Mach Motion Products

User Manual

UIM241XX series
RS232 Interface
Integrated
Miniature Stepper Motor Motion Controller

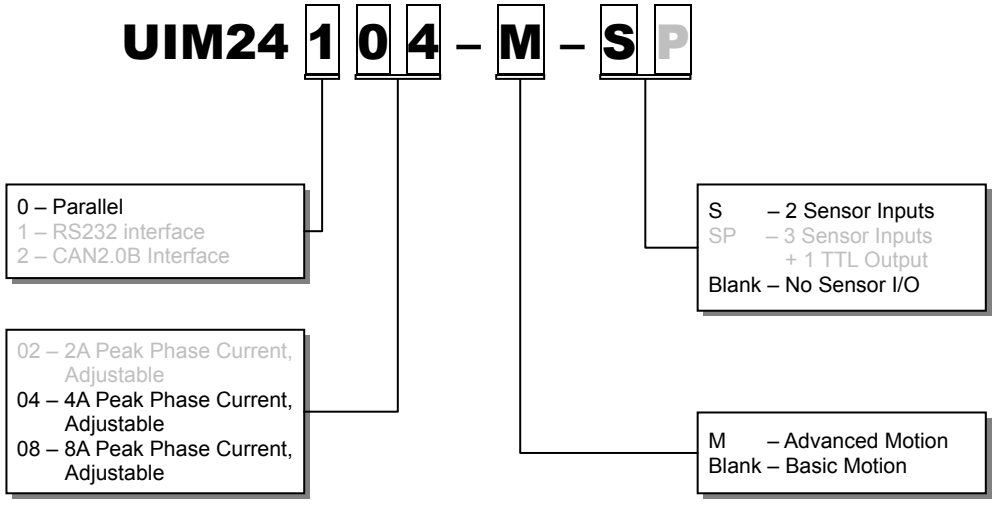
Please pay attention to the following before using the Mach Motion Products:

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

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in this publication regarding controller applications and the like is provided only for your convenience and may be superseded by updates.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

[UIM241XX Order Information]

In order to serve you quicker and better, when order UIM240XX series controllers, please provide the product part number in the following format.



Note: Options in gray is not applicable.

Examples:

UIM24104
UIM24104-M
UIM24104-S
UIM24104-M-S

UIM24104 / UIM24108
RS232 Interface
Miniature Integrated Stepper Motor Motion Controller

Features

Miniature Integral Design

- Miniature size 42.3mm*42.3mm*13.5mm
- Fit onto motors seamlessly
- Die-cast aluminum enclosure, improving heat transfer and durability

Motor Drive Characteristics

- Wide supply voltage range 12 ~ 40VDC
- Output current 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, 6600 RPM max speed for NEMA 17 motor (half-step) and 4000 RPM for NEMA 17/23 motor (quad-step).

RS232 Interface

- RS232 three-wire serial communication
- Max baud rate 57600 bps

Embedded DSP Microprocessor

- Embedded 64-bit calculation precision high-performance digital signal processing micro controller
- Absolute position counter, reset by instruction or sensor input
- Advanced motion control module, from 0 to 4000 RPM in 0.25s and from 0 to 6600 RPM in 0.5s (for NEMA17/23)
- 2 sensor input ports, can be configured as 2 digital or 1digital 1 analog input ports （12bit）
- 8 programmable real-time event-based change notifications (similar to interrupters)
- 5 programmable actions triggered by 6 sensor events
- Simple, intuitive, rich instructions
- User-friendly interface

Description

UIM24104 and UIM24108 are miniature stepper motor controllers with RS232 interface. User device can command these controllers through RS232 using ASCII coded instructions. Instructions are simple, intuitive and fault-tolerating. User is not required to have advance knowledge on stepper motor driving.

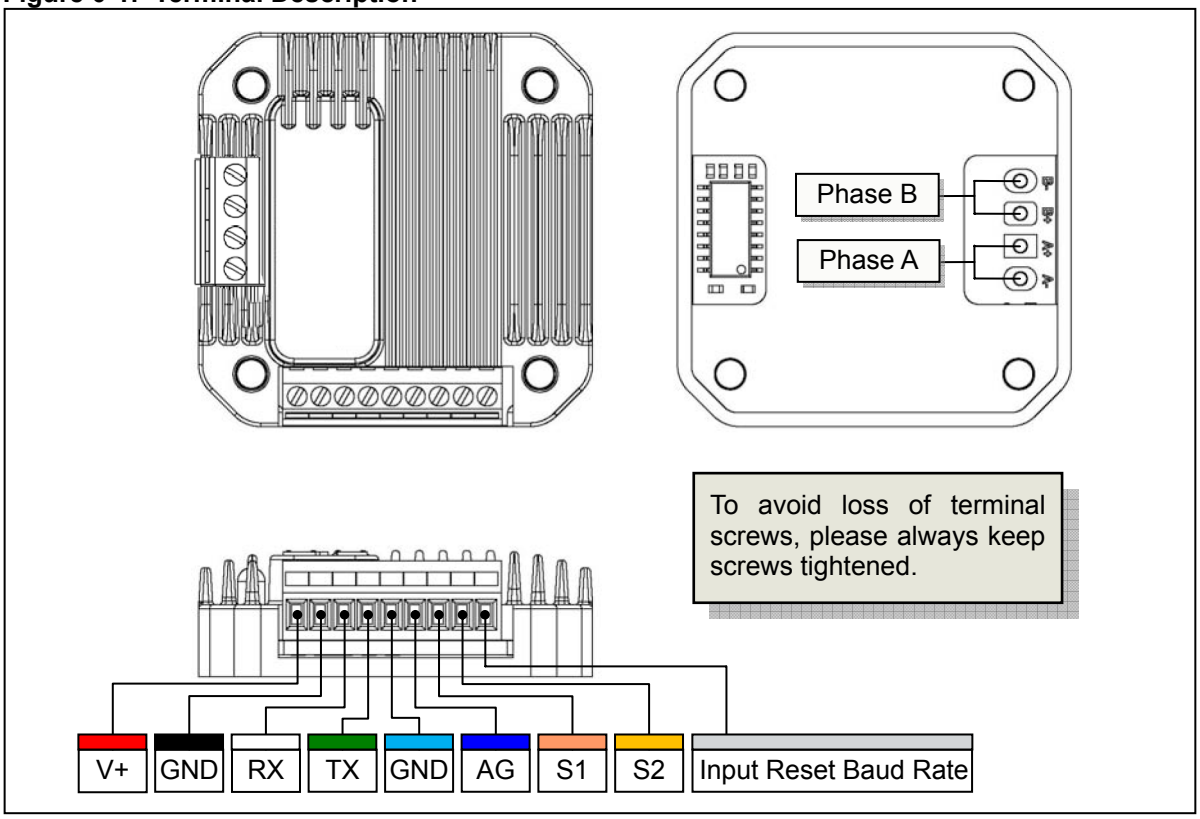
UIM241 controller's architecture comprises communication system, basic motion control system, absolute position counter, and real-time event-based change notification system. Embedded 64-bit calculation precision DSP controller guarantees the real-time processing of the motion control and change notifications.

There are 2 optional advanced modules for selection: Advanced Motion Control Module and Input Control Module. With UIM241's Advanced Motion Control, selected NEMA 17/23 motors can ramp up to 6600 RPM in 0.5 seconds and 4000 RPM in 0.25 seconds (online video available for downloading).

UIM241 controllers can be mounted onto NEMA17/23/34/42 series stepper motor through adapting flanges. Controller thickness is less than 14 mm. 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



Description of Screw Terminals

Terminal No. / Color	Description		Input / Output			
			MIN	NOM	MAX	UNIT
1 / Red	V+	Supply voltage	12		40	VDC
2 / Black	GND	Supply voltage ground		0		VDC
3 / White	RX	To the RX pin on user device(1)	-25		+25	VDC
4 / Green	TX	To the TX pin on user device(1)	-13.2		+13.2	VDC
5 / Cyan	GND	To signal ground on user device(2)		0		VDC
6 / Blue	AG	Analog Ground for Sensor(2)	0		5	VDC
7 / Pink	S1	Sensor 1 Input	0		5	VDC
8 / Yellow	S2	Sensor 2 Input	0		5	VDC
9 / Gray	Input Reset Baud Rate		0		5	VDC

Note:
(1) Please refer to "Typical Application" section for details.
(2) Internally linked to supply voltage ground

Motor Wiring Pads (at the bottom of the controller)

Pad A+ / A- : Connect to the stepper motor phase A

Pad B+ / B- : Connect to the stepper motor phase B

WARNING: To avoid damaging, make sure the phase winds are connected correctly. Resistance between leads of different phases is usually > 100KΩ. Resistance between leads of the same phase is usually < 100Ω.

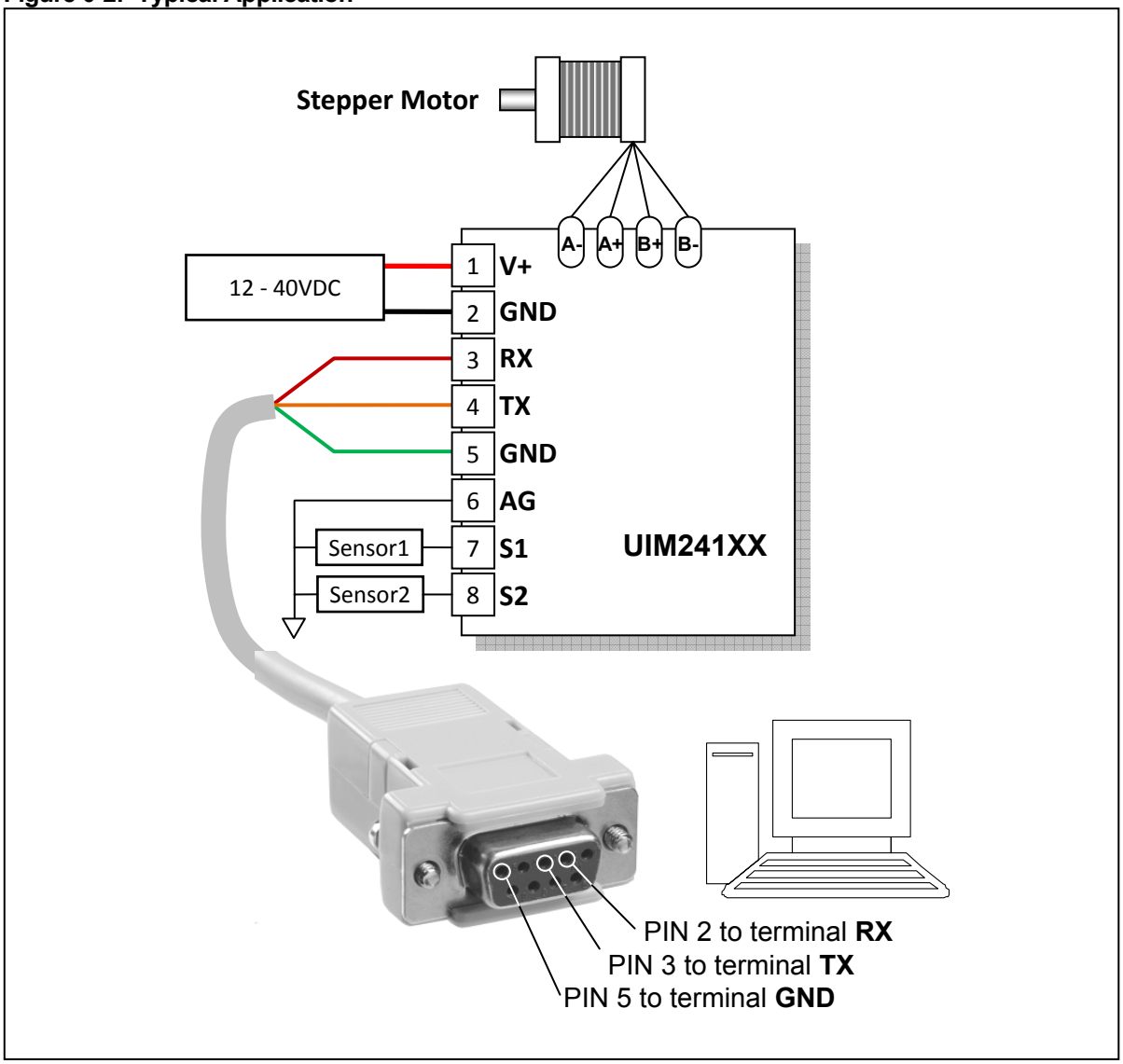
Typical Application

UIM241xx controllers use 3-wire RS232 interface to communicate with user devices. Terminal 3 should be connected to the RX pin on user device (e.g. pin 2 on a DB9 connector or pin 3 on a DB25 connector); Terminal 4 should be connected to the TX pin on user device (e.g. pin 3 on a DB9 connector or pin 2 on a DB25 connector); Terminal 5 should be connected to the GND pin on user device (e.g. pin 5 on a DB9 connector or pin 7 on a DB25 connector). An example for wiring using DB9 connector is provided in the following figure.

