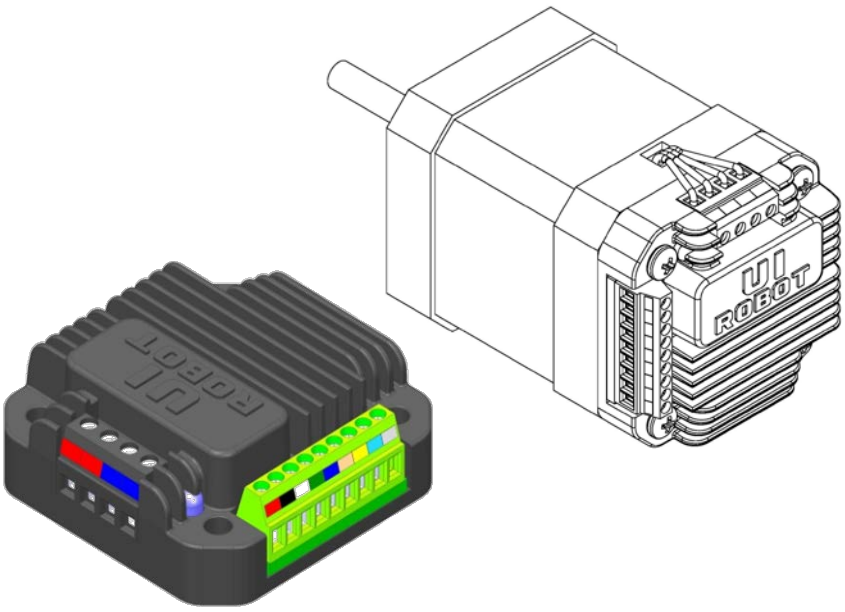
UIROBOT®
United Intelligence Robot Technology

User Manual

UIM242XX Series
CAN2.0B Instruction Control
Miniature Integrated Stepper Motor Controller



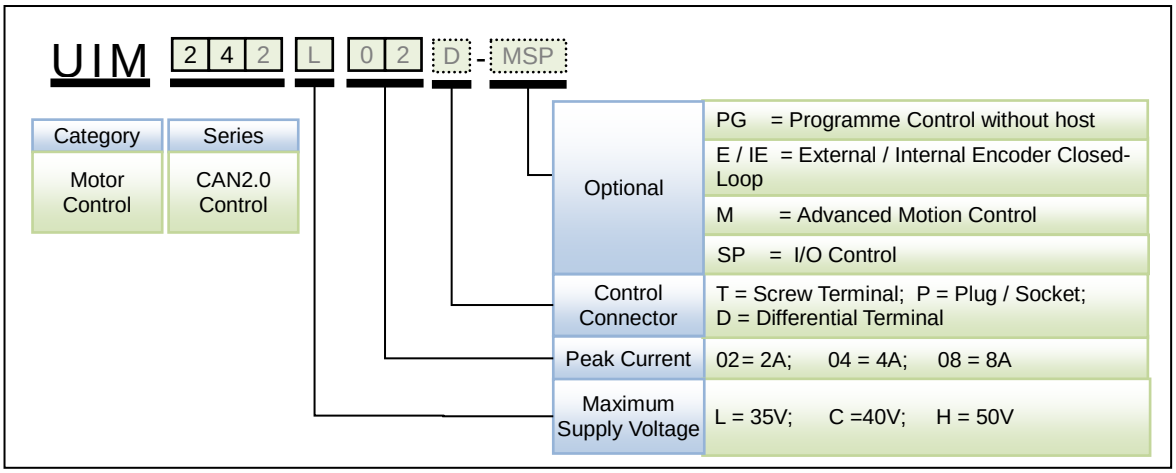
Please pay attention to the following before using the UIROBOT products:

1. UIROBOT products meet the specification contained in their particular Data Sheet.
2. UIROBOT will only work with the customer who respects the Intellectual Property (IP) protection.
3. Attempts to break UIROBOT's IP protection feature may be a violation of the local Copyright Acts. If such acts lead to unauthorized access to UIROBOT's IP work, UIROBOT has a right to sue for relief under that Act.

[UIM242XX Ordering Information]

In order to serve you quicker and better, please provide the product number in following format.

UIM242XX PART NUMBERING SYSTEM



Note:

1） Peak current is decided by max. supply voltage (See in Table 0-1).
2） -H product (Max. supply voltage is 50V) is custom made, please contact with salesmans before purchase.
3） Default control connector is T (screw terminal), if not selected.
4） -D product (Differential Terminal) is custom made, please contact with salesmans before purchase.
5） -PG (Programme Control without Host), need the hardware model be 1232 or higher.

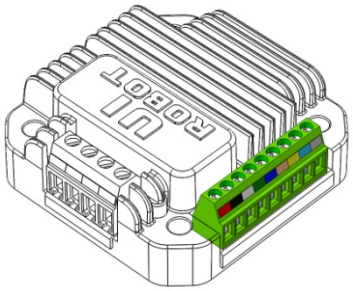
Table 0-1 Correspondence between Max. Supply Voltage and Peak Current

Voltage / Current	L（35V）	C（40V）	H（50V）
2A	√	√	√
4A	×	√	√
8A	×	√	√

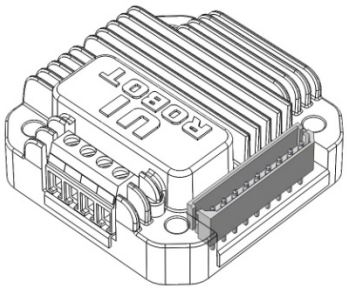
Examples：

UIM242L02T，UIM242L02D， UIM242C04P-MSP， UIM242H08P-IE

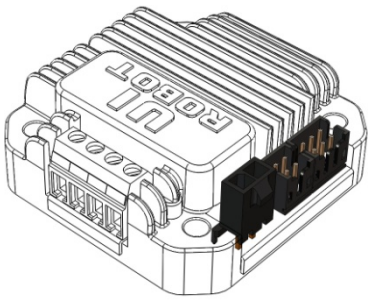
Examples of Control Connector options:



Screw Terminal



Rectangular Plug / Socket



Differential Terminal

UIM24202 / 04 / 08
CAN2.0Instruction Control
Miniature Integrated Stepper Motor Contrller

Miniature Integral Design

- Miniature size 42.3mm*42.3mm*16.5mm
- Fit onto motors seamlessly
- Die-cast aluminum enclosure, improving heat dissipation and durability

Motor Driving Characteristics

- Wide supply voltage range 12 ~ 50VDC*
- Output current 2/4/8A, instruction adjustable
- Full to 16th micro-step resolution
- Dual full H-bridge with PWM constant current control

Network Communication

- CAN2.0 A / B
- 2-wire interface, max 1M bps operation, long distance
- Differential bus, high noise immunity, max 100 nodes

Embedded DSP Microprocessor

- Hardware DSP, 64bits calculating precision
- Simple instructions, intuitive and fault-tolerating
- Intelligent control, intuitive and fault-tolerating
- SDK and underlying control drive of host
- VC++, C, C# , VB demo

Advanced Motion Control

- Absolute position record / feedback, Power-failure position protection
- Quadrature encoder based closed-loop control
- linear and non-linear acceleration and deceleration, S-curve, PT/PVT displacement control
- Backlash compensation

Advanced I / O Control (without host)

- 3 sensor input ports, 1 analog input (12bit)
- 1 TTL output
- 3 trigger mode (continuous / intermittent / single)
- 6 independent motion parameter group
- Pre-set action controlled by I / O
- I / O real-time event-based change notification
- 12 real-time event based change notifications
- 13 programmable actions

Others

- Initial status configurate
- Auto-lock when emergency
- User program
- Regeneration discharge module (sold separately)

*-H product (Max. supply voltage is 50V) is custom made, please contact with salesmans before purchase.

General Description

UIM24202 / UIM24204 / UIM24208 are miniature stepper motor controllers with CAN network interface. UIM242 controllers can be mounted onto NEMA17/23/34/42 series stepper motor through adapting flanges. Total thickness of the controller is less than 16.5mm.

With UIM242 controller, it is simple to construct a control system. Users can control the whole “motor-sensor-third party actuator” system through their own CAN based host by using “SimpleCAN” protocol. Users also can control the system through a gateway produced by UIrobot, such as UIM2501, USBC9100 and PCI120, by using RS232 based string or “SimpleCAN” protocol. One gateway can network with up to 100 UIM242 controllers.

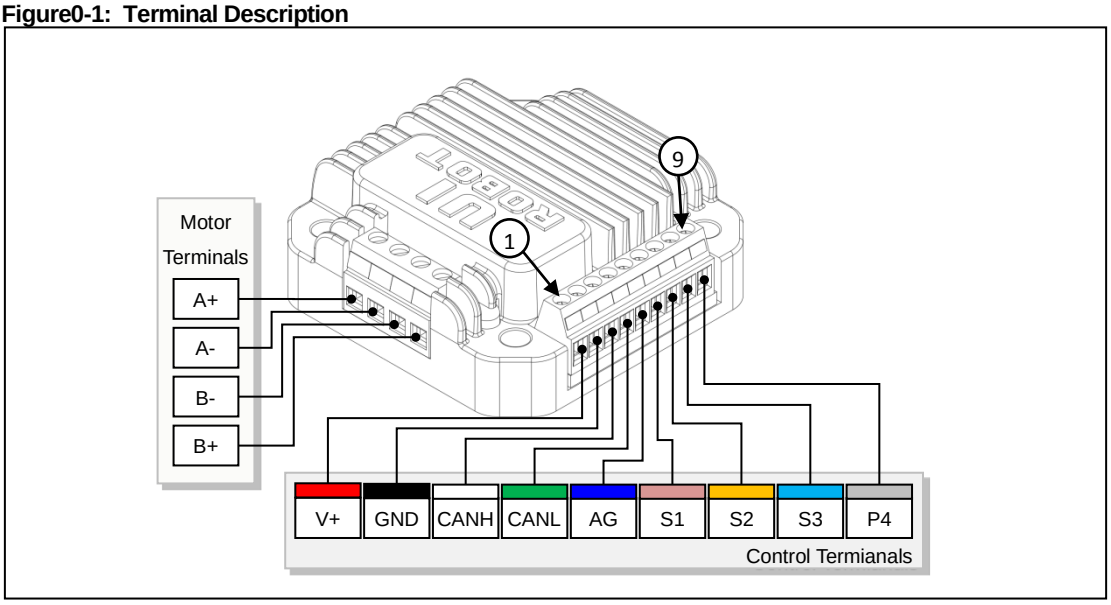
UIM242 can realize open-loop and encoder-based closed-loop control. Its architecture includes communication system, basic motion control system, advanced motion control module (linear/non-linear acceleration/deceleration, S-curve PT/PVT displacement control), sensor input control module, TTL output control module and user programming module.

Embedded 64-bit calculating precision DSP controller guarantees the entire control process finish within 1 millisecond. Instructions are simple and intuitive. UIROBOT provides free Microsoft Windows based VB/VC demo software and corresponding source code.

Enclosure is made of die-cast aluminum to provide a rugged durable protection and improves the heat dissipation.

TERMINAL DESCRIPTION(-T/P)

Figure0-1: Terminal Description



Control Terminals

Terminal No.	Designator	Description
1	V+	Supply voltage, 12 - 50VDC*
2	GND	Supply voltage ground
3	CANH	CAN signal dominant high
4	CANL	CAN signal dominant low
5	AG	Analog ground for sensors
6	S1	Sensor input port 1
7	S2	Sensor input port 2
8	S3	Sensor input port 3
9	P4	TTL signal output port

Motor Terminals

Terminal No.	Description
A+ / A-	Connect to the stepper motor phase A
B+ / B-	Connect to the stepper motor phase B

*-H product (Max. supply voltage is 50V) is custom made, please contact with salesmans before purchase.

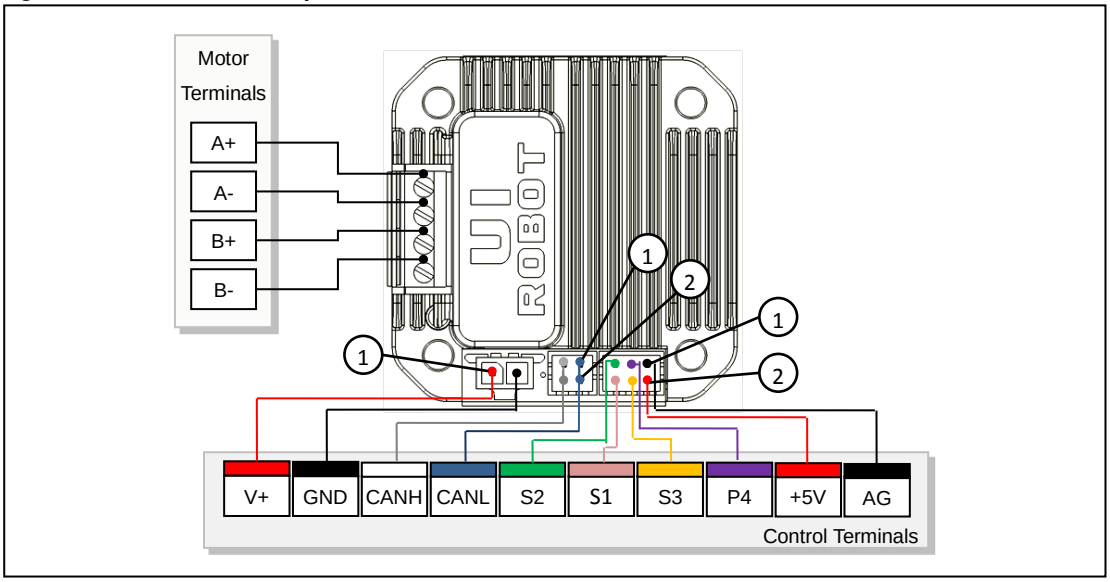
WARNING: Incorrect connection of phase winds will permanently damage the controller!

Resistance between leads of different phases is usually > 100KΩ. Resistance between leads of the same phase is usually < 100Ω. It can simply measured by a multimeter.

WARNING: Except supply voltage port and motor terminal, voltage on port must be kept between -0.3~5.3V. Otherwise, the controller will be damaged.

TERMINAL DESCRIPTION(-D)

Figure0-2: Termianal Description



Control Terminals

Terminal No.	Designator	Description
Two-core1	V+	Supply voltage, 12 - 40VDC
Two-core2	GND	Supply voltage ground
Four-core3, 4	CANH	CAN signal dominant high
Four-core1, 2	CANL	CAN signal dominant low
Six-core1	AG	Analog ground for sensors
Six-core2	+5V	Voltage output (5V, 80mA)
Six-core3	P4	TTL signal output port
Six-core4	S3	Sensor input port 3
Six-core5	S2	Sensor input port 2
Six-core6	S1	Sensor input port 1

Motor Terminals

Terminal No.	Description
A+ / A-	Connect to the stepper motor phase A
B+ / B-	Connect to the stepper motor phase B

Note: -D product (Differential Terminal) is custom made, please contact with salesmans before purchase.

TYPICAL APPLICATION

UIM242 controllers can work standalone or within a CAN network. Working standalone means only one UIM242 controller is linked to the CAN based host (such as UIM2501). When working in a CAN network, up to 100 UIM242 controllers can be linked together.

Under both scenarios, sensor input S1/S2/S3 should be connected to terminal 6/7/8, and signal ground should be connected to terminal 5. Furthermore, please be aware:

- User is responsible for the power supply for sensors.
- Voltage on terminal 6/7/8/9 must be kept between -0.3V and 5.3V
- Signal line of TTL output port P4 should be connected to port 9,and signl ground should be connected to AG port (port 5)
- For TTL output, the max sourcing / sinking current must be kept in 0~20mA.
- Output voltage of P4 is 0~5 V (Relative to Port 5)
- If using an external encoder, channel A should be connected to S1; channel B to S2; GND to AG.

Futhermore, users must note:

- **Live line work is forbidden.** Live line work will cause ground-wire missing: the supply voltage (red port) is on, while the supply voltage ground (black port) is not on. In this case, the supply voltage flows into the CAN driver chip, then flows into other controllers in the net through CAN bus, and finally causes damage to numbers of controllers.

- **All controller and gateway must be common-grounded.** Connect the ground wire of all controllers and gateway through one wire. If there are two ground (G1 and G2)in CAN bus, once a high-power device on G1 ground is on, the voltage on G1 will be pulled up instantly (higher than dozens volt), then this high-voltage will flow into G2 through CAN bus. Normally, the voltage on CAN bus is only 2.5V, so the dozens-volt differential will cause damage to all CAN bus chip and controllers.

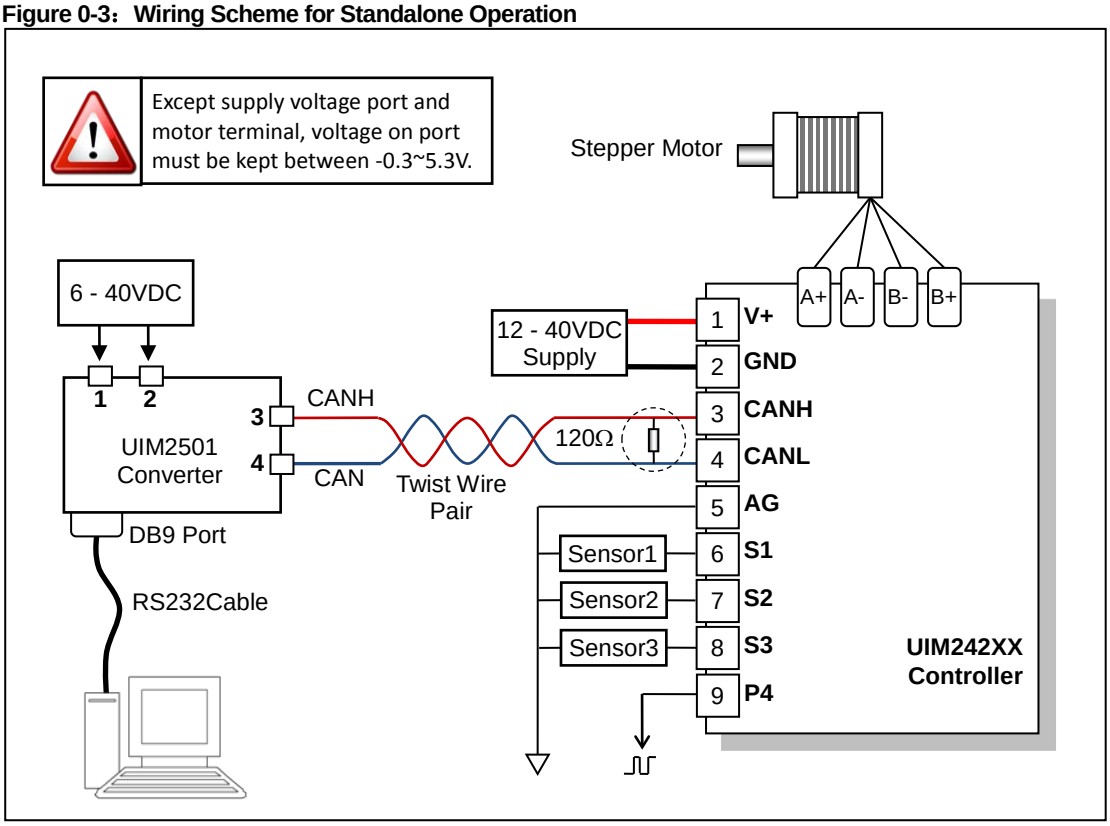
Standalone Operation

When working standalone, user can use the wiring scheme shown in figure 0-3.

Please note that, this wiring scheme should be used for setting the ID of a UIM242 controller.

For long distance transfer, both ends of the CAN bus should be terminated with120Ω terminating resistors. As UIM2501 converter has a build-in terminating resistor, user only needs to attach a resistor at the other end of the bus. Please refer to the UIM2501 user manual for how to enable the UIM2501 converter's terminating resistor. CANH and CANL should use a twisted wire pair.

Figure 0-3：Wiring Scheme for Standalone Operation



Warning: Live line work is forbidden.

Warning: All controller and gateway must be common-grounded.

