User Manual

UIM241XX Series
RS232 Instruction Control
Miniature Integrated Stepper Motor Controller
V1.2
Please pay attention to the following before using the UIROBOT products:

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

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in this publication regarding controller applications and the like is provided only for your convenience and may be superseded by updates.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

UIROBOT MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. UIROBOT disclaims all liability arising from this information and its use. Use of UIROBOT products in life support and/or safety applications is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless UIROBOT from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any UIROBOT intellectual property rights.

[Trade Mark/ Layout-design/Patent]

The UIROBOT name and logo are registered trademarks of UIROBOT Ltd. in the P.R. China and other countries. UIROBOT’s UIM24XXX series Step Motor Controllers, UIM25XX series CAN-RS232 Converter and their layout designs are patent protected.

[Contact]

SHANGHAI UNITED INTELLIGENCE ROBOTICS INC.
ADD : Rm 202-203, Building Y2, No, 112 Liang Xiu Rd, Zhangjiang Innopark, Shanghai, China
Z I P : 201203
T E L:021 – 61182435(sales,Marketing)
F A X:021 – 61182431
N E T:www.uirobot.com
Email:info@uirobot.com

[UIM241XX Ordering Information]

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

**UIM241XX PART NUMBERING SYSTEM**

<table>
<thead>
<tr>
<th>Category</th>
<th>Series</th>
<th>Function (Optional)</th>
<th>Control Connector</th>
<th>Peak Current</th>
<th>Maximum Supply Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motor Control</td>
<td>RS232 Control</td>
<td>PG = Host-free Program Control</td>
<td>T = Screw Terminal; P = Plug / Socket;</td>
<td>02 = 2A; 04 = 4A; 08 = 8A</td>
<td>L = 28V; C =40V; H=48V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E / IE = External / Internal Encoder Closed-Loop</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>M = Advanced Motion Control</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SP = I/O Control</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:**

1) Peak current is decided by max. supply voltage (See in Table 0-1).
2) Default control connector is T (screw terminal), if not selected.

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

<table>
<thead>
<tr>
<th>Voltage Current</th>
<th>L (28V)</th>
<th>C (40V)</th>
<th>H (48V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2A</td>
<td>✓</td>
<td>✓*</td>
<td>✓*</td>
</tr>
</tbody>
</table>
UIM241XX Miniature Integrated Stepper Motor Controller

<table>
<thead>
<tr>
<th></th>
<th>4A</th>
<th>×</th>
<th>√</th>
<th>√*</th>
</tr>
</thead>
<tbody>
<tr>
<td>8A</td>
<td>×</td>
<td>√</td>
<td>√*</td>
<td></td>
</tr>
</tbody>
</table>

*: Custom made, please contact with salesmans before purchase.

Examples:

UIM241L02P; UIM241C04T-MS; UIM241C08P-IE;

Examples of Control Connector options:

- Screw Terminal
- Rectangular Plug / Socket
GENERAL DESCRIPTION

UIM24102 / UIM24104 / UIM24108 are miniature stepper motor controllers with RS232 interface. UIM241 controllers can be mounted onto NEMA17/23/34/42 series stepper motor through adapting flanges. User device can command these controllers through RS232 protocol using ASCII coded instructions. Instructions are simple, intuitive and fault-tolerating. User is not required to have advanced knowledge on stepper motor driving.

UIM241 can realize open-loop and encoder-based closed-loop control. UIM241’s architecture includes communication system, basic motion control system, absolute position counter, 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.

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

<table>
<thead>
<tr>
<th>Terminal No.</th>
<th>Designator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>V+</td>
<td>Supply voltage, 12 - 48VDC</td>
</tr>
<tr>
<td>2</td>
<td>GND</td>
<td>Supply voltage ground</td>
</tr>
<tr>
<td>3</td>
<td>RX</td>
<td>To the RX pin on user device</td>
</tr>
<tr>
<td>4</td>
<td>TX</td>
<td>To the TX pin on user device</td>
</tr>
<tr>
<td>5</td>
<td>GND</td>
<td>To signal ground on user device</td>
</tr>
<tr>
<td>6</td>
<td>AG</td>
<td>Analog ground for sensors</td>
</tr>
<tr>
<td>7</td>
<td>S1</td>
<td>Sensor input port 1</td>
</tr>
<tr>
<td>8</td>
<td>S2</td>
<td>Sensor input port 2</td>
</tr>
<tr>
<td>9</td>
<td>RST</td>
<td>Reset R232 baud rate to 9600</td>
</tr>
</tbody>
</table>

Motor Terminals

<table>
<thead>
<tr>
<th>Terminal No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A+ / A-</td>
<td>Connect to the stepper motor phase A</td>
</tr>
<tr>
<td>B+ / B-</td>
<td>Connect to the stepper motor phase B</td>
</tr>
</tbody>
</table>

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

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

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

If no special instructions, users use UIROBOT controllers must strictly follow the specification that one controller can only drive one motor. Be careful to avoid connecting more than one motor to the controller. The user should be responsible for the loss caused by that error operation, our company will be not responsible for it.
TYPICAL APPLICATION

Wiring of UIM241 is simple.

