UIROBOT

User Manual

UIM342C / UIM342S / UIM342XS
Motion Controller for Servo Stepper Motor
with CAN Interface

Intellectual Property Protection Statement

- UIROBOT products meet the specification contained in their particular UIROBOT user manual;
- UIROBOT is willing to work with the customer who is concerned about intellectual property protection;
- Any attempt to destroy the code protection function of UIROBOT products may be regarded as a violation of intellectual property protection laws and regulations. If this behavior leads to the acquisition of software or other results protected by intellectual property rights without UIROBOT's authorization, UIROBOT has the right to file a lawsuit to stop this behavior in accordance with the Act.

Disclaimer

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. UIROBOT MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE**.** UIROBOT disclaims all liability arising from this information and its use. Use of UIROBOT Products in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless UIROBOT from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any UIROBOT intellectual property rights.

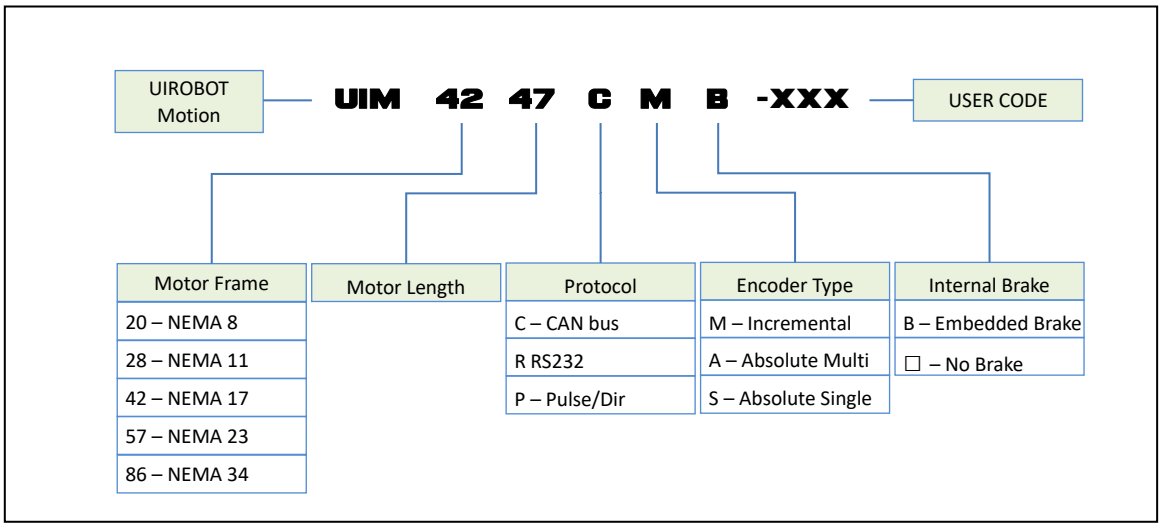
Trademark and Design Statement

- UIROBOT's name and logo is a registered trademark of UIROBOT in China and other countries or regions.
- The appearance designs of UIROBOT's products have applied for patent protection.

Contact information

United Intelligence Robotics (Zhejiang) Co., Ltd.
Address: Bldg B3, Innovation Park, Pinghu, Jiaxing City, Zhejiang Province, China 314200
Tel: +86 0573-85235951
Email: support@uirobot-zj.com
Web: www.uirobot.com

UIM342 Servo Stepper Motor Order Code



Revision History

Manual version	Revision date	change
V 1.0	November 1st, 2023	Initial version

UIM342C / UIM342S / UIM342XS
Motion Controller for Servo Stepper Motor
With CAN Bus Interface

The UIM342 series are advanced motion controllers designed for servo stepper motors, featuring a CAN Bus interface.

These controllers integrate several key functional modules: **Communication Unit** / **Motion Control Unit** / **Motor Driver** / **Feedback Control Unit** / **Input Logic Control** and **DSP System** (for processing instructions, replying, notifications, coordinating other functional modules, executing preloaded user programs, and providing real-time status updates)

The major control loop is executed within 1 millisecond, ensuring responsiveness.

Key Features:

- **High-Performance DSP System:** Executes all control loops within 1 millisecond.
- **Comprehensive SDK:** Includes SDK, DLL, LIB, and SO files.
- **Sample Codes:** Available in C++, C#, and other languages.
- **Cross-Platform Support:** Compatible with Linux, Windows 32-bit, and 64-bit systems.

Control System

- Robust DSP hardware
- Fault tolerance, fail safe user interface
- Input and event change notification

Motor Driver

- Wide Voltage Input: Operates on 24 ~ 48 VDC.(1)
- Adjustable Phase Current: Up to 8 A.(1)
- PWM Constant Current
- Micro-Stepping: 1 ~ 1/64 resolution
- No vibration or noise at low speeds.
- Protection: Overcurrent, overvoltage, overheating, etc.

Closed-loop Control

- High positioning accuracy, high response, Maximum speed up to 3000 rpm, it's depends on the Motor type
- Speed control, position control
- Backlash Compensation
- Stall detection

I/O Logic Control

- 3 digital inputs
- 1 output port (5V / 50mA)
- 3 trigger types (Continuous/Low-pass Filter/One-time)
- Input change notification
- 11 preset actions can be attached to the I/O port trigger

CAN Networking

- Active CAN 2.0, 1 Mbps Max.
- 1500V photoelectric isolation
- Support 3 types IDs:
  1) Node ID
  2) Group ID
  3) Global ID

Other features

- Emergency trigger lockout
- Integrated design with motor
- Aluminum alloy casing, sturdy and durable for easy heat dissipation
- Size as small as 20 mm x 20 mm x 16 mm

Note:
(1) Depending on the actual controller.

Contents

Safety

To prevent personal injury and property damage, please be sure to pay attention to the following before use:

	Precautions	Consequences of Neglect
⚠	Do not use it in humid , corrosive, flammable gas environments or places near flammable substances	Fire / Malfunction
⚠	Do not use the wire when it is soaked in oil/water	Fire / Malfunction
⚠	For motors with shaft keyway, do not touch the keyway with bare hands	Personal injury
⚠	Never touch the rotating parts of a running motor	Personal injury
⚠	Do not touch the running motor, it may be very hot	Personal injury
⚠	Do not cause the wires to be damaged or subjected to excessive external force, heavy pressure, or clamping	Fire / Malfunction
⚠	Do not hold the cable or motor shaft when transporting	Personal injury / Malfunction
⚠	Never hit the motor	Malfunction
⚠	Do not frequently power on / off	Malfunction
⚠	Never modify, disassemble or repair by yourself	Fire / Malfunction / Personal injury
⚠	Power supply voltage must meet the product requirements, and the Power supply current must be 1.5 times larger than the product requirements	Malfunction
⚠	Cut off the power when not in use for a long time	Fire / Malfunction / Personal injury

Maintenance

Please perform regular maintenance and inspection on the controller for safe use. Please pay attention to the following during maintenance and inspection:

① When performing the insulation test on the drive, be sure to disconnect all connections.
② Do not use gasoline, thinner, alcohol, acidic and alkaline cleaning agents to avoid damage to the casing.

Daily inspections and periodic inspections should be carried out according to the following items.

Type	Period	Check Item
Daily Inspection	Every day	• Confirm the operating temperature and humidity • No foreign matter entering • Abnormal vibration, sound and odor • Abnormal power supply voltage • Damaged wiring parts
Periodic Inspection	1 year	• No looseness in the fastening parts • Broken terminal blocks and loose fastening parts

Hardware

Figure 0-1: UIM342XS Wiring Diagram

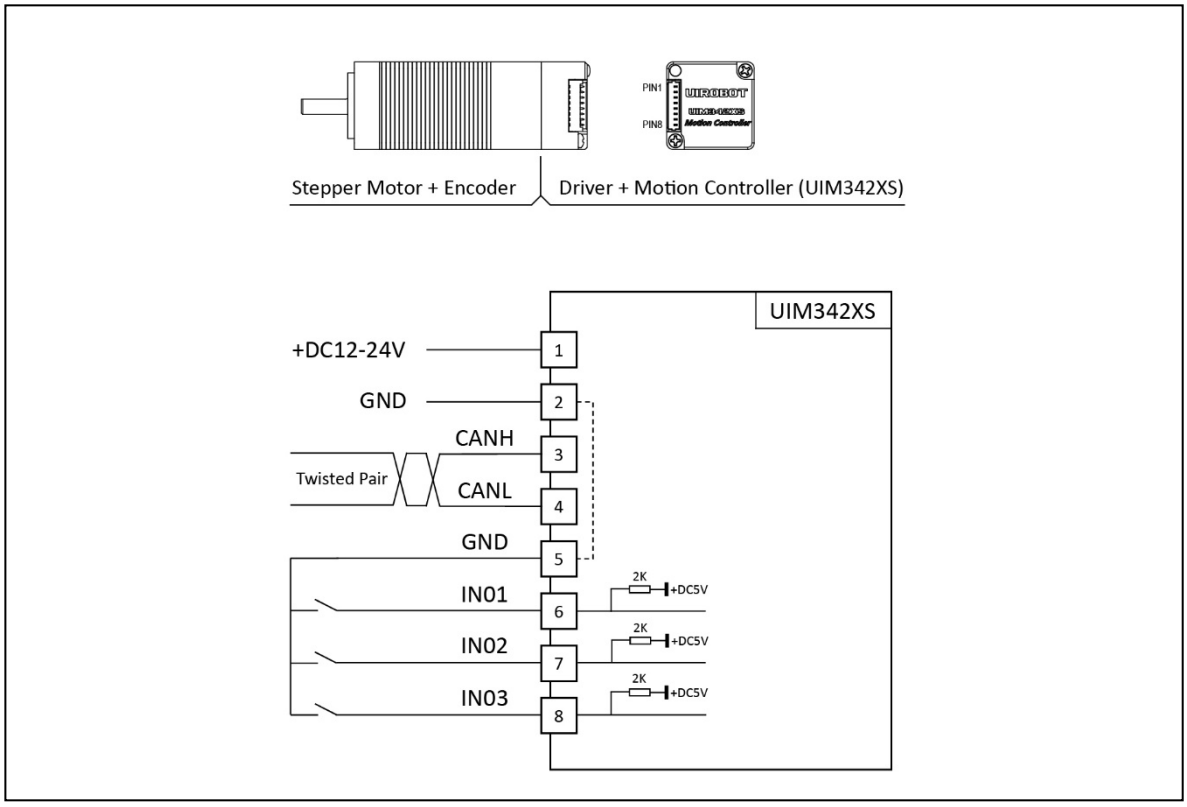


Figure 0-2: UIM342S Wiring Diagram

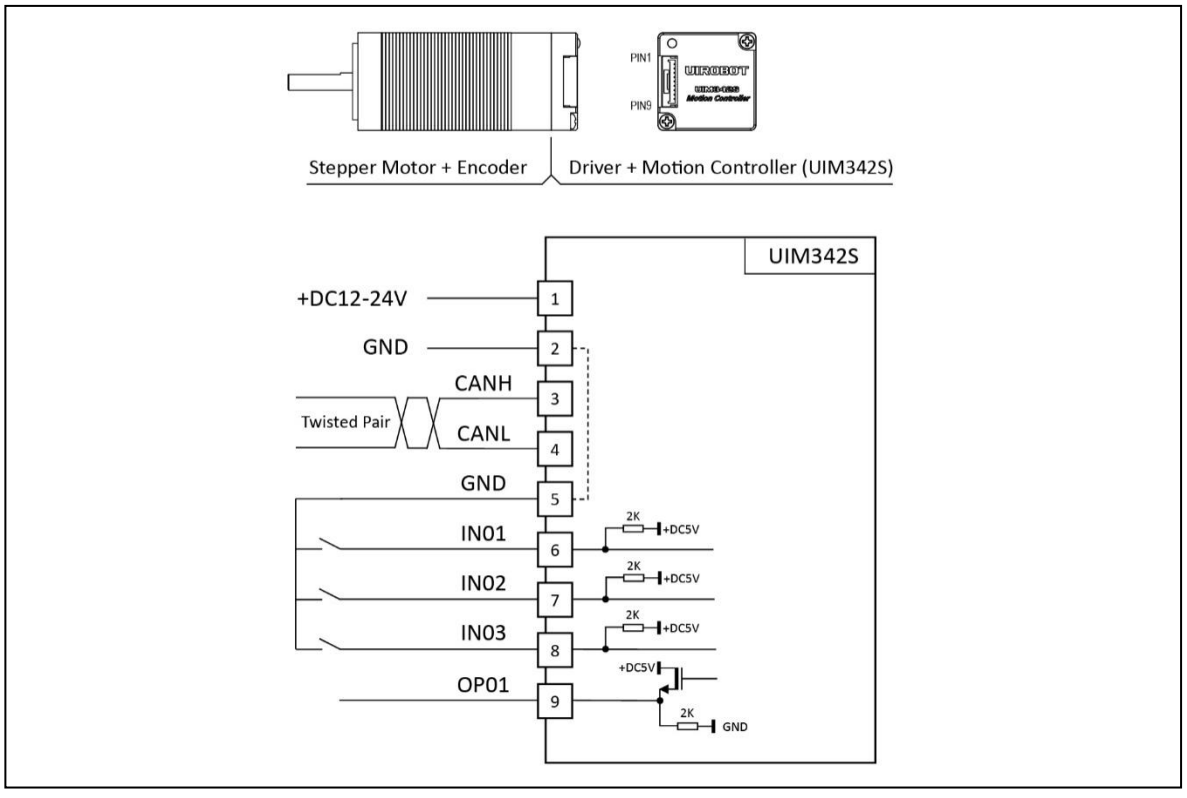


Figure 0-3: UIM342C Wiring Diagram (Socket Style)

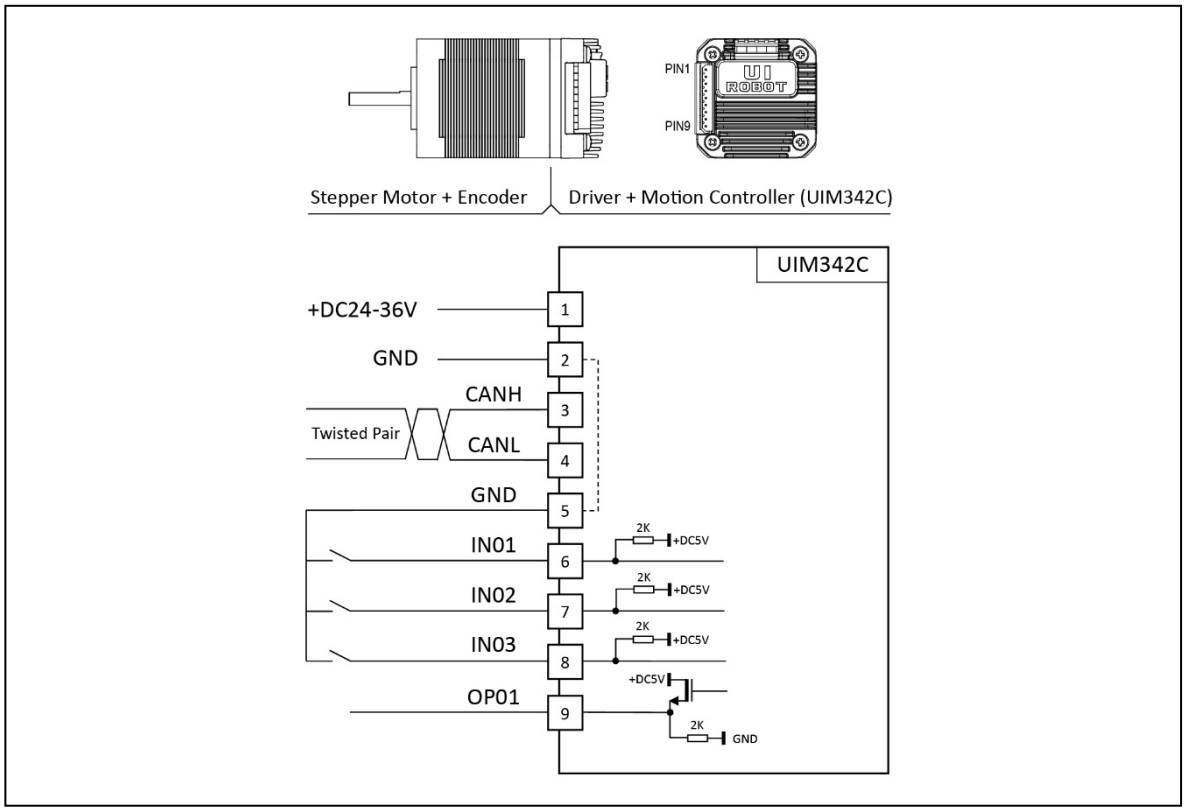
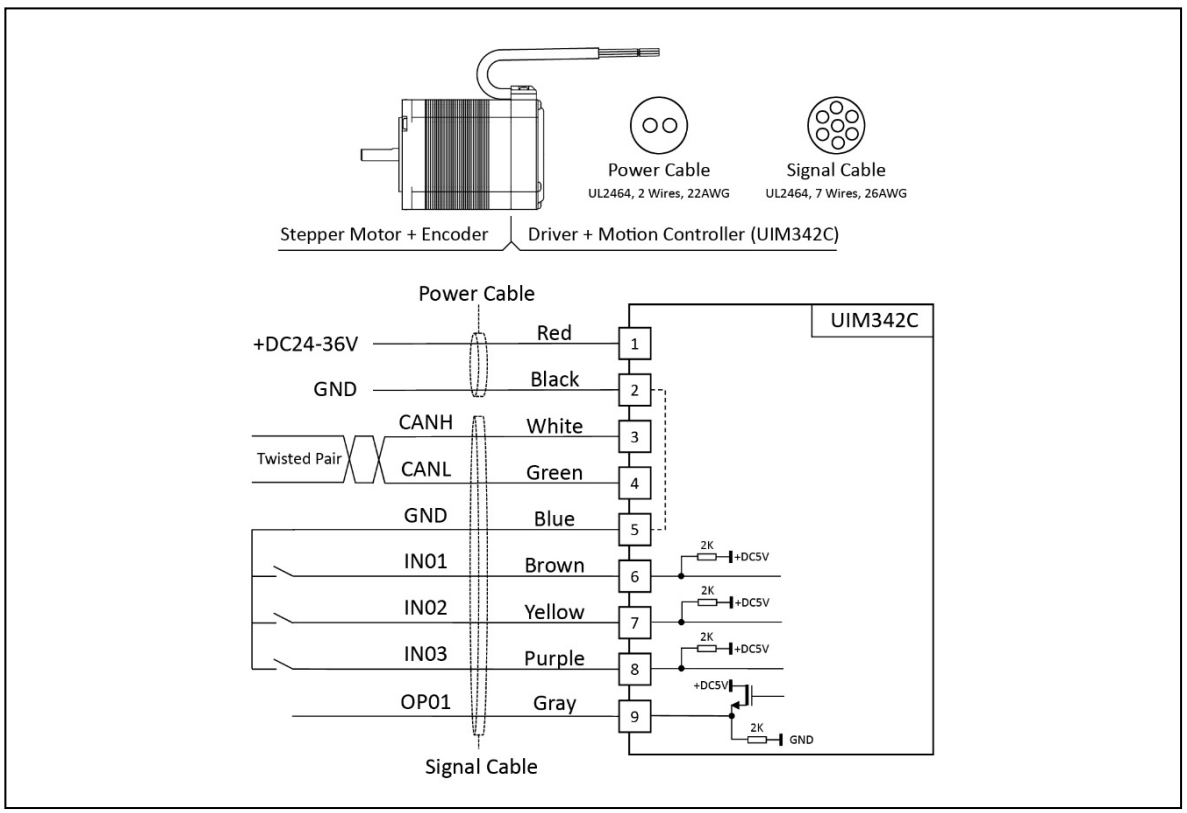


