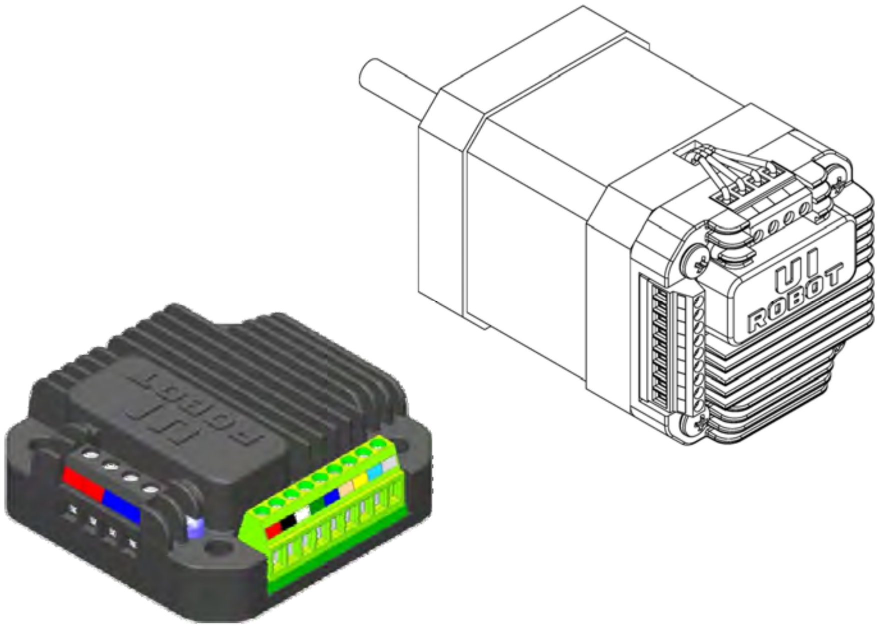
UIROBOT®
United Intelligence Robot Technology

User Manual

UIM242XX Series

CAN2.0B Instruction Control

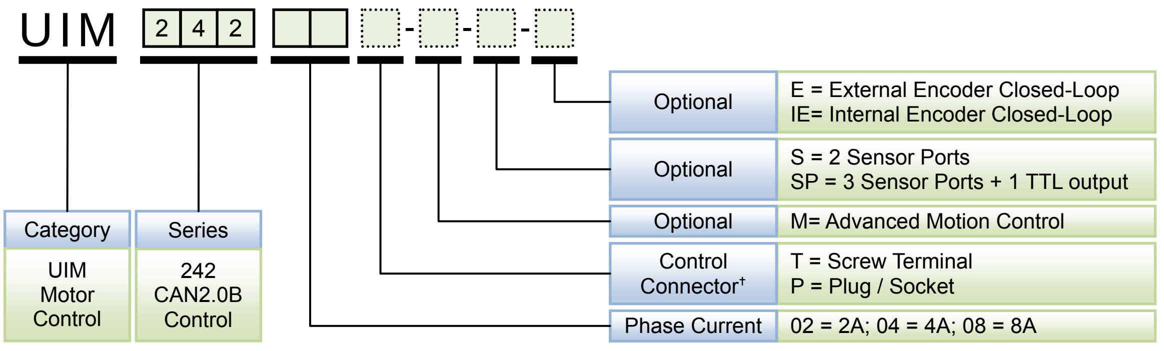
Miniature Integrated Stepper Motor Controller



[UIM242XX Ordering Information]

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

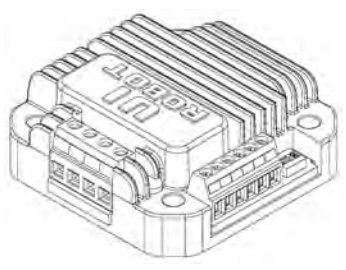
UIM242XX PART NUMBERING SYSTEM

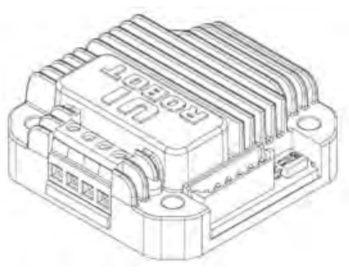


Note: † If not selected, the code box can be deleted. Default control connector is T (screw terminal), if not selected.

Examples: UIM24204P, UIM24202T-M, UIM24208-M-S-E, UIM24204-S

Examples of Control Connector options:


Screw Terminal


Rectangular Plug / Socket

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

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 ~ 40VDC
- Output current 2/4/8A, instruction adjustable
- Full to 16th micro-step resolution
- Dual full H-bridge with PWM constant current control
- Accurate micro-stepping and current control

CAN2.0B Active Communication

- 2-wire interface
- Max 1M bps operation, long distance
- Max 100 nodes
- Differential bus, high noise immunity

Embedded DSP Microprocessor

- Hardware DSP, 64bits calculating precision
- Quadrature encoder based closed-loop control
- Advanced motion control, linear and non-linear acceleration and deceleration, S-curve, PT/PVT displacement control
- Power-failure position protection
- 3 sensor input ports, 1 analog input (12bit)
- 1 TTL output
- 12 real-time event based change notifications
- 9 programmable actions triggered by 8 sensor events
- Simple instructions, intuitive and fault-tolerating

General Description

UIM24002 / UIM24204 / UIM24208 are miniature stepper motor controllers with CAN network interface. Through a CAN-RS232 converter (UIM2501), user device can operate a network of up to 100 UIM242 controllers through RS232 ASCII coded instructions. Instructions are simple, intuitive and fault-tolerating. User is not required to have knowledge on stepper motor driving and CAN network.

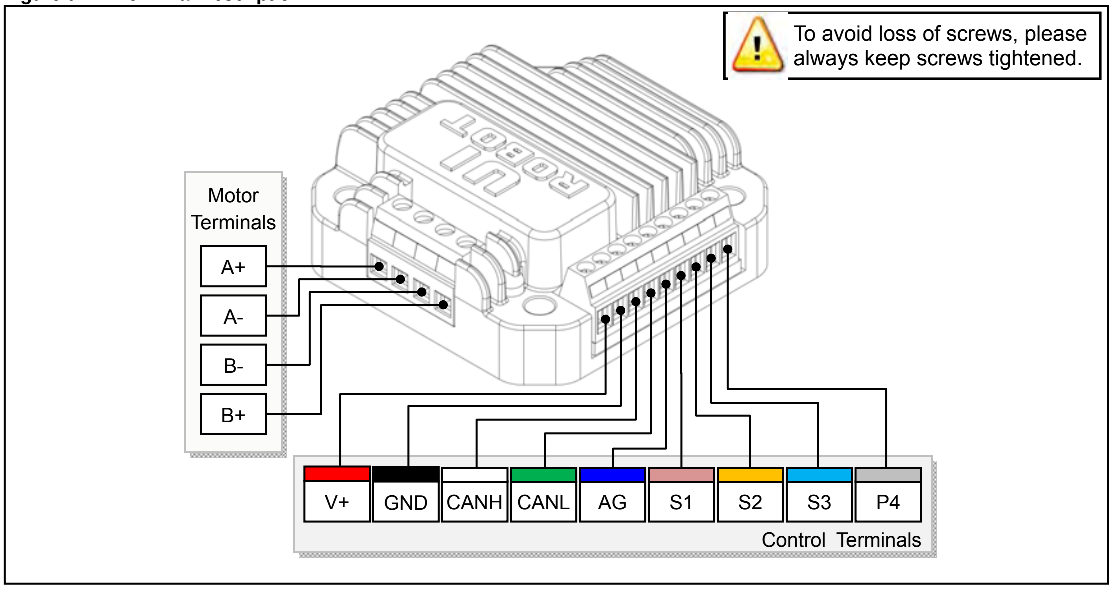
UIM242's architecture includes communication system, basic motion control system, Quadrature encoder interface and real-time event-based change notification system. Furthermore, there are three optional modules can be installed per customer request: Advanced Motion Control Module (linear/non-linear acceleration/deceleration, S-curve PT/PVT displacement control), Encoder-based Closed-loop Control Module and Sensor Input Control Module.

Embedded 64-bit calculating precision DSP controller guarantees the real-time processing of the motion control and change notifications. Entire control process is finished within 1 millisecond.

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

Terminal Description

Figure 0-1: Terminal Description



Control Terminals

Terminal No.	Designator	Description
1	V+	Supply voltage, 12 - 40VDC
2	GND	Supply voltage ground
3	CANH	CAN signal dominant high
4	CANL	CAN signal dominant low
5	AG	Analog ground for sensors(1)
6	S1	Sensor input port 1
7	S2	Sensor input port 2
8	S3	Sensor input port 3
9	P4	TTL signal output port

Note: (1) Internally linked to supply voltage ground.

Motor Terminals

Terminal No.	Designator	Description
1/2	A+ / A-	Connect to the stepper motor phase A
3/4	B- / B+	Connect to the stepper motor phase B

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.

Typical Application

UIM242 controllers can work standalone or within a CAN network. When working in a CAN network, up to 100 UIM242 controllers can be linked together using a minimum of 2 twisted wires.

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. TTL output should be connected to terminal 9, and signal ground should be connected to terminal 5.

Furthermore, please be aware:

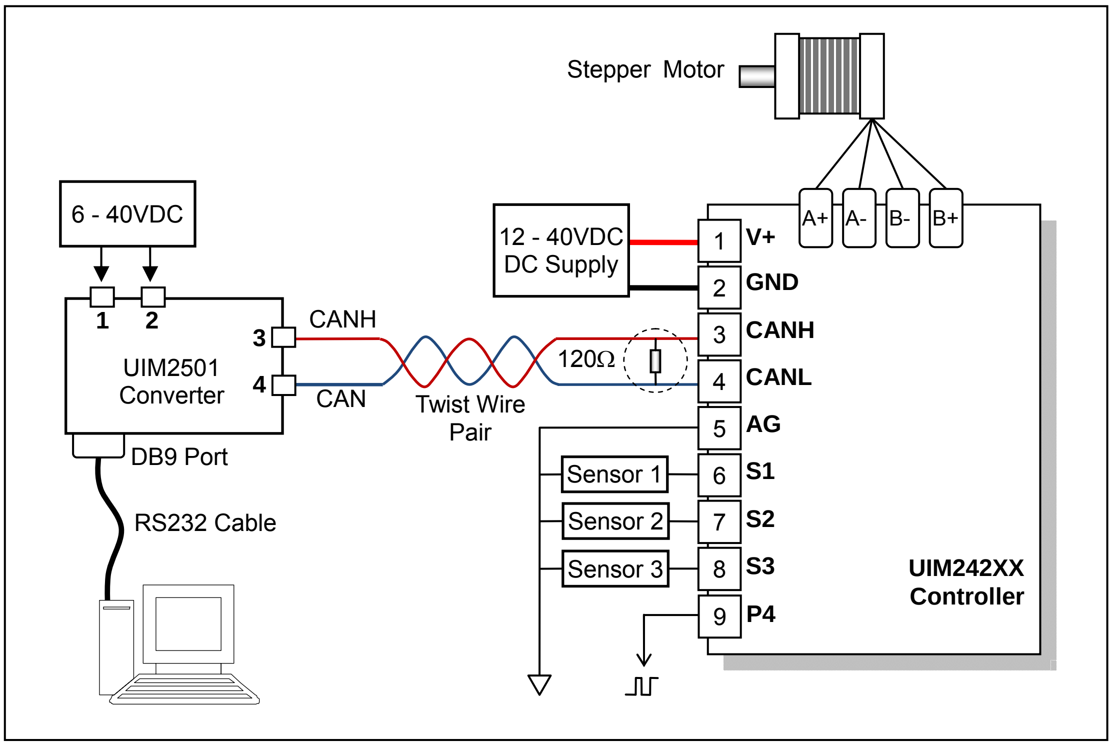
- User is responsible for the power supply for sensors.
- Voltage on terminal 6/7/8/9 must be kept between -0.3V and 5.3V, or smoke will be produced.
- For TTL output, the max sourcing / sinking current must be kept in 0~20mA.
- If using an external encoder, channel A should be connected to S1; channel B to S2; GND to AG.

Standalone Operation

When working standalone, user can use the wiring scheme shown in figure 0-2. 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 0-2: Wiring Scheme for Standalone Operation



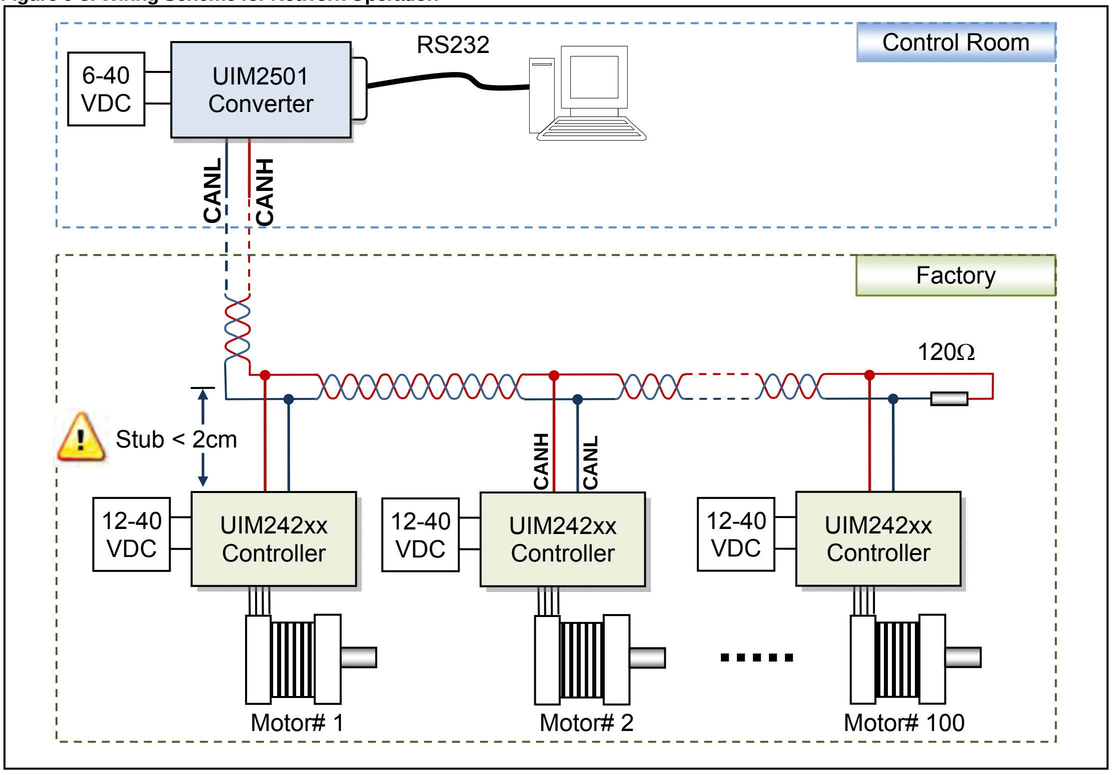
Network Operation

UIM242 controllers can work in a CAN network. In figure 0-3, 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 0-2.