Notice: When wiring sensors,

1. User is responsible for the power supply of the sensor.
2. Voltage on terminal 7 and 8 must be kept between 5.3V and -0.3V.

Figure 0-2: Typical Application



Characteristics

Absolute Maximum Ratings (†)

Supply Voltage..... 10V to 40V
Voltage on S1/S2 with respect to AG..... -0.3V to 5.3V
Maximum output current sunk by S1/S2.....20 mA
Maximum output current sourced by S1/S2.....20 mA
Voltage on RX with respect to GND ..... -25V to +25V
Voltage on TX with respect to GND..... -13.2V to +13.2V
Ambient temperature under bias……..... -20°C to +85°C
Storage temperature..... -50°C to +150°C

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

Electrical Characteristics（Ambient Temperature 25℃）

Supply Power Voltage	12V ～ 40VDC
Motor Output Current	Max 4A/8A per phase (Adjustable through instruction）
Driving Mode	PWM constant current
Stepping Resolution	full-step, half-step, quarter-step, and sixteenth-step
Insulation Resistance	>100MΩ
Dielectric Strength	0.5KV in one minute

Communication (Ambient Temperature 25℃）

Communication protocol	RS232
Wiring Method	3-wire:TX, RX, GND
Baud Rate	MAX 57600 bps, Instruction adjustable

Environment Requirements

Cooling		Free Air
Working	Environment	Avoid dust, oil mist and corrosive gases
	Temperature	-20 ℃ ～+ 85 ℃
	Humidity	<80%RH，no condensation, no frosting
	Vibration	3G Max
Storage Temperature		-50 ℃ ～+ 150 ℃

Size and Weight

Size	42.3mm x 42.3mm x 13.5mm (L*W*H)
Wight	0.1 kg

Table of Contents

OVERVIEW

1.0 Overview

UIM24104/UIM24108 Miniature stepper motor controller communicates with the user devices through RS232 protocol. The controller has a size of less than 42.3 x 42.3 x 13.5 mm, and is designed to mount onto NEMA 17/23/34/42 series stepper motor seamlessly through adapting flanges.

UIM24104 and UIM24108 are capable of providing 1.5~4A and 3~8A adjustable output current. Output current can be adjusted through instruction in a real time fashion. Once set, the value is stored in the onboard non-volatile memory. The controller also has high-speed current compensation to facilitate the motor's high speed performance. This series of controllers can work with 12 ~ 40VDC power supply.

UIM241XX controller's basic architecture includes communication system, basic motion control system, and absolute position counter and real-time event-based change notification system. UIM241XX's advanced add-on modules include advanced motion control module and sensor input control module (can automatically process sensor events locally). With UIM241's Advanced Motion Control, selected NEMA 17/23 motors can ramp up to 6600 RPM in 0.5 seconds and 4000 RPM in 0.25 seconds. There are videos on both modules available for downloading. Interested user can go to www.uirobot.com to download.

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

1.1 Basic Control System

UIM241XX controller's basic control system comprises communication system, basic motion control system, and absolute position counter and real-time event-based change notification system.

1.1.1 Communication

UIM241XX communicates with user device through RS232 protocol. User device controls UIM241XX though ASCII coded instructions. Communication baud rate can be changed through instruction, and will be burn into on-board EEPROM.

1.1.2 Basic Motion Control

User device can control following basic motions/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 0 ~ 65,000 pulses/sec, and angular displacement input range is 0 ~ 2,000,000,000 pulses.

1.1.3 Absolute Position Counter

UIM241XX has a hardware pulse counter. Output of the counter is signed. The counter can be reset either by user instruction or automatically 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 can be automatically generated message feedback to the user device, given the corresponding configuration through user instruction.

1.1.4 Real-time Change Notification (RTCN)

Similar to CPU's interrupters, UIM241XX 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.

UIM241XX's RTCN system supports 8 events: displacement control done, absolution zero position, sensor 1 rising edge, falling edge, sensor 2 rising edge, falling edge, analog input beyond upper threshold, analog input lower than lower threshold.

All RTCNs can be enabled or disabled by instructions.

1.2 Advanced Motion Control Module

Advanced motion control module is capable to perform following functions without the help of user device: Uniform Acceleration / Deceleration, Nonlinear Acceleration / Deceleration, S-curve Displacement Control and Auto Direction Control. UIM241XX provides 2 methods to define acceleration/deceleration rate:

1. Absolute Value. Input range: 1 ~ 65,000,000 PPS/Sec (pulse/sec^2).
2. Period, i.e., expect period for the acceleration/deceleration to finish. Input range: 1 ~ 60,000 milliseconds.

The input range of the displacement control is 0 ~ 2 billion pulses (steps), with the user defined direction. Under advanced control, the actual direction is controlled by the module itself. Once the displacement control is done, there could be a RTCN, depending on the user configuration. The time from it's done to the RTCN being sent is less than 1 millisecond.

Advanced motion control module can be disabled/enabled by user instruction.

1.3 Sensor Input Control Module

UIM241XX's Sensor Input Control Module supports 2 channels of TTL input or 1 channel digital and 1 channel analog input. User instruction can configure the input type. On board Analog to Digital (ADC) converter is of 12bits accuracy, 50K Hz sampling rate. The analog input is further averaged over 16 samples before output top user. The update rate is 1K Hz.

For digital inputs, user can configure the desired action and REMF when each sensor's status (sensor events) changes. There are 5 actions that can be bound to 6 sensor events:

1. Start running according the user preset speed and acceleration/deceleration rate. Direction is determined by the sensor voltage level.
2. Sudden stop.
3. Decelerate to stop.
4. Reset absolute position counter.
5. Processing displacement control using the motion parameters preset by the user (e.g., SPD, STP, acceleration/deceleration, maximum start speed, maximum diminish speed, etc.).

1.4 Instructions

Instructions for UIM241XX 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;" or "SPD: 1000;" or "SPD 1000;" or "SPD1000;" or "SPD %?&%* 1000;"

In case the user enters a wrong instruction, the controller will return an ACK of error message. Incorrect instructions will not the executed to prevent accidents.

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

INSTRUCTION AND FEEDBACK STRUCTURE

2.0 Instruction and Feedback Structure

Once UIM241XX 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, UIM241XX will further send back a message to notice the user device of the completion of the instruction. Before a new instruction is received, UIM241XX will keep current working status (e.g. running, stop, etc.)

2.1 Instruction Structure

An instruction is a message sent by the user device to UIM241XX to indicate or command curtain operation.

Instructions of UIM241XX obey following rules:

1. Total length (including the terminating semicolon" **;**") of a single instruction is no more than 20 characters.
2. Coded with standard 7 bits ASCII code (1~127). Expended ASCII code is NOT accepted.
3. Have the following structure:

   Instruction Symbol = Value;

   Where,

   Instruction Symbol comprises several 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** part.

   Terminator is the semicolon "**;**" Instruction without terminator will cause the UIM241XX to wait until the presence of the "**;**" In most situation, that will cause unpredictable results.

   Notice, the equal symbol "=" has no effective usage. User can use other characters except "**{**"and "**}**".

4. Only the first letters of an instruction is used by the UIM241XX. Therefore following 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; ENABLE;

Normally, the user device will receive an ACK message on every instruction sent. Thus 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 above situation, user can use 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; };

UIM241XX will only send back 1 ACK on receiving above message.

In above example, "**{**" and "**}**" is called **Macro Operator.** Instructions between a 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 UIM241XX to send back all desired parameters of the basic motion control. (i.e., status of desired: Enable/disable, Current, Micro-stepping, Auto current reduction, Direction, Speed, and Displacement.)

Actually, user can send the null instruction alone, to check the status of above parameters.

In addition, if there is no null instruction ";" after the "}", there will be no ACK message at all.

2.3 Feedback Message Structure

Feedback Message is the message sent to user device from UIM241XX controller. The maximum length of feedback messages is 13 bytes.

Feedback messages from UIM241XX 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 controller network. For UIM241XX, Controller ID is always 0.

Message ID denotes the property of the current message.

For example, 0xCC 0x00 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 16bits data and 32 bits data. Obviously, 16bits data takes three 7bits data, and 32bits data needs five 7btis data to represent.

Terminator denotes the end of a feedback message. UIM241XX controller utilizes "**0xFF**" as the terminator.

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

Figure 2-1: Conversion from three 7bits message data to a 16bits data

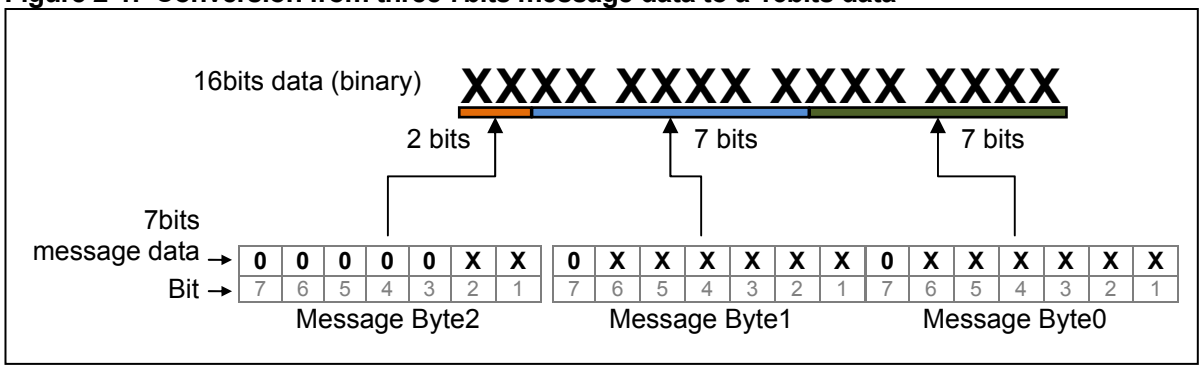
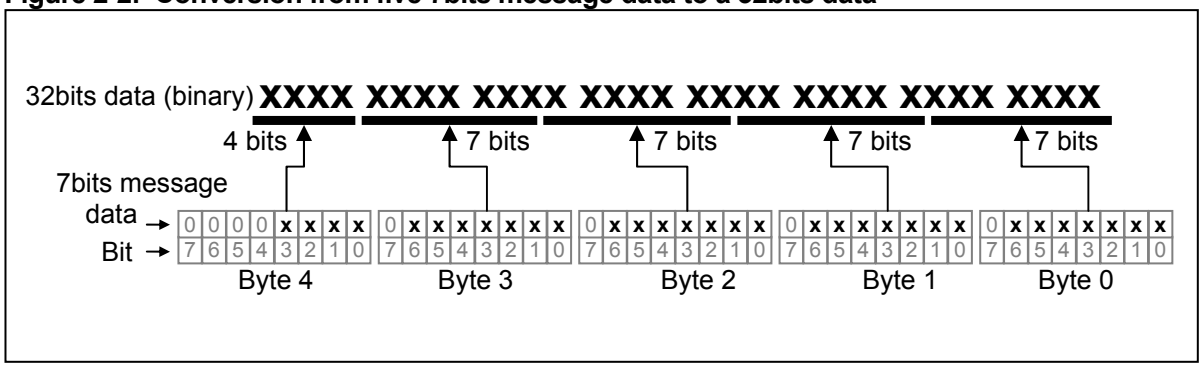


Figure 2-2: Conversion from five 7bits message data to a 32bits data



REAL-TIME CHANGE NOTIFICATION

3.0 Real-time Change Notification (RTCN)

UIM241XX 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 0.5 milliseconds. If using the 57600 baud rate, the transfer time on the RS232 bus is around 0.8 milliseconds. Therefore, the time from the event happening till user device gets the information is less than 1.5 milliseconds.

The structure of a RTCN is shown below:

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

The RTCN system is able to response to following events:

Table 3-1: Real-time Change Notification Events

No	Event	Message ID	Description
1	Falling edge of sensor 1	0xA0	Voltage on S1 port: High >>>Low
2	Rising edge of sensor 1	0xA1	Voltage on S1 port: Low >>>High
3	Falling edge of sensor 2	0xA2	Voltage on S2 port: High >>>Low
4	Rising edge of sensor 2	0xA3	Voltage on S2 port: Low >>>High
5	Exceed upper limits	0xA1*	Analog input > user preset upper limit
6	Below lower limit	0xA0**	Analog input < user preset lower limit
7	Displacement control complete	0xA8	Displacement control is fulfilled and the desired position is reached
8	Zero Position	0xA9	When the absolute position counter reaches / passes the zero position

Note:

* When S1 is configured as analog, 0xA1 denotes event 5, otherwise 0xA1 denotes event 2.

** When S1 is configured as analog, 0xA0 denotes event 6, otherwise 0xA0 denotes event 3.

3.1 Enable / Disable RTCN

Every RTCN can be enabled or disabled by user instruction.