Figure 0-4: UIM342C Wiring Diagram (Cable Style)



Instruction Set Summary

Classification	Mnemonic	Control Word	Function	Object	Chapter
Protocol	PP[i]	0x01	Set / Get	Protocol Parameters	5.1
System	IC[i]	0x06	Set / Get	Initial Configuration	5.2
	IE[i]	0x07	Set / Get	Information Enable	5.3
	ML	0x0B	Get	Model	5.4
	SN	0x0C	Get	Serial Number	5.5
	ER[i]	0x0F	Clear / Get	Error Report	5.6
	QE[i]	0x3D	Set / Get	Quadrature Encoder	5.7
	SY[i]	0x7E	Set	System Operation	5.8
Motor Driver	MT[i]	0x10	Set / Get	Motor Driver parameters	5.9
	MO	0x15	Set / Get	Motor On /Off	5.10
Motion Control	BG	0x16	Set	Begin Motion	5.11
	ST	0x17	Set	Stop Motion	5.12
	MF	0x18	Get	Motion Parameter Frame	5.13
	AC	0x19	Set / Get	Acceleration	5.14
	DC	0x1A	Set / Get	Deceleration	5.15
	SS	0x1B	Set / Get	Cut-in speed	5.16
	SD	0x1C	Set / Get	Stop Deceleration	5.17
	JV	0x1D	Set / Get	Jog Velocity	5.18
	SP	0x1E	Set / Get	PTP Speed	5.19
	PR	0x1F	Set / Get	Position Relative	5.20
	PA	0x20	Set / Get	Position Absolute	5.21
	OG	0x21	Set	Set Origin	5.22
	BL	0x2D	Set / Get	Backlash Compensation	5.23
	MS[i]	0x11	Clear / Get	Motion Status	5.24
	DV[i]	0x2E	Get	Desired Values	5.25
I / O	IL[i]	0x34	Set / Get	Input Logic	5.26
	TG[i]	0x35	Set / Get	Trigger	5.27
	DI	0x37	Set / Get	Digital I /O	5.28
Notification	RT	0x5A	Auto Sent	Real-Time Inform	5.29

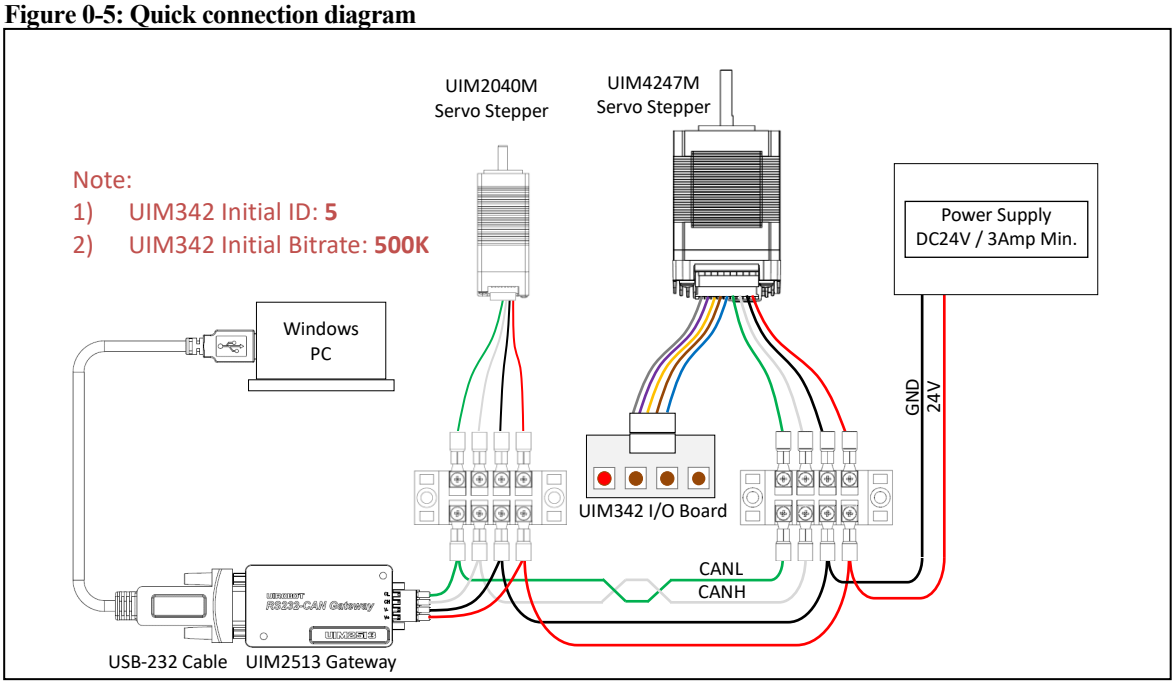
Quick Start

The following explains how to quickly build and run a motor system consisting of 2 servo stepper motors equipped with UIM342 controllers and 1 UIM2513 gateway.

For the convenience of users, the development kit (UIMEVA_342KIT) provided by UIROBOT Company includes the experimental materials used in the system (excluding Windows PC).

1) Connect the wiring as shown below. Be sure to check the wiring again before powering on to make sure it is correct.

Figure 0-5: Quick connection diagram

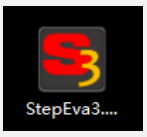


WARNING

- Avoid using star connections. When the CAN cable length exceeds 20 meters, dedicated CAN cables should be used. The length of branch cable to each node should not exceed 20 cm.
- UIM2513 has a built-in terminal resistor, which can be activated by the toggle switch. It is recommended to connect a 120-ohm terminal resistor to the other end of the CAN cable.
- Strictly avoid hot-plugging while the power is on. Hot-plugging may lead to ground loss (i.e. the power supply V+ is connected while V- is disconnected). In such instances, power V+ will flow into other UIM devices via the CAN cable, causing the burnout of multiple UIM devices.
- Connect all UIM devices to a common ground. Activating a high-power device can raise the voltage on one ground significantly. Without a common ground, this elevated voltage may flow via the CAN cable to the ground of other UIM devices, risking the burnout of multiple devices.

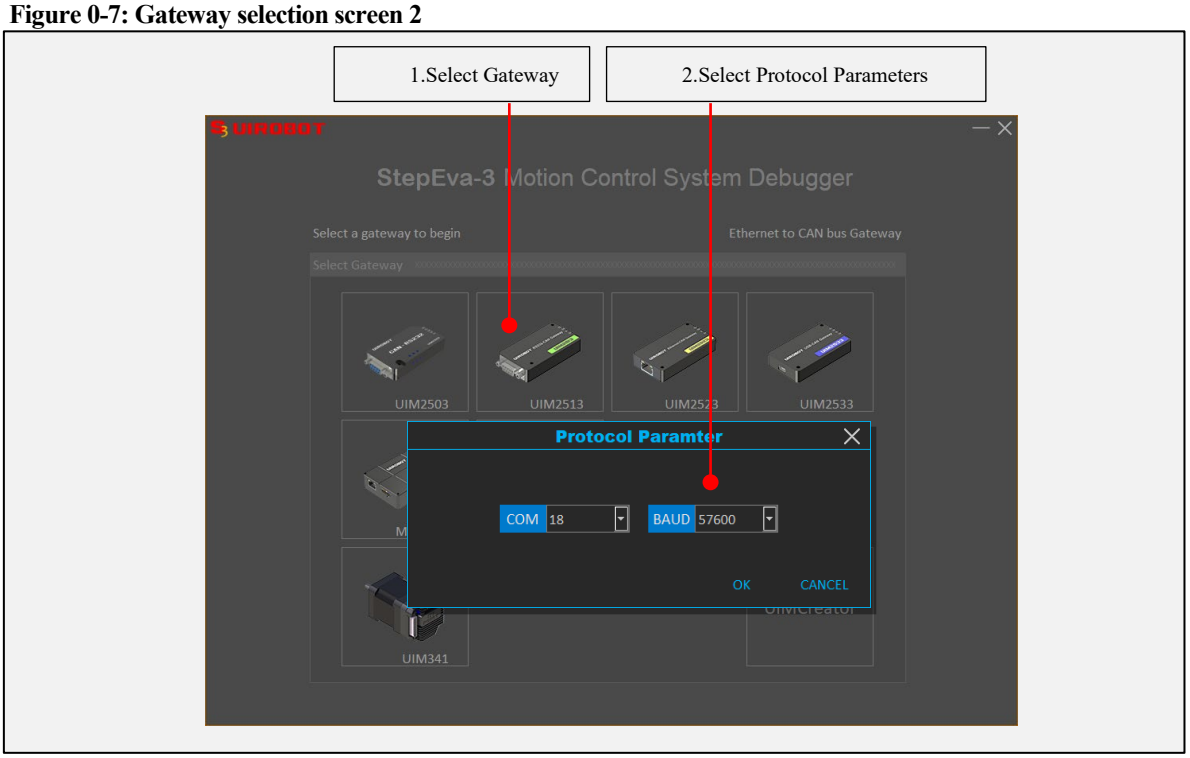
2) Download and click to run the Windows based control panel “StepEva3”.

Figure 0-6: Icon of StepEva3



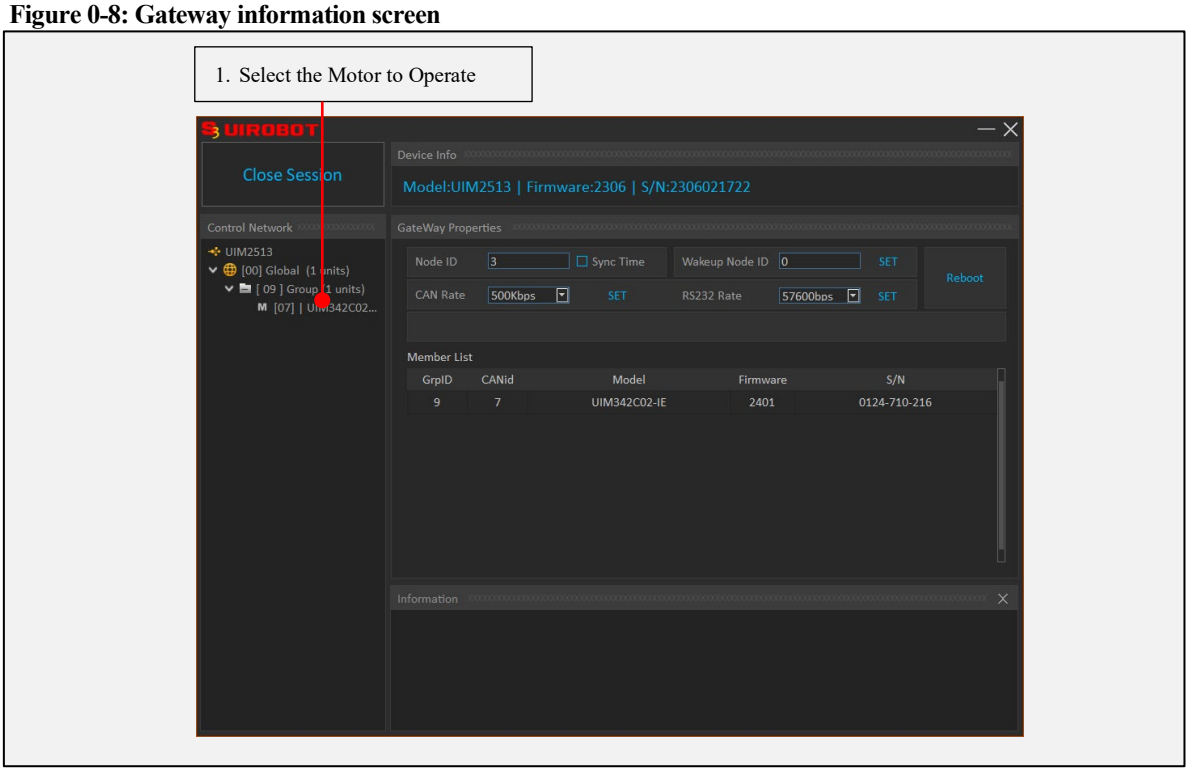
3) UIM2513 (RS232 -CAN) should be selected here.

Figure 0-7: Gateway selection screen 2



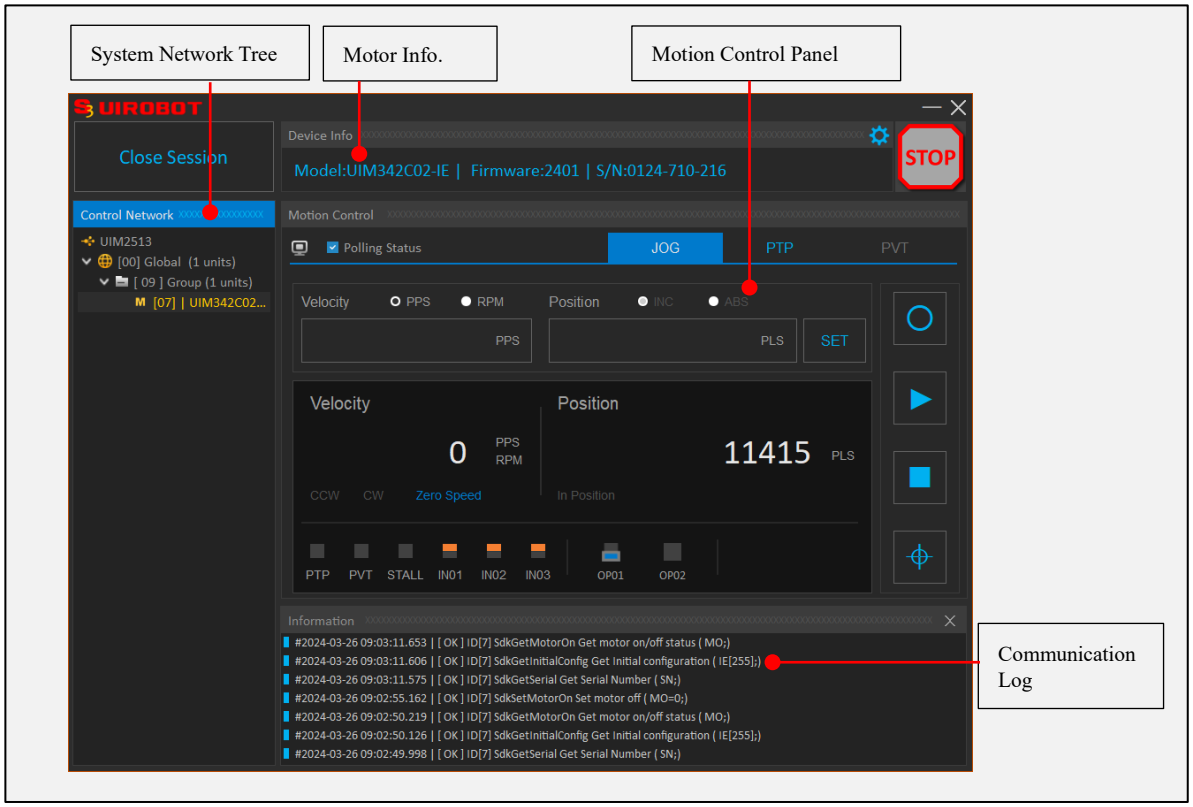
4) Select the UIM342C02-IE to operate in the device list on the left.