UIM241xx controllers use 3-wire RS232 interface to communicate with user devices. Terminal 3 should be connected to the RX of user device; Terminal 4 should be connected to the TX of user device; Terminal 5 should be connected to the GND of user device. An example is provided in figure 0-2. User can use an existing RS232 cable or a converted cable.

If the sensor inputs are used, make sure the signal are wired to the terminal 7 and/or terminal 8, and the signal ground are wired to the terminal 6. Furthermore, please be aware:

- User is responsible for the power supply for sensors,
- Voltage on terminal 7 and 8 must be kept between -0.3V and 5.3V
- If using an external encoder, channel A should be connected to S1; channel B to S2; GND to AG

Figure 0-2: Typical Application

- 12 ~ 48VDC
- PIN2 to RX (UIM terminal 3)
- PIN3 to TX (UIM terminal 4)
- PIN5 to GND (UIM terminal 5)
- Terminal 9 is for resetting Baud Rate (details in chapter 3.4)
## INSTRUCTION SET SUMMARY

### Network Communication

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Feedback Header</th>
<th>Message ID</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BDR\text{\textsuperscript{\textbullet}};</td>
<td>Set RS232 communication baud rate</td>
<td>AA</td>
<td>BD</td>
<td>53</td>
</tr>
</tbody>
</table>

### Model Check

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Feedback Header</th>
<th>Message ID</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDL;</td>
<td>Check the model of controller</td>
<td>CC</td>
<td>DE</td>
<td>70</td>
</tr>
</tbody>
</table>

### Function Configuration

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Feedback Header</th>
<th>Message ID</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENA\text{\textsuperscript{\textbullet}};</td>
<td>Set enable time, boot time after \text{\textbullet} ms enable</td>
<td>AA</td>
<td>A0</td>
<td>58</td>
</tr>
<tr>
<td>ENAxFFFF;</td>
<td>Check enable time</td>
<td>AA</td>
<td>A0</td>
<td>59</td>
</tr>
<tr>
<td>ICF\text{\textsuperscript{\textbullet}};</td>
<td>Set initial configuration register</td>
<td>AA</td>
<td>DA</td>
<td>61</td>
</tr>
<tr>
<td>ICF;</td>
<td>Check initial configuration register</td>
<td>AA</td>
<td>DA</td>
<td>62</td>
</tr>
<tr>
<td>MCF\text{\textsuperscript{\textbullet}};</td>
<td>Set master configuration register</td>
<td>AA</td>
<td>B0</td>
<td>65</td>
</tr>
<tr>
<td>MCF;</td>
<td>Check master configuration register</td>
<td>AA</td>
<td>B0</td>
<td>66</td>
</tr>
<tr>
<td>SCF\text{\textsuperscript{\textbullet}};</td>
<td>Set sensor control configuration register</td>
<td>AA</td>
<td>C0</td>
<td>80</td>
</tr>
<tr>
<td>SCF;</td>
<td>Check sensor control configuration register</td>
<td>AA</td>
<td>C0</td>
<td>81</td>
</tr>
</tbody>
</table>

### General Check

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Feedback Header</th>
<th>Message ID</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>;</td>
<td>Check desired motor status</td>
<td>AA</td>
<td>-</td>
<td>50</td>
</tr>
<tr>
<td>FBK;</td>
<td>Check current motor status</td>
<td>CC</td>
<td>-</td>
<td>60</td>
</tr>
<tr>
<td>SFB;</td>
<td>Check sensor status</td>
<td>CC</td>
<td>C1</td>
<td>62</td>
</tr>
</tbody>
</table>

### Motor Configuration

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Feedback Header</th>
<th>Message ID</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACR\text{\textsuperscript{\textbullet}};</td>
<td>Set auto-current reduction ratio \text{\textbullet}</td>
<td>AA</td>
<td>-</td>
<td>51</td>
</tr>
<tr>
<td>ACR;</td>
<td>Check auto-current reduction ratio</td>
<td>AA</td>
<td>BA</td>
<td>52</td>
</tr>
<tr>
<td>CUR\text{\textsuperscript{\textbullet}};</td>
<td>Set output phase current \text{\textbullet}</td>
<td>AA</td>
<td>-</td>
<td>56</td>
</tr>
<tr>
<td>ENA;</td>
<td>Enable H-bridge circuit</td>
<td>AA</td>
<td>-</td>
<td>57</td>
</tr>
<tr>
<td>MCS\text{\textsuperscript{\textbullet}};</td>
<td>Set micro-stepping resolution</td>
<td>AA</td>
<td>-</td>
<td>67</td>
</tr>
<tr>
<td>OFF;</td>
<td>Disable H-bridge circuit</td>
<td>AA</td>
<td>-</td>
<td>75</td>
</tr>
</tbody>
</table>
## Motion Control

