User Manual

UIM342AB / UIM342SAB / UIM342XSAB

Motion Controller for Servo Stepper Motor

with CAN Interface

Absolute Multi-turn Encoder

& Interpolated Motion Control

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in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates.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.

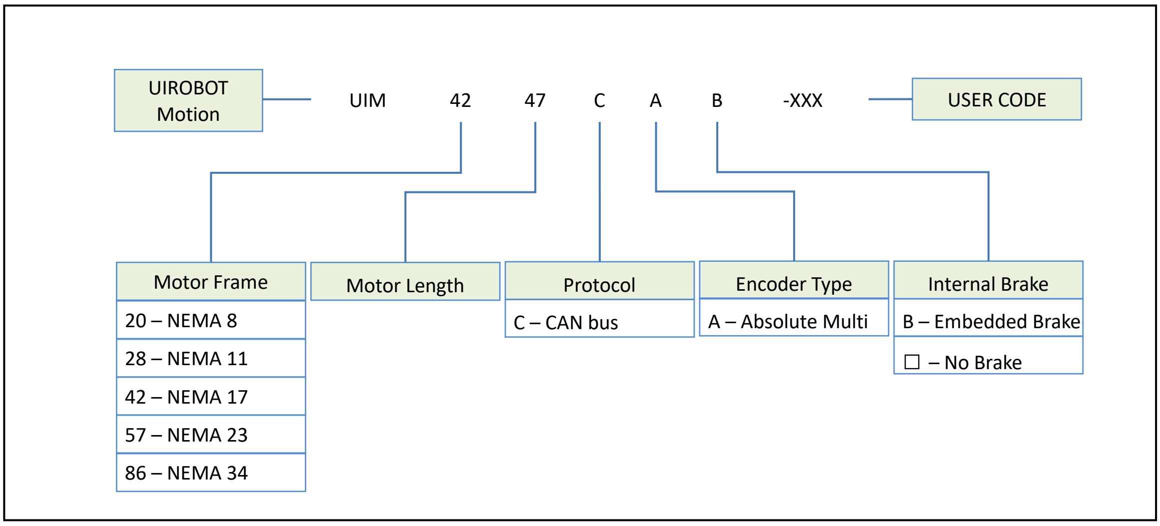
Trademark and Design Statement

- 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
V 2.0	July 1st, 2025	Interpolated Motion Control Added

UIM342AB / UIM342SAB / UIM342XSAB
Motion Controller for Servo Stepper Motor
With CAN Bus Interface
&
Absolute Multi-Turn Encoder

The UIM342AB series are advanced motion controllers designed for servo stepper motors, featuring a CAN Bus interface and Absolute Multi-Turn Encoder.

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:

- **Absolute Multi-Turn Encoder, with internal rechargeable battery.**
- **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

Advanced Motion

– Interpolated Motion for multi-axis coordination, Speed Control and Position Control
– High positioning accuracy, high response, Maximum speed up to 3000 rpm, it's depends on the Motor type
– Backlash Compensation
– Stall detection

Motor Driver

– Wide Voltage Input: Operates on 24 ~ 48 VDC (1)
– Adjustable Phase Current: Up to 8 A (1)
– Micro-Stepping Resolution: 1 ~ 1/128
– No vibration or noise at low speeds.
– Protection: Overcurrent, overvoltage, overheating, etc.

Note:
(1) Depending on the actual controller.

Absolute Encoder

– Resolution: Single 17-bit, Multi 32-bit (+/-2^{31} Turns)
– Embedded Rechargeable Battery, standby 1+ year, life time 50+ years.
– 1.5uA ultra-low standby power consumption.

I/O Logic Control

– 3 Digital Inputs, 1 Output (5V / 50mA)
– Input change notification
– 13 programmable actions can be attached to the I/O trigger

CAN Networking

– Active CAN 2.0, 1 Mbps Max.
– 3 types IDs (Node ID, Group ID, Global ID) for synchronized motion.

Other features

– Emergency Lockdown
– 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

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

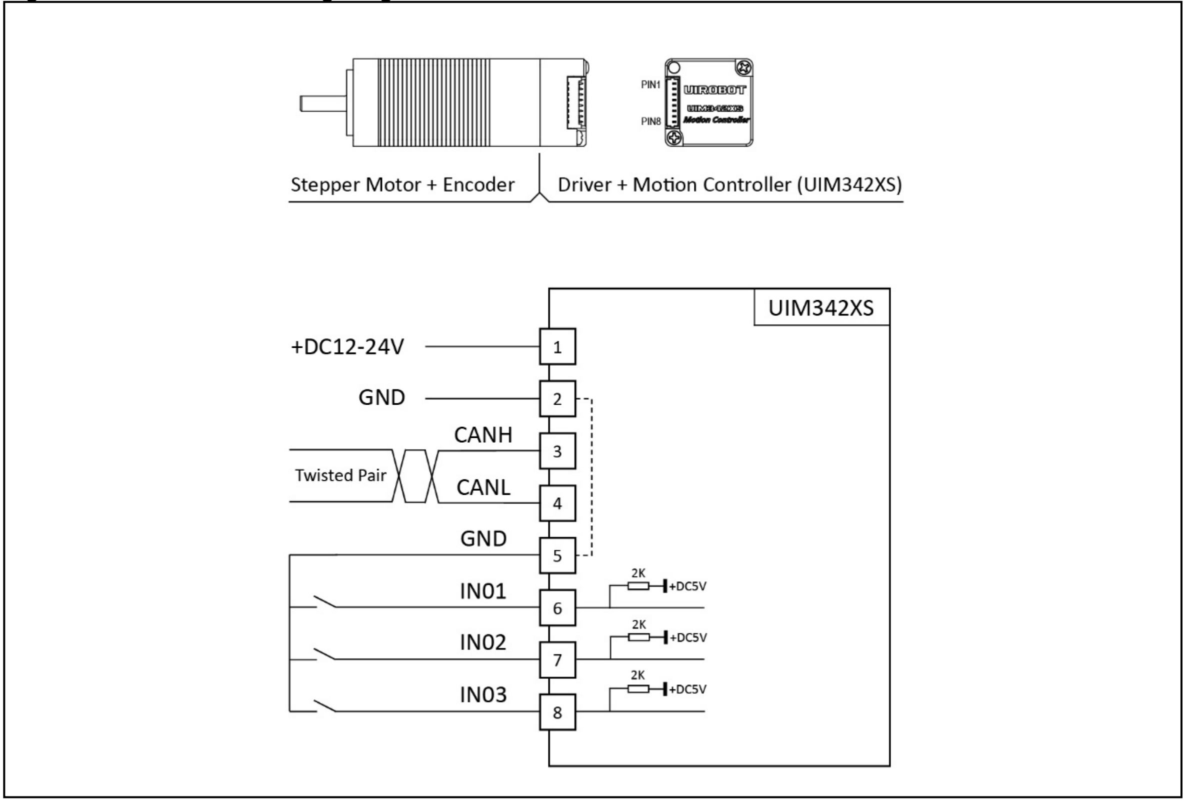


Figure 0-2: UIM342S Wiring Diagram

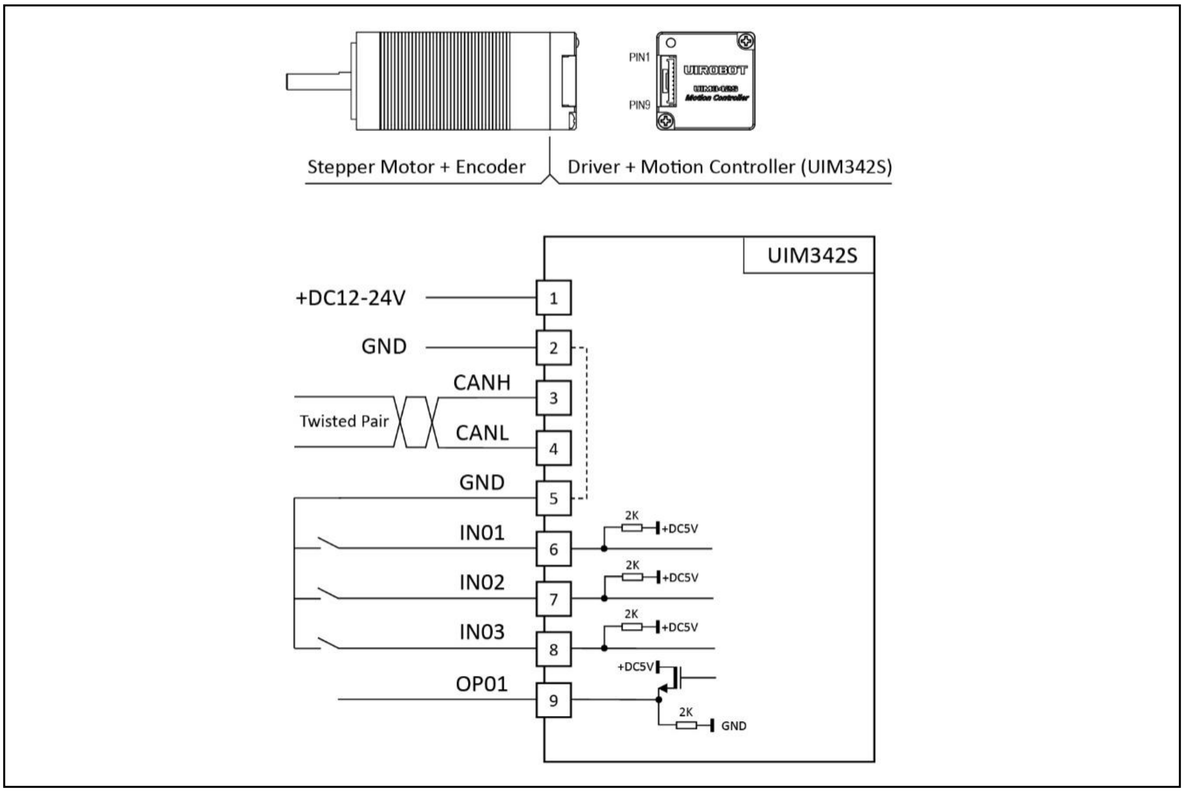


Figure 0-3: UIM342C Wiring Diagram (Socket Style)

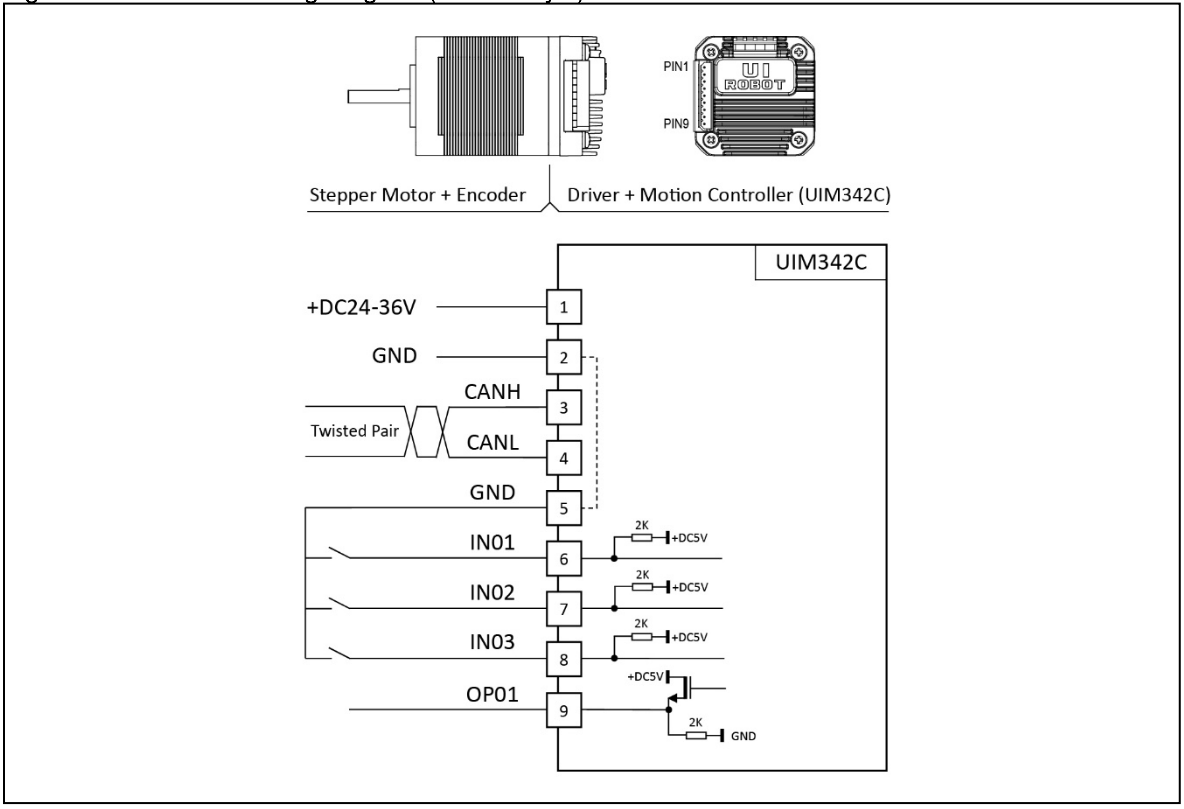
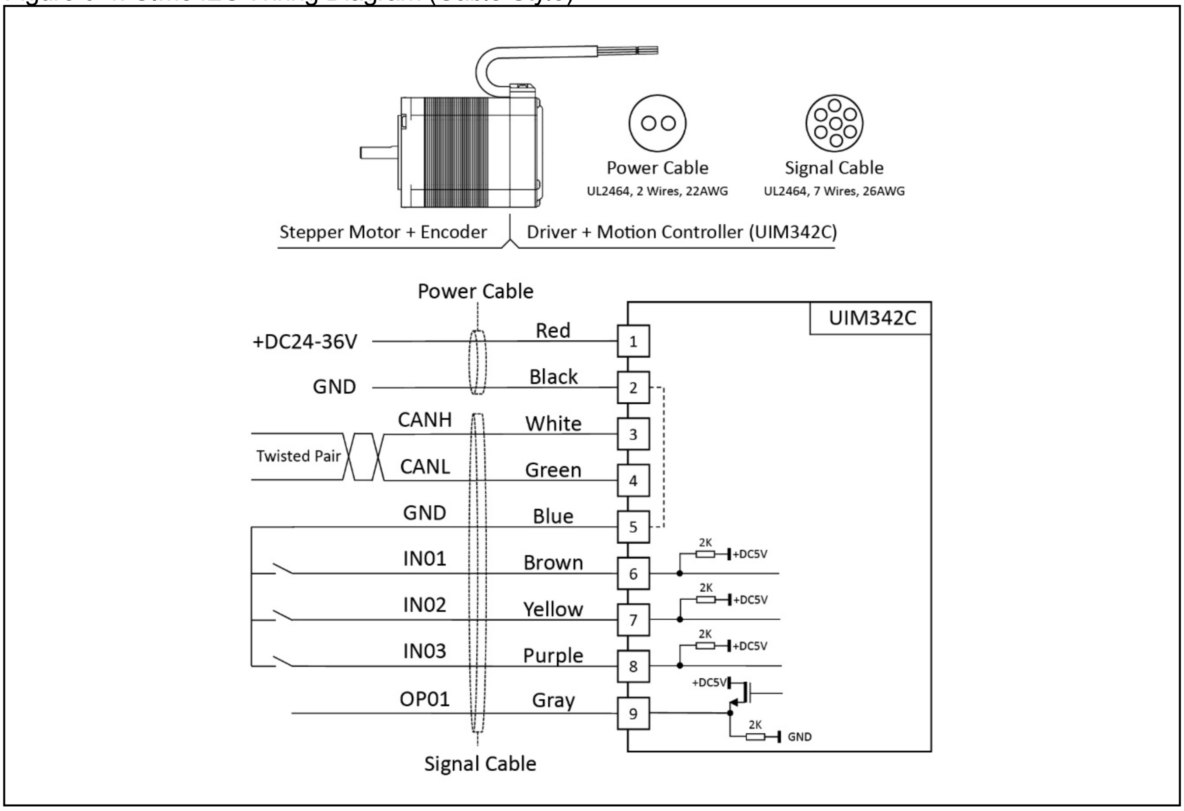


Figure 0-4: UIM342C Wiring Diagram (Cable Style)



INSTRUCTION SUMMARY

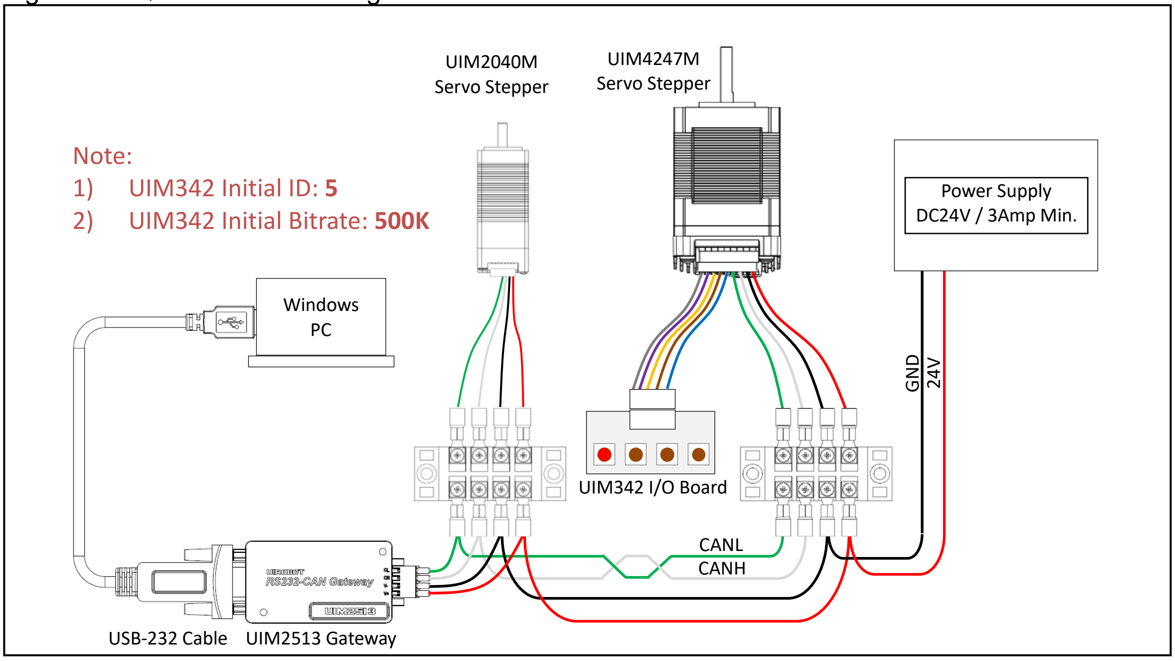
Classification	Mnemonic	Control Word	Function	Object	Chapter
Protocol	PP[i]	0x01	Set / Get	Protocol Parameters	6.1
System	IC[i]	0x06	Set / Get	Initial Configuration	6.2
	IE[i]	0x07	Set / Get	Information Enable	6.3
	ML	0x0B	Get	Model	6.4
	SN	0x0C	Get	Serial Number	6.5
	ER[i]	0x0F	Clear / Get	Error Report	6.6
	QE[i]	0x3D	Set / Get	Quadrature Encoder	6.7
	SY[i]	0x7E	Set	System Operation	6.8
Motor Driver	MT[i]	0x10	Set / Get	Motor Driver parameters	6.9
	MO	0x15	Set / Get	Motor On /Off	6.10
Motion Control	BG	0x16	Set	Begin Motion	6.11
	ST	0x17	Set	Stop Motion	6.12
	MF	0x18	Get	Motion Parameter Frame	6.13
	AC	0x19	Set / Get	Acceleration	6.14
	DC	0x1A	Set / Get	Deceleration	6.15
	SS	0x1B	Set / Get	Cut-in speed	6.16
	SD	0x1C	Set / Get	Stop Deceleration	6.17
	JV	0x1D	Set / Get	Jog Velocity	6.18
	SP	0x1E	Set / Get	PTP Speed	6.19
	PR	0x1F	Set / Get	Position Relative	6.20
	PA	0x20	Set / Get	Position Absolute	6.21
	OG	0x21	Set	Set Origin	6.22
	BL	0x2D	Set / Get	Backlash Compensation	6.23
	MS[i]	0x11	Clear / Get	Motion Status	6.24
	DV[i]	0x2E	Get	Desired Values	6.25
I / O	IL[i]	0x34	Set / Get	Input Logic	6.26
	TG[i]	0x35	Set / Get	Trigger	6.27
	DI	0x37	Set / Get	Digital I /O	6.28
Notification	RT	0x5A	Auto Sent	Real-Time Inform	6.29
Interpolated Motion	MP[i]	0x22	Set / Get	PVT/PT Motion Parameters	6.30
	PV	0x24	Set / Get	PVT/PT Motion Begin Index	6.31
	QP[N]	0x25	Set / Get	Position Value of PVT Queue	6.32
	QV[N]	0x26	Set / Get	Velocity Value of PVT Queue	6.33
	QT[N]	0x27	Set / Get	Time Interval of PVT Queue	6.34
	QF	0x29	Set	Quick Feed PVT Queue	6.35
	PT[N]	0x23	Set / Get	Position Value of PT Queue	6.36

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.