Figure 0-8: Gateway information screen



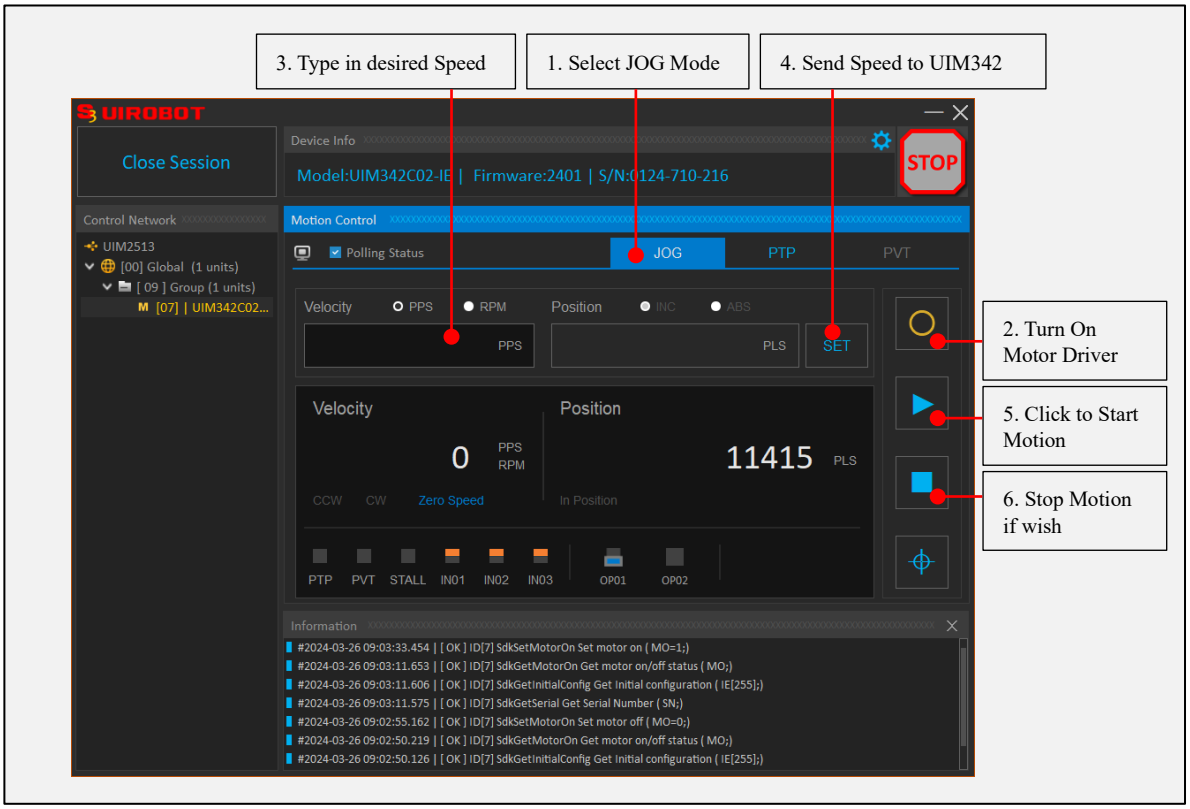
5) Set motion parameters and operate the motor to move.

Figure 0-9: Motion control interface



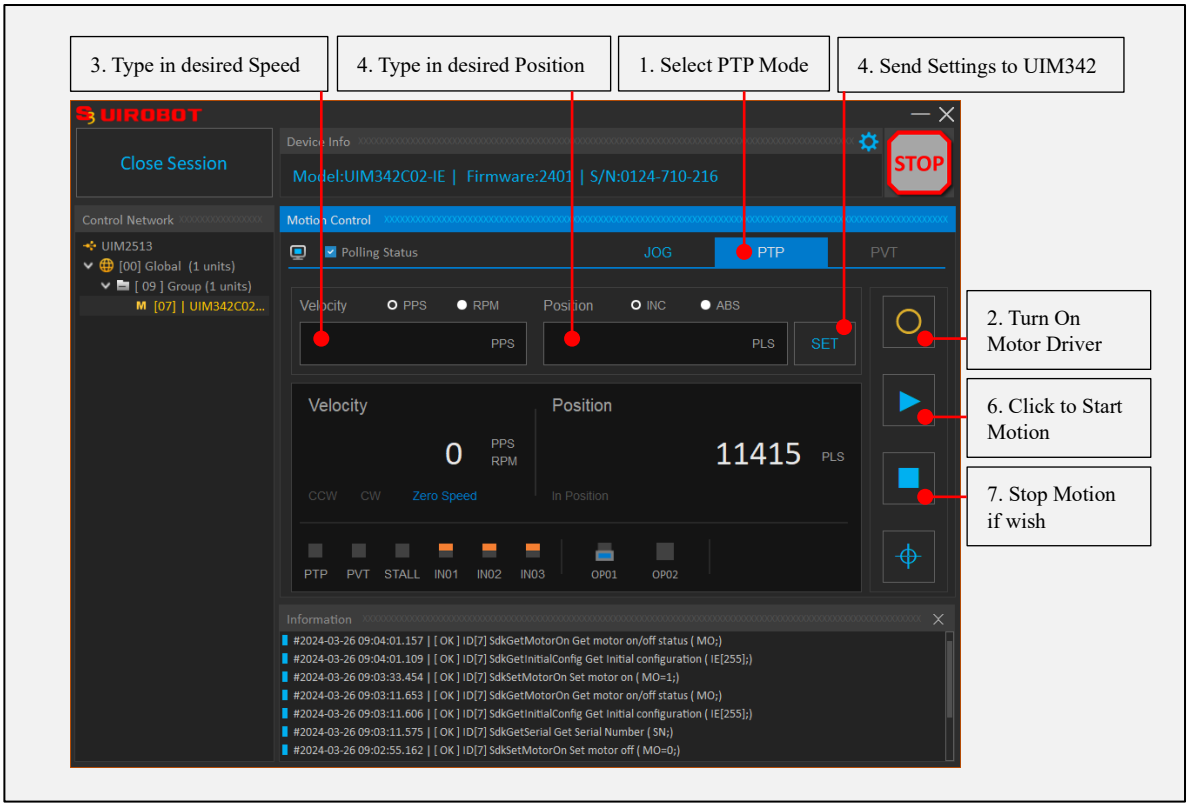
JOG mode, follow steps 1 ...6 below to control the motor rotation.

Figure 0-10: Motion control interface, JOG mode



PTP mode, follow steps 1 to 5 below to control the motor rotation.

Figure 0-11: Motion control interface, PTP mode



Specifications

Absolute Maximum Ratings

Ambient temperature under bias.....-40 °C to 85 °C
Storage temperature ..... -65 °C to +150 °C
Voltage on V+ with respect to V- (UIM342XS / 342S).....10 V to 36 V
Voltage on V+ with respect to V- (UIM342C)…….....10 V to 36 V
Voltage on V+ with respect to V- (UIM342H).....20 V to 36 V
Voltage on input ports with respect to V- ..... -0.3 V to 5.3 V
Maximum current sourced by output port….....100 mA
CANH /CANL voltage with respect to V-.....- 40 V to + 40 V

Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operational listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

Operating Conditions (at ambient temperature 25 °C)

	Specification		
	UIM342XS/342S	UIM342C	UIM342H
Supply Voltage (DC)	10V– 28V	20V-40V	20V-50V
Driver Capacity	Peak 1 / 2 / 4 / 8 Amp		
Current Control	PWM constant current		
Micro-Stepping	full, 1/2/4/8/16/32/64		

Operating Environment

Cooling	Free Air
Environment	Avoid dust, oil mist and corrosive gases
Operating temperature	-40 ℃ ~85 ℃
Humidity	< 80% RH, no condensation, no frost
Vibration	3 G Max
Storage	-65 ℃ ~ 150 ℃

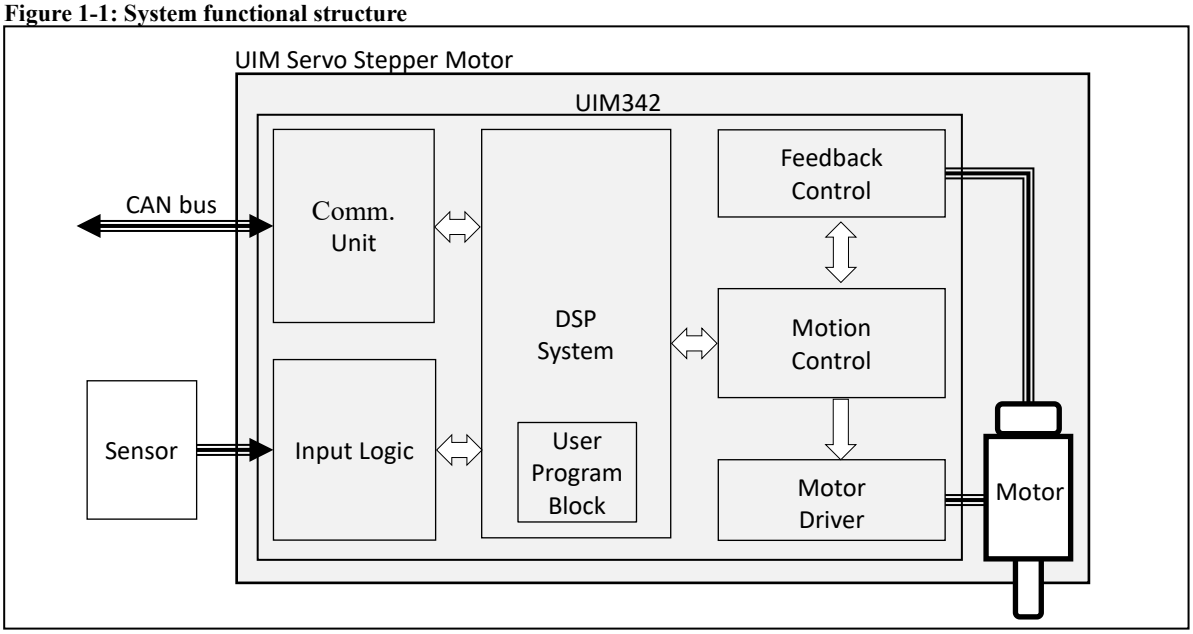
Communication Interface

CAN	Active CAN 2.0
CAN Physical	Two-wire system, CANH, CANL, twisted pair
CAN Driver	• Max. 1 Mbps • Meets ISO-11898 standard physical layer requirements. • Differential bus

1.0 Introduction

UIM342 motion controller can realize open-loop control or closed-loop control. Its main functional modules include: Communication Unit, Motion Control Unit, Motor Driver, Feedback Control Unit, Input Logic Control and DSP System (processing instruction / reply / notification, and coordinating other functional modules, executing preloaded user programs and real-time status notifications, etc.). All control loops are completed within 1 millisecond.

Figure 1-1: System functional structure

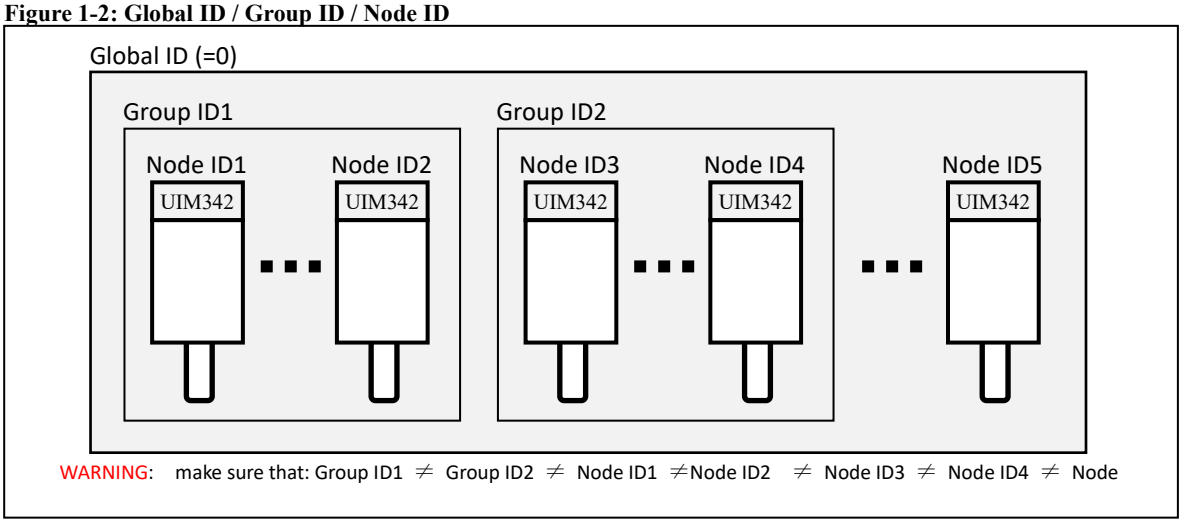


1.1 Communication Unit

Active CAN 2.0 B hardware and software is used, and CAN bit rate is configurable from 125 K to 1000 Kbps by instructions.

To improve the coordination performance of multi-modules, UIM342 supports 3 types of CAN ID: Global ID (=0), Group ID and Node ID. The Group ID and Node ID are configurable via instructions. UIM342 will accept the message that has a CAN ID matches any one of the three IDs.

Figure 1-2: Global ID / Group ID / Node ID



1.2 Motion Control

Contains hardware and software for speed control and position control in open-loop or closed-loop modes.

In addition of normal functions like speed, relative position, absolute position, UIM342 also provides functions such as backlash compensation and automatic reciprocating oscillation to facilitate medical device's needs.

1.3 Motor Driver

Supports 1/2/4/8/16/32/64 micro-stepping, with almost no vibration/noise when running at low speed.

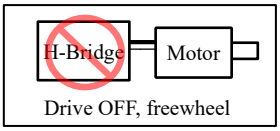
In the low-speed range, compared with AC and DC servo systems, the UIM342 Servo Stepper Motor has higher torque, smaller size, and excellent cost effectiveness.

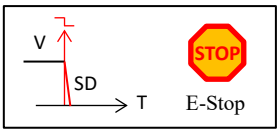
1.4 Input Logic Control

UIM342 supports 3 digital inputs. User can configure the actions to be performed on the logic level change.

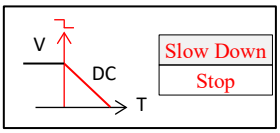
There are total 6 motion parameter sets. Each input port has 2 sets of motion parameters, one for rising edge, and another for falling edge. Each motion parameter set includes: acceleration AC and deceleration DC, +/- speed SP, +/- displacement PR, +/- position PA.

There are 11 functional behaviors that can be attached to the rising/falling edge of the inputs:

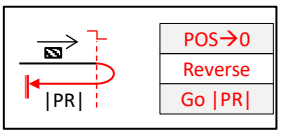
1) 

2) 

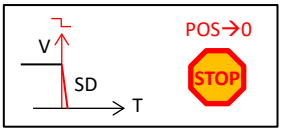
Emergency stop (using SD)

3) 

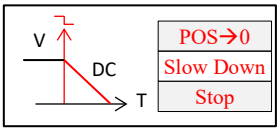
Decelerating to stop (using DC)

4) 

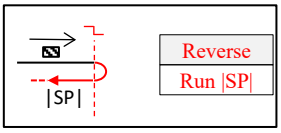
Set origin then go reversed relative position (using |PR|, SP, AC, DC)

5) 

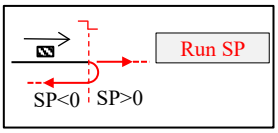
Set origin then Emergency stop (using SD)

6) 

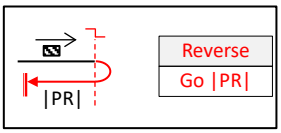
Set origin then decelerate to stop (using DC)

7) 

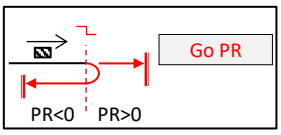
Reversed Jog (using |SP|, AC, DC)

8) 

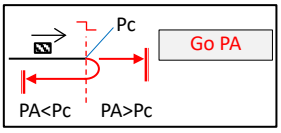
Jog (using SP, AC, DC)

9) 

Go reverse relative position (using |PR|, SP, AC, DC)

10) 

Go relative position (using PR, SP, AC, DC)

11) 

Go absolute position (using PA, SP, AC, DC)

1.5 Logic after Motor Stall

After "motor stall" is detected, by default, the motor shaft will be stopped and locked.

However, set "`IL[16] =1`" will change the logic to "turn motor driver off (freewheel)" after a "stall" situation is detected.

1.6 Real Time Notification

UIM342 can automatically send messages to the user after detecting the occurrence of a preset event. The time from detection to event occurrence to feedback is less than 1 millisecond.

UIM342 supports real-time notification for following events:

- Reach desired position (in PTP mode),
- Rising edge and falling edge of input 1, 2, and 3,
- Detection of motor stall, and
- Other alarm/warnings.

1.7 System development SDK

The instruction set for UIM342 motion controller is simple and highly fault-tolerant. If an incorrect command is entered, the motion controller will return an error message to the host computer. Wrong instructions are not executed to avoid accidents.

UIROBOT Company provides free System Development Kit (SDK), and provides free C++ and C# software control demonstration source code and demonstration software based on Visual studio.