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Feedback Header</th>
<th>Message ID</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLC(\eta);</td>
<td>Set backlash compensation value (\eta)</td>
<td>AA</td>
<td>DE</td>
<td>54</td>
</tr>
<tr>
<td>BLC;</td>
<td>Check backlash compensation value</td>
<td>AA</td>
<td>DE</td>
<td>55</td>
</tr>
<tr>
<td>MAC(\eta);</td>
<td>Set acceleration rate (\eta)</td>
<td>AA</td>
<td>B1</td>
<td>63</td>
</tr>
<tr>
<td>MAC;</td>
<td>Check acceleration rate</td>
<td>AA</td>
<td>B1</td>
<td>64</td>
</tr>
<tr>
<td>MDE(\eta);</td>
<td>Set deceleration rate (\eta)</td>
<td>AA</td>
<td>B2</td>
<td>68</td>
</tr>
<tr>
<td>MDE;</td>
<td>Check deceleration rate</td>
<td>AA</td>
<td>B2</td>
<td>69</td>
</tr>
<tr>
<td>MMD(\eta);</td>
<td>Set maximum cessation speed (\eta)</td>
<td>AA</td>
<td>B4</td>
<td>71</td>
</tr>
<tr>
<td>MMD;</td>
<td>Check maximum cessation speed</td>
<td>AA</td>
<td>B4</td>
<td>72</td>
</tr>
<tr>
<td>MMS(\eta);</td>
<td>Set maximum starting speed (\eta)</td>
<td>AA</td>
<td>B3</td>
<td>73</td>
</tr>
<tr>
<td>MMS;</td>
<td>Check maximum starting speed</td>
<td>AA</td>
<td>B3</td>
<td>74</td>
</tr>
<tr>
<td>ORG(\eta);</td>
<td>Set zero/origin position</td>
<td>AA</td>
<td>B7</td>
<td>76</td>
</tr>
<tr>
<td>ORG;</td>
<td>Reset the position to a given value (\eta)</td>
<td>AA</td>
<td>B7</td>
<td>77</td>
</tr>
<tr>
<td>POS(\eta);</td>
<td>Set desired position (\eta) (open-loop control)</td>
<td>AA</td>
<td>B7</td>
<td>78</td>
</tr>
<tr>
<td>POS;</td>
<td>Check current position</td>
<td>CC</td>
<td>B0</td>
<td>79</td>
</tr>
<tr>
<td>SPD(\eta);</td>
<td>Set the desired speed (\eta)</td>
<td>AA</td>
<td>B5</td>
<td>83</td>
</tr>
<tr>
<td>SPD;</td>
<td>Check current speed</td>
<td>CC</td>
<td>B2</td>
<td>84</td>
</tr>
<tr>
<td>STO;</td>
<td>Store motion control parameters</td>
<td>AA</td>
<td>D1</td>
<td>87</td>
</tr>
<tr>
<td>STO(\eta);</td>
<td>Bind motion control parameters to sensor edge</td>
<td>AA</td>
<td>D1</td>
<td>88</td>
</tr>
<tr>
<td>STP(\eta);</td>
<td>Set desired incremental displacement (\eta)</td>
<td>AA</td>
<td>B6</td>
<td>89</td>
</tr>
<tr>
<td>STP;</td>
<td>Check current incremental displacement</td>
<td>CC</td>
<td>B3</td>
<td>90</td>
</tr>
</tbody>
</table>

## I/O Control

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Feedback Header</th>
<th>Message ID</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>STG(\eta);</td>
<td>Set digital input sampling mode</td>
<td>AA</td>
<td>C9</td>
<td>85</td>
</tr>
<tr>
<td>STG;</td>
<td>Check digital input sampling mode</td>
<td>AA</td>
<td>C9</td>
<td>86</td>
</tr>
</tbody>
</table>
CHARACTERISTICS

Absolute Maximum Ratings

- Supply voltage: 10V to 50V
- Voltage on S1/S2 with respect to GND: -0.3V to +5.3V
- Maximum output current sunk 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

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)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply Power Voltage</td>
<td>12V - 48VDC</td>
</tr>
<tr>
<td>Motor Output Current</td>
<td>Max 2A/4A/8A per phase (instruction adjustable)</td>
</tr>
<tr>
<td>Driving Mode</td>
<td>PWM constant current</td>
</tr>
<tr>
<td>Stepping Resolution</td>
<td>full-step, half-step, 1/4, 1/8 and 1/16 step</td>
</tr>
</tbody>
</table>

Communication (Ambient Temperature 25°C)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>RS232</td>
</tr>
<tr>
<td>Wiring method</td>
<td>3-wire: TX, RX, GND</td>
</tr>
<tr>
<td>Baud Rate</td>
<td>Max 57600 bps, instruction adjustable, Firmware reset to 9600</td>
</tr>
</tbody>
</table>