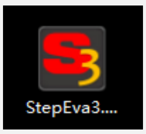
– UIM2513 has a built-in terminal resistor, which can be activated by the toggle switch. It is recommended to connect a 120-ohm terminal resistor to the other end of the CAN cable.

– 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.

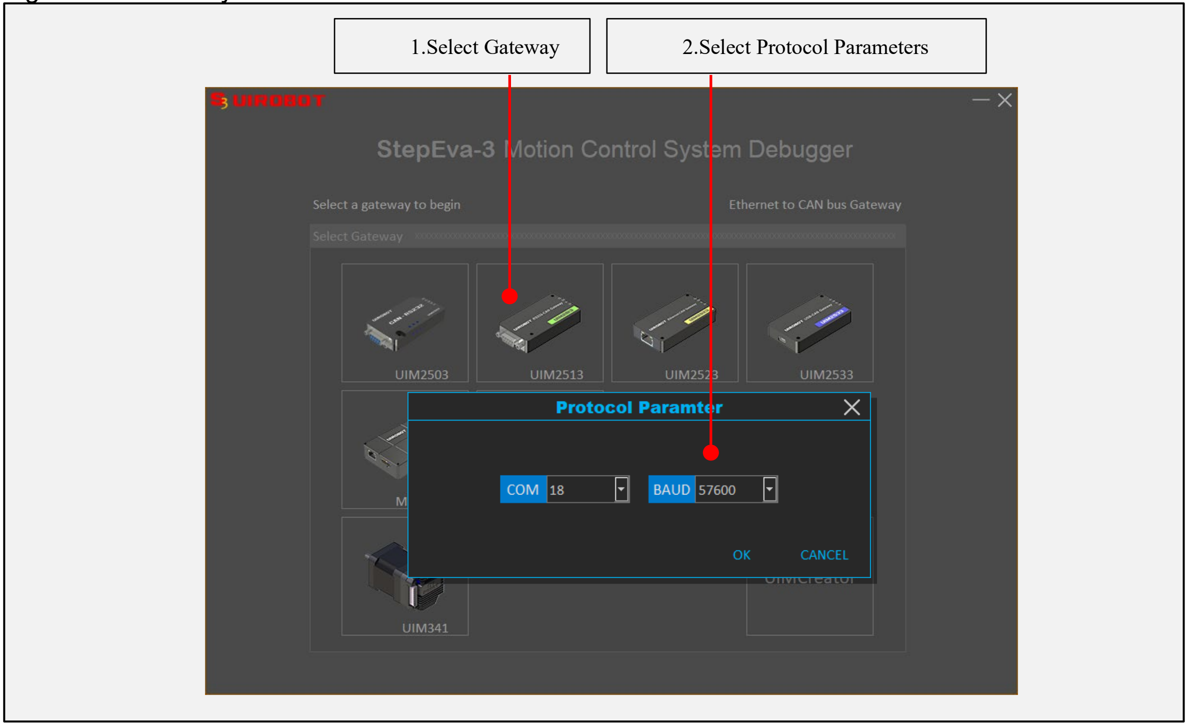
2) Download and click to run the Windows based control panel “StepEva3”.

Figure 0-6: Icon of StepEva3



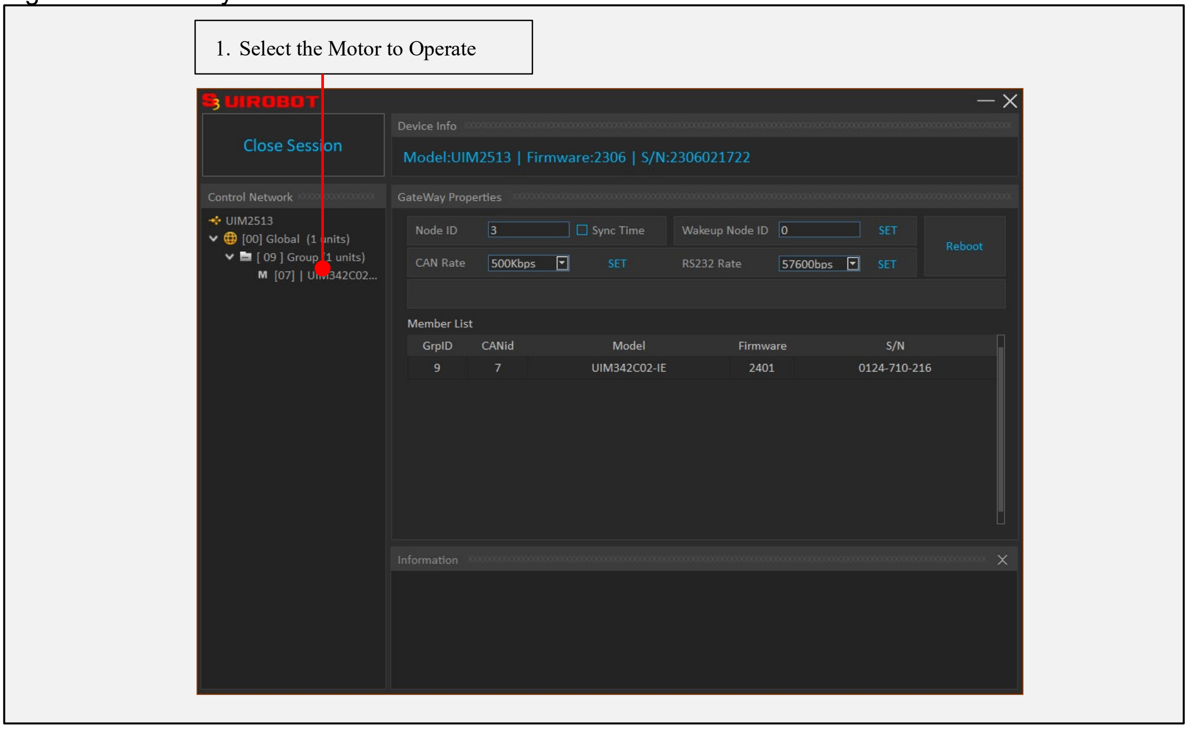
3) UIM2513 (RS232 -CAN) should be selected here.

Figure 0-7: Gateway selection screen 2



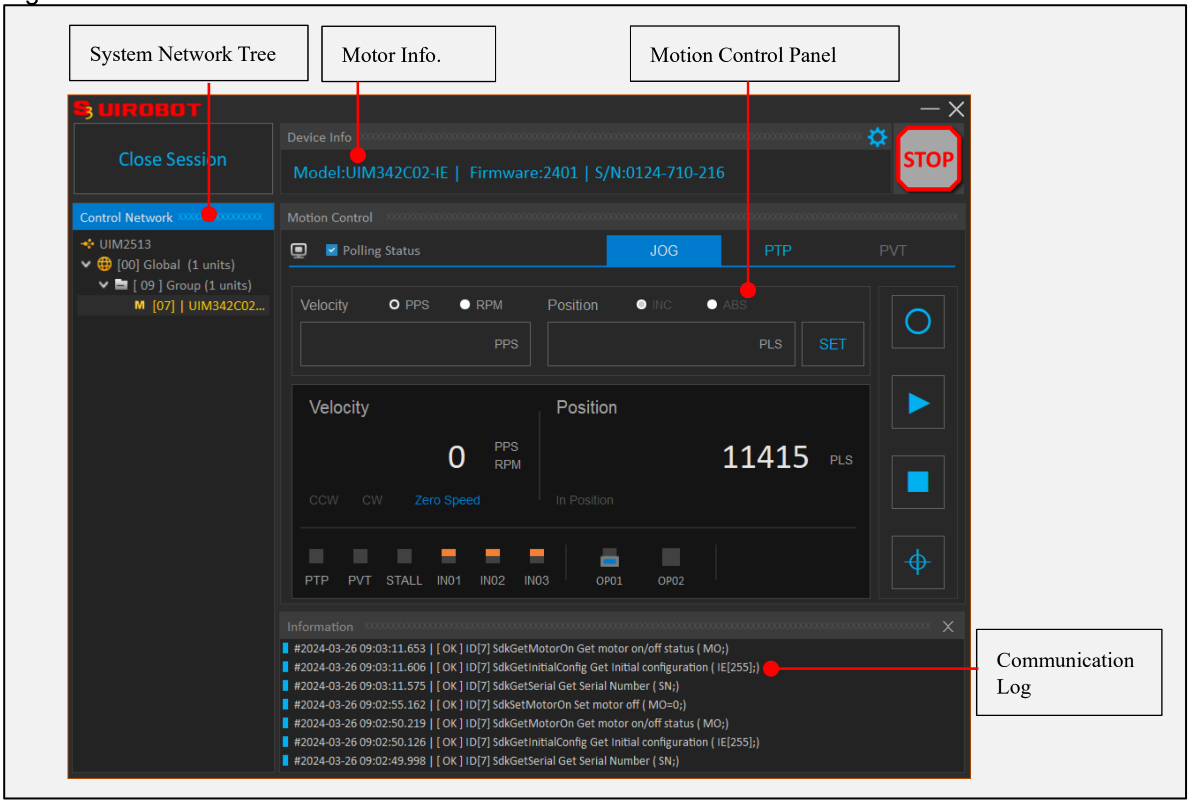
4) Select the UIM342C02-AB to operate in the device list on the left.

Figure 0-8: Gateway information screen



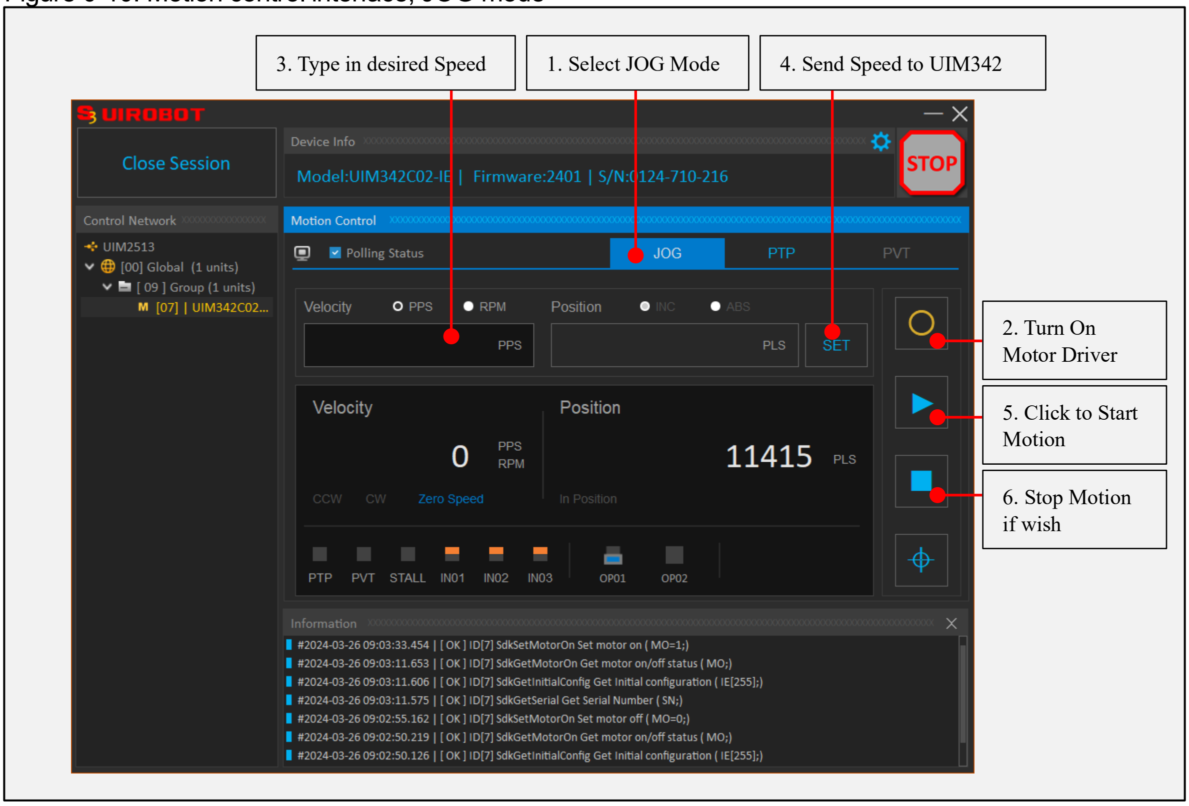
5) Set motion parameters and operate the motor to move.

Figure 0-9: Motion control interface



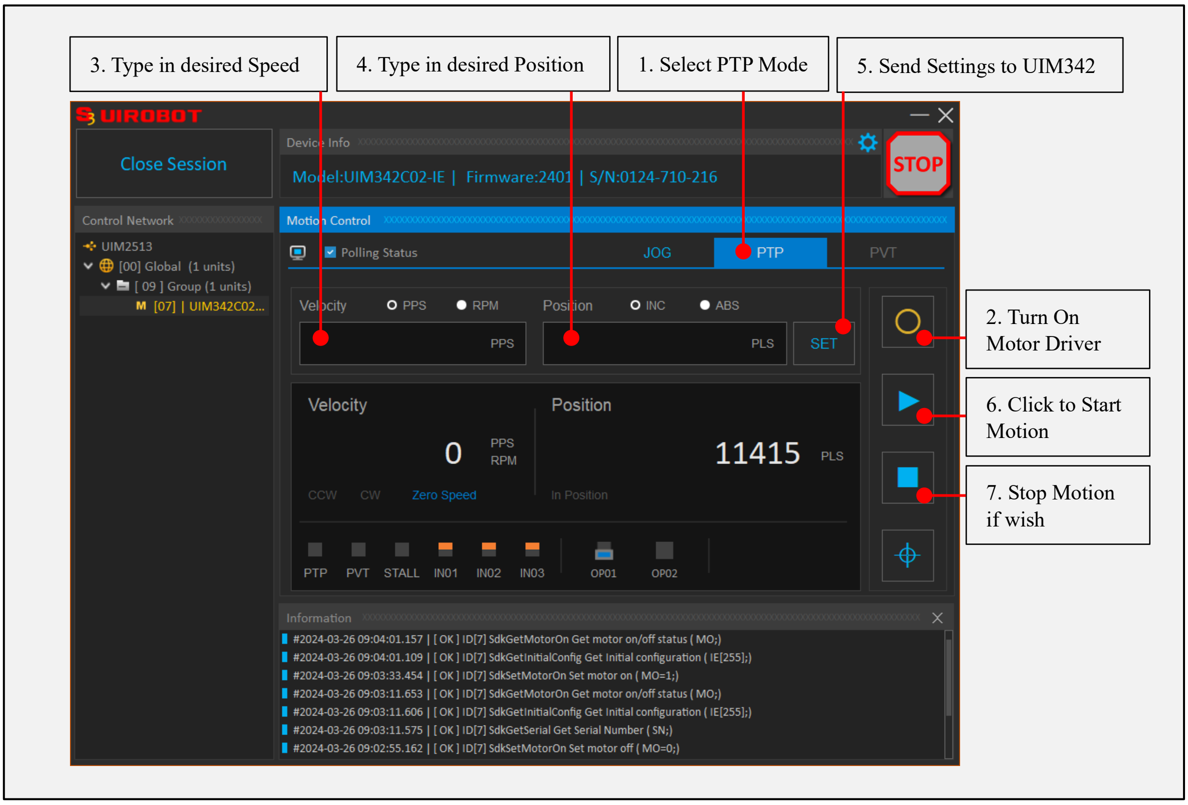
For JOG Mode, follow steps 1 ...6 below to control the motor rotation.

Figure 0-10: Motion control interface, JOG mode



For 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- (UIM342XSAB / 342SAB)…..10 V to 30 V
Voltage on V+ with respect to V- (UIM342AB)...12 V to 50 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	UIM342
Supply Voltage (DC)	12V– 28V	12V– 28V	16V– 48V
Driver Capacity	Peak 1 / 2 / 4 / 8 Amp		
Current Control	PWM constant current		
Micro-Stepping	1/2/4/8/16/32/64/128		

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 ℃

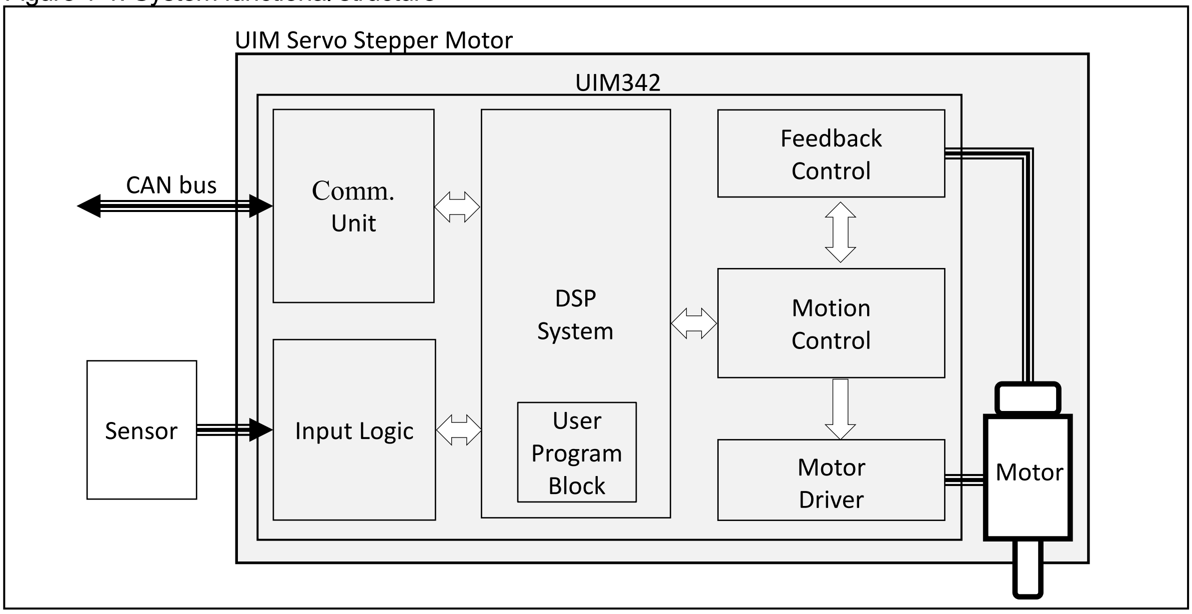
Communication Interface

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.