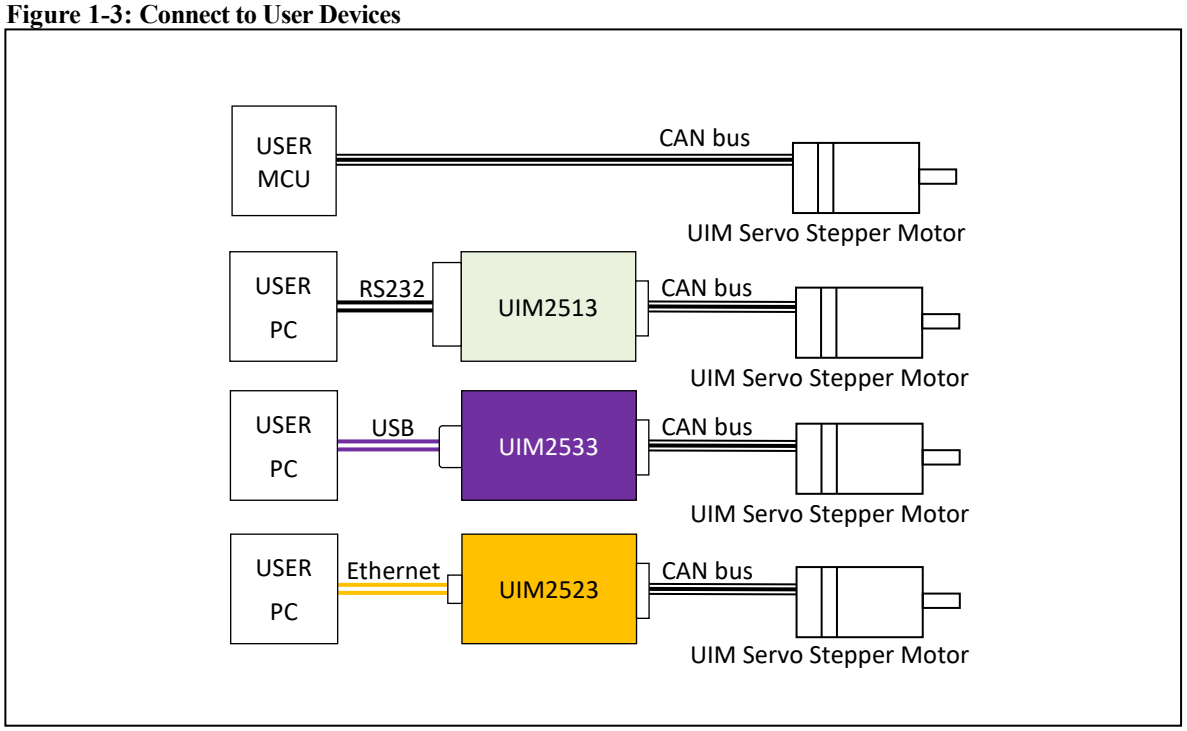
Meanwhile, UIROBOT provides R&D services for user control system solution development.

1.8 Connect to User Devices

There are 4 ways to connect the UIM342 to the user devices, as shown in Figure 1-3:

1） Directly through CAN Bus if the user controller has a CAN port;
2） Using the UIM2513 RS232-CAN gateway (sold separately);
3） Using the UIM2533 USB-CAN gateway (sold separately);
4） Using the UIM2523 Ethernet-CAN gateway (sold separately).

Figure 1-3: Connect to User Devices



2.0 Protocol

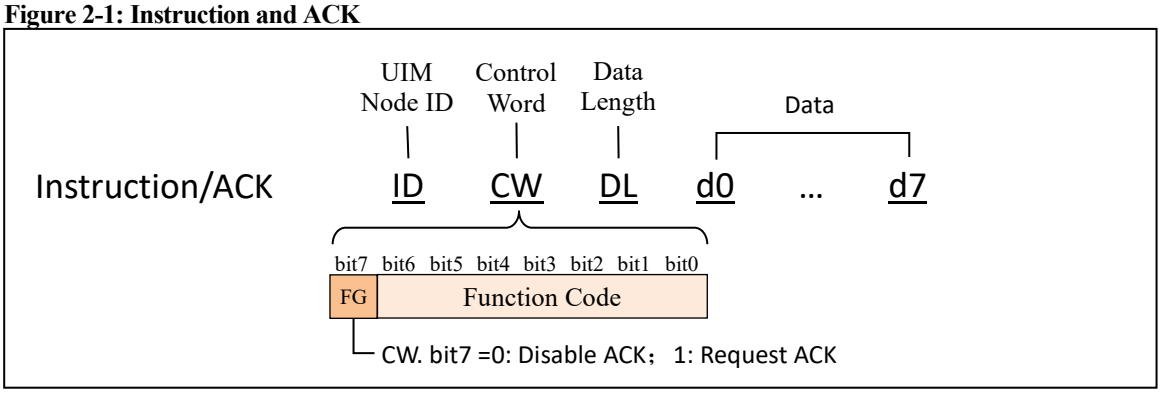
This section introduces the instructions, reply / ACK, error reporting, and real-time status and alarm notifications supported by UIM342.

2.1 Instruction and Reply (ACK)

Instruction is the commanding message sent from the user computer to the UIM342. Before receiving an instruction, UIM342 will not act.

ACK is a feedback sent from UIM342, after receiving an instruction. For setup instruction, the ACK message is typically a repeat of the instruction; for query instruction, the ACK message provides the queried data.

Figure 2-1: Instruction and ACK



ID Device ID (1 byte), could be one of 3 types as listed below:

1） UIM342 Node ID (configurable via instruction),
2） Group ID of UIM devices (configurable via instruction), or
3） Global ID = 0, fixed.

CW Control Word (1 byte):

1） CW.bit <6:0> is the function code (0x00…0x7F);
2） For instructions,
CW.bit7 = 1, ACK is expected;
CW.bit7 = 0, No ACK needed.
3） For ACK, CW.bit7 is fixed to 0.

DL Valid number of bytes of data (1 byte).

d0…d7 Data bytes, least significant byte first.

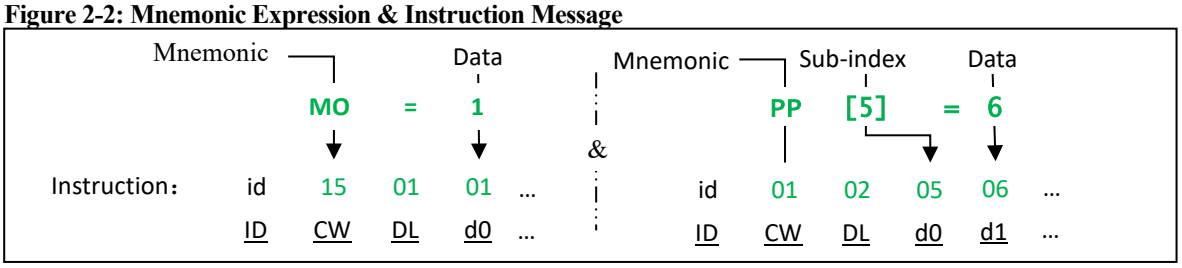
2.2 Mnemonic

For user convenience, UIM products provide a set of mnemonics used to represent various instruction codes. For example:

Mnemonic	CW	Function
MO	0x15	Turn on/off motor drive

The relation between mnemonic expressions and instructions is shown below:

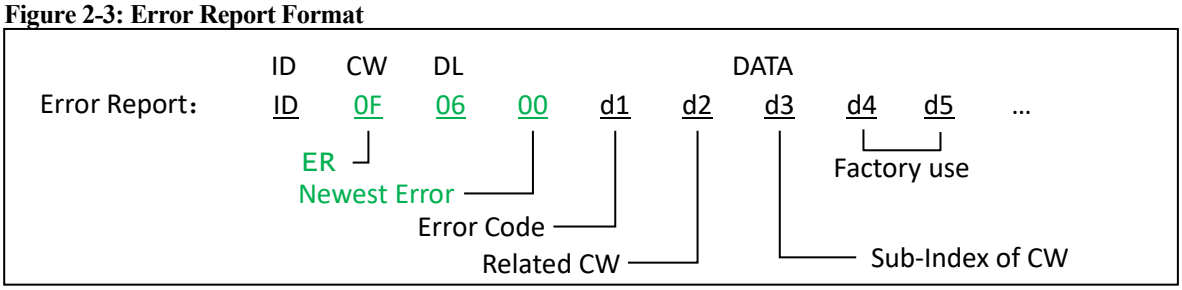
Figure 2-2: Mnemonic Expression & Instruction Message



2.3 Error Report

When UIM342 receives a wrong instruction, it will not execute the instruction, instead, it will reply an error message. Meanwhile, users can use `ER[i]` to get the latest error and the previous 7 historical errors. The format of an error message is listed below:

Figure 2-3: Error Report Format



d1 Error Code;

Error Code for UIM342	Description
0x32	Instruction Syntax error
0x33	Instruction Data error
0x34	Instruction Sub-Index error
0x3C	SD value is less than DC value
0x3D	Current instruction is not allowed when motor is running
0x3E	BG is not allowed when motor driver is OFF
0x3F	BG is not allowed during emergency stopping
0x41	OG is not allowed when motor is running

d2 CW related to the error, e.g., d2 = `DI` indicates an error on previous `DI` instruction.

d3 Sub-Index of the CW related to the error, e.g., d2= `PP`; d3 = 5 indicates an error on previous `PP[5]` instruction.

d4, d5 Factory use, don't care.

2.4 Real-Time Status and Alarm Notification

Real-time notifications are messages the UIM controller automatically sends to the user device, including execution status, alarms, and I/O port level change notifications.

UIM342 is equipped with a notification/information enable configuration register. Some real-time notifications can be enabled / disabled by setting this register. The notification enable register is read and written by the instruction `IE[i]`. Refer to "5.3 `IE[i]` Information Enable" for details.

The real-time status notification format sent by UIM342 to the user is as follows:

Figure 2-4: Real-time Notification

Real Time Notification	ID	5A	DL	d0	d1	…	d7

Types of notifications are shown in the table below: (The message examples are all assumed to be produced by site 5(ID=5))

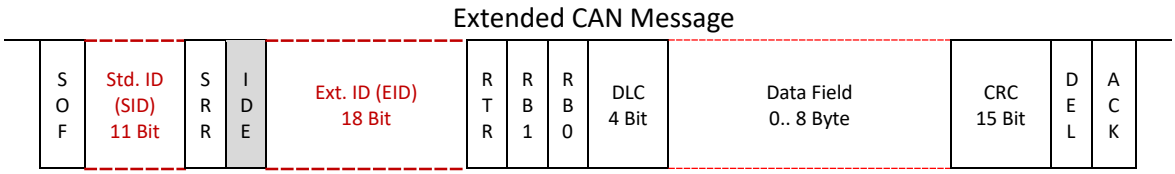
d0	d1	Type	Notification message example	Remark
-	-	-	ID CW DL d0 d1 d2 d3 d4 d5 d6 d7	-
0	0x0A	Alarm	05 5A 05 00 0A 00 00 00 00 00 00	System emergency stop and lock
0	0x19	Alarm	05 5A 05 00 19 00 00 00 00 00 00	Speed over limit
0	0x1A	Alarm	05 5A 05 00 1A 00 00 00 00 00 00	Exceed Lower Working Limit
0	0x1B	Alarm	05 5A 05 00 1B 00 00 00 00 00 00	Exceed Upper Working Limit
0	0x1D	Alarm	05 5A 05 00 1D 00 00 00 00 00 00	Motor Stall Detected
0	0x1E	Alarm	05 5A 05 00 1E 00 00 00 00 00 00	Encoder Error
0	0x1F	Alarm	05 5A 05 00 1F 00 00 00 00 00 00	Encoder Battery Low
0x01	0	Status	05 5A 01 01 00 00 00 00 00 00 00	Input 1 falling edge detection
0x02	0	Status	05 5A 01 02 00 00 00 00 00 00 00	Input 1 rising edge detection
0x03	0	Status	05 5A 01 03 00 00 00 00 00 00 00	Input 2 falling edge detection
0x04	0	Status	05 5A 01 04 00 00 00 00 00 00 00	Input 2 rising edge detection
0x05	0	Status	05 5A 01 05 00 00 00 00 00 00 00	Input 3 falling edge detection
0x06	0	Status	05 5A 01 06 00 00 00 00 00 00 00	Input 3 rising edge detection
0x29	0	Status	05 5A 08 29 00 00 00 p0 p1 p2 p3	PTP positioning completed, [p3:p0] = current position

2.5 Direct CAN Communication

When directly communicating with UIM342 devices via CAN Bus, the user CAN controller needs to be configured as follows:

1） TQ number, use one of the following options:
- 1 M bps use 8 TQ;
- 800 K bps use 10 TQ;
- Any other bps, use 16 TQ.

2） Synchronous jump: 1 TQ.
3） Sample once at each sampling time.
4） Phase Segment 2: 2 TQ.
5） Filter and MASK are configured to receive all messages.
6） Set SRR = 1; IDE = 1; RTR = 0; RB1 = 0; RB0 = 0;



EID, SID, DLC, and data0…data7 of the CAN message should be processed as follows:

Sending instructions to UIM342:

```
SID = ((ID<<1) & 0x003F) | 0x0100;  //ID (Consumer, UIM's Address)
EID = (((ID<<1) & 0x00C0)<<8) | CW; //CW (Control Word)
DLC = DL;                           //DL (Effective Data Length)
CANTX_B0 = d0;
CANTX_B1 = d1;
CANTX_B2 = d2;
CANTX_B3 = d3;
CANTX_B4 = d4;
CANTX_B5 = d5;
CANTX_B6 = d6;
CANTX_B7 = d7;

Note: Some controllers may use 32bits CAN-ID, which = (SID<<18) | EID;
```

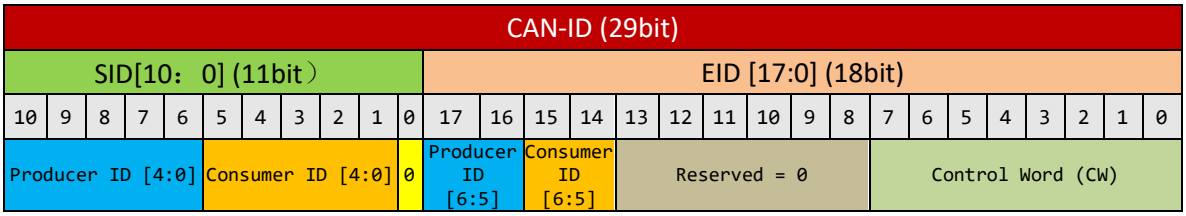
Parsing received messages from UIM342:

```
ID = ((EID>>11)&0x0060)|((SID>>6)&0x001F); //ID (Producer, UIM's Address)
CW = EID&0x00FF;                          //CW (Control Word)
DL = DLC&0x000F;                          //DL (Effective Data Length)
d0 = CANRX_B0;
d1 = CANRX_B1;
d2 = CANRX_B2;
d3 = CANRX_B3;
d4 = CANRX_B4;
d5 = CANRX_B5;
d6 = CANRX_B6;
d7 = CANRX_B7;

Note: Some controllers may use 32bits CAN-ID, which = (SID<<18) | EID;
```

To help further understanding the CAN protocol (i.e., SimpleCAN3.0) used to control the UIM342, more details and examples are listed below.

CAN-ID for SimpleCAN3.0



Producer ID	ID of the controller (Master); For user master controller, should use ID = 4.
Consumer ID	ID of UIM342 (Initial ID = 5, range 5…127), can be modified via Instructions.
Control Word	Control Word for UIM342; For instance, CW = 0x95 is to enable the motor driver.

Calculation of SID:

```
SID = ((Consumer ID << 1) & 0x003F) | 0x0100;
```

Calculation of EID:

```
EID = (((Consumer ID << 1) & 0x00C0) << 8) | CW;
```

In case of your controller only supports the CAN-ID (32-bit) format, calculate as follows:

```
CAN-ID = SID << 18 | EID;
```

Example

Using a CAN master device with ID 4 to enable (i.e., MO=1;) a UIM342 motor driver with ID 5

Producer ID = 4

Consumer ID = 5

CW = 0x95 (MO)

Data = 1;

Calculate SID: SID = ((5 << 1) & 0x003F) | 0x0100 = 0x010A;

Calculate EID: EID = (((5 << 1) & 0x00C0) << 8) | CW = 0x0095;

Calculate CAN-ID: CAN-ID = 0x010A << 18 | 0x0095 = 0x04280095;

DLC = 0x01;

Data0 = 0x01;

Send following message from user controller to UIM342:

CAN-ID (hex)	Date Field(hex)
04280095	01

If the user computer needs to specify DLC, please send following:

CAN-ID (hex)	DLC (hex)	Date Field (hex)
04280095	01	01

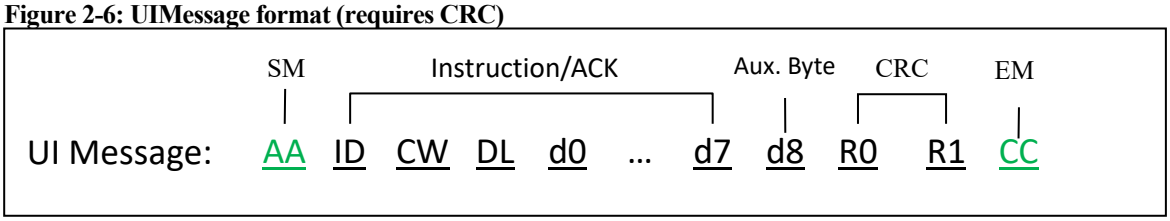
More examples of CAN message used to control the UIM342 motors are listed below.

Can-ID (hex)	Data (hex)	Function
04280095	00	MO = 0, set drive off
04280095	01	MO = 1, set drive on
0428009D	10 27	JV = 10000pps, set jog speed 10000pps
0428009D	F0 d8 FF FF	JV =-10000pps, set jog speed -10000pps
04280096	*No data*	BG, set begin motion
04280097	*No data*	ST, set stop motion
0428009F	80 0C	PR = 3200
0428009F	80 F3 FF FF	PR =-3200
042800A0	80 0C	PA = 3200
042800A0	80 F3 FF FF	PA =-3200
0428009E	80 0C	SP = 3200

2.6 RS232 / USB / Ethernet Gateway

Gateway such as UIM2513, UIM2523, UIM2533 Messages send / receive are all UIMessage format.