Environment Requirements

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cooling</td>
<td>Free air</td>
</tr>
<tr>
<td>Working environment</td>
<td>Avoid dust, oil mist and corrosive gases</td>
</tr>
<tr>
<td>Working temperature</td>
<td>-40 °C ~ 85°C</td>
</tr>
<tr>
<td>Humidity</td>
<td>&lt;80%RH, no condensation, no frosting</td>
</tr>
<tr>
<td>Vibration</td>
<td>3G Max</td>
</tr>
<tr>
<td>Storage temperature</td>
<td>-50 °C ~ 150 °C</td>
</tr>
</tbody>
</table>

Size and Weight

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>42.3mm x 42.3mm x 16.5mm</td>
</tr>
<tr>
<td>Weight</td>
<td>0.1 kg</td>
</tr>
</tbody>
</table>
CONTENTS

General Description .................................................................................................................. 4
Terminal DESCRIPTION ............................................................................................................ 5
Typical Application .................................................................................................................... 7
Instruction set summary ............................................................................................................ 8
Characteristics .......................................................................................................................... 10
1.0 Overview .......................................................................................................................... 13
1.1 Basic Control System ........................................................................................................ 13
1.2 Advanced Motion Control Module ................................................................................... 14
1.3 Sensor Input Control Module ............................................................................................ 14
1.4 Instructions and Interface ................................................................................................. 15
2.0 Instruction and Feedback Structure ................................................................................. 16
2.1 Instruction Structure ......................................................................................................... 16
2.2 Macro Operator and Null Instruction ............................................................................... 16
3.0 RS232 communication .................................................................................................... 18
3.1 User Device RS232 Port Configuration ............................................................................ 18
3.2 Hand-Shaking ................................................................................................................... 18
3.3 Reset Baud Rate to Factory Default 9600 ....................................................................... 18
3.4 Instruction List .................................................................................................................. 19
4.0 Real-time Change Notification ....................................................................................... 20
4.1 RTCN Structure ................................................................................................................ 20
4.2 Enable/Disable RTCN ....................................................................................................... 20
5.0 initial and HARDware/Firmware Configuration .............................................................. 21
5.1 Initial Configuration Register (Firmware version: 1232 or higher) .................................. 21
5.2 Auto-enable ...................................................................................................................... 22
5.3 User Program .................................................................................................................. 22
5.4 Master Configuration Register ....................................................................................... 22
5.5 Instruction List .................................................................................................................. 23
6.0 Basic Control Instructions ............................................................................................... 24
6.1 General Introduction of Motion Control Modes ............................................................... 24
6.2 Basic Instruction Acknowledgment (ACK) .................................................................... 27
6.3 Motor Status Feedback Message .................................................................................... 28
6.4 Instruction List .................................................................................................................. 29
7.0 Advanced Motion Control ............................................................................................... 30
7.1 Linear Acceleration .......................................................................................................... 30
7.2 Linear Deceleration .......................................................................................................... 30
7.3 Nonlinear Acceleration ................................................................................................. 30
7.4 Nonlinear Deceleration ................................................................................................... 32
7.5 S-curve Displacement Control ....................................................................................... 33
7.6 Direction Control and Position Counter ......................................................................... 34
7.7 Backlash Compensation ................................................................................................. 35
7.8 Advanced Motion Control Instructions .......................................................................... 35
7.9 Enable/disable Advanced Motion Control Module (MCFG) ........................................... 36
7.10 Instruction List ............................................................................................................... 36
8.0 Sensor Input Control ........................................................................................................ 38
8.1 Rising and Falling Edge .................................................................................................... 39
8.2 Analog Input and Thresholds ........................................................................................... 39
8.3 Digital Input Sampling Mode ............................................................................................ 40
8.4 Sensor Event, Action and Binding .................................................................................. 40
8.5 Introduction to Sensor Input Control Instructions ........................................................... 41
9.0 instruction