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

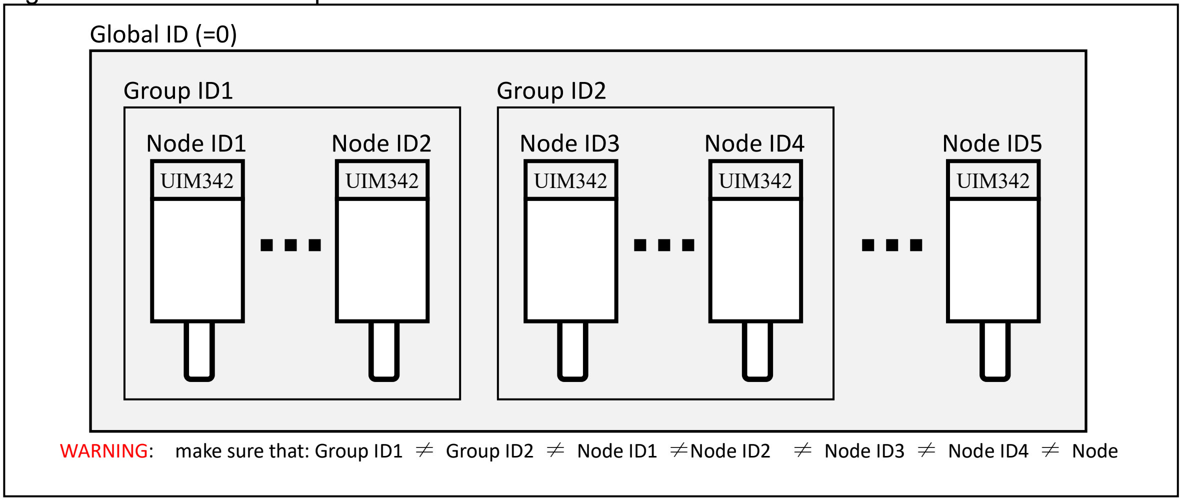


1.1 Communication

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 1/2/4/8/16/32/64/128 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 13 functional behaviors that can be attached to the rising/falling edge of the inputs:

1) Turn Motor Driver Off, Free wheel.
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) .
12) Turn Motor Driver Off (Free wheel) + Lock Down the control system (i.e inquiry only).
13) Emergency stop (using SD) + Lock Down the control system (i.e inquiry only).

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.

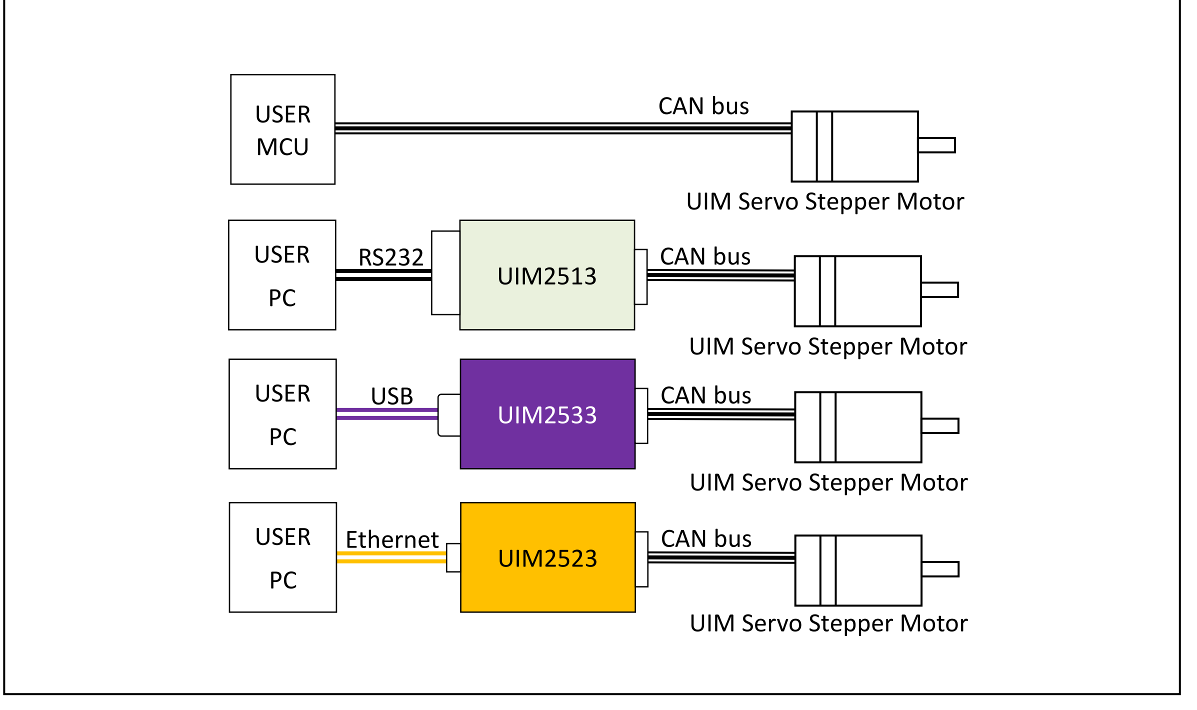
Meanwhile, UIROBOT provides R&D services for user control system solution development.

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

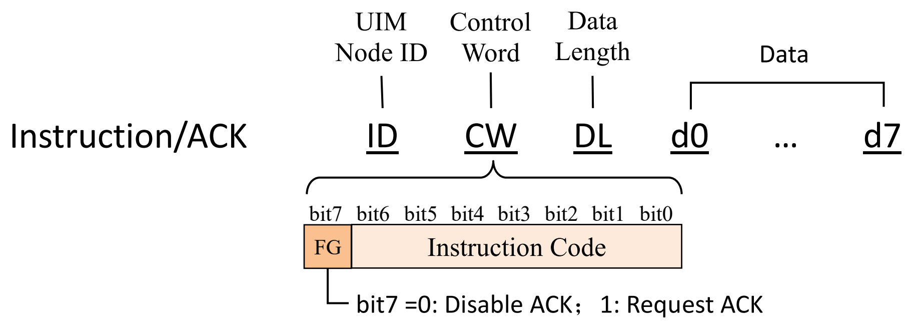
This section introduces the instructions, reply / ACK, error reporting, and real-time status and alarm notifications supported by UIM342.

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

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

d0…d7 Data bytes, (least significant byte first).

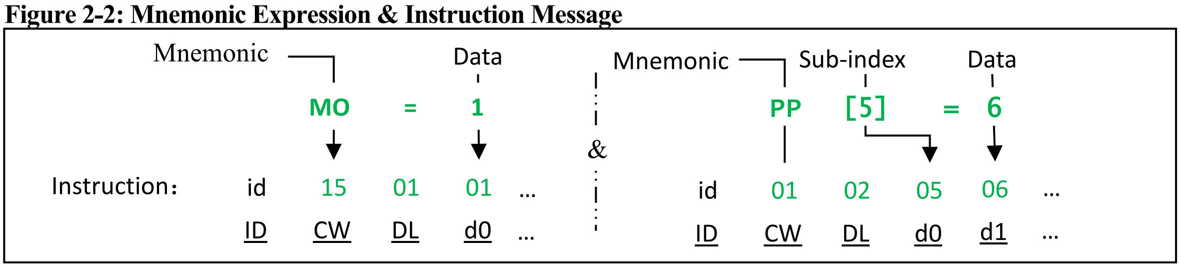
2.2 Mnemonic

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 the motor driver

The relation between mnemonic expressions and instructions is shown below:

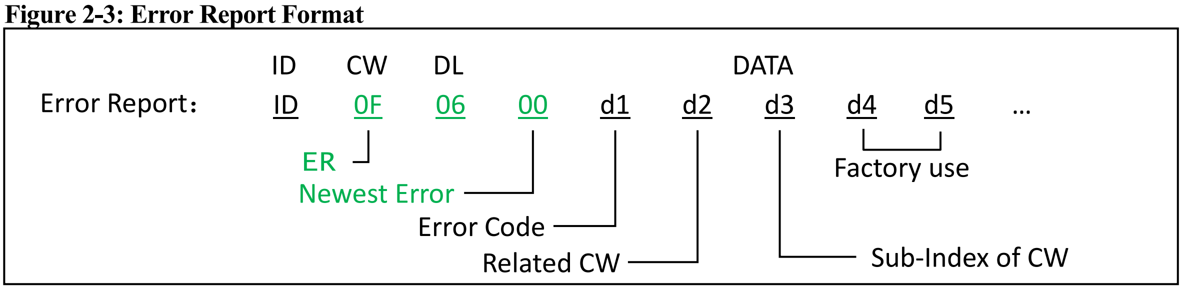
Figure 2-2: Mnemonic Expression & Instruction Message



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;

Extended CAN Message

SOF	Std. ID (SID) 11 Bit	SRR	IDE	Ext. ID (EID) 18 Bit	RTR	RB1	RB0	DLC 4 Bit	Data Field 0.. 8 Byte	CRC 15 Bit	DEL	ACK

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

<table>
<tr><td colspan="29">CAN-ID (29bit)</td></tr>
<tr><td colspan="11">SID[10：0] (11bit）</td><td colspan="18">EID [17:0] (18bit)</td></tr>
<tr><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="5">Producer ID [4:0]</td><td colspan="5">Consumer ID [4:0]</td><td>0</td><td colspan="2">Producer ID [6:5]</td><td colspan="2">Consumer ID [6:5]</td><td colspan="6">Reserved = 0</td><td colspan="8">Control Word (CW)</td></tr>
</table>

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
```

<u>Example</u>

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 UI Message format.

Figure 2-5: UI Message format (requires CRC)

	SM	Instruction/ACK						Aux. Byte	CRC		EM
UI Message:	AA	ID	CW	DL	d0	…	d7	d8	R0	R1	CC

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 2-6: UI Message format (no CRC)

	SM	Instruction/ACK						Aux Byte	CRC		EM
UI Message:	AD	ID	CW	DL	d0	…	d7	d8	00	00	CC

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

3.0 Motion Control

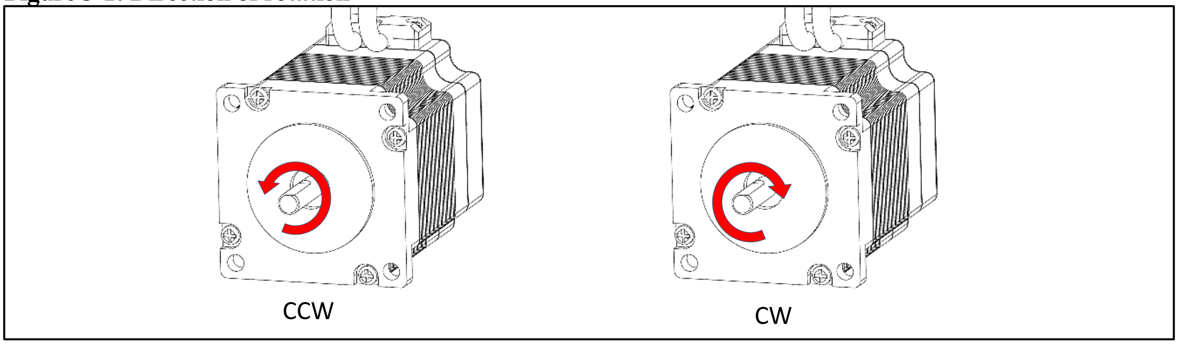
UIM342 motion controller supports speed control (JOG) mode and point to point (PTP) position control mode. User can Set, Get and Store motion parameters through instructions. Meanwhile, user can also receive the working status of the motor through real-time notification.

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)**

The **UIM342** controls motor speed and displacement, stopping when the desired position is reached. It adjusts the speed to its maximum while ensuring precise position accuracy.

– The **SP** instruction sets the target speed for position control.
– The **PA** instruction defines the absolute position.
– The **PR** instruction defines the relative position from the current location.

After setting the PA/PR instruction, use the **BG** instruction to initiate movement. Multiple UIM342 controllers can be programmed with PA/PR and started simultaneously by sending the BG command to a specific group ID or the global ID.

Note: The motor's actual speed and direction are determined by the position deviation (desired position - actual position). If the position deviation is minimal, the motor may reach the target position before achieving the desired speed.

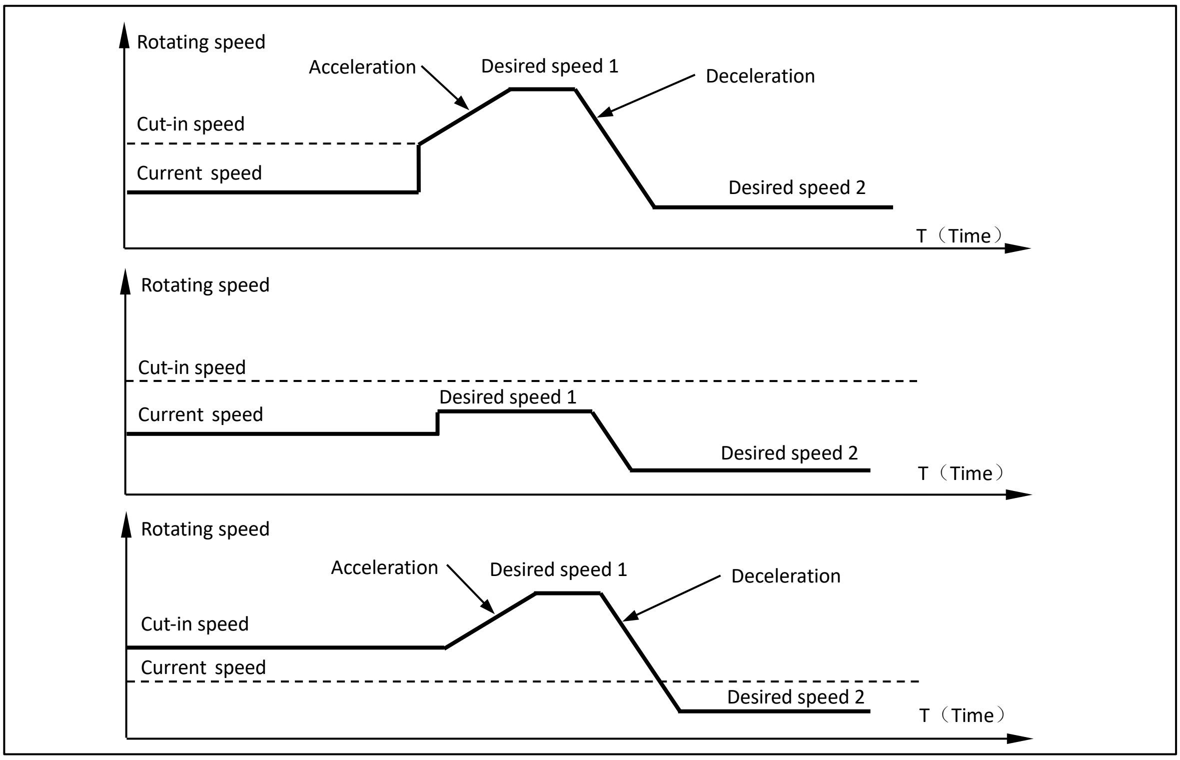
- **Interpolated Motion (PVT/PT)**

The UIM342AB controller has an Absolute Multi-turn Encoder and supports the curve interpolation motion (PVT/PT Motion). Through the PVT/PT (Position-Velocity-Time) control, the user only needs to set key control points (position, speed, and time interval), the controller will automatically generate a continuous and smooth motion trajectory between these points, ensuring that the position, speed, and acceleration are continuous throughout the motion process, avoiding sudden changes and shocks.

3.3 Acceleration, Deceleration and Cut-in speed

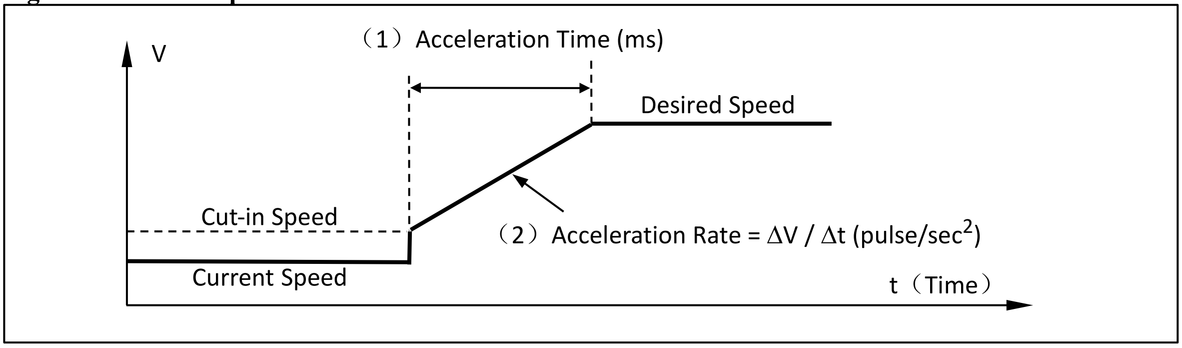
In order to improve the response of the motor and avoid the resonance, the acceleration (deceleration) process includes a cut-in speed.

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 like screw-nut or rack-and-pinion transmissions often experience backlash, which causes a delay when reversing direction. This leads to accumulated error that must be compensated.

The UIM342 addresses this issue by providing backlash compensation. The compensation value can be adjusted using the BL instruction, with a range from 0 to 65535 pulses. This helps to minimize errors and improve system accuracy.

3.5 Closed Loop / Open Loop Control

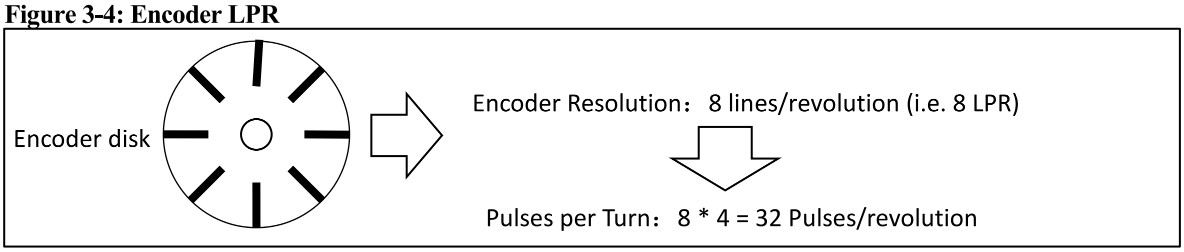
The **UIM342** motor controller supports both open-loop and closed-loop control:

- **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 a quadrature encoder provides feedback for precise displacement control.

In closed-loop mode, the quadrature encoder ensures accurate motor positioning by delivering displacement feedback.

LPR (Lines per Revolution) refers to the number of lines engraved on the encoder disk. For each full motor revolution, the encoder generates 4 times the LPR value in pulses.

Figure 3-4: Encoder LPR



CPR (Counts Per Revolution) defines the number of pulses required for the motor to complete one full revolution. This value is critical for closed-loop control in the UIM342, as it directly affects positioning accuracy. The CPR is calculated as:

`CPR = Micro-stepping Resolution × 200`

(200 is the number of full steps per revolution for a standard stepper motor.)

For example, if the micro-stepping resolution is 16:

```
CPR = 16 × 200 = 3200
```

This means 3200 pulses are needed for one full rotation.

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.

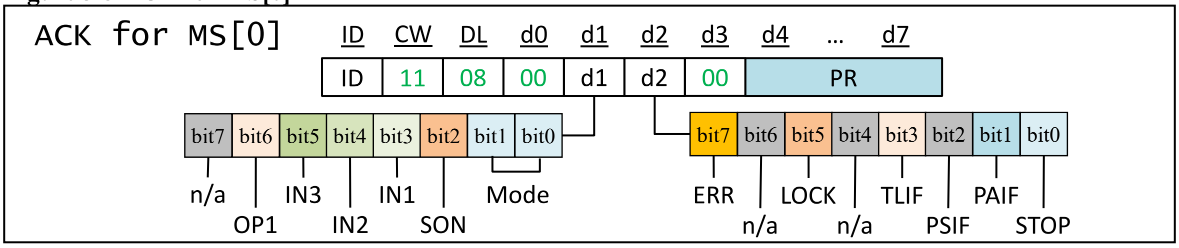
3.7 Acquire Motion Status

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 3-5 ACK for MS[0]



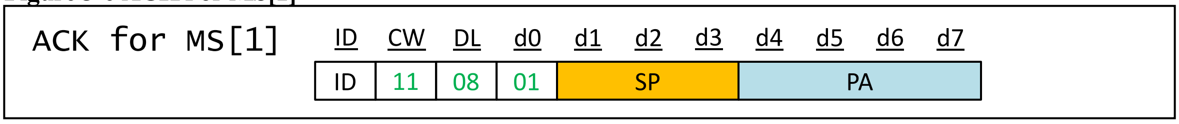
	name	description	value
d1.bit<1:0>	Mode	motion mode	0 = JOG; 1 = PTP
d1.bit2	SON	motor driver	0 = OFF; 1 = ON
d1.bit3	IN1	IN1 Logic Level	0 = Low; 1 = High
d1.bit4	IN2	IN2 Logic Level	0 = Low; 1 = High
d1.bit5	IN3	IN3 Logic Level	0 = Low; 1 = High
d1.bit6	OP1	OP1 Logic Level	0 = Low; 1 = High
d1.bit7	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	PSIF	PVT Stopped	0 = NO; 1 = YES
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 3-6 ACK f or MS[1]



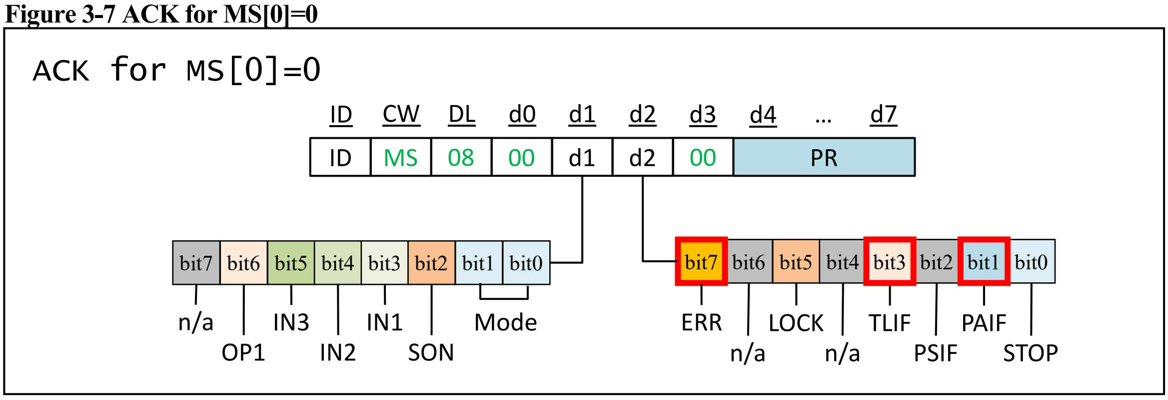
	name	description	value
d3:d2:d1	SP	Current speed	Signed 24 bit integer, LSB received first
d7:d6:d5:d4	PA	Current absolute position	Signed 32 bit integer, LSB received first

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 3-7 ACK for MS[0]=0



4.0 Interpolated Motion

Curve interpolation motion control is an advanced motion control technology. The motion controller automatically generates a continuous and smooth motion trajectory (including straight segments and various curves) based on several control points preset by the user. This control method not only improves motion precision but also effectively reduces mechanical shocks and vibrations, achieving higher control precision and system stability. This technology is widely used in CNC machines, industrial robots, and automation production lines for precise motor trajectory control.

The UIM342AB controller has an Absolute Multi-turn Encoder and supports the curve interpolation motion (PVT/PT Motion). The integrated motor series with UIM342AB includes UIM2040CA, UIM2851CA, UIM4247CA/CAB, UIM5756CA/CAB, UIM8696CA/CAB, etc.

Through the PVT/PT (Position-Velocity-Time) control, the user only needs to set key control points (position, speed, and time interval), the controller will automatically generate a continuous and smooth motion trajectory between these points, ensuring that the position, speed, and acceleration are continuous throughout the motion process, avoiding sudden changes and shocks.

Based on PVT/PT motion, multiple UIM motors can achieve precise coordinated motion to complete complex spatial trajectory control tasks. The system not only ensures synchronization between axes but also guarantees the smoothness and accuracy of the motion trajectory.

PVT/PT motion includes two type of motions: the PVT motion and the PT motion. This chapter introduces the functions and applications of the UIM342AB in interpolation motion control.

Definition of Terms used in this chapter:

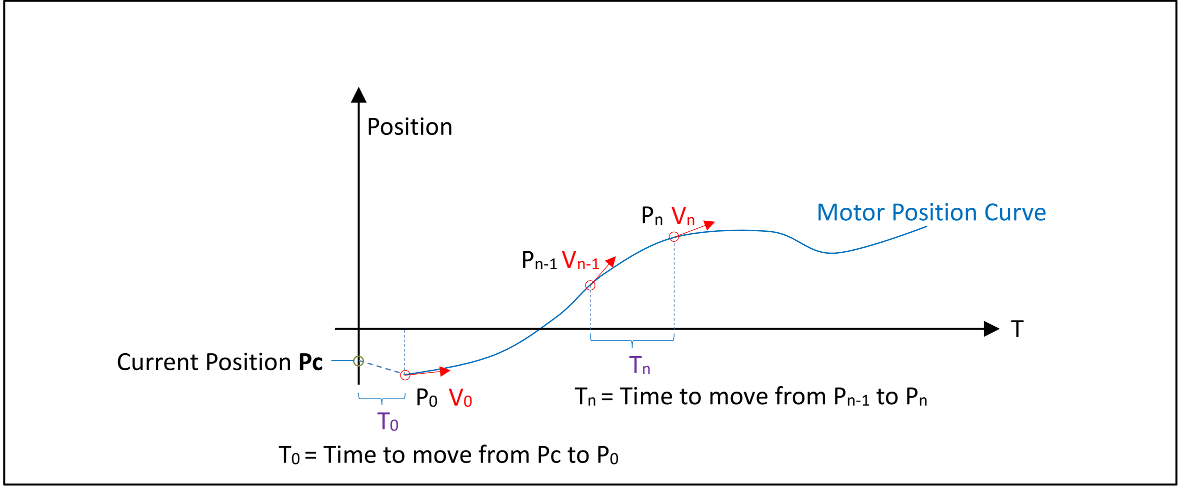
- **PVT**: Position – Velocity – Time.
- **PT**: Position – Time.
- **PVT[n]**: A data array comprised of Position, Velocity and Time, with index **n** (**range 0…255**).
- **PT[n]**: A data array comprised of Position and Time, with index **n** (**range 0…511**).
- **PVT Table**: A data structure in the UIM342AB that stores sets of PVT[n]/PT[n].
- **Queue**: An array of valid PVT[n]/PT[n] to be executed.
- **Queue Level**: The number of pending PVT[n]/PT[n] in a PVT table waiting to be executed.
- **Queue Low**: A state where the Queue Level is low.
- **Queue Empty**: A state where the Queue Level is zero.

4.1 PVT Motion

In PVT (Position-Velocity-Time) motion, the user defines a set of motion points, each with a position, speed, and time. These points are interpolated using **cubic spline** to ensure smooth transitions.

Each PVT point specifies **Position, velocity** (both absolute) and **Time** (relative)

Figure 4.1 Definition of PVT point N (P[n] / V[n] / T[n])



To initiate PVT motion at a precise moment, a BG command can be sent to a group of motors (using Group-ID) or to all motors (using Global-ID), making PVT very suitable for coordinated control of multi-axis systems.

To ensure synchronization, the UIM gateway's synchronization function aligns the absolute time counters between all drives, with microsecond-level precision (for detailed information, please refer to the UIM gateway manual).

Instructions used to manage the PVT motion are listed below:

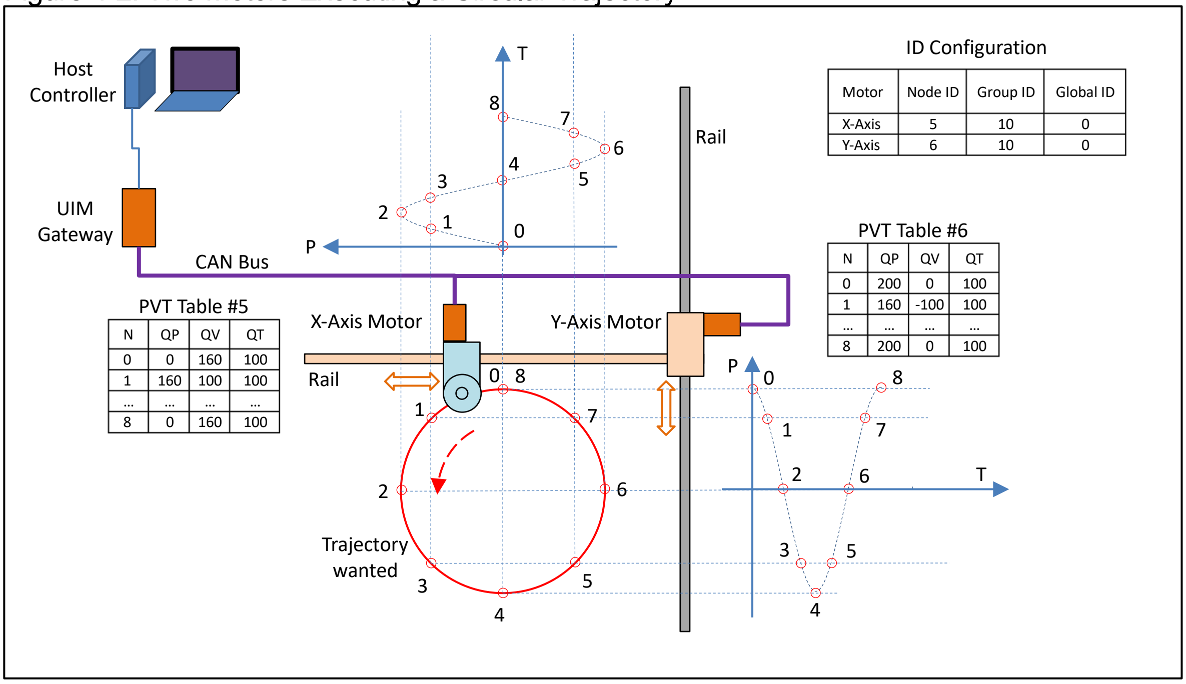
– `MP[i]` : Used to set PVT/PT Motion Configuration.
– `PV` : Select the PVT/PT Motion and the index of the starting row n
– `QP[n], QV[n], QT[n], QF` : PVT[n] data feeding.

Example: Two Motors Executing a Circular Trajectory

The electro-mechanical system structure (as shown below) includes:

1) Two linear actuators mounted along the X and Y axes.
2) Two motors connected to the Host Controller through the UIM gateway (such as UIM2513, UIM2523), using a CAN bus in series.

Figure 4-2: Two Motors Executing a Circular Trajectory



ID Configuration

Motor	Node ID	Group ID	Global ID
X-Axis	5	10	0
Y-Axis	6	10	0

PVT Table #5

N	QP	QV	QT
0	0	160	100
1	160	100	100
...	...	...	...
8	0	160	100

PVT Table #6

N	QP	QV	QT
0	200	0	100
1	160	-100	100
...	...	...	...
8	200	0	100

Steps to Achieve Circular Motion Using PVT Control (only for illustration):

1) **Motion Analysis and Mathematical Modeling**
   - The X-axis motor follows a sine wave shape
   - The Y-axis motor follows a cosine wave shape
2) **Assign Motor Node IDs and Group IDs (using PP[i]):**
   - X-axis motor: `PP[7]=5; PP[8]=10;  // Node ID=5, Group ID=10`
   - Y-axis motor: `PP[7]=5; PP[8]=10;  // Node ID=6, Group ID=10`
3) **Set PVT motion configurations for each motor:**
   - Motor #5: `MP[1]=0; MP[2]=8; MP[3]=3; MP[4]=0; MP[5]=0;`
   - Motor #6: `MP[1]=0; MP[2]=8; MP[3]=3; MP[4]=0; MP[5]=0;`
4) **Load the PVT table for each motor:**
   - Motor #5: `QP[0]=xx; QV[0]=xx; QT[0]=xx; …QP[8]=xx; QV[8]=xx; QT[8]=xx;`
   - Motor #6: `QP[0]=xx; QV[0]=xx; QT[0]=xx; …QP[8]=xx; QV[8]=xx; QT[8]=xx;`
5) **Enable motor drivers (if not already enabled).**
6) **Set the start point for each motor's PVT sequence:**
   - Motor #5: `PV=0;          // starting from the first row`
   - Motor #6: `PV=0;          // starting from the first row`
7) **Start the PVT motion for the motor group:**
   - Motor Group #10: `BG;      // start motion`

4.2 PVT Motion Instructions

The table below lists the control and parameter-setting commands used in PVT motion:

Parameter	Remark
SD (Stop Deceleration)	Rate of deceleration when motion is killed by queue empty or exception.
MP[i] (Motion Parameters)	MP[1] – First valid row in the PVT table. MP[2] – Last valid row in the PVT table. MP[3] – P/V/T data management mode. 0: FIFO Mode 1: Single Mode 3: Loop Mode MP[4] – Time for PT Motion. 0: PVT motion 5…65535: Time interval for PT motion MP[5] – "Queue low" alert value (quantity of remaining PVT row). MP[6] – Index of the next available writing row.
QP[N], QV[N], QT[N]	Set P/V/T data to UIM342
QF	Quick Feeding P/V/T to UIM342, only used in FIFO mode.
PV	Set PVT Motion and inform the index of the starting PVT row.

These instructions allow access to the PVT table and the configuration of the PVT motion.

To best understand the values of a given PVT[n], explanations are given as below:

- **Index = 0**

  After time interval QT[0], the motor moves from the Current Position to QP[0] with a speed of QV[0].

- **Index = 1**

  After time interval QT[1], the motor moves from QP[0] to QP[1] with a speed of QV[1].

- **Index = N**

  After time interval QT[N], the motor moves from QP[N-1] to QP[N] with a speed of QV[N].

4.3 PVT Motion Management

UIM342AB supports following 3 data management modes for using PVT data:

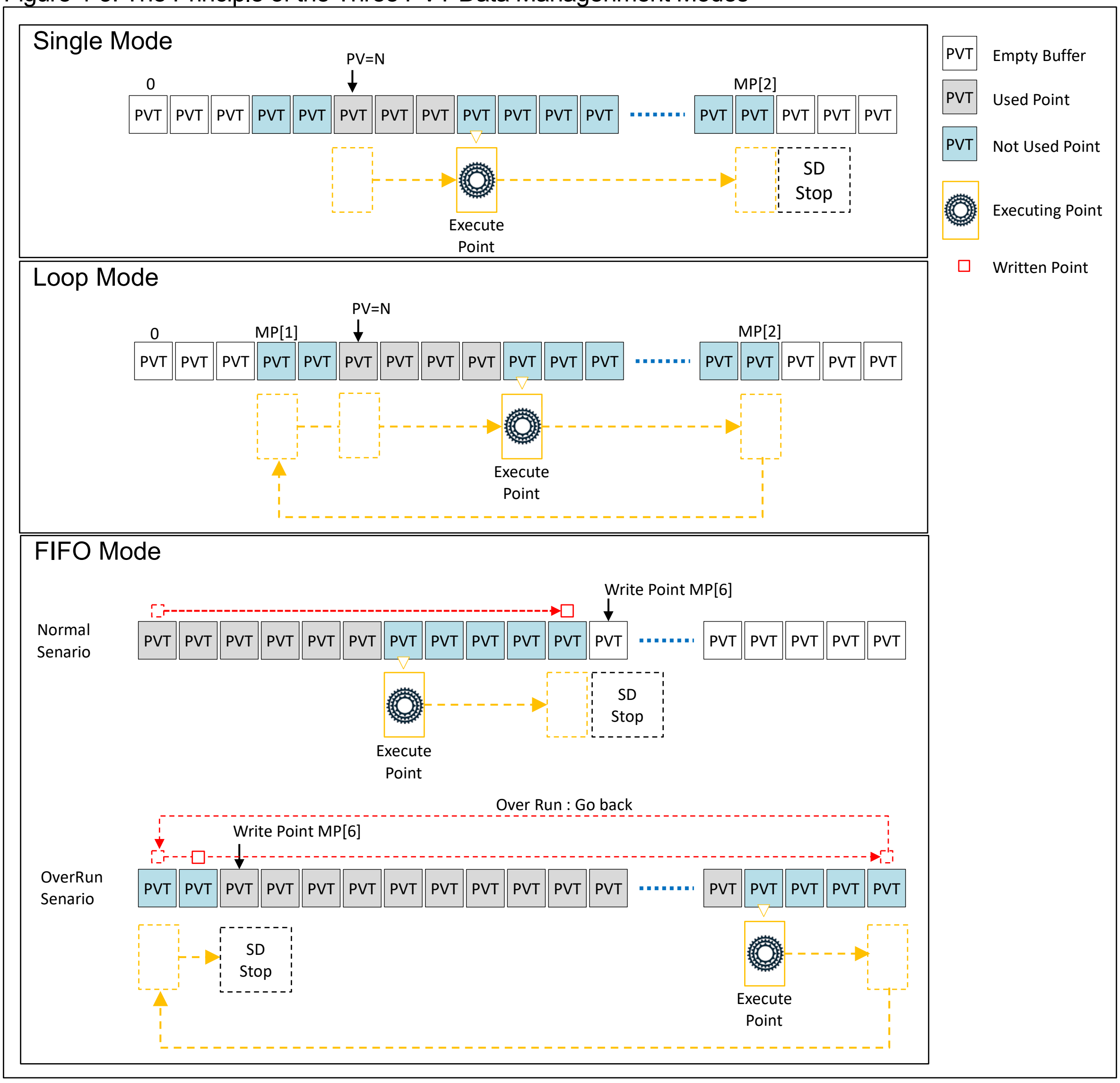
– **Single Mode**
  After the motion starts, UIM342 will move from PVT[n] (n is specified by PV) to PVT[y] (y is defined by MP[2]) and stop once it completes PVT[y].

– **Loop Mode**
  After the motion starts, UIM342 will move from PVT[n] (n is specified by PV) to PVT[y] (y is defined by MP[2]) and then return to PVT[x] (x is defined by MP[1]).

  This cycle will repeat endlessly.

– **FIFO Mode**
  The host loads a small number of PVT[n]. After the motion starts, UIM342 will begin from PVT[0]. The host will continuously supply additional PVT[n] during the motion.

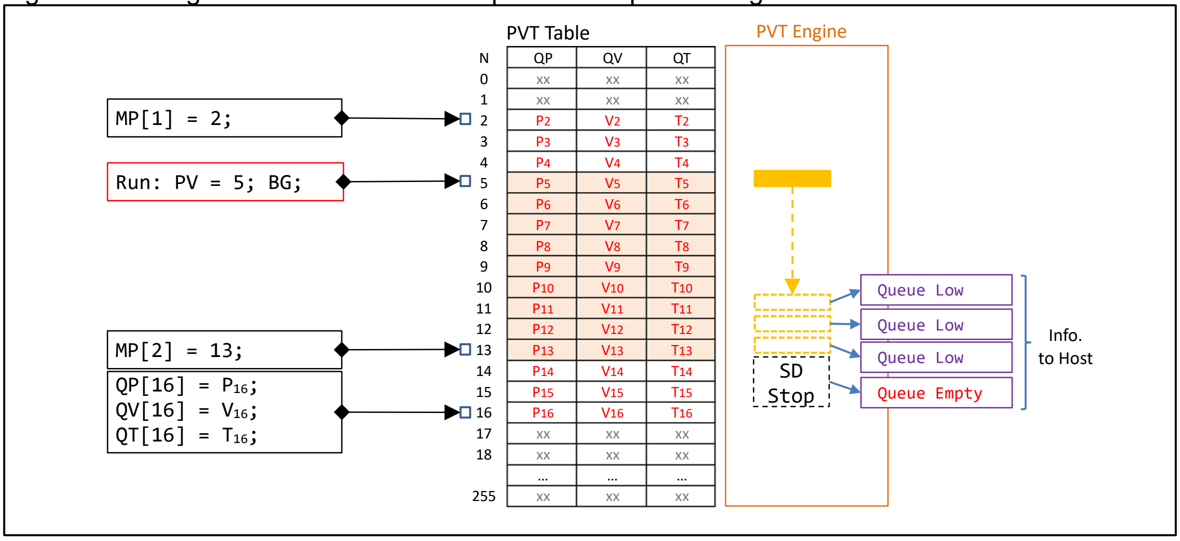
Figure 4-3: The Principle of the Three PVT Data Management Modes



4.3.1 Single Mode

When MP[3] = 1, PVT motion operates in **Single Mode**. The following diagram illustrates how to setup this mode and the UIM342 behaviors.

Figure 4-4: Single Mode Command and Operation Sequence Diagram



Setup Steps

1) Set PVT configuration:
   - `MP[4] = 0;    // Set PVT motion (instead of PT motion)`
   - `MP[3] = 1;    // Set Single Mode`
   - `MP[5] = 3;    // Set "Queue Low" alert for the last 3 PVT rows`
   - `MP[2] = 13;   // Set QP[13]/QV[13]/QT[13] as the last valid PVT rows`
   - `MP[1] = 2;    // Set QP[2]/QV[2]/QT[2] as the first valid PVT row`

2) Load the PVT lists:
   - `MP[0] = 0;    // Clear and reset the UIM342 PVT Table`
   - `QP[2] = P2; QV[2] = V2; QT[2] = T2;  // Load PVT data into the 2nd row`

     `...`
   - `QP[16] = P16; QV[16] = V16; QT[16] = T16;`

3) Enable the driver module (if not already enabled):
   - `MO = 1;       // Enable the motor driver`

4) Initialize the PVT mode and set the start point:
   - `PV = 5;       // Set PVT Motion, with start point of QP[5]/QV[5]/QT[5]`

5) Start the motion:
   - `BG;           // Begin Motion`

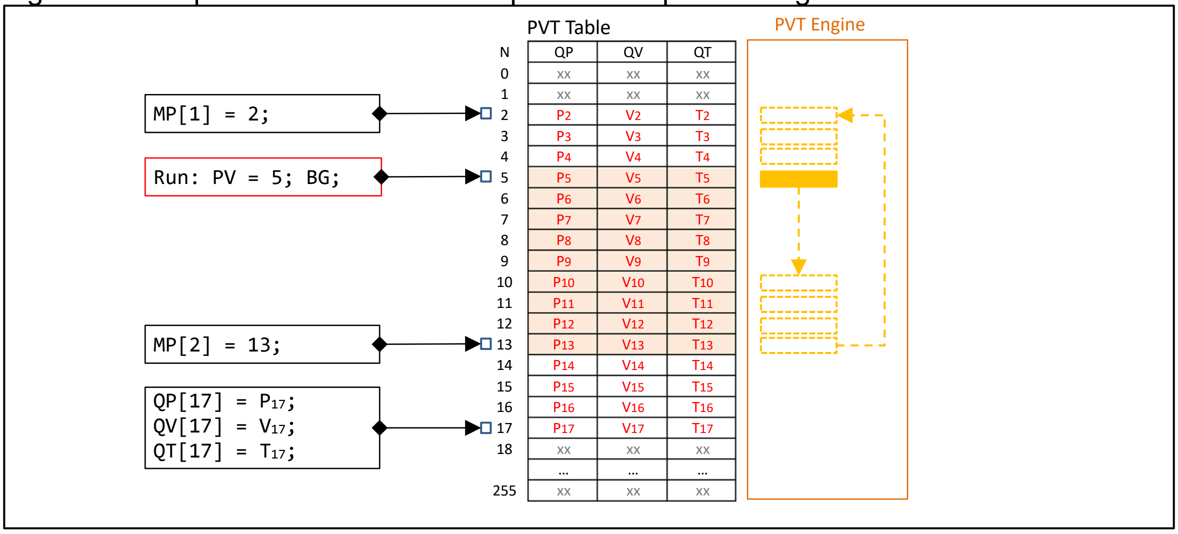
UIM342 Behavior

1) Upon receiving the BG command, PVT motion starts from the 5th row.
2) After calculating the pulse count and frequency, it moves the motor from its current position to QP[5] in QT[5] milliseconds, with the speed of QV[5]. Then UIM342 proceeds to the 6th row.
3) A "Queue Low" Alert will be sent to the host at row 11, 12 and 13.
4) After executing row 13, the motor enters SD stop, sends a "Queue Empty" alarm to the host, and switches the motion mode to JOG.

4.3.2 Loop Mode

When MP[3] = 3, PVT motion operates in **Loop Mode**. The following diagram illustrates how to setup this mode and the UIM342 behaviors.

Figure 4-5: Loop Mode Command and Operation Sequence Diagram



Setup Steps

1) Set PVT configuration:

- `MP[4] = 0;      // Set PVT motion (instead of PT motion)`
- `MP[3] = 3;      // Set Loop Mode`
- `MP[5] = 0;      // Don't care in Loop Mode`
- `MP[2] = 13;     // Set QP[13]/QV[13]/QT[13] as the last valid PVT rows`
- `MP[1] = 2;      // Set QP[2]/QV[2]/QT[2] as the first valid PVT row`

2) Load the PVT list:

- `MP[0] = 0;      // Clear and reset the UIM342 PVT Table`
- `QP[2] = P2; QV[2] = V2; QT[2] = T2;  // Load PVT data into the 2nd row`

  `...`
- `QP[17] = P17; QV[17] = V17; QT[17] = T17;`

3) Enable the driver module (if not already enabled):

- `MO = 1;         // Enable the motor driver`

4) Initialize the PVT mode and set the start point:

- `PV = 5;         // Set PVT Motion, with start point of QP[5]/QV[5]/QT[5]`

5) Start the motion:

- `BG;            // Begin Motion`

UIM342 Behavior

1) Upon receiving the BG command, UIM342 starts executing from the 5th row.

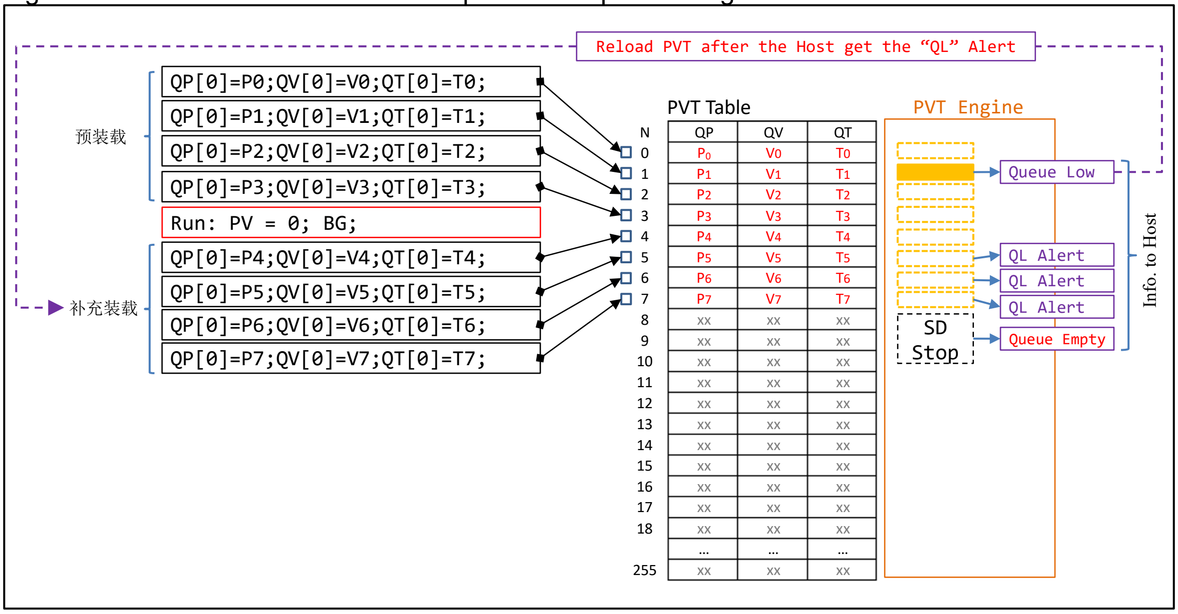
2) After calculating the pulse count and frequency, it moves the motor from the current position to QP[5] in QT[5] milliseconds, with the speed of QV[5]. Then UIM342 proceeds to the 6th row.

3) After the row 13 is completed, the PVT engine reads back to the 2nd row (as defined by MP[1]) and continues to loop.

4.3.3 FIFO Mode (First-In, First-Out)

When MP[3] = 0, PVT motion operates in **FIFO Mode**. The following diagram illustrates how to setup this mode and the UIM342 behaviors.

Figure 4-6: FIFO Mode Command and Operation Sequence Diagram



Setting Steps

1) Set PVT configuration

- `MP[4] = 0;   // Set PVT motion (instead of PT motion)`
- `MP[3] = 0;   // Set FIFO Mode`
- `MP[5] = 3;   // Set “Queue Low” alert for the last 3 PVT rows`
- `MP[2] = 0;   // Set QP[13]/QV[13]/QT[13] as the last valid PVT rows`
- `MP[1] = 0;   // Set QP[2]/QV[2]/QT[2] as the first valid PVT row`

2) Load the PVT list:

- `MP[0] = 0;   // Clear and reset the UIM342 PVT Table`
- `QP[0] = P0; QV[0] = V0; QT[0] = T0;   // Send PVT data to UIM342`
- `QP[0] = P1; QV[0] = V1; QT[0] = T1;   // Don’t care about Index i`
- `QP[0] = P2; QV[0] = V2; QT[0] = T2;`
- `QP[0] = P3; QV[0] = V3; QT[0] = T3;`

Note: Before starting the PVT motion, at least MP[5] + 1 rows of PVT should be loaded.

3) Enable the driver module (if not already enabled):

- `MO = 1;      // Enable the motor driver`

4) Initialize the PVT mode and set the start point:

- `PV = 0;      // Set PVT Motion, FIFO must START from index 0`

5) Start the motion:

- `BG;          // Begin Motion`

6) Load new PVT to UIM342 after receive the “Queue Low” Alert

- `QP[0]= P4; QV[0] = V4; QT[0] = T4;`
- `QP[0]= P5; QV[0] = V5; QT[0] = T5;`
- `QP[0]= P6; QV[0] = V6; QT[0] = T6;`
- `QP[0]= P7; QV[0] = V7; QT[0] = T7;`

Note: The number of supplementary PVTs to load should be based on actual application needs. Loading too many increases startup time; loading too few causes more frequent reloads, placing greater strain on the host.

UIM342 Behavior

1) Upon receiving the BG command, the PVT engine starts to run.
2) When PVT Queue Level (the quantity of remaining PVT rows) ≤ MP[5], a "low water" alert will be sent.
3) When new PVT data arrives, the PVT engine assigns it to the position indicated by MP[6], then increments MP[6]. The valid PVT Queue Level also increases by one.
4) If PVT Queue Level = 0, the motor will SD stop, sends a "Queue Empty" alarm to the host, and switches to JOG mode.

4.3.4 Quick Feeding (QF)

When using FIFO mode, the three instructions QP[i], QV[i], QT[i] can be replaced by a single QF instruction, as defined below:

Figure 4-7: Data Structure of the QF Instruction

QF=N

ID	CW	DL	d0	d1	d2	d3	d4	d5	d6	d7
ID	QF	08	T	V0	V1	V2	P0	P1	P2	P3
			QT		QV			QP		

Where:

[d0] Time, 8-bit unsigned integer, range 1..255

[d3:d2:d1] Speed, 24-bit signed integer, range $-2^{23}..+(2^{23}-1)$

[d7:d6:d5:d4] Position, 32-bit signed integer, range $-2^{31}..+(2^{31}-1)$

To convert QV from a signed 24-bit integer to a signed 32-bit integer, the general approach is to extend the sign of the 24-bit number to fit into 32 bits.

1) The most significant bit (MSB) of QV = [d3] & 0x80;
2) MSB=0 (positive), the 32 bit value will simply copy the bits. For example:

```
24-bit:          00000000 00000101 11000000 (decimal 14464)
32-bit: 00000000 00000000 00000101 11000000 (decimal 14464)
```

3) MSB=1 (negative), the 32 bit value needs to set the Highest Byte to 0xFF. For example:

```
24-bit:          11111111 11111011 01010000 (decimal -14464)
32-bit: 11111111 11111111 11111011 01010000 (decimal -14464)
```

4.4 PT Motion

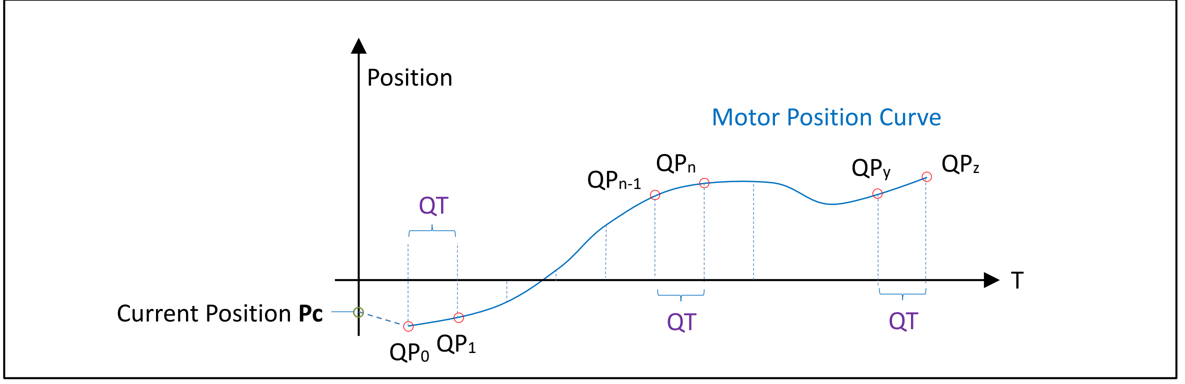
In PT motion, the user only specifies a sequence of absolute positions with equal time intervals, where the time intervals must be an integer multiple of 1 millisecond. Between user-defined positions, the drive interpolates smooth motion.

Similar to PVT motion, PT motion enables precise synchronization across multiple drives. The Host Controller transmits motion data via one compact CAN message (e.g., instruction `PT[n]`), allowing for continuous, real-time updates and virtually unlimited PT sequences.

Each PT point specifies **Position (QP)** and **Time (QT)** (fixed, as defined by MP[4])

The same PVT engine controls the PT motion, using QP and QT, while QV is internally calculated from QP and QT.

Figure 4-8 Definition of PT point N (P[n] / T[n])



To initiate PT motion at a precise moment, a BG command can be sent to a group of motors (using Group-ID) or to all motors (using Global-ID), making PT suitable for coordinated control of multi-axis systems.

To ensure synchronization, the UIM gateway's synchronization function aligns the absolute time counters between all drives, with microsecond-level precision (for detailed information, please refer to the UIM gateway manual).

The Velocity of each point will be calculated by UIM342AB as following:

– **Begin Point:** QV0 = (QP1 - QP0) / QT;
– **Normal Point:** QVn = (QPn+1 - QPn-1) / (2*QT);
– **Final Point:** QVz = (QPz - QPy) / QT;

Following instructions are used to manage the PVT motion:

– MP[i] : Motion Parameter Configuration
– PV : Select PVT Motion and index of the starting row
– PT : PT data feeding

4.5 PT Motion Instructions

The table below lists the control and parameter-setting instructions used in the PT motion:

Parameter	Remark
SD (Stop Deceleration)	Rate of deceleration when motion is killed by queue empty or exception.
MP[N] (Motion Parameters)	MP[1] – First valid row in the PVT table. MP[2] – Last valid row in the PVT table. MP[3] – P/V/T data management mode. 0: FIFO Mode 1: Single Mode 3: Loop Mode MP[4] – Time for PT Motion. 0: PVT motion 5…30000: Time interval of PT motion (Unit: ms) MP[5] – "Queue low" alert value (quantity of remaining PVT row). MP[6] – Index of the next available writing row.
PT[N]	Set Position of PT Motion to UIM342, N is the row index.
PV	Set PVT/PT Motion and inform the index of the starting row.

PT[N] instruction is used to set the position parameter of the PT list inside the UIM342AB controller. Each row in the PT list represents a motion control point. The motion control points should be interpreted as follows:

- **Index = 0**

  After time QT (as defined by MP[4]), the motor moves from the current position to PT[0].

- **Index = 1**

  After time QT, the motor moves from QP[0] to QP[1].

- **Index = N**

  After time QT, the motor moves from QP[n-1] to QP[n].

At any control point, QV is calculated by UIM342AB (refer to section 4.4 for the calculation).

4.6 PT Motion Management

PT motion management is the same as PVT motion management.

4.7 PVT / PT Comparison

The following table compares PVT and PT motions:

Feature	PVT	PT
Max PVT Table size	256 rows	512 rows
Velocity calculation QV	Specified by user	By UIM342AB
Acceleration calculation	By UIM342AB	By UIM342AB
Time Interval QT	Fixed, as defined by MP[4]	Specified by user, 5...255ms
Cyclical motion support	Yes	Yes
On-the-fly motion programming	Yes	Yes

The following table provides recommendations when selecting the PVT or PT motion:

Feature	Preferred
Long preprogrammed motions	PT
Ease of use	PT
Variable QT for more accurate motion	PVT
Synchronized, multiple-axis motions	PVT

4.8 PVT / PT Termination

PVT motion terminates upon of the following events:

1) The motor is shut down, either by MO=0 or by an exception.
2) Another mode of motion is set active (e.g., by PA=xxx; BG;). If so, the new motion command will be executed immediately.
3) The PVT Queue runs out of data.
   - For Single Mode, this occurs when the PVT sets defined by MP[2] is executed.
   - For FIFO Mode, this occurs when the PVT Queue is empty. If so, the PVT motion is stopped immediately, using the SD deceleration.

Note: Make sure that the final QV is zero, in order to make the PVT motion terminates neatly and the stop at the end of the motion.

4.9 Precautions for Using PVT/PT

Key points to pay attention to when operating in PVT mode:

1) **Instruction Order**:
   - Configure MP[i] at first, since MP[4] will affect the selection of PVT or PT.
   - PVT data must be fed in the order of QP, QV and QT, especially in FIFO mode.
   - The PV=N instruction sets the index of the starting row N in the PVT table.
   - The BG command triggers the motion starting from the coordinates indicated by the read pointer.

2) **Dynamic Programming:**

   – It is possible to update the PVT table during motion, allowing dynamic changes.
   – In loop mode (MP[3] = 3), the table can be modified to generate infinite non-periodic motions.

3) **Efficient Table Management**:

   – The Host tracks the execute and write indexes of the PVT table to manage table space.
   – The Host controls the writing process and checks MP[6] when index is not clear.
   – The PV instruction allows the host to check the status of the current execute pointer.

4) **Non-Interfering Updates**:

   The host can prepare the next motion segment while the current segment is running, ensuring smooth transitions between segments.

5) **Modification Restrictions**:

   Any attempt to modify an active segment will result in an error.

5.0 I/O & Input Logic Control

The input logic control module has the following functions:

Processing and reporting the logic levels of input ports,

Executing user preset actions, when the input logic level change is detected:

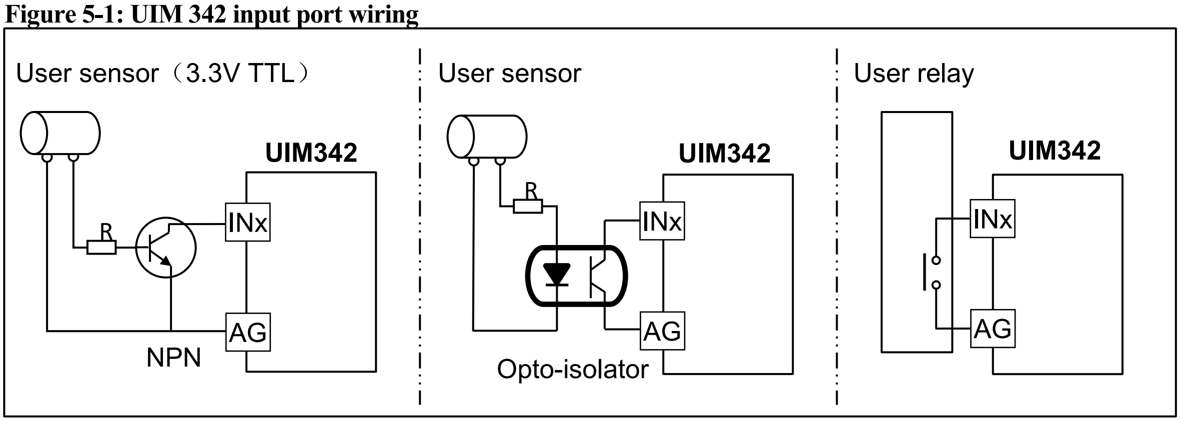
1) Turn off the Motor Driver (freewheel);
2) Emergency stop;
3) Decelerate until stop;
4) Set origin then turn around and go to relative position;
5) Set origin then Emergency stop;
6) Set origin then decelerate until stop;
7) Turn around and move in constant speed (Jog);
8) Start to move in constant speed (Jog);
9) Turn around and go to relative position;
10) Go to relative position;
11) Go to absolute position;
12) Turn off the Motor Driver (freewheel) + Lock down motion control (become read only);
13) Emergency stop + Lock down motion control (become read only);

5.1 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 5-1.

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 5-1: UIM 342 input port wiring



5.2 Trigger type and Input Filter

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

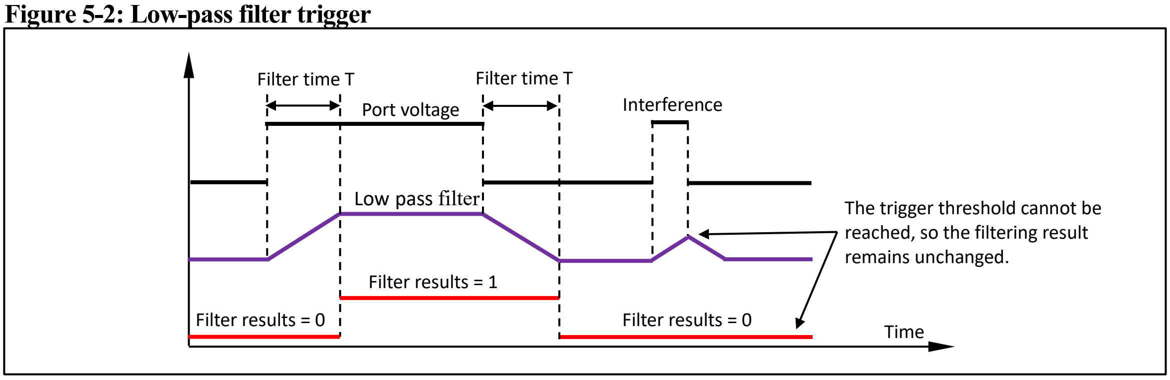
- **Continuous**

This type allows the UIM342AB continuously sampling the voltage on input ports. If a logic level change is detected, UIM342AB will immediately notified the user and execute the preset logic actions. To use this trigger, set "`TG[i]=0`", i=0, 1 or 2.

- **Low-Pass Filter**

This type allows the UIM342AB performing digital low-pass filtering on the input and before collects the level changes. This can eliminate jitters and prevent interference from digital inputs.

Figure 5-2: Low-pass filter trigger



To use this mode, set "`TG[i]` = T" (i=0, 1 or 2). This instruction sets the filter time T (valid range is 1...60000 ms) of Input i.

- **One-Time Shoot**

In this mode, UIM342 will processing a port level change only once. After that, the action 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.

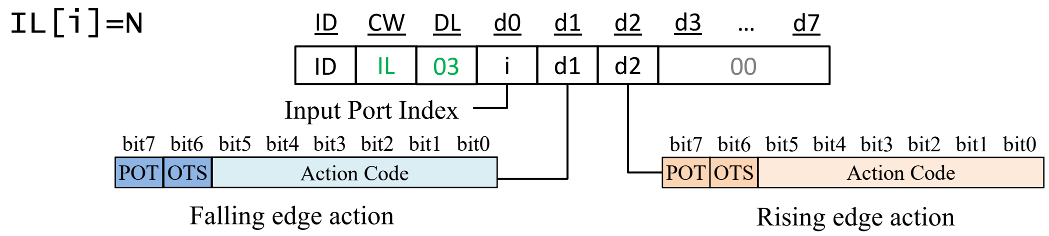
To use this mode, please refer to the following section "Configure Input Logic Action", and set the OTS bit to 1.

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

5.3 Configure Input Logic Action

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

Figure 5-3: Instruction to attach Logic Action



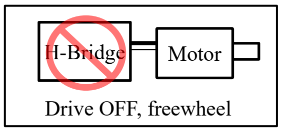
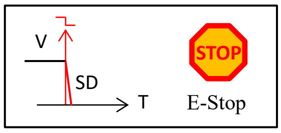
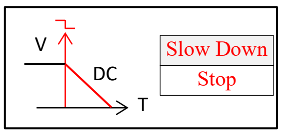
POT – Power-On Trigger
0 = Force to trigger on Power-up, disregard the actual input. This will create a power-on action.
1 = Disable Power-On Trigger.

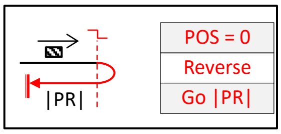
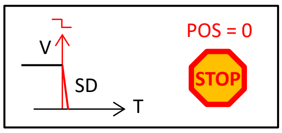
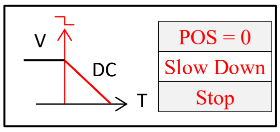
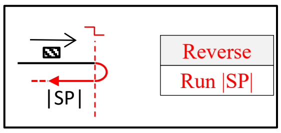
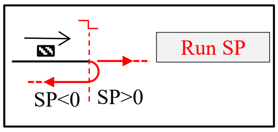
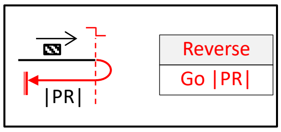
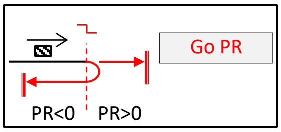
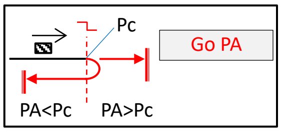
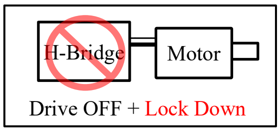
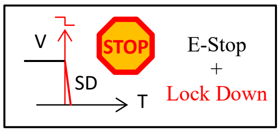
OTS – One Time Shoot
0 = Continuous executing Action Code as long as an input logic level change is detected.
1 = Reset the Action Code when once the Action Code is triggered and executed.

i – Index of Input Ports

0, 1, 2 = 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)	

<table>
<tr><td>0x04</td><td>Set origin then go reversed relative position (using |PR|, SP, AC, DC)</td><td></td></tr>
<tr><td>0x05</td><td>Set origin then Emergency stop (using SD)</td><td></td></tr>
<tr><td>0x06</td><td>Set origin then decelerate to stop (using DC)</td><td></td></tr>
<tr><td>0x07</td><td>Reversed Jogging (using |SP|, AC, DC)</td><td></td></tr>
<tr><td>0x08</td><td>Jog (using SP, AC, DC)</td><td></td></tr>
<tr><td>0x09</td><td>Go reverse relative position (using |PR|, SP, AC, DC)</td><td></td></tr>
<tr><td>0x0A</td><td>Go relative position (using PR, SP, AC, DC)</td><td></td></tr>
<tr><td>0x0B</td><td>Go absolute position (using PA, SP, AC, DC)</td><td></td></tr>
<tr><td>0x0D</td><td>Turn off Motor Driver, then Lock Down the Motor Controller (can inquiry but cannot set motion or output)</td><td></td></tr>
<tr><td>0x0E</td><td>Emergency stop, then Lock Down the Motor Controller (can inquiry but cannot set motion or output)</td><td></td></tr>
</table>

5.4 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, 2 represents IN1, IN2, IN3 respectively.

6.0 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:
   - INST – Instruction, from Host (user) to UIM342.
   - ACK – Acknowledgment / Reply, from UIM342 to Host (user).

6.1 P[i] Protocol Parameter

Protocol Parameter

CW	No ACK	0x01	Need ACK	0x81
Data Format	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

Get "CAN Bitrate", PP[5];