Network Operation

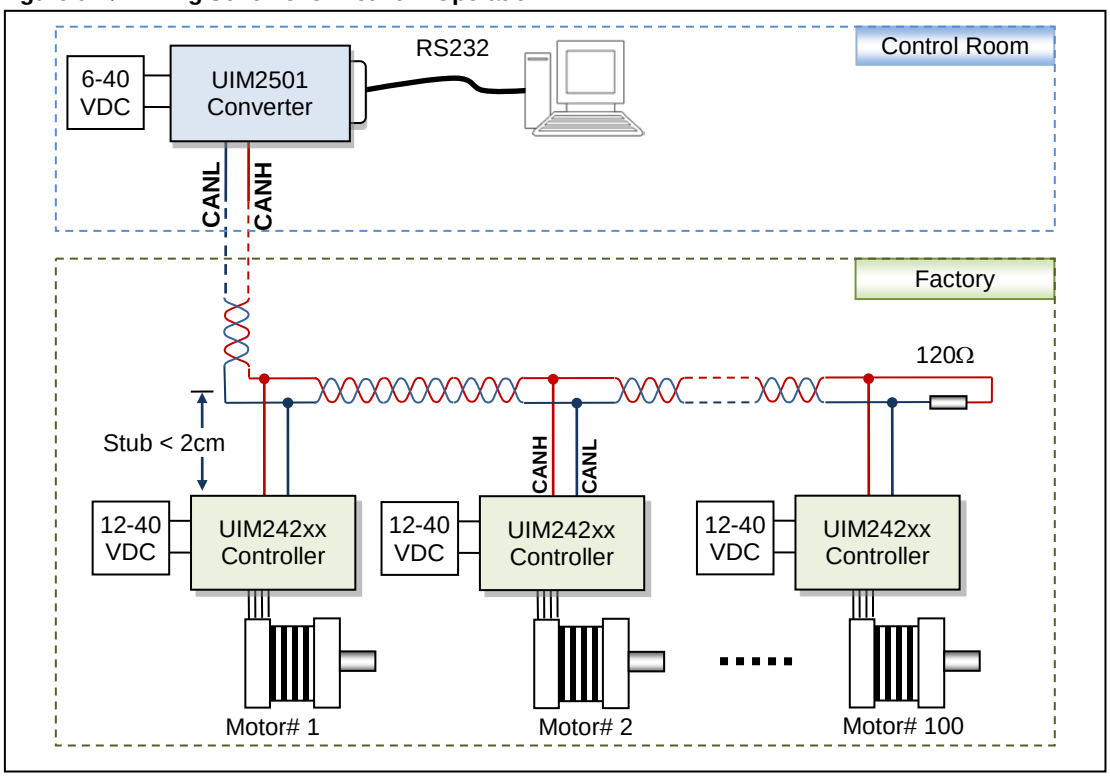
CAN bus provides a reliable and simple method of network constructing.

In figure 0-4, a wiring scheme is presented for such network operation with one RS232/CAN converter connected with multiple UIM242XX controllers. For detailed terminal wiring on each controller, please refer to figure 0-3.

Note：

- All nodes are connected onto a twist wire pair.
- Star connection scheme must be avoided.
- The stub must not exceed 2cm each (The shorter, the better).
- Both ends of the bus should be terminated with120Ω terminating resistors. Shielded 120 ohm CAN bus cable is recommended if the transfer distance is over 50 meters.
- In practice only one terminating resistor is need at the other end of CAN bus since UIM2501 already has a built-in terminating resistor. To activate this built-in terminating resistor, see UIM2501 user manual.

Figure 0-4：Wiring Scheme for Network Operation

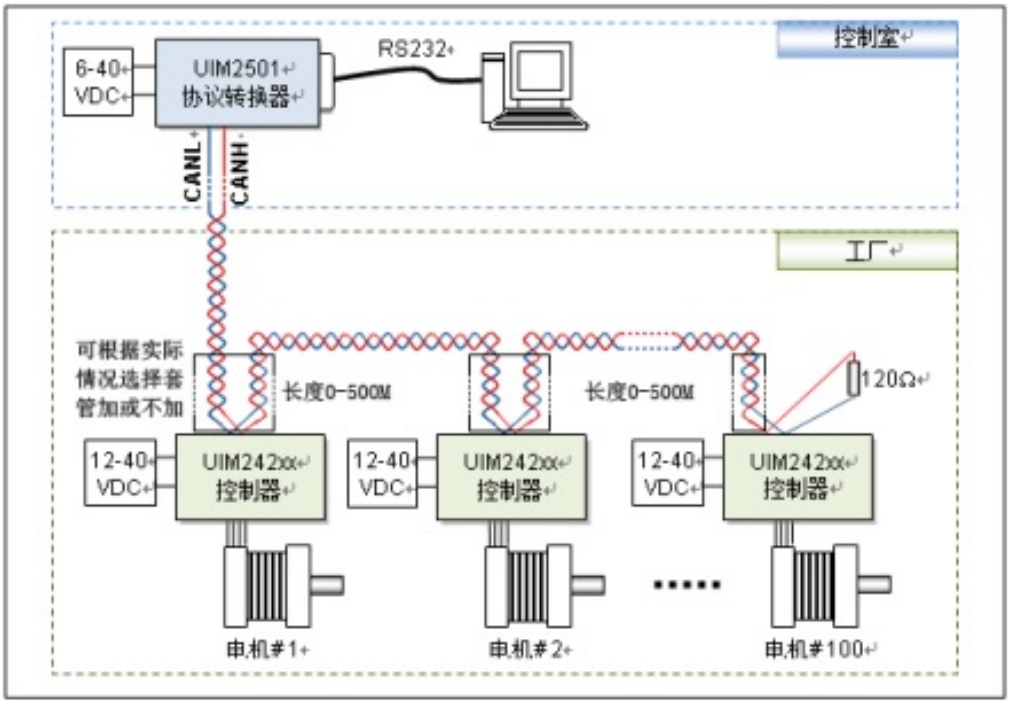


Warning: Live line work is forbidden.

Warning: All controller and gateway must be common-grounded.

There is another wiring scheme of network in Figure 0-4. When wiring in this way, the length of stub need not be shorten than 2CM, it is more flexible:

Figure 0-4：Wiring Scheme for Network Operation-2



Warning: Live line work is forbidden.

Warning: All controller and gateway must be common-grounded.

INSTRUCTION SET SUMMARY

Network Communication

Realized by gateway UIM2501/USBC9100/PCIC120, please refer to user manual of gateway for details.

Instruction	Description	Feedback Header	Message ID
BTRη;	Set CAN network communication bit rate index	AA	BC
BTR;	Check current CAN network bit rate index	AA	BC
SETη;	Assign an to UIM242 controller	AA	DD
gOFF;	Disable H-bridge circuit	AA	AD
gCURη;	Set output phase current	AA	AD
gACRη;	Enable/disable automatic current reduction	AA	AD
gMCSη	Set micro-stepping resolution	AA	AD
gORG;	Set zero/origin position	CC	AD
gSPDη;	Set the desired speed, the sign decides direction	AA	AD
gSTPη;	Set relative position, the sign decides direction	AA	AD
gPOSη;	Set desired position, the sign decides direction	AA	AD
gQECη;	Set encoder based position, the sign decides direction	AA	AD
gDOUTη;	Set output TTL level	AA	AD

Model Check

Instruction	Description	Feedback Header	Message ID	Page
MDL;	Check the model of controller	CC	DE	80

Function Configuration

Instruction	Description	Feedback Header	Message ID	Page
ENAη;	Set enable time, boot time after η ms enable	AA	A0	68
ENAxFFFF;	Check enable time	AA	A0	69
ICFη;	Set initial configuration register	AA	DA	71
ICF;	Check initial configuration register	AA	DA	72
MCFη;	Set master configuration register	AA	B0	75
MCF;	Check master configuration register	AA	B0	76
SCFη;	Set sensor control configuration register η	AA	C0	90
SCF;	Check sensor control configuration register	AA	C0	92

General Check

Instruction	Description	Feedback Header	Message ID	Page
;	Check desired motor status	AA	-	59
FBK;	Check current motor status	CC	-	70
SFB;	Check sensor status	CC	C1	93

Motor Configuration

Instruction	Description	Feedback Header	Message ID	Page
ACRη;	Set auto-current reduction ratio η	AA	-	60
ACR;	Check auto-current reduction ratio	AA	BA	61
CURη;	Set output phase current η	AA	-	64
ENA;	Enable H-bridge circuit	AA	-	67
MCSη;	Set micro-stepping resolution	AA	-	77
OFF;	Disable H-bridge circuit	AA	-	85

Motion Control

Instruction	Description	Feedback Header	Message ID	Page
BLCη;	Set backlash compensation value η	AA	DE	62
BLC;	Check backlash compensation value	AA	DE	63
MACη;	Set acceleration rate η	AA	B1	73
MAC;	Check acceleration rate	AA	B1	74
MDEη;	Set deceleration rate η	AA	B2	78
MDE;	Check deceleration rate	AA	B2	79
MMDη	Set maximum cessation speed η	AA	B4	81
MMD;	Check maximum cessation speed	AA	B4	82
MMSη;	Set maximum starting speed η	AA	B3	83
MMS;	Check maximum starting speed	AA	B3	84
ORG;	Set zero/origin position	AA	B7	86
ORGη;	Reset the position to a given value η	AA	B7	87
POSη;	Set desired position η (open-loop control)	AA	B7	88
POS;	Check current position	CC	B0	89
SPDη;	Set the desired speed η	AA	B5	94
SPD;	Check current speed	CC	B2	95
STO;	Store motion control parameters	AA	D1	98
STOη;	Bind motion control parameters to sensor edge	AA	D1	99
STPη;	Set desired incremental displacement η	AA	B6	100
STP;	Check current incremental displacement	CC	B3	101

I/O Control

Instruction	Description	Feedback Header	Message ID	Page
DOUη;	Set output TTL levelη	AA	C1	65
DOU;	Check current output TTL level	AA	C1	66
STGη;	Set digital input sampling mode	AA	C9	96
STG;	Check digital input sampling mode	AA	C9	97

CHARACTERISTICS

Absolute Maximum Ratings

Supply voltage..... 10V to 50V*
Voltage on S1/S2/S3/P4 with respect to GND.....-0.3V to +5.3V
Maximum output current sunk by S1/S2/S3/P4.....20 mA
Maximum output current sourced by S1/S2/S3/P4.....20 mA
Ambient temperature under bias..... -20°C to +85°C
Storage temperature..... -50°C to +150°C

NOTE：Working under environment exceeding the above maximum value could result in permanent damage to controller. Working under conditions at the maximum value is not recommended as operation at maximum value for extended period may have negative effect on device reliability.

*-H product (Max. supply voltage is 50V) is custom made, please contact with salesmans before purchase.

Electrical Characteristics（Ambient Temperature 25°C）

Supply Power Voltage	12V - 50VDC*
Motor Output Current	Max 2A/4A/8A per phase (instruction adjustable)
Driving Mode	PWM constant current
Stepping Resolution	full-step, half-step, 1/4, 1/8 and 1/16 step

*-H product (Max. supply voltage is 50V) is custom made, please contact with salesmans before purchase.

Communication (Ambient Temperature 25°C）

Protocol	Active CAN 2.0
Wiring method	2-wire，CANH、CANL
CAN bus drive	• Supports 1 Mb/s operation • ISO-11898 standard physical layer requirements • Short-circuit protection • High voltage transient protection • Auto-thermal shutdown protection • Up to 100 nodes can be connected • Differential bus, high noise immunity

Environment Requirements

Cooling	Free air
Working environment	Avoid dust, oil mist and corrosive gases
Working temperature	-40 °C ～ 85°C
Humidity	<80%RH，no condensation, no frosting
Vibration	3G Max
Storage temperature	-50 °C ～ 150 °C

Size and Weight

Size	42.3mm x 42.3mm x 16.5mm
Weight	0.1 kg

CONTENTS

1.0 OVERVIEW

UIM242XX are miniature integrated stepper motor controllers with CAN2.0B Active bus communication capability.

UIM242 has a size of 42.3mm*42.3mm*16.5mm and is designed to mount onto NEMA17/23/34/42 stepper motors seamlessly. UIM24202 can provide 0.7-2A output current; UIM24204 can provide 1.5-4A output current; UIM24208 can provide 3-8A output current. Current value is adjustable within the range through instructions. Once set, the value is stored in EEPROM. UIM242XX controller also has the function of high speed current compensation to offset the effect of Back Electromotive Force (BEMF) of motor at high speed and therefore to facilitate motor's high-speed performance. UIM242XX series of controllers work with 12 ~ 40VDC power supply.

UIM242XX can perform open-loop control. The control system comprises communication system, basic motion control system, absolute position counter, and real-time event-based change notification system. There are also two optional modules to be added on customer request:*Advanced Motion Module* (linear/non-linear acceleration/deceleration, S-curve PV/PVT displacement control)*, and Sensor Input control Module.*

The embedded 64-bit calculation precision DSP controller guarantees the real-time processing of the motion control and change notifications (similar to the interrupters of CPU). Entire control process is finished within 1 millisecond.

UIM242 controller applies CAN2.0B communication protocol, which, due to its high-speed (1 million bit rate) long-distance (10km) transference and high noise immunity, is widely used in applications with serious signal interference and yet requiring high reliability, such as automobile industry, automated manufacturing and traffic control. The whole CAN bus network is based on a twisted wire pair. Similar to the network of home appliances, multiple UIM242 controllers are connected to the twisted pair in parallel just like multiple pulps connected to the two-wire power cord. CAN bus network boosts many advantages, one of them is controllers never compete for bus transference.

A UIM2501 CAN-R232 converter is used to connect UIM242 controller(s) to user device through serial port. Meanwhile, ASCII-coded instructions from user device are converted and transfers in CAN protocol in high speed to long distance reliably to control stepper motor(s)' motion parameters such as direction, speed, steps, micro-steps, current, enable and disable the H-bridge. For network operation, each controller should be set a unique ID and up to 100 UIM242 controllers can be controlled through this UIM2501 converter.

1.1 Basic Control System

UIM242 controller's basic control system comprises communication system, basic motion control system, absolute position counter, and real-time event-based change notification system.

Communication System

CAN bus protocol communication is used to realize the control to UIM242. Through one CAN-RS232 converter (the UIM2501), user device can command multiple UIM242 controllers through RS232 using ASCII coded instructions. The CAN bit rate can be changed through instruction.

Basic Motion Control

UIM242 has a build-in basic motion control system. User device can control the following basic motion parameters through instructions in real-time: direction, speed, angular

displacement, phase current, micro-stepping, and enable/disable the H-bridge, etc. Speed input range is +/-65,000 pulses/sec, and displacement input range is +/-2,000,000,000 pulses.

Absolute Position Counter

UIM242 has a hardware pulse counter. The counter can be reset either by user instruction or automatically by the configurable sensor input event. Under most conditions, through the advanced motion control, this counter can provide the absolute position of the motor with enough accuracy. When the counter reaches zero position, there could be automatically generated message feedback to the user device, given the corresponding configuration through user instruction.

Furthermore, with the encoder-based closed-loop control module, the UIM242 can perform self closed-loop control.

Real-time Change Notification (RTCN)

Similar to CPU's interrupters, UIM242XX can automatically generate certain messages after predefined events and sends them to the user device. The time is less than 1 millisecond from the occurring of the event to the message being sent. Message transfer time depends on the baud rate of the RS232 setup. The transfer time will be less than 1 millisecond if the baud rate is set to 57600. UIM242XX's RTCN system supports 12 events: displacement control done absolution position reset; sensor 1/2/3 rising edge and falling edge; analog input beyond upper threshold, analog input lower than lower threshold; and TTL status, etc. All RTCNs can be enabled or disabled by instructions.

1.2 Advanced Motion Control Module

With advanced motion control module installed, UIM242XX controller can maintain linear and non-linear acceleration/deceleration, S-curve displacement control, PT/PVT control, auto direction control, etc. There are two ways to define acceleration/deceleration rate:

1.Value Mode: Input range: 1 ~ 65,000,000 PPS/Sec (pulse/sec2).

2.Period Mode: Input range: 1 ~60,000 milliseconds (time to fulfill the acceleration or deceleration).

The input range of the displacement control is +/- 2 billion pulses (steps). In advanced motion control mode, the actual direction is decided by module calculation. When displacement is in place, there will be a RTCN (Instruction configurable). Advanced motion control module can be disabled/enabled through user instruction.

1.3 Sensor Input Control Module

UIM242's Sensor Input Control Module supports 3 channels of sensor input. They can accept a TTL level input of 0~5V. There is 1 channel can be configured as analog input (Precision: 12bit; Sample frequency: 50K; mean of 16 calculation; Update frequency: 1000Hz). User can configure the desired automatic action triggered by sensor status change. There are 9 actions listed below that can be triggered by sensor event:

- Start and run forwardly at preset-speed and acceleration
- Start and run reversely at preset-speed and acceleration
- Change direction and run at preset-speed and acceleration
- Forword displacement control follow the preset motion parameters (speed, displacement, acceleration)
- Reverse displacement control follow the preset motion parameters (speed, displacement, acceleration)

- Direction-change displacement control follow the preset motion parameters (speed, displacement, acceleration)
- Decelerate at preset deceleration until stop
- Emergency stop
- Reset position and encoder counter
- Reset position and encoder counter + Reverse displacement control follow the preset motion parameters (speed, displacement, acceleration)
- Reset position and encoder counter + Decelerate at preset deceleration until stop
- Reset position and encoder counter + Emergency stop
- Off

1.4 TTL Output Control Module

UIM242's TTL Output Control Module supports 1 channel of TTL voltage level output. The output port P4 is capable of providing +/-20mA sourcing or sinking current. In practice, please keep the current consumption as low as possible to avoid overheating the controller. Port P4 also can output setting level when detects events list below (pre-configuration):

- Run/Stop status. The output voltage level is determined by if the speed is zero or not.
- Direction change. The output voltage level is determined by if the current motor direction is forward or reverse.
- Origin point hit. The output voltage level is determined by if current position is zero point or just crosses over the zero point.

1.5 Instructions and Interface

Instructions for UIM242XX are simple, intuitive and fault-tolerating.

For example, in order to command a speed of 1000 steps/sec, the following instructions are all valid: "SPD = 1000;", "SPD: 1000;", "SPD 1000;", "SPD1000;" or even "SPD %?&%* 1000;".

In case that a wrong instruction is entered, the controller will return an ACK of error message. Incorrect instructions will not be executed to prevent accidents.

UIROBOT provides free Microsoft Windows based VB / VC demo software and corresponding source code to facilitate the quick start of user device side programming.

2.0 INSTRUCTION AND FEEDBACK STRUCTURE

Once UIM242XX receives a message (instructions) from the user device, it will first ACK back (repeat) the received instruction, and then execute the instruction. UIM242XX will further send back a message to inform the user device of the completion of the instruction. Before a new instruction is received, UIM242XX will keep current working status (e.g. running, stop, etc.)

2.1 UIM242 Message Communication Mode

Host computer realizes motion control through message. Furthermore, host obtain controller status and controller update feedback information to host also through message. Therefore, user must know the structure of the message first.

Message of UIM has two forms listing below:

1. String based on RS232 (Figure 2-1), and
2. CAN message based on UI simpleCAN (Figure 2-2).

If there is no special version, all messages are based on RS232 in this manual.

For details of CAN message, please refer to UI simple CAN programming manual, or contact with technical support of UIrobot.

Host sends string message to UI gateway (UIM2501) through RS232 serial port, then the gateway converst the message into CAN message based on SimpleCAN, and sends it to specified UIM242 controller. Similarly, feedback message sent by UIM242 is based on SimpleCAN, the gateway converts it to string based on RS232, and sends it to host.

Figure 2-1: Hardware scheme for communication based on RS232 string

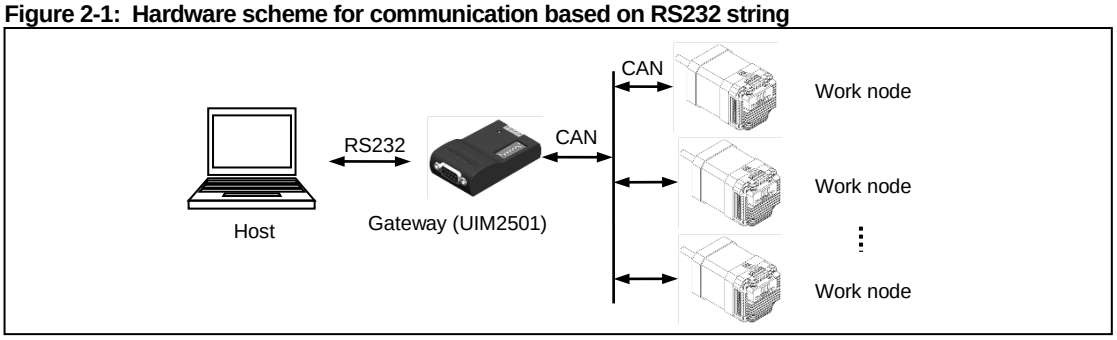
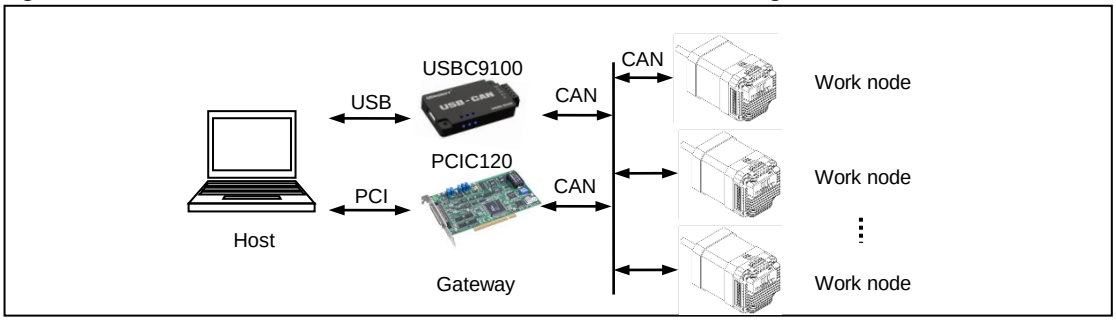


Figure2-2: Hardware scheme for communication based on CAN message



2.2 Instruction Structure

An instruction is a message sent from the user device to UIM242 to Comment certain operation. Instructions of UIM242 follow the rules listed below:

INS η; or **INSx η;** or **INS;**

Instruction symbol **INS** comprises three letters with no space between them, and is not case sensitive. If there is an x (INSx), then it means the value is hexadecimal. Value **η** comprises set of numbers. Some instructions have no value, such as “SPD;”, “STP;” etc. Each instruction must end with semicolon (;). Instruction without semicolon will cause unpredictable results.