In network operation, all nodes are connected onto a twist wire pair, as displayed in figure 0-3. Star connection scheme must be avoided. Meanwhile, 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 100 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 0-3: Wiring Scheme for Network Operation



Instruction Set Summary

Instruction	Description	Feedback Header	Message ID
MDL=X;	Check the model of controller with ID of x	0xCC	0xDE
MCFG=X;	Set master configuration register	0xAA	0xB0
MCFG;	Check master configuration register	0xAA	0xB0
ENA;	Enable H-bridge circuit	0xAA	-
OFF;	Disable H-bridge circuit	0xAA	-
CUR=X;	Set output phase current	0xAA	-
ACR=X;	Enable/disable automatic current reduction	0xAA	-
MCS=X;	Set micro-stepping resolution	0xAA	-
DIR=X;	Set motor direction (**obsoleted**)	0xAA	-
ORG;	Set zero/origin position	0xCC	0xB0
SPD=X;	Set the desired speed	0xAA	0xB5
SPD;	Check current speed	0xCC	0xB2
STP=X;	Set desired incremental displacement	0xAA	0xB6
STP;	Check current incremental displacement	0xCC	0xB3
POS=X;	Set desired position	0xAA	0xB7
POS;	Check current position	0xCC	0xB0
FBK;	Check current motor status	0xCC	-
MACC=X;	Set acceleration rate	0xAA	0xB1
MACC;	Check acceleration rate	0xAA	0xB1
MDEC=X;	Set deceleration rate	0xAA	0xB2
MDEC;	Check deceleration rate	0xAA	0xB2
MMSS=X;	Set maximum starting speed	0xAA	0xB3
MMSS;	Check maximum starting speed	0xAA	0xB3
MMDS=X;	Set maximum cessation speed	0xAA	0xB4
MMDS;	Check maximum cessation speed	0xAA	0xB4
SCFG=X;	Set sensor control configuration register	0xAA	0xC0
SCFG;	Check sensor control configuration register	0xAA	0xC0
SFBK;	Check sensor status	0xCC	0xC1
STORE;	Store motion control parameters	0xAA	0xD1
QER=X;	Set quadrature encoder’s resolution	0xAA	0xC2
QER;	Check quadrature encoder’s resolution	0xAA	0xC2
QEC=X;	Set desired quadrature encoder’s position	0xAA	0xB8
QEC;	Check current quadrature encoder’s position	0xCC	0xB1
DOUT=X;	Set output TTL level	0xAA	0xC1
DOUT;	Check current output TTL level	0xAA	0xC1
BTR=X;	Set CAN bus bit rate	0xAA	0xBC
BTR;	Check CAN bus bit rate	0xAA	0xBC

Characteristics

Absolute Maximum Ratings

Supply voltage	10V to 40V
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.

Electrical Characteristics（Ambient Temperature 25°C）

Supply Power Voltage	12V ~ 40VDC
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

Communication (Ambient Temperature 25°C）

Protocol	Active CAN 2.0B
Wiring method	2-wire，CANH、CANL
CAN bus drive	• Supports 1 Mb/s operation • ISO-11898 standard physical layer requirements • Short-circuit 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
Wight	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 or encoder-based closed-loop motion control. The control system comprises communication system, basic motion control system, absolute position counter, quadrature encoder interface and real-time event-based change notification system. There are also four optional modules to be added on customer request: *Advanced Motion Control Module* (linear/non-linear acceleration/deceleration, S-curve PV/PVT displacement control), *Encoder-based Closed-loop Control Module*, and *Sensor Input control Module* and *TTL Output 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.

UIM242's enclosure is made of die-cast aluminum to provide a rugged durable protection and improves the heat dissipation.

1.1 Basic Control System

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

Communication System

Through one CAN-RS232 converter (the UIM2501), user device can command multiple UIM242 controllers through RS232 using ASCII coded instructions. The UIM2501 translates the instructions from RS232 to CAN2.0B and sends the instructions to the target controller according to the controller ID that has been specified by user device. The CAN bit rate can be changed through instruction.

Basic Motion Control

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.

Open loop position control is possible using UIM242 controller. When desired position is reached, there could be automatically generated message feedback to the user device, given the corresponding configuration through user instruction.

Absolute Position Counter/Quadrature Encoder Interface

UIM242 has a hardware pulse counter. Output of the counter is signed. 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.

UIM242 controller has Quadrature Encoder Interface and can work with quadrature encoder when sensor input module is installed. 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 8 events: displacement control done, falling edge, analog input beyond upper threshold, analog input lower than lower threshold. All RTCNs can be enabled or disabled by instructions.

Similar to CPU's interrupters, UIM242 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.

UIM242's RTCN system supports 12 events: position/displacement control complete; 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).

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. Input types are configured through instruction. There is 1 channel can be configured as analog input. The on-board ADC converter has 12bit accuracy and 50K Hz sampling rate. Analog input is averaged over 16 samples.

User can configure the desired automatic action triggered by sensor status change. There are 9 actions listed below that can be triggered by 8 sensor events:

1. Start and Run Reversely (DIR=0).
2. Start and Run Forwardly (DIR=1).
3. Decelerate until Stop.
4. Reset position and encoder counter + Decelerate until Stop.
5. Emergency Stop.
6. Reset position and encoder counter + Emergency Stop.
7. Reverse (DIR=0) displacement control.
8. Forward (DIR=1) displacement control.
9. Reset position and encoder counter.

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.

The output level can be controlled by:

1. User instruction
2. One of the following events: a) run/stop status; b) direction change, and c) origin point hit.

1.5 Encoder-based Closed-loop Control Module

With the encoder-based closed-loop control module, UIM242 controller can perform self closed-loop motion control. Without this module, UIM242 can still interface with a quadrature encoder and provide reading to user device, but the self closed-loop is not available.

1.6 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. If the real-time change notification (RTCN) is enabled, 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 Instruction Structure

An instruction is a message sent from the user device to UIM242XX to command certain operation.

Instructions of UIM242XX 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 as follows:

   Instruction Symbol = Value; or

   Instruction Symbol;

   Where,

   Instruction Symbol comprises letters with no space between them, and is not case sensitive.

   Value comprises set of numbers, with no other characters between them. Some instructions have no Value, such as "SPD;", "STP;" etc.

   Terminator is the semicolon ";". Instruction without terminator will cause the UIM242XX to wait until the presence of the ";". In most situations, that will cause unpredictable results.

   Note: the equal symbol "=" is optional. User can use other characters except "{" and "}".
4. Only the first three letters of an instruction are used by the UIM242XX. Therefore the following two instructions are the same: "ENABLE;" and "ENA;"

2.2 Macro Operator and Null Instruction

In practice, users will combine several instructions together and send them at once. For example:

CUR=20; MCS=16; DIR=1; SPD=5000; ENA;

Normally, the user device will receive an ACK message on every instruction sent. Thus the above instruction set will cause 5 ACK messages being transferred on the RS232 bus. Especially for those basic motion instructions like SPD, DIR, MCS, which have the same ACK, sending a set of ACK is unnecessary. 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; DIR=1; SPD=5000; ENABLE; };

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.

2.3 Feedback Message Structure

Feedback Message is the message sent to user device from UIM 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]

Header denotes the start of a feedback message. There are 3 kinds of headers:

1. 0xAA represents the ACK message, which is a repeat of the received instruction.
2. 0xCC represents the status feedback, which is a description of current working status.
3. 0xEE represents the error message.

Controller ID is the identification number of current controller in a CAN network.

Message ID denotes the property of the current message.

For example, 0xCC 0x05 0xA0 0xFF, where 0xA0 denotes that the current message means a falling edge happened at sensor S1 port.

Data has a 7bits data structure. In figure 2-1and figure 2-2, examples are shown on how to convert a set of 7bits data into 16bit data and 32 bits data. Obviously, 16bit data takes three 7bits data, and 32bits data needs five 7btis data to represent.

Terminator denotes the end of a feedback message. UIM242XX controller utilizes "0xFF"as the terminator.

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.

Figure2-1: Conversion from three 7bits message to a 16bits data

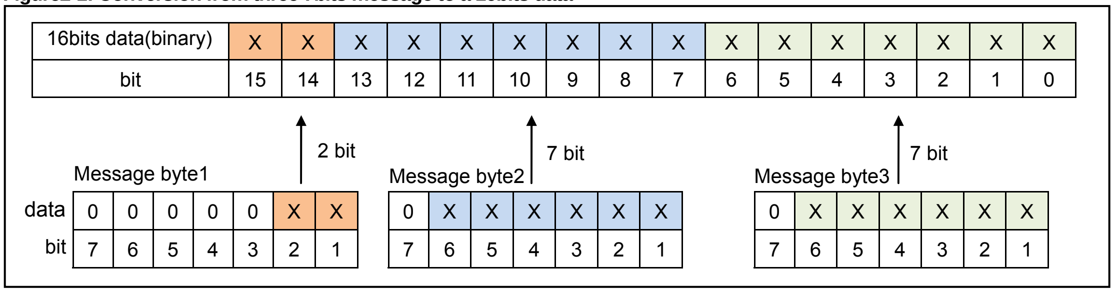
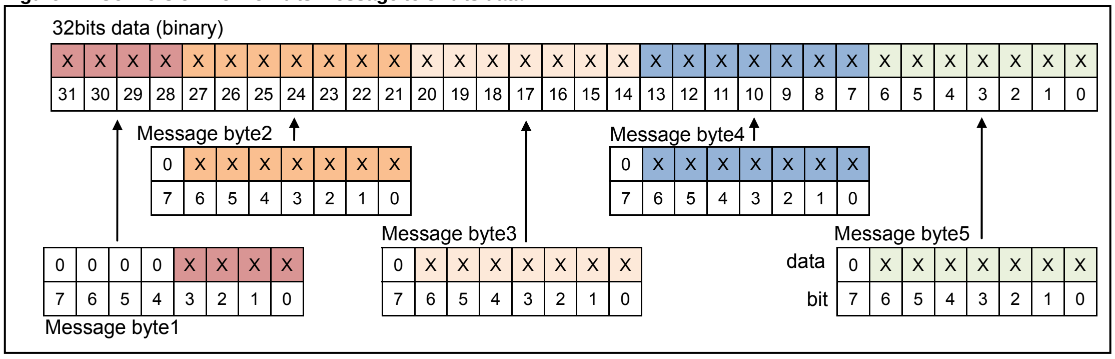


Figure2-2: Conversion from 5 7bits message to 32bits data



3.0 CAN2.0 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. ACK and/or feedback messages are sent back from UIM242 controllers to the UIM2501 and then translated into RS232 messages, and sent back to user device.

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. 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

In order to communicate properly, every UIM242XX controller needs to have a unique identification code (ID, or address), even in standalone operation (Figure 0-2).

Every UIM242xx controller has a factory default ID of 5. User can change the ID through instruction. 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. If two or more UIM242 controllers are connected to a UIM2501 during ID assignment operation, the process will fail.

3.2 Check Controller Model (MDL)

MDL=x;	
Function	Check the Model, installed optional modules and firmware version of the UIM242 controller of ID = **x**.
Variable	Integer x = 5, 6 … 125
Feedback	0xCC [Controller ID] 0xDE 0x18 0x2 [CUR] [ASM] V2 V1 V0 0xFF
Comment	0xDE is the Message ID of instruction MDL. [CUR] denotes the Max phase current. e.g., “20” means 2.0 A. [ASM] denotes the installed optional modules. It has the following structure: bit 7 6 5 4 3 2 1 0 --- Meaning 0 Int. QE Closed-loop Adv. Motion No. of Sensor Ports For example, if bit 4 is 1, the Advanced Motion Control module is installed. V2 – V0 denote the firmware version. Data is in 7 bits format. Conversion from three 7bits message data to a 16bits data is illustrated in figure 2-1.

3.3 CAN2.0B Bit rate and Global Instructions

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

UIM242XX 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.

4.1 RTCN Structure

The structure of an RTCN message is shown below:

0xAA [Controller ID] [Message ID] 0xFF

For UIM242, the Controller ID is preset by user.

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

Figure 3-1: Real-time change notification events

No.	Event	Message ID	Description
1	falling edge of S1	0xA0	Voltage on S1: High >>>Low
2	rising edge of S1	0xA1	Voltage on S1: Low >>>High
3	falling edge of S2	0xA2	Voltage on S2: High >>>Low
4	rising edge of S2	0xA3	Voltage on S2: Low >>>High
5	falling edge of S3	0xA4	Voltage on S3 port: High >>>Low
6	rising edge of S3	0xA5	Voltage on S3 port: Low >>>High
7	TTL output P4 low	0xA6	Voltage on P4 port: High >>>Low
8	TTL output P4 high	0xA7	Voltage on P4 port: Low >>>High
9	exceed upper limits	0xA1/0xA5*	Analog input > user preset upper limit
10	below lower limit	0xA0/0xA4**	Analog input < user preset lower limit
11	displacement control complete	0xA8	The desired position is reached
12	zero position	0xA9	Position counter reaches/passes zero

Note:

* When S1 is configured as analog, 0xA1 denotes event 9, otherwise 0xA1 denotes event 2.
When S3 is configured as analog, 0xA5 denotes event 9, otherwise 0xA5 denotes event 6.

** When S1 is configured as analog, 0xA0 denotes event 9, otherwise 0xA0 denotes event 1.
When S3 is configured as analog, 0xA4 denotes event 9, otherwise 0xA4 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 Hardware/Firmware Configuration

UIM242's hardware and firmware can be configured through user instructions. This can be achieved through writing the corresponding configuration registers.

There are 6 configuration registers for UIM242: *Master Configuration Register, Sensor Input Control Register, TTL Output Control Register and 2 Analog Threshold Registers*. In this chapter, only the *Mater Configuration Register* is described. User can find details about the other 5 registers in their corresponding chapters.