Figure 2-6: UIMessage format (requires CRC)



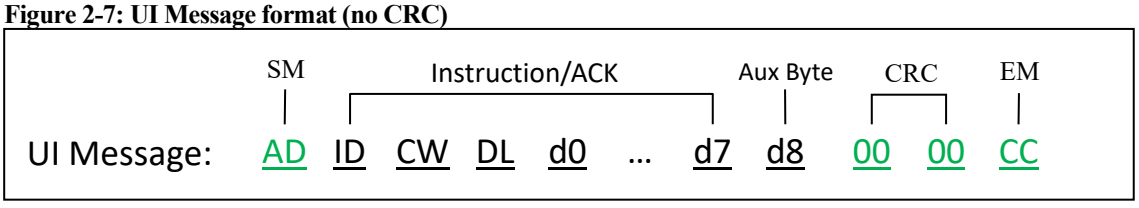
SM Start of Message. If need CRC, using 0xAA, else using 0xAD instead,

d8 Auxiliary byte, don't care,

R1:R0 RTU CRC16, range covers AA…d8, refer to Appendix- 1 for calculation source code,

EM End of Message; is fixed to 0xCC.

Figure 2-7: UI Message format (no CRC)



Notice: UIMessage is in Hex Format, and the length of the message is fixed to 16;

3.0 Motion Control

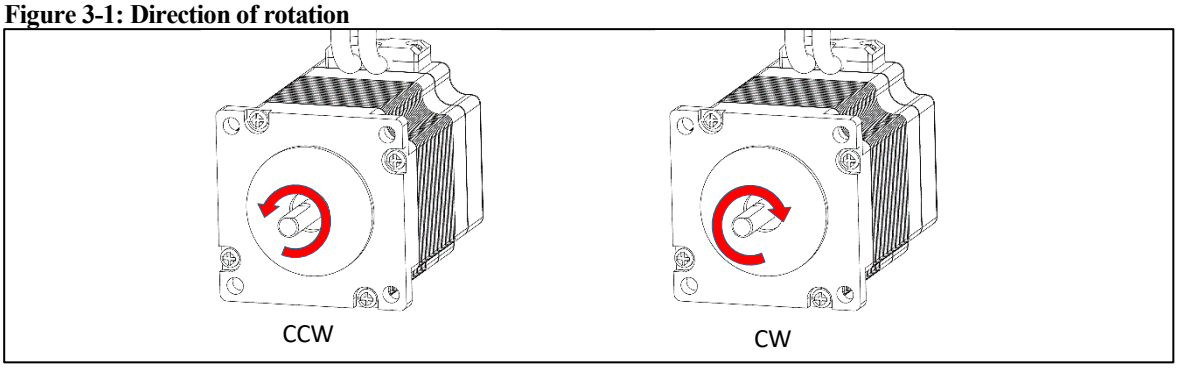
UIM342 motion controller supports speed control mode and point to point (PTP) position control mode. User can Set, Get and Store motion parameters through instructions. Meanwhile, user can also receive the working status of the motor through real-time notification.

This chapter briefs the motion control related functions of UIM342.

3.1 Direction of Rotation

Facing the output shaft, direction of the motor rotation is defined as CW if the shaft rotates clockwise, and CCW if the shaft rotates count clockwise.

Figure 3-1: Direction of rotation



Set "IC[1]=1" to configure the position counter increase when the motor is running in CCW direction (Factory default). Otherwise, set "IC[1]=0" to configure the position counter increase when the motor is running in CW direction.

3.2 Motion Control Modes

- **Speed control (JOG)**

UIM342 controls the motor speed to reach the desired speed set by the user through an acceleration or deceleration process.

The JV instruction sets the desired speed value, and the sign (+/-) of the value determines the direction of the rotation. In addition, the AC instruction sets the acceleration rate and the DC instruction sets deceleration rate.

After a JV instruction, the BG instruction must be set to start the movement. In this manner, multiple UIM342 controllers can be set JV one by one, and start moving at exactly the same time via sending BG to the group ID or the global ID.

- **Position control (PTP)**

UIM342 controls the speed and displacement of the motor and stops when it reaches the desired position. UIM342 approaches the desired speed to the maximum extent while ensuring that the position is accurately reached.

The SP instruction sets the speed used in the position control. The PA instruction sets the absolute position, and the PR instruction sets the relative position with regards to the current position.

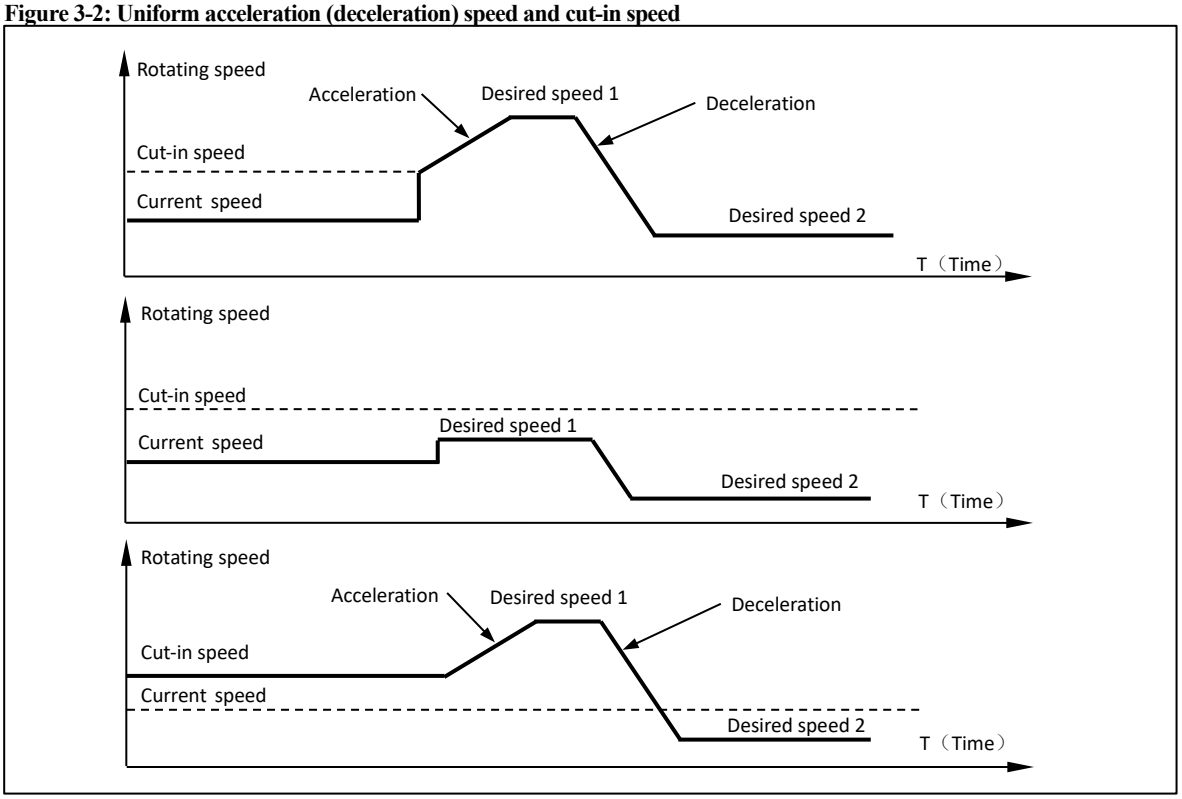
After PA/PR instruction, the BG instruction must be set to start the movement. In this manner, multiple UIM342 controllers can be set PA/PR one by one, and start moving at exactly the same time via sending BG to the group ID or the global ID.

Please note, the actual speed/direction of the motor is determined by the position deviation (i.e., desired position - actual position). If the position deviation is too small, the motor may reach the desired position before reaching the desired speed.

3.3 Acceleration, Deceleration and Cut-in speed

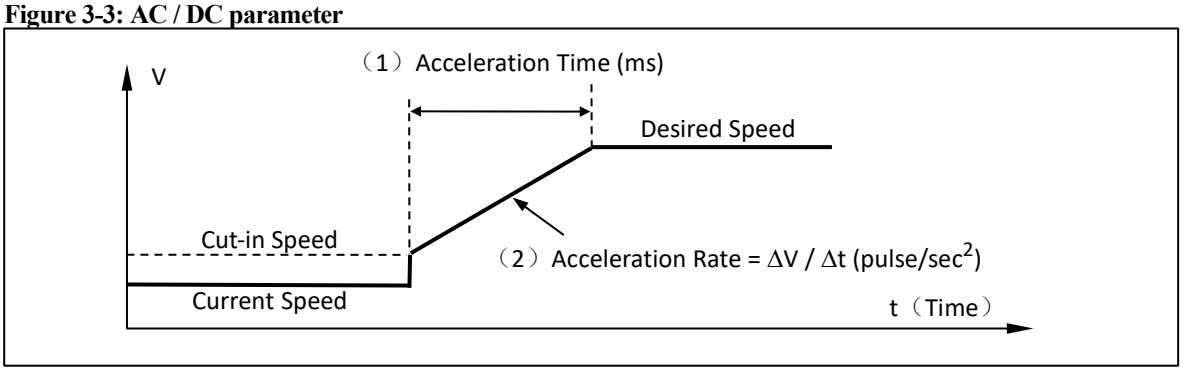
In order to improve the response of the motor and avoid the resonance, the acceleration (deceleration) process includes a cut-in speed.

Figure 3-2: Uniform acceleration (deceleration) speed and cut-in speed



Instructions AC, DC, SS are used to set acceleration, deceleration and cut-in speed. Acceleration and deceleration can be defined by the time of the process, or the rate of acceleration and deceleration. The two definitions are shown in the figure below.

Figure 3-3: AC / DC parameter



3.4 Backlash Compensation

Mechanical systems (such as screw nut transmission or rack and pinion transmission) have backlash problems. For example, there is a certain gap between the screw and the nut. When the forward rotation changes to reverse rotation, within a certain angle, although the screw rotates, the nut still have to wait for the gap to disappear (the side of the force-bearing side has to wait for the gap to

disappear).can drive the workbench to move later. Due to the existence of backlash, when the reverse motion starts, the accumulated error continues to increase and then becomes stable after the backlash is fully compensated.

UIM342 provides a backlash compensation function to reduce the impact of backlash on mechanical transmission accuracy. Use BL instruction to set the backlash compensation value.

The unit of backlash compensation setting value is pulse, range (0…65536).

3.5 Closed Loop / Open Loop Control

The UIM342 motor controller UIM342 supports both open loop control and closed-loop control.

Open Loop Control – Set IC[6]=0 to enable this mode, which operates without feedback.

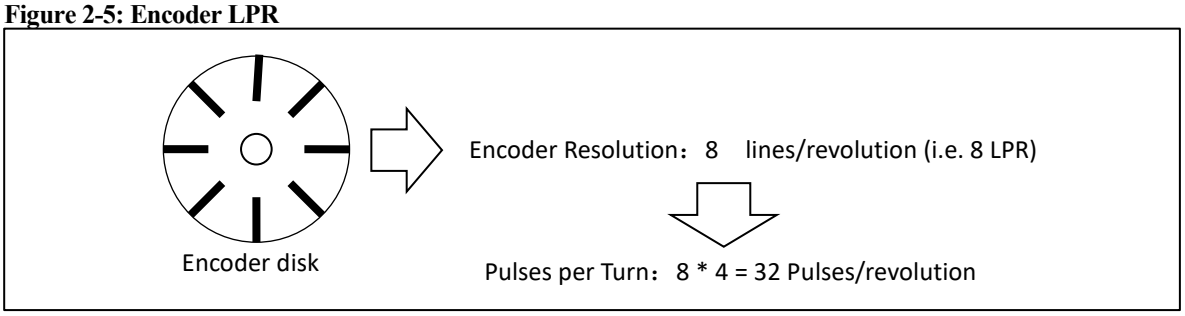
Closed-Loop Control - Set IC[6]=1 to activate this mode, where feedback is provided by a quadrature encoder to ensure accurate displacement control.

In the closed-loop control mode, a quadrature encoder is used for displacement feedback, providing more precision in motor positioning.

Lines per Revolution (LPR)

LPR is the number of lines engraved on the encoder's disk. For each full revolution of the motor, the encoder completes one full rotation, generating 4* LPR pulses.

Figure 2-5: Encoder LPR



Counts per Revolution (CPR)

CPR specifies the number of pulses needed to rotate the motor one full turn. This is essential for closed-loop control in the UIM342, as it determines the pulse requirements for precise positioning. The CPR for the motor is determined by the formula:

$$CPR = \text{micro-stepping resolution} * 200$$

Where **200** is the number of steps per revolution for a standard stepper motor in full-step mode.

[Example Calculation]

For a micro-stepping resolution of 16, the CPR would be:

$$CPR = 16 * 200 = 3200$$

This means 3200 pulses are required to make the motor complete one full revolution.

3.6 Stall Detection

The Stall Tolerance is the maximum allowed difference (pulses) between the encoder feedback and the actual driving pulses sent. When missing steps (pulses) accumulates to the Stall Tolerance, the stall will be detected. UIM 342 will handle the stall detection event.

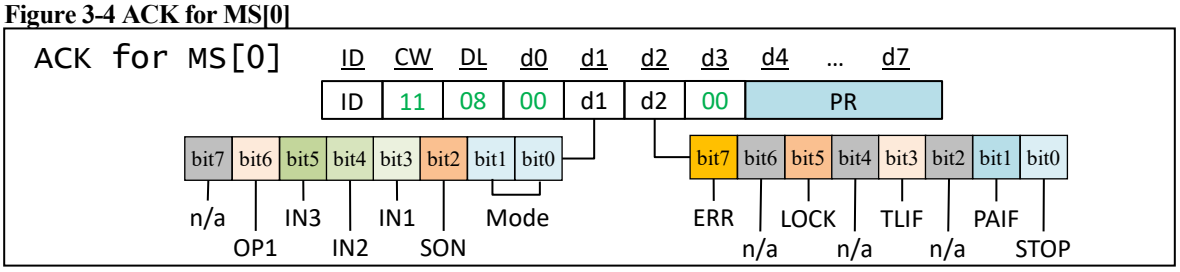
3.7 Acquire Motion Status

To facilitate quick query of motion status, the `MS[i]` instruction can be used as described below.

- **Get Status Flags and Relative Position**

`MS[0]` is used to get the current motion status flags and the relative position. The ACK message for the `MS[0]` instruction is explained in the figure below:

Figure 3-4 ACK for MS[0]



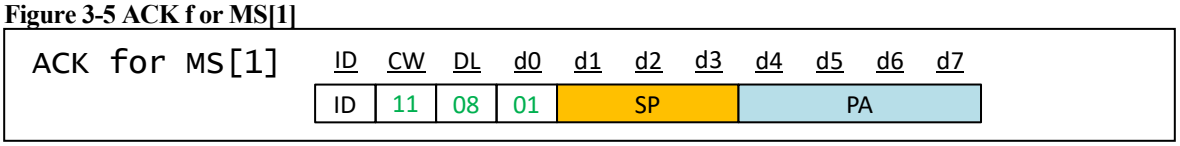
	name	description	value
d1.bit<1:0>	Mode	motion mode	0 = JOG; 1 = PTP
d1.bit2	SON	motor driver	0 = OFF; 1 = ON
d1.bit3	IN1	IN1 Logic Level	0 = Low; 1 = High
d1.bit4	IN2	IN2 Logic Level	0 = Low; 1 = High
d1.bit5	IN3	IN3 Logic Level	0 = Low; 1 = High
d1.bit6	OP1	OP1 Logic Level	0 = Low; 1 = High
d1.bit7	n/a		0
d2.bit0	STOP	Motor is in stationary	0 = NO; 1 = YES
d2.bit1	PAIF	Motor is in position	0 = NO; 1 = YES
d2.bit2	n/a		0
d2.bit3	TLIF	Motor stall is detected	0 = NO; 1 = YES
d2.bit4	n/a		0
d2.bit5	LOCK	System is locked down	0 = NO; 1 = YES
d2.bit6	n/a		0
d2.bit7	ERR	System error is detected	0 = NO; 1 = YES
d3	n/a		00
d7:d6:d5:d4	PR	Current relative position	Signed 32 bit integer, LSB received first

Note: LSB - least significant byte

- **Get Speed and Absolute Position**

`MS[1]` is used to get the current speed and absolute position.

Figure 3-5 ACK f or MS[1]



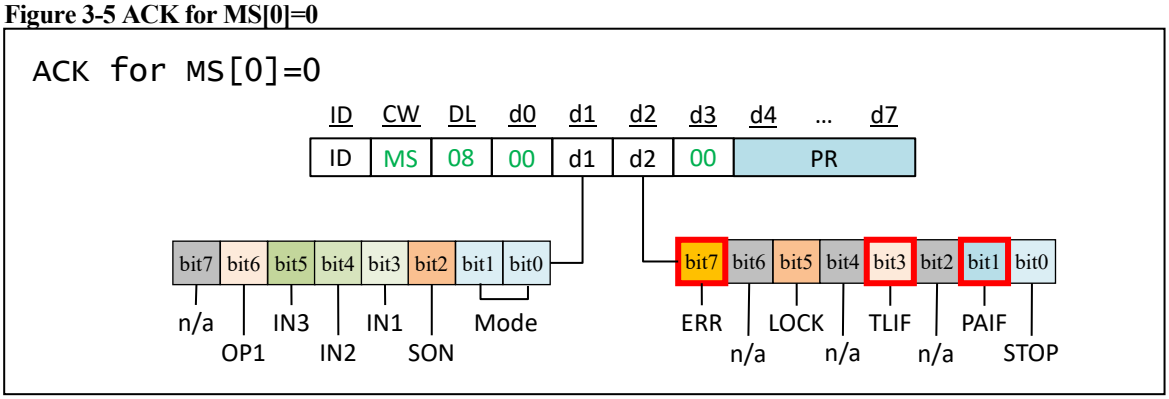
	name	description	value
d3:d2:d1	SP	Current speed	Signed 24 bit integer, LSB received first
d7:d6:d5:d4	PA	Current absolute position	Signed 32 bit integer, LSB received first

Note: LSB - least significant byte

- **Clear Status Flags**

"MS[0]=0" is used to clear the flag bit PAIF, TLIF and ERR. ACK message for the "MS[0]=0" instruction is explained in the figure below (values in the red boxes are cleared):

Figure 3-5 ACK for MS[0]=0



4.0 Input Logic Control

The input logic control module has the following functions:

1) Processing and reporting the logic levels of input ports,
2) Executing the following user preset actions, when the input logic level change is detected:
   - Motor Driver OFF,
   - Emergency stop (using SD),
   - Decelerating to stop (using DC),
   - Set origin then go reversed relative position (using |PR|, SP, AC, DC),
   - Set origin then Emergency stop (using SD),
   - Set origin then decelerate to stop (using DC),
   - Reversed Jog (using |SP|, AC, DC),
   - Jog (using SP, AC, DC),
   - Go reverse relative position (using |PR|, SP, AC, DC),
   - Go relative position (using PR, SP, AC, DC),
   - Go absolute position (using PA, SP, AC, DC).