9.1 Instruction Structure

9.2 Feedback Message Structure

9.3 Instruction Description

1. ; Check desired motor status

2. ACRη Set auto-current reduction ratio

3. ACR Check auto-current reduction ratio

4. BDRη Set RS232 Baud Rate

5. BLClη Backlash compensation

6. BLC Check backlash compensation

7. CURη Motor Current Adjusting

8. ENA H-Bridge Enable

9. ENAη Set enable time

10. ENAfFFFF Check enable time

11. FBK Motor Status Feedback Inquiry

12. ICFη Initial Configuration Register Instruction

13. ICF Check Initial Configuration Register

14. MACη Set Acceleration Rate

15. MAC Check Current Acceleration Rate

16. MCFη / MCFx Set Master Configuration Register Instruction

17. MCF Check Master Configuration Register

18. MCSη Setup Micro Stepping

19. MDEη Set Deceleration Rate

20. MDE Check Current Deceleration Rate

21. MDL Check Controller Model

22. MMDη Set Maximum Cessation Speed

23. MMD Check current Maximum Cessation Speed

24. MMSη Set Maximum Starting Speed

25. MMS Check current Maximum Starting Speed

26. OFF H-Bridge Disable

27. ORG Reset Position Counter

28. ORGη Reset Position Counter

29. POSη Position Control

30. POS Check Current Position

31. SCFη / SCFxη Set Sensor Configuration

32. SCF Check the value of Sensor Configuration

33. SFB Check Sensor Data

34. SPDη Speed Adjusting

35. SPD Check Current Speed

36. STGη Set Digital Input Sampling Mode

37. STG Check Digital Input Sampling Mode

38. STO EEPROM Store

39. STOη Parameter Banding

40. STPη Displacement Control

41. STPη Check Displacement
1.0 OVERVIEW

UIM241 miniature integrated stepper motor controllers communicate with user device using RS232 protocol. The user device controls UIM241 through ASCII coded instructions. Communication baud rate can be changed through instruction.

UIM241 has a size of 42.3mm*42.3mm*16.5mm and is designed to mount onto NEMA17/23/34/42 stepper motors seamlessly. UIM24102 can provide 0.7-2A output current; UIM24104 can provide 1.5-4A output current; UIM24108 can provide 3-8A output current. Current value is adjustable within the range through instructions. Once set, the value is stored in EEPROM. UIM241XX 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. UIM241XX series of controllers work with 12 ~ 48VDC power supply.

UIM241XX can perform open-loop control. The control system comprises communication system, basic motion control system, absolute position counter, and real-time event-based change notification system. There are also two optional modules to be added on customer request: Advanced Motion Module (linear/non-linear acceleration/deceleration, S-curve PV/PVT displacement control), and Sensor Input control Module.

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

1.1 Basic Control System

UIM241 controller’s basic control system comprises communication system, basic motion control system, absolute position counter, and real-time event-based change notification system.

Communication System

UIM241 controller communicates with user device using RS232 protocol. User device controls the UIM241 controller through ASCII coded instructions. Communication baud rate can be changed through instruction.

Basic Motion Control

UIM241 has a build-in basic motion control system. User device can control the following basic motion parameters through instructions in real-time: direction, speed, angular displacement, phase current, micro-stepping, and enable/disable the H-bridge, etc. Speed input range is +/-65,000 pulses/sec, and displacement input range is +/- 2,000,000,000 pulses.

Absolute Position Counter

UIM241 has a hardware pulse counter. The counter can be reset either by user instruction or automatically by the configurable sensor input event. Under most conditions, through the advanced motion control, this counter can provide the absolute position of the motor with enough accuracy. When the counter reaches zero position, there could be automatically generated message feedback to the user device, given the corresponding configuration through user instruction.

Furthermore, with the encoder-based closed-loop control module, the UIM241 can perform self closed-loop control.
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, absolute position reset, sensor 1/2 rising edge and falling edge, analog input beyond upper threshold, analog input lower than lower threshold, etc. All RTCNs can be enabled or disabled by instructions.

1.2 Advanced Motion Control Module

With advanced motion control module installed, UIM241XX controller can maintain linear and non-linear acceleration/deceleration, S-curve displacement control, PT/PVT control, auto direction control, etc. There are two ways to define acceleration/deceleration rate:

1. Value Mode: Input range: 1 ~ 65,000,000 PPS/Sec (pulse/sec2).
2. Period Mode: Input range: 1 ~ 60,000 milliseconds (time to fulfill the acceleration or deceleration).

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

1.3 Sensor Input Control Module

UIM241’s Sensor Input Control Module supports 2 channels of sensor input. They can accept a TTL level input of 0~5V. There is 1 channel can be configured as analog input (Precision: 12bit; Sample frequency: 50K; mean of 16 calculation; Update frequency: 1000Hz). User can configure the desired automatic action triggered by sensor status change. There are 13 actions listed below that can be triggered by sensor event:

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

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;", "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 UIM241XX receives a message (instructions) from the user device, it will first ACK back (repeat) the received instruction, and then execute the instruction. UIM241XX will further send back a message to inform 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 from the user device to UIM241 to Comment certain operation. Instructions of UIM241 follow the rules listed below:

- INS η; or INSx η; or INS;

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

Feedback Message is the message sent to user device from UIM241 controller. The maximum length of feedback messages is 13 bytes.

Feedback messages from UIM241 follow the structure below:

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

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

Controller ID is the identification number of current controller in a network (also known as Node ID). For UIM241, it is always 00..

Message ID denotes the property of the current message.

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

Terminator denotes the end of a feedback message. UIM241 controller utilizes "FF" or “FE” as the terminator. If terminator is “FF”, it means there is no follow-up message; If terminator is “FE”, it means there has follow-up messages.