Enable/disable the RTCN is achieved by writing to the Master Configuration Register's ORGIE bit (MCFG<5>), STPIE bit (MCFG<4>), 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 S12CON. Please refer to chapter 8.0 for details.

HARDWARE / FIRMWARE CONFIGURATION

4.0 Hardware / Firmware Configuration

UIM241XX's hardware and firmware can be configured through user instructions to maintained flexibility of the system. This can be achieved through writing the corresponding configuration register(s).

There are 4 configuration registers on UIM241XX: Master Configuration Register, Sensor Input Control Register and 2 Analog Threshold Registers. In this chapter, only the Mater Configuration Register will be described. User can find details about the other three registers in Chapter 8.0.

4.1 Master Configuration Register

Master Configuration Register is used to enable/disable the major functional hardware and firmware. Once configured, it will be effective immediately and its value will be burned into the on board EEPROM automatically and will not affect any real-time process.

Master Configuration Register is a 16bits register with the following structure:

Upper Byte

Name	**ANE**	**CHS**	-	-	-	**CM**	**AM**	**DM**
Bit	15	14	13	12	11	10	9	8

Lower Byte

Name	-	-	**ORGIE**	**STPIE**	-	-	**S2IE**	**S1IE**
Bit	7	6	5	4	3	2	1	0

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 (for UIM241XX, this bit is always 0)

1 = Analog input on port S3 (Only for UIM241XX)

Bit 13-11 Unimplemented. Read as 0.

Bit 10 **CM** 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 input. The instruction value is treated as the speed increment every second. Unit is pps/sec, or pulse/ (square second)

1 = Period input. The instruction value is treated as the time period to accelerate to the desired the speed. Unit is millisecond.

Bit 8 **DM** Deceleration Mode

0 = Value input. The instruction value is treated as the speed decrement every second. Unit is pps/sec, or pulse/ (square second)

1 = Period input. The instruction value is treated as the time period to decelerate to the desired the speed. Unit is millisecond.

Bit 7-6 Unimplemented. Read as 0.

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

0 = Disable the Original (zero) position Real-time Change notification.

1 = Enable the Original (zero) position Real-time Change notification.

Bit 4 **STPIE** Displacement Control (STP) Completion RTCN

0 = Disable the displacement control completion RTCN.

1 = Enable the displacement control completion RTCN.

Bit 3-2 Unimplemented. Read as 0.

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

4.2 Master Configuration Register Instruction（MCFG）

Master Configuration Register can be written using the following instruction:

Instruction MCFG

Function	Setup the value of the Master Configuration Register		
Syntax	**MCFG = x;**	Variable	Integer x = 0, 1 … 65535
ACK	**0xAA 0 0xB0 CFG2 CFG1 CFG0 0xFF** 0xB0 is the Message ID of MCFG. CFG2 ~ CFG0 are message data, can be converted to a 16bits data.		
Comment	When setup, user first fill each bit of the master configuration register with desired number 0 or 1. Then convert to the decimal based number. Finally use the number as the instruction data for MCFG. i.e., replace the x in above instruction with the number calculated. Note, only decimal format number can be accepted.		

Example 4.2.1：

User sent ： MCFG = 34611；

ACK ： 0xAA 0 0xB0 0x2 0xE 0x33 0xff

Explain ： Convert 0x2 0xE 0x33 into 16bits data, we get: 0x8733 (That is 34611 decimal)

4.3 Check Master Configuration Register

If user needs to check the current value of the Master Configuration Register, following instruction can be used:

Instruction MCFG

Function	Check the value of the Master Configuration Register		
Syntax	**MCFG ;**	Variable	-
ACK	**0xAA 0 0xB0 CFG2 CFG1 CFG0 0xFF** 0xB0 is the Message ID of MCFG. CFG2 ~ CFG0 are message data, can be converted to a 16bits data.		
Comment			

RS232 COMMUNICATION

5.0 RS232 Communication

UIM241xx controllers communicate and exchange information with user devices through the RS232 serial protocol. The RS232 configuration of user device, the hand-shaking methods, and the instruction used to change the baud rate will be introduce in this Chapter, along with the method to reset the baud rate to factory default.

5.1 User Device RS232 Port Configuration

To communicate with UIM241XX, user device needs to have following RS232 port settings:

1. 8 bits data
2. 1 stop bit
3. None Parity

5.2 Hand-Shaking

If user device knows the baud rate, it can start sending instructions without hand-shaking.

Hand-shaking is more used as a method to check the existence and firmware version of the controller. Hand-shaking is considered successful, if the user device receives a greeting message starting with **0xAA, 0xAB, 0xAC**.

Under following two situations the UIM2501 will issue a greeting message:

1. When UIM241XX is powering up.
2. When UIM241XX receives following ASCII message: **ABC;** (case sensitive and ended with a semicolon)

A greeting message from UIM241XX has the following structure:

0xAA 0xAB 0xAC 0x18 0x1 [current] 0 0 0 0 [ver1] [ver0] 0xFF

Where,

0xAA 0xAB 0xAC	denotes the greeting message.
0x19, 0x1	denotes the UIM2501 controller.
[**current**]	is the maximum motor current the controller can provide.
[**ver1**:**ver0**]	denotes the firmware version.

5.3 Baud Rate Change Instruction（BDR）

Any out-of-box UIM241XX controller has a factory default baud rate 9600. User can use the 9600 baud rate to connect to a new UIM241XX controller.

To change to a different baud rate, user can use the instruction BDR as described below.

On receiving the BDR instruction, the new baud rate will be stored in the EEPROM and will take effect after the controller is restarted.

Instruction BDR

Function	Change the RS232 baud rate		
Syntax	**BDR = x;**	Variable	Integer x = 9600 … 57600
ACK	**0xAA [Reserved] 0xBD 0xFF** 0xBD is the Message ID of instruction BDR. The **[Reserved]** is for factory use.		
Comment	New Baud Rate will be stored in the controller's non-volatile memory (EEPROM). New baud rate will take effect after the controller is restarted.		

5.4 Reset Baud Rate to Factory Default 9600

In case that user forgets the baud rate and cannot establish the connection, following process can reset the baud rate to the factory default of 9600:

1. Reboot the controller.
2. In 10 seconds, short the baud rate reset pad (figure 0-1 on page 4) to analog ground (screw terminal AG <6>) for 2 times, with an interval around 1 second.
3. Each time, the LED on the controller will flash. If exceed 10 seconds, please restart from step 1.
4. If step 2 is successful, the LED will turn off for one second and re-lit. That indicates the baud rate has been changed to 9600 and ready to use.
5. The BDR instruction can be used to change the 9600 Baud Rate.

BASIC CONTROL FUNCTIONS

6.0 Basic Control Functions

UIM241XX controllers support following basic functions and instructions. These basic instructions are also valid for advanced motion control, given the advanced motion control module is installed and enabled.

1. ENABLE

   This is the instruction to enable the motor driving H-bridge.

2. OFFLINE

   This is the instruction to disable the motor driving H-bridge.

3. CUR

   This is the instruction to set the motor working phase current.

4. ACR

   This is the instruction to activate / deactivate the automatic current reduction function which will supply the working phase current to the motor when the motor is running and supply the idle phase current when the motor is stopped.

5. MCS

   This is the instruction to set the micro stepping.

6. SPD

   This is the instruction to set the desired PPS (pulse per second) to run the motor. Note that the actual motor speed depends also on the micro stepping setup. Refer to example 6.7.1 (section 6.7) for details.

7. STP

   This is the instruction to set the desired steps (pulses) to move the motor. Note that the actual motor angular displacement speed depends also on the micro stepping setup. Refer to example 6.8.1 (section 6.8) for details.

8. DIR

   This is the instruction to set the desired direction for the motor to move.

9. ORG

   This is the instruction to reset the hardware based absolute position counter.

10. FBK

    This is the instruction to ask the controller to send back current motor working status.

11. POS

    This is the instruction to ask the controller to send back the motor's absolute position.

6.1 H-Bridge Enable Instruction（ENABLE）

Instruction ENABLE

Function	Enable the stepper motor driver (i.e. H-bridge driving circuit)		
Syntax	**ENABLE;**	Variable	N/A
ACK	Refer to the “Basic Instruction ACK” (section 6.11) for details.		
Comment	ENABLE instruction turns on the dual H-bridge motor driving circuit.		

6.2 H-Bridge Disable Instruction（OFFLINE）

Instruction OFFLINE

Function	Disable the stepper motor driver (i.e. H-bridge driving circuit)		
Syntax	**OFFLINE;**	Variable	N/A
ACK	Refer to the “Basic Instruction ACK” (section 6.11) for details.		
Comment	OFFLINE instruction turns off the dual H-bridge motor driving circuit. Once an OFFLINE 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.		

6.3 Motor Current Adjusting Instruction（CUR）

Instruction CUR

Function	Set the value of output current		
Syntax	**CUR = x;**	Variable	Integer x = 0,1 … 80
ACK	Refer to the “Basic Instruction ACK” (section 6.11) for details.		
Comment	UIM24104/08 is able to provide maximum 4A/8A peak phase current to the motor. The actual current is commanded through this instruction. It is executed in real-time. 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 execution.		

6.4 Automatic Current Reduction Instruction（ACR）

Instruction ACR

Function	Enable/disable ACR (automatic current reduction) function		
Syntax	**ACR = x;**	Variable	Integer x = 0 or 1
ACK	Refer to the “Basic Instruction ACK” (section 6.11) for details.		
Comment	To reduce the power consumption and temperature raise, UIM241xx controller has the build-in function to automatically reduce output current by half, 0.2 seconds after the motor stopped. Caution has to be taken before enable this function, since current reduction also means holding torque reduction. This function is enabled by default. If ACR = 1; the function is enabled, vice versa.		

6.5 Micro Stepping Setup Instruction（MCS）

Instruction MCS

Function	Set/change micro step resolution		
Syntax	**MCS = x;**	Variable	Integer x = 1,2,4,16
ACK	Refer to the “Basic Instruction ACK” (section 6.11) for details.		
Comment	x = 1, 2, 4, 16 represents the full-step, half-step, quarter-step, 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. Incorrect instructions will be discarded without execution.		

6.6 Motion Direction Instruction（DIR）

Instruction DIR

Function	Set the desired motor direction		
Syntax	**DIR = x;**	Variable	Integer x = 0,1
ACK	Refer to the “Basic Instruction ACK” (section 6.11) for details.		
Comment	DIR =0; or DIR =1; only denotes two opposite turning direction. Actual motor turning direction also depends on the physical wiring situation. For example, swapping the wiring between A+ and A- , or B + and B-, can lead to an opposite turning direction.		

6.7 Speed Adjusting Instruction（SPD）

Instruction SPD

Function	Set the desired speed		
Syntax	**SPD = x;**	Variable	Integer x = 0,1,2… 65000
ACK	Refer to the “Basic Instruction ACK” (section 6.11) for details.		
Comment	Speed is defined as how many steps per second, PPS (Pulses per Second) or Hz. x = 0, 1 ... 65000 represents 0, 1 ... 65000 pulses / sec sent.		

Example 6.7.1：

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*

6.8 Displacement Control Instruction （STP）

Instruction STP

Function	Set the desired steps or micro steps (if MCS≠1) beyond current position		
Syntax	**STP = x;**	Variable	Integer x = 0,1 ... 2,000,000,000
ACK	Refer to the “Basic Instruction ACK” (section 6.11) for details.		
Comment	x is the number of the total pulses will be sent to the motor. A hardware based relative pulse counter will check the actual pulses sent. The actual angular displacement is also depends on the micro-stepping resolution. If an instruction of “STP=0;” is received before the previous STP instruction is fulfilled, the motor will stop running (i.e. SPD is reset to 0), and previous STP instruction is considered fulfilled. If the STP instruction is received when the motor is already running, the steps that the motor will run are counted when the STP instruction is executed.		