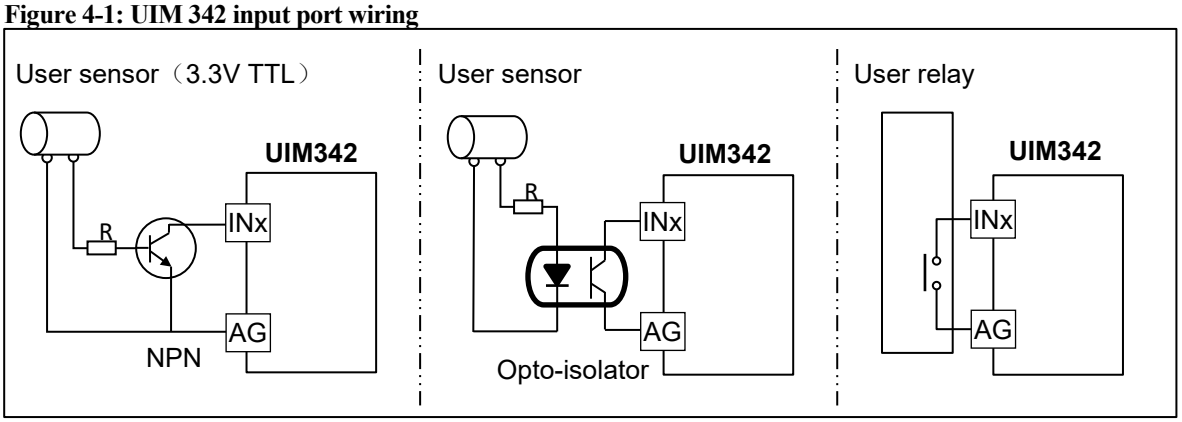
4.1 Wiring the Sensor

UIM342 input ports only takes 5V TTL signal.

If a 3.3V TTL sensor is to be used, an NPN transistor should be added as shown in figure 4-1.

It is recommended to use a photoelectric isolation module between the sensor and the INx port of the controller. The photoelectric isolation module has strong anti-interference properties and is recommended for use in factory environments and situations with significant interference. Please adjusting the resistance R to ensure that the current through the isolator is around 10 mA.

Figure 4-1: UIM 342 input port wiring



4.2 Trigger modes

The voltage on each input port can be processed with three types of trigger modes:

- **Continuous Trigger Mode**

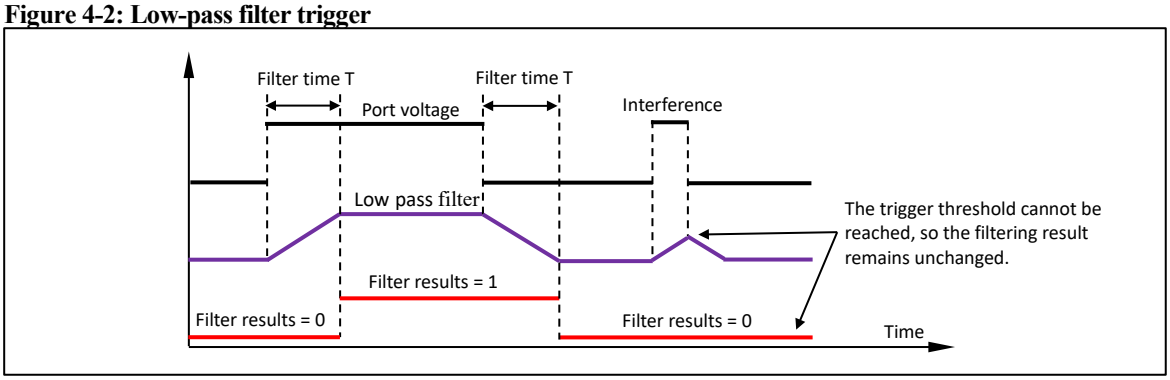
In this mode, UIM342 continuously samples the voltage on input ports. If a logic level change is detected, UIM342 will immediately notified the user and execute the preset logic actions.

To use this mode of trigger, set "TG[i]=0", i=0, 1 or 2.

- **Low Pass Filter Mode**

In this mode, UIM342 performs digital low-pass filtering on the input and then collects the level changes. This can eliminate jitters and prevent interference from digital inputs.

Figure 4-2: Low-pass filter trigger



To use this mode, set "`TG[i]` = T", i=0, 1 or 2.

This instruction sets the filter time of IN i to T ([1 ... 60000] ms).

- **One-Time Trigger Mode**

In this Trigger Mode, UIM342 will processing a port level change only once.

After that, the actions attached to the input port will be cleared and reset to none. User must set `IL[i]` again, in order to use the input logic again.

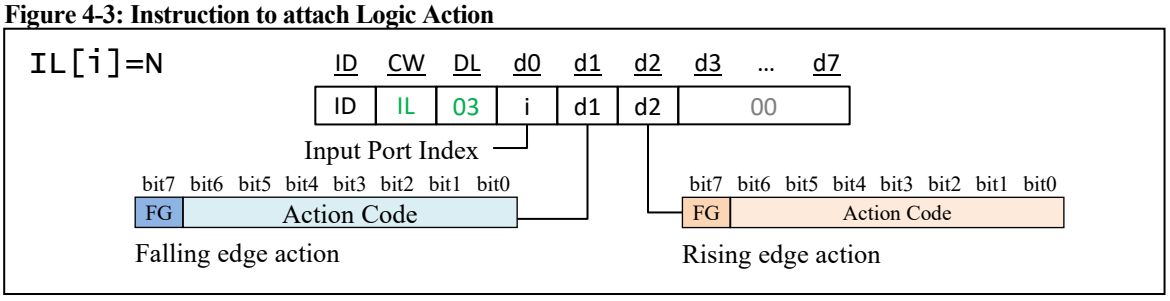
To use this mode, set "`TG[i]` = 60001", i=0, 1 or 2.

This trigger mode is useful in finding/setting the position origin, preventing the oscillation caused by repeated detection.

4.3 Configure / Attach Input Logic Action

Action attached to the input port i is configured by the instruction `IL[i]` (i=0, 1 or 2), which is defined as follows:

Figure 4-3: Instruction to attach Logic Action



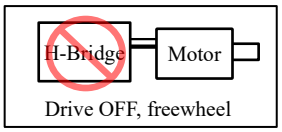
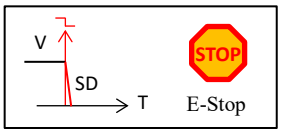
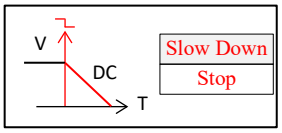
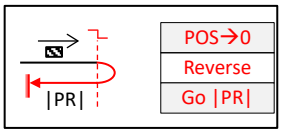
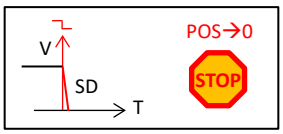
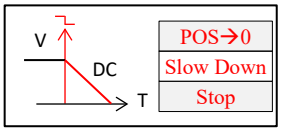
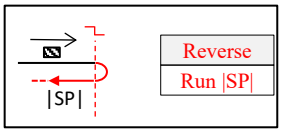
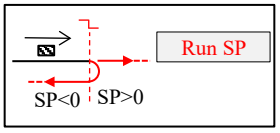
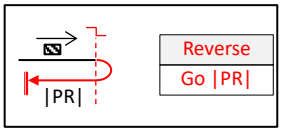
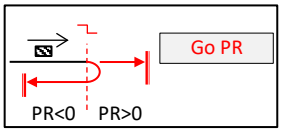
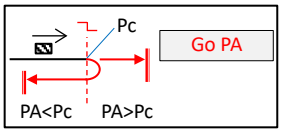
FG – Flag for Power-On Execution.

0 = Execute the action code automatically when power on, disregard the input logic level. This will create a power-on action.

1 = Prohibit Executing the Action Code when power on, until an input logic level change happened.

i – Index of Input Ports. i = 0, 1 or 2 represents IN1, IN2 and IN3 respectively.

Action Codes are listed below, that can be attached to the rising/falling edge of the inputs:

Action Code	Action Description	Illustration
0x00	No action	n/a
0x01	Turn the Motor Driver off, freewheel	
0x02	Emergency stop (using SD)	
0x03	Decelerating to stop (using DC)	
0x04	Set origin then go reversed relative position (using |PR|, SP, AC, DC)	
0x05	Set origin then Emergency stop (using SD)	
0x06	Set origin then decelerate to stop (using DC)	
0x07	Reversed Jogging (using |SP|, AC, DC)	
0x08	Jog (using SP, AC, DC)	
0x09	Go reverse relative position (using |PR|, SP, AC, DC)	
0x0A	Go relative position (using PR, SP, AC, DC)	
0x0B	Go absolute position (using PA, SP, AC, DC)	

4.4 Real-time Notification for Input Change

When the logic level change on a specific input port is detected, there will be a real-time notification message sent to user. That message could be enabled or disabled by setting "`IE[i]`=1" or "`IE[i]`=0" respectively. Here, i=0, 1 or 2 represents IN1, IN2 and IN3 respectively.

5.0 Instruction Set

This chapter provides details of instructions supported by UIM342.

Note:

1. Unless otherwise specified, all message bytes are in hex format;
2. Examples assume that the UIM342 address is 5 (ID=5) ;
3. Abbreviation definitions as below:
   a) INST – Instruction
   b) ACK – Acknowledgment / Reply

5.1 PP[i] Protocol Parameter

Protocol Parameter

CW	Disable ACK	0x01	Request ACK	0x81
DATA	Sub-Index(i)	unsigned 8 bit	Data	unsigned 8 bit

Description

PP[i] **Get Protocol Parameters**

INST data length 1 Data d0 (=i)

ACK data length 2 Data d0 (=i), d1

PP[i]=N **Set Protocol Parameters**

INST data length 2 Data d0 (=i), d1

ACK data length 2 Data d0 (=i), d1

i	Description	Value (N)
5	CAN bit rate (bps)	0: 1000K 1: 800K 2: 500K 3: 250K 4: 125K
7	Node ID	5…126
8	Group ID	5…126

Example GET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	81	01	08	00	00	00	00	00	00	00

Sub-index d0 = 8 (Get Group ID).

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	01	02	08	0A	00	00	00	00	00	00

Sub-index d0 = 8 (Group ID). Data d1 = 0x0A (=10).

Example SET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	81	02	05	02	00	00	00	00	00	00

Sub-index d0 = 5 (CAN bit rate). Data d1 = 2 (SET 500K).

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	01	02	05	02	00	00	00	00	00	00

Sub-index d0 = 5 (CAN bit rate). Data d1 = 2 (=500K).

Note

- The value of PP[i] will be automatically saved to FLASH memory with a 10,000 write lifespan.
- Within a specific CAN network, Node IDs and Group IDs of all UIM devices should never be overlapped.
- Protocol Parameters will take effectiveness after reboot the UIM device.

5.2 IC[i] Initial Configuration

System Settings

CW	Disable ACK	0x06	Request ACK	0x86
DATA	Sub-Index(i)	unsigned 8 bit	Data	unsigned 16 bit

Description

IC[i] **Get initial state configuration of UIM342 when Power-On**

INST data length 1 Data d0 (=i)

ACK data length 3 Data d0 (=i), d1, d2

IC[i]=N **Set initial state configuration of UIM342 when Power-On**

INST data length 3 Data d0 (=i), d1, d2

ACK data length 3 Data d0 (=i), d1, d2

i	Description	Value (N)	
0	Turn motor driver on after power is on	0: disable	1: enable;
1	Positive motor direction	0: CW	1: CCW
2	Execute the user program after power is on	0: NO	1: YES
3	Lock down the system when Input Logic E-Stop is triggered.	0: disable	1: enable
4	Units of AC and DC	0: pulse/sec^2	1: millisecond
5	Encoder type	0: Incremental	1: Absolute
6	Using Closed-loop control	0: Open loop	1: Closed-loop
7	Enable software limit	0: disable	1: enable

Example GET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	86	01	00	00	00	00	00	00	00	00

Sub-index d0 = 0 (Turn motor driver on after power is on).

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	06	03	00	01	00	00	00	00	00	00

Sub-index d0 = 0 (Turn motor driver on after power is on);
Data [d2:d1] = 0x0001 (Enable).

Example SET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	86	03	06	01	00	00	00	00	00	00

Sub-index d0 = 6 (Using Closed-loop control);
Data [d2:d1] = 0x0001 (SET Closed-loop control).

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	06	03	06	01	00	00	00	00	00	00

Sub-index d0 = 6 (Using Closed-loop control);
Data [d2:d1] = 0x0001 (Closed-loop control).

Note

- Once "IC[i]=N" is set, the UIM device will reboot. Getting "IC[i]" will not cause a reboot.
- When using an absolute encoder, IC[1] must be 1 .
- Once the System is locked down, only reboot will unlock the system.

5.3 IE[i] Information Enable

System Settings

CW	Disable ACK	0x07	Request ACK	0x87
DATA	Sub-Index(i)	unsigned 8 bit	Data	unsigned 16 bit

Description

IE[i] **Get Information Enable Configuration**

INST data length 1 Data d0 (=i)

ACK data length 3 Data d0 (=i), d1, d2

IE[i]=N **Set Information Enable Configuration**

INST data length 3 Data d0 (=i), d1, d2

ACK data length 3 Data d0 (=i), d1, d2

i	Description	Value (N)	
0	Port IN1 change notification	0: disable	1: enable
1	Port IN2 change notification	0: disable	1: enable
2	Port IN3 change notification	0: disable	1: enable
8	PTP positioning finish notification	0: disable	1: enable

Example GET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	87	01	00	00	00	00	00	00	00	00

Sub-index d0 = 0 (Get IN1 I.E. configure).

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	07	03	00	01	00	00	00	00	00	00

Sub-index d0 = 0 (IN1 I.E. configure); Data [d2:d1] = 0x0001 (Enable).

Example SET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	87	03	01	01	00	00	00	00	00	00

Sub-index d0 = 1 (IN2 I.E. configure); Data [d2:d1] = 0x0001 (SET Enable).

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	07	03	01	01	00	00	00	00	00	00

Sub-index d0 = 0 (IN2 I.E. configure); Data [d2:d1] = 0x0001 (Enabled).

Note

- Set/Get the information enable configuration.
- To receive the real-time change notification, the corresponding information enable bit must be set (=1).
- Refer to chapter 2.4 for more details about real-time change notification.

5.4 ML Model

System Settings

CW	Disable ACK	n/a	Request ACK	0x8B
DATA	n/a			

Description

ML **Get the model, function module and firmware version**

INST data length	0	Data	n/a
ACK data length	8	Data	d0, d1, d2, d3, d4, d5, d6, d7

Example GET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	8B	00	00	00	00	00	00	00	00	00

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	0B	08	20	0A	11	00	00	00	00	00

For UIM342 series, please refer to the following table for model information.

model	d0	d1	d2	d3	d5:d4	d7:d6
342XS	0x20	0x0A	0x11	xx	Firmware version	xxxx
342S	0x21	0x0A	0x11	xx	Firmware version	xxxx
342C02	0x22	0x14	0x13	xx	Firmware version	xxxx
342C04	0x22	0x28	0x13	xx	Firmware version	xxxx
342C08	0x22	0x50	0x13	xx	Firmware version	xxxx
342H08	0x22	0x50	0x19	xx	Firmware version	xxxx

x – Factory use, don't care.

5.5 SN Serial Number

System Settings

CW	Disable ACK	0x0C	Request ACK	0x8C
DATA	n/a			
Description	**SN** **Get the serial number of the device** INST data length 0 Data n/a ACK data length 8 Data d0, d1, d2, d3, d4, d5, d6, d7			

Example GET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	8C	00	00	00	00	00	00	00	00	00

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	0C	08	01	02	03	04	05	06	07	08

Data
[d3:d2:d1:d0] = 0x04030201 (Serial number 0067305985).
[d5:d4] = 0x0605 (Manufacturer ID 1541).
[d7:d6] = 0x0807 (Vendor ID 2055).

5.6 ER[i] Error Report

System Settings

CW	Disable ACK	0x0F	Request ACK	0x8F
DATA	Sub-Index(i)	unsigned 8 bit	Data	unsigned 8 bit

Description

ER[i] **Get Error Report**

INST data length 1 Data d0 (=i)

ACK data length 6 Data d0 (=i), d1, d2, d3, d4, d5

ER[i]=0 **Clear Error Report**

INST data length 2 Data d0 (=i), d1, d2

ACK data length 6 Data d0 (=i), d1, d2, d3, d4, d5

i	Description
0	Get / Clear The latest error content
6	Get / Clear Power on error
10	Get / Clear latest errors
11…18	Get / Clear the last 2…9 errors

Refer to chapter 2.3 “Error Report” for the description of d0, d1, d2, d3, d4, d5.