Feedback Message is the message sent to user device from UID828 controller. The maximum length of feedback messages is 13 bytes.

Feedback messages from UIM242 (**through UIM2501**) follow the structure below:

[Header] [Controller ID] [Message ID] [Data] [Terminator]

There are 3 kinds of headers: AA、CC and EE.

Controller ID the identification number of current controller in a CAN network (also known as Node ID). Scope: 5 – 125.

Message ID denotes the property of the current message.

Data has a 7bits data structure. High is in front, and low is in the back. The 7bits data can be translated into 16bits data through the shifting operation. One 16bit data takes three 7bits data to represent.

Terminator denotes the end of a feedback message. UIM242 controller utilizes “FF” or “FE” as the terminator. If terminator is “FF”, it means there is no follow-up message; If terminator is “FE”, it means there has follow-up messages.

Note: there are two types of feedback that has NO message ID: ACK message and Motor Status feedback (controller’s response to FBK instruction). Other messages could have NO data, such as some real-time change notification messages.

2.3 Macro Operator and Null Instruction

In practice, users will combine several instructions together and send them at once. Normally, the user device will receive an ACK message on every instruction sent, these message will cause pressure on CAN bus. Especially for those basic motion instructions like SPD, DIR, MCS, which have the same ACK, sending a set of ACK is unnecessary. For example:

CUR 20; MCS 16; SPD 5000; ENA;

The above instruction set will cause 4 ACK messages being transferred on the RS232 bus.

To facilitate the above situation, user can use the following method to send a set of instructions:

{Instruction 1; Instruction 2; …Instruction N; }; (N<10)

For example:

{CUR 20; MCS 16; SPD 5000; ENA; };

UIM242XX will only send back 1 ACK on receiving the above message.

In the above example, “{” and “}” is called **Macro Operator**. Instructions between a pair of macro operators will get no ACK message.

The semicolon at the end of the instruction set has no letter or number before it. That is called **Null Instruction**. The only purpose of a Null Instruction is to tell the UIM242XX to feedback all the inquired parameters of the basic motion control. (i.e. Enable/disable, Current, Micro-stepping, Auto current reduction, Direction, Speed, and Displacement) Actually, user can simply send the null instruction";" alone to check the status of the above parameters. If there is no null instruction ";" after the "}" in the above example, there will be no ACK message at all.

3.0CAN2.0 COMMUNICATION

In order to communicate with UIM242 controller, a UIM2501 CAN-RS232 Converting Controller is required between the user device and the UIM242. The user device sends ASCII coded instructions through RS232 port to the UIM2501 converter. Inside UIM2501, the RS232 based instructions are translated into CAN messages and sent to UIM242 controllers.

With this UIM2501 converter, the user does not have to understand and deal with CAN bus operations but still enjoy the advantages of CAN bus, such as high speed, long distance, interference immunity, network, and easy wiring. UIM2501 is small in size, and is set up near the host, so the communication is quick and efficient. UIM2501 supports 57600 bps RS232 baud rate. The instruction takes less than 2ms (0.002s) to transfer from user machine to UIM242XX. At the same time, it only takes 50~100 us to transfer a message through SimpleCAN. This ensures the real-time of the system.

For detailed instructions and operations on the communication between user device and UIM2501, please refer to the UIM2501 user manual.

3.1 Controller ID Assignment

Before operation, a unique identification number (i.e., ID or address) is assigned to every UIM242 controller needs to be. ID is used to identify which object is the instruction send to, and where the ACK is from.

Every UIM242xx controller has a factory default ID of 5. User can change the ID through instruction. Before assign an ID to a UIM242XX controller, please make sure the UIM2501 controller and the UIM242XX controller are connected together using the standalone operation scheme (Figure 0-3). A motor is not necessary.

For detailed process and instructions for Controller ID assignment, please see the UIM2501 user manual.

Please Note: If there are two or more UIM242 controllers with the same ID in a network, the network may not work properly. Before assign an ID to a UIM242XX controller, please make sure the UIM2501 controller and the UIM242XX controller are connected together using the standalone operation scheme.

3.2 Instruction List

The following table shows the instructions mentioned in this chapter, the detail of those instructions is descriped at the end of the document.

Instruction	Description	Page
MDL;	Check the model of controller	80

For details about CAN2.0B bit rate setting and global instructions, please see the UIM2501 user manual.

Note: Incorrect bit rate can result in communication failure or unstable.

4.0 REAL-TIME CHANGE NOTIFICATION

UIM242 controllers support Real-time Change Notification (RTCN). Similar to interrupter of CPU, a RTCN is generated and sent when a user predefined event happens. The length of a RTCN is 4 bytes. The time from the occurrence of the event to the sending of the RTCN is less than 1 millisecond. The time is decided by baud rate. The transfer time is ahout 1ms (0.001s) when the baud rate is 57600. Then, it takes only 1.5ms from an event happening to a RTCN being received.

4.1 RTCN Structure

The structure of an RTCN message is shown below:

CC [Controller ID] [Message ID] FF

The RTCN system is able to response to the following events:

Table3-1: Real-time change notification events

No.	Event	Message ID	Description
1	falling edge of S1	A0	Voltage on S1: High >>>Low
2	rising edge of S1	A1	Voltage on S1: Low >>>High
3	falling edge of S2	A2	Voltage on S2: High >>>Low
4	rising edge of S2	A3	Voltage on S2: Low >>>High
5	falling edge of S3	A4	Voltage on S3 port: High >>>Low
6	rising edge of S3	A5	Voltage on S3 port: Low >>>High
7	TTL output P4 low	A6	Voltage on P4 port: High >>>Low
8	TTL output P4 high	A7	Voltage on P4 port: Low >>>High
9	exceed upper limits	A1/A5*	Analog input > user preset upper limit
10	below lower limit	A0/A4**	Analog input < user preset lower limit
11	displacement control complete	A8	The desired position is reached
12	zero position	A9	Position counter reaches/passes zero

Note:

***** When S1 is configured as analog, A1 denotes event 9, otherwise A1 denotes event 2.
When S3 is configured as analog, A5 denotes event 9, otherwise A5 denotes event 6.

****** When S1 is configured as analog, A0 denotes event 10, otherwise A0 denotes event 1.
When S3 is configured as analog, A4 denotes event 10, otherwise A4 denotes event 5.

4.2 Enable/Disable RTCN

Every RTCN can be enabled or disabled through user instruction. Enable/disable the RTCN is achieved by the writing to the Master Configuration Register's ORGIE bit (MCFG<5>), STPIE bit (MCFG<4>), P4IE bit (MCFG<3>), S3IE bit (MCFG<2>), S2IE bit (MCFG<1>) and S1IE bit (MCFG<0>). Please refer to section 4.1 for details.

Please note, to realize the sensor event control, user needs to further configure the sensor control registers S34CON and S12CON. Please refer to Chapter 8.0 for details.

5.0 INITIAL AND HARDWARE/FIRMWARE CONFIGURATION

UIM242's hardware and firmware can be configured through user instructions. There are 5 configuration registers for UIM242: *Initial Configuration Register, Master Configuration Register, S12CON, S34CON and Analog Threshold Register*. In this chapter, only the *Initial Configuration Register* and *Mater Configuration Register* are described. User can find details about the other registers in their corresponding chapters.

5.1 Initial Configuration Register (hardware version: 1232 or higher)

Initial configuration register is used to decide the initial status of the controllers after power-on. Once configured, its value will be burned into the on-board EEPROM, and the controller will auto reboot. Initial configuration register is a 16bits register with following structure:

ICFG

Bit	*15*	*14*	*13*	*12*	*11*	*10*	*9*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Value	X	X	X	X	X	X	X	X	X	X	X	X	Elock	PROG	CCW	ENA

Bit15-4 **Unimplemented. Read as 0.**

Bit3 **Elock, Lock when emergency events happen**

0 = After the sensor is emergency stop or power-off, the controller is unlock, and can execute instructions.
1 = After the sensor is emergency stop or power-off, the controller is lock, and receives no instruction. It needs to reboot the controller to unlock it.

Bit2 **Execute user program after power-on (Future function)**

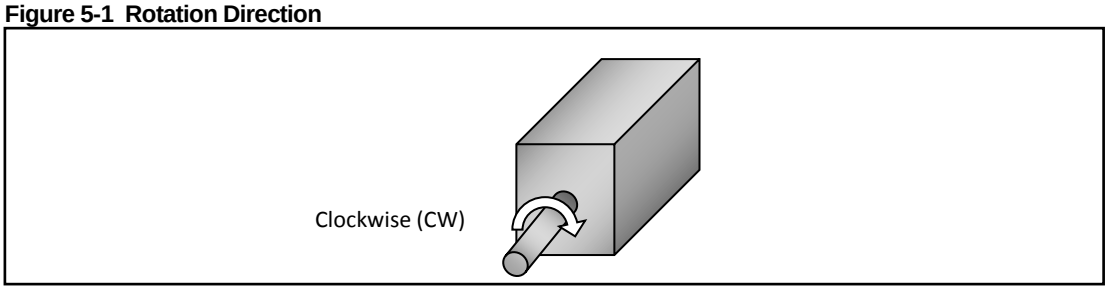
Bit1 **CCW, Adjust rotation direction (Figure 5-1)**

0 = Set CW is positive; when turn CW, displacement counter accumulate; otherwise, displacement counter decrease.
1 = Set anti-CW is positive; when turn anti-CW, displacement counter accumulate; otherwise, displacement counter decrease.

Bit0 **ENA, Auto-enable after powr-on**

0 = Disable the function (Auto-enable after power-on)
1 = Enable the function, auto-enable the controller after the pre-set time when power is on

Figure 5-1 Rotation Direction



5.2 Auto-enable

Once ICFG.ENA is set to 1, UIM242 will auto enable the H-Bridge of motor after the power is on for T ms, the interval time (T) can be set through instruction. For details of the instruction, please refer to Chapter 10.

5.3 User Program

User can program on UIM242. Once ICFG.PROG is set to 1, UIM242 will execute user program after the power is on. For details, please refer to “UIM Programming Manual”.

UIM242 still can execute user instructions when user program is running.

5.4 Master Configuration Register

Master Configuration Register is used to enable/disable the hardware/firmware functions.Once configured, it will be effective immediately and its value will be burned into the on-board EEPROM. The burning process will not affect any real-time process.Master Configuration Register is a 16bits register with the following structure:

MCFG

bit	*15*	*14*	*13*	*12*	*11*	*10*	*9*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
value	ANE	CHS	QEI	X	QEM	CM	AM	DM	X	STLIE	ORGIE	STPIE	P4IE	S3IE	S2IE	S1IE

Bit15 **ANE Enable / Disable Analog Input**
0 = Disable the analog input, all sensor are set to digital input
1 = Enable the analog input

Bit14 **CHS Analog Input Channel**
0 = Analog input on port S1
1 = Analog input on port S3

Bit13 **QEI Enable/Disable Quadrature Encoder Interface**
0 = Disable Quadrature Encoder Interface
1 = Enable Quadrature Encoder Interface

Bit12 **Unimplemented. Read as 0.**

Bit11 **QEM Enable/Disable Quadrature Encoder-based Closed-loop Control Module**
0 = Disable Quadrature Encoder-based Closed-loop Control Module
1 = Enable Quadrature Encoder-based Closed-loop Control Module

Bit10 **CM Advanced Motion Control Mode**
0 = Disable advanced motion control module, use basic control mode
1 = Enable advanced motion control module

Bit9 **AM Acceleration Mode**
0 = Value mode. Unit is pps/sec, or pulse/ (square second)
1 = Period mode. Unit is millisecond.

Bit8 **DM Deceleration Mode**
0 = Value mode. Unit is pps/sec, or pulse/ (square second)
1 = Period mode. Unit is millisecond.

Bit7 **Unimplemented. Read as 0.**

Bit6 **STLIE Locked-rotor Detection Variation Notification**
0 = Disable locked-rotor detection variation notification (only for closed-loop)

1 = Enable locked-rotor detection variation notification. Once the error between pulsing counter and encoder counter is overstep, a message will be send to user device automatically.

Bit5 **ORGIE Origin (Zero) Position RTCN**
0 = Disable the Origin (zero) position RTCN.
1 = Enable the Origin (zero) position RTCN. Once the value of pulsing counter or encoder counter is zero, a message will be send to user device automatically.

Bit4 **STPIE Displacement Control (STP/POS/QEC) Completion RTCN**
0 = Disable the displacement control completion RTCN.
1 = Enable the displacement control completion RTCN. Once the displacementinstruction has been executed, a message will be send to user device automatically.

Bit3 **P4IE P4 Status Change RTCN**
0 = Disable P4 status change RTCN
1 = Enable P4 status change RTCN

Bit2 **S3IE S3 Status Change RTCN**
0 = Disable S3 status change RTCN
1 = Enable S3 status change RTCN

Bit1 **S2IE S2 Status Change RTCN**
0 = Disable sensor port 2 (S2) status change RTCN
1 = Enable S2 status change RTCN

Bit0 **S1IE S1 Status Change RTCN**
0 = Disable sensor port 1 (S1) status change RTCN
1 = Enable S1 status change RTCN

5.5 Instruction List

The following table shows the instructions mentioned in this chapter, the detail of those instructions is descriped at the end of the document.

Instruction	Description	Page
ICFη;	Set initial configuration register	71
ICF;	Check initial configuration register	72
MCFη;	Set master configuration register	75
MCF;	Check master configuration register	76

6.0BASIC CONTROL INSTRUCTIONS

UIM242 controllers support abundant motion control instructions. The instructions of UIm242 are valid for both basic motion control (without acceleration/deceleration or S-curve displacement control) and advanced motion control (if the module is installed and enabled). User can select either basic or advanced motion control by configuring the Master Configuration Registration (MCFG).

In this Chapter, introduction to UIM242XX motion control modes is provided.

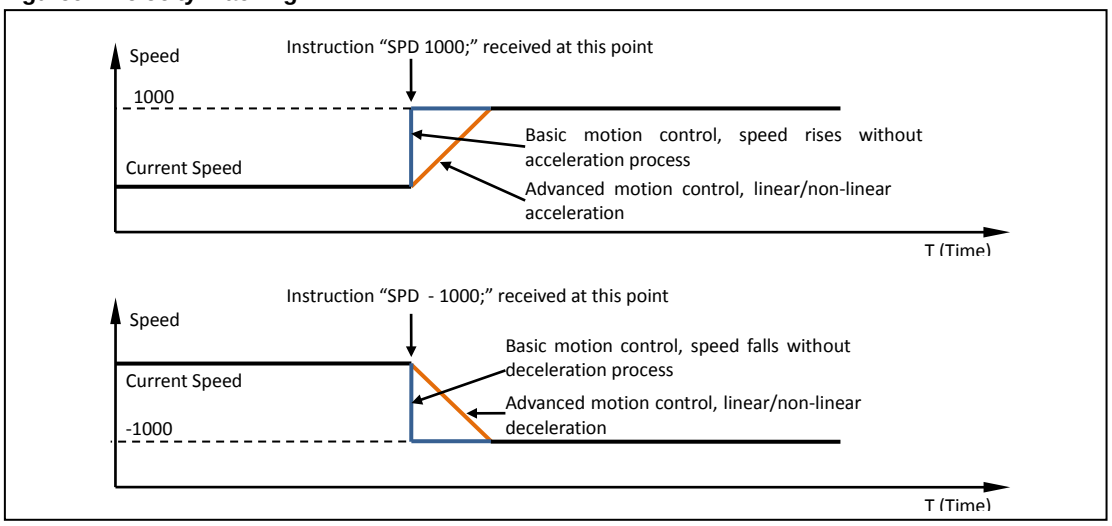
6.1 General Introduction of Motion Control Modes

There are three motion control modes for UIM242XX controller: Velocity Tracking (VT), Position Tracking (PT) and Position Velocity Tracking (PVT).

Velocity Tracking (VT)

In the Velocity Tracking (VT) mode, UIM242XX controller controls the motor speed to track desired speed.

Figure6-1 Velocity Tracking



Please note that: Sign (+/-) of the value of SPD instruction instructs the motion direction. For example: both the instruction "SPD=1000;" and "SPD=+1000;" make motor run forward at 1000pps. Meanwhile, the instruction "SPD= -1000;" can cause motor to run backward at 1000pps.

If Advanced Motion Control Module is installed, speed control can be achieved through linear or non-linear acceleration/deceleration. For details, please refer to Chapter 6.0 Advanced Motion Control. If Advanced Motion Control Module is not installed, once a SPD instruction is received, motor speed will be set to desired speed.

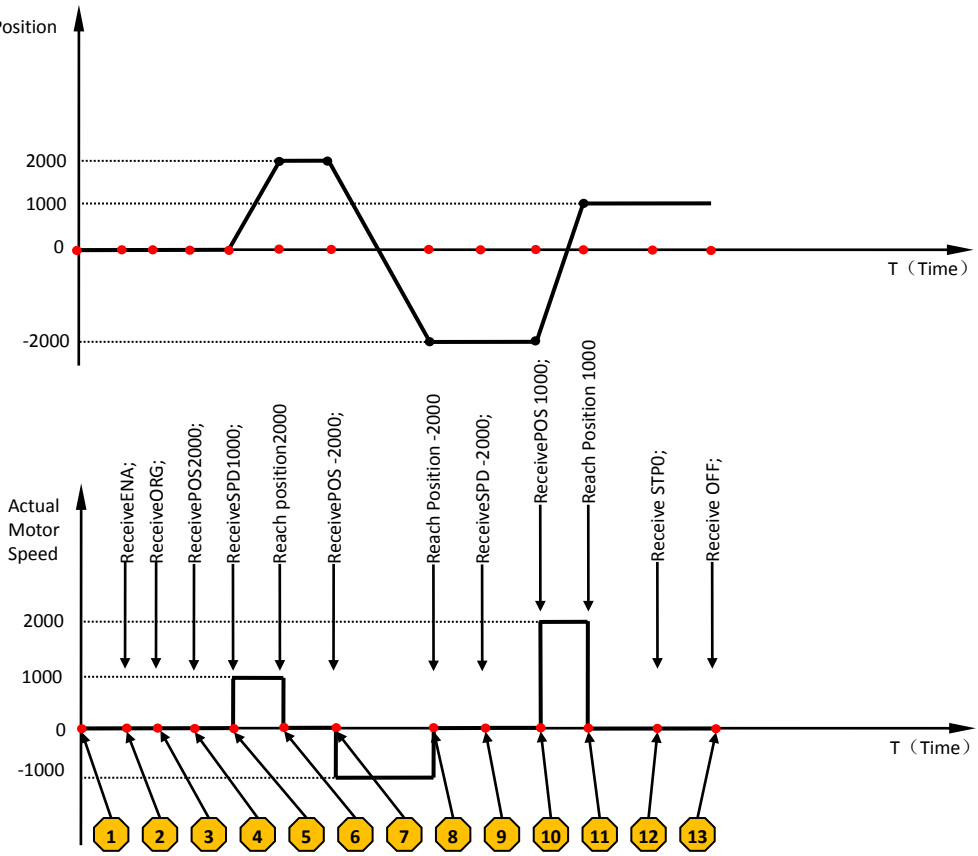
Position Tracking (PT)

In the Position Tracking (PT) mode, UIM242 controller will keep motor running at a speed close to the set value until it reaches the desired steps. After setting the desired speed, user can enter desired positions or incremental displacement continuously or discontinuously. UIM242 controller will make sure that the desired position is achieved when trying to approach the desired speed to the greatest extent.

As shown in Figure 6-2, UIM242 controller operates in PT mode automatically on receiving position instruction such as POS, STP until an instruction of “STP=0;” is given.（STP is a displacement control instruction. Logically “STP 0;” means no displacement. It is contradictory to send a displacement instruction of no displacement. Therefore, UIM242 will take this instruction as a request to shift from PT mode to VT mode.）

In PT mode, the actual speed, direction and desired displacement are related to deviation of actual displacement. When sign of desired speed and displacement deviation is different, the actual direction is decided by displacement deviation, while actual speed is set to absolute value of desired speed. Once deviation of desired and actual displacement is too small, and the acceleration is also too small, then it may cause the following situation: the motor has already reached the desired position, but it still has not reached the desired speed.