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

2.2 Macro Operator and Null Instruction

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

CUR 20; MCS 16; SPD 5000; ENA;

The above instruction set will cause 4 ACK messages being transferred on the RS232 bus.
To facilitate the above situation, user can use the following method to send a set of instructions:

\{'Instruction 1; Instruction 2; ...Instruction N;\}; (N<10)

For example:

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

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

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

The semicolon at the end of the instruction set has no letter or number before it. That is called Null Instruction. The only purpose of a Null Instruction is to tell the UIM242XX to feedback all the inquired parameters of the basic motion control. (i.e. Enable/disable, Current, Micro-stepping, Auto current reduction, Direction, Speed, and Displacement)

Actually, user can simply send the null instruction ";;" alone to check the status of the above parameters. If there is no null instruction ";;" after the "}" in the above example, there will be no ACK message at all.
3.0 RS232 COMMUNICATION

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

3.1 User Device RS232 Port Configuration

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

• 8 bits data
• 1 stop bit
• None Parity

3.2 Hand-Shaking

Any out-of-box UIM241 controller has a factory default baud rate 9600. User can use the 9600 baud rate to connect to a new UIM241 controller.

If the baud rate has been changed, the 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. 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. Under following two situations the UIM241XX will issue a greeting message:

1. When UIM241XX is powered up.
2. When UIM241XX receives following ASCII message: ABC; A message started with AA, AB, AC at the user device implies a successful hand-shake.

A greeting Message from UIM241XX has the following structure:

<table>
<thead>
<tr>
<th>Byte</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>AA</td>
<td>AB</td>
<td>AC</td>
<td>18</td>
<td>01</td>
<td>CUR</td>
<td>Module</td>
<td>Firmware Version</td>
<td>00</td>
<td>00</td>
<td>FF</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Where,

AA AB AC denotes the greeting message
18 01 denotes the UIM241 controller.

[CUR] denotes the maximum motor current the controller can provide.
[Module] denotes the optional control modules the controller installed

[Firmware Version] denotes the firmware version. Data is in 7 bits format. Conversion from three 7bits message data to a 16bits integer is illustrated in figure 9-1.

3.3 Reset Baud Rate to Factory Default 9600

In case of forgotten the baud rate and cannot establish the connection, please take the following steps to reset the baud rate to factory default of 9600.

1. Reboot the controller.
UIM241XX Miniature Integrated Stepper Motor Controller

2. In 10 seconds, short the terminal 9 (figure 0-1) to analog ground (terminal 6) for 2 times, with intervals around 1 second.

3. Each time, the LED on the controller will flash. If exceed 10 seconds, please restart from step 1.

4. If 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. Use BDR instruction to change the baud rate to desired value.

3.4 Instruction List

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

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BDRη1;</td>
<td>Set the RS232 communication baud rate of UIM241XX controller to η1</td>
<td>53</td>
</tr>
<tr>
<td>MDL;</td>
<td>Check the Model, installed optional modules and firmware version</td>
<td>57</td>
</tr>
</tbody>
</table>
4.0 REAL-TIME CHANGE NOTIFICATION

UIM241 controllers support Real-time Change Notification (RTCN). Similar to interrupter of CPU, a RTCN is generated and sent when a user predefined event happens. The length of a RTCN is 4 bytes. The time from the occurrence of the event to the sending of the RTCN is less than 1 millisecond. The time is decided by baud rate. The transfer time is about 0.8ms (0.0008s) when the baud rate is 57600. Then, it takes only 1.5ms from an event happening to a RTCN being received.

4.1 RTCN Structure

The structure of an RTCN message is shown below:

```
CC [Controller ID] [Message ID] FF
```

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

<table>
<thead>
<tr>
<th>No.</th>
<th>Event</th>
<th>Message ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>falling edge of S1</td>
<td>A0</td>
<td>Voltage on S1: High &gt;&gt;&gt;Low</td>
</tr>
<tr>
<td>2</td>
<td>rising edge of S1</td>
<td>A1</td>
<td>Voltage on S1: Low &gt;&gt;&gt;High</td>
</tr>
<tr>
<td>3</td>
<td>falling edge of S2</td>
<td>A2</td>
<td>Voltage on S2: High &gt;&gt;&gt;Low</td>
</tr>
<tr>
<td>4</td>
<td>rising edge of S2</td>
<td>A3</td>
<td>Voltage on S2: Low &gt;&gt;&gt;High</td>
</tr>
<tr>
<td>5</td>
<td>exceed upper limits</td>
<td>A1*</td>
<td>Analog input &gt; user preset upper limit</td>
</tr>
<tr>
<td>6</td>
<td>below lower limit</td>
<td>A0**</td>
<td>Analog input &lt; user preset lower limit</td>
</tr>
<tr>
<td>7</td>
<td>displacement control complete</td>
<td>A8</td>
<td>The desired position is reached</td>
</tr>
<tr>
<td>8</td>
<td>zero position</td>
<td>A9</td>
<td>Position counter reaches/passes zero</td>
</tr>
</tbody>
</table>

Note:
* When S1 is configured as analog, A1 denotes event 9, otherwise A1 denotes event 2.
** When S1 is configured as analog, A0 denotes event 10, otherwise A0 denotes event 1.

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>), S2IE bit (MCFG<1>) and S1IE bit (MCFG<0>). Please refer to section 5.4 for details.