5.1 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:

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	X	ORGIE	STPIE	P4IE	S3IE	S2IE	S1IE

Bit 15 **ANE Enable / Disable Analog Input**

0 = Disable the analog input, port S1 is digital
1 = Enable the analog input

Bit 14 **CHS Analog Input Channel**

0 = Analog input on port S1
1 = Analog input on port S3

Bit 13 **QEI Enable/Disable Quadrature Encoder Interface**

0 = Disable Quadrature Encoder Interface
1 = Enable Quadrature Encoder Interface

Bit 12 **Unimplemented. Read as 0.**

Bit 11 **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

Bit 10 **CM Advanced Motion Control Mode**

0 = Disable advanced motion control module, use basic control mode
1 = Enable advanced motion control module

Bit 9 **AM Acceleration Mode**

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

Bit 8 **DM Deceleration Mode**

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

Bit 7-6 **Unimplemented. Read as 0.**

Bit 5 **ORGIE Origin (Zero) Position RTCN**

0 = Disable the Origin (zero) position RTCN.
1 = Enable the Origin (zero) position RTCN.

Bit 4 **STPIE Displacement Control (STP/POS/QEC) Completion RTCN**

0 = Disable the displacement control completion RTCN.
1 = Enable the displacement control completion RTCN.

Bit 3	P4**IE**	**P4 Status Change RTCN**
	0 = Disable P4 status change RTCN 1 = Enable P4 status change RTCN	
Bit 2	**S3IE**	**S3 Status Change RTCN**
	0 = Disable S3 status change RTCN 1 = Enable S3 status change RTCN	
Bit 1	**S2IE**	**S2 Status Change RTCN**
	0 = Disable sensor port 2 (S2) status change RTCN 1 = Enable S2 status change RTCN	
Bit 0	**S1IE**	**S1 Status Change RTCN**
	0 = Disable sensor port 1 (S1) status change RTCN 1 = Enable S1 status change RTCN	

5.2 Master Configuration Register Instruction (MCFG)

MCFG = x;		
Function	Setup Master Configuration Register	
Variable	Integer x = 0, 1 … 65535, or Hexadecimal x= 0x0000 … 0xFFFF	
ACK	0xAA [Controller ID] 0xB0 CFG2 CFG1 CFG0 0xFF	
Comment	0xB0 is the Message ID of MCFG CFG2 – CFG0 denotes the master configuration register value. See figure 2-1 for how to convert to a 16bit integer. If x using decimal, first fill each bit of the master configuration register with 0 or 1, and then convert them to a decimal based number. If x using hexadecimal, the number must start with “0x”.	
Example	User Send	MCFG=34611; or MCFG=0x8733;
	ACK Message	0xAA 0x00 0xB0 0x02 0x0E 0x33 0xFF
	Interpretation	Convert 0x2 0xE 0x33 to 16bit integer, we get: 0x8733 (That is 34611 decimal)

5.3 Check Master Configuration Register

MCFG;	
Function	Check the value of the Master Configuration Register
Variable	N/A
ACK	0xAA [Controller ID] 0xB0 CFG2 CFG1 CFG0 0xFF
Comment	0xB0 is the Message ID of MCFG. CFG2 – CFG0 denotes the master configuration register value. See figure 2-1 for how to convert to a 16bit integer.

6.0 Basic Control Instructions

UIM242XX controllers support the following basic control instructions.

	Instruction	Function	Example
1	ENA	Enable the motor driving circuit	ENA;
2	OFF	Disable the motor driving circuit	OFF;
3	CUR	Set desired motor phase current	CUR=17; CUR17;
4	MCS	Set micro-stepping resolution	MCS16;
5	ACR	Enable / disable Automatic Current Reduction	ACR=1; ACR1;
6	DIR	Set desired motor direction (**obsoleted**)	
7	SPD	Set desired speed PPS (pulse per second) Check present speed	SPD65000; SPD-65000;
8	STP	Set desired incremental displacement Check present incremental displacement	STP =-30000;
9	FBK	Inquiry present motor working status	FBK;
10	ORG	Reset the position/encoder counter	ORG;
11	POS	Set desired position Check present position	POS+20000000;

The above instructions 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 first provided, followed by detailed description of above instructions.

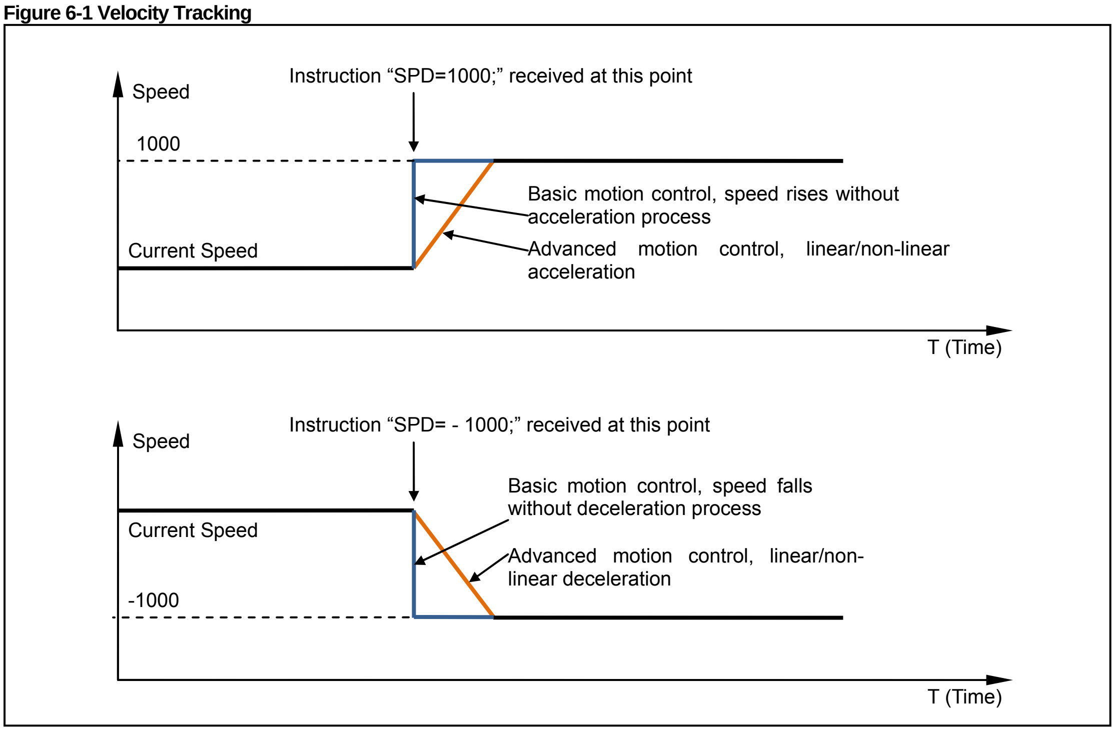
6.1 General Introduction of Motion Control Modes

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.

Figure 6-1 Velocity Tracking


Speed
Instruction "SPD=1000;" received at this point
1000
Basic motion control, speed rises without acceleration process
Current Speed
Advanced motion control, linear/non-linear acceleration
T (Time)
Speed
Instruction "SPD= - 1000;" received at this point
Basic motion control, speed falls without deceleration process
Current Speed
Advanced motion control, linear/non-linear deceleration
-1000
T (Time)

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.

– The DIR instruction is obsoleted. However, if a DIR instruction occurs after an SPD instruction, it will still affect motor direction.

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.

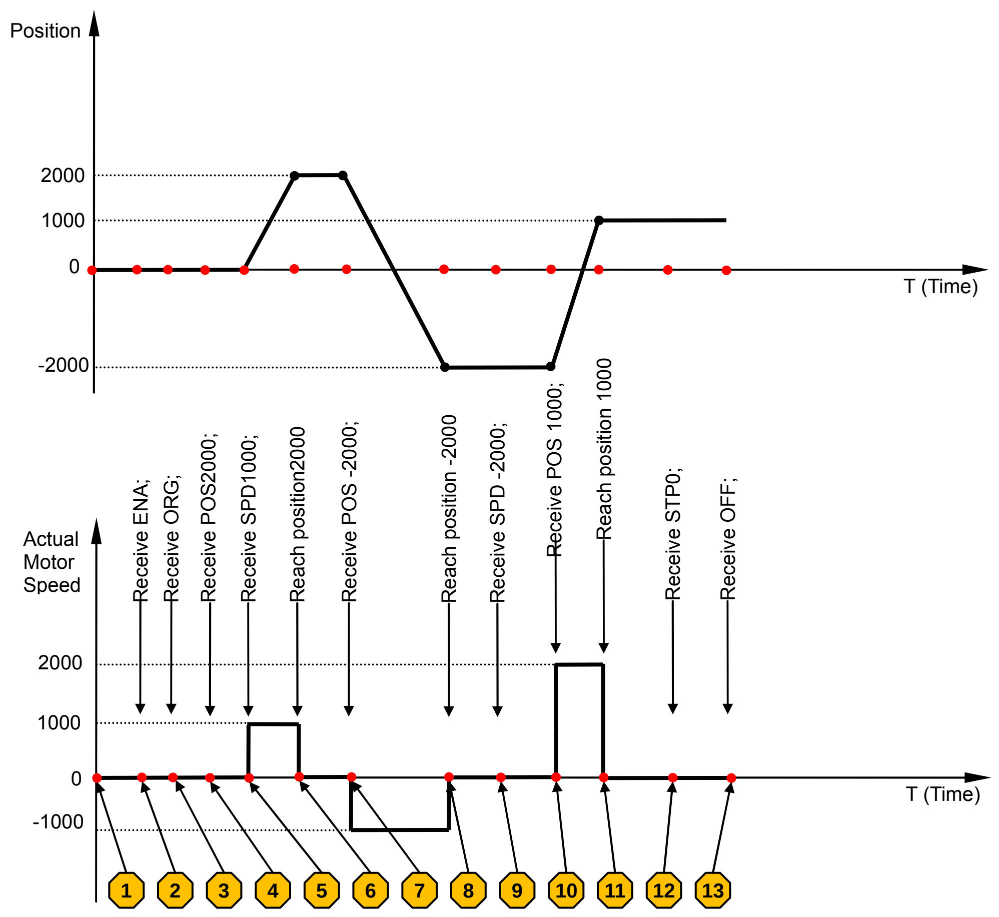
Position Tracking (PT)

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 or QEC 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.

Figure 6-2 Position Tracking Mode (without acceleration/deceleration)



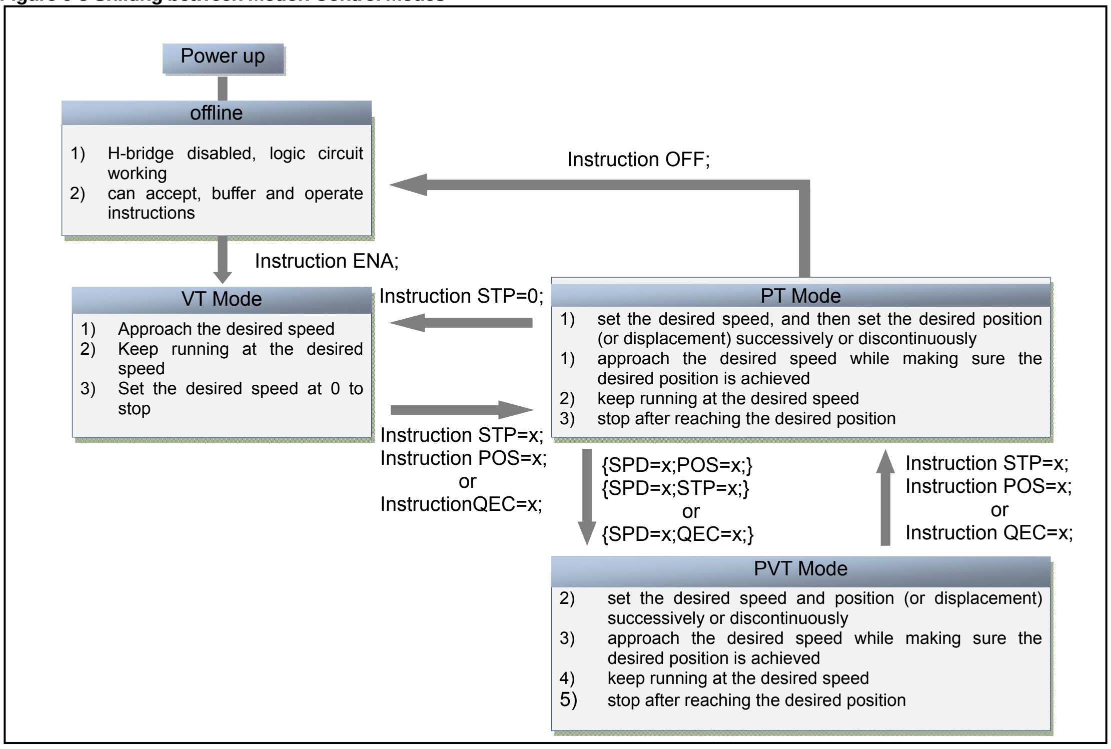
No.	Operation or Event	Control Mode	Desired Position	Current Position	Position Error	Desired Speed	Motor Direction	Motor Speed
1	Power up	VT	0	Stored position	- Stored position	0	1	0
2	ENA	VT	0	Stored position	- Stored position	0	1	0
3	ORG	VT	0	0	0	0	1	0
4	POS	PT	2000	0	2000	0	1	0
5	SPD	PT	2000	0	2000	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:

Figure 6-3 Shifting between Motion Control Modes


Power up
offline
1) H-bridge disabled, logic circuit working
2) can accept, buffer and operate instructions
Instruction OFF;
Instruction ENA;
VT Mode
1) Approach the desired speed
2) Keep running at the desired speed
3) Set the desired speed at 0 to stop
Instruction STP=0;
PT Mode
1) set the desired speed, and then set the desired position (or displacement) successively or discontinuously
1) approach the desired speed while making sure the desired position is achieved
2) keep running at the desired speed
3) stop after reaching the desired position
Instruction STP=x; Instruction POS=x; or InstructionQEC=x;
{SPD=x;POS=x;} {SPD=x;STP=x;} or {SPD=x;QEC=x;}
Instruction STP=x; Instruction POS=x; or Instruction QEC=x;
PVT Mode
2) set the desired speed and position (or displacement) successively or discontinuously
3) approach the desired speed while making sure the desired position is achieved
4) keep running at the desired speed
5) stop after reaching the desired position

6.2 H-Bridge Enable Instruction (ENA)

ENA;	
Function	Enable the stepper motor driver (i.e. H-bridge driving circuit).
Variable	N/A
ACK	Refer to the following “Basic Instruction ACK” for details.
Comment	Only after the H-bridge enabled, can the controller drive the motor.