Figure6-2 Position Tracking Mode (without acceleration/deceleration)



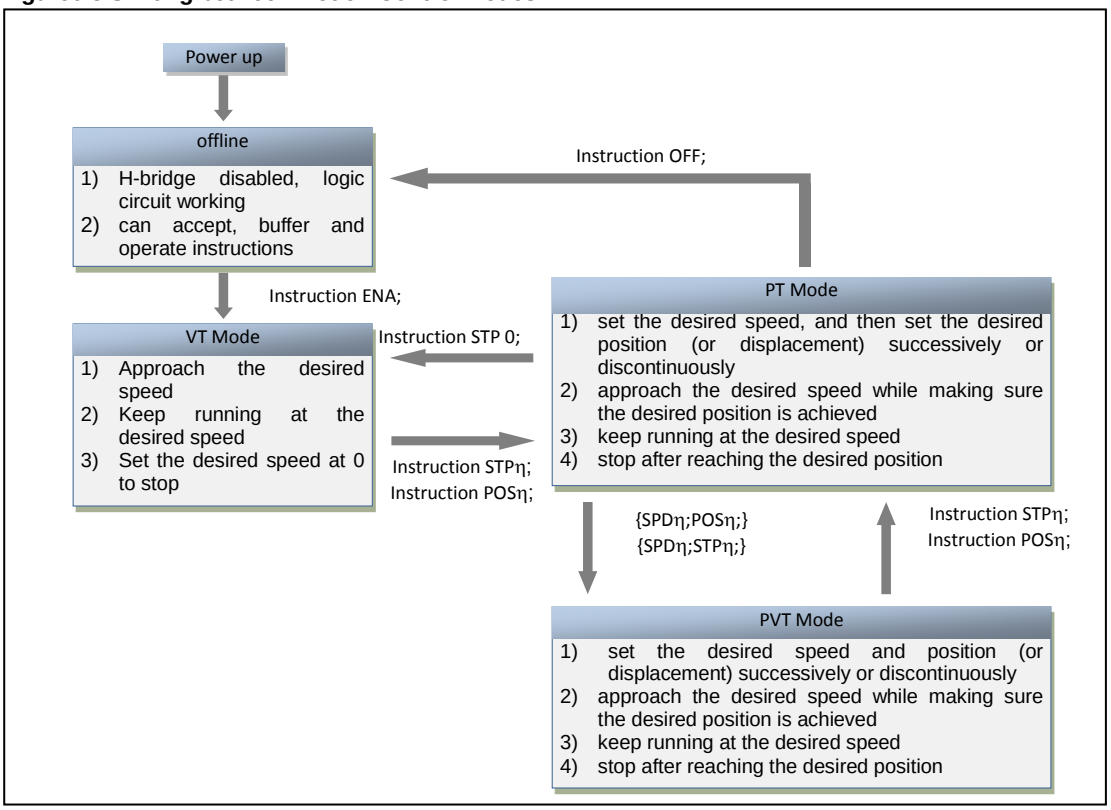
No.	Operation or Event	Control Mode	Desired Position	Current Position	Position Error	Desired Speed	Motor Direction	Motor Speed
1	Power up	VT	0	Stored position	- Stored position	0	1	0
2	ENA	VT	0	Stored position	- Stored position	0	1	0
3	ORG	VT	0	0	0	0	1	0
4	POS	PT	2000	0	2000	0	1	0
5	SPD	PT	2000	0	2000	1000	1	1000
6	Position reached	PT	2000	2000	0	1000	1	0
7	POS	PT	-2000	2000	-4000	1000	0	1000
8	Position reached	PT	-2000	-2000	0	1000	0	0
9	SPD	PT	-2000	-2000	0	-2000	0	0
10	POS	PT	1000	-2000	3000	-2000	1	2000
11	Position reached	PT	1000	1000	0	1000	1	0
12	PT mode off	VT	1000	1000	0	0	1	0
13	OFF	VT	0	1000	-1000	0	1	0

Position Velocity Tracking (PVT)

Position Velocity Tracking (PVT) mode is an extended mode of Position Tracking (PT) mode. In this mode, user can enter both desired position and desired speed.

UIM242XX controller will instruct motor to run at the desired speed until it reaches the desired position and then stop. User can enter, successively or discontinuously, both desired speed and desired position. Shifting between the three modes is displayed in the following chart:

Figure6-3 Shifting between Motion Control Modes



6.2 Basic Instruction Acknowledgment (ACK)

Upon receiving an instruction, the UIM242XX controller will immediately send back an Acknowledgment (ACK) message. There are only two ACK messages for all of them, as described below.

Error Message

If the received instruction is incorrect, UIM242 will issue an error message and the incorrect instruction will not be executed.

EE [Error Code] FF

Where, EE denotes an error message.

The error code is list below:

Error Code	65	66
Meaning	Syntax Error	Value Error

Basic ACK Message

When a valid instruction is received, the UIM242 will send back a basic ACK message. The basic ACK message contains all desired settings. Specifically, following information is included in the ACK message: STP, SPD, DIR, MCS, CUR, ENABLE/OFFLINE, and ACR. The basic ACK message is 13bytes long and has a structure as shown below:

Byte	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*
Value	AA	Controller ID	ASB	CUR	SPD2	SPD1	SPD0	STP4	STP3	STP2	STP1	STP0	FF

Where,

1. AA denotes a basic ACK message, is a kind of reply to instructions received.
2. ASM (Assembled byte) structure:

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
value	N/A(=0)	ACR	ENA / OFF	DIR	MCS – 1（0=full step，15=1/16 step)			

3. CUR (desired phase current) structure:

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
value	N/A(=0)	Phase Current (e.g. 27 = 2.7 Amp)						

4. SPD2 – SPD0 denotes the desired motor speed. See figure 10-1 for how to convert to a signed 16bit integer.
5. STP4 – STP0 denotes the desired motor displacement. See figure 10-2 for how to convert to a signed 32bit integer.

6.3 Motor Status Feedback Message

Upon receiving the FBK instruction, the controller will send back the feedback message comprising the following up-to-date motor status: incremental displacement, speed, direction, micro-stepping resolution, and phase current, enabled/offline status and ACR status.

The feedback Message is 13 bytes long in the following format:

Byte	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*
Value	CC	Controller ID	ASB	CUR	SPD2	SPD1	SPD0	STP4	STP3	STP2	STP1	STP0	FF

Where,

1. CC denotes a Motor Status Feedback Message. (i.e., the present value of motor status)
2. [ASB] (assembled) byte structure:

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
value	N/A(=0)	ACR	ENA / OFF	DIR	MCS – 1（0=full step，15=1/16 step)			

3. [CUR] (current phase current) structure

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
value	N/A(=0)	Phase Current (e.g. 27 = 2.7 Amp)						

4. SPD2 – SPD0 denotes the current motor speed. See figure 10-1 for how to convert to a signed 16bit integer.
5. STP4 – STP0 denotes the current motor displacement. See figure 10-2 for how to convert to a signed 32bit integer.

For more details on above conversion, please refer to the source code of the provided demo software. These software and related source code are VC++/VB based and free.

6.4 Instruction List

The following table shows the instructions mentioned in this chapter, the detail of those instructions is descriped at the end of the document.

Instruction	Description	Page
ACRη;	Set auto-current attenuation ratio η	60
ACR;	Check auto-current attenuation ratio	61
CURη;	Set output phase current η	64
ENA;	Enable H-bridge circuit	67
ENAη;	Set enable time, boot after η ms enable	68
ENAxFFFF;	Check enable time	69
FBK;	Check current motor status	70
MCSη;	Set micro-stepping resolution	77
OFF;	Disable H-bridge circuit	85
ORG;	Set zero/origin position	86
ORGη;	Reset the position to a given value η	87
POSη;	Set desired position η (open-loop control)	88
POS;	Check current position	89
SPDη;	Set the desired speed η	94
SPD;	Check current speed	95
STPη;	Set desired incremental displacement η	100
STP;	Check current incremental displacement	101

7.0 ADVANCED MOTION CONTROL

UIM242XX has an optional Advanced Motion Control Module (sold separately) to perform linear/non-linear acceleration/deceleration and S-curve displacement and position control. User can specify corresponding motion control parameters through instructions. Instructions for the advanced motion control includes all the basic motion instructions and 6 additional instructions.

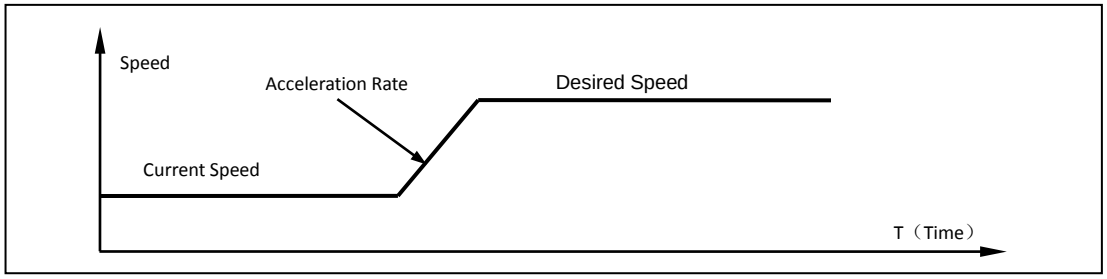
Values of these instructions will be stored in the EEPROM, the burning process will not affect any real-time process.Once the parameters are set, the controller will perform the advanced motion control automatically. At any time, user can use instructions (e.g., FBK, POS, SPD, etc.) to get the current status of the motor.

In this chapter, the Advanced Motion Control processes are introduced。

7.1 Linear Acceleration

Linear acceleration is defined as acceleration at constant rate. The relationship between the speed and time is shown in figure 7-1. After the acceleration rate and desired speed is set(MAC and SPD), UIM242 controller will perform the acceleration process automatically.

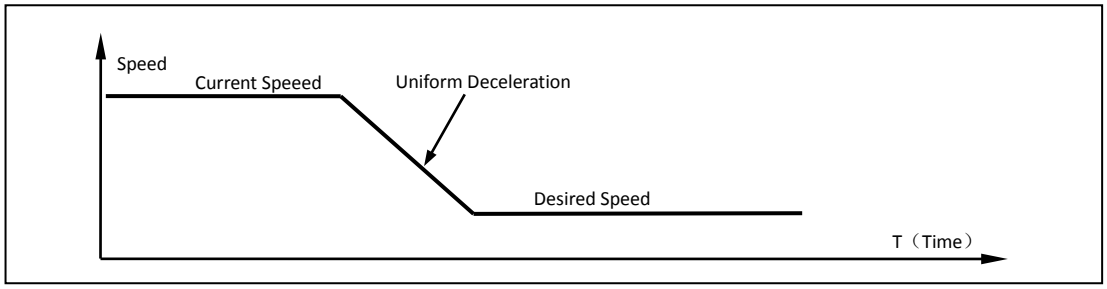
Figure7-1: Linear Acceleration Control



7.2 Linear Deceleration

Linear deceleration is defined as deceleration at constant rate. The relationship between the speed and time is shown in figure 7-2. After the deceleration rate and desired speed is set(MDE and SPD), UIM242 controller will perform the deceleration process automatically.

Figure7-2: Linear Deceleration Control



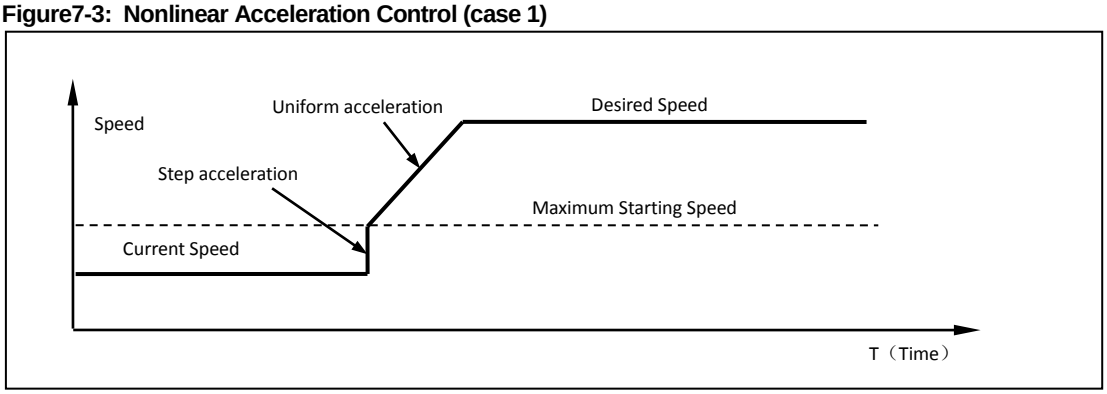
7.3 Nonlinear Acceleration

To minimize the response time and to avoid resonance point, user can use UIM242XX's non-linear acceleration function. Experiments show that through non-linear acceleration,

UIM242XX can make NEMA17/23 4000RPM (quad step) in 0.25 seconds. UIM242XX controller has the following non-linear acceleration functions.

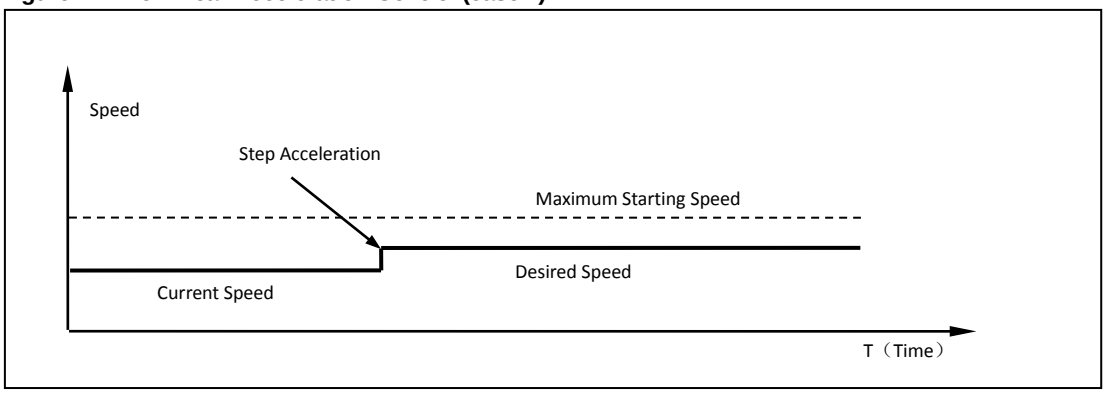
If the desired speed is higher than a certain value (i.e. the Maximum Starting Speed, defined by instruction), and current motor speed is lower than the Max. Starting Speed, then the motor speed will first step up to the Max Starting Speed and then linearly accelerated according to the acceleration rate.

Figure7-3: Nonlinear Acceleration Control (case 1)



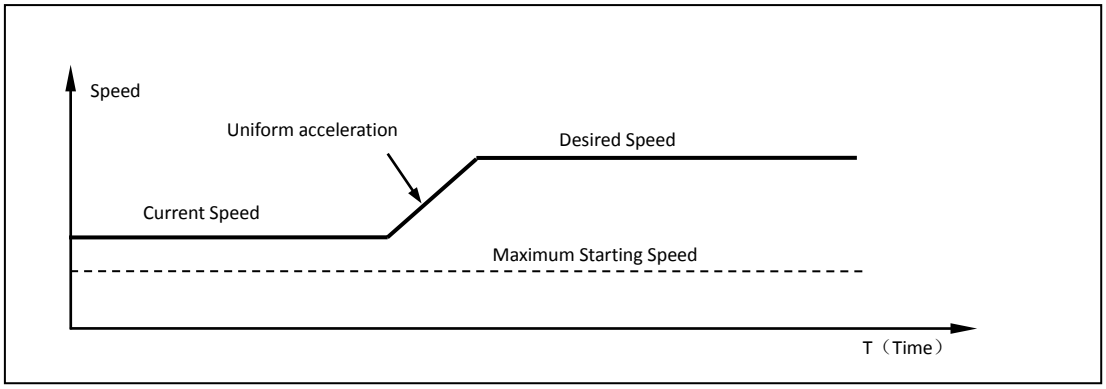
If the desired speed is less than the Max Starting Speed, then the motor speed will step up to the desired speed immediately.

Figure7-4: Nonlinear Acceleration Control (case 2)



If the current speed is higher than the Max Starting Speed, the UIM242 will use the linear Acceleration Control Algorithm to control the speed.

Figure7-5: Nonlinear Acceleration Control (case 3)

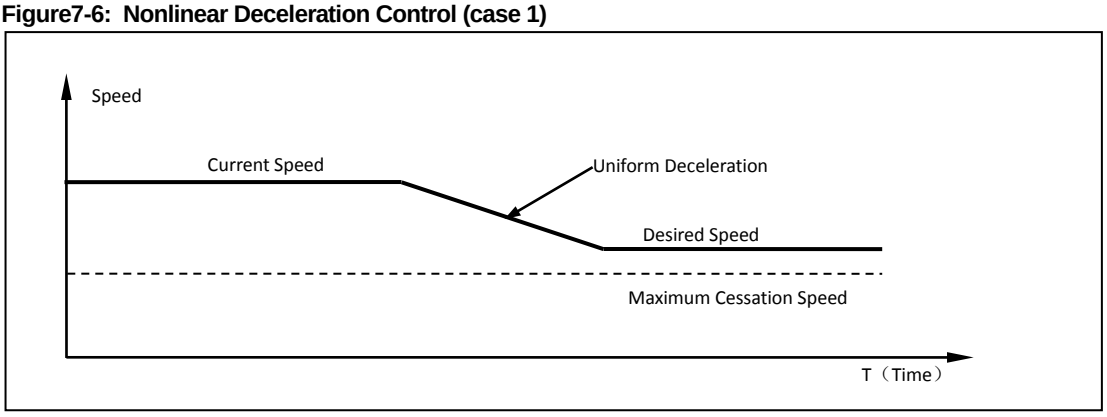


7.4 Nonlinear Deceleration

Similar to the nonlinear acceleration control, there are three cases and corresponding control algorithms as listed below.

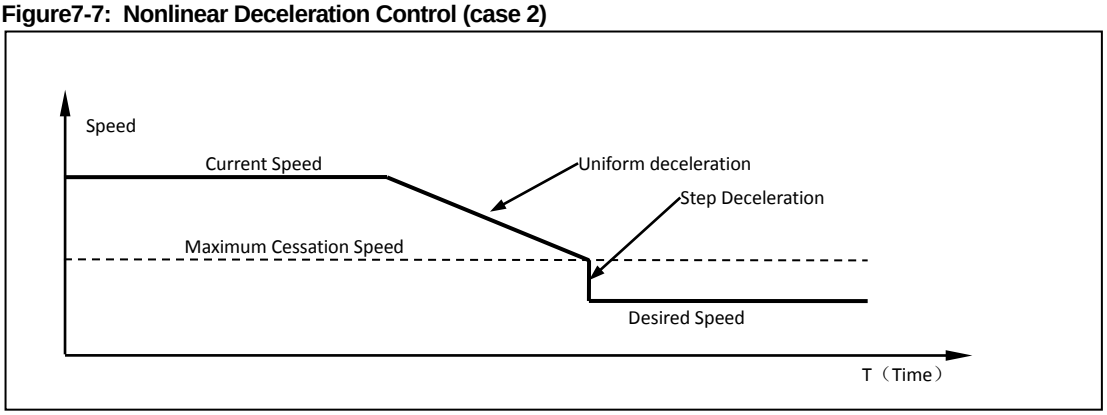
If the desired speed is higher than a certain user preset value (i.e. the Maximum Cessation Speed), UIM242XX will use the Uniform Deceleration Control algorithm.

Figure7-6: Nonlinear Deceleration Control (case 1)



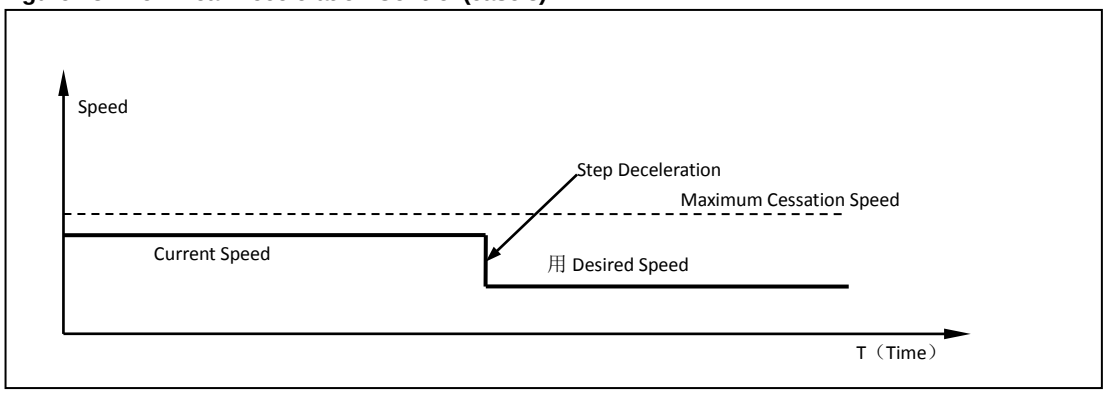
If desired speed is lower than the Max Cessation Speed and current motor speed is higher than the Max. Cessation Speed, the Uniform Deceleration Control will be first applied and followed by a step deceleration to the desired speed.