Please note, to realize the sensor event control, user needs to further configure the sensor control registers S12CON and ATCON. Please refer to Chapter 8.0 for details.
UIM241’s hardware and firmware can be configured through user instructions. There are 4 configuration registers for UIM241: Initial Configuration Register, Master Configuration Register, S12CON and Analog Threshold Register. In this chapter, only the Initial Configuration Register and Mater Configuration Register are described. User can find details about the other registers in their corresponding chapters.

### 5.1 Initial Configuration Register (Firmware version: 1232 or higher)

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

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>X</td>
</tr>
<tr>
<td>14</td>
<td>X</td>
</tr>
<tr>
<td>13</td>
<td>X</td>
</tr>
<tr>
<td>12</td>
<td>X</td>
</tr>
<tr>
<td>11</td>
<td>X</td>
</tr>
<tr>
<td>10</td>
<td>X</td>
</tr>
<tr>
<td>9</td>
<td>X</td>
</tr>
<tr>
<td>8</td>
<td>X</td>
</tr>
<tr>
<td>7</td>
<td>X</td>
</tr>
<tr>
<td>6</td>
<td>X</td>
</tr>
<tr>
<td>5</td>
<td>X</td>
</tr>
<tr>
<td>4</td>
<td>Elock</td>
</tr>
<tr>
<td>3</td>
<td>PROG</td>
</tr>
<tr>
<td>2</td>
<td>CCW</td>
</tr>
<tr>
<td>1</td>
<td>ENA</td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

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

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

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

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

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

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

#### Bit0 ENA, Auto-enable after power-on

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

![Figure 5-1 Rotation Direction](image)
5.2 Auto-enable

Once ICFG.ENA is set to 1, UIM241 will auto enable the H-Bridge of motor after the power is on for T ms, the interval time (T) can be set through instruction. For details of the instruction, please refer to Chapter 9.

5.3 User Program

User can program on UIM241. Once ICFG.PROG is set to 1, UIM241 will execute user program after the power is on. For details, please refer to “UIM Programming Manual”. UIM241 still can execute user instructions when user program is running.

5.4 Master Configuration Register

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

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>ANE</td>
</tr>
<tr>
<td>14</td>
<td>CHS</td>
</tr>
<tr>
<td>13</td>
<td>QEI</td>
</tr>
<tr>
<td>12</td>
<td>X</td>
</tr>
<tr>
<td>11</td>
<td>QEM</td>
</tr>
<tr>
<td>10</td>
<td>CM</td>
</tr>
<tr>
<td>9</td>
<td>AM</td>
</tr>
<tr>
<td>8</td>
<td>DM</td>
</tr>
<tr>
<td>7</td>
<td>X</td>
</tr>
<tr>
<td>6</td>
<td>X</td>
</tr>
<tr>
<td>5</td>
<td>ORGIE</td>
</tr>
<tr>
<td>4</td>
<td>STPIE</td>
</tr>
<tr>
<td>3</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>S2IE</td>
</tr>
<tr>
<td>0</td>
<td>S1IE</td>
</tr>
</tbody>
</table>

Bit 15 ANE  Enable/Disable Analog Input
0 = Disable the analog input, all sensor are set to digital input
1 = Enable the analog input, Port S1 can be set to analog input

Bit 14 CHS  Analog Input Channel
0 = Analog input on port S1
1 = Analog input on port S3 (only for UIM242)
Note: This bit is always 0, for UIM241, means only S1 can be configured as Analog Input.

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. Once the value of pulsing counter or encoder counter is zero, a message will be send to user device automatically.

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

Bit3 - 2  Unimplemented. Read as 0.

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

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

5.5 Instruction List

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

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICF</td>
<td>Set initial configuration register</td>
<td>61</td>
</tr>
<tr>
<td>ICF;</td>
<td>Check initial configuration register</td>
<td>62</td>
</tr>
<tr>
<td>MCF</td>
<td>Set master configuration register</td>
<td>65</td>
</tr>
<tr>
<td>MCF;</td>
<td>Check master configuration register</td>
<td>66</td>
</tr>
</tbody>
</table>
6.0 BASIC CONTROL INSTRUCTIONS

UIM241 controllers support abundant motion control instructions. The instructions of UIm241 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 UIM241XX motion control modes is provided.

6.1 General Introduction of Motion Control Modes

There are three motion control modes for UIM241XX controller: Velocity Tracking (VT), Position Tracking (PT) and Position Velocity Tracking (PVT).

Velocity Tracking (VT)

In the Velocity Tracking (VT) mode, UIM241XX controller controls the motor speed to track desired speed.

![Figure 6-1 Velocity Tracking](image)

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

If Advanced Motion Control Module is installed, speed control can be achieved through linear or non-linear acceleration/deceleration. For details, please refer to Chapter 7.0 Advanced Motion Control. If Advanced Motion Control Module is not installed, once a SPD instruction is received, motor speed will be set to desired speed.