6.3 H-Bridge Disable Instruction (OFF)

OFF;	
Function	Disable the stepper motor driver (i.e. H-bridge driving circuit).
Variable	N/A
ACK	Refer to the “Basic Instruction ACK” for details.
Comment	OFF instruction turns off the dual H-bridge motor driving circuit. Once an OFF instruction is executed, the motor will have no power supply, the power consumption is cut to minimum (the logic circuit is still working). User needs to use the ENABLE instruction to turn the motor driver back to working.

6.4 Motor Current Adjusting Instruction(CUR)

CUR = x;	
Function	Set the output phase current to x.
Variable	Integer x = 0、1 … 80
ACK	Refer to the “Basic Instruction ACK” for details.
Comment	Integers 0 ... 80 represent 0 ... 8.0 amps. Once received, the current value will be stored in the controller’s 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.

6.5 Automatic Current Reduction Instruction (ACR)

ACR = x;	
Function	Enable/disable ACR (automatic current reduction) function.
Variable	Integer x = 0,1
ACK	Refer to the “Basic Instruction ACK” for details.
Comment	If ACR = 1; the function is enabled, vice versa. When ACR is enabled, the current will be reduced after motor stops, which means a decrease of holding torque. Value of this instruction will be stored in EEPROM.

6.6 Micro Stepping Setup Instruction (MCS)

MCS = x;	
Function	Set micro-stepping resolution.
Variable	Integer x = 1, 2, 4, 8, 16
ACK	Refer to the “Basic Instruction ACK” for details.
Comment	x = 1, 2, 4, 8, 16 represents the full, half, quarter, eighth and sixteenth step resolution, respectively. Once received, the MCS value will be stored in the controller’s 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.

6.7 Motion Direction Instruction (DIR)

DIR = x; (obsoleted, do not use)	
Function	Set the desired motor direction.
Variable	Integer x = 0, 1
ACK	Refer to the “Basic Instruction ACK” for details.
Comment	Motor direction is now determined by the sign of the speed. The actual motor direction also depends on the wiring between motor and controller.

6.8 Absolute Position Counter Reset Instruction (ORG)

ORG;	
Function	Reset the position/encoder counter, create an origin point.
Variable	N/A
Feedback	0xCC [Controller ID] 0xB0 0x00 0x00 0x00 0x00 0x00 0xFF
Comment	0xCC indicates that a feedback message is received 0xB0 is the message ID of ORG

6.9 Speed Adjusting Instruction (SPD)

SPD = x;	
Function	Set the desired speed to x.
Variable	Integer x = - 65535…-1, 0, 1 … + 65535
ACK	0xAA [Controller ID] 0xB5 SPD2 SPD1 SPD0 0xFF
Comment	0xAA indicates confirm of instruction (ACK) 0xB5 is the message ID for desired speed (SPD) SPD2 – SPD0 denotes the desired motor speed. See figure 2-1 for how to convert to a signed 16bit integer. Unit is pulse/second, PPS or Hz. The sign of the value decides motor direction. If no "+" or "-" specified before "x", it is taken as "+". Once H-bridge is enabled, motor starts running on receiving the instruction "SPD=x;" (x≠0) until another instruction "SPD=0;" is given.
Example	For a 1.8° stepper motor, if the SPD =100; User sent: SPD = 100; If MCS = 1; motor speed = 1.8*100 = 180°/sec = 30 rpm If MCS =16; motor speed = 1.8*100/16 = 11.25°/ s = 1.875rpm

6.10 To Check Current Speed (SPD)

SPD;	
Function	Check current speed.
Variable	N/A
Feedback	0xCC [Controller ID] 0xB2 SPD2 SPD1 SPD0 0xFF
Comment	0xCC denotes feedback of current status 0xB2 is the message ID of current speed (SPD) SPD2 – SPD0 denotes the current motor speed. See figure 2-1 for how to convert to a signed 16bit integer. Unit is pulse/second, PPS or Hz. The sign of the value denotes motor direction.

6.11 Displacement Control Instruction (STP)

STP = x;	
Function	Set the desired incremental displacement (steps or micro-steps if MCS≠1).
Variable	Integer x = - 2,000,000,000…-1, 0, 1 … + 2,000,000,000
ACK	0xAA [Controller ID] 0xB6 STP4 STP3 STP2 STP1 STP0 0xFF
Comment	0xB6 is the message ID of STP STP4 – STP0 denotes the desired motor displacement. See figure 2-2 for how to convert to a signed 32bit integer. Displacement is essentially defined as counts of the pulse or encoder counter. Therefore the actual motor displacement is also relative to the micro-stepping resolution or encoder resolution. If an STP=0; 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;
	User sent: STP = 200;
	If MCS = 1, motor rotation angle = 1.8 * 200 = 360°
	If MCS = 16, motor rotation angle = 1.8 * 200 / 16 = 22.5°

6.12 To check STP displacement

STP;	
Function	Check current incremental displacement.
Variable	N/A
Feedback	0xCC [Controller ID] 0xB3 STP4 STP3 STP2 STP1 STP0 0xFF
Comment	0xCC denotes current status feedback 0xB3 is the message ID of current incremental displacement (STP) STP4 – STP0 denotes the current incremental displacement. See figure 2-2 for how to convert to a signed 32bit integer. Displacement is essentially defined as counts from the pulse counter or encoder. Therefore the actual angular displacement is relative to micro-stepping resolution or encoder resolution.

6.13 Position Control Instruction (POS)

POS=x;	
Function	Set desired position (for open-loop control).
Variable	Integer x = - 2,000,000,000…-1, 0, 1 … + 2,000,000,000
ACK	0xAA [Controller ID] 0xB7 P4 P3 P2 P1 P0 0xFF
Comment	0xB7 is the message ID of desired position (POS) P4 – P0 denotes the desired absolute position. See figure 2-2 for how to convert to a signed 32bit integer. Position is essentially recorded from counts of the pulse counter. Therefore the actual motor position is also relative to the micro-stepping resolution. The position counter records the total pulses sent to motor. When the direction is positive (DIR=1), the counter increases by 1; when the direction is negative (DIR=0), the counter decreases by 1. Therefore, the value of the counter is a signed 32bits integer, with positive representing the final position is of the same direction of DIR=1, and vice versa. POS position control is open-loop control. The absolute position counter only resets (back to zero) in two situations: 1. User issues the instruction ORG (described later); 2. User pre-configured sensor ORG event takes place. 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.

6.14 Check Current Position (POS)

POS;	
Function	Check current position.
Variable	N/A
Feedback	0xCC [Controller ID] 0xB0 P4 P3 P2 P1 P0 0xFF
Comment	0xB0 is the message ID of current position (POS) P4 – P0 denotes the desired absolute position. See figure 2-2 for how to convert to a signed 32bit integer. Position is essentially recorded from counts of the pulse counter. Therefore the actual motor position is also relative to the micro-stepping resolution.

6.15 Basic Instruction Acknowledgment (ACK)

Upon receiving an instruction, the UIM242XX controller will immediately send back an Acknowledgment (ACK) message. For all basic instructions describe before except SPD, STP, POS and ORG, 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.

There are two kinds of errors: Syntax error and value error (i.e., variable is incorrect). The structure of an error message is:

0xEE [Error Code] 0xFF

Where,

0xEE denotes an error message.

The error code is list below:

Error Code	0x65	0x66
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	0xAA	Controller ID	ASM	CUR	SPD2	SPD1	SPD0	STP4	STP3	STP2	STP1	STP0	0xFF

Where,

1. 0xAA denotes a basic ACK message
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 2-1 for how to convert to a signed 16bit integer. Unit is pulse/second, PPS or Hz. The sign of the value decides motor direction.
5. STP4 – STP0 denotes the desired motor displacement. See figure 2-2 for how to convert to a signed 32bit integer. Displacement is essentially defined as counts from the pulse counter or encoder. Therefore the actual angular displacement is relative to micro-stepping resolution or encoder resolution.

6.16 Motor Status Feedback Inquiry Instruction (FBK)

If user wants to check the current motor status, following instruction should be used. Please note that, motor status and desired settings could be different.

FBK;	
Function	Check the current motor status.
Variable	N/A
Feedback	See the following section
Comment	FBK is the abbreviation for Feedback.

6.17 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	0xCC	Controller ID	ASM	CUR	SPD2	SPD1	SPD0	STP4	STP3	STP2	STP1	STP0	0xFF

Where,

1. 0xCC denotes a Motor Status Feedback Message. (i.e., the present value of motor status)
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 (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 2-1 for how to convert to a signed 16bit integer. Unit is pulse/second, PPS or Hz. The sign of the value decides motor direction.
5. STP4 – STP0 denotes the current motor displacement. See figure 2-2 for how to convert to a signed 32bit integer. Displacement is essentially defined as counts from the pulse counter or encoder. Therefore the actual angular displacement is relative to micro-stepping resolution or encoder resolution.

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.

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 5 additional instructions. Once the advanced motion control module is enabled, all basic control instructions are automatically turned into advanced control instructions.

	Instruction	**Function**	**Example**
1	MCFG	Enable/disable the advanced motion control module.	MCFG1792;
2	MACC	Set the acceleration rate	MACC=200;
3	MDEC	Set the deceleration rate	MDE500;
4	MMSS	Set the Maximum Starting Speed	MMS1600;
5	MMDS	Set the Maximum Cessation Speed	MMDS1000;

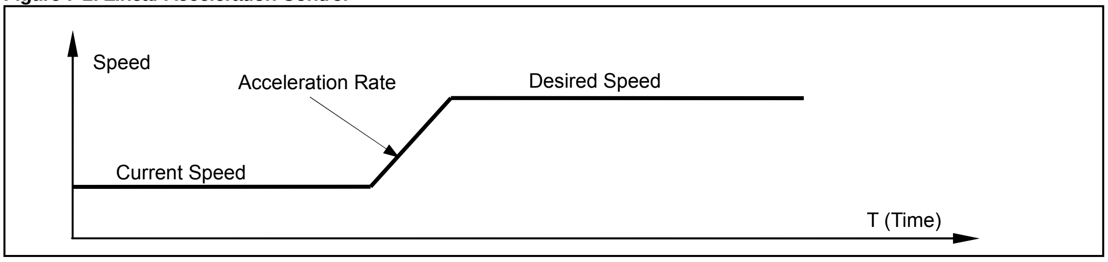
It takes less than 1 millisecond for the specified parameter to take effect after the instruction is received. Values of above instructions will be stored in the EEPROM. Once the parameters are set, the controller will perform the advanced motion control automatically. At any time, user can use instructions (e.g., FBK, POS, SPD, etc.) to get the current status of the motor.

In this chapter, the Advanced Motion Control processes are first introduced, followed by introduction to above 5 instructions.

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, UIM242 controller will perform the acceleration process automatically.

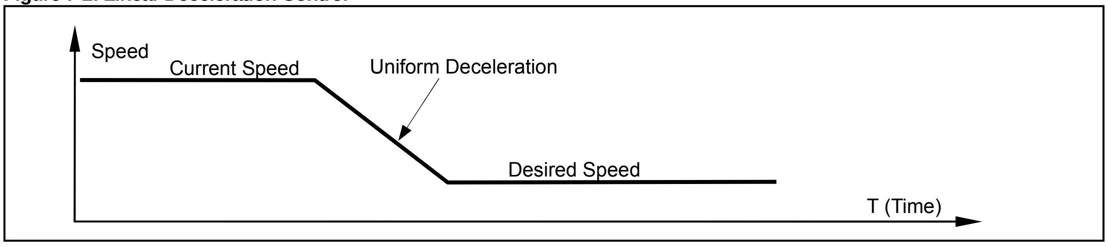
Figure 7-1: Linear Acceleration Control



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, UIM242 controller will perform the deceleration process automatically.

Figure 7-2: Linear Deceleration Control

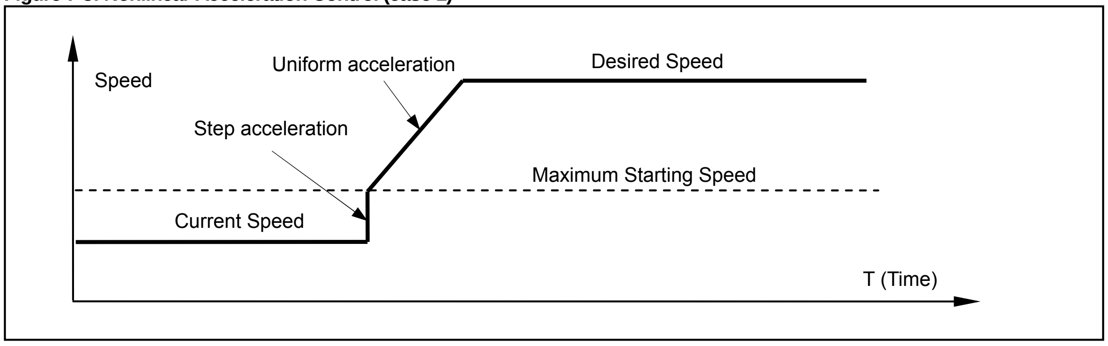


7.3 Nonlinear Acceleration

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.

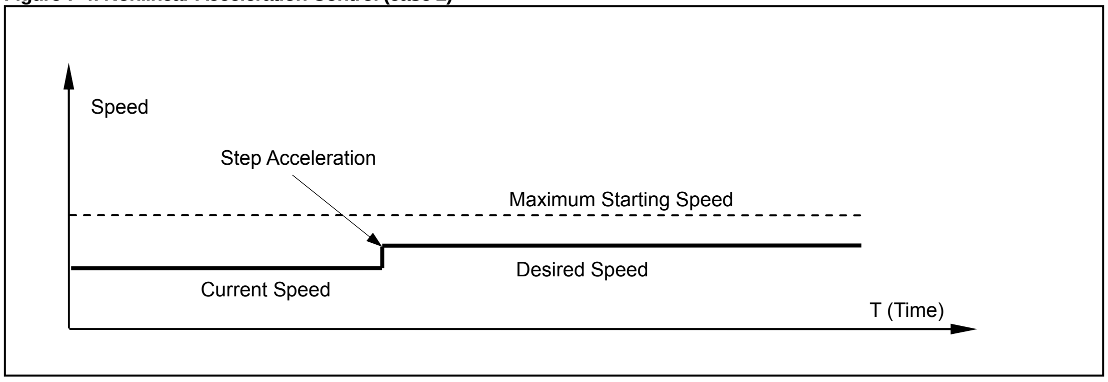
If the desired speed is higher than a certain value (i.e. the Maximum Starting Speed, defined by instruction), and current motor speed is lower than the Max. Starting Speed, then the motor speed will first step up to the Max Starting Speed and then linearly accelerated according to the acceleration rate, as shown in figure 7-3.