Figure7-7: Nonlinear Deceleration Control (case 2)



If the desired speed is lower than the Max Cessation Speed and current motor speed is lower than Max. Cessation Speed, then the speed will be adjusted to the desired speed through step deceleration.

Figure7-8: Nonlinear Deceleration Control (case 3)

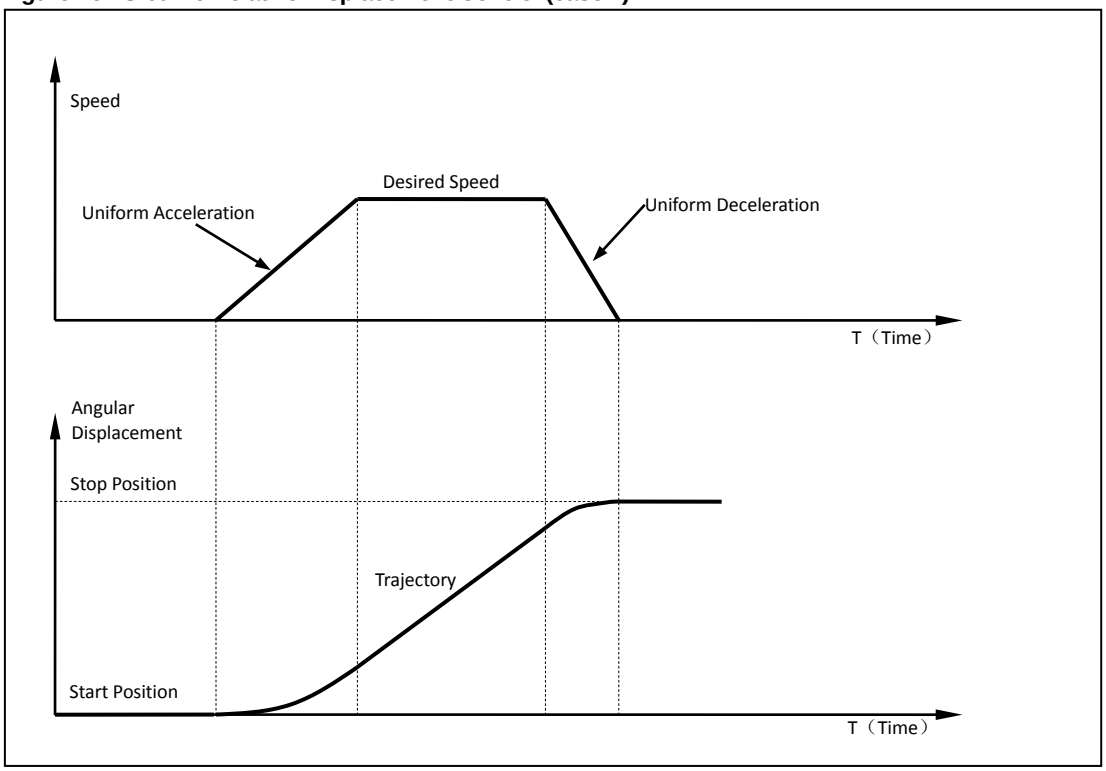


Note: Setting the Maximum Starting Speed or the Maximum Cessation Speed to 0(zero) will force the controller use Linear Acceleration / Deceleration Control Algorithm.

7.5 S-curve Displacement Control

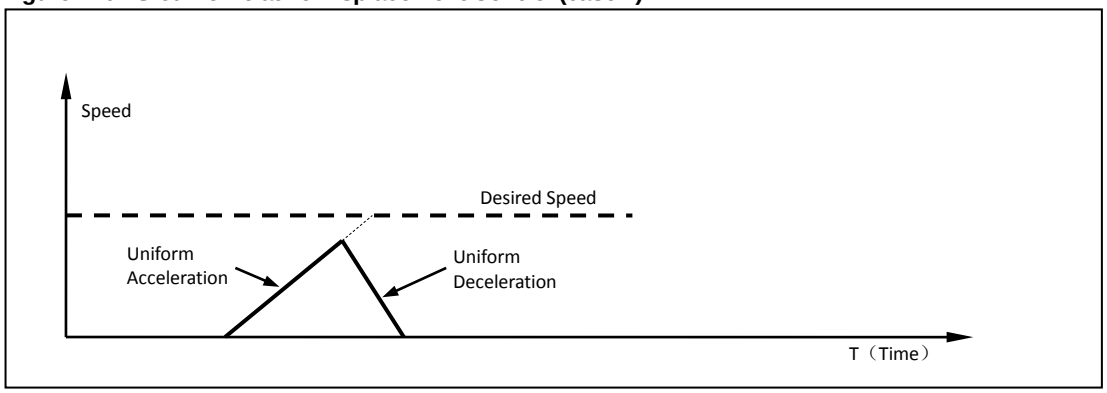
S-curve displacement control essentially is the displacement control under the linear acceleration and deceleration speed control. The name is originated from the shape of the motion trajectory. The original S-curve displacement control is the acceleration-coast-deceleration speed control. In the entire trajectory, there is no knee point, which makes the motion very smooth without impact or vibration. The control process is shown in figure 7-9.

Figure7-9: S-curve Relative Displacement Control (case 1)



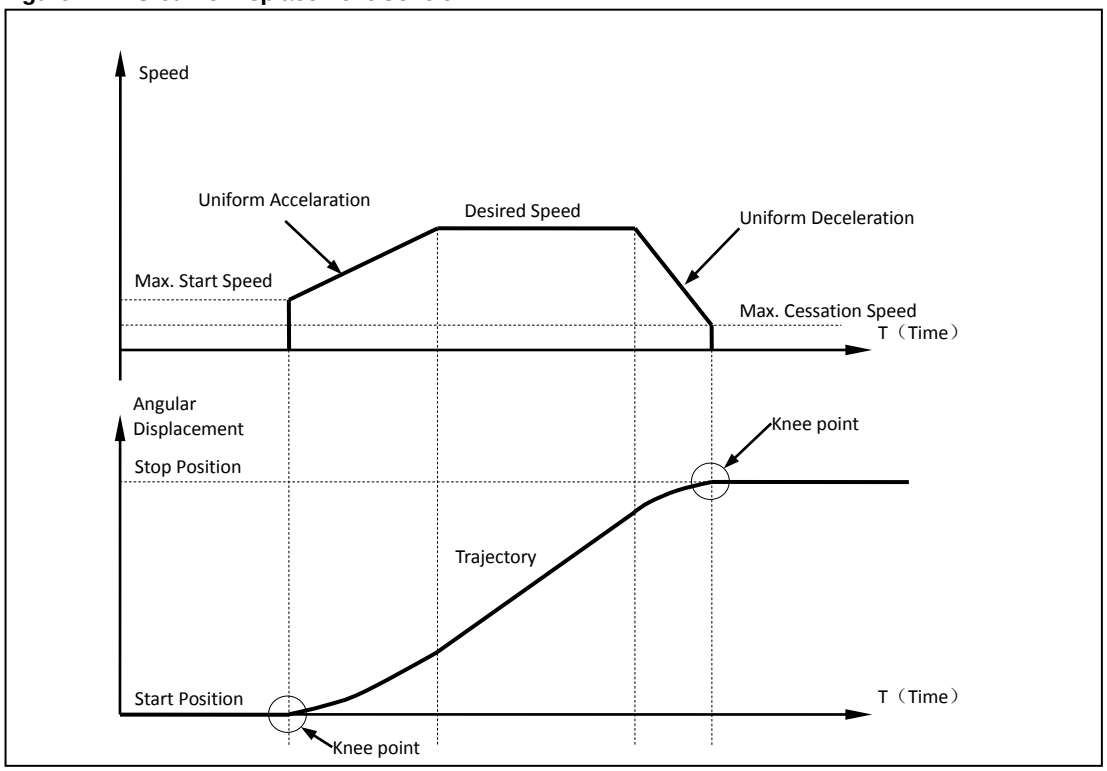
In the control process, UIM242XX's advance motion control module will continuously calculate the deceleration happening point (time) and then perform the deceleration to guarantee that when desired displacement is reached, the speed is right zero. The entire calculation time is around 20 micro-seconds with 64bit accuracy. In practice, when the desired displacement is small and the desired speed is high, deceleration starts before the desired speed is achieved to ensure that the speed decelerate to right zero when desired displacement is completed. The process is shown in figure 7-10.

Figure7-10: S-curve Relative Displacement Control (case 2)



All the acceleration/deceleration methods may be applied in the S-curve displacement control, including linear acceleration/deceleration and non-linear acceleration/deceleration which is not described in the above figures though. Please note that for the non-linear acceleration/deceleration, as there are knee points in its trajectory, is not suitable for applications requiring motion smoothness. In this case, user can set the maximum start speed and maximum cessation speed at zero to disable non-linear acceleration/deceleration. This process is shown is figure 7-11.

Figure7-11: S-curve Displacement Control



7.6 Direction Control and Position Counter

When the user enables the advanced motion control module, the actual motor direction is controlled by the module. This is because if the user input commands a motion direction different from the current motion direction, the desired direction cannot be executed immediately.

UIM242 has two types of position counters: absolute position counter and displacement counter.

Absolute position counter is for recording the absolute position of motor. The actual angular displacement is also relative to micro stepping. The value recorded in absolute position counter will be stored automatically on Power Failure situation and can only be cleared on user instruction or preset sensor event. The counter will increase or decrease according to ICFG.CWW and the actual direction of motor. Absolute position counter value can be read through POS instruction.

Displacement counter is mainly used for displacement control. The former information is cleared when it receives a new displacement instruction. It can also be used to record the displacement since last time it was cleared.

7.7 Backlash Compensation

Backlash is a ubiquitous matter for mechanical system (e.g.: screw nut transmission or gear rack transmission). For example, there is a gap between screw and nut, once the rotation direction is change, in certain angle, though the screw is turing, the nut will not drive the table moving until the gap is eliminate, this gap is known as backlash, which is reflected in the rotation angle of screw. Quantitatively, if the screw rotates clockwise to drive the nut moving 5mm forward, then, rotates anticlockwise for the same cycles, the nut will moving backword 4.99mm, the difference between the two value is the backlash.

Because of backlash, once reverse motion starts, the accumulative error will increase until the backlash is compensate, then the accumulative error tends to be steady. The influence caused by backlash is considerable in a reciprocating motion.

UIM242 controllers provide the function of backlash compensation to reduce the influence on mechanical transmission accuracy.

To compensate backlash, user needs to set a reference backlash first, then once there is a backlash, user can compensate it by sending instruction BLC. Since this instruction compensate backlash automatically when motion direction changes, and the direction before can not get automatically, then it will be thought as no backlash exsiting at the initial moment. Therefore, user must ensure that there is no backlash before sending instruction BLC.

The units of backlash compensation value is pulse, the range is 0 ~ 65536 (recommended value <5000), the default value is 0.

7.8 Advanced Motion Control Instructions

There are 6 additional instructions added as listed below.

1) Enable / disable MCFG: MCF; User can clear the CM bit of Master Configuration Register (MCFG<CM>=0) to disable the module or set the CM bit (MCFG<CM>=1) to enable the module.

2) Set acceleration: MAC; There are two ways to set the acceleration rate:(Figure7-12):

   Value mode If the AM bit of the Master Configuration Register is clear to zero (MCFG<AM>=0), then the value of the instruction will be interpreted as the value of the acceleration rate. The range of the input value is 1 ~ 65,000,000 and unit is pulse/sec/sec or pulse / square-second.

   Period mode If the AM bit of Master Configuration Register is set to one (MCFG<AM>=1), then the value of the instruction will be interpreted as the period of the acceleration, or in other words, the time used for motor to accelerate to the desired speed from current speed. The range of the input value is 1 ~ 60,000 milliseconds, i.e., 0.001~ 60 seconds.

3) Set deceleration: MDE; Similar to mACC, the deceleration also has two ways to set as listed below.

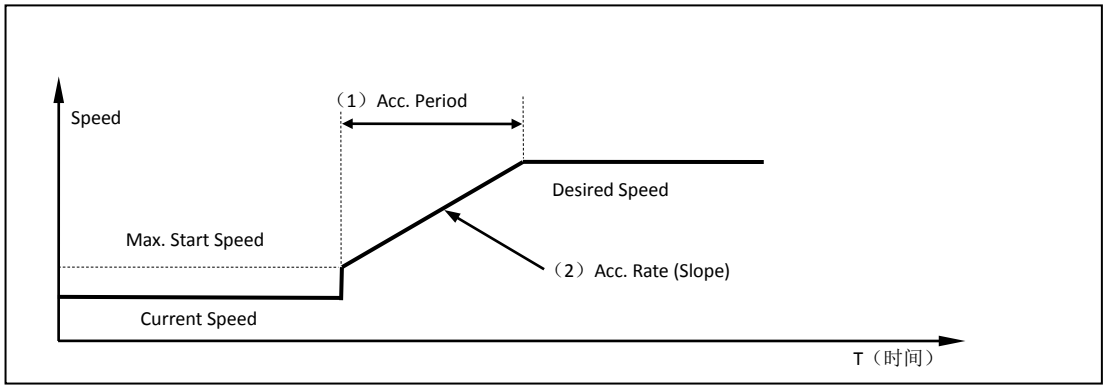
Value mode If the DM bit of the Master Configuration Register is clear to zero (MCFG<DM>=0), then the value of the instruction will be interpreted as the value of the deceleration rate. The range of the input value is 1 ~ 65,000,000 and unit is pulse/sec/sec or pulse / square-second.

Period mode If the DM bit of Master Configuration Register is set to one (MCFG<DM>=1), then the value of the instruction will be interpreted as the period of the acceleration, or in other words, the time used for motor to decelerate to the desired speed from current speed. The range of the input value is 1 ~ 60,000 milliseconds, i.e., 0.001~ 60 seconds.

4) Set maximum starting speed: MMS
5) Set maximum cessation speed MMD
6) Set backlash compensation value: BLC

Max starting speed and max cessation speed has been described in front section. The unit of MMS and MMD are pps.

Figure7-12: Two modes to Set the of Acceleration Rate



7.9 Enable/disable Advanced Motion Control Module (MCFG)

Advanced Motion Control Module can be enabled or disabled by setting the CM bit of MCFG (MCFG<10>). Setting the CM bit (MCFG<CM>=1) will enable the module and clearing the CM bit (MCFG<CM>=0) will disable the advanced motion control module. (For details of setting, please refer to Section 5.1 Master Configuration Register.) Meanwhile, the AM and DM bit of MCFG also defines the input methods of acceleration/deceleration.

7.10 Instruction List

The following table shows the instructions mentioned in this chapter, the detail of those instructions is descriped at the end of the document.

Instruction	Description	Page
BLCη;	Set backlash compensation value η	62
BLC;	Check backlash compensation value	63
MACη;	Set acceleration rate η	73
MAC;	Check acceleration rate	74
MDEη;	Set deceleration rate η	78
MDE;	Check deceleration rate	79

MMDη;	Set maximum cessation speed η	81
MMD;	Check maximum cessation speed	82
MMSη;	Set maximum starting speed η	83
MMS;	Check maximum starting speed	84

8.0SENSOR INPUT CONTROL

UIM242XX Motion Controller has an optional (sold separately) Sensor Control Module which supports three sensor input ports: S1,S2 and S3. Port S2 can be configured for digital input (0-5V). Port S1 and S3 can be configured for either digital input or analog input.

Besides digital input condition circuit, UIM242XX has a 12 bits ADC (analog/digital converter) and a 5V reference voltage. If the input voltage is 0~5V, the feedback value will be 0~4095. The ADC sample rate is 50K Hz. The analog feedback value is a mathematic average of 16 samples, and the update rate is 1000 Hz. Regardless of whether it's digital or analog, the input voltage cannot exceed -0.3V ~ 5.3V, otherwise permanent damage can be done.

Besides measuring the voltage input and providing the reads to the user device when inquired, the sensor control module is able to carry out a certain control action when a sensor event happens. Actions and sensor events can be defined by instructions. With the Sensor Control Module, UIM242 can perform motion controls without the user device.

There are 8 sensor events that can be configured, as listed below:

Table8-1: Sensor Events

No.	Sensor Events	Description
1	S1 Falling Edge	S1 Voltage Level Change, High >>>Low
2	S1 Rising Edge	S1 Voltage Level Change, Low >>>High
3	S2 Falling Edge	S2 Voltage Level Change, High >>>Low
4	S2 Rising Edge	S2 Voltage Level Change, Low >>>High
5	S3 Falling Edge	S3 Voltage Level Change, High >>>Low
6	S3 Rising Edge	S3 Voltage Level Change, Low >>>High
7	Exceeding the Upper Limit	Analog input voltage is higher than user defined upper limit
8	Exceeding the Lower Limit	Analog input voltage is lower than user defined lower limit

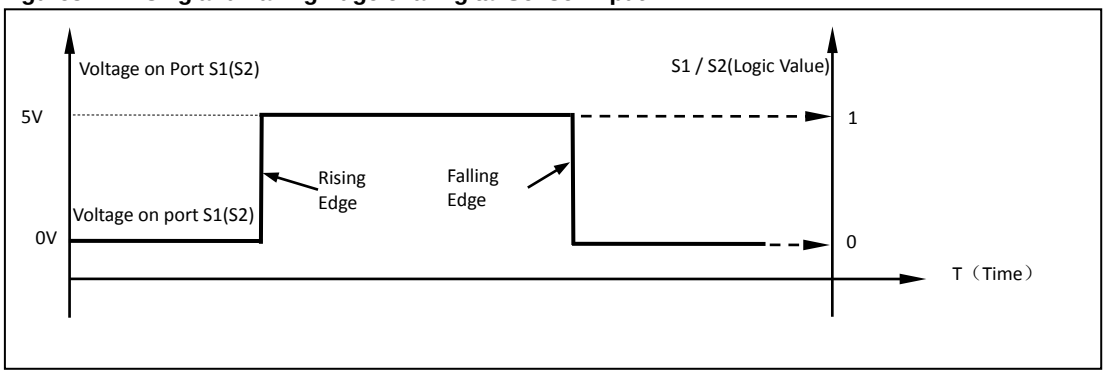
There are 13 actions that can be furthermore bound to sensor events:

- Start and run forwardly at preset-speed and acceleration
- Start and run reversely at preset-speed and acceleration
- Change direction and run at preset-speed and acceleration
- Forword displacement control follow the preset motion parameters (speed, displacement, acceleration)
- Reverse displacement control follow the preset motion parameters (speed, displacement, acceleration)
- Direction-change displacement control follow the preset motion parameters (speed, displacement, acceleration)
- Decelerate at preset deceleration until stop
- Emergency stop
- Reset position and encoder counter
- Reset position and encoder counter + Reverse displacement control follow the preset motion parameters (speed, displacement, acceleration)
- Reset position and encoder counter + Decelerate at preset deceleration until stop
- Reset position and encoder counter + Emergency stop
- Off

8.1 Rising and Falling Edge

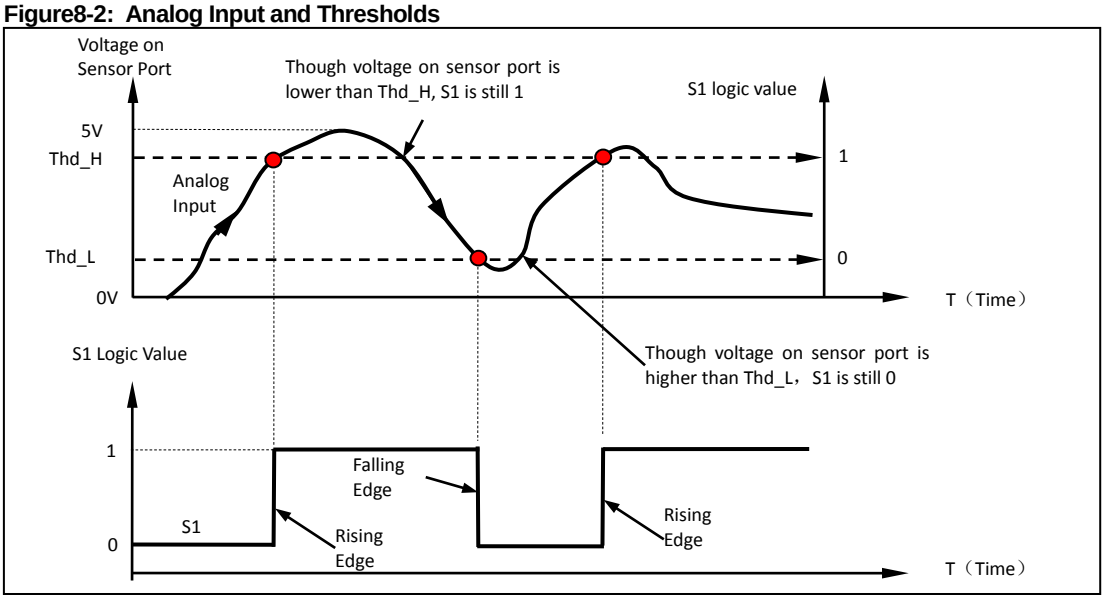
When port S1 and S2 is configured for digital input, if the sensor module detects a voltage change on S1(S2) from 0V to 5V, an Sx rising-edge event will be created, meanwhile S1(S2) is assigned a logic value 1 (i.e. S1=1). If the sensor module detects a change on S1(S2) from 5V to 0V, an S1(S2) falling-edge event will be created, meanwhile S1(S2)=0.