Example GET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	8F	01	00	00	00	00	00	00	00	00

Sub-index d0 = 0 (Get the newest error).

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	0F	06	00	33	81	05	00	00	00	00

Sub-index d0 = 0 (The newest error).

Data

d1 = 0x33 (Error code = 0x33, i.e. instruction data error).

d2 = 0x81 (CW related to the error = 0x81, i.e. PP).

d3 = 0x05 (Sub-Index of the CW related to the error = 5).

d4, d5 = 0x00 (Reserve).

The newest error = "In the instruction PP[5]=N, the data is illegal".

Example SET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	8F	02	00	00	00	00	00	00	00	00

Sub-index d0 = 0 (Latest error); Data d1 = 0 (SET clear).

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	0F	06	00	00	00	00	00	00	00	00

Sub-index d0 = 0 (Latest error); Data d5…d1 = 0 (Cleared, no errors).

5.7 QE[i] Quadrature Encoder

System Settings

CW	Disable ACK	0x3D	Request ACK	0xBD
DATA	Sub-Index(i)	unsigned 8 bit	Data	unsigned 16 bit

Description

QE[i] **Get Encoder Parameters**

INST data length 1 Data d0 (=i)
ACK data length 3 Data d0 (=i), d1, d2

QE[i]=N **Set Encoder Parameters**

INST data length 3 Data d0 (=i), d1, d2
ACK data length 3 Data d0 (=i), d1, d2

i	Description	Value (N)
0	Lines per revolution of encoder (LPR)	1..65535,
1	Stall tolerance	10..65535 counts
2	Single-turn bits of absolute encoder	17, 20.bits
3	Battery status of absolute encoder	1: OK 0: Low
4	Counts per revolution (CPR = Micro Steps*200)	200..25600 (1..128*200)

Example GET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	BD	04	04	00	00	00	00	00	00	00

Sub-index d0 = 4 (Counts per revolution for closed-loop control).

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	3D	03	04	80	0C	00	00	00	00	00

Sub-index d0 = 4 (Counts per revolution for closed-loop control);
Data [d2:d1] = 0x0C80 (3200).

Example SET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	BD	03	00	E8	03	00	00	00	00	00

Sub-Index d0 = 0 (Lines per revolution of encoder);
Data [d2:d1] = 0x03E8 (SET 1000).

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	3D	03	00	E8	03	00	00	00	00	00

Sub-Index d0 = 0 (Lines per revolution of encoder);
Data [d2:d1] = 0x03E8 (1000).

Note

- QE[0] LPR is a key physical property of the encoder. Setting it incorrectly can cause unpredictable behavior.
- Save QE[i] to EEPROM (EEP) only when the motor driver is off. If saved while the motor driver is on, QE[i] will remain in RAM and be lost upon power off.
- Adjust QE[4] CPR by changing the micro-stepping resolution (MT[0]). After adjustment, power off and restart the system for changes to take effect.

5.8 SY[i] System Operation

System Settings

CW	Disable ACK	0x7E	Request ACK	n/a
DATA	Sub-Index(i)	unsigned 8 bit	Data	n/a

Description

SY[i] System Operation

INST data length 1 Data d0 (=i)

No ACK

i	Description
1	Reboot the system
2	Restore factory defaults

Example SET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	7E	01	01	00	00	00	00	00	00	00

Sub-Index d0 = 1 (Reboot the device).

ACK : n/a

5.9 MT[i] Motor Driver

Motor Driver

CW	Disable ACK	0x10	Request ACK	0x90
DATA	Sub-Index(i)	unsigned 8 bit	Data	unsigned 16 bit

Description

MT[i] **Get Motor Drive Parameters**

INST data length 1 Data d0 (=i)
ACK data length 3 Data d0 (=i), d1, d2

MT[i]=N **Set Motor Drive Parameters**

INST data length 3 Data d0 (=i), d1, d2
ACK data length 3 Data d0 (=i), d1, d2

i	Description	Value (N)
0	Micro-stepping resolution	1/2/4/8/16/32/64
1	Working current	5 ... 80 (= 0.5 ... 8.0 Amp)
2	Percentage of idle current over working current	0...100 (= 0...100%)
3	Delay of automatic enable after power-on	0...60000 (millisecond)

Example GET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	90	01	01	00	00	00	00	00	00	00

Sub-index d0 = 1 (Working current).

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	10	03	01	1C	00	00	00	00	00	00

Sub-index d0 = 1 (Working current);
Data [d2:d1] = 0x001C (2.8 Amp).

Example SET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	90	01	00	10	00	00	00	00	00	00

Sub-index d0 = 0 (Micro-stepping resolution);
Data [d2:d1] = 0x0010 (SET 16).

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	10	03	00	10	00	00	00	00	00	00

Sub-index d0 = 0 (Micro-stepping resolution);
Data [d2:d1] = 0x0010 (16).

Note

Only when the Motor Driver is off, value of MT[i] will be saved to EEP. Otherwise, MT[i] will only exist in RAM and will be lost when power off.

When setting micro-stepping resolution, counts per revolution for closed-loop control needs to be set simultaneously. After the setting is completed, it needs to be powered off and restarted to take effect.

5.10 MO Motor Driver On /Off

Motor Driver

CW	Disable ACK	0x15	Request ACK	0x95
DATA	unsigned 8 bit			

Description

MO **Get Motor Drive ON/OFF status**

INST data length	0	Data	n/a
ACK data length	1	Data	d0

MO=N **Set Motor Drive ON/OFF**

INST data length	1	Data	d0
ACK data length	1	Data	d0

N = 0: OFF;
N = 1: ON.

Example GET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	95	00	00	00	00	00	00	00	00	00

Get motor driver on/off status.

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	15	01	01	00	00	00	00	00	00	00

Data d0 = 1 (Motor driver is ON).

Example SET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	95	01	00	00	00	00	00	00	00	00

Data d0 = 0 (SET motor driver OFF).

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	15	01	00	00	00	00	00	00	00	00

Data d0 = 0 (Motor driver is OFF).

Note

- Make sure the motor driver is ON (MO=1), before sending instructions such as SP, PR, JV and BG etc. Otherwise, unexpected movement will happen (caused by previous residual actions) when turning on the motor driver.
- The motor can be driven only after the motor driver is ON.
- In OFF status, the motor is freewheel, but the logic circuit still works.

5.11 BG Begin Motion

Motion Control

CW	Disable ACK	0x16	Request ACK	0x96
DATA	n/a			
Description	**BG** **Begin Motion** INST data length 0 Data n/a ACK data length 4 Data d0, d1, d2, d3			
Example GET	INST ID CW DL d0 d1 d2 d3 d4 d5 d6 d7 05 96 00 00 00 00 00 00 00 00 Begin Motion. ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7 05 16 04 00 00 00 00 00 00 00 00 Data d0…d3 = 0 (don't care).			
Note	Activate the newly set parameters and start moving. Applied to PTP and JOG motion control. The motor will not move until BG is entered. Motion parameters (e.g. SP/PR/PA/JV) will then be activated and motion will begin. When multiple UIM342 are to move at the same time, user can set the parameters like SP/PA to each node ID first, and then send BG to the global ID or group ID.			

5.12 ST Stops Motion

Motion Control

CW	Disable ACK	0x17	Request ACK	0x97
DATA	n/a			
Description	**ST** **Stop Motion** INST data length 0 Data n/a ACK data length 0 Data n/a			
Example GET	INST ID CW DL d0 d1 d2 d3 d4 d5 d6 d7 05 97 00 00 00 00 00 00 00 00 Stop Motion. ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7 05 17 00 00 00 00 00 00 00 00			
Note	Stop current movement, using stop deceleration (SD) to decelerate.			

5.13 MF Motion Parameter Frame

Motion Control

CW	Disable ACK	0x18	Request ACK	0x98
DATA	unsigned 8 bit			

Description

MF=N **Set/select motion parameter group to be operated.**

INST data length 1 Data d0

ACK data length 1 Data d0

N	Description
0	Select the motion parameter group for normal operation
1	Reserved
2	Select the motion parameter group for the rising edge of IN1
3	Select the motion parameter group for the falling edge of IN1
4	Select the motion parameter group for the rising edge of IN2
5	Select the motion parameter group for the falling edge of IN2
6	Select the motion parameter group for the rising edge of IN3
7	Select the motion parameter group for the falling edge of IN3

Example SET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	98	01	03	00	00	00	00	00	00	00

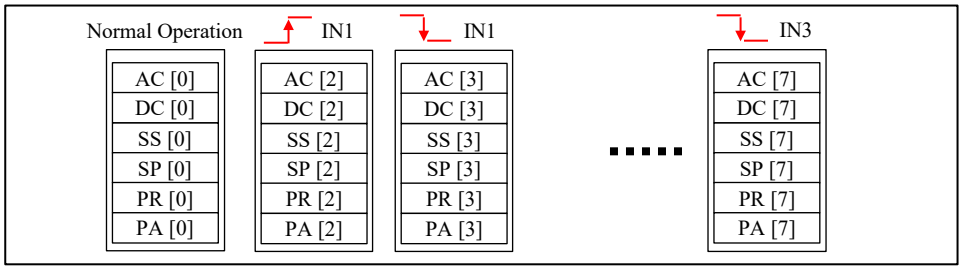
Data d0 = 3 (Select the motion parameter group for the falling edge of IN1).

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	18	01	03	00	00	00	00	00	00	00

Data d0 = 3 (The motion parameter group for the falling edge of IN1).

Note

The UIM342 supports different motion parameter groups for preset actions triggered by different sources. MF=N provides a method to use same instruction (e.g. AC) to assign a motion parameter to different groups.



- For example, to set the acceleration of IN1 falling edge to 1000, using: MF=1; AC=1000; To get the acceleration of IN1 falling edge, using: MF=1; AC;
- After setting/getting a parameter, MF value will automatically reset to 0.

5.14 AC Acceleration

Motion Control

CW	Disable ACK	0x19	Request ACK	0x99
DATA	unsigned 32 bit			

Description

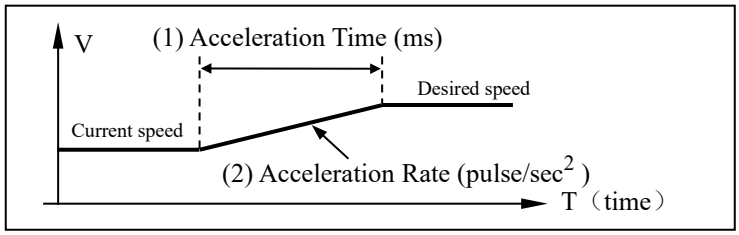
AC **Get Acceleration**

INST data length	0	Data	n/a
ACK data length	4	Data	d0, d1, d2, d3

AC=N **Set Acceleration**

INST data length	4	Data	d0, d1, d2, d3
ACK data length	4	Data	d0, d1, d2, d3

When IC[4]=0, acceleration is defined as rate, N=1…65,000,000 (pulse/sec^2).
When IC[4]=1, acceleration is defined as time, N=1…60,000 (ms)



Example GET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	99	00	00	00	00	00	00	00	00	00

Get AC.

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	19	04	E8	03	00	00	00	00	00	00

Data [d3:d2:d1:d0] = 0x03e8 (The acceleration is 1000).

Example SET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	99	00	F4	01	00	00	00	00	00	00

Data [d3:d2:d1:d0] = 0x01f4 (SET acceleration 500).

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	19	04	F4	01	00	00	00	00	00	00

Data [d3:d2:d1:d0] = 0x01f4 (The acceleration is 500).

Note

- Only when the Motor Driver is off, value of AC will be saved to EEP. Otherwise, AC will only exist in RAM and will be lost when power off.
- After setting, AC will takes effect when the next BG is commanded.

5.15 DC Deceleration

Motion Control

CW	Disable ACK	0x1A	Request ACK	0x9A
DATA	unsigned 32 bit			

Description

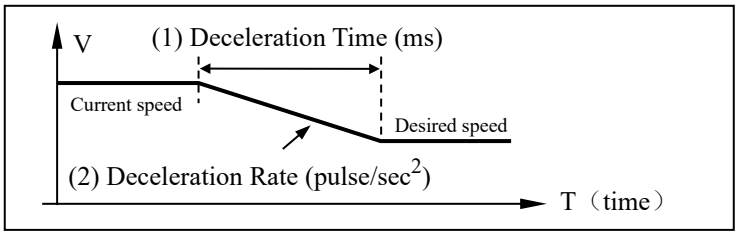
DC **Get Deceleration**

INST data length	0	Data	n/a
ACK data length	4	Data	d0, d1, d2, d3

DC=N **Set Deceleration**

INST data length	4	Data	d0, d1, d2, d3
ACK data length	4	Data	d0, d1, d2, d3

When IC[4]=0, deceleration is defined as rate, N=1…65,000,000 (pulse/sec^2).
When IC[4]=1, deceleration is defined as time, N=1…60,000 (ms)



Example GET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	9A	00	00	00	00	00	00	00	00	00

Get DC.

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	1A	04	E8	03	00	00	00	00	00	00

Data [d3:d2:d1:d0] = 0x03e8 (The deceleration is 1000).

Example SET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	9A	04	E9	03	00	00	00	00	00	00

Data [d3:d2:d1:d0] = 0x03e9 (SET deceleration 1001).

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	1A	04	E9	03	00	00	00	00	00	00

Data [d3:d2:d1:d0] = 0x03e9 (The deceleration is 1001).

Note

- Only when the Motor Driver is off, value of DC will be saved to EEP. Otherwise, DC will only exist in RAM and will be lost when power off.
- After setting, DC will takes effect when the next BG is commanded.

5.16 SS Cut-in Speed

Motion Control

CW	Disable ACK	0x1B	Request ACK	0x9B
DATA	unsigned 32 bit			

Description

SS **Get Cut-in speed**

INST data length	0	Data	n/a
ACK data length	4	Data	d0, d1, d2, d3

SS=N **Set Cut-in speed**

INST data length	4	Data	d0, d1, d2, d3
ACK data length	4	Data	d0, d1, d2, d3

N = 0... 2^{32} pulse/sec.

Example GET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	9B	00	00	00	00	00	00	00	00	00

Get SS.

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	1B	04	E8	03	00	00	00	00	00	00

Data [d3:d2:d1:d0] = 0x03e8 (Cut-in Speed is 1000).

Example SET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	9B	04	E9	03	00	00	00	00	00	00

Data [d3:d2:d1:d0] = 0x03e9 (SET cut-in speed 1001).

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	1B	04	E9	03	00	00	00	00	00	00

Data [d3:d2:d1:d0] = 0x03e9 (The cut-in speed is 1001).

Note

- Only when the Motor Driver is off, value of SS will be saved to EEP. Otherwise, SS will only exist in RAM and will be lost when power off.

5.17 SD Stop Deceleration

Motion Control

CW	Disable ACK	0x1C	Request ACK	0x9C
DATA	unsigned 32 bit			

describe

SD **Get Stop Deceleration**

INST data length	0	Data	n/a
ACK data length	4	Data	d0, d1, d2, d3

SD=N **Set Stop Deceleration**

INST data length	4	Data	d0, d1, d2, d3
ACK data length	4	Data	d0, d1, d2, d3

N = 0... 2^{32} pulse/sec^2.

Example GET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	9C	00	00	00	00	00	00	00	00	00

Get SD.

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	1C	04	00	6A	18	00	00	00	00	00

Data [d3:d2:d1:d0] = 0x00186A00 (Stop deceleration is 1,600,000).

Example SET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	9C	04	00	6A	18	00	00	00	00	00

Data [d3:d2:d1:d0] = 0x00186A00 (SET stop deceleration 1,600,000).

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	1C	04	00	6A	18	00	00	00	00	00

Data [d3:d2:d1:d0] = 0x00186A00 (Stop deceleration is 1,600,000).

Note

Mechanical movement cannot be stopped suddenly, otherwise it may cause impact and damage to the machine. Therefore a suitable maximum allowable acceleration value must be set. The SD value is designed for that purpose.

The following situations will trigger the use of SD value:

1) Instruction ST;
2) The input logic of emergency stop is triggered;
3) Other situation need the emergency stop, such as soft bumper limit is hit.

- Only when the Motor Driver is off, value of SD will be saved to EEP. Otherwise, SD will only exist in RAM and will be lost when power off.

5.18 JV Jog Velocity

Motion Control

CW	Disable ACK	0x1D	Request ACK	0x9D
DATA	signed 32-bit			

Description

JV **Get Current Speed**

INST data length	0	Data	n/a
ACK data length	4	Data	d0, d1, d2, d3

JV=N **Set Desired Speed + Set Speed (Jog) Mode**

INST data length	4	Data	d0, d1, d2, d3
ACK data length	5	Data	d0, d1, d2, d3, d4

N = -2^{31}...$+2^{31}$ pulse/sec.

Example GET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	9D	00	00	00	00	00	00	00	00	00

Get current speed.

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	1D	04	E8	03	00	00	00	00	00	00

Data [d3:d2:d1:d0] = 0x000003e8 (Current speed 1000).

Example SET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	9D	04	18	FC	FF	FF	00	00	00	00

Data [d3:d2:d1:d0] = 0xfffffc18 (SET desired speed -1000).

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	2E	05	02	18	FC	FF	FF	00	00	00

CW = 0x2E (i.e. DV);
Sub-index d0 = 2 (Desired speed).
Data [d4:d3:d2:d1] = 0xfffffc18 (Desired speed -1000).