Example 6.8.1：

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

User sent: STP = 200;

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

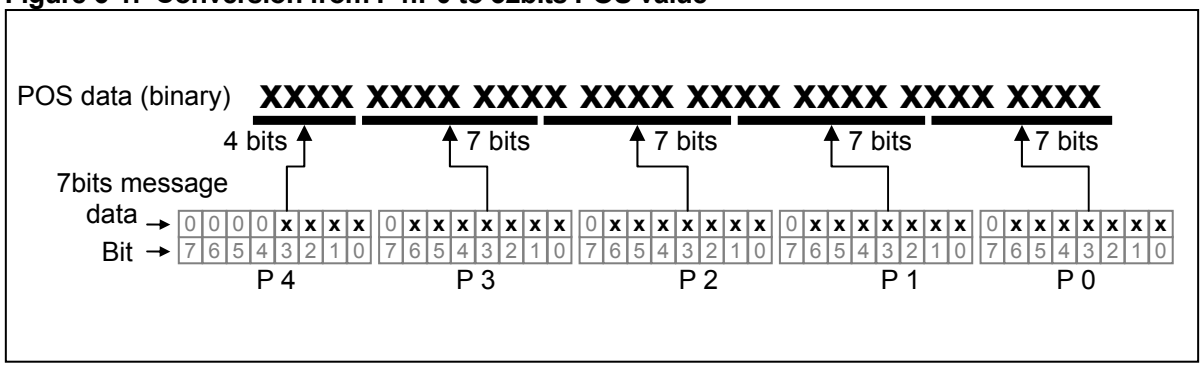
If MCS = 16, motor rotation angle = 1.8 * 200 / 16 = 22.5°

6.9 Absolute Position Inquiry Instruction （POS）

Instruction POS

Function	Check the hardware based absolute position counter (absolute position of the motor)		
Syntax	**POS;**	Variable	N/A
Feedback Message	**0xCC 0 0xB0 P4 P3 P2 P1 P0 0xFF** 0xB0 is the Message ID of POS. (Note, when the message header is 0xAA, 0xB0 is the Message ID of MCFG. Refer to section 4.3 for details.) P4 ~ P0 are message data, can be converted to a 32bits data.		
Comment	P4 ~ P0 represent the value of hardware absolute position counter. It provides a measurement of the absolute position of motor. The hardware absolute position counter only resets (back to zero) under three situations as listed below: 1） Right after the system powers up. 2） User issues the instruction ORG (see the next section). 3） User pre-configured sensor event-based action to ORG, and that event happens. The hardware absolute 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, vise versa. Please note the actual motor angular displacement also depends on the micro-stepping resolution as shown in example 6.8.1.		

Figure 6-1: Conversion from P4:P0 to 32bits POS value



6.10 Absolute Position Counter Reset Instruction（ORG）

Instruction ORG

Function	Reset the hardware based absolute position counter (absolute position of the motor) to zero, to create an origin point.		
Syntax	**ORG;**	Variable	N/A
Feedback Message	**0xCC 0 0xB0 0x0 0x0 0x0 0x0 0x0 0xFF** 0xB0 is the Message ID of POS. (Note, when the message header is 0xAA, 0xB0 is the Message ID of MCFG. Refer to section 4.3 for details.) P4 ~ P0 are message data. Here, they are all zeros.		
Comment	ORG is the abbreviation of “Origin”.		

6.11 Basic Instruction ACK

Upon receiving an instruction, the UIM241XX controller will immediately send back an ACK (Acknowledgment) message. For all basic instructions describe before except POS and ORG, there are only two ACK messages for all of them, as described below.

6.11.1 Error Message

If the received instruction is incorrect, UIM241xx controllers will issue an error message. There are two kinds of errors: Syntax error and value error (the variable range 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

6.11.2 Basic ACK Message

When a valid instruction is received, the UIM241XX controller will send back a basic ACK message. The basic ACK message contains all desired settings including the most current setting. Specifically, following information is included in the ACK message: STP, SPD, DIR, MCS, CUR, ENABLE/OFFLINE, and ACR. The basic ACK message is 13 bytes long and has a structure as shown below.

0xAA 0x0 [ASM byte] [current] SPD2 SPD1 SPD0 STP4 STP3 STP2 STP1 STP0 0xFF

Where,

1. 0xAA denotes a normal ACK message.

2. ASM (assembled) byte structure:

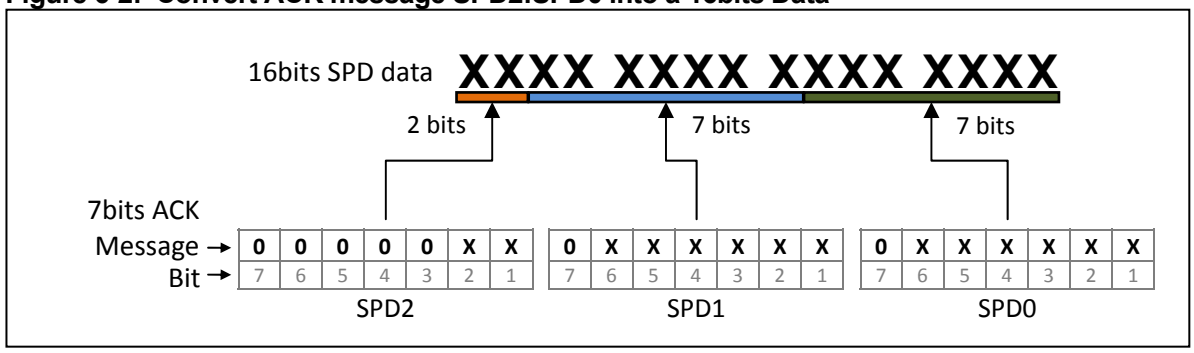
Value	N/A Read 0	ACR	1 = Enable 0 = Offline	DIR	MCS - 1 （0 = full step，15 = 1/16 step)			
Bit	7	6	5	4	3	2	1	0

3. CUR (desired phase current) structure

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

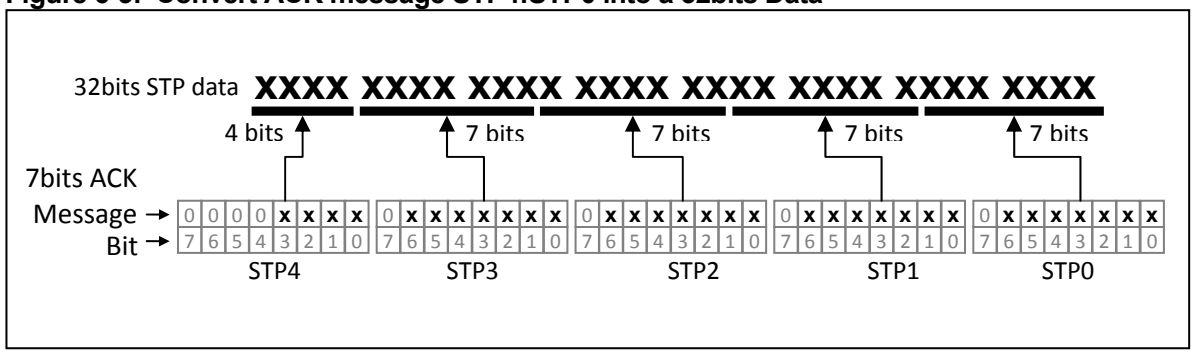
4. Desired speed is represented through SPD2, SPD1, SPD0, using following structure. Bit shifting operation can be used to convert these three bytes into a 16bits data.

Figure 6-2: Convert ACK message SPD2:SPD0 into a 16bits Data



5. Desired displacement is represented through STP4, STP3, STP2, STP1, STP0, suing following structure. Bit shifting operation can be used to convert these three bytes into a 32bits data.

Figure 6-3: Convert ACK message STP4:STP0 into a 32bits Data



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 are free for downloading. User can go to www.uirobot.com to download free copies.

6.12 Motor Status Feedback Inquiry Instruction（FBK）

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

Instruction FBK

Function	Check the current motor status.		
Syntax	**FBK;**	Variable	N/A
Feedback Message	Refer to the “Motor Status Feedback message” (section 6.13) for details.		
Comment	FBK is the abbreviation of “Feed Back”.		

6.13 Motor Status Feedback Message

Upon receiving the FBK instruction, the controller will send back the feedback message comprising following up-to-date motor status: relative displacement/steps, speed, direction, micro step resolution, current, enabled/offline status and ACR status. The feedback Message is 13 bytes long within the following format.

0xCC 0x0 [ASM byte] [current] SPD2 SPD1 SPD0 STP4 STP3 STP2 STP1 STP0 0xFF

Where,

1. 0xCC denotes a Motor Status Feedback Message.
2. ASM (assembled) byte structure:

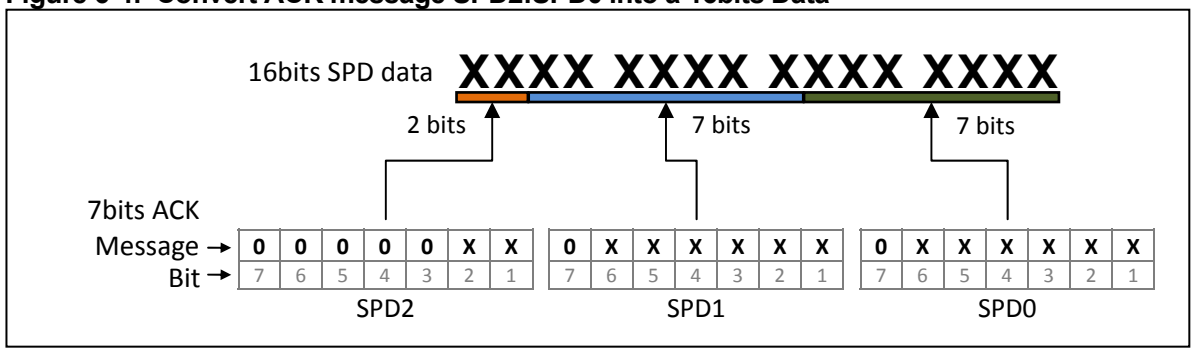
Value	N/A Read 0	ACR	1 = Enable 0 = Offline	DIR	MCS - 1 （0 = full step，15 = 1/16 step）			
Bit	7	6	5	4	3	2	1	0

3. CUR (current phase current) structure

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

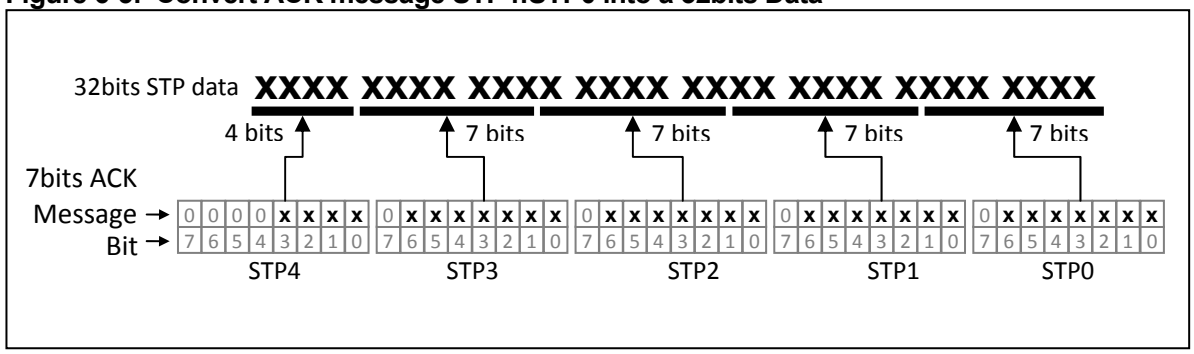
4. Current speed is represented through SPD2, SPD1, SPD0, using following structure. Bit shifting operation can be used to convert these three bytes into a 16bits data.

Figure 6-4: Convert ACK message SPD2:SPD0 into a 16bits Data



5. Current displacement is represented through STP4, STP3, STP2, STP1, STP0, suing following structure. Bit shifting operation can be used to convert these three bytes into a 32bits data.

Figure 6-5: Convert ACK message STP4:STP0 into a 32bits Data



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 are free for downloading. User can go to www.uirobot.com to download free copies.

ADVANCED MOTION CONTROL

7.0 Advanced Motion Control

UIM241XX has an optional (sold separately) module to perform the advanced motion control including uniform or non-linear acceleration/deceleration and S-curve relative displacement 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.

The 5 additional instructions are listed below.

1. MCFG

   This is the instruction to enable or disable the advanced motion control module. User can clear the CM (MCFG<10>) bit of Master Configuration Register (i.e. CM=0) to disable the module or set the CM bit (i.e. CM =1) to enable the module.

2. mACC

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

3. mDEC

   Similar to mACC, the deceleration also has two input methods as listed below.

4. mMSS

   This is the instruction to set the Maximum Starting Speed.

5. mMDS

   This is the instruction to set the Maximum Cessation Speed.

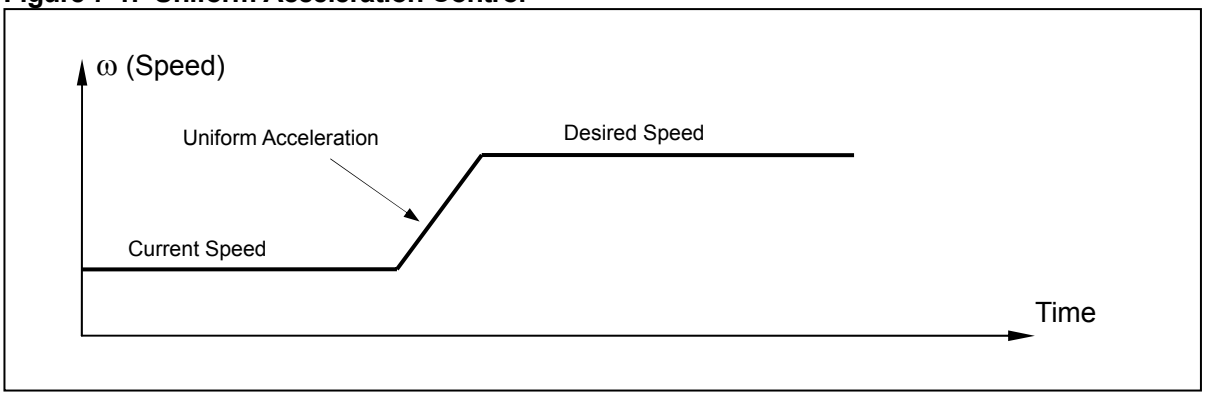
Not counting the communication time, it takes less than 1 millisecond for the specified parameter to take effect after the instruction is received. In addition, values of above instructions will be automatically stored in the EEPROM for future usage, and the storage will not affect any real time process. Once the parameters are set, the controller will perform the advanced motion control automatically. At any time, use can use FBK and POS instructions to get the current status of the motor.