Figure8-1: Rising and Falling Edge of a Digital Sensor Input



8.2 Analog Input and Thresholds

Figure8-2: Analog Input and Thresholds



Sensor input port S1 can be configured for analog input by instruction. To do that, user needs to first enable the analog input function by set the ANE bit of the master configuration register (i.e., MCFG<ANE> =1). Then, user needs to select the analog input port by clear the CHS bit of the master configuration register (i.e., make MCFG<CHS> =0). Once configured, the analog voltage on port S1 can be obtained by instruction SFB.

In order to use the sensor events, user may need to further setup the input upper and lower thresholds (i.e., AH / AL in figure 8-2). If the sensor module detects the analog input voltage is changing from lower than AH to high than AH, an S1 rising-edge event will be created, meanwhile S1 is assigned a logic value 1 (i.e. S1=1). If the sensor module detects a change on S1 from higher than AL to lower than AL, an S1 falling-edge event will be created, meanwhile S1=0. Otherwise, S1 is kept unchanged.

8.3 Digital Input Sampling Mode

Digital input of UIM242 has three sampling mode:

1） Continuous sampling

2） Intermittent sampling

3） Single sampling

Continuous Sampling

In continuous sampling mode, UIM242 controllers detect level fluctuation at port S1/S2/S3 uninterruptrdly. Once a fluctuation happens, controllers will call corresponding program, execute pre-set actions, and (or) send a message to user device.

If user sets the sampling interval to 0 by using instruction STG, the controllers will work in continuous sampling mode.

Intermittent Sampling

In intermittent sampling mode, user needs to set sampling interval T (1~60000ms) at first.

Once a fluctuation is detected at one port, UIM242 controllers will not detect the level fluctuation at this port until (T+1) ms later.

When working in this mode, it is available for prevention and treatment of disturb and shake eliminating of digital input.

If user sets the sampling interval to T (1 ~ 60000) by using instruction STG, the controllers will work in intermittent sampling mode, and sampling interval is T.

Single Sampling

In single sampling mode, once a fluctuation is detected at one port, UIM242 controllers will not detect the level fluctuation at this port until user configures the corresponding control bit of S12CON (or S34CON) again.

If user sets the sampling interval to T (> 60000) by using instruction STG, the controllers will work in single sampling mode.

8.4 Sensor Event, Action and Binding

UIM242XXs support 8 sensor events as listed in section 8.0. There are 13 actions that can be bound to those 8 sensor events. Binding means assigning a sensor action to a sensor event. The binding between events and actions are realized through the configuration of the Sensor Control Register S12CON. An action-code is needed when configuring sensor registers.

- Start and run forwardly at preset-speed and acceleration (code: 10)
- Start and run reversely at preset-speed and acceleration (code: 2)
- Change direction and run at preset-speed and acceleration (code: 14)
- Forword displacement control follow the preset motion parameters (speed, displacement, acceleration) (code: 13)
- Reverse displacement control follow the preset motion parameters (speed, displacement, acceleration) (code: 5)
- Direction-change displacement control follow the preset motion parameters (speed, displacement, acceleration) (code: 9)
- Decelerate at preset deceleration until stop (code: 3)
- Emergency stop (code: 4)
- Reset position and encoder counter (code: 6)

- Reset position and encoder counter + Reverse displacement control follow the preset motion parameters (speed, displacement, acceleration) (code: 7)
- Reset position and encoder counter + Decelerate at preset deceleration until stop (code: 11)
- Reset position and encoder counter + Emergency stop (code: 12)
- Off (code: 15)

8.5 Introduction to Sensor Input Control Instructions

There are only 5 instructions related to the sensor input control.

1. MCF (Master Configuration Register)

   The ANE bit (MCFG<15>) and CHS bit (MCFG<14>) of the master configuration register define the digital/analog input of the sensor port. The S1IE bit (MCFG<0>) and S2IE bit (MCFG<1>) enable/disable the sensor real-time change notification (RTCN). See section 5.1 for details.

2. SCF (Sensor Configuration Register)

   SCF is used to configure following sensor input control registers: S12CON, S34CON, ATCONH 和 ATCONL。

3. STG (Sensor Trigger Configuration)

   STG is used to configure sensor trigger mode, UIM242 has three trigger mode: *Single Trigger, Continouns Trigger* and *N ms Intermittent Trigger*.

4. STO (Sensor Parameter Store into EEPROM)

   STO is used for storing parameters such as S12CON, ATCONH, ATCONL, SPD, and STP into EEPROM so that Sensor Input Control Module can perform the control when user device is absent.

5. SFB (Sensor Status Feedback)

   At any time and under any scenario, using the instruction SFB can always read back the logic value of S1 and S2 as well as the analog measurement (given MCFG<ANE>=1, MCFG<CHS> =0).

8.6 Sensor Input Control Register S12CON

S12CON（Sensor 1/2 Control）defines the binding relationship between S1 and S2 sensor events and actions, as well as the activation of corresponding RTCNs. It is a 16bits register inside the controller, and can be configured using the instruction SCF. When writing to it user needs to affix a 4bits suffix-code to point to this register. For details of SCF, please refer to chapter 10.

The suffix-code for S12CON is 0000 (binary). S12CON structure is as follows:

Bit	*15*	*14*	*13*	*12*	*11*	*10*	*9*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Defination	S2RACT				S2FACT				S1RACT				S1FACT			

Bit 15-12 **S2RACT<3:0>** S2 Rising-edge Action

Bit 11-8 **S2FACT<3:0>** S2 Falling-edge Action

Bit 7-4 **S1RACT<3:0>** S1 Rising-edge Action

Bit 3-0 **S1FACT<3:0>** S1 Falling-edge Action

The binding relationship between S1 and S2 sensor events and actions is as follow:

ACT Code（binary）	Action	RTCN or Not
0000	N/A	No RTCN (Ignore MCFG<S2IE><S1IE>)
0001	N/A	Depends on MCFG<S2IE><S1IE>
0010	Start and Run Reversely	Depends on MCFG<S2IE><S1IE>
0011	Decelerate until Stop	Depends on MCFG<S2IE><S1IE>
0100	Emergency Stop	Depends on MCFG<S2IE><S1IE>
0101	Reverse Displacement Control	Depends on MCFG<S2IE><S1IE>
0110	Reset position	Depends on MCFG<S2IE><S1IE>
0111	Reset position + Dispalcement Control	Depends on MCFG<S2IE><S1IE>
1001	Direction-change displacement control	Depends on MCFG<S2IE><S1IE>
1010	Start and Run Forwardly	Depends on MCFG<S2IE><S1IE>
1011	Reset position + Decelerate until Stop	Depends on MCFG<S2IE><S1IE>
1100	Reset position + Emergency Stop	Depends on MCFG<S2IE><S1IE>
1101	Forward Displacement Control	Depends on MCFG<S2IE><S1IE>
1110	Change direction and run	Depends on MCFG<S2IE><S1IE>
1111	OFF	Depends on MCFG<S2IE><S1IE>

8.7 Sensor Input Control Register S34CON

S34CON（Sensor3 / Port4 Control）defines the binding relationship between S3 sensor events and actions, as well as the activation of corresponding RTCNs. It is a 16bits register inside the controller, and can be configured using the instruction SCF. When writing to it user needs to affix a 4bits suffix-code to point to this register. For details of SCF, please refer to chapter 10.

In addition, S34CON is also used to configure the TTL output port and the events that drive the output level. In this chapter, only the S3 related configuration is described.

The suffix-code for S34CON is 0001 (binary). S34CON structure is as follows:

Bit	*15*	*14*	*13*	*12*	*11*	*10*	*9*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Defination	STALL				P4LVL	P4EVENT			S3RACT				S3FACT			

Bit 15-12 **Stall Alarm.** Only for closed-loop control, please refer to UIM242 closed-loop control manual.

Bit 11-8 **P4LVLP4EVENT<2:0>** P4 TTL Output Control

Please refer to chapter 9.0 for details.

Bit 7-4 **S3RACT<3:0>** S3 Rising-edge Action

Bit 3-0 **S3FACT<3:0>** S3 Falling-edge Action

The binding relationship between S3 and S2 sensor events and actions is as follow:

ACT Code（binary）	Action	RTCN or Not
0000	N/A	No RTCN (Ignore MCFG<S2IE><S1IE>)
0001	N/A	Depends on MCFG<S2IE><S1IE>
0010	Start and Run Reversely	Depends on MCFG<S2IE><S1IE>
0011	Decelerate until Stop	Depends on MCFG<S2IE><S1IE>
0100	Emergency Stop	Depends on MCFG<S2IE><S1IE>
0101	Reverse Displacement Control	Depends on MCFG<S2IE><S1IE>
0110	Reset position	Depends on MCFG<S2IE><S1IE>
0111	Reset position + Dispalcement Control	Depends on MCFG<S2IE><S1IE>
1001	Direction-change displacement control	Depends on MCFG<S2IE><S1IE>
1010	Start and Run Forwardly	Depends on MCFG<S2IE><S1IE>
1011	Reset position + Decelerate until Stop	Depends on MCFG<S2IE><S1IE>
1100	Reset position + Emergency Stop	Depends on MCFG<S2IE><S1IE>
1101	Forward Displacement Control	Depends on MCFG<S2IE><S1IE>
1110	Change direction and run	Depends on MCFG<S2IE><S1IE>
1111	OFF	Depends on MCFG<S2IE><S1IE>

8.8 Analog Threshold Control Register ATCONH & ATCONL

ATCONH（Analog Threshold Control High）and ATCONL define the upper and lower limit of the analog threshold. Both registers are 16bits registers in the controller memory space, configured through SCF instructions. However, when configuring, user needs to affix a 4bits suffix-code to point to a specific register. The suffix-code for ATCONL is 0010 (binary), the suffix-code for ATCONH is 0011 (binary).

ATCONH structure is as follows:

Bit	*15*	*14*	*13*	*12*	*11*	*10*	*9*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Defination	Reserved				AH <11:0>											

Bit 15-12 Unimplemented, read as 0.

Bit 11-0 **AH<11:0>** Upper limit of analog threshold.

ATCONL structure is as follows:

Bit	*15*	*14*	*13*	*12*	*11*	*10*	*9*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Defination	Reserved				AL <11:0>											

位 15-12 Unimplemented, read as 0.

位 11-0 **AL<11:0>** Lower limit of analog threshold.

Note: ATCONH / ATCONL input range is 0 ~ 4095, with 0 corresponding to 0V and 4095 corresponding to 5V. (4095 is the maximum of a 12bits data).

8.9 Instruction List

The following table shows the instructions mentioned in this chapter, the detail of those instructions is descriped at the end of the document.

Instruction	Description	Page
SCFη;	Set sensor control configuration register η	90
SCF;	Check sensor control configuration register	92
SFB;	Check sensor status	93
STGη;	Set digital input sampling mode	96
STG;	Check digital input sampling mode	97
STO;	Store motion control parameters	98
STOη;	Bind motion control parameters to sensor edge	99

8.10 Example of S12CON Configuration

When configuring S12CON, user needs to first fill every bit of the S12CON according to the information provided in previous sections, and then affixes the suffix code 0000 (binary). An example is provided below.

System Description

A reciprocating mobile platform has one ON/OFF stroke limit sensor at each end. When the mobile table hit the sensor, a 0V presents.

Requirements:

1. As soon as one sensor S2 is hit, the stepper motor starts to run reversely until the table hits the other sensor S1.
2. As soon as S1 is hit, the stepper motor starts to run positively, until the table hits S2.

Realization:

1. First stop the motor by sending: **OFF;**
2. We are not interested in the rising edge, so set S2RACT<3:0> = 0000
3. It is required Start and Run Reversely on S2 failing edge, so, set S2FACT<3:0> =0010
4. Same as 2, set S1RACT<3:0> = 0000
5. It is required Start and Run Forwardly on S1 failing edge, so, set S1FACT<3:0> =1010
6. Fill the S12CON with above bits, get: S12CON = 0000 0010 0000 1010 (binary)
7. Affix the suffix-code 0000 to S12CON, get:

   SCFG = 0000 0010 0000 1010 0000 (binary)=0x20A0 (hex)=8352 (decimal)
8. Send instruction:**SCFx 20A0;** or **SCF 8352;**
9. Set up desired speed, by sending instruction: **SPD 5000;**
10. Burn parameters into EEPROM, by sending: **STO;**
11. Press any one of the limit sensors, the mobile platform will work.
12. Disconnect the user device, and restart the UIM242 controller, the system will automatically run.

13. If enable auto-feedback, once the motor touches limit switch, user device will receive a feedback message of falling-edge on port S1/S2.

8.11 Example of ATCONH, ATCONL Configuration

Similar to S12CON configuration, user needs to first fill every bit of the ATCONH (ATCONL) according to the information provided in previous sections, and then affixes the suffix code 0011 (0010). An example is provided below.

System Description

A reciprocating mobile platform has one linear potentiometer attached to the mobile table. Within the stroke range, the potentiometer outputs 0.6V ~4V.

Requirements:

1. As soon as the sensor output is less than 0.6V, the stepper motor starts to run forward until the potentiometer outputs arrives 4V.
2. As soon as the sensor output is higher than 4V, the stepper motor starts to run backward (DIR=0) until the potentiometer outputs reaches 0.6V.

Realization:

1. First stop the motor by sending:**OFF;**
2. Set MCFG<ANE>=1, MCFG<CHS> =0, MCFG<S1IE> =1, get:
   MCFG = 1000 0000 0000 0001 (binary) = 0x8001 (hex) = 32769 (decimal)
3. Send instruction: **MCF x8001;** or **MCF 32769;**
4. It is required Start and Run Forwardly on S1 falling edge (when analog input < 0.6V), therefore, S1FACT<3:0> =1010
5. It is required Start and Run Reversely on S1 rising edge (when analog input >4V), therefore, S1RACT<3:0> =0010
6. Fill the S12CON with above bits, get: S12CON = 0000 0000 0010 1010 (binary)
7. Add suffix-code 0000 (for S12CON), get:
   SCFG = 0000 0000 0010 1010 0000 (binary)= 0x2A0 (hex)= 672 (decimal)
8. Send instruction: **SCF x2A0;** or **SCF 672;**
9. Calculate the upper limit:(4V/5V)*4095 = 3276 = 0000 1100 1100 1100（binary）
10. Add suffix-code 0011 (for ATCONH), get:
    SCFG= 0000 1100 1100 1100 0011 (binary)= 0xCCC3 (hex)= 52419 (decimal)
11. Send instruction **SCF xCCC3;** or **SCF 52419;**
12. Calculate the lower limit:（0.6V/5V）*4095 = 491 (value is rounded)= 0000 0001 1110 1011 (binary)
13. Add suffix-code 0010 (for ATCONL), get:
    SCFG= 0000 0001 1110 1011 0010 (binary)= 0x1EB2 (hex)= 7858 (decimal)
14. Send instruction: **SCF x1EB2;** or **SCF 7858;**
15. Set desired speed, by sending instruction: **SPD 5000;**
16. Burn parameters into EEPROM, by sending: **STO;**
17. Send instruction: **ENA;**
18. The system starts to work continuously.
19. Disconnect the user device, and restart the UIM242 controller, the system will automatically run.

9.0 TTL OUTPUT CONTROL

UIM242 controller has an optional TTL Output Control Module (sold separately) that supports 1 channel of TTL level output. This output port (P4) is capable of providing 20mA sourcing or sinking current. In practice, please keep the current as low as possible to prevent overheating the controller. Port P4 also can output setting level when detects events list below (pre-configuration):

1. Run/Stop status. The output voltage level is determined by if the speed is zero or not.
2. Direction change. The output voltage level is determined by if the current motor direction is forward or reverse.
3. Origin point hit. The output voltage level is determined by if current position is zero point or just crosses over the zero point.

9.1 Introduction to TTL Output Control Instructions

There are 3 instructions related to the TTL output control.

1) **MCF** The P4IE bit (MCFG<3>) of the master configuration register enables/disables the P4 real-time change notification (RTCN). For details, please refer to section 5.1.
2) **SCF** is used to configure the register S34CON. S34CON is shared by Sensor 3 and TTL output. When it works as TTL output, it defines the relationship between events and outpult level.
3) **DOU** is used to directly control the TTL output voltage level as well as check current voltage level.

9.2 TTL Output Control Register S34CON

For TTL output control, the upper byte of S34CON defines the binding between a certain event and the output voltage level. S34CON is a 16-bit register inside the controller, and can be configured using the instruction SCF. When writing to it user needs to affix a 4bits suffix-code to point to this register. The suffix-code for S34CON is 0001 (binary).

In addition, S34CON is also used for sensor input control. In this chapter, only the TTL output control related configuration is described.

S34CON Structure

Bit	*15*	*14*	*13*	*12*	*11*	*10*	*9*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Defination	STALL				P4LVL	P4EVENT			S3RACT				S3FACT			

Bit 15-12 **Stall Alarm.** Only for closed-loop control, please refer to UIM242 closed-loop control manual.

Bit 11 **P4LVL** Port P4 output voltage level

0 = If the event defined by P4EVENT code happens, P4 output = 0V
1 = If the event defined by P4EVENT code happens, P4 output = 5V

Bit 10-8 **P4EVENT<2:0>** P4 Output Driving Events

P4EVENT (binary)	Action	RTCN or Not
000	No action. Output is controlled by instruction	Depends on MCFG<P4IE>
001	When SPD>0, Output = P4LVL, vice versa.	Depends on MCFG<P4IE>

010	When move forward, Output = P4LVL, vice versa.	Depends on MCFG<P4IE>
011	When POS=0, Output = P4LVL, vice versa.	Depends on MCFG<P4IE>

Bit 7-0 **S3RACT<3:0>，S3FACT<3:0>** S3 Input Control, Please refer to section 8.7 for more information.

9.3 Output Control Configuration Instruction（SCF）

Please refer to chapter 8 for detailed information.

9.4 Instruction List

The following table shows the instructions mentioned in this chapter, the detail of those instructions is descriped at the end of the document.

Instruction	Descripition	Page
DOUη;	Set output TTL level η	65
DOU;	Check current output TTL level	66

9.5 Example of TTL Output Control and S34CON Configuration

Writing to the S34CON is realized through instruction SCFG. Before writing to the S34CON, user needs to first fill every bit of the S34CON according to the information provided in previous sections, and then affixes the suffix code 0001 (binary). An example is provided below.

System Description

A reciprocating mobile platform

Requirements

1. When motor moves forward, P4 outputs 5V.
2. When motor moves backward, P4 outputs 0V.
3. Need RTCN every time P4 changes.

Realization

1. First stop the motor by sending **OFF;**
2. Set MCFG<P4IE>=1, get:

   MCFG = 0000 0000 0000 1000 (binary) = 0x8 (hex)= 8 (decimal)
3. Send instruction: **MCF 8;**
4. Set P4EVENT <2:0>=010, link to direction event
5. Set P4LVL=1, so when motor moves forward, P4 will output 5V
6. Fill the S34CON with above bits, get: S34CON = 0000 1010 0000 0000 (binary)
7. Affix the suffix-code 0001 to S34CON, get:

   SCFG =0000 1010 0000 0000 0001 (binary)=0xA001 (hex)=40961 (decimal)
8. Send instruction: **SCF xA001;** or **SCF 40961;**
9. Send instruction: **ENA；**

10. Run the motor. There are numerous ways to run the motor. The easiest way is using **SPDη;**

During the motion, once actual direction (NOT desired direction) is plus, P4 will output 5V and RTCN. Vice verse.

10.0 REGENERATION DISCHARGE

10.1 Regeneration Electric Energy

Regeneration electric energy is the electric energy generated when the UIM all-in-one motor works in generator mode.

The motor will work in generator mode when following situation:

1. Deceleration, reversal cuased by external force (or its own rotor inertance);
2. Overlarge deceleration in the controlled moderating process;
3. Motion drived by the reversed towing caused by load on vertical axis;
4. Continuous motion caused by load;
5. Motion caused by rotor intertance (unexpected off-line and shutoff of the H-bridge MOS).

Generally, the electric energy can be absorbed when charging the smoothing capacitor of UIM controller. Charge accumulation of the capacitor in a short time will cause pumping voltage, and direct voltage rise. Once the pumping voltage is higher than the voltage those control devices can withstand, the controller will be damaged.