Note

- Instruction JV sets the desired motor speed and switch the control mode to speed mode.
- JV will take effective after a following BG instruction.
- Reply of the JV=N is the desired value DV[2]
- Once set, if want to get the desired value of speed, DV[2] should be used.

5.19 SP PTP Speed

Motion Control

<table>
<tr><td>CW</td><td>Disable ACK</td><td>0x1E</td><td>Request ACK</td><td>0x9E</td></tr>
<tr><td>DATA</td><td colspan="4">signed 32-bit</td></tr>
<tr><td>Description</td><td colspan="4">
SP Get Current Speed
INST data length 0 Data n/a
ACK data length 4 Data d0, d1, d2, d3

SP=N Set Desired Speed
INST data length 4 Data d0, d1, d2, d3
ACK data length 5 Data d0, d1, d2, d3, d4

N = -2^{31}...$+2^{31}$ pulse/sec.
</td></tr>
<tr><td>Example
GET</td><td colspan="4">
<table>
<tr><td>INST</td><td>ID</td><td>CW</td><td>DL</td><td>d0</td><td>d1</td><td>d2</td><td>d3</td><td>d4</td><td>d5</td><td>d6</td><td>d7</td></tr>
<tr><td></td><td>05</td><td>9E</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td></tr>
</table>
Get current speed.
<table>
<tr><td>ACK</td><td>ID</td><td>CW</td><td>DL</td><td>d0</td><td>d1</td><td>d2</td><td>d3</td><td>d4</td><td>d5</td><td>d6</td><td>d7</td></tr>
<tr><td></td><td>05</td><td>1E</td><td>04</td><td>18</td><td>FC</td><td>FF</td><td>FF</td><td>00</td><td>00</td><td>00</td><td>00</td></tr>
</table>
Data [d3:d2:d1:d0] = 0xfffffc18 (Current speed -1000).
</td></tr>
<tr><td>Example
SET</td><td colspan="4">
<table>
<tr><td>INST</td><td>ID</td><td>CW</td><td>DL</td><td>d0</td><td>d1</td><td>d2</td><td>d3</td><td>d4</td><td>d5</td><td>d6</td><td>d7</td></tr>
<tr><td></td><td>05</td><td>9E</td><td>04</td><td>E8</td><td>03</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td></tr>
</table>
Data [d3:d2:d1:d0] = 0x000003e8 (SET desired speed +1000).
<table>
<tr><td>ACK</td><td>ID</td><td>CW</td><td>DL</td><td>d0</td><td>d1</td><td>d2</td><td>d3</td><td>d4</td><td>d5</td><td>d6</td><td>d7</td></tr>
<tr><td></td><td>05</td><td>2E</td><td>05</td><td>02</td><td>E8</td><td>03</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td></tr>
</table>
CW = 0x2E (i.e. DV);
Sub-index d0 = 2 (Desired speed).
Data [d4:d3:d2:d1] = 0x000003e8 (Desired speed +1000).
</td></tr>
<tr><td>Note</td><td colspan="4">
• In PTP mode, PR / PA should be set first, and then SP, otherwise error #50 will be returned.
• To set input logic speed, SP (not JV) should be used.
</td></tr>
</table>

5.20 PR Position Relative

Motion Control

CW	Disable ACK	0x1F	Request ACK	0x9F
DATA	signed 32-bit			

Description

PR **Get Current Relative Position**

INST data length	0	Data	n/a
ACK data length	4	Data	d0, d1, d2, d3

PR=N **Set Desired Relative Position + Set Position (PTP) Mode**

INST data length	4	Data	d0, d1, d2, d3
ACK data length	5	Data	d0, d1, d2, d3, d4

N = -2^{31}...$+2^{31}$ pulse.

Example GET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	9F	00	00	00	00	00	00	00	00	00

Get current relative position.

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	1F	04	18	FC	FF	FF	00	00	00	00

Data [d3:d2:d1:d0] = 0xfffffc18 (Current relative position -1000).

Example SET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	9E	04	E8	03	00	00	00	00	00	00

Data [d3:d2:d1:d0] = 0x000003e8 (SET desired relative position +1000).

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	2E	05	03	E8	03	00	00	00	00	00

CW = 0x2E (i.e. DV);
Sub-index d0 = 3 (Desired relative position).
Data [d4:d3:d2:d1] = 0x000003e8 (Expected relative position +1000).

Note

- Relative Position is the displacement added to the current position.
- In PTP mode, PR / PA should be set first, and then SP, otherwise error #50 will be returned.
- PR will take effective after a following BG instruction.

5.21 PA Position Absolute

Motion Control

CW	Disable ACK	0x20	Request ACK	0xA0
DATA	signed 32-bit			

Description

PA **Get Current Absolute Position**

INST data length	0	Data	n/a
ACK data length	4	Data	d0, d1, d2, d3

PA=N **Set Desired Absolute Position + Set Position (PTP) Mode**

INST data length	4	Data	d0, d1, d2, d3
ACK data length	5	Data	d0, d1, d2, d3, d4

N = -2^{31}...$+2^{31}$ pulse.

Example GET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	A0	00	00	00	00	00	00	00	00	00

Get current absolute position.

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	20	04	18	FC	FF	FF	00	00	00	00

Data [d3:d2:d1:d0] = 0xfffffc18 (Current absolute position -1000).

Example SET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	A0	04	E8	03	00	00	00	00	00	00

Data [d3:d2:d1:d0] = 0x000003e8 (SET desired absolute position +1000).

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	2E	05	04	E8	03	00	00	00	00	00

CW = 0x2E (i.e. DV);
Sub-index d0 = 4 (Desired absolute position).
Data [d4:d3:d2:d1] = 0x000003e8 (Expected absolute position +1000).

Note

- In PTP mode, PR / PA should be set first, and then SP, otherwise error #50 will be returned.
- PA will take effective after a following BG instruction.

5.22 OG Set Origin

Motion Control

CW	Disable ACK	0x21	Request ACK	0xA1
DATA	n/a			
Description	**OG** **Set The Origin, i.e. clear the position counter** INST data length 0 Data n/a ACK data length 0 Data n/a			
Example SET				

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	A1	00	00	00	00	00	00	00	00	00

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	21	00	00	00	00	00	00	00	00	00

Note

- Set the current position as the origin and clear the position counter to zero.
- When the motor is moving, instructions OG is disabled. But during motion, it is possible for input logic to set the origin.

5.23 BL Backlash Compensation

Motion Control

CW	Disable ACK	0x2D	Request ACK	0xAD
DATA	unsigned 32 bit			

Description

BL **Get Backlash Compensation Value**

INST data length	0	Data	n/a
ACK data length	4	Data	d0, d1, d2, d3

BL=N **Set Backlash Compensation Value**

INST data length	4	Data	d0, d1, d2, d3
ACK data length	4	Data	d0, d1, d2, d3

N = 0...+2^{32} pulse.

Example GET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	AD	00	00	00	00	00	00	00	00	00

Get current absolute position.

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	2D	04	E8	03	00	00	00	00	00	00

Data [d3:d2:d1:d0] = 0x03e8 (The backlash compensation is 1000).

Example SET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	AD	04	E9	03	00	00	00	00	00	00

Data [d3:d2:d1:d0] = 0x03e9 (SET backlash compensation 1001).

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	2D	04	E9	03	00	00	00	00	00	00

Data [d3:d2:d1:d0] = 0x000003e9 (The backlash compensation is 1001).

5.24 MS[i] Motion Status

Motion Control

CW	Disable ACK	0x11	Request ACK	0x91
DATA	Sub-Index(i)	unsigned 8 bit	Data	unsigned 8 bit

Description

MS[0] **Get Status Flags and Relative Position**

INST data length 1 Data d0=0;
ACK data length 8 Data d0=0, d1. . .d7;

MS[1] **Get the Current Speed and Absolute Position**

INST data length 1 Data d0=1;
ACK data length 8 Data d0=1, d1. . .d7;

MS[0]=0 **Clear Status Flags**

INST data length 2 Data d0=0, d1=0;
ACK data length 8 Data d0=0, d1. . .d7;

Please refer to chapter 3.7 “Acquire Motion Status” for details.

Example GET

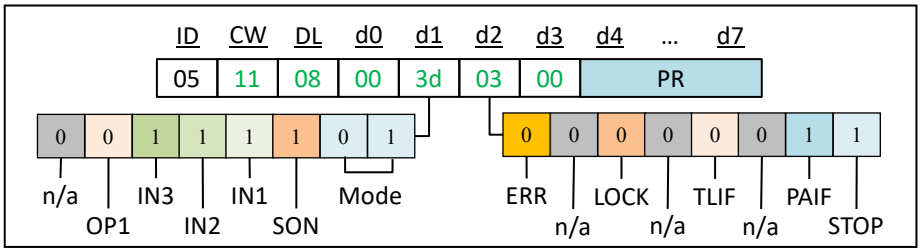
INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	91	01	00	00	00	00	00	00	00	00

Sub-Index d0 = 0 (Get the current motion status).

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	11	08	00	3D	03	00	18	FC	FF	FF

Sub-Index d0 = 0 (Current motion status);

Data



Motion mode = PTP; Motor driver = ON; IN1/2/3 level = 1; OP1 level = 0;
Speed is zero = YES; PTP Positioning = In Position;
Current relative position [d7:d6:d5:d4] = 0xfffffc18 = -1000 (pulse);

Example SET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	91	02	00	00	00	00	00	00	00	00

Sub-Index d0 = 0; Data d1 = 0 (Clear motion status flags).

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	11	08	00	3D	01	00	00	00	00	00

Sub-Index d0 = 0 (Current motion status). Data (See Example GET)

5.25 DV[i] Desired Values

Motion Control

CW	Disable ACK	0x2E	Request ACK	0xAE
DATA	Sub-Index(i)	unsigned 8 bit	Data	n/a

Description

DV[i] **Get Desired Value**

INST data length 1 Data d0(=i);

ACK data length 5 Data d0(=i), d1. . .d4;

i	Description	Data [d4:d3:d2:d1]
0	Current motion mode	0: JOG, 1: PTP
1	Desired motor current	0…80 : 0.0…8.0 Amp
2	Desired speed	-2^{31}...$+2^{31}$ pulse/sec
3	Desired relative position	-2^{31}...$+2^{31}$ pulse
4	Desired absolute position	-2^{31}...$+2^{31}$ pulse

Example GET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	AE	01	02	00	00	00	00	00	00	00

Sub-Index d0 = 2 (Get Desired speed).

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	2E	05	02	E8	03	00	00	00	00	00

Sub-index d0 = 2 (Desired speed).

Data [d4:d3:d2:d1] = 0x000003e8 (Desired speed +1000).

Note

5.26 IL[i] Input Logic

Input Logic

CW	Disable ACK	0x34	Request ACK	0xB4
DATA	Sub-Index(i)	unsigned 8 bit	Data	unsigned 16 bit

Description

IL[i] Get Sensor Trigger Action

INST data length 1 Data d0 (=i)
ACK data length 3 Data d0 (=i), d1, d2

IL[i]=N Set Sensor Trigger Action

INST data length 3 Data d0 (=i), d1, d2
ACK data length 3 Data d0 (=i), d1, d2

i	Description
0	Action after IN1 is triggered
1	Action after IN2 is triggered
2	Action after IN3 is triggered
3	Behavior when motor stalls, N=0: lock down, N=1: freewheel

Please refer to chapter 4.3 “Configure / Attach Input Logic Action” for details.

Example GET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	B4	01	00	00	00	00	00	00	00	00

Sub-index d0 = 0 (Get IN1 port trigger action).

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	34	03	00	06	80	00	00	00	00	00

Sub-index d0 = 0 (IN1 port trigger action).

Data d1, d2 = 0x80 (Both edge: Power on trigger action: disabled, no action).

Example SET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	B4	03	00	06	80	00	00	00	00	00

Sub-index d0 = 0 (IN1 port trigger action).

d1 = 0x06 (Falling edge: Power on trigger action: enable; Set origin then decelerate to stop).

d2 = 0x80 (Rising edge: Power on trigger action: disable, no action).

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	34	03	00	06	80	00	00	00	00	00

Sub-index d0 = 0 (IN1 port trigger action).

Data d1 = 0x06 (same as above); d2 = 0x80 (same as above).

Note

- Define actions when input port logic level change is detected.
- When the Motor Driver is off, set value of IL will be saved to EEP. Otherwise, IL value will be only in the RAM.

5.27 TG[i] Trigger

Input Logic

CW	Disable ACK	0x35	Request ACK	0xB5
DATA	Sub-Index(i)	unsigned 8 bit	Data	unsigned 16 bit

Description

TG[i] Get The Input Trigger Mode

INST data length 1 Data d0 (=i)
ACK data length 3 Data d0 (=i), d1, d2

TG[i]=N Set The Input Trigger Mode

INST data length 3 Data d0 (=i), d1, d2
ACK data length 3 Data d0 (=i), d1, d2

i	Description
0	Trigger mode of IN1
1	Trigger mode of IN2
2	Trigger mode of IN3

N	Description
0	Trigger continuously
1 … 60000	Trigger after Low pass filter, filter time 1…60000 ms
60001	One-Time Trigger. After triggered, IL[i] needs to be set again to work.

Please refer to chapter 4.2 “Trigger Modes” for details.

Example GET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	B5	01	01	00	00	00	00	00	00	00

Sub-index d0 = 1 (Get IN2 trigger mode).

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	35	03	00	00	00	00	00	00	00	00

Sub-index d0 = 1 (IN2); Data [d2:d1] = 0 (Trigger continuously).

Example SET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	B5	03	00	64	00	00	00	00	00	00

Sub-index d0 = 1 (IN1); Data [d2:d1] = 0x0064 (Low-pass filter time 100 ms).

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	35	03	00	64	00	00	00	00	00	00

Sub-index d0 = 1 (IN1); Data [d2:d1] = 0x0064 (Low-pass filter time 100 ms).

Note

- Set trigger mode for input ports.
- When the Motor Driver is off, set value of TG will be saved to EEP. Otherwise, TG value will be only in the RAM.

5.28 DI Digital I /O

Input Logic

CW	Disable ACK	0x37	Request ACK	0xB7
DATA	unsigned 32 bit			

Description

DI **Get Digital Input And Output**

INST data length	0	Data	n/a
ACK data length	4	Data	d0, d1, d2(=0), d3(=0)

DI=N **Set Digital Output**

INST data length	4	Data	d0, d1(=1), d2(=0), d3(=0) d0 = desired output
ACK data length	4	Data	d0, d1, d2(=0), d3(=0)

Example GET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	B7	00	00	00	00	00	00	00	00	00

Get input and output logic level.

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	37	04	03	00	00	00	00	00	00	00

d0.bit0 = 1 (IN1 = 1) ; d0.bit1 = 1 (IN2 = 1) ; d0.bit2 = 0 (IN3 = 0) ;
d1.bit0 = 0 (OP1 = 0).

Example SET

INST	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	B7	04	01	01	00	00	00	00	00	00

Set output logic level to 1.

ACK	ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
	05	37	04	03	01	00	00	00	00	00	00

d0.bit0 = 1 (IN1 = 1) ; d0.bit1 = 1 (IN2 = 1) ; d0.bit2 = 0 (IN3 = 0) ;
d1.bit0 = 0 (OP1 = 1).

5.29 RT Real-Time Inform

Real-time Notification

CW	0x5A
DATA	n/a
Description	Refer to chapter 2.4 “Real-Time Status and Alarm Notification" for details.
Example Message	ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7 05 5A 01 01 00 00 00 00 00 00 00 Data d0 = 1 (Falling edge is detected on IN1).

Appendix-1 RTU CRC16 Source Code

UIMessage uses CRC-16 (Modbus) algorithm. For details, please refer to Online CRC-8 CRC-16 CRC-32 Calculator (crccalc.com) . The source code used in the calculation is listed below.

```
// CRC low byte table
unsigned char tblCRCLo[256] = {
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7, 0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C,
0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09, 0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9,
0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C , 0xDC, 0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17,
0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3, 0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF 2, 0x32, 0x36,
0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4, 0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A, 0x3B,
0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x2 8, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF, 0x2D,
0xED, 0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26, 0x22, 0xE 2, 0xE3, 0x23, 0xE1, 0x21, 0x20,
0xE0, 0xA0, 0x60, 0x61, 0xA1, 0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4, 0x6C,
0xAC , 0xAD, 0x6D, 0xAF, 0x6F, 0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68, 0x78, 0xB8, 0xB9,
0x79, 0xBB, 0x7B, 0x7A, 0xBA, 0xBE, 0x7E, 0x7F, 0xBF , 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5, 0x77,
0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0, 0x50, 0x90, 0x91, 0x5 1, 0x93, 0x53, 0x52, 0x92,
0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C, 0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B,
0x5B, 0x9 9, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D,
0x4D, 0x4C, 0x8C, 0x44, 0x84, 0x85, 0x45, 0x87, 0x4 7, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83, 0x41, 0x81, 0x80, 0x40 };

// CRC high byte table
unsigned char tblCRCHi[256] = {
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80,
0x41, 0x0 0, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC 0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x8 1, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x4 0, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x0 0, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00,
0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC 1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x8 0,
0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x4 0, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x0 1, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC 0, 0x80,
0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40 };

// Calculate CRC, * buf points to 0xAA ID CW … d8 data sequence, q ty = 13 bytes
unsigned short RtuCrc16(unsigned char* buf, unsigned int qty)
{
	unsigned char crcH = 0xFF;
	unsigned char crcL = 0xFF;
	int idx = 0;

	while (qty--)
	{
		idx = crcL ^ *buf++;
		crcL = crcH ^ tblCRCHi[idx];
		crcH = tblCRCLo[idx];
	}

	return (crcH << 8) | crcL;
}
```