Figure 7-3: Nonlinear Acceleration Control (case 1)



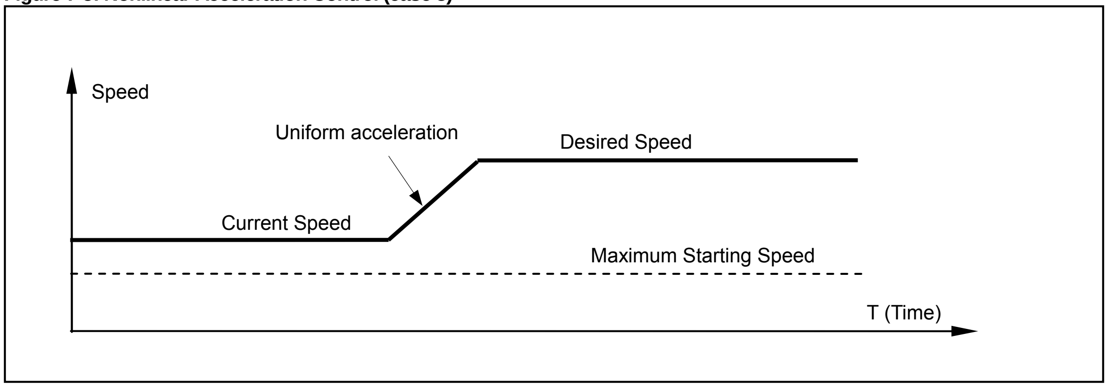
If the desired speed is less than the Max Starting Speed, then the motor speed will step up to the desired speed immediately, as shown in figure 7-4.

Figure 7-4: 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.

Figure 7-5: Nonlinear Acceleration Control (case 3)

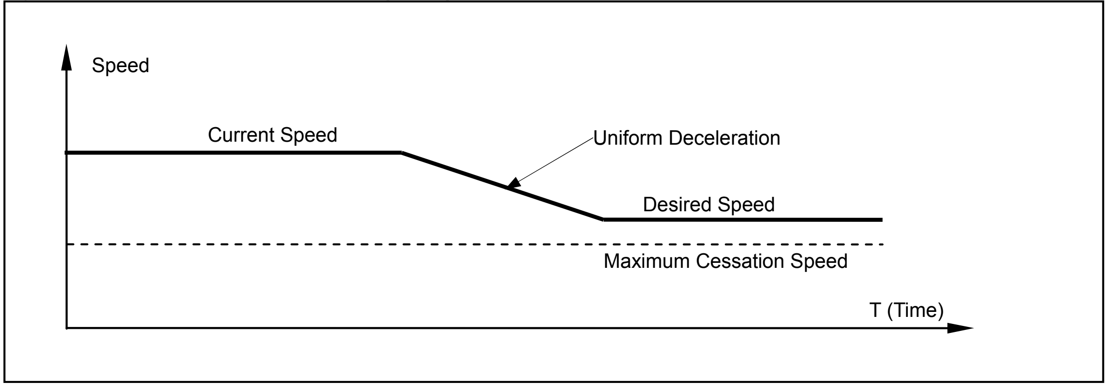


7.4 Nonlinear Deceleration

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

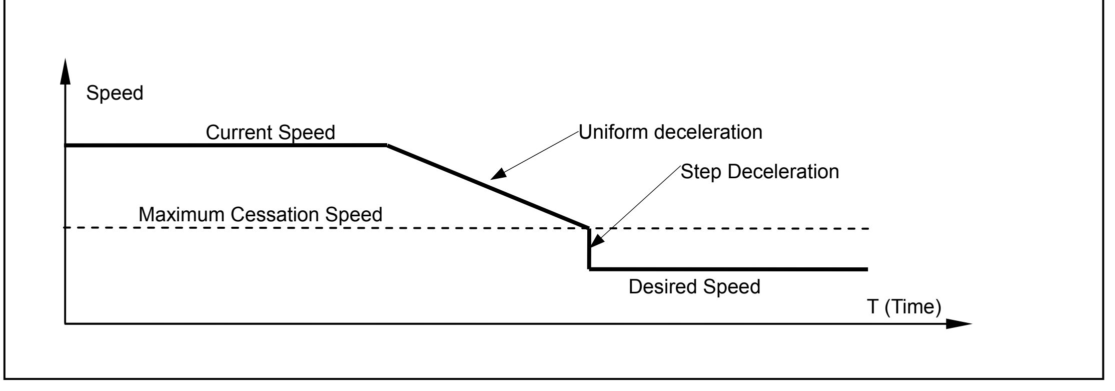
If the desired speed is higher than a certain user preset value (i.e. the Maximum Cessation Speed), UIM242XX will use the Uniform Deceleration Control algorithm.

Figure 7-6: Nonlinear Deceleration Control (case 1)



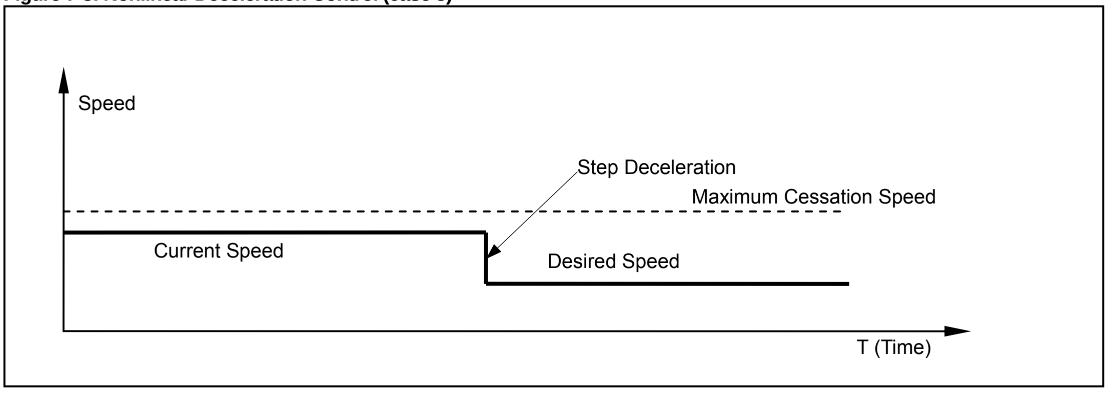
If desired speed is lower than the Max Cessation Speed and current motor speed is higher than the Max. Cessation Speed, the Uniform Deceleration Control will be first applied and followed by a step deceleration to the desired speed.

Figure 7-7: 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.

Figure 7-8: Nonlinear Deceleration Control (case 3)

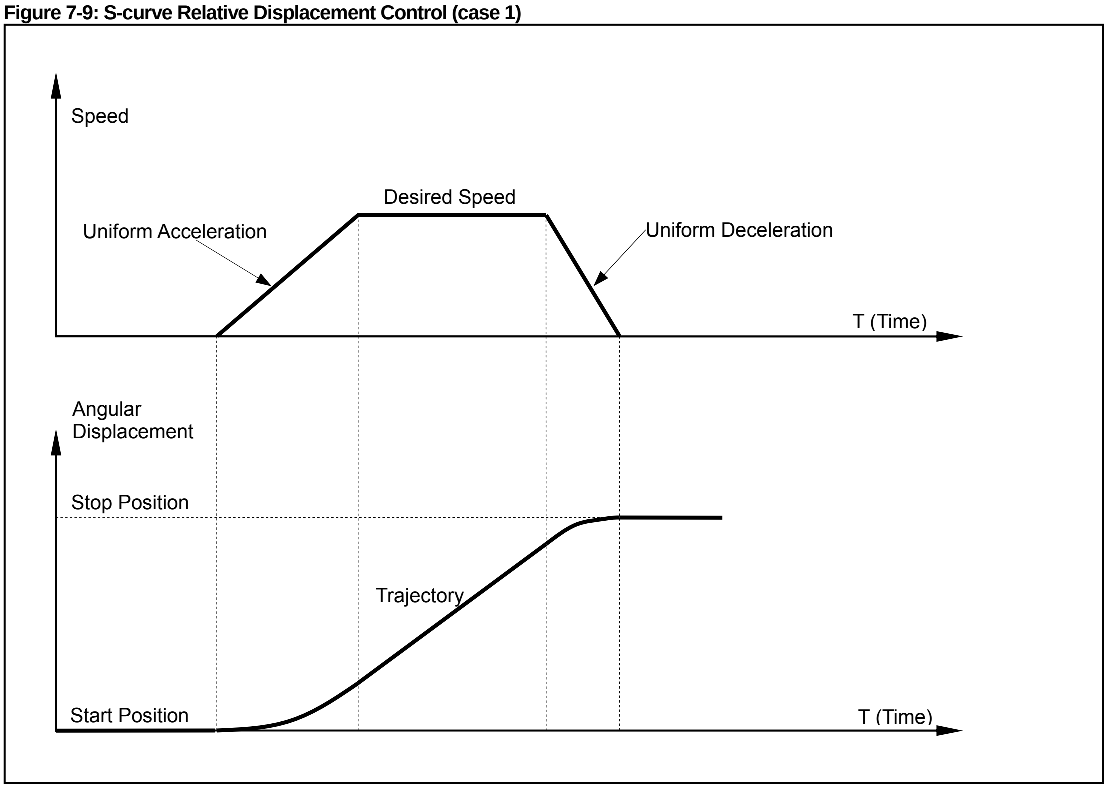


Note: Setting the Maximum Starting Speed or the Maximum Cessation Speed to 0(zero) will force the controller use Linear Acceleration / Deceleration Control Algorithm.

7.5 S-curve Displacement Control

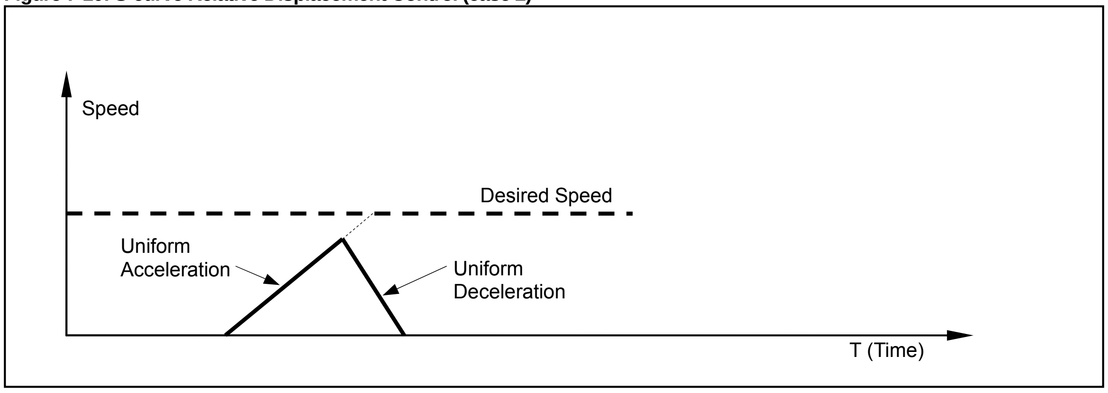
S-curve displacement control essentially is the displacement control under the linear acceleration and deceleration speed control. The name is originated from the shape of the motion trajectory. The original S-curve displacement control is the acceleration-coast-deceleration speed control. In the entire 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.

Figure 7-9: S-curve Relative Displacement Control (case 1)



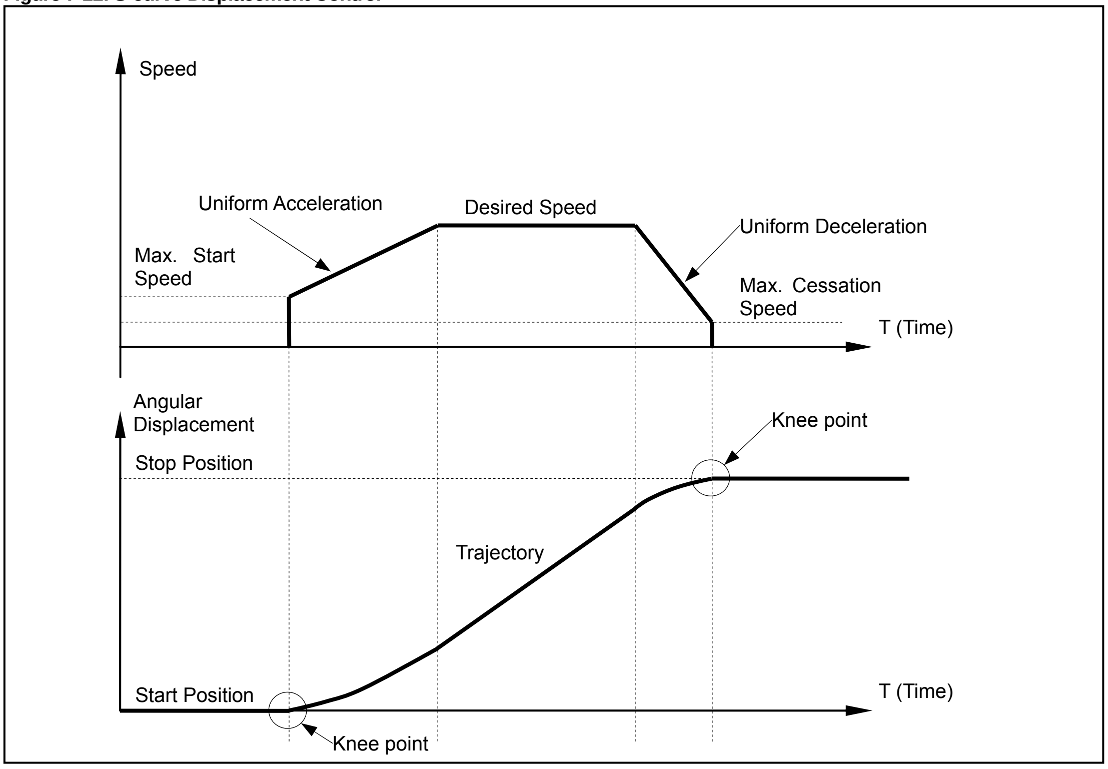
In the control process, UIM242XX's advance motion control module will continuously calculate the deceleration happening point (time) and then perform the deceleration to guarantee that when desired displacement is reached, the speed is right zero. The entire calculation time is around 50 micro-seconds 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.