For example, a NEMA 34 motor runs at a speed of 300rpm or higher, once it receives an instruction to stop or turn off, the controller will be damaged; When the load of a NEMA 34 motor makes it reversed towing, the pumping voltage on MOS of H-bridge will higer than 100V, then the controller will be damaged.

10.2 UIM Regeneration Discharge Module

When pumping voltange is higher than the voltage those control devices can withstand, UIM regeneration diacharge module will absorb regeneration electric energy, reduce pumping voltage, stabilize working voltage, to avoid damage on UIM controller.

11.0 INSTRUCTION

This chapter describes the detail of the instructions mentioned in this document.

Please note, in this user manual, unless otherwise specified, all messages are based on structure, form and parsing method of RS232 character string messages. For structure, form and parsing method of CAN message based on UI SimpleCAN, please refer to UI SimpleCAN programming manual.

11.1 Instruction Structure

An instruction is a message sent from the user device to motion controller to command certain operatio. Instructions of UID828 follow the rules listed below:

1. Length of an instruction (including the ending semicolon ";") should be within 20 characters
2. Coded with standard 7 bits ASCII code (1-127). Expended ASCII code is NOT accepted.
3. Instruction structure is as follow:

INS η ;

or **INSx η ;**

or **INS ;**

Where,

INS	**Instruction Symbol**	Comprises three letters with no space between them, and is not case sensitive. If there is an x (INSx), then it means the value is hexadecimal. Please note, if η is hexadecimal, then the data must have an even number of digits, such as 00, 01, 0A. A data has an odd number of digits will cause erroes, for example, 001, 10A are illegal input.
η	**Value**	Comprises set of numbers, with no other characters between them. Some instruction have no value, such as "SPD;" "STP;" etc.
;	**Terminator**	Each instruction must end with semicolon (;) Note: Instruction without terminator will cause unpredictable results.

11.2 Feedback Message Structure

Feedback Message is the message sent to user device from motion controller. The length of feedback message is not regular, maximum length is 13 bytes.

Structure of feedback message from UIM242XX (through UIM2501) is as follow:

[Header] [Controller ID] [Message ID] [Data] [Terminator]

Header

The start of a feedback message

There are 3 kinds of headers:

– AA represents the ACK message, which is a repeat of the received instruction.
– CC represents the status feedback, which is a description of current working status.
– EE represents the error message.

Controller ID

The identification number of current controller in a CAN network (also known as Node ID)

Scope: 5 – 125.

Message ID

The property of the current message

For example, CC 05 A0 FF, where A0 denotes that there is a low level on sensor 1. For details, please refer to following sections.

Data

Has a 7bits data structure. High is in front, and low is in the back.

In figure 10-1 and 10-2, examples are shown on how to convert a set of 7bits data into 16bit data and 32bit data.

Obviously, one 16bit data takes three 7bit data to represent, and one 32bit data takes five 7bit data.

Terminator

The end of a feedback message. UIM motion controller utilizes “FF” or “FE” as the terminator. If terminator is “FF”, it means there is no follow-up message; If terminator is “FE”, it means there has follow-up messages.

Note: there are two types of feedback that has NO message ID: ACK message and Motor Status feedback (controller’s response to FBK instruction). Other messages could have NO data, such as some real-time change notification messages.

Figure11-1: Conversion from three 7bits message data to a 16bits data

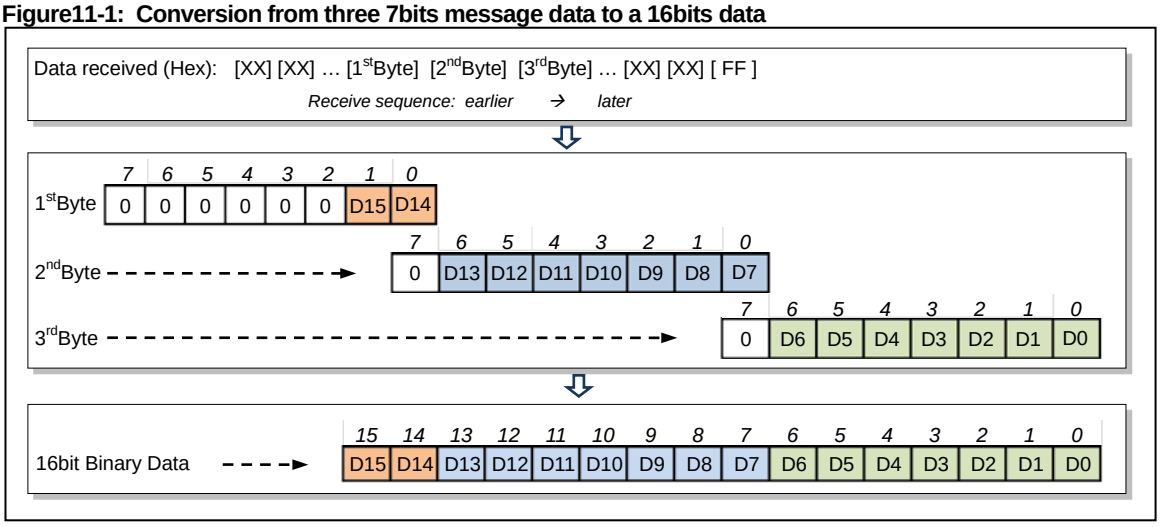
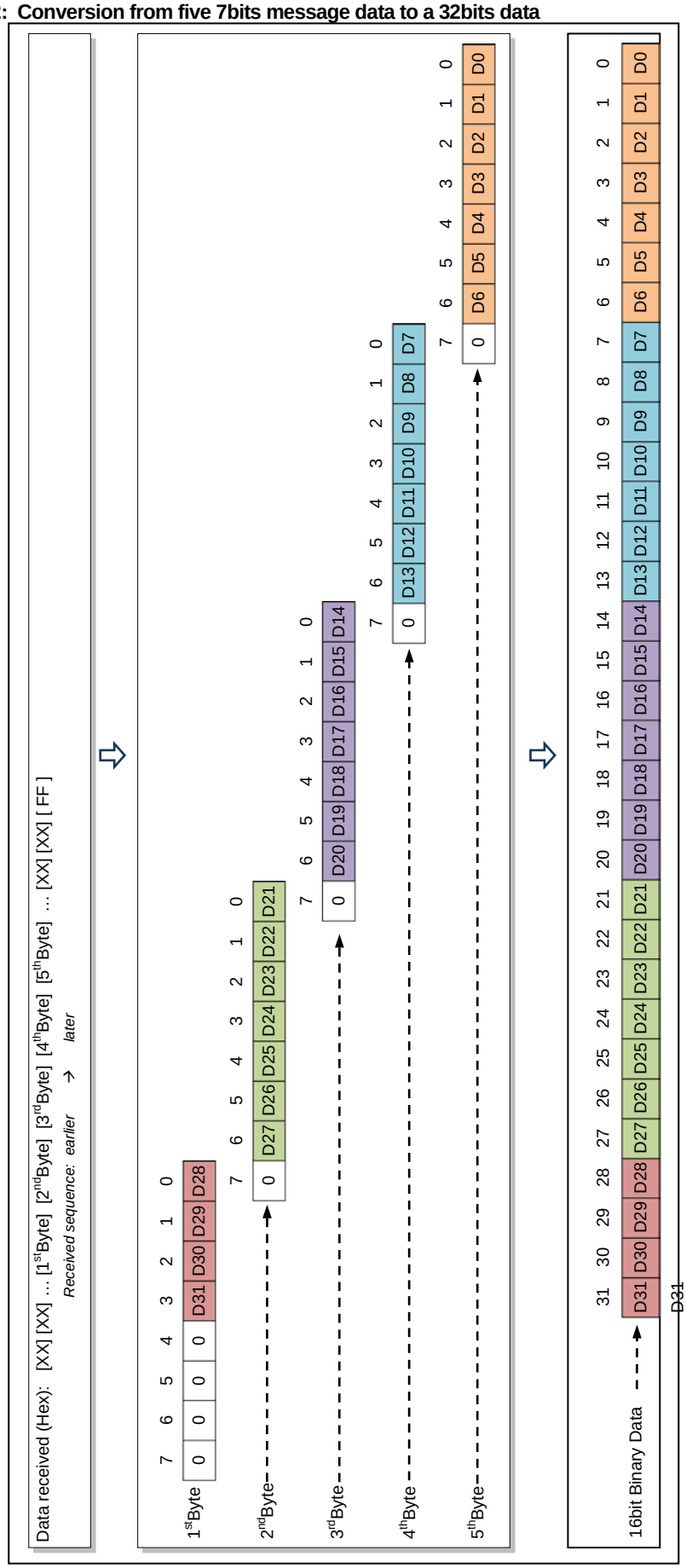


Figure11-2: Conversion from five 7bits message data to a 32bits data



11.3 Instruction Description

This section describes the detail of the instructions mentioned in this document. (in the alphabetic order)

1. ; Check desired motor status

Format: ;

Description: Check desired motor status

ACK: AA [Controller ID] [ASB] [CUR] [V0] [V1] [V2] [P0] [P1] [P2] [P3] [P4] FF

Comment: [ASB] >> Received data 0

[CUR] >> Received data 1

[V0] ~ [P4] >> Received data 2 ~ 9

[ASB] structure:

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Value	N/A(=0)	ACR	ENA / OFF	DIR	MCS – 1(0 = full step，15 = 1/16 step)			

[CUR] structure:

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Value	N/A(=0)	Phase Current (e.g. 27 = 2.7 Amp)						

[V0] ~ [V2] is the converted value for desired speed (16 bits) (Figure 11-1)

[P0] ~ [P4] is the converted value for desired displacement (32 bits) (Figure 11-2)

2. ACRη Set auto-current reduction ratio

Format: ACRη;

Description: set auto-current reduction ratio η

η = 0,1,⋯,99.

η = 0, disable auto-current reduction. Standby-CUR = working current.

η = 1, in standeby mode, current reduces to 50%. Standby-CUR = working current / 2.

η = 2,3,⋯,99, in standeby mode, current reduces to 2,3,⋯,99%. Standby-CUR = working current * η / 100.

ACK: η = 0 or η = 1, ACK is the same as ACK of “6. ENA”

η = 2,3,⋯,99, ACK is as follow:

AA [Controller ID] BA [A0] FF

Comment: BA >> Message ID of instruction ACRη;

[A0] >> Received data 0，A0 = η

Note: ACR is short for Automatic Current Reduce.

When ACR is enabled, the current will be reduced after motor stop, which means a decrease of holding torque. Value of this instruction will be stored in EEPROM.

η = 2,3,⋯,99 require controller hardware version being 1232 or higher.

3. ACR Check auto-current reduction ratio

Format: ACR;

Description: Check auto-current reduction ratio

ACK: AA [Controller ID] BA [A0] FF

Comment: Refer to ACK comment of instruction ACRη;

Note: Require controller hardware version being 1232 or higher.

4. BLCη Backlash compensation

Format: BLCη;

Description: Set value of backlash compensation in reciprocating motion

η = 0,1,⋯,65535 (Unsigned integer)

Units: pps (open-loop)

ACK: AA [Controller ID] DE [B0] [B1] [B2] FF

Comment: DE >> Message ID of instruction BLCη;

[B0] ~ [B2] >> Received data 0 ~ 2

[B0] ~ [B2] is the converted value for the value of backlash compensation (16 bits) (Figure 11-1)

5. BLC Check backlash compensation

Format: BLC;

Description: Check the value of backlash compensation in reciprocating motion

ACK: AA [Controller ID] DE [B0] [B1] [B2] FF

Comment: Refer to ACK comment of instruction BLCη;

6. CURη Motor Current Adjusting

Format: CURη;

Description: Set the output phase current to η.

η = 0,1,⋯,80 (unsigned integer)

0…80 represent 0…8.0 amps.

ACK: AA [Controller ID] [ASB] [CUR] [V0] [V1] [V2] [P0] [P1] [P2] [P3] [P4] FF

Comment: [ASB] >> Received data 0

[CUR] >> Received data 1

[V0] ~ [P4] >> Received data 2 ~ 9

[ASB] structure:

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Value	N/A(=0)	ACR	ENA / OFF	DIR	MCS – 1(0 = full step，15 = 1/16 step)			

[CUR] structure:

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Value	N/A(=0)	Phase Current (e.g. 27 = 2.7 Amp)						

[V0] ~ [V2] is the converted value for desired speed (16 bits) (Figure 11-1)

[P0] ~ [P4] is the converted value for desired displacement (32 bits) (Figure 11-2)

Note: Value of this instruction will be stored in EEPROM.

If the received current value is not one of the above integers, an Error ACK will be sent to the user device through RS232. Incorrect instructions will be discarded without being executed.

7. DOUη Set TTL Output

Format: DOUη;

Description: Set TTL output level

η = 0, 1。

ACK: AA [Controller ID] C1 [P4] FF

Comment: C1 >> The message ID of DOUη;

P4 >> The logic level of the TTL output

P4 = 1 means the output is 5V

P4 = 0 means the output is 0V

Note: Using DOUη; will affect S34CON.

Once DOUη; instruction is receiced, UIM242 controller will clear P4LVL and P4EVENT<2:0>. Therefore, if user wants to re-bind the events to the output control, user needs to reconfigure S34CON. This is to prevent potential confliction between user instruction and events controlled output.

8. DOU Check TTL Output Level

Format: DOU;

Description: Check current TTL output level

ACK: AA [Controller ID] C1 [P4] FF

COMMENT: Refer to ACK comment of DOUη;

Note: Using DOU; will NOT affect S34CON.

9. ENA H-Bridge Enable

Format: ENA;

Description: Enable the stepper motor driver (i.e. H-bridge driving circuit).

ACK: AA [Controller ID] [ASB] [CUR] [V0] [V1] [V2] [P0] [P1] [P2] [P3] [P4] FF

Comment: [ASB] >> Received data 0

[CUR] >> Received data 1

[V0] ~ [P4] >> Received data 2 ~ 9

[ASB] structure:

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Value	N/A(=0)	ACR	ENA / OFF	DIR	MCS – 1(0 = full step，15 = 1/16 step)			

[CUR] structure:

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Value	N/A(=0)	Phase Current (e.g. 27 = 2.7 Amp)						

[V0] ~ [V2] is the converted value for desired speed (16 bits) (Figure 11-1)

[P0] ~ [P4] is the converted value for desired displacement (32 bits) (Figure 11-2)

Note: Only after the H-bridge enabled, can the controller drive the motor.

10. ENAη Set enable time

Format: ENAη;

Description: Set auto-enable time register ENAtimer.

Controller auto-enable the H-bridge circuit afer power is on for η ms.

η = 1,2,⋯,60000;

ACK: AA [Controller ID] A0 [E0] [E1] [E2] FF

Comment: A0 >> Message ID of instruction ENAη;

[E0] ~ [E2] >> Received data 0 ~ 2

[E0] ~ [E2] is the converted value for auto-ENA time (16 bits) (Figure 11-1), units: ms.

Note: This instruction sets ENAtimer only, can not enable controller.

In order to enable controller after pre-set time, user needs to configure initial configuration register by using instruction ICF after ENAtimer is set.

Require controller hardware version being 1232 or higher.

11. ENAxFFFF Check enable time

Format: ENAxFFFF;

Description: Check auto-enable time

ACK: AA [Controller ID] A0 [E0] [E1] [E2] FF

Comment: Refer to ACK comment of instruction ENAη;.

Note: This instruction checks ENAtimer only, can not enable controller.
Require controller hardware version being 1232 or higher.

12. FBK Motor Status Feedback Inquiry

Format: FBK;

Description: Check the current motor status

ACK: AA [Controller ID] [ASB] [CUR] [V0] [V1] [V2] [P0] [P1] [P2] [P3] [P4] FF

Comment: [ASB] >> Received data 0

[CUR] >> Received data 1

[V0] ~ [P4] >> Received data 2 ~ 9

Structure of [ASB] is as follow:

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Defination	N/A(=0)	ACR	ENA / OFF	DIR	MCS – 1（0 = full step,15 = 1/16 step)			

Structure of [CUR] is as follow:

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Defination	N/A(=0)	Phase Current (e.g. 27 = 2.7 Amp)						

[V0] ~ [V2] is the converted value for the current motor speed. (16 bits) (Figure11-1)

[P0] ~ [P4] is the converted value for the current motor displacement. (32 bits) (Figure11-2)

Note: User can get current motor status by using this instruction at any time.

Please note: current status is different from desired status.

13. ICFxη Initial Configuration Register Instruction

Format: ICFxη;

Description: Configure the initial configuration register (InitCFG)

Parameter η has two bytes, structure is as follow:

Byte	*0*	*1*
Defination	D0	D1

Where,

[D1 D0] compose a hexadecimal 16bit data. D1 is high, D0 is low.

Example:

Initial Configuration = 0x1234;

Then send instruction: ICFx 34 12;

ACK: AA [Controller ID] DA [C0] [C1] [C2] FF

Comment: DA >> Message ID of instruction ICFxη;

[C0] ~ [C2] >> Received data 0 ~ 2

[C0] ~ [C2] is the converted value for the value of initial configuration register (16 bits) (Figure 11-1)

Note: Require controller hardware version being 1232 or higher.

14. ICF Check Initial Configuration Register

Format: ICF;

Description: Check initial configuration register

ACK: AA [Controller ID] DA [C0] [C1] [C2] FF

Comment: Refer to ACK comment of ICFxη;

Note: Require controller hardware version being 1232 or higher.

15. MACη Set Acceleration Rate

Format: MACη;

Description: Set the acceleration rate to η.

η = 1、2 … 65,000,000; (Pre-requiring MCFG<AM> = 0, value mode)

η = 1、2 … 60,000; (Pre-requiring MCFG<AM> = 1, period mode)

ACK: AA [Controller ID] B1 [FG] [A0] [A1] [A2] [A3] [A4] FF

Comment:

B1	>>	The message ID of MACη;
[FG]	>>	Equal to the AM bit of the MCFG Denote the input mode (value / period): FG =1, unit: ms; FG =0, unit: pps/s;
[A0] ~ [A4]	>>	Received data 0 ~ 4

[A0] ~ [A4] is the converted value for the value of the acceleration rate (32 bits) (Figure 11-2).

16. MAC Check Current Acceleration Rate

Format: MAC;

Description: Check current acceleration rate

ACK: AA [Controller ID] B1 [FG] [A0] [A1] [A2] [A3] [A4] FF

Comment: Refer to ACK comment of MACη;.

17. MCFη / MCFxη Master Configuration Register Instruction

Format: MCFη; or MCFxη;

Description: Setup Master Configuration Register

1） If η is decimal,

Use format: MCFη;

Where, η = 0,1,…65535 (16 bits unsigned integer)

2） If η is hexadecimal,

Use format MCFxη;

Where, η has 2 bytes, and the structure is as follow:

Byte	*0*	*1*
Defination	D0	D1

Where,

[D1 D0] compose a hexadecimal 16bit data, D1 is high, D0 is low.

For example:

Master Configuration = 0x1234;

Then send instruction MCFx 34 12;

Each Byte must have an even number of digits or letters.

ACK: AA [Controller ID] B0 [C0] [C1] [C2] FF

Comment: B0 >> The Message ID of MCFη;

[C0] ~ [C2] >> Received data 0 ~ 2

[C0] ~ [C2] is the converted value for the value of master configuration register. (16 bits) (Figure 11-1)

Note: If η using decimal, first convert the 16 bits binary number to a decimal based number.

Example: Instruction : MCF34611; or MCFx8733;

ACK: AA 05 B0 02 0E 33 FF

Comment: Convert 02 0E 33 to 16 bits (2Bytes) data, get:

0x8733 (34611 decimal).Here assume, Controller ID=5.

18. MCF Check Master Configuration Register

Format: MCF;

Description: Check the value of the Master Configuration Register

ACK: AA [Controller ID] B0 [C0] [C1] [C2] FF

Comment: Refer to ACK comment of MCFη;

19. MCSη Setup Micro Stepping

Format: MCSη;

Description: Set micro-stepping resolution.

η = 1,2,4,8,16 (unsigned integer);

η = 1, 2, 4, 8, 16 represents the full, half, quarter, eighth and sixteenth step resolution, respectively.

ACK: AA [Controller ID] [ASB] [CUR] [V0] [V1] [V2] [P0] [P1] [P2] [P3] [P4] FF

Comment: [ASB] >> Received data 0

[CUR] >> Received data 1

[V0] ~ [P4] >> Received data 2 ~ 9

[ASB] structure:

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Value	N/A(=0)	ACR	ENA / OFF	DIR	MCS – 1(0 = full step，15 = 1/16 step)			

[CUR] structure:

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Value	N/A(=0)	Phase Current (e.g. 27 = 2.7 Amp)						

[V0] ~ [V2] is the converted value for desired speed (16 bits) (Figure 11-1)

[P0] ~ [P4] is the converted value for desired displacement (32 bits) (Figure 11-2)

Note: Real-time update micro-stepping. MCS is short forMicrostepping.

Once received, the MCS value will be stored in the controller's EEPROM.

