2002	I8196_ERR_DSP_CRC_ERROR

-2003	I8196_ERR_DSP_UNKNOWN_COMMAND
-2004	I8196_ERR_DSP_MULTIBLOCK_CMD
-2005	I8196_ERR_DSP_ACC_TYPE
-2006	I8196_ERR_DSP_DEC_TYPE
-2007	I8196_ERR_DSP_CMD_NOT_ALLOWED
-2011	I8196_ERR_DSP_INHIBIT_BY_EMG
-2012	I8196_ERR_DSP_INHIBIT_BY_MPG_EMG
-2013	I8196_ERR_DSP_INHIBIT_BY_PEL
-2014	I8196_ERR_DSP_INHIBIT_BY_MEL
-2015	I8196_ERR_DSP_INHIBIT_BY_ALM
-2016	I8196_ERR_DSP_INHIBIT_BY_RDY
-2101	I8196_ERR_DSP_GROUP_OUT_RANGE
-2102	I8196_ERR_DSP_GROUP_ALREADY_FREE
-2103	I8196_ERR_DSP_GROUP_ASSIGNED
-2104	I8196_ERR_DSP_GROUP_INUSE
-2105	I8196_ERR_DSP_GROUP_NOT_INUSE
-2106	I8196_ERR_DSP_AXIS_ASSIGNED
-2201	I8196_ERR_DSP_CONTROL_MODE
-2301	I8196_ERR_DSP_BUFFER_FULL
-2302	I8196_ERR_DSP_BUFFER_INFO
-2303	I8196_ERR_DSP_BUFFER_TYPE
-2304	I8196_ERR_DSP_BUFFER_SIZE
-2305	I8196_ERR_DSP_BUFFER_INUSE
-2306	I8196_ERR_DSP_BUFFER_NOT_READY
-2401	I8196_ERR_DSP_STOP_BY_P_CHANGE
-2402	I8196_ERR_DSP_CLEAR_STOP
-2410	I8196_ERR_DSP_STOP_BY_GINP
-2430	I8196_ERR_DSP_STOP_BY_AXIS_IO
-2501	I8196_ERR_DSP_NO_AVAILABLE_MACRO
-2502	I8196_ERR_DSP_MACRO_INUSE
-2503	I8196_ERR_DSP_MACRO_EMPTY
-2601	I8196_ERR_DSP_HOMING_IN_PROGRESS
-3601	I8196_ERR_DSP_OUTPUT_SATURATION
-3611	I8196_ERR_DSP_ERR_MSG_BUF_EMPTY
-3612	I8196_ERR_DSP_ERR_MSG_BUF_OVERFLOW
-1	I8196_ERR_DSP_TIMEOUT_ERROR
-2	I8196_ERR_DSP_HW_ID_ERROR
-3	I8196_ERR_DSP_AXIS_OUT_RANGE
-4	I8196_ERR_DSP_ADDR_OUT_RANGE
-5	I8196_ERR_DSP_VALUE_OUT_RANGE
-6	I8196_ERR_DSP_FPGA_DL_FAILED
-101	I8196_ERR_DSP_DA_AUTO_UPDATE
-102	I8196_ERR_DSP_DA_BUSY
-201	I8196_ERR_DSP_CMP_INUSE
-32768	I8196_ERR_DSP_NOT_IMPLEMENT