These 5 instructions will be further discussed in the following sections.

7.1 Uniform Acceleration

Uniform acceleration is defined as during the acceleration period, the acceleration rate is constant. The relationship between the speed and time is shown in figure 7-1. Once user set the acceleration and desired speed through instructions (i.e., mACC and SPD), UIM241XX controller will realize the process automatically with 64bit calculation accuracy.

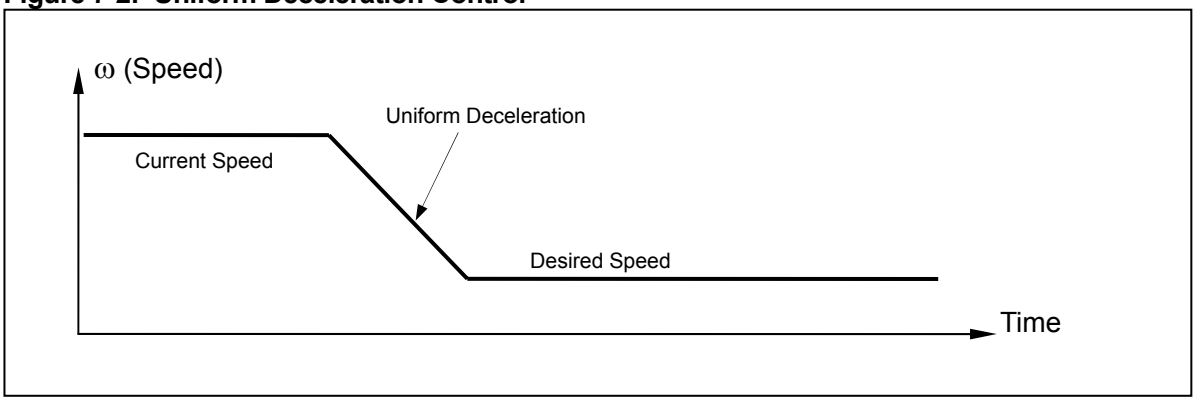
Figure 7-1: Uniform Acceleration Control



7.2 Uniform Deceleration

Uniform deceleration is defined as during the deceleration period, the deceleration rate is constant. The relationship between the speed and time is shown in figure 7-2. Once user set the acceleration and desired speed through instructions (i.e., mDEC and SPD), UIM241XX controller will realize the process automatically with 64bit calculation accuracy.

Figure 7-2: Uniform Deceleration Control



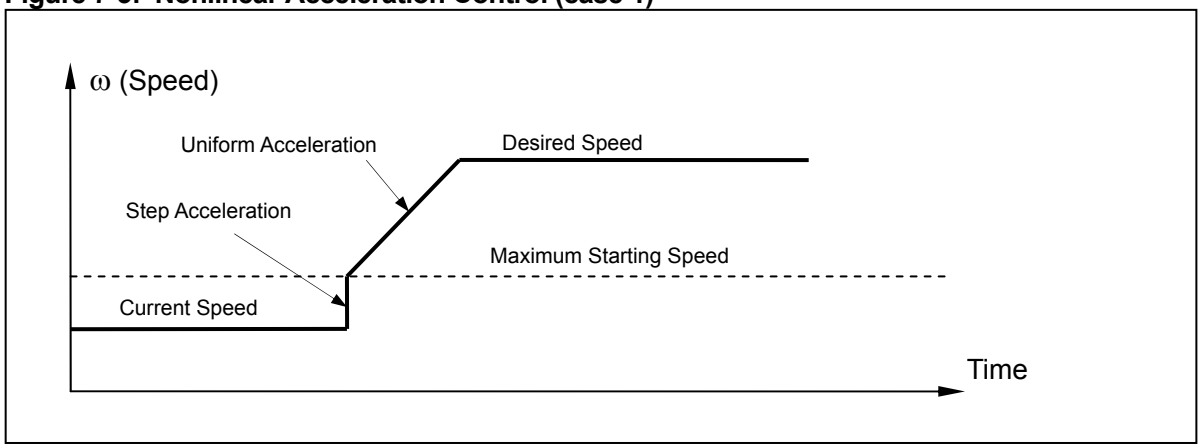
7.3 Nonlinear Acceleration

In order to minimize the response time, make the motor reaching the desired speed in the shortest time without sacrificing stepping accuracy, user may want to utilize the nonlinear acceleration/deceleration capability of the UIM241XX controller. Experiments proved that only by using the nonlinear acceleration, can the NEMA 17/23 reach a 6600 RPM (half step) speed in 0.5 seconds and a 4000 RPM (quad step) speed in 0.25 second.

UIM241XX controllers are equipped with the Nonlinear Acceleration Control function as describe below.

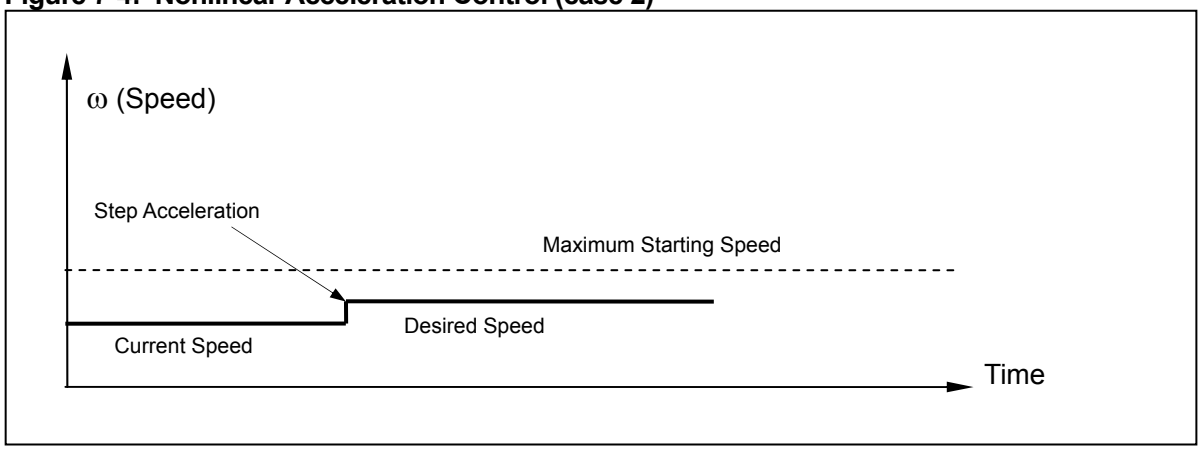
1. If user input desired speed is higher than a certain value (i.e. the Maximum Starting Speed, defined by user through instruction), and current motor speed is lower than the Max. Starting Speed, then the motor speed is first stepped-up to the Max. Starting Speed and then uniformly accelerated according to the user preset acceleration rate. This process is shown in figure 7-3.

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



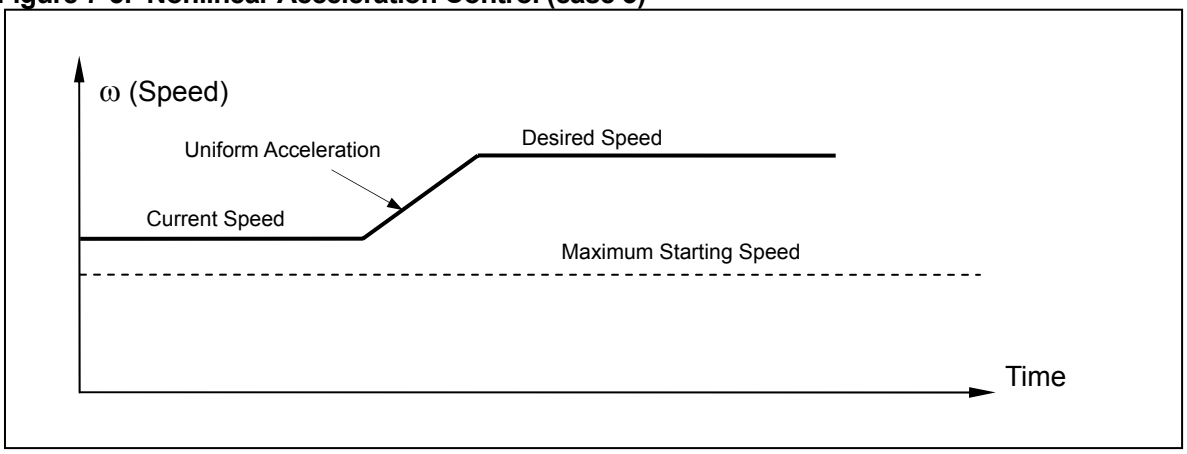
2. If user input desired speed is less than the Max. Starting Speed, then the motor speed is stepped-up to the desired speed immediately, as shown in figure 7-4 .

Figure 7-4: Nonlinear Acceleration Control (case 2)



3. If current speed is higher than the Max. Starting Speed, the UIM241XX will use the Uniform Acceleration Control Algorithm to control the speed.

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

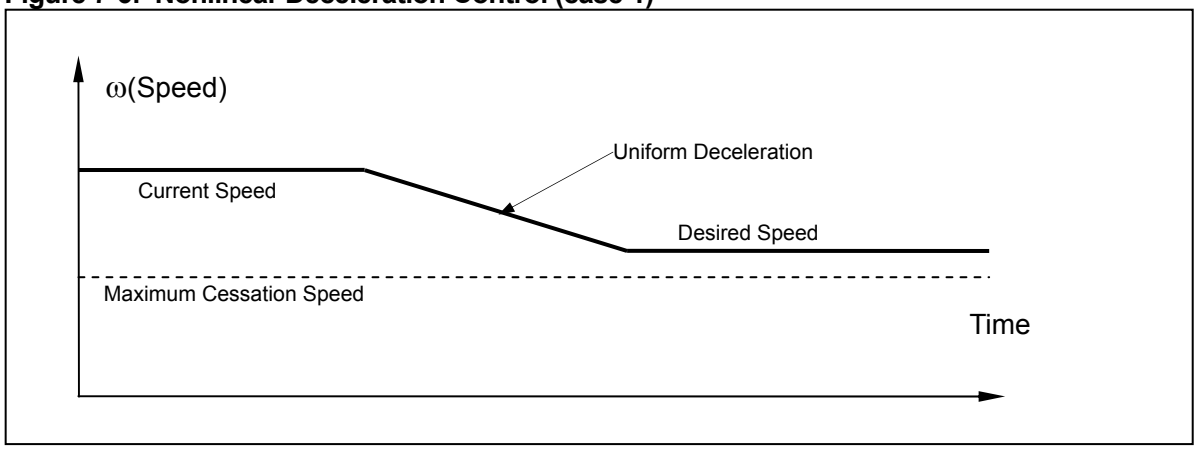


7.4 Nonlinear Deceleration

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

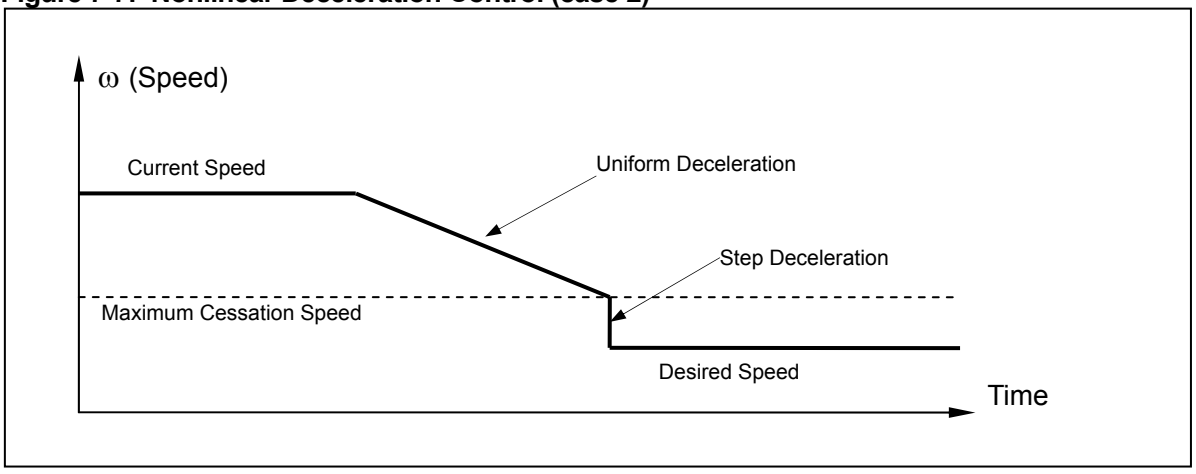
1. If the desired speed is high than a certain user preset value (i.e. the Maximum Cessation Speed), then the UIM241XX will use the Uniform Deceleration Control algorithm.