20. MDEη Set Deceleration Rate

Format: MDEη;

Description: Set the deceleration rate to η.

η = 1、2 … 65,000,000;(Pre-requiring MCFG<DM> = 0, value mode)

η = 1、2 … 60,000; (Pre-requiring MCFG<DM> = 1, period mode)

ACK: AA [Controller ID] B2 [FG] [D0] [D1] [D2] [D3] [D4] FF

Comment:

B2 >> The message ID of MDEη;

[FG] >> Equal to the DM bit of the MCFG 的 DM

Denote the input mode (value / period):

FG =1, unit: ms;

FG =0, unit: pps/s;

[D0] ~ [D4] >> Received data 0 ~ 4

[D0] ~ [D4] is the converted value for the value of the deceleration rate (32 bits) (Figure11-2).

21. MDE Check Current Deceleration Rate

Format: MDE;

Description: Check current deceleration rate

ACK: AA [Controller ID] B2 [FG] [D0] [D1] [D2] [D3] [D4] FF

Comment: Refer to ACK comment of MDEη;.

22. MDLη Check Controller Model

Format: MDLη;

Description: Check the Model, installed optional modules and firmware version of the UIM242 controller of ID =η.

η = 5, 6 … 125.

ACK: CC [Controller ID] DE 18 02 [CUR] [asb] [V0] [V1] [V2] FF

Note: [ ] denotes one byte, the data is hexadecimal.

Comment:

DE	>>	Message ID of instruction MDLη;
[CUR]	>>	The Max phase current. e.g., “20” means 2.0 A.
[asb]	>>	The installed optional modules and sensor ports.
[V0] ~ [V2]	>>	Received data 0 ~ 2

[V0] ~ [V2] is the converted value for the firmware version (12 bits) (Figure 11-1).

Structure of [asb] is as follow:

Bit	*7*	*6*	*5*	*4*	*3 2 1 0*
Defination	0	Int. QE	Closed-loop	Adv. Motion	No. of sensor port

For example, if bit 4 is 1, the Advanced Motion Control module is installed.

23. MMDη Set Maximum Cessation Speed

Format: MMDη;

Description: Set the maximum cessation speed at η.

η = 1、2 … 65,000,000; (unsigned integer)

ACK: AA [Controller ID] B4 [M0] [M1] [M2] FF

COMMENT: B4 >> The message ID of MMDη;

[M0] ~ [M2] >> Received data 0 ~ 2

[M0] ~ [M2] is the converted value for the value of maximum cessation speed. (16 bits) (Figure 11-1).

Unit: pps (pulse/second)

24. MMD Check current Maximum Cessation Speed

Format: MMD;

Description: Check the maximum cessation speed

ACK: AA [Controller ID] B4 [M0] [M1] [M2] FF

Comment: Refer to ACK comment of MMDη;.

25. MMSη Set Maximum Starting Speed

Format: MMSη;

Description: Set the maximum starting speed at η.

η = 1、2 … 65,000,000; (unsigned integer)

ACK: AA [Controller ID] B3 [M0] [M1] [M2] FF

Comment: B3 >> The message ID of MMSη;

[M0] ~ [M2] >> Received data 0 ~ 2

[M0] ~ [M2] is the converted value for the value of maximum starting speed. (16 bits) (Figure11-1).

Unit: pps (pluse/second).

26. MMS Check current Maximum Starting Speed

Format: MMS;

Description: Check the maximum starting speed

ACK: AA [Controller ID] B3 [M0] [M1] [M2] FF

Comment: Refer to ACK comment of MMSη;

27. OFF H- Bridge Disable

Format: OFF;

Description: Disable the stepper motor driver (i.e. H-bridge driving circuit).

ACK: AA [Controller ID] [ASB] [CUR] [V0] [V1] [V2] [P0] [P1] [P2] [P3] [P4] FF

Comment: [ASB] >> Received data 0

[CUR] >> Received data 1

[V0] ~ [P4] >> Received data 2 ~ 9

[ASB] structure:

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Value	N/A(=0)	ACR	ENA / OFF	DIR	MCS – 1(0 = full step，15 = 1/16 step)			

[CUR] structure:

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Value	N/A(=0)	Phase Current (e.g. 27 = 2.7 Amp)						

[V0] ~ [V2] is the converted value for desired speed (16 bits) (Figure 11-1)

[P0] ~ [P4] is the converted value for desired displacement (32 bits) (Figure 11-2)

Note: Turns off the dual H-bridge motor driving circuit.

Once controller is OFF, most devices of controller (including MOSFET) are turn off, the motor is free, and the logical circuit can work.

28. ORG Reset Position Counter

Format: ORG;

Description: Reset the position/encoder counter to zero (0), is equivalent to instruction ORG=0;

ACK: AA [Controller ID] B7 [P0] [P1] [P2] [P3] [P4] FF

Comment: Please refer to “31. POS;”.

29. ORGη Reset Position Counter

Format: ORGη;

Description: Reset the position/encoder counter to a given value η.
η = - 2147483647 ~ + 2147483647 (signed integer)
η = 0, reset the position/encoder counter to zero (0)
η ≠ 0, reset the position/encoder counter to a given value η

ACK: AA [Controller ID] B7 [P0] [P1] [P2] [P3] [P4] FF

Comment: Please refer to “31. POS;”.

30. POSη Position Control

Format: POSη;

Description: Set desired position (for open-loop control)

η = - 2147483647 ~ + 2147483647 (signed integer)

ACK: AA [Controller ID] B7 [P0] [P1] [P2] [P3] [P4] FF

Comment: B7 >> The message ID of desired position

[P0] ~ [P4] >> Received data 0 ~ 4

[P0] ~ [P4] is the converted value for the desired absolute position (32 bits) (Figure10-2)

Note: Position is essentially recorded from counts of the pulse counter.

The position counter records the total pulses sent to motor. When the direction is positive, the counter increases by 1; when the direction is negative, the counter decreases by 1. When ICFG.CW = 0, consider clockwise as forword direction; when ICFG.CW = 1, consider anticlockwise as forword direction.

The absolute position counter only resets (back to zero) in two situations:

User issues the instruction ORG

User pre-configured sensor ORG event takes place.

User needs pay attention to the two notes list below:

Power Failure Protection: Should a Power Failure situation happen, the value of the pulse counter will be pushed into EEPROM and restored when reboot next time. However, passive movement after power off cannot be recorded.

The actual motor position is also relative to the micro-stepping resolution.

31. POS Check Current Position

Format: POS;

Description: Check the value of current hardware absolute pulse counter, i.e. current absolute position of the motor.

ACK: CC [Controller ID] B0 [P0] [P1] [P2] [P3] [P4] FF

Comment: B0 >> The message ID of current position

[P0] ~ [P4] >> Received data 0 ~ 4

[P0] ~ [P4] is the converted value for the desired absolute position (32 bits) (Figure 11-2)

This position is relative to origin / zero position of couter.

32. SCFη / SCFxη Set Sensor Configuration

Format: SCFη; or SCFxη;

Description: Configure S12CON、S34CON、ATCONH and ATCONL.

1） When η is decimal:

Instruction type is SCFη;

Where, η = 0,1 … 1048575 (24 bits unsigned integer)

Refer to Chapter 8.

2） When η is hexadecimal:

Instruction is SCFxη;

Where η has 3 bytes, the structure is as follow:

Byte	*0*	*1*	*2*
Defination	D0	D1	IDX

Where,

[D1 D0] compose a hexadecimal 16bit data, D1 is high, D0 is low.

IDX = 0,1,2,3 denotes configuration of S12CON, S34CON, ATCONH and ATCONL separately.

Example:

Set S12CON as 0x1234;

Then send instruction SCFx 34 12 00; (00 is suffix)

Each Byte must have an even number of digits or letters.

ACK: AA [Controller ID] C0 [S0] [S1] [S2] [S3] [S4] [AL0] [AL1] [AH0] [AH1] FF

Comment:

C0 >> The message ID of SCFη;

[S0] ~ [S4] >> Received data 0 ~ 4

[AL0] ~ [AL1] >> Received data 5 ~ 6

[AH0] ~ [AH1] >> Received data 7 ~ 8

[S0] ~ [S4] is the converted value for [S34CON : S12CON] (32 bits) (Figure 11-2), where, 16 bits high denotes S34CON, 16 bits low denotes S12CON.

[AL0] [AL1] is the converted value for lower limit of analog threshold ATCONL (12 bits) (Figure 11-1)

[AH0] [AH1] is the converted value for upper limit of analog threshold ATCONH (12 bits) (Figure 11-1)

Note: The suffix-code for S12CON is 00 (hexadecimal)

The suffix-code for S34CON is 01 (hexadecimal)

The suffix-code for ATCONH is 02 (hexadecimal)

The suffix-code for ATCONL is 03 (hexadecimal)

33. SCF Check the value of Sensor Configuration

Format: SCF;

Description: Check the current value of S12CON\S34CON\ATCONH and ATCONL

ACK: AA [Controller ID] C0 [S0] [S1] [S2] [S3] [S4] [AL0] [AL1] [AH0] [AH1] FF

Comment: Refer to ACK comment of SCFη;

34. SFB Check Sensor Data

Format: SFB;

Description: Check sensor readings and status

ACK: CC [Controller ID] C1 [D1] [D2] [D3] [AN0] [AN1] FF

COMMENT:

C1 >> The message ID of SFB;

[D1] ~ [D3] >> Received data 1 ~ 3

[AN0] ~ [AN1] >> Received data 4 ~ 5

[D1] ~ [D3] represent the logic level of S1, S2 and S3 respectively (0/1).

[AN0] [AN1] is the converted value for analog input (12 bits). (Figure 11-1)

AN1 and AN0 are 0 if no analog input port is configured.

Note: This instruction can be used for sensor data inquiry at any time and under any condition.

35. SPDη Speed Adjusting

Format: SPDη;

Description: Set the desired speed to η.

η = - 65535…-1, 0, 1 … + 65535; (signed integer)

ACK: AA [Controller ID] B5 [V0] [V1] [V2] FF

Comment: B5 >> The message ID for desired speed

[V0] ~ [V2] >> Received data 0 ~ 2

[V0] ~ [V2] is the converted value for the value of desired speed. (16 bits) (Figure11-1)

Unit: Pluse/ Second,PPS or Hz.

The sign of speed decides direction. If no "+" or "-" specified before "x", it is taken as "+".

Note: Once H-bridge is enabled, motor starts running on receiving the instruction SPDη (η≠0) until another instruction "SPD0;" is given.

Example: For a 1.8° stepper motor, if the SPD =100;

User sent: SPD100;

If MCS=1, motor speed = 1.8 * 100 = 180°/sec = 30 rpm

If MCS=16, motr speed = 1.8 * 100 / 16 = 11.25°/sec = 1.875 rpm

36. SPD Check Current Speed

Format: SPD;

Description: Check current speed

ACK: CC [Controller ID] B2 [V0] [V1] [V2] FF

Comment: B2 >> The message ID of current speed

[V0] ~ [V2] >> Received data 0 ~ 2

[V0] ~ [V2] is the converted value for the value of desired speed. (16 bits) (Figure11-1)

Unit: Pluse/ Second,PPS or Hz.

The sign of speed decides direction. If no “+” or “-” specified before “x”, it is taken as “+”.

37. STGxη Set Digital Input Sampling Mode

Format: STGxη;

Description: Set sampling mode of digital input: continnous, intermittent and single sampling.

Structure of η:

Byte	*0*	*1*	*2*
Defination	D0	D1	IDX

Where,

[D1 D0] compose a hexadecimal 16bit data, D1 is high, D0 is low.

IDX = 00,01,02 (hexadecimal) denotes configurating sensor 1,2,3;

[D1 D0] = 0000,0001,0002,⋯,EA60 denotes that the sensor will not be triggered until 0,1,2,⋯,60000ms after last sampling. This can eliminate the shake of sensor signal.

[D1 D0] > EA60, denotes signle sampling, once triggered, the S12CON (For S1, S2) and S34CON (For S3) must be configurated again.

ACK: AA [Controller ID] C9 [S0] [S1] [S2] [S3] [S4] [S5] [S6] [S7] [S8] FF

Comment: C9 >> Message ID of instruction STGxη;

[S0] ~ [S8] >> Received data 0 ~ 8

[S0] ~ [S2] is the converted value for sampling mode of sensor 1 (16 bits) (Figure 11-1)

[S3] ~ [S5] is the converted value for sampling mode of sensor 2 (16 bits)

[S6] ~ [S8] is the converted value for sampling mode of sensor 3 (16 bits)

Example: Requirements:

1） Sensor 1;

2） Intermittent mode, interval is 200ms.

Then:

1） IDX = 00 (hexadecimal)

2） [D1 D0] = 200 (decimal) = 00C8 (hexadecimal),

So, D0 = C8, D1=00; (hexadecimal)

3） Then send instruction STGx C8 00 00;

38. STG Check Digital Input Sampling Mode

Format: STG;

Description: Check digital input sampling mode of S1, S2, 和 S3

ACK: AA [Controller ID] C9 [S0] [S1] [S2] [S3] [S4] [S5] [S6] [S7] [S8] FF

Comment: Refer to ACK comment of instruction STGxη;

39. STO EEPROM Store

Format: STO;

Description: Banding motion parameters to motions (Triggered by sensor edge or controllered by instruction)

Motion parameters include:

1） Acceleration

2） Deceleration

3） Max. starting speed

4） Max. cessation speed

For sensor, there also has:

5） Speed

6） Displacement

ACK: AA [Controller ID] D1 FF

Comment: D1 >> The message ID of STO;

Note: This instruction will affect real time performance.

It takes around 20 ms for the instruction to be executed. It is recommended that sending this instruction when the motor is idle, and wait 20ms before sending other instructions.

40. STOη Parameter Banding

Format: STOη;

Description: Banding motion parameters to motions (Triggered by sensor edge or controllered by instruction)

η = 0,1,⋯,7

η = 0, >> Montion controlled by instrcution

η = 1, >> Only for close-loop

η = 2, >> Motion triggered by rising edge of S1

η = 3, >> Motion triggered by falling edge of S1

η = 4, >> Motion triggered by rising edge of S2

η = 5, >> Motion triggered by falling edge of S2

η = 6, >> Motion triggered by rising edge of S3

η = 7, >> Motion triggered by falling edge of S3

ACK: AA [Controller ID] D1 FF

Comment: D1 >> Message ID of instruction STO;

Note: Require controller hardware version being 1232 or higher.

This instruction will affect real time performance. It takes around 15 ms for the instruction to be executed. It is recommended that sending this instruction when the motor is idle, and wait 20ms before sending other instructions. Before set parameters, disable the controller first.

Default setting for STO0: 300/300/0/0/0/0, it can be configurated by instruction.

Parameters for each edge can be different. Not all parameters are needed; the non-value parameter will be assigned as the value of parameters for STO0

Example: Disable the controller: OFF;

Set 1st group of parameters: SPDη; STPη; MACη; MDEη; MMSη; MMDη;

Banding it to rising edge of S1: STO2;

……

Set 6th group of parameters: SPDη; STPη; MACη; MDEη; MMSη; MMDη;

Banding it to falling edge of S3: STO7;

41. STPη Displacement Control

Format: STPη;

Description: Set the desired incremental displacement, i.e. the displacement relative to current position

η = - 2147483647 ~ + 2147483647 (signed integer)

ACK: AA [Controller ID] B6 [P0] [P1] [P2] [P3] [P4] FF

Comment: B6 >> The message ID of STPη;

[P0] ~ [P4] >> Received data 0 ~ 4

[P0] ~ [P4] is the converted value for the desired motor displacement (32 bits) (Figure10-2)

Note: Displacement is essentially defined as counts of the pulse or encoder counter.

Actual pulse sended to motor is controlled by displacement counter. The actual motor displacement is also relative to the micro-stepping resolution or encoder resolution.

If an STP0; instruction is received before the former STP instruction is completed, UIM242 will execute the current instruction and stop motor. The former STP instruction is regarded as being completed. Meanwhile, system will shift from PT mode to VT mode.

If an STP instruction is received while the motor is already running, the former steps will not be counted in the displacement of current STP instruction.

Example: For a 1.8° stepper motor, if STP = 200 pulse;

User sent: STP200;

If MCS=1, motor rotation angle = 1.8 * 200 = 360°

If MCS=16, motor rotation angle = 1.8 * 200 / 16 = 22.5°

42. STP Check Displacement

Format: STP;

Description: Check current incremental displacement.

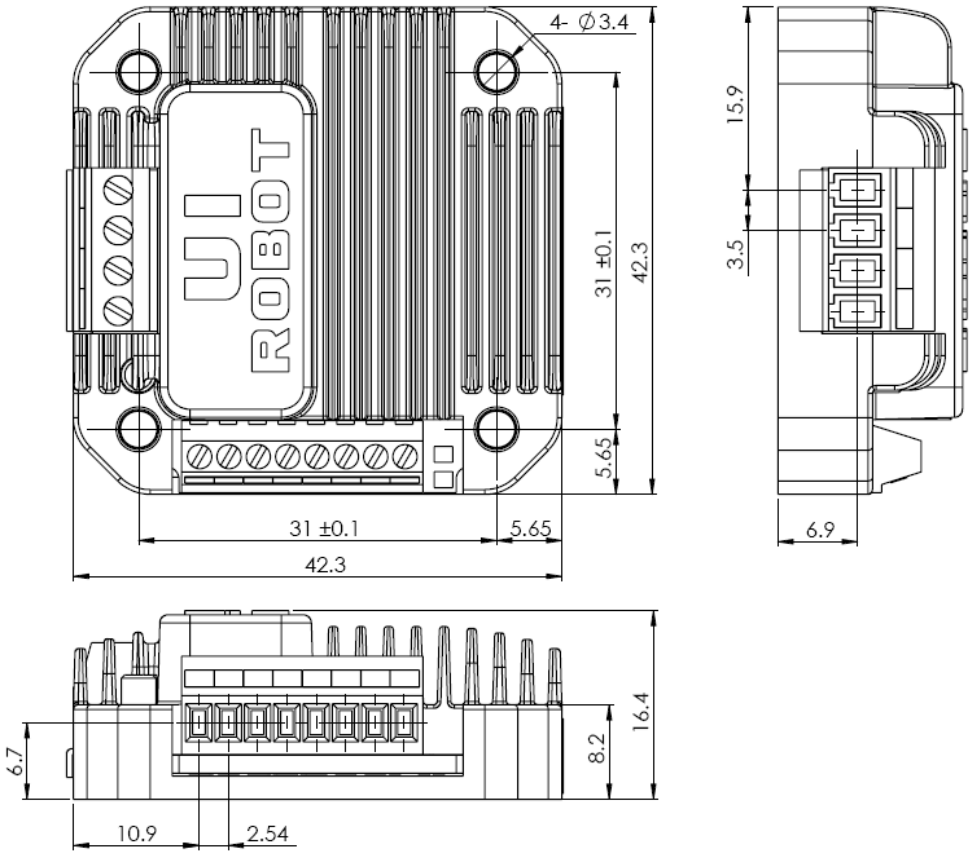
ACK: CC [Controller ID] B3 [P0] [P1] [P2] [P3] [P4] FF

Comment: B3 >> The message ID of current incremental displacement

[P0] ~ [P4] >> Received data 0 ~ 4

[P0] ~ [P4] is the converted value for the current incremental displacement (32 bits) (Figure11-2)

APPENDIX A DIMENSIONS

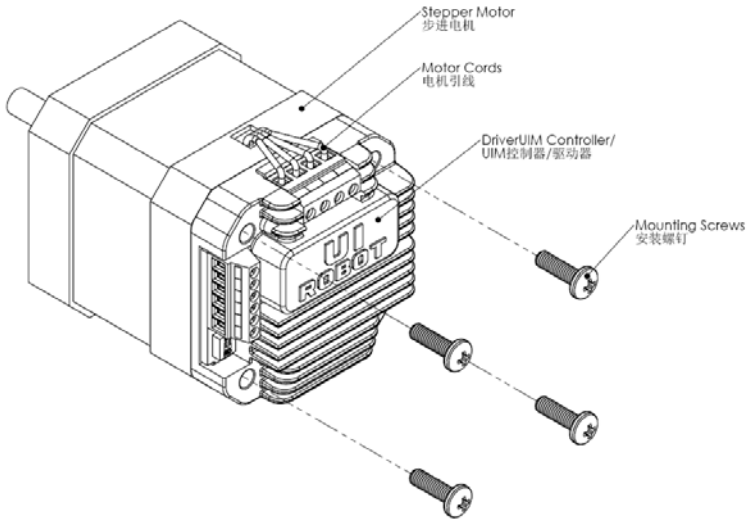


Units: mm

APPENDIX B INSTALLATION INSTRUCTION

NEMA 17 (do not need flange)

1. Fix the UIM controller on motor with screw (2 or 4)
2. Connect the motor pin to motor terminal of UIM controller



NEMA 23/34/42 (need flange)

1. Fix flange on motor
2. Fix the UIM controller on flange with screw
3. Connect the motor pin to motor terminal of UIM controller