```
INS  ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
     05 81 01 05 00 00 00 00 00 00 00

ACK  ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
     05 01 02 05 03 00 00 00 00 00 00
```

Index [d0] = 0x05; (PP[5]).
Data [d1] = 0x03; (PP[5] = 3, CAN Bitrate = 250K).

Get "Node ID", PP[7];

```
INS  ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
     05 81 01 07 00 00 00 00 00 00 00

ACK  ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
     05 01 02 07 05 00 00 00 00 00 00
```

Index [d0] = 0x07; (PP[7]).
Data [d1] = 0x07; (PP[7] = 5, Node ID = 5).

Get "Group ID", PP[8];

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

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

Index [d0] = 0x08; (PP[8]).
Data [d1] = 0x0A; (PP[8] = 10, Group ID = 10).

<table>
<tr><td>Example
SET</td><td>
<u>Set “CAN Bitrate=500K”, PP[5]=2;</u>

INS   ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
       05 81 02 05 02 00 00 00 00 00 00
       Index [d0] = 0x05; (PP[5]).
       Data [d1] = 0x02; (PP[5] = 10).

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

<u>Set “CAN Node ID=12”, PP[7]=12;</u>

INS   ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
       05 81 02 07 0B 00 00 00 00 00 00
       Index [d0] = 0x07; (PP[7]).
       Data [d1] = 0x0B; (PP[7] = 12).

ACK   ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
       05 01 02 07 0B 00 00 00 00 00 00

<u>Set “CAN Group ID=20”, PP[8]=20;</u>

INS   ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
       05 81 02 08 14 00 00 00 00 00 00
       Index [d0] = 0x08; (PP[8]).
       Data [d1] = 0x14; (PP[8] = 20).

ACK   ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
       05 01 02 08 14 00 00 00 00 00 00
</td></tr>
<tr><td>Note</td><td>
– 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.
– The set value will be saved to EEPROM only when MO=0; otherwise, it stays in RAM and is lost after power off.
</td></tr>
</table>

6.2 IC[i] Initial Configuration

System Settings

<table>
<tr><td>CW</td><td>No ACK</td><td>0x06</td><td>Need ACK</td><td>0x86</td></tr>
<tr><td>Data Format</td><td colspan="2">i (Unsigned 8-bit)</td><td colspan="2">Data Format (Unsigned 16-bit)</td></tr>
<tr><td>Description</td><td colspan="4">

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

<table>
<tr><td>i</td><td>Description</td><td colspan="2">Value (N)</td></tr>
<tr><td>0</td><td>Auto-Enable Motor Driver after Power up</td><td>0: disable</td><td>1: enable;</td></tr>
<tr><td>1</td><td>Positive Motor Direction</td><td>0: CW</td><td>1: CCW</td></tr>
<tr><td>3</td><td>Enable Lockdown System</td><td>0: disable</td><td>1: enable</td></tr>
<tr><td>4</td><td>Units for AC and DC</td><td>0: pulse/sec²</td><td>1: millisecond</td></tr>
<tr><td>6</td><td>Using Closed-loop Control</td><td>0: Open loop</td><td>1: Closed-loop</td></tr>
<tr><td>7</td><td>Enable Software Limit</td><td>0: disable</td><td>1: enable</td></tr>
</table>

</td></tr>
<tr><td>Example GET</td><td colspan="4">

Get the Setting of “Auto-Enable Motor Driver after Power up”, IC[0];

<pre>
INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
     05 86 01 00 00 00 00 00 00 00 00
</pre>
Index [d0] = 0x00; (IC[0])

<pre>
ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
     05 06 03 00 01 00 00 00 00 00 00
</pre>
Index [d0] = 0x00; (IC[0])
Data [d2:d1] = 0x0001; (IC[0] = Enabled).

Get the Setting of “Positive Motor Direction”, IC[1];

<pre>
INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
     05 86 01 01 00 00 00 00 00 00 00
</pre>
Index [d0] = 0x01; (IC[1])

<pre>
ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
     05 06 03 01 01 00 00 00 00 00 00
</pre>
Index [d0] = 0x01; (IC[1])
Data [d2:d1] = 0x0001; (IC[1] = 1, CCW).

</td></tr>
</table>

<table>
<tr><td>Example GET</td><td>

Get the Setting of "Enable Lockdown System", IC[3];

```
INS   ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
      05 86 01 03 00 00 00 00 00 00 00
```
Index [d0] = 0x03; (IC[3])

```
ACK   ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
      05 06 03 03 01 00 00 00 00 00 00
```
Index [d0] = 0x03; (IC[3])
Data [d2:d1] = 0x0001; (IC[3] = 1, Lockdown Control System is Enabled).

Get the Setting of "Units for AC and DC", IC[4];

```
INS   ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
      05 86 01 04 00 00 00 00 00 00 00
```
Index [d0] = 0x04; (IC[4])

```
ACK   ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
      05 06 03 04 01 00 00 00 00 00 00
```
Index [d0] = 0x04; (IC[4])
Data [d2:d1] = 0x0001; (IC[4] = 1, AC/DC's units is millisecond).

Get the Setting of "Using Closed-loop Control", IC[6];

```
INS   ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
      05 86 01 06 00 00 00 00 00 00 00
```
Index [d0] = 0x06; (IC[6])

```
ACK   ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
      05 06 03 06 01 00 00 00 00 00 00
```
Index [d0] = 0x06; (IC[6])
Data [d2:d1] = 0x0001; (IC[6] = Using Closed-loop Control).

Get the Setting of "Enable Software Limit", IC[7];

```
INS   ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
      05 86 01 07 00 00 00 00 00 00 00
```
Index [d0] = 0x07; (IC[7])

```
ACK   ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
      05 06 03 07 01 00 00 00 00 00 00
```
Index [d0] = 0x07; (IC[7])
Data [d2:d1] = 0x0001; (IC[7] = Using Software Limits).

</td></tr>
</table>

Example SET

Set Auto-Enable Motor Driver after Power up, IC[0]=1;

```
INS  ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
     05 86 01 00 01 00 00 00 00 00 00
```
Index [d0] = 0x00; (IC[0])
Data [d2:d1] = 0x0001; (IC[0] = 1, Enable).

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

Set Positive Motor Direction = CCW, IC[1]=1;

```
INS  ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
     05 86 01 01 01 00 00 00 00 00 00
```
Index [d0] = 0x01; (IC[1])
Data [d2:d1] = 0x0001; (IC[1] = 1, CCW).

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

Set Enable Lockdown System = Enable, IC[3]=1;

```
INS  ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
     05 86 01 03 01 00 00 00 00 00 00
```
Index [d0] = 0x03; (IC[3])
Data [d2:d1] = 0x0001; (IC[3] = 1, Lockdown Control System is Enabled).

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

Set the units for AC and DC to millisecond, IC[4]=1;

```
INS  ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
     05 86 01 04 01 00 00 00 00 00 00
```
Index [d0] = 0x04; (IC[4])
Data [d2:d1] = 0x0001; (IC[4] = 1).

```
ACK  ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
     05 06 03 04 01 00 00 00 00 00 00
```

Get the Setting of "Using Closed-loop Control", IC[6];

```
INS  ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
     05 86 01 06 01 00 00 00 00 00 00
```
Index [d0] = 0x06; (IC[6])
Data [d2:d1] = 0x0001; (IC[6] = Using 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
```

Example SET	Set Enable Software Limits, IC[7]=1; `INS  ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `     05 86 01 07 01 00 00 00 00 00 00` *Index [d0] = 0x07; (IC[7])* *Data [d2:d1] = 0x0001; (IC[7] = Using Software Limits).* `ACK  ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `     05 06 03 07 01 00 00 00 00 00 00`
Note	– 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. – The set value will be saved to EEPROM only when MO=0; otherwise, it stays in RAM and is lost after power off.

6.3 IE[i] Inform Enable

System Settings

CW	No ACK	0x07	Need ACK	0x87
Data Format	i (Unsigned 8-bit)		Data (Unsigned 16-bit)	

Description

IE[i] — **Get Inform Enable Configuration**

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

IE[i]=N — **Set Inform 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

Get IN1's IE configure, IE[0];

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

Index [d0] = 0x00; (IN1's IE configure)

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

Index [d0] = 0x00; (IN1's IE configure)
Data [d2:d1] = 0x0001; (Enabled).

Example INS

Set Enable IN2's change notification, IE[1]=1;

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

Index [d0] = 0x01; (IN2's change notification).
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
```

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.
- The set value will be saved to EEPROM only when MO=0; otherwise, it stays in RAM and is lost after power off.

6.4 ML Model

System Settings

CW	No ACK	n/a	Need ACK	0x8B
Data Format	Segment Data			

Description

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

INST data length	0	Data	n/a
ACK data length	8	Data	d0… d7

Example GET

Get Model, ML;

```
INS  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 01 00 00 00 00 00
```

For UIM342CM series, please refer to the following table for model info.

model	d0	d1	d2	d3	d5:d4	d7:d6
342XS	0x20	0x0A	0x01	xx	Firmware version	xxxx
342S	0x21	0x0A	0x01	xx	Firmware version	xxxx
342H02	0x22	0x14	0x09	xx	Firmware version	xxxx
342H04	0x22	0x28	0x09	xx	Firmware version	xxxx
342H08	0x22	0x50	0x09	xx	Firmware version	xxxx

x – Factory use, don’t care.

For UIM342AB series, please refer to the following table for model info.

model	d0	d1	d2	d3	d5:d4	d7:d6
342XSAB	0x20	0x0A	0x02	xx	Firmware version	xxxx
342SAB	0x21	0x0A	0x02	xx	Firmware version	xxxx
342H02AB	0x22	0x14	0x0A	xx	Firmware version	xxxx
342H04AB	0x22	0x28	0x0A	xx	Firmware version	xxxx
342H08AB	0x22	0x50	0x0A	xx	Firmware version	xxxx

x – Factory use, don’t care.

6.5 SN Serial Number

System Settings

CW	No ACK	n/a	Need ACK	0x8C
Data Format	Segment Data			
Description	**SN** **Get the serial number of the device** INST data length 0 Data n/a ACK data length 8 Data d0…d7			
Example GET	Get the Serial Number, SN; `INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `05 8C 00 00 00 00 00 00 00 00 00` *Index [d0] = 0x00;* `ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `05 0C 08 01 02 03 04 05 06 07 08` *[d3:d2:d1:d0] = 0x04030201 (Serial number 0067305985).* *[d5:d4] = 0x0605 (Manufacturer ID 1541).* *[d7:d6] = 0x0807 (Vendor ID 2055).*			

6.6 ER[i] Error Report

System Settings

<table>
<tr><td>CW</td><td>No ACK</td><td>0x0F</td><td>Need ACK</td><td>0x8F</td></tr>
<tr><td>Data Format</td><td colspan="2">i (Unsigned 8-bit)</td><td colspan="2">Data (Unsigned 8-bit)</td></tr>
<tr><td>Description</td><td colspan="4">
ER[i] Get Error Information
INST data length 1 Data d0 (=i)
ACK data length 6 Data d0 (=i), d1, d2, d3, d4, d5
ER[0]=0 Clear All Error Info.
INST data length 2 Data d0 (=i), d1, d2
ACK data length 6 Data d0 (=i), d1, d2, d3, d4, d5
<table>
<tr><td>i</td><td>Description</td></tr>
<tr><td>0</td><td>Get the Newest Error Info. / Reset All Error Info.</td></tr>
<tr><td>10…18</td><td>Get the Historic 1st…9th (Newest…Oldest) Error Info.</td></tr>
</table>
Refer to chapter 2.3 “Error Report” for the description of d0…d5.
</td></tr>
<tr><td>Example GET</td><td colspan="4">
<u>Get the Newest Error Info., ER[0];</u>
INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
05 8F 01 00 00 00 00 00 00 00 00
Index [d0] = 0x00;
ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
05 0F 06 00 33 81 05 00 00 00 00
Index [d0] = 0x00;
Data [d1] = 0x33; (Error code = 0x33, i.e. instruction data error).
Data [d2] = 0x81; (CW related to the error = 0x81, i.e. PP).
Data [d3] = 0x05; (Sub-Index of the CW related to the error = 5).
Data [d5:d4] = 0x00; (Reserve).
The newest error = "In the instruction PP[5]=N, the data is illegal".
</td></tr>
<tr><td>Example INS</td><td colspan="4">
<u>Reset All Error, ER[0]=0;</u>
INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
05 8F 02 00 00 00 00 00 00 00 00
Index [d0] = 0x00; (All Error).
Data [d1] = 0x00; (Reset Error).
ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
05 0F 06 00 00 00 00 00 00 00 00
Index [d0] = 0x00; (All Error).
Data [d1]… [d5] = 0; (No Error).
</td></tr>
</table>

6.7 QE[i] Quadrature Encoder

System Settings

CW	No ACK	0x3D	Need ACK	0xBD
Data Format	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

Get Counts per Revolution for closed-loop control, QE[1];