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



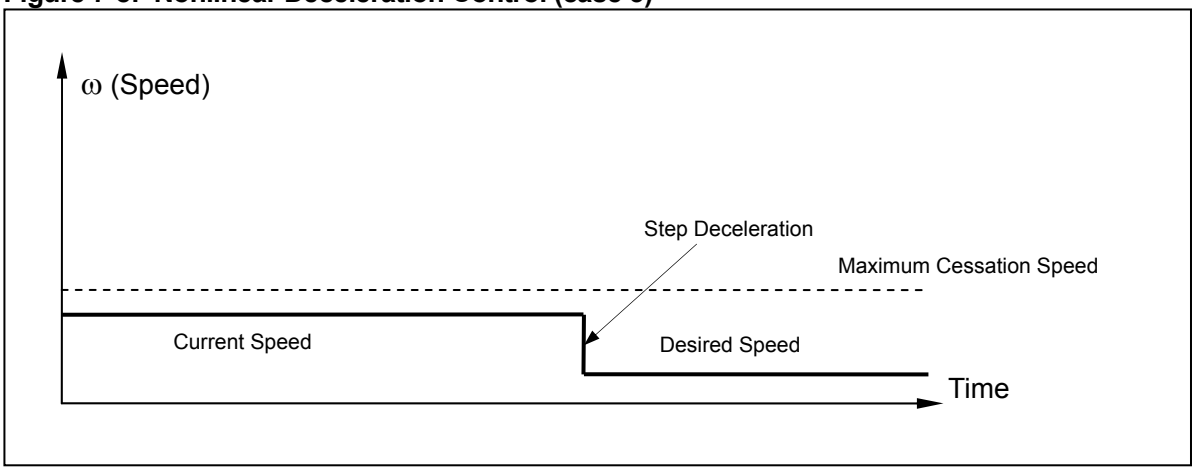
2. If desired speed is less than the Max. Cessation Speed and current motor speed is higher than the Max. Cessation Speed, the Uniform Deceleration Control will be first applied and followed by a step deceleration to the desired speed.

Figure 7-7: Nonlinear Deceleration Control (case 2)



3. 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 immediately.

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



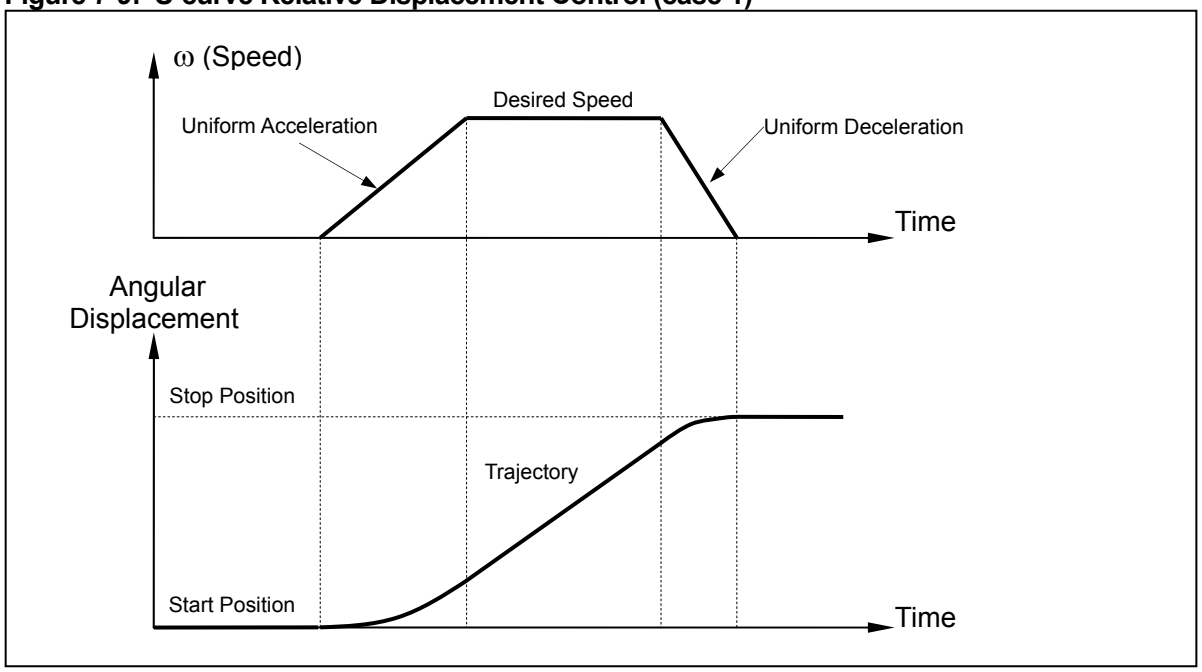
Note: Setting the Maximum Starting Speed or the Maximum Cessation Speed to 0 (zero), will force the controller use Uniform Acceleration / Deceleration Control Algorithm.

7.5 S-curve Relative Displacement Control

S-curve relative displacement control essentially is the relative displacement control under the uniform acceleration/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.

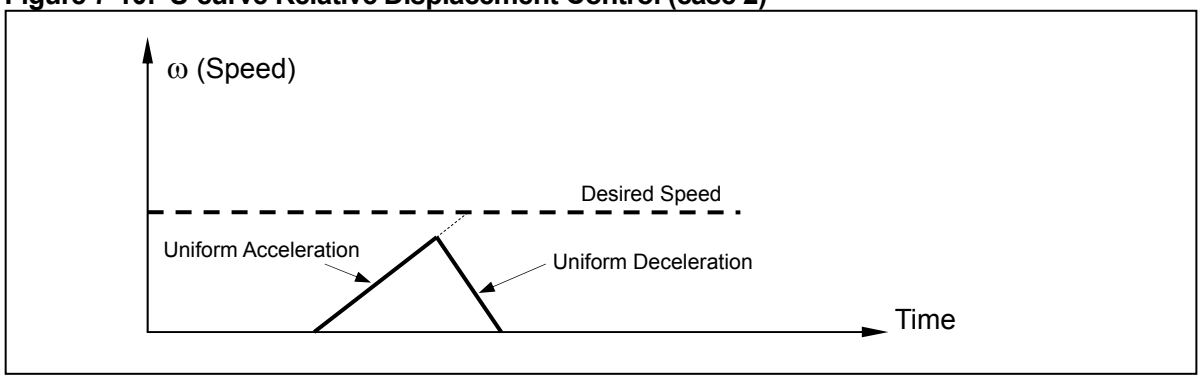
In the control process, UIM241XX's advance motion control module will continuously calculate the deceleration happening point (time) and then perform the deceleration to guarantee that when desired displacement is reached, the speed is right zero. The entire calculation time is around 50 micro-seconds (50 x 10^{-6} second) with 64bit accuracy.

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



If the user input speed is too large, UIM241XX will perform deceleration control before the motor speed reaching the user desired speed, in order to guarantee that when desired displacement is reached, the speed is right zero. The process is shown in figure 7-10.

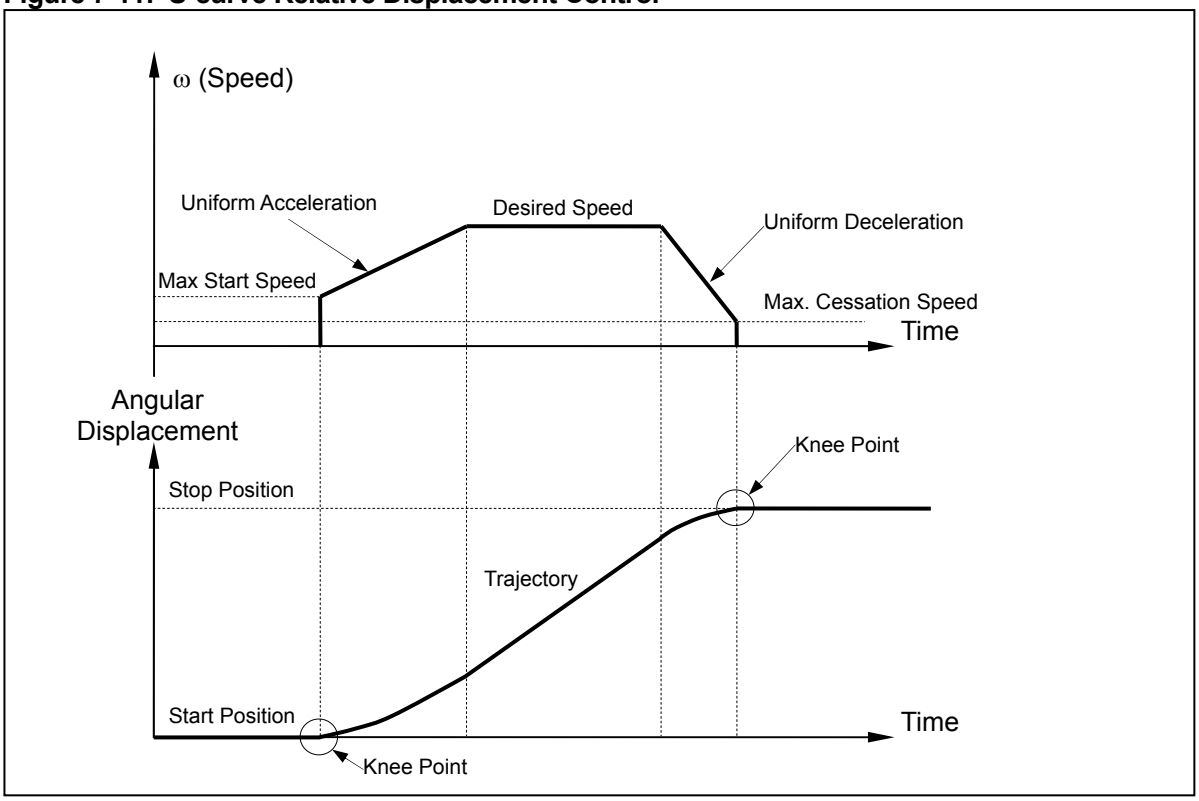
Figure 7-10: S-curve Relative Displacement Control (case 2)



UIM241XXs also support nonlinear acceleration / deceleration based relative displacement

control, although that is not the exact S-curve displacement control. Nonlinear acceleration/ deceleration based displacement control process is shown in figure 7-11. Please note that the nonlinear displacement control described above may not suitable for applications requiring motion smoothness, since there are knee points on the trajectory.

Figure 7-11: S-curve Relative Displacement Control



7.6 Automatic Direction Control

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.

In addition, the input values of SPD and STP are unsigned integers. Therefore, when user wants to input a negative value, the DIR must to be used before the SPD and STP instructions.

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.

These 5 instructions are 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 input methods to set the acceleration rate:

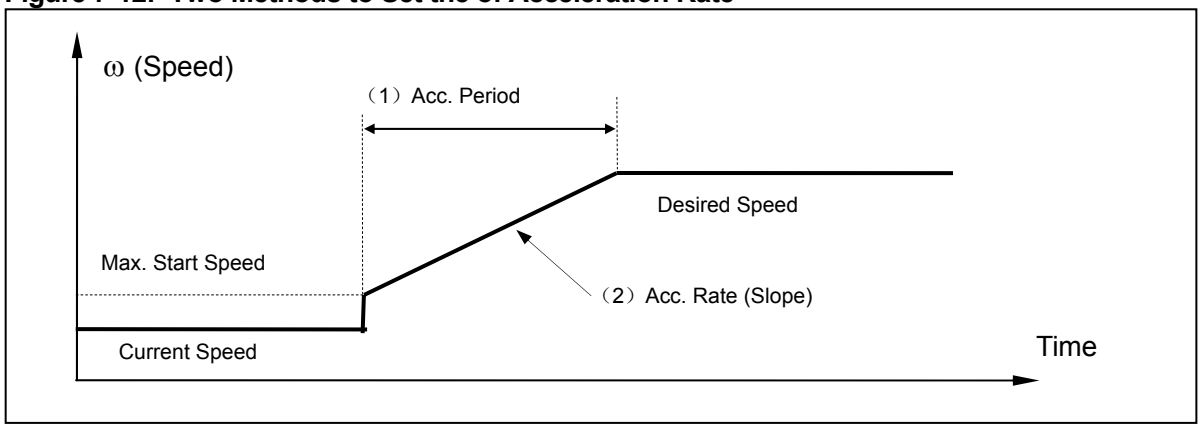
   a. Absolute Value method.