Figure 7-10: 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.

Figure 7-11: 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. The motor must first be decelerated to zero speed before turned to the desired direction.

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. When DIR=1, the counter (pulse) increases and when DIR=0, the counter decreases. 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 Advanced Motion Control Instructions

Once the advanced motion control module is enabled, all basic control instructions are automatically turned into advanced control instructions. This transition is transparent to the user. Furthermore, there are 5 additional instructions added as listed below.

1. MCFG

   This is the instruction to enable or disable the advanced motion control module. 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. mACC

   This is the instruction set the acceleration rate. There are two ways to set the acceleration rate:

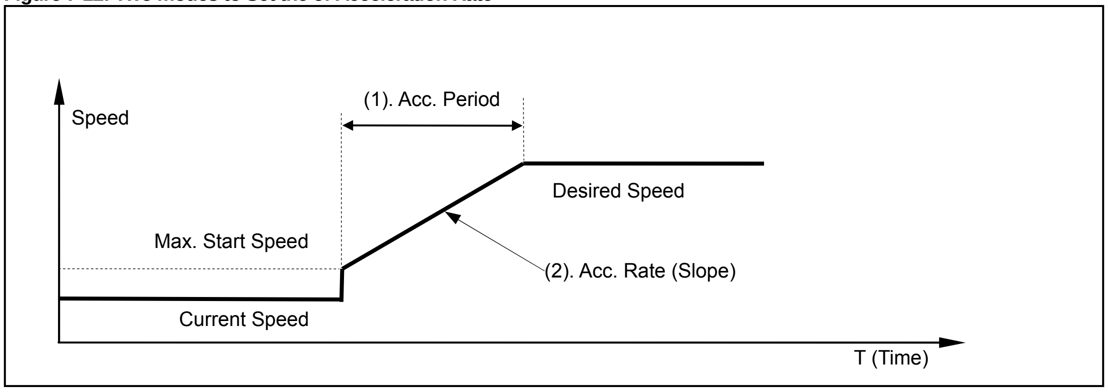
   a. Value mode

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.

b. 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.

Figure 7-12: Two modes to Set the of Acceleration Rate



3. mDEC

Similar to mACC, the deceleration also has two ways to set as listed below.

a. 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.

b. 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. mMSS

This is the instruction to set the Maximum Starting Speed.

5. mMDS

This is the instruction to set the Maximum Cessation Speed.

Since the definitions of Maximum Starting Speed and Maximum Cessation Speed were already discussed in the previous sections, here they are omitted. The unit of Maximum Starting Speed and Maximum Cessation Speed is pps (pulse per second).

7.8 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.

7.9 Acceleration Rate Setup Instruction (mACC)

Value Mode (pre-requiring MCFG<AM> = 0)

mACC= x;	
Function	Set the acceleration rate to x (in value mode).
Variable	Integer x = 1、2 … 65,000,000
ACK	0xAA [Controller ID] 0xB1 ACF AC4 AC3 AC2 AC1 AC0 0xFF
Comment	0xB1 is the message ID of mACC. AC4 – AC0 represents the value of the acceleration rate. See figure 2-2 for how to convert to an unsigned 32bit integer. ACF = the AM bit of the MCFG (here always =0). ACF = 0 means the input value will be interpreted as the acceleration rate with the unit of pps/s (or pulse/square-second). mACC is the abbreviation of “motion Acceleration”.

Period Mode (pre-requiring MCFG<AM> = 1)

mACC= x;	
Function	Set the acceleration period to x (in period mode)
Variable	Integer x = 1、2 … 60,000
ACK	0xAA [Controller ID] 0xB1 ACF AC4 AC3 AC2 AC1 AC0 0xFF
Comment	0xB1 is the message ID of mACC. AC4 – AC0 represents the value of the acceleration period. See figure 2-2 for how to convert to an unsigned 32bit integer. ACF = the AM bit of the MCFG (here always =1). ACF = 1 means the input value will be interpreted as period of acceleration with the unit of milliseconds.

Check the Current Acceleration Rate

mACC;	
Function	Check current acceleration rate.
Variable	N/A
Feedback	0xAA [Controller ID] 0xB1 ACF AC4 AC3 AC2 AC1 AC0 0xFF
Comment	See comments in above two modes.

7.10 Deceleration Rate Setup Instruction (mDEC)

Value Mode (pre-requiring MCFG<DM> = 0)

mDEC= x;	
Function	Set the deceleration rate to x (in value mode).
Variable	Integer x = 1、2 … 65,000,000
ACK	0xAA [Controller ID] 0xB2 DCF DC4 DC3 DC2 DC1 DC0 0xFF
Comment	0xB2 is the Message ID of mDEC. DC4 – DC0 represents the value of the deceleration rate. See figure 2-2 for how to convert to an unsigned 32bit integer. DCF = the DM bit of the MCFG (here always =0). DCF = 0 means the input value will be interpreted as the deceleration rate with the unit of pps/s (or pulse/square-second). mDEC is the abbreviation of “motion Deceleration”.

Period Method (pre-requiring MCFG<DM> = 1)

mDEC=x;	
Function	Set the deceleration rate to x (in period mode).
Variable	integer x = 1、2 … 60,000
ACK	0xAA [Controller ID] 0xB2 DCF DC4 DC3 DC2 DC1 DC0 0xFF
Comment	0xB2 is the message ID of mDEC. DC4 – DC0 represents the value of the deceleration period. See figure 2-2 for how to convert to an unsigned 32bit integer. DCF = the DM bit of the MCFG (here always =1). DCF = 1 means the input value will be interpreted as period of deceleration with the unit of milliseconds.

Check the Current Deceleration Rate

mDEC;	
Function	Check current deceleration rate.
Variable	N/A
ACK	0xAA [Controller ID] 0xB2 DCF DC4 DC3 DC2 DC1 DC0 0xFF
Comment	See comments in above two modes.

7.11 Maximum Starting Speed Setup Instruction (mMSS)

Set the Maximum Starting Speed

mMSS=x;	
Function	Set the Maximum Starting Speed at x.
Variable	Integer x = 1、2 … 65,000
ACK	0xAA [Controller ID] 0xB3 MS2 MS1 MS0 0xFF
Comment	0xB3 is message ID of mMSS MS2 – MS0 represents the value of Maximum Starting Speed. See figure 2-1 for how to convert to an unsigned 16bit integer. mMSS is the abbreviation of “motion Maximum Starting Speed”. Unit: pps (pulse/second).

Check current Maximum Starting Speed

mMSS;	
Function	Check the Maximum Starting Speed
Variable	N/A
ACK	0xAA [Controller ID] 0xB3 MS2 MS1 MS0 0xFF
Comment	See comments in above table.

7.12 Maximum Cessation Speed Setup Instruction (mMDS)

Set the Maximum Cessation Speed

mMDS=x;	
Function	Set the Maximum Cessation Speed at x.
Variable	Integer x = 1、2 … 65,000
ACK	0xAA [Controller ID] 0xB3 MD2 MD1 MD0 0xFF
Comment	0xB4 is the message ID for mMDS. MD2 – MD0 represents the value of Maximum Cessation Speed. See figure 2-1 for how to convert to an unsigned 16bit integer. mMDS is the abbreviation of “motion Maximum Deceleration Speed”.(mMCS is not used to avoid confusing with the micro stepping instruction MCS.) Unit: pps (pulse/second).

Check current Maximum Cessation Speed

mMDS;	
Function	Check the Maximum Cessation Speed.
Variable	N/A
ACK	0xAA [Controller ID] 0xB3 MD2 MD1 MD0 0xFF
Comment	See comments in above table.

8.0 Sensor Input Control

UIM242XX Motion Controller has an optional (sold separately) Sensor Control Module which supports two sensor input ports: S1 and S2. Both sensor input ports accept digital TTL input from 0V-5V. Furthermore, port S1 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 for S1,S2and S3, as listed below:

Table 8-1: 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

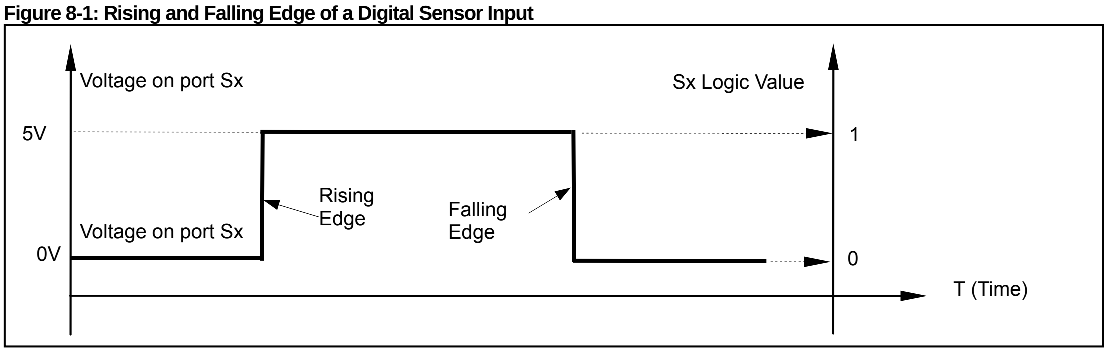
There are 9 actions that can be furthermore bound to sensor events:

1. Start and run backward (DIR=0) at desired speed and acceleration rate
2. Start and run forward (DIR=1) at desired speed and acceleration rate
3. Decelerate at the desired rate until stop
4. Reset position counter + decelerate at the desired rate until stop
5. Emergency stop.
6. Reset position counter + Emergency stop.
7. Execute backward displacement control (DIR=0) using user preset motion parameters.
8. Execute forward displacement control (DIR=1) using user preset motion parameters.
9. Reset position counter

8.1 Rising and Falling Edge

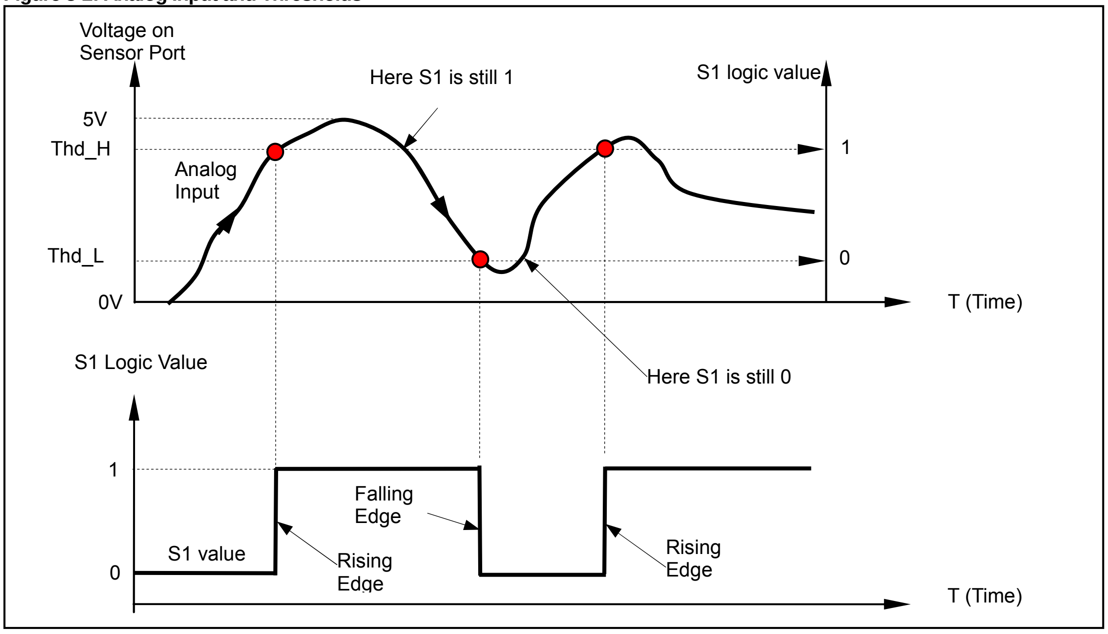
When port Sx (x=1, 2) is configured for digital input, if the sensor module detects a voltage change on Sx from 0V to 5V, an Sx rising-edge event will be created, meanwhile Sx is assigned a logic value 1 (i.e. Sx=1). If the sensor module detects a change on Sx from 5V to 0V, an Sx falling-edge event will be created, meanwhile Sx=0.

Figure 8-1: Rising and Falling Edge of a Digital Sensor Input



8.2 Analog Input and Thresholds

Figure 8-2: 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 sFBK.

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 Sensor Event, Action and Binding

A sensor event is defined as the sensor voltage change matches a user-defined condition. Binding means assigning a sensor action to a sensor event. UIM242XXs support 6 sensor events as listed in section 8.0. There are 9 actions that can be bound to those 6 sensor events. The binding between events and actions are realized through the configuration of the Sensor Control Register S12CON. These 9 actions are described below:

1. Start and Run Backward (DIR=0)

   Run Backwards means starting and continuously running the motor backward using motion parameters (i.e., SPD, ACC/DEC, MSS/MDS, etc.) stored in the EEPROM. (Motion direction is defined by the S12CON.) Before making usage of this action, user has to first configure the S12CON, setup the desired speed (SPD), and (if applicable) the acceleration rate, maximum starting speed, etc. After that, user has to burn the SPD and S12CON into the EEPROM using the STORE instruction.

2. Start and Run Forward (DIR=1)

   Same as above, except that the direction is opposite (forward instead of backward).

3. Deceleration until Stop

   Decelerating the motor speed until stop according to the motion parameters (i.e., mDEC, mMDS) stored in the EEPROM. To use this action, the advanced motion control must be enabled.

4. Absolute Position Counter Reset + Deceleration until Stop

   Absolute Position Counter will be reset when the sensor event happens and then deceleration process starts according to motion parameters (or advanced motion parameter) stored in EEPROM until stop.

5. Emergency Stop

   Set speed to zero immediately when the sensor event happens to force the motor stop.

6. Absolute Position Counter Reset + Emergency Stop

   Absolute Position Counter will be reset when the sensor event happens and set speed to zero.