```
INS  ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
     05 BD 01 04 00 00 00 00 00 00 00
```

Index [d0] = 0x04;

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

Data [d2:d1] = 0x0C80; (Counts per revolution = 3200).

Example INS

Set Lines per Revolution of the encoder, QE[0]=1000;

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

Index [d0] = 0x00; (Lines per Revolution)
Data [d2:d1] = 0x03E8; (Lines per Revolution = 1000).

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

Note

– QE[0] (LPR) is a physical property of the encoder. Setting it incorrectly can cause unpredictable behavior.

– 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.

– The set value will be saved to EEPROM only when MO=0; otherwise, it stays in RAM and is lost after power off.

6.8 SY[i] System Operation

System Settings

CW	No ACK	0x7E	Need ACK	n/a
Data Format	i (Unsigned 8-bit)		No Data	
Description	**SY[i] System Operation** INST data length 1 Data d0 (=i) No ACK			
Example SET	Reboot the device, SY[1]; `INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `05 7E 01 01 00 00 00 00 00 00 00` *Index [d0] = 0x01;* Restore factory defaults, SY[2]; `INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `05 7E 01 02 00 00 00 00 00 00 00` *Index [d0] = 0x02;*			

i	Description
1	Reboot the system
2	Restore factory defaults

6.9 MT[i] Motor Driver

Motor Driver

<table>
<tr><td>CW</td><td>No ACK</td><td>0x10</td><td>Need ACK</td><td>0x90</td></tr>
<tr><td>Data Format</td><td colspan="2">i (Unsigned 8-bit)</td><td colspan="2">Data (Unsigned 16-bit)</td></tr>
<tr><td>Description</td><td colspan="4">

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

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

</td></tr>
<tr><td>Example GET</td><td colspan="4">

<u>Get Working Current, MT[1];</u>

INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
05 90 03 01 00 00 00 00 00 00 00
Index [d0] = 0x01; (Get 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
Data [d2:d1] = 0x001C; (Working Current = 2.8Amp).

</td></tr>
<tr><td>Example INS</td><td colspan="4">

<u>Set Micro-stepping resolution to 16, MT[0]=16;</u>

INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
05 90 03 00 10 00 00 00 00 00 00
Index [d0] = 0x00; (Micro-stepping Resolution).
Data [d2:d1] = 0x0010; (Micro-stepping Resolution = 16).

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

</td></tr>
<tr><td>Note</td><td colspan="4">

– The set value will be saved to EEPROM only when MO=0; otherwise, it stays in RAM and is lost after 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.

</td></tr>
</table>

6.10 MO Motor Driver On /Off

Motor Driver

<table>
<tr><td>CW</td><td>No ACK</td><td>0x15</td><td>Need ACK</td><td>0x95</td></tr>
<tr><td>Data Format</td><td colspan="4">Unsigned 8-bit</td></tr>
<tr><td>Description</td><td colspan="4">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.</td></tr>
<tr><td>Example GET</td><td colspan="4"><u>Get motor driver on/off status, MO;</u>
INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
05 95 00 00 00 00 00 00 00 00 00

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] = 0x01; (The Motor Driver is ON).</td></tr>
<tr><td>Example INS</td><td colspan="4"><u>Set the Motor Driver OFF, MO=0;</u>
INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
05 95 01 00 00 00 00 00 00 00 00
Data [d0] = 0x00; (SET the 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</td></tr>
<tr><td>Note</td><td colspan="4">– 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.</td></tr>
</table>

6.11 BG Begin Motion

Motion Control

CW	No ACK	0x16	Need ACK	0x96
Data Format	n/a			
Description	**BG** **Begin Motion** INST data length 0 Data n/a ACK data length 4 Data d0, d1, d2, d3			
Example SET	Begin Motion, BG; `INS  ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `     05 96 00 00 00 00 00 00 00 00 00` `ACK  ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `     05 16 00 00 00 00 00 00 00 00 00` *Data [d3:d2:d1:d0] = 0x00000000; (don't care).*			
Note	– Activate the newly set parameters and initiate motion. This applies to JOG, PTP, and PVT motion control. The motor will remain stationary until BG is entered. Once BG is entered, motion parameters (e.g., SP/PR/PA/JV/PV) will be activated, and motion will begin. – For simultaneous movement of multiple UIM342 units, the user can first set parameters such as SP/PA for each node ID, and then send BG to the global or group ID.			

6.12 ST Stops Motion

Motion Control

CW	No ACK	0x17	Need ACK	0x97
Data Format	n/a			
Description	**ST** **Stop Motion** INST data length 0 Data n/a ACK data length 0 Data n/a			
Example SET	Stop Motion, ST; `INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `05 97 00 00 00 00 00 00 00 00 00` *Data [d0] = 0x03; (MF = 3).* `ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `05 17 00 00 00 00 00 00 00 00 00`			
Note	– Stop current movement, using stop deceleration (SD) to decelerate.			

6.13 MF Motion Parameter Frame

Motion Control

<table>
<tr><td>CW</td><td>No ACK</td><td>0x18</td><td>Need ACK</td><td>0x98</td></tr>
<tr><td>Data Format</td><td colspan="4">Unsigned 8-bit</td></tr>
<tr><td>Description</td><td colspan="4">

`MF=N` **Set/select motion parameter group to be operated.**
INST data length 1 Data d0
ACK data length 1 Data d0

<table>
<tr><th>N</th><th>Description</th></tr>
<tr><td>0</td><td>Select the motion parameter group for normal operation</td></tr>
<tr><td>1</td><td>Reserved</td></tr>
<tr><td>2</td><td>Select the motion parameter group for the rising edge of IN1</td></tr>
<tr><td>3</td><td>Select the motion parameter group for the falling edge of IN1</td></tr>
<tr><td>4</td><td>Select the motion parameter group for the rising edge of IN2</td></tr>
<tr><td>5</td><td>Select the motion parameter group for the falling edge of IN2</td></tr>
<tr><td>6</td><td>Select the motion parameter group for the rising edge of IN3</td></tr>
<tr><td>7</td><td>Select the motion parameter group for the falling edge of IN3</td></tr>
</table>

</td></tr>
<tr><td>Example
SET</td><td colspan="4">

Select the motion parameter group for the falling edge of IN1, MF=3;

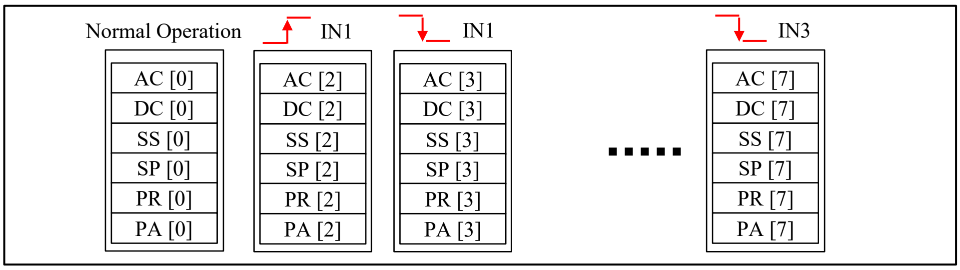
```
INS  ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
     05 98 01 03 00 00 00 00 00 00 00
```

Data [d0] = 0x03; (MF = 3).

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

</td></tr>
<tr><td>Note</td><td colspan="4">

– 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.



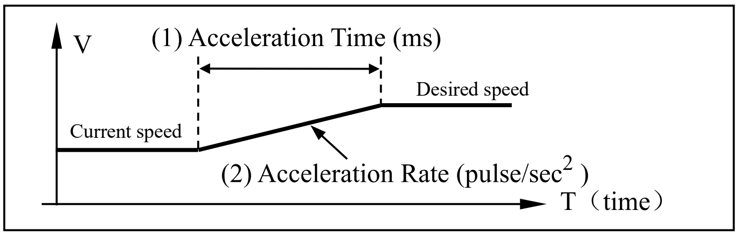
– For example, to set the acceleration of IN1 falling edge to 1000, using: MF=1; AC=1000; To get the acceleration of IN1 falling edge, using: MF=1; AC;

– After setting/getting a parameter, MF value will automatically reset to 0.

</td></tr>
</table>

6.14 AC Acceleration

Motion Control

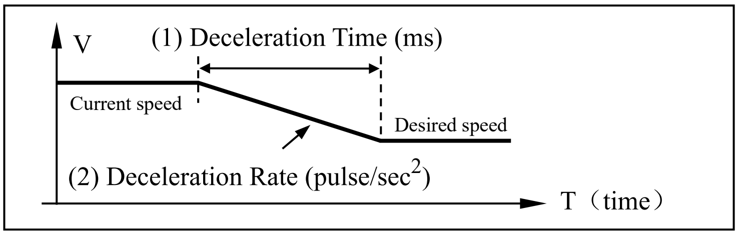
CW	No ACK	0x19	Need ACK	0x99
Data Format	Unsigned 32-bit Range: 0…2^{32} pulse/sec^2			
Description	**AC** **Get Acceleration** INST data length 0 Data n/a ACK data length 4 Data d0, d1, d2, d3 **AC=N** **Set Acceleration** INST data length 4 Data d0, d1, d2, d3 ACK data length 4 Data d0, d1, d2, d3 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	Get Acceleration, AC; INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7 05 99 00 00 00 00 00 00 00 00 00 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] = 0x000003e8 (The Acceleration is 1000).*			
Example SET	Set Acceleration 500, AC=500; INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7 05 99 04 F4 01 00 00 00 00 00 00 *Data [d3:d2:d1:d0] = 0x000001e4 (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			
Note	– The set value will be saved to EEPROM only when MO=0; otherwise, it stays in RAM and is lost after power off. – After setting, AC will takes effect when the next BG is commanded.			

6.15 DC Deceleration

Motion Control

<table>
<tr><td>CW</td><td>No ACK</td><td>0x1A</td><td>Need ACK</td><td>0x9A</td></tr>
<tr><td>Data Format</td><td colspan="4">Unsigned 32-bit
Range: 0…2^{32} pulse/sec^2</td></tr>
<tr><td>Description</td><td colspan="4">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)

</td></tr>
<tr><td>Example
GET</td><td colspan="4"><u>Get Deceleration, DC;</u>

<code>INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7</code>
<code>   05 9A 00 00 00 00 00 00 00 00 00</code>

<code>ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7</code>
<code>   05 1A 04 E8 03 00 00 00 00 00 00</code>
Data [d3:d2:d1:d0] = 0x000003e8 (The Deceleration is 1000).</td></tr>
<tr><td>Example
INS</td><td colspan="4"><u>Set Deceleration 1000, DC=1000;</u>

<code>INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7</code>
<code>   05 9A 04 E8 03 00 00 00 00 00 00</code>
Data [d3:d2:d1:d0] = 0x000003e8 (SET Deceleration 1000).

<code>ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7</code>
<code>   05 1A 04 E8 03 00 00 00 00 00 00</code></td></tr>
<tr><td>Note</td><td colspan="4">– The set value will be saved to EEPROM only when MO=0; otherwise, it stays in RAM and is lost after power off.

– After setting, DC will takes effect when the next BG is commanded.</td></tr>
</table>

6.16 SS Cut-in Speed

Motion Control

CW	No ACK	0x1B	Need ACK	0x9B
Data Format	Unsigned 32-bit Range: 0…2^{32} pulse/sec^2			
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			
Example GET	Get Cut-in speed, SS; `INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `    05 9B 00 00 00 00 00 00 00 00 00` `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] = 0x000003e8 (Cut-in Speed is 1000).*			
Example INS	Set Cut-in speed 1001, SS=1001; `INS 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] = 0x000003e9 (SET cut-in speed 1001).* `ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `    05 9C 04 00 6A 18 00 00 00 00 00`			
Note	– The set value will be saved to EEPROM only when MO=0; otherwise, it stays in RAM and is lost after power off.			

6.17 SD Stop Deceleration

Motion Control

CW	No ACK	0x1C	Need ACK	0x9C
Data Format	Unsigned 32-bit Range: 0…2^{32} pulse/sec^2			
Description	**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			
Example GET	Get Stop Deceleration, SD; `INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `    05 9C 00 00 00 00 00 00 00 00 00` `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 = 1,600,000).*			
Example INS	Set Stop Deceleration 1,600,000, SD=1600000; `INS 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] = 0xFFFFFC18; (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`			
Note	– Mechanical movement should not be stopped abruptly, as it may cause impact and damage to the machine. Therefore, a suitable maximum acceleration value must be set, which is the purpose of the SD value. – 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. – The set value will be saved to EEPROM only when MO=0; otherwise, it stays in RAM and is lost after power off.			

6.18 JV Jog Velocity

Motion Control

<table>
<tr><td>CW</td><td>No ACK</td><td>0x1D</td><td>Need ACK</td><td>0x9D</td></tr>
<tr><td>Data Format</td><td colspan="4">Signed 32-bit
Range: $-2^{31}...+(2^{31} -1)$ pulse/sec</td></tr>
<tr><td>Description</td><td colspan="4">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</td></tr>
<tr><td>Example GET</td><td colspan="4"><u>Get current speed, JV;</u>
INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
05 9D 00 00 00 00 00 00 00 00 00

ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
05 1D 04 18 FC FF FF 00 00 00 00
Data [d3:d2:d1:d0] = 0xFFFFFC18; (Current speed = -1000).</td></tr>
<tr><td>Example INS</td><td colspan="4"><u>Set desired velocity -1000, JV= -1000;</u>
INS 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; (JV = -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 Velocity).
Data [d4:d3:d2:d1] = 0xFFFFFC18; (Desired relative position -1000).</td></tr>
<tr><td>Note</td><td colspan="4">– 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.</td></tr>
</table>

6.19 SP PTP Speed

Motion Control

<table>
<tr><td>CW</td><td>No ACK</td><td>0x1E</td><td>Need ACK</td><td>0x9E</td></tr>
<tr><td>Data Format</td><td colspan="4">Signed 32-bit
Range: $-2^{31}...+(2^{31}-1)$ pulse/sec</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</td></tr>
<tr><td>Example GET</td><td colspan="4">Get current speed, SP;
INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
05 9E 00 00 00 00 00 00 00 00 00
ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
05 1E 04 18 FC FF FF 00 00 00 00
Data [d3:d2:d1:d0] = 0xFFFFFC18; (Current speed = -1000).</td></tr>
<tr><td>Example SET</td><td colspan="4">Set desired velocity +1000, SP= +1000;
INS 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;
ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
05 2E 05 02 E8 03 00 00 00 00 00
CW = 0x2E; (i.e. DV);
Sub-index d0 = 2; (Desired Velocity).
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>

6.20 PR Position Relative

Motion Control

<table>
<tr><td>CW</td><td>No ACK</td><td>0x1F</td><td>Need ACK</td><td>0x9F</td></tr>
<tr><td>Data Format</td><td colspan="4">Signed 32-bit
Range: -2^{31}...+(2^{31} -1) pulse</td></tr>
<tr><td>Description</td><td colspan="4">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</td></tr>
<tr><td>Example
GET</td><td colspan="4"><u>Get current relative position, PR;</u>
INS   ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
      05 9F 00 00 00 00 00 00 00 00 00
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).</td></tr>
<tr><td>Example
SET</td><td colspan="4"><u>Set desired relative position +1000, PR= +1000;</u>
INS   ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
      05 9F 04 E8 03 00 00 00 00 00 00
Data [d3:d2:d1:d0] = 0x000003e8;
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; (Desired relative position +1000).</td></tr>
<tr><td>Note</td><td colspan="4">– 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.</td></tr>
</table>

6.21 PA Position Absolute

Motion Control

<table>
<tr><td>CW</td><td>No ACK</td><td>0x20</td><td>Need ACK</td><td>0xA0</td></tr>
<tr><td>Data Format</td><td colspan="4">Signed 32-bit
Range: $-2^{31}...+(2^{31}-1)$ pulse</td></tr>
<tr><td>Description</td><td colspan="4">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</td></tr>
<tr><td>Example GET</td><td colspan="4"><u>Get current absolute position, PA;</u>
INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
05 A0 00 00 00 00 00 00 00 00 00

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; (PA= -1000).</td></tr>
<tr><td>Example INS</td><td colspan="4"><u>Set desired absolute position +1000, PA= +1000;</u>
INS 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;

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; (Desired absolute position +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.

– PA will take effective after a following BG instruction.</td></tr>
</table>

6.22 OG Set Origin

Motion Control

CW	No ACK	0x21	Need ACK	0xA1
Data Format	n/a			
Description	**OG** — **Set the current position as the origin by clearing the position counter** INST data length 0 Data n/a ACK data length 0 Data n/a			
Example SET	**Set Origin, OG;** `INS 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	– OG is disabled while the motor is moving. – The origin can be set via input logic during motion.			

6.23 BL Backlash Compensation

Motion Control

CW	No ACK	0x2D	Need ACK	0xAD
Data Format	Data (unsigned 32-bit) Range: 0…65535			
Description	**BL** **Get Backlash Compensation Value** INST data length 0 Data n/a ACK data length 4 Data d0, d1, d2(=0), d3(=0) **BL=N** **Set Backlash Compensation Value** INST data length 4 Data d0, d1, d2(=0), d3(=0) ACK data length 4 Data d0, d1, d2(=0), d3(=0)			
Example GET	Get Backlash Compensation Value, BL; INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7 05 AD 00 00 00 00 00 00 00 00 00 ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7 05 2D 04 E8 03 00 00 00 00 00 00 *Data [d1:d0] = 0x03E8; (BL=1000 pulse).* *Data [d3:d2] = 0; (Don't care, always=0).*			
Example INS	Set Backlash Compensation =1001, BL=1001; INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7 05 AD 00 E9 03 00 00 00 00 00 00 *Data [d1:d0] = 0x03E9; (BL=1001 pulse).* *Data [d3:d2] = 0; (Don't care, always=0).* ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7 05 2D 04 E9 03 00 00 00 00 00 00			
Note	– The set value will be saved to EEPROM only when MO=0; otherwise, it stays in RAM and is lost after power off.			

6.24 MS[i] Motion Status

Motion Control

<table>
<tr><td>CW</td><td>No ACK</td><td>0x11</td><td>Need ACK</td><td>0x91</td></tr>
<tr><td>Data Format</td><td colspan="2">i (unsigned 8-bit)</td><td colspan="2">Data (unsigned 8-bit)</td></tr>
<tr><td>Description</td><td colspan="4">

MS[0] **Get Status Flags and Relative Position**
INST data length 1 Data d0=0;
ACK data length 8 Data d0=0, d1. . .d7;

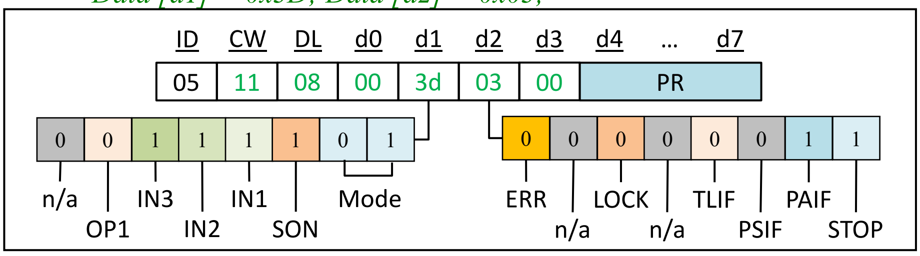
MS[1] **Get the Current Speed and Absolute Position**
INST data length 1 Data d0=1;
ACK data length 8 Data d0=1, d1. . .d7;

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.
</td></tr>
<tr><td>Example GET</td><td colspan="4">

Get Current Motion Status, MS[0];

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

ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
05 11 08 00 3D 03 00 18 FC FF FF
Data [d0] = 0x00 (Index = 0).
Data [d1] = 0x3D; Data [d2] = 0x03;


Current Relative Position [d7:d6:d5:d4] = 0xFFFFFC18 = -1000 (pulse);
</td></tr>
<tr><td>Example SET</td><td colspan="4">

Reset Motion Status, MS[0]=0;

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

ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
05 11 08 00 3D 01 00 18 FC FF FF
</td></tr>
</table>

6.25 DV[i] Desired Values

Motion Control

CW	No ACK	0x2E	Need ACK	0xAE
Data Format	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

Get Desired Velocity, DV[2];

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

Data [d0] = 0x02 (Index = 2, Desired Velocity).

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

Data [d0] = 0x02 (Index = 2, Desired Velocity).

Data [d4:d3:d2:d1] = 0x000003e8 (Desired Velocity = +1000 p/s).

6.26 `IL[i]` Input Logic

Input Logic

<table>
<tr><td>CW</td><td>No ACK</td><td>0x34</td><td>Need ACK</td><td>0xB4</td></tr>
<tr><td>Data Format</td><td colspan="2">i (unsigned 8 bit)</td><td colspan="2">Data (unsigned 16 bit)</td></tr>
<tr><td>Description</td><td colspan="4">
IL[i] Get Sensor Trigger Action
INST data length 1 Data d0 (=i)
ACK data length 3 Data d0 (=i), d1, d2

IL[i]=X Set Sensor Trigger Action
INST data length 3 Data d0 (=i), d1, d2
ACK data length 3 Data d0 (=i), d1, d2

<table>
<tr><th>i</th><th>Description</th></tr>
<tr><td>0</td><td>Action after IN1 is triggered</td></tr>
<tr><td>1</td><td>Action after IN2 is triggered</td></tr>
<tr><td>2</td><td>Action after IN3 is triggered</td></tr>
<tr><td>3</td><td>Behavior when motor stalls, N=0: lock down, N=1: freewheel</td></tr>
</table>
Please refer to chapter 5.3 “Configure Input Logic Action” for details.
</td></tr>
<tr><td>Example GET</td><td colspan="4">
<u>Get IN1 trigger action, IL[0];</u>
INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
05 B4 01 00 00 00 00 00 00 00 00
Data [d0] = 0x00 (Index = 0, IN1).
ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
05 34 03 00 80 80 00 00 00 00 00
Data [d0] = 0x00 (Index = 0, IN1).
Data [d1] = 0x80 (Falling edge: POT = disabled, Action Code = 0).
Data [d2] = 0x80 (Rising edge: POT = disabled, Action Code = 0).
</td></tr>
<tr><td>Example SET</td><td colspan="4">
<u>Set IN1 Trigger Action, ILx 00 06 08;</u>
INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
05 B4 03 00 06 80 00 00 00 00 00
Data [d0] = 0x00 (Index = 0, IN1).
Data [d1] = 0x06 (Falling edge: POT = Enable; Action Code = 6).
Data [d2] = 0x80 (Rising edge: POT = Disable; Action Code = 0).
ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
05 34 03 00 06 80 00 00 00 00 00
</td></tr>
<tr><td>Note</td><td colspan="4">– The set value will be saved to EEPROM only when MO=0; otherwise, it stays in RAM and is lost after power off.</td></tr>
</table>

6.27 TG[i] Trigger

Input Logic

CW	No ACK	0x35	Need ACK	0xB5
Data Format	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 for IN1 (Port 1)
1	Trigger mode for IN2 (Port 2)
2	Trigger mode for IN3 (Port 3)

N	Description
0	Trigger continuously
1 … 60000	Trigger after Low pass filter, filter time 1…60000 ms

Please refer to chapter 5.2 "Trigger Type and Input Filter" for details.

Example GET

Get IN2 Trigger Type, TG[1];