      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 absolute 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 method

      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 Methods to Set the of Acceleration Rate



3. mDEC

   Similar to mACC, the deceleration also has two input methods as listed below.

   a. Absolute Value method.

      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 absolute 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 method.

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）

Even if the UIM241XX is equipped with advanced motion control module, user can enable or disable the module through instruction.

This is realized through setting or clearing the CM bit (MCFG<10>) of the master configuration register. 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.

Details on how to write the master configuration module can be found in section 4.1.

7.9 Acceleration Rate Setup Instruction（mACC）

7.9.1 Absolute Value Method (pre-requiring MCFG<AM> = 0)

Instruction mACC

Function	Set the acceleration rate		
Syntax	**mACC=x;**	Variable	Integer x = 1, 2 … 65,000,000
ACK	**0xAA [Controller ID] 0xB1 ACF AC4 AC3 AC2 AC1 AC0 0xFF** 0xB1 is the Message ID of mACC. AC4 ~ AC0 represents the value of the acceleration rate. ACF = the AM bit of the MCFG (here always =0). ACF=0 means the value input will be interpreted as the absolute acceleration rate.		
Comment	mACC is the abbreviation of “motion Acceleration”. Unit: pps/s (pulse/sec/sec, pulse/squre-second)		

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

Instruction mACC

Function	Set the acceleration rate		
Syntax	**mACC=x;**	Variable	Integer x = 1, 2 … 60,000
ACK	**0xAA [Controller ID] 0xB1 ACF 0x0 0x0 AC2 AC1 AC0 0xFF** 0xB1 is the Message ID of mACC. AC2 ~ AC0 represents the value of the acceleration period. ACF = the AM bit of the MCFG (here always =1). ACF=1 means the value input will be interpreted as the desired time for acceleration.		
Comment	mACC is the abbreviation of “motion Acceleration”. Unit: milliseconds		

7.10 Check the Current Acceleration Rate

Instruction mACC

Function	Check current acceleration rate		
Syntax	**mACC;**	Variable	N/A
ACK	**0xAA [Controller ID] 0xB1 ACF AC4 AC3 AC2 AC1 AC0 0xFF** 0xB1 is the Message ID of mACC. AC4 ~ AC0 represents the value of the acceleration rate/period. ACF = the AM bit of the MCFG (=0 or 1). This value determines the method used by the system to interpret the AC4~AC0 value. When ACF =1, the unit is milliseconds When ACF =0, the unit is pps/sec or pulse/square-second		
Comment			

7.11 Deceleration Rate Setup Instruction（mDEC）

7.11.1 Absolute Value Method (pre-requiring MCFG<DM> = 0)

Instruction mDEC

Function	Set the deceleration rate		
Syntax	**mDEC=x;**	Variable	Integer x = 1, 2 … 65,000,000
ACK	**0xAA [Controller ID] 0xB2 DCF DC4 DC3 DC2 DC1 DC0 0xFF** 0xB2 is the Message ID of mDEC. DC4 ~ DC0 represents the value of the deceleration rate. DCF = the DM bit of the MCFG (here always =0). DCF=0 means the value input will be interpreted as the absolute deceleration rate.		
Comment	mDEC is the abbreviation of “motion deceleration”. Unit: pps/s (pulse/sec/sec, pulse/square-second)		

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

Instruction mDEC

Function	Set the deceleration rate		
Syntax	**mDEC=x;**	Variable	Integer x = 1, 2 … 60,000
ACK	**0xAA [Controller ID] 0xB2 DCF 0x0 0x0 DC2 DC1 DC0 0xFF** 0xB2 is the Message ID of mDEC. DC4 ~ DC0 represents the value of the deceleration period. DCF = the DM bit of the MCFG (here always =1). DCF=1 means the value input will be interpreted as the desired time for deceleration.		
Comment	mDEC is the abbreviation of “motion deceleration”. Unit: milliseconds		

7.12 Check the Current Deceleration Rate

Instruction mDEC

Function	Check current deceleration rate		
Syntax	**mDEC;**	Variable	N/A
ACK	**0xAA [Controller ID] 0xB2 DCF DC4 DC3 DC2 DC1 DC0 0xFF** 0XB2 is the Message ID of mDEC. DC4 ~ DC0 represents the value of the deceleration rate/period. DCF = the DM bit of the MCFG (=0 or 1). This value determines the method used by the system to interpret the DC4~DC0 value. When DCF =1, the unit is milliseconds When DCF =0, the unit is pps/sec or pulse/square-second		
Comment			

7.13 Maximum Starting Speed Setup Instruction（mMSS）

Instruction mMSS

Function	Set the Maximum Starting Speed.		
Syntax	**mMSS=x;**	Variable	Integer x = 1, 2 … 65,000
ACK	**0xAA [Controller ID] 0xB3 MS2 MS1 MS0 0xFF** 0XB3 is the Message ID of mMSS. MS2 ~ MS0 represents the value of the Maximum Starting Speed.		
Comment	mMSS is the abbreviation of “motion Maximum Starting Speed”. Unit: pps (pulse/second)		

7.14 Check the Current Maximum Starting Speed

Instruction mMSS

Function	Check the Maximum Starting Speed.		
Syntax	**mMSS;**	Variable	N/A
ACK	**0xAA [Controller ID] 0xB3 MS2 MS1 MS0 0xFF** 0XB3 is the Message ID of mMSS. MS2 ~ MS0 represents the value of the Maximum Starting Speed.		
Comment	mMSS is the abbreviation of “motion Maximum Starting Speed”. Unit: pps (pulse/second)		

7.15 Maximum Cessation Speed Setup Instruction（mMDS）

Instruction mMDS

Function	Set the Maximum Cessation Speed.		
Syntax	**mMDS=x;**	Variable	Integer x = 1, 2 … 65,000
ACK	**0xAA [Controller ID] 0xB4 MD2 MD1 MD0 0xFF** 0XB4 is the Message ID of mMDS. MD2 ~ MD0 represents the value of the Maximum Cessation Speed.		
Comment	mMDS is the abbreviation of “motion Maximum Deceleration Speed”. (The reason not using mMCS is the possible confusing with the micro stepping instruction MCS.) Unit: pps (pulse/second)		

7.16 Check the Current Maximum Cessation Speed

Instruction mMDS

Function	Check the Maximum Cessation Speed.		
Syntax	**mMDS;**	Variable	N/A
ACK	**0xAA [Controller ID] 0xB4 MD2 MD1 MD0 0xFF** 0XB4 is the Message ID of mMDS. MD2 ~ MD0 represents the value of the Maximum Cessation Speed.		
Comment	mMDS is the abbreviation of “motion Maximum Deceleration Speed”. (The reason not using mMCS is the possible confusing with the micro stepping instruction MCS.) Unit: pps (pulse/second)		

SENSOR INPUT CONTROL

8.0 Sensor Input Control

UIM241XX Motion Controller has an optional (sold separately) Sensor Control Module which supports three 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, UIM241XX 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 digital or analog input, the input voltage cannot exceed -0.3V ~ 5.3V, otherwise permanent damage may happen.

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, UIM241 can perform motion controls without the user device.

There are 6 sensor events can be configured for S1 and S2, as listed below:

Table 8-1: Sensor Events

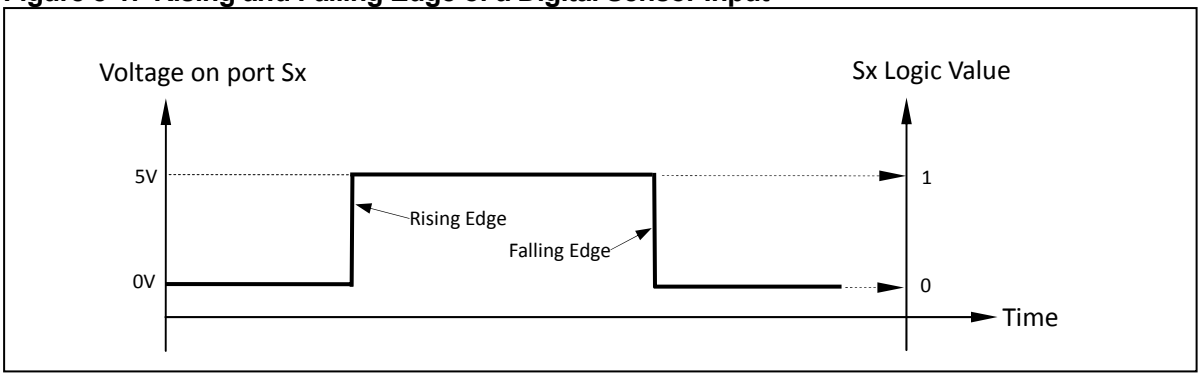
No.	Sensor Event	Description
1	S1 Falling Edge	S1 Voltage Level Change, High >>>Low
2	S1 Rising Edge	S1 Voltage Level Change, Low >>>High
3	S2 Falling Edge	S2 Voltage Level Change, High >>>Low
4	S2 Rising Edge	S2 Voltage Level Change, Low >>>High
5	Exceeding Upper Limit	S1 analog input voltage is higher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

There are 5 actions that can be bound to sensor events:

1. Start and Run using user preset motion parameters.
2. Sudden / Emergent stop.
3. Decelerate until stop.
4. Reset absolute position counter.
5. Execute displacement control using user preset motion parameters.

8.1 Rising and Falling Edge

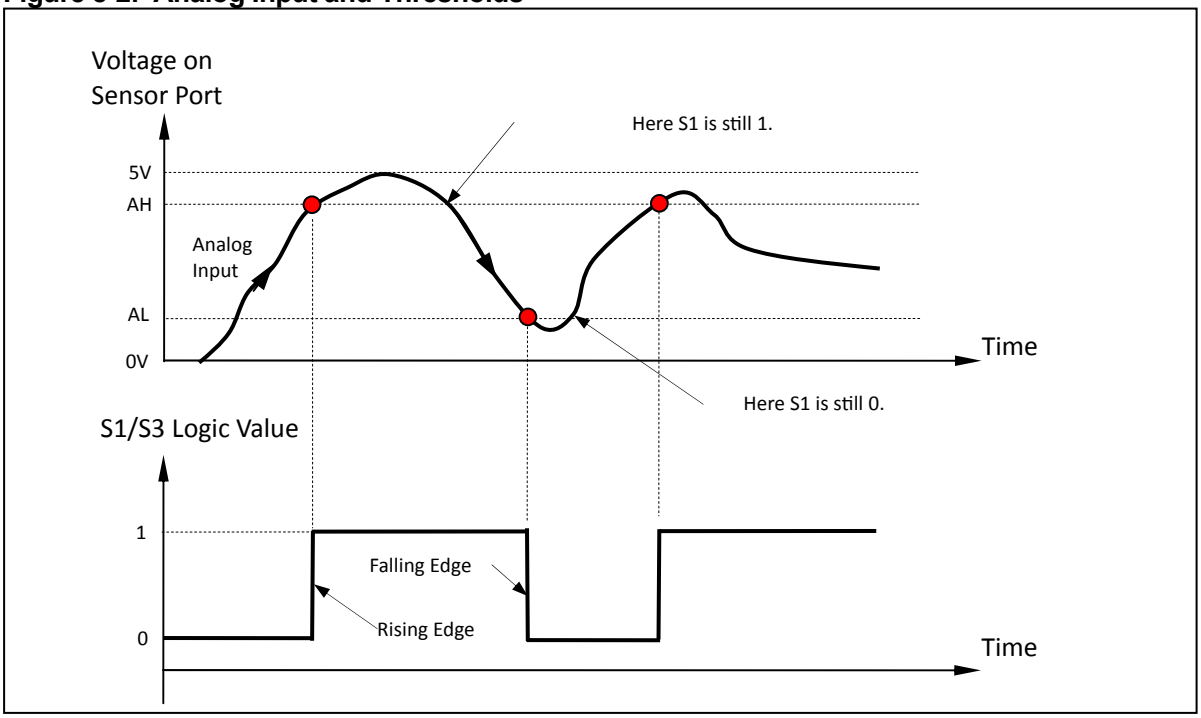
Figure 8-1: Rising and Falling Edge of a Digital Sensor Input



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.

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 immediately 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. UIM241XXs support 6 sensor events as listed in section 8.0. There are 5 actions that can be bound to those 6 sensor events. The IDs for these 5 actions are from 2 to 6. An action ID is a number used to assign an action to a sensor event. The binding between events and actions are realized through the configuration of the Sensor Control Register S12CON. These 5 actions are described below:

1. Run (ID=2)

   Run means starting and continuously running the motor according to 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. Relative Displacement (ID=5)

   Relative Displacement means controlling the motor to realize a given displacement (steps) according to motion parameters (i.e., SPD, STP, ACC/DEC, MSS/MDS, etc.) stored in the EEPROM. Motion's direction is defined by the S12CON. 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.