7. Backward (DIR=0) Relative Displacement Control

   Control the motor to realize a backward displacement using motion parameters (i.e., SPD, STP, ACC/DEC, MSS/MDS, etc.) stored in the EEPROM. Before making usage of this action, user has to first configure the S12CON, setup the desired speed (SPD), the desired displacement (STP), and (if applicable) the acceleration rate, maximum starting speed, etc. After that, the user has to burn the parameters into the EEPROM using the STORE instruction.

8. Forward (DIR=1) Relative Displacement

   Same as above, except that the displacement control is forward instead of backward.

9. Absolute Position Counter Reset

   This action resets the absolute position counter to zero and creates a zero position or origin.

8.4 Introduction to Sensor Input Control Instructions

There are only 4 instructions related to the sensor input control.

1. MCFG

   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. SCFG (Sensor Configuration Register)

   SCFG is used to configure following sensor input control registers: S12CON and Analog threshold control register ATCONH and ATCONL.

3. STORE (Sensor Parameter Store into EEPROM)

   STORE 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.

4. sFBK (Sensor Status Feedback)

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

8.5 Sensor Input Control Register S12CON

S12CON (Sensor 1/2 Control) defines the binding relationship between S1 and 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 SCFG. When writing to it user needs to affix a 4bits suffix-code to point to this register. For details of SCFG, please refer to Section 8.7.

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
value	S2RACT				S2FACT				S1RACT				S1FACT			

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

S2RACT（binary）	**Action**	**RTCN or Not**
0000	N/A	No RTCN (Ignore MCFG<S2IE>).
0001	N/A	Depends on MCFG<S2IE>
0010	Start and Run Reversely	Depends on MCFG<S2IE>
1010	Start and Run Forwardly	Depends on MCFG<S2IE>
0011	Decelerate until Stop	Depends on MCFG<S2IE>
1011	Reset position and encoder counter + Decelerate until Stop	Depends on MCFG<S2IE>
0100	Emergency Stop	Depends on MCFG<S2IE>
1100	Reset position and encoder counter + Emergency Stop	Depends on MCFG<S2IE>
0101	Reverse Displacement Control	Depends on MCFG<S2IE>
1101	Forward Displacement Control	Depends on MCFG<S2IE>
0110	Reset position and encoder counter	Depends on MCFG<S2IE>

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

S2FACT（binary）	**Action**	**RTCN or Not**
0000	N/A	No RTCN (Ignore MCFG<S2IE>).
0001	N/A	Depends on MCFG<S2IE>
0010	Start and Run Reversely	Depends on MCFG<S2IE>
1010	Start and Run Forwardly	Depends on MCFG<S2IE>
0011	Decelerate until Stop	Depends on MCFG<S2IE>
1011	Reset position and encoder counter + Decelerate until Stop	Depends on MCFG<S2IE>
0100	Emergency Stop	Depends on MCFG<S2IE>
1100	Reset position and encoder counter + Emergency Stop	Depends on MCFG<S2IE>
0101	Reverse Displacement Control	Depends on MCFG<S2IE>
1101	Forward Displacement Control	Depends on MCFG<S2IE>
0110	Reset position and encoder counter	Depends on MCFG<S2IE>

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

S1RACT（binary）	**Action**	**RTCN or Not**
0000	N/A	No RTCN (Ignore MCFG<S1IE>).
0001	N/A	Depends on MCFG<S1IE>
0010	Start and Run Reversely	Depends on MCFG<S1IE>
1010	Start and Run Forwardly	Depends on MCFG<S1IE>
0011	Decelerate until Stop	Depends on MCFG<S1IE>
1011	Reset position and encoder counter + Decelerate until Stop	Depends on MCFG<S1IE>
0100	Emergency Stop	Depends on MCFG<S1IE>
1100	Reset position and encoder counter + Emergency Stop	Depends on MCFG<S1IE>
0101	Reverse Displacement Control	Depends on MCFG<S1IE>

1101	Forward Displacement Control	Depends on MCFG<S1IE>
0110	Reset position and encoder counter	Depends on MCFG<S1IE>

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

S1FACT（binary）	**Action**	**RTCN or Not**
0000	N/A	No RTCN (Ignore MCFG<S1IE>).
0001	N/A	Depends on MCFG<S1IE>
0010	Start and Run Reversely	Depends on MCFG<S1IE>
1010	Start and Run Forwardly	Depends on MCFG<S1IE>
0011	Decelerate until Stop	Depends on MCFG<S1IE>
1011	Reset position and encoder counter + Decelerate until Stop	Depends on MCFG<S1IE>
0100	Emergency Stop	Depends on MCFG<S1IE>
1100	Reset position and encoder counter + Emergency Stop	Depends on MCFG<S1IE>
0101	Reverse Displacement Control	Depends on MCFG<S1IE>
1101	Forward Displacement Control	Depends on MCFG<S1IE>
0110	Reset position and encoder counter	Depends on MCFG<S1IE>

8.6 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 SCFG. When writing to it user needs to affix a 4bits suffix-code to point to this register. For details of SCFG, please refer to chapter 8.8.

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.

S34CON Structure

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value	x				P4LVL	P4EVENT			S3RACT				S3FACT			

Bit 15-12 Unimplemented. Read as 0

Bit 11-8 **P4LVL : P4EVENT<2:0>** P4 TTL Output Control

Please refer to section 10.2 for details.

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

S3RACT（binary）	**Action**	**RTCN or Not**
0000	N/A	No RTCN (Ignore MCFG<S3IE>).
0001	N/A	Depends on MCFG<S3IE>
0010	Start and Run Reversely	Depends on MCFG<S3IE>
1010	Start and Run Forwardly	Depends on MCFG<S3IE>
0011	Decelerate until Stop	Depends on MCFG<S3IE>
1011	Reset position and encoder counter + Decelerate until Stop	Depends on MCFG<S3IE>
0100	Emergency Stop	Depends on MCFG<S3IE>
1100	Reset position and encoder counter + Emergency Stop	Depends on MCFG<S3IE>
0101	Reverse Displacement Control	Depends on MCFG<S3IE>
1101	Forward Displacement Control	Depends on MCFG<S3IE>
0110	Reset position and encoder counter	Depends on MCFG<S3IE>

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

S3FACT（binary）	Action	RTCN or Not
0000	N/A	No RTCN (Ignore MCFG<S3IE>).
0001	N/A	Depends on MCFG<S3IE>
0010	Start and Run Reversely	Depends on MCFG<S3IE>
1010	Start and Run Forwardly	Depends on MCFG<S3IE>
0011	Decelerate until Stop	Depends on MCFG<S3IE>
1011	Reset position and encoder counter + Decelerate until Stop	Depends on MCFG<S3IE>
0100	Emergency Stop	Depends on MCFG<S3IE>
1100	Reset position and encoder counter + Emergency Stop	Depends on MCFG<S3IE>
0101	Reverse Displacement Control	Depends on MCFG<S3IE>
1101	Forward Displacement Control	Depends on MCFG<S3IE>
0110	Reset position and encoder counter	Depends on MCFG<S3IE>

8.7 Analog Threshold Control Register ATCON & ATCONL

ATCONH and ATCONL define the upper and lower limit of the analog threshold.

Both registers are 16bits registers in the controller memory space, configured through SCFG 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
value	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
value	Reserved				AL <11:0>											

Bit 15-12 Unimplemented, read as 0.

Bit 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).

8.8 Sensor Configuration Instruction (SCFG)

SCFG=x；	
Function	Configure the S12CON, S34CON, ATCONH and ATCONL.
Variable	Integer x = 0,1 … 1048575, or hexadecimal x=0x00000 … 0xFFFFF
ACK	0xAA [Controller ID] 0xC0 S4 S3 S2 S1 S0 AL1 AL0 AH1 AH0 0xFF
Comment	0xC0 is the message ID of SCFG. S4 – S0 represent the value of [S34CON:S12CON]. See figure 2-2 for how to convert an unsigned 32bit integer. The higher 16 bits of the 32bit integer represent S34CON and the lower 16 bits represent S12CON. AL1 – AL0 represent the value of the lower limit of analog input. AH1 – AH0 represent the value of the upper limit of the analog input. See figure 2-1 for how to convert above bytes to unsigned 16bit integers. S34CON, S12CON, ATCONH and ATCONL are 16bits registers in the controller. But when configuring them using SCFG, user has to affix a 4bits suffix code to specify the register to be written. The suffix code for S12CON is 0000 (binary); The suffix code for S34CON is 0001 (binary); The suffix code for ATCONL is 0010 (binary); The suffix code for ATCONH is 0011(binary).

8.9 Check the Value of S12CON, S34CON, ATCONH and ATCONL

SCFG；	
Function	To check the current value of S12CON, S34CON, ATCONH and ATCONL
Variable	N/A
ACK	0xAA [Controller ID] 0xC0 S4 S3 S2 S1 S0 AL1 AL0 AH1 AH0 0xFF
Comment	See comments in above table.

8.10 EEPROM Store Instruction (STORE)

STORE instruction is used to burn the values of Sensor Control Configuration, Analog Thresholds, desired speed, and desired displacement into the EEPROM so that the Sensor Input Module automatically participates in system control when user device is absent. STORE instruction will affect the system’s real-time performance.

STORE;	
Function	Burn MCFG, sensor CFG, motion control parameters into EEPROM.
Variable	N/A
ACK	0xAA [Controller ID] 0XD1 0xFF
Comment	0xD1 is the Message ID of STORE. STORE is used to burn MCFG, sensor configuration, speed, displacement, acceleration/deceleration rate, etc., into EEPROM STORE 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.

8.11 Sensor Data Inquiry Instruction (SFBK)

SFBK；	
Function	Check sensor readings and status.
Variable	N/A
ACK	0xCC [Controller ID] 0xC1 D1 D2 D3 AN1 AN0 0xFF
Comment	0xC1 is the message ID of SFBK. D1, D2 and D3 represent the logic level of S1, S2 and S3 respectively (0/1). AN1 – AN0 is the converted value for analog input (12 bits). See figure 2-1 for how to convert above bytes to an unsigned 16bit integer. AN1 and AN0 are 0 if no analog input port is configured. This instruction can be used for sensor data inquiry at any time and under any condition.

8.12 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). Then, user can use the instruction SCFG to realize the configuration. An example is provided below.

Example 8.11：

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. Otherwise, a 5V presents.

Requirements:

1. As soon as one sensor S2 is hit, the stepper motor starts to run reversely (DIR=0) until the table hits the other sensor S1.
2. As soon as S1 is hit, the stepper motor starts to run positively (DIR=1), until the table hits the S2.
3. Keep the reciprocating motion without the user control device.

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 1, 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) = 0x020A0 (hex) = 8352 (decimal)
8. Send instruction: **SCFG = 0x20A0;** or **SCFG = 8352;**
9. Set up desired speed, by sending instruction: **SPD=5000;**
10. Burn parameters into EEPROM, by sending: **STORE;**
11. Press any one of the limit sensors, the mobile platform will work.
12. If user enables the RTCNs, the user device will get feedback every time the S1 or S2 is hit.
13. Disconnect the user device, and restart the UIM242 controller, the system will automatically run.

8.13 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.

Example 8.12：

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 (DIR=1) 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.
3. Keep the reciprocating motion without the user control device.

Realization:

1. First stop the motor by sending: **OFF;**
2. Set MCFG<ANE>=1, MCFG<CHS> =0, get:

   MCFG = 1000 0000 0000 0001 (binary) = 0x8001 (hex) = 32769 (decimal)
3. Send instruction: **MCFG = 0x8001;** or **MCFG = 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) = 0x002A0 (hex) = 672 (decimal)
8. Send instruction: **SCFG = 0x2A0;** or **SCFG = 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) = 0x0CCC3 (hex) = 52419(decimal)
11. Send instruction: **SCFG = 0xCCC3;** or **SCFG = 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) = 0x01EB2 (hex) = 7858 (decimal)
14. Send instruction: **SCFG = 0x1EB2;** or **SCFG = 7858;**
15. Set desired speed, by sending instruction: **SPD=5000;**
16. Burn parameters into EEPROM, by sending: **STORE;**
17. Initiate the motion by sending: **ENABLE;**
18. The system starts to work continuously.
19. Disconnect the user device, and restart the UIM242 controller, the system will automatically run.

9.0 Encoder and Closed-loop Control

Quadrature Encoder (also known as Incremental Encoder or Optical Encoder) is used for tracking the angular position and velocity of rotary motion. It can be applied for closed-loop control of various motors. A typical quadrature encoder consists of a slotted wheel for motor shaft and a transmitter/detection module for detection of the slot on the wheel. Usually there are 3 channels - channels A, B and Z (INDEX). Information from the three channels can be read and decoded to provide motion status of shaft, including position and velocity.

The relationship between channel A (QEA) and channel B (QEB) is as simple as which phase leads. When phase A leads B, then the shaft is rotating in the clockwise direction. When phase B leads A, then the shaft is rotating in the counter-clockwise direction. Channel Z is called index pulse which is generated per revolution as a reference for tracking of absolute position.

The quadrature signals from encoder can be decoded into four types of messages, the order of which reverse when rotation direction is reversed. The phase signals and index pulses are detected by encoder and further decoded to produce a count up pulse (for one direction of shaft rotation) or a countdown pulse (for the other direction of shaft rotation).

UIM242 controller has a built-in quadrature encoder (hereinafter referred to as encoder) interface circuit, which is capable of decoding encoder signals of less than 200KHz input. Another option is user can connect external encoder of their own choice to UIM242 controller, using S1 and S2 ports for channel A and B. In this case, however, INDEX decoding function is not available. S1/S2 supports 0-5V TTL input. The input range for S1 and S2 ports of UIM242 controller is -0.3V ~ 5.3V. Any input beyond this range can result in permanent damage. Also, for this case, encoder power supply is to be provided by user.

For UIROBOT UIM242 controller with internal encoder, the S1 and S2 ports are not occupied and therefore are available for sensors. Whether the encoder is built-in or external, the controlling mode and the instructions are the same.

Instructions relative to encoder control function are listed below:

	Instruction	**Function**	**Example**
1	MCFG	enable encoder function	MCFG1792;
2	QEC	encoder-based position control	QEC= - 200000;
3	STP	encoder-based displacement control	STP500;
4	QER	set encoder resolution	QER=500;

9.1 Enable/Disable Encoder and Closed-loop Control Module (MCFG)

Enable Encoder Interface

The Encoder Decoding Module is enabled / disabled through configuring the QEI bit of MCFG (MCFG<13>). When MCFG<QEI>=0, the encoder decoding module is disabled; when MCFG< QEI>=1, the encoder decoding module is enabled. If external encoder is used, S1 and S2 ports must be used for channel A and channel B respectively. If user chooses UIROBOT internal encoder, S1 and S2 ports are available for sensors. Please note encoder interface is a standard module which is available as long as Sensor Input Module is installed.