```
INS  ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
     05 B5 01 01 00 00 00 00 00 00 00
```
Data [d0] = 0x01 (Index = 1, IN2).

```
ACK  ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
     05 35 03 01 00 00 00 00 00 00 00
```
Data [d0] = 0x01 (Index = 1, IN2).
Data [d2:d1] = 0x0000 (TG[0] = 0, Continous Type).

Example SET

Set IN1 Low-pass Filter Time Constant 100 ms, TG[0]=100;

```
INS  ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
     05 B5 03 00 64 00 00 00 00 00 00
```
Data [d0] = 0x00 (Index = 0, 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
```

Note

– The set value will be saved to EEPROM only when MO=0; otherwise, it stays in RAM and is lost after power off.

6.28 DI Digital I /O

Input Logic

CW	No ACK	0x37	Need ACK	0xB7
Data Format	Segment Data			
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) **Dixd0…d7** **Set Digital Output** INST data length 4 Data d0, d1(=1), d2(=0), d3(=0) ACK data length 4 Data d0, d1, d2(=0), d3(=0)			
Example GET	Get digital input & output logic level, DI; `INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `05 B7 00 00 00 00 00 00 00 00 00` `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	Set output 1 logic level = HIGH, DI = 1; `INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `05 B7 04 01 01 00 00 00 00 00 00` *d0.bit0 = 1 (OP1 = 1);* *d1.bit0 = 1 (OP1 Mask = 1, valid).* `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 = 1 (OP1 = 1).*			

6.29 RT Real-Time Inform

Real-time Notification

CW	0x5A
Data Format	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 10 00 00 00 00 00 00 *Data [d0] = 0x01 (Falling edge is detected on IN1).*

6.30 MP[i] PVT Motion Parameter

PVT Motion

CW	No ACK	0x22	Need ACK	0xA2
Data Format	i (unsigned 8-bit)		Data (unsigned 16-bit)	
Description	MP[0] – Get Current Queue Level; Reset the PVT Table. MP[1] – Get/Set First valid row in the PVT Table. MP[2] – Get/Set Last valid row in the PVT Table. MP[3] – Get/Set PVT data management mode. 0: FIFO Mode 1: Single Mode 3: Loop Mode MP[4] – Get/Set Time for PT Motion. 0: PVT motion MP[5] – Get/Set “Queue low” alert value (quantity of remaining PVT row). MP[6] – Get/Set Index of the next available writing row.			
Example GET	Get Current Queue Level, MP[0]; `INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `    05 A2 01 00 00 00 00 00 00 00 00` `ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `    05 22 03 00 10 00 00 00 00 00 00` *Data [d0] = 0x00 (Index = 0).* *Data [d2:d1] = 0x0010 (MP[0] = 16, Queue Level = 16).* Get the first valid row in the PVT Table, MP[1]; `INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `    05 A2 01 01 00 00 00 00 00 00 00` `ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `    05 22 03 01 02 00 00 00 00 00 00` *Data [d0] = 0x01 (Index = 1).* *Data [d2:d1] = 0x0002 (MP[1] = 2).* Get the last valid row in the PVT Table, MP[2]; `INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `    05 A2 01 02 00 00 00 00 00 00 00` `ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `    05 22 03 02 20 00 00 00 00 00 00` *Data [d0] = 0x02 (Index = 2).* *Data [d2:d1] = 0x0020 (MP[2] = 32).*			

Example GET	<u>Get current PVT mode, MP[3];</u> `INS   ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `      05 A2 01 03 00 00 00 00 00 00 00` `ACK   ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `      05 22 03 03 00 00 00 00 00 00 00` *Data [d0] = 0x03 (Index = 3).* *Data [d2:d1] = 0x0000 (MP[3] = 0, FIFO Mode).* <u>Get current PT time interval, MP[4];</u> `INS   ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `      05 A2 01 04 00 00 00 00 00 00 00` `ACK   ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `      05 22 03 04 32 00 00 00 00 00 00` *Data [d0] = 0x04 (Index = 4).* *Data [d2:d1] = 0x0032 (MP[4] = 50, PT motion time interval = 50 ms).* <u>Get “Queue Low” notify value, MP[5];</u> `INS   ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `      05 A2 01 05 00 00 00 00 00 00 00` `ACK   ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `      05 22 03 05 03 00 00 00 00 00 00` *Data [d0] = 0x05 (Index = 5).* *Data [d2:d1] = 0x0003 (MP[5] = 3).* *Note: MP[5] range 0…253.* <u>Get the index next available of writing row, MP[6];</u> `INS   ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `      05 A2 01 06 00 00 00 00 00 00 00` `ACK   ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `      05 22 03 06 05 00 00 00 00 00 00` *Data [d0] = 0x06 (Index = 6).* Data [d2:d1] = 0x0005 (MP[6] = 5, index of next available writing = 5).

Example SET	Set Reset the PVT Table , MP[0] = 0; `INS  ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `     05 A2 03 00 00 00 00 00 00 00 00` *Data [d0] = 0x00 (Index = 0).* *Data [d2:d1] = 0x0000 (MP[0] = 0).* `ACK  ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `     05 22 03 00 00 00 00 00 00 00 00` Set the first valid row in the PVT Table, MP[1]=2; `INS  ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `     05 A2 03 01 02 00 00 00 00 00 00` *Data [d0] = 0x01 (Index = 1).* *Data [d2:d1] = 0x0002 (MP[1] = 2).* `ACK  ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `     05 22 03 01 02 00 00 00 00 00 00` Set the last valid row in the PVT Table, MP[2]=20; `INS  ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `     05 A2 03 02 14 00 00 00 00 00 00` *Data [d0] = 0x02 (Index = 2).* *Data [d2:d1] = 0x0014 (MP[2] = 14).* `ACK  ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `     05 22 03 02 14 00 00 00 00 00 00` Set PVT Loop Mode, MP[3]=3; `INS  ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `     05 A2 03 03 03 00 00 00 00 00 00` *Data [d0] = 0x03 (Index = 3).* *Data [d2:d1] = 0x0003 (MP[3] = 3, Loop Mode).* `ACK  ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `     05 22 03 03 03 00 00 00 00 00 00` Set current PT time interval=100ms, MP[4]=100; `INS  ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `     05 A2 03 04 64 00 00 00 00 00 00` *Data [d0] = 0x04 (Index = 4).* *Data [d2:d1] = 0x0064 (MP[4] = 100).* `ACK  ID CW DL d0 d1 d2 d3 d4 d5 d6 d7` `     05 22 03 04 64 00 00 00 00 00 00` ***Important Note**: Set MP[4] = 0; if using PVT motion.*

<table>
<tr><td>Example
SET</td><td>
<u>Set “Queue Low” notification value, MP[5]=5;</u>

<code>INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7</code>
<code>   05 A2 03 05 05 00 00 00 00 00 00</code>

<code>ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7</code>
<code>   05 22 03 05 05 00 00 00 00 00 00</code>
Data [d0] = 0x05 (Index = 5).
Data [d2:d1] = 0x0005 (MP[5] = 5).
Note: MP[5] range 0…253.

<u>Set the index of the next writing row, MP[6]=12;</u>

<code>INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7</code>
<code>   05 A2 03 06 0C 00 00 00 00 00 00</code>
Data [d0] = 0x06 (Index = 6).
Data [d2:d1] = 0x000C (MP[6] = 12).

<code>ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7</code>
<code>   05 22 03 06 0C 00 00 00 00 00 00</code>
</td></tr>
</table>

6.31 PV Set PVT Motion

PVT Motion

<table>
<tr><td>CW</td><td>No ACK</td><td>0x23</td><td>Need ACK</td><td>0xA3</td></tr>
<tr><td>Data Format</td><td colspan="4">Data (unsigned 16-bit)
Range: 0…65535</td></tr>
<tr><td>Description</td><td colspan="4">PV   Get the index of PVT execute row
INST data length 0 Data n/a
ACK data length 8 Data d0…d7
PV=X   Set PVT Motion & the index of starting row
INST data length 2 Data d0, d1
ACK data length 8 Data d0…d7</td></tr>
<tr><td>Example GET</td><td colspan="4"><u>Get index of current executing row;</u>
SET ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
05 A3 00 00 00 00 00 00 00 00 00
ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
05 23 08 21 00 xx xx xx xx xx xx
Data [d1:d0] = 0x0021 (executing point = 33).
Data [d4:d3:d2:d1] = 0xFFFFFC18 (QP[5] = -1000).
xx – don't care.</td></tr>
<tr><td>Example SET</td><td colspan="4"><u>Set PVT Mode and Starting Row = 2, PV = 2;</u>
SET ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
05 A3 02 02 00 00 00 00 00 00 00
Data [d1:d0] = 0x0002 (PV = 2).
ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
05 23 02 02 00 xx xx xx xx xx xx
xx – don't care.</td></tr>
</table>

6.32 QP[N] Queued Position

PVT Motion

CW	No ACK	0x25	Need ACK	0xA5
Data Format	N (unsigned 8-bit) Range: 0…255		Data (signed 32-bit) Range: -2^{31} … $+(2^{31}-1)$ pulse	
Description	**QP[N]** **Get Position value of row N** INST data length 1 Data d0 (=n) ACK data length 5 Data d0 (=n), d1, d2, d3, d4 **QP[N]=X** **Set Position value of row N** INST data length 5 Data d0 (=n), d1, d2, d3, d4 ACK data length 5 Data d0 (=n), d1, d2, d3, d4			
Example GET	Get position value of PVT table row 5; INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7 05 A5 01 05 00 00 00 00 00 00 00 ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7 05 25 05 05 18 FC FF FF 00 00 00 *Data [d0] = 0x05; (Index = 5).* *Data [d4:d3:d2:d1] = 0xFFFFFC18 (QP[5] = -1000).*			
Example SET	Set QP[3] = 1000 pulse; INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7 05 A6 05 03 E8 03 00 00 00 00 00 *Data [d0] = 0x03; (Index = 3).* *Data [d4:d3:d2:d1] = 0x000003E8 (QP[3] = 1000).* ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7 05 26 05 03 E8 03 00 00 00 00 00			

6.33 QV[N] Queued Velocity

PVT Motion

CW	No ACK	0x26	Need ACK	0xA6
Data Format	N (unsigned 8-bit) Range: 0…255		Data (signed 32-bit) Range: -2^{31} … $+(2^{31}-1)$ p/s	
Description	**QV[N]** **Get Velocity value of row N** INST data length 1 Data d0 (=n) ACK data length 5 Data d0 (=n), d1, d2, d3, d4 **QV[N]=X** **Set Velocity value of row N** INST data length 5 Data d0 (=n), d1, d2, d3, d4 ACK data length 5 Data d0 (=n), d1, d2, d3, d4			
Example GET	Get velocity value of PVT table row 5; INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7 05 A6 01 05 00 00 00 00 00 00 00 ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7 05 26 05 05 18 FC FF FF 00 00 00 *Data [d0] = 0x05; (Index = 5).* *Data [d4:d3:d2:d1] = 0xFFFFFC18 (QV[5] = -1000).*			
Example SET	Set QV[6] = 3 p/s; INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7 05 A6 05 06 03 00 00 00 00 00 00 *Data [d0] = 0x05; (Index = 6).* *Data [d4:d3:d2:d1] = 0x00000003 (QV[6] = 3).* ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7 05 26 03 06 03 00 00 00 00 00 00			

6.34 QT[N] Queued Time

PVT Motion

CW	No ACK	0x27	Need ACK	0xA7
Data Format	N (unsigned 8-bit) Range: 0…255		Data(unsigned 8-bit) Range: 5...255 ms	
Description	**QT[N]** **Get Time value of row N** INST data length 1 Data d0 (=n) ACK data length 3 Data d0 (=n), d1, d2 **QT[N]=X** **Set Time value of row N** INST data length 3 Data d0 (=n), d1, d2 ACK data length 3 Data d0 (=n), d1, d2			
Example GET	Get Time value of PVT table row 5; SET ID CW DL d0 d1 d2 d3 d4 d5 d6 d7 05 A7 01 05 00 00 00 00 00 00 00 ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7 05 27 03 05 18 00 00 00 00 00 00 *Data [d0] = 0x05; (Index = 5).* *Data [d2:d1] = 0x0018 (QT[5] = 24 ms).*			
Example SET	Set QT[6] = 100 ms; SET ID CW DL d0 d1 d2 d3 d4 d5 d6 d7 05 A7 01 06 64 00 00 00 00 00 00 *Data [d0] = 0x06; (Index = 6).* *Data [d2:d1] = 0x0064 (QT[6] = 100 ms).* ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7 05 27 03 06 64 00 00 00 00 00 00			

6.35 QF Quick Feeding PVT data

PVT Motion

<table>
<tr><td>CW</td><td>No ACK</td><td>0x29</td><td>Need ACK</td><td>0xA9</td></tr>
<tr><td>Data Format</td><td colspan="4">Segment data;</td></tr>
<tr><td>Description</td><td colspan="4">QF[N] Get P/V/T value of row N
INST data length 1 Data d0 (=N)
ACK data length 8 Data d0…d7
QFxd0…d7 Set P/V/T value of row N
INST data length 8 Data d0…d7
ACK data length 8 Data d0…d7</td></tr>
<tr><td>Example GET</td><td colspan="4">Get QP[10]/QV[10]/QT[10], QF[10];
INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
05 A9 01 0A 00 00 00 00 00 00 00
Data [d0] = 0x0A (Index =10, QF[10]).
ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
05 29 08 64 E8 03 00 10 72 00 00
Data [d0] = 0x64 (QT=100 ms).
Data [d3:d2:d1] = 0x0003E8 (QV=1000 p/s)
Data [d7:d6:d5:d4] = 0x00002710 (QP=10000 p/s)
Note: Index will not be included in the ACK message.</td></tr>
<tr><td>Example SET</td><td colspan="4">Set QP = 10000; QV = -1000; QT = 50;
INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
05 A9 08 32 18 FC FF 10 27 00 00
Note: Don't care about row index, as it will be handled by the PVT engine
ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7
05 29 08 32 18 FC FF 10 27 00 00
Data [d0] = 0x32 (QT=50 ms).
Data [d3:d2:d1] = 0xFFFC18 (QV= -1000 p/s)
Data [d7:d6:d5:d4] = 0x00002710 (QP=10000 p/s)
Note: Index will not be included in the ACK message.</td></tr>
</table>

6.36 PT[N] Set PT data

PVT Motion

<table>
<tr><td>CW</td><td>No ACK</td><td>0x24</td><td>Need ACK</td><td>0xA4</td></tr>
<tr><td>Data Format</td><td colspan="4">Segment data;</td></tr>
<tr><td>Description</td><td colspan="4">PT[N] Get PT Position value of row N
INST data length 2 Data d0, d1
ACK data length 8 Data d0…d7
PTxd0…d7 Set PT Position value of row N
INST data length 8 Data d0…d7
ACK data length 8 Data d0…d7</td></tr>
<tr><td>Example GET</td><td colspan="4">Get Position value of row 266;
<code>INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7</code>
<code>05 A4 02 0A 01 00 00 00 00 00 00</code>
Data [d1:d0] = 0x010A (index=266).
<code>ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7</code>
<code>05 24 08 0A 01 A0 86 01 00 00 00</code>
Data [d1:d0] = 0x010A (index=266).
Data [d5:d4:d3:d2] = 0x0001A086 (QP=100,000 p/s).
Data [d7:d6] Don't care.</td></tr>
<tr><td>Example SET</td><td colspan="4">Set Position value of row 266 to 100,000 (pulse);
<code>INS ID CW DL d0 d1 d2 d3 d4 d5 d6 d7</code>
<code>05 A4 02 0A 01 A0 86 01 00 00 00</code>
Data [d1:d0] = 0x010A (index=266).
Data [d5:d4:d3:d2] = 0x0001A086 (QP=100,000 p/s).
Data [d7:d6] Don't care.
<code>ACK ID CW DL d0 d1 d2 d3 d4 d5 d6 d7</code>
<code>05 24 08 0A 01 A0 86 01 00 00 00</code></td></tr>
</table>

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” , qty = 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;
}
```