3. Sudden/Emergent Stop (ID=4)

   Sudden/Emergent Stop means cutting the current speed to zero to force the motor stop immediately.

4. Deceleration Stop (ID=3)

   Deceleration Stop means decelerating the motor speed until stop according to the motion parameters (i.e., DEC, MDS) stored in the EEPROM. To use this action, the advanced motion control must be enabled.

5. Reset the Absolute Position Counter (ID=6)

   This action reset the Absolute Position Counter to zero, creates a zero position or origin.

Once bound, it takes around 1 millisecond (maximum 2ms) for the action to take place after the sensor event happens.

8.4 Introduction to Sensor Control Instructions

There are only 3 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). For details, please refer to section 4.1.

2. SCFG

   The instruction SCFG can be used to configure following two sensor input control registers:

   a. Sensor input control register S12CON

   b. Analog threshold control register ATCONH, ATCONL

3. sFBK

   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 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.

The suffix-code for S12CON is 0000 (binary).

8.5.1 S12CON Structure

Upper Byte

Name	**S2RD**	**S2RACT**			**S2FD**	**S2FACT**		
Bit	15	14	13	12	11	10	9	8

Lower Byte

Name	**S1RD**	**S1RACT**			**S1FD**	**S1FACT**		
Bit	7	6	5	4	3	2	1	0

Bit 15 **S2RD** S2 Rising-edge Motion Direction

0 = for actions that require a specific direction (e.g. Run), DIR = 0.

1 = for actions that require a specific direction (e.g. Run), DIR = 1.

Bit 14-12 **S2RACT<2:0>** Action code for S2 Rising-edge event

000 = No action. No RTCN (Ignore MCFG<S2IE>).

001 = No action. RTCN depends on MCFG<S2IE>.

010 = Run. RTCN depends on MCFG<S2IE>.
011 = Deceleration Stop. RTCN depends on MCFG<S2IE>.
100 = Sudden Stop. RTCN depends on MCFG<S2IE>.
101 = Relative Displacement. RTCN depends on MCFG<S2IE>.
110= Reset Absolute Position Counter. RTCN depends on MCFG<S2IE>.

Bit 11 **S2FD** S2 Falling-edge Motion Direction

0 = for actions that require a specific direction (e.g. Run), DIR = 0.
1 = for actions that require a specific direction (e.g. Run), DIR = 1.

Bit 10-8 **S2FACT<2:0>** Action code for S2 Falling edge event

000 = No action. No RTCN (Ignore MCFG<S2IE>).
001 = No action. RTCN depends on MCFG<S2IE>.
010 = Run. RTCN depends on MCFG<S2IE>.
011 = Deceleration Stop. RTCN depends on MCFG<S2IE>.
100 = Sudden Stop. RTCN depends on MCFG<S2IE>.
101 = Relative Displacement. RTCN depends on MCFG<S2IE>.
110 = Reset Absolute Position Counter. RTCN depends on MCFG<S2IE>.

Bit 7 **S1RD** S1 Rising-edge Motion Direction

0 = for actions that require a specific direction (e.g. Run), DIR = 0.
1 = for actions that require a specific direction (e.g. Run), DIR = 1.

Bit 6-4 **S1RACT<2:0>** Action code for S1 Rising-edge event

000 = No action. No RTCN (Ignore MCFG<S1IE>).
001 = No action. RTCN depends on MCFG<S1IE>.
010 = Run. RTCN depends on MCFG<S1IE>.
011 = Deceleration Stop. RTCN depends on MCFG<S1IE>.
100 = Sudden Stop. RTCN depends on MCFG<S1IE>.
101 = Relative Displacement. RTCN depends on MCFG<S1IE>.
110 = Reset Absolute Position Counter. RTCN depends on MCFG<S1IE>.

Bit 3 **S1FD** S1 Falling-edge Motion Direction

0 = for actions that require a specific direction (e.g. Run), DIR = 0.
1 = for actions that require a specific direction (e.g. Run), DIR = 1.

Bit 2-0 **S1FACT<2:0>** Action code for S1 Falling-edge event

000 = No action. No RTCN (Ignore MCFG<S1IE>).
001 = No action. RTCN depends on MCFG<S1IE>.
010 = Run is bonded. RTCN depends on MCFG<S1IE>.
011 = Deceleration Stop. RTCN depends on MCFG<S1IE>.
100 = Sudden Stop. RTCN depends on MCFG<S1IE>.
101 = Relative Displacement. RTCN depends on MCFG<S1IE>.
110 = Reset Absolute Position Counter. RTCN depends on MCFG<S1IE>.

8.6 Analog Threshold Control Register ATCON

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

Both registers are 16bits registers in the controller memory space. However, when write to them, 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).

8.6.1 ATCONH Structure

Upper Byte

Name	-				AH（11:8）			
Bit	15	14	13	12	11	10	9	8

Lower Byte

Name	AH（7:0）							
Bit	7	6	5	4	3	2	1	0

Bit 15-12 Unimplemented. Read as 0.

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

8.6.2 ATCONL Structure

Upper Byte

Name	-				AL（11:8）			
Bit	15	14	13	12	11	10	9	8

Lower Byte

Name	AL（7:0）							
Bit	7	6	5	4	3	2	1	0

Bit 15-12 Unimplemented. Read as 0.

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

Notice:

ANTHD input range is 0 ~ 4095, with 0 corresponding to 0V and 4095 corresponding to 5V. (4095 is the maximum of a 12bits data).

8.7 Sensor Registers Writing Instruction (SCFG)

Instruction SCFG

Function	Writing the S12CON, ATCONH and ATCONL		
Syntax	**SCFG=x;**	Variable	Integer x = 0, 1 … 1048575 (20bits)
ACK	**0xAA 0x0 0xC0 0x0 0x0 S2 S1 S0 AL1 AL0 AH1 AH0 0xFF** 0xC0 is the Message ID of SCFG. S2 ~ S0 represent the value of 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.		
Comment	Writing to the S12CON and ATCON is realized through instruction SCFG. S12CON and ATCON are 16bits registers in the controller. But when using the SCFG, user has to affix a 4bits suffix code to point the desired register to be written. The suffix code for S12CON is 0000 (binary) The suffix code for ATCONL is 0010(binary) The suffix code for ATCONH is 0011(binary) Details on how to using the SCFG is given in the examples 8.9.1 and 8.9.2.		

8.8 Check the Value of S12CON, ATCONH and ATCONL

Instruction SCFG

Function	Check the current value of S12CON, ATCONH and ATCONL		
Syntax	**SCFG;**	Variable	N/A
ACK	**0xAA 0x0 0xC0 0x0 0x0 S2 S1 S0 AL1 AL0 AH1 AH0 0xFF** 0xC0 is the Message ID of SCFG. S2 ~ S0 represent the value of 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.		
Comment			

8.9 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.

STORE instruction will affect real-time performance. The process will take around 20 milliseconds.

Instruction STORE

Function	Burn MCFG, sensor CFG, Motion Parameters into EEPROM		
Syntax	**STORE;**	Variable	N/A
ACK	**0xAA 0 0xD1 0xFF** 0xD1 is the Message ID of STORE.		
Comment	STORE instruction is used to burn MCFG, Sensor Configuration, Analog Thresholds, desired speed, desired displacement, acceleration/deceleration rate, etc. STORE instruction will affect real time performance. It takes around 20 ms for the instruction to be executed. Therefore, it is suggested that user issues this instruction when the motor is not working, and wait 20ms after this instruction is issued before send other instructions.		

8.10 Examples of Sensor Input Control

8.10.1 Writing the S12CON

Before writing to the 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 8.10.1:

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. We are not interested in the rising edge, therefore S2RD =0 and S2RACT<2:0> = 000
2. It is required DIR=0 on S2 failing edge, therefore, set S2FD=0 and S2FACT<2:0> =010 (Action Run)
3. Same reason as 1, set S1RD =0 and S1RACT<2:0> = 000
4. It is required DIR=1 on S1 failing edge, therefore, set S1FD=1 and S1FACT<2:0> =010 (Action Run)
5. Fill the S12CON with above bits, get: S12CON = 0000 0010 0000 1010 (binary)
6. Affix the suffix-code 0000 to S12CON, get: 0000 0010 0000 1010 0000
7. Convert above value to decimal : 0000 0010 0000 1010 0000 (binary) = 8352
8. **Send instruction: SCFG = 8352;**
9. Set up desired speed, by **sending instruction: SPD=5000;**
10. Burn parameters into EEPROM, by **sending: STORE;**
11. **Press any one limit sensor**, the mobile platform will work.
12. If user enables the RTAFs, the user device will get feedback every time the S1 or S2 is hit.
13. Disconnect the user device, and restart the UIM241XX controller, the system will automatically run.

8.10.2 Writing the ATCONH, ATCONL

Similar to S12CON, 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 8.10.2:

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:

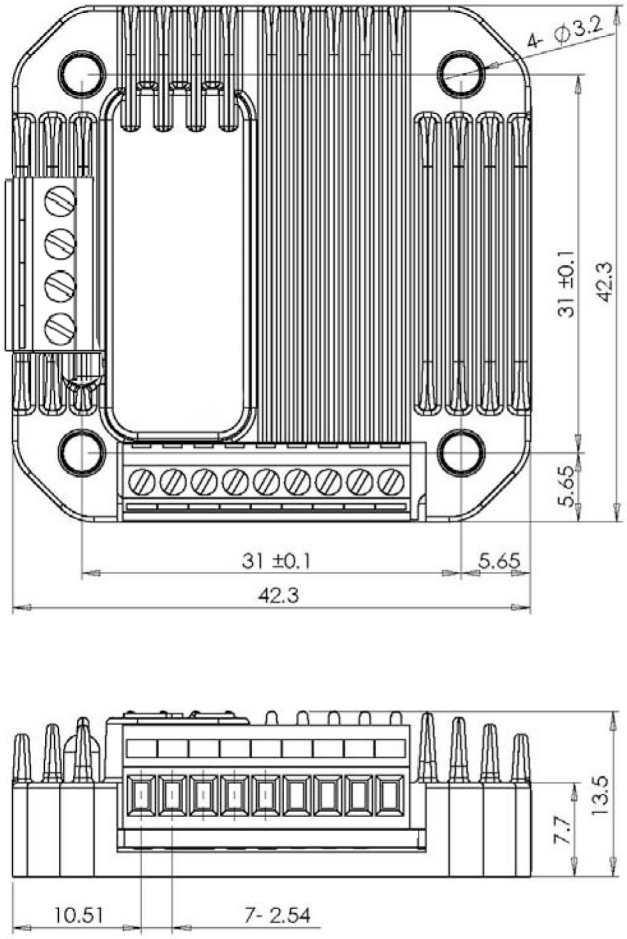
1. As soon as the sensor output is less than 0.6V, the stepper motor starts to run positively (DIR=1) until the potentiometer outputs arrives 4V.
2. As soon as the sensor output is higher than 4V, the stepper motor starts to run reversely (DIR=0) until the potentiometer outputs reaches 0.6V.
3. Keep the reciprocating motion without the user control device.

Realization:

1. Set MCFG<ANE>=1, MCFG<CHS> =0, by **sending: MCFG = 32769;**

   Here, 32769 = 1000 0000 0000 0001 (binary). i.e., ANE=1, CHS=0, S1IE=1, no advanced motion control (user can enable it yourself).
2. It is required DIR=1 on S1 falling edge (when analog input < 0.6V), therefore, S1FD=1 and S1FACT<2:0> =010 (Action Run)
3. It is required DIR=0 pn S1 rising edge (when analog input >4V), therefore, S1RD=0 and S1FACT<2:0> =010 (Action Run)
4. Fill the S12CON with above bits, get S12CON = 0000 0000 0010 1010 (binary)
5. Affix the suffix-code 0000 to S12CON, get: 0000 0000 0010 1010 0000
6. Convert above value to decimal : 0000 0000 0010 1010 0000 = 672
7. **Send instruction: SCFG = 672;**
8. Calculate the upper limit: (4V/5V)*4095 = 3276 = 0000 1100 1100 1100
9. Affix the suffix-code 0011, get: 0000 1100 1100 1100 0011
10. Convert above value to decimal : 0000 1100 1100 1100 0011 = 52419
11. **Send instruction: SCFG = 52419;**
12. Calculate the lower limit: (0.6V/5V)*4095 = 491 (value is rounded)
13. Affix the suffix-code 0010, convert to decimal, then **send instruction: SCFG = 7858;**
14. Set up desired speed, by **sending instruction: SPD=5000;**
15. Burn parameters into EEPROM, by **sending: STORE;**
16. Initiate the motion by **sending: ENABLE;**
17. The system starts to work continuously.
18. Disconnect the user device, and restart the UIM241XX controller, the system will automatically run.

Appendix A Dimensions



Unit: mm

Appendix B Installation