Enable Closed-loop Control Module

The Encoder-based Closed-loop Control Module (hereinafter referred to as Closed-loop Control Module) is enabled by configuring the QEM bit of MCFG (MCFG<11>). When MCFG<QEM>=0, this module is disabled; when MCFG<QEI>=1, it is enabled.

Please note, closed-loop control module is a must even if user uses external encoders. Otherwise, UIM242 controller can only read the external encoder data, but cannot maintain closed-loop motion control with this data. However, if the internal encoder is installed, Closed-loop Control Module is automatically included.

For master configuration register (MCFG), please refer to Section 5.1.

9.2 Closed-loop Position Control Instruction (QEC)

QEC=x;	
Function	Set desired encoder position to x (for closed-loop control).
Variable	Integer x = - 2,000,000,000…-1, 0, 1 … + 2,000,000,000
ACK	0xAA [Controller ID] 0xB8 Q4 Q3 Q2 Q1 Q0 0xFF
Comment	0xB8 is the message ID of desired encoder position (QEC). Q4-Q0 represents the desired quadrature encoder position. See figure 2-2 for how to convert to a signed 32bit integer. Actual motor position is also relative to the encoder resolution. The encoder counter records encoder pulses. When the direction is positive (DIR=1), the counter increases; when the direction is negative (DIR=0), the counter decreases. Therefore, the value of the counter is a signed 32bits integer, with positive representing the final position is of the same direction of DIR=1, and vice versa. Encoder counter can only be reset/cleared under following situations: 1. Commanded by user instruction ORG 2. User preset sensor ORG event happens Please also be aware: 1. Power Failure Protection. Should a Power Failure situation happen, the value of the encoder counter will be pushed into EEPROM and restored when reboot next time. However, passive movement after power off cannot be recorded. 2. For every slot, the encoder counter records 4 pulses. E.g., when QEC=500, the encoder counter records 500*4 = 2000 pulses each turn. QEC is the abbreviation for Quadrature Encoder Counter.

QEC instruction is basically of the same use as POS.

The difference is that POS is for open-loop control while QEC is for closed-loop control. When closed-loop control module is enabled (MCFG<QEM> = 1), QEC instruction can be used; however, a POS instruction can only leads to an error ACK (except when it is used for status inquiry). On the other hand, in open-loop control, POS instruction can be used while QEC instruction can only be used for status inquiry (provided that an encoder is included in the system whose QER is correctly configured and the Encoder Decoding Module is enabled, i.e. MCFG<QEI> = 1).

9.3 Check Current Encoder Position

QEC;	
Function	to check current encoder position.
Variable	N/A
Feedback	0xCC [Controller ID] 0xB1 Q4 Q3 Q2 Q1 Q0 0xFF
Comment	0xB1 is the message ID of current encoder position (QEC). Q4-Q0 represents the desired quadrature encoder position. See figure 2-2 for how to convert to a signed 32bit integer.

9.4 Quadrature Encoder Resolution Setting Instruction (QER)

QER=x;	
Function	to set the quadrature encoder resolution at x
Variable	Integer x=0, 1 … 65000
ACK	0xAA [Controller ID] 0xC2 R2 R1 R0 0xFF
Comment	0xC2 is the message ID of QER. R2-R0 represents encoder resolution. See figure 2-1 for how to convert to an unsigned 16bit integer. QER is the abbreviation for Quadrature Encoder Resolution.

⚠ **WARNING:** Incorrect QER value can result in unpredictable closed-loop control operations!

9.5 Check Quadrature Encoder Resolution

QER;	
Function	to check current quadrature encoder resolution
Variable	N/A
ACK	0xAA [Controller ID] 0xC2 R2 R1 R0 0xFF
Comment	0xC2 is the message ID for QER. R2-R0 represents encoder resolution. See figure 2-1 for how to convert to an unsigned 16bit integer.

9.6 Duality of STP Instruction

When closed-loop control module is enabled (MCFG<QEM>=1), STP=x defines encoder-based relative position instead of relative pulse. On the contrary when this module is disabled, STP=x defines relative pulse.

9.7 SPD Instruction Definition

Whether closed-loop control module is enabled or not, SPD=x; defines pulses sent to motor per second.

9.8 Restrictions on POS Instruction

In the closed-loop control mode, an instruction of “POS=x” will generate an error ACK, but the instruction “POS;” can be used to check the current pulses accumulated since the origin point was set (increases for positive running; decrease for reverse running).

Similarly, in open-loop control mode, an instruction of “QEC=x” will generate an error ACK, but the instruction “QEC;” can be used to check the quadrature encoder pulses accumulated since the origin point was set (increases for positive running; decrease for reverse running).

10.0 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.

The output voltage level can be controlled by:

1. User instruction, or
2. One of the following events:
   a) Run/Stop status. The output voltage level is determined by if the speed is zero or not.
   b) Direction change. The output voltage level is determined by if the current motor direction is forward or reverse.
   c) Origin point hit. The output voltage level is determined by if current position is zero point or just crosses over the zero point.

10.1 Introduction to TTL Output Control Instructions

There are 3 instructions related to the TTL output control.

1. MCFG

   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. SCFG

   The instruction SCFG is used to configure the register S34CON

3. DOUT

   This instruction is used to directly control the TTL output voltage level as well as check current voltage level.

10.2 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 SCFG. 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
value	x				P4LVL	P4EVENT			S3RACT				S3FACT			

Bit 15-12 Unimplemented. Read as 0.

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 DIR =1, 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 8.6 for more information.

10.3 Output Control Configuration Instruction (SCFG)

Please refer to chapter 8 for detailed information.

10.4 TTL Output Instruction (DOUT)

DOUT=x；	
Function	Set TTL output level.
Variable	Integer x = 0,1
ACK	0xAA [controller ID] 0xC1 P4 0xFF
Comment	0xC1 is the message ID of DOUT. P4 is the logic level of the TTL output. P4=1 means the output is 5V, while P4=0 means the output is 0V. DOUT is the abbreviation of “Digital Output”. When DOUT = 1, TTL output is 5V. When DOUT = 0, TTL output is 0V. **NOTE**: Using DOUT=x; will affect S34CON. Once DOUT instruction is received, 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.

10.5 Check TTL Output Level

DOUT;	
Function	To check current TTL output level
Variable	N/A
ACK	0xAA [controller ID] 0xC1 P4 0xFF
Comment	0xC1 is the message ID of DOUT. P4 is the logic level of the TTL output. P4=1 means the output is 5V, while P4=0 means the output is 0V. Using DOUT; will NOT affect S34CON.

10.6 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 the motion DIR=1, P4 outputs 5V.
2. When the motion DIR=0, P4 outputs 0V.
3. Need RTCN every time P4 changes.

Realization

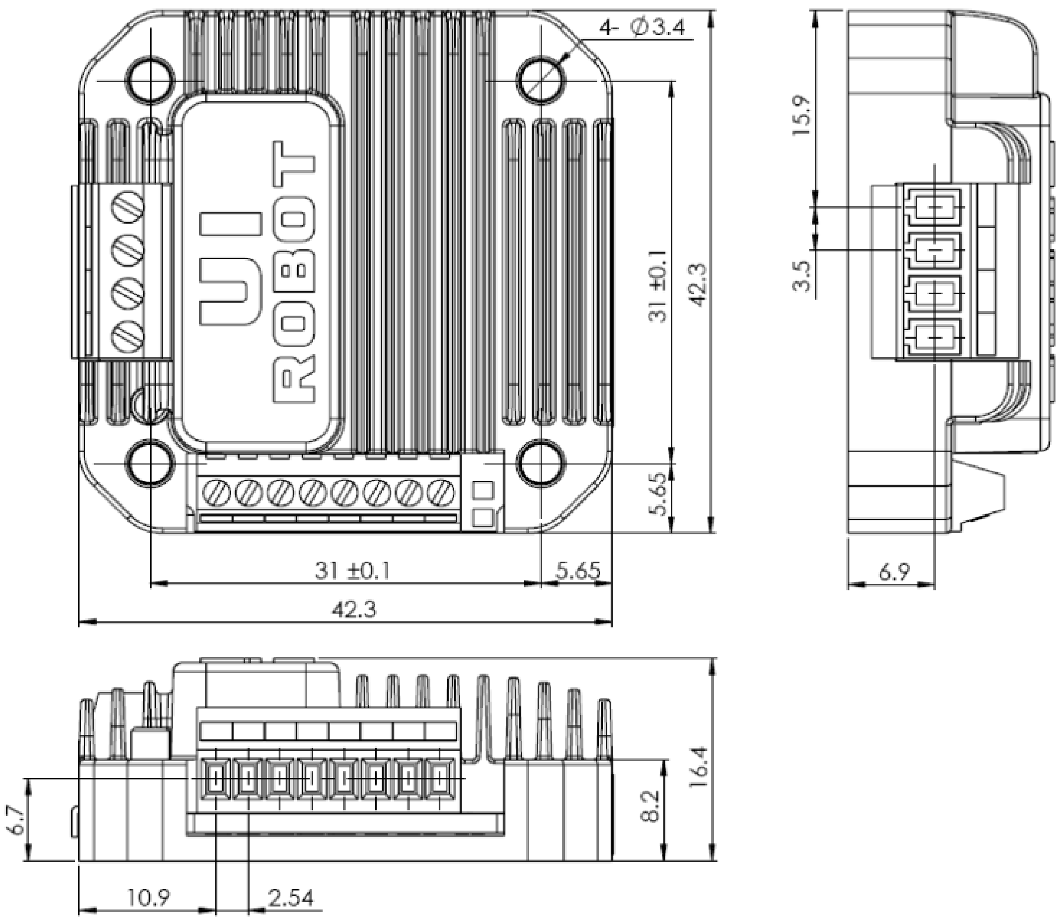
1. First stop the motor by sending: **OFF;**
2. Set MCFG<P4IE>=1, get:

   MCFG = 0000 0000 0000 1000 (binary) = 0x0008 (hex) = 8 (decimal)
3. Send instruction: **MCFG = 8;**
4. Set P4EVENT <2:0>=010, link to direction event.
5. Set P4LVL=1, so when DIR=1, 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) = 0x0A001 (hex) = 40961(decimal)
8. Send instruction: **SCFG = 0xA001;** or **SCFG = 40961;**
9. Send instruction: **ENA;**
10. Run the motor. There are numerous ways to run the motor. The easiest way is using **SPD=x;**

During the motion, please watch the output level change each time the motor change direction, and pay attention to the RTCN on the PC screen.

APPENDIX A Dimensions

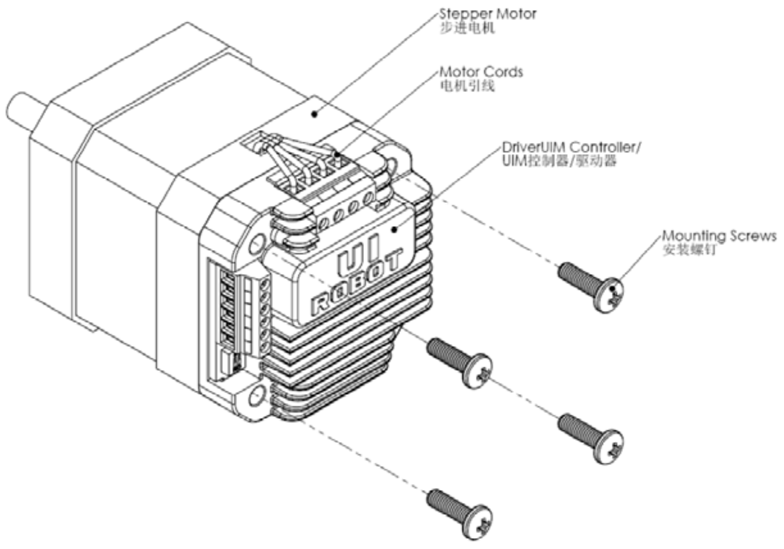


Unit：mm

APPENDIX B Installation

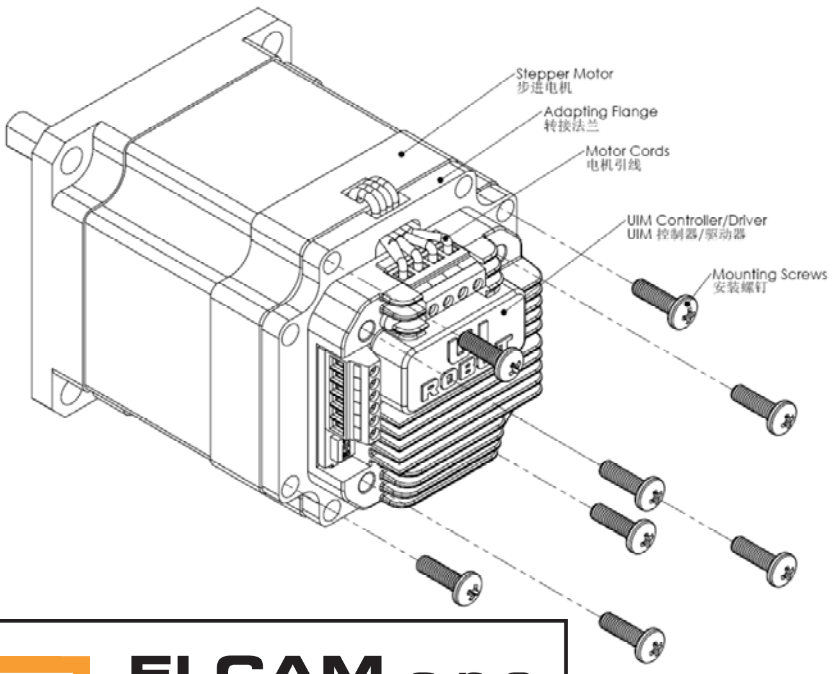
NEMA 17 Stepper Installation (without adapting flange)

1. Screw mount UIM controller / driver onto the motor
2. Wire the motor leads.



NEMA 23 and Larger Stepper Installation (with adapting flange)

1. Screw mount the adapting flange onto the motor
2. Screw mount UIM controller / driver onto the adapting flange
3. Wire the motor leads.


Stepper Motor
步进电机
Adapting Flange
转接法兰
Motor Cords
电机引线
UIM Controller/Driver
UIM 控制器/驱动器
Mounting Screws
安装螺钉