Position Tracking (PT)

In the Position Tracking (PT) mode, UIM241 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. UIM241 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, UIM241 controller operates in PT mode automatically on receiving position instruction such as POS, STP until an instruction of “STP 0;” is given. (STP is a displacement control instruction. Logically “STP 0;” means no displacement. It is contradictory to send a displacement instruction of no displacement. Therefore, UIM241 will take this instruction as a request to shift from PT mode to VT mode.)

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

UIM241XX 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:
6.2 Basic Instruction Acknowledgment (ACK)

Upon receiving an instruction, the UIM241XX controller will immediately send back an Acknowledgment (ACK) message. There are only two ACK messages for all of them, as described below.

Error Message

If the received instruction is incorrect, UIM241 will issue an error message and the incorrect instruction will not be executed.

EE [Error Code] FF
Where, EE denotes an error message.

The error code is list below:

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>65</td>
<td>Syntax Error</td>
</tr>
<tr>
<td>66</td>
<td>Value Error</td>
</tr>
</tbody>
</table>

Basic ACK Message

When a valid instruction is received, the UIM241 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 13 bytes long and has a structure as shown below:

<table>
<thead>
<tr>
<th>Byte</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>AA</td>
<td>00</td>
<td>ASB</td>
<td>CUR</td>
<td>SPD2</td>
<td>SPD1</td>
<td>SPD0</td>
<td>STP4</td>
<td>STP3</td>
<td>STP2</td>
<td>STP1</td>
<td>STP0</td>
<td>FF</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>N/A (=0)</td>
<td>ACR</td>
<td>ENA / OFF</td>
<td>DIR</td>
<td>MCS – 1 (0 = full step, 15 = 1/16 step)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3. CUR (desired phase current) structure:

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>N/A (=0)</td>
<td>Phase Current (e.g. 27 = 2.7 Amp)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4. SPD2 – SPD0 denotes the desired motor speed. See figure 9-1 for how to convert to a signed 16bit integer.
5. STP4 – STP0 denotes the desired motor displacement. See figure 9-2 for how to convert to a signed 32bit integer.

### 6.3 Motor Status Feedback Message

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

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

<table>
<thead>
<tr>
<th>Byte</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>CC</td>
<td>0</td>
<td>ASB</td>
<td>CUR</td>
<td>SPD2</td>
<td>SPD1</td>
<td>SPD0</td>
<td>STP4</td>
<td>STP3</td>
<td>STP2</td>
<td>STP1</td>
<td>STP0</td>
<td>FF</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>N/A (=0)</td>
<td>ACR</td>
<td>ENA / OFF</td>
<td>DIR</td>
<td>MCS – 1 (0 = full step, 15 = 1/16 step)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3. [CUR] (current phase current) structure

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>N/A (=0)</td>
<td>Phase Current (e.g. 27 = 2.7 Amp)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4. SPD2 – SPD0 denotes the current motor speed. See figure 9-1 for how to convert to a signed 16bit integer.
5. STP4 – STP0 denotes the current motor displacement. See figure 9-2 for how to convert to a signed 32bit integer.

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

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

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACRη;</td>
<td>Set auto-current attenuation ratio η</td>
<td>51</td>
</tr>
<tr>
<td>ACR;</td>
<td>Check auto-current attenuation ratio</td>
<td>52</td>
</tr>
<tr>
<td>CURη;</td>
<td>Set output phase current η</td>
<td>56</td>
</tr>
<tr>
<td>ENA;</td>
<td>Enable H-bridge circuit</td>
<td>57</td>
</tr>
<tr>
<td>ENAη;</td>
<td>Set enable time, boot after η ms enable</td>
<td>58</td>
</tr>
<tr>
<td>ENAxFFFF;</td>
<td>Check enable time</td>
<td>59</td>
</tr>
<tr>
<td>FBK;</td>
<td>Check current motor status</td>
<td>60</td>
</tr>
<tr>
<td>MCSη;</td>
<td>Set micro-stepping resolution</td>
<td>67</td>
</tr>
<tr>
<td>OFF;</td>
<td>Disable H-bridge circuit</td>
<td>75</td>
</tr>
<tr>
<td>ORG;</td>
<td>Set zero/origin position</td>
<td>76</td>
</tr>
<tr>
<td>ORGη;</td>
<td>Reset the position to a given value η</td>
<td>77</td>
</tr>
<tr>
<td>POSη;</td>
<td>Set desired position η (open-loop control)</td>
<td>78</td>
</tr>
<tr>
<td>POS;</td>
<td>Check current position</td>
<td>79</td>
</tr>
<tr>
<td>SPDη;</td>
<td>Set the desired speed η</td>
<td>83</td>
</tr>
<tr>
<td>SPD;</td>
<td>Check current speed</td>
<td>84</td>
</tr>
<tr>
<td>STPη;</td>
<td>Set desired incremental displacement η</td>
<td>89</td>
</tr>
<tr>
<td>STP;</td>
<td>Check current incremental displacement</td>
<td>90</td>
</tr>
</tbody>
</table>
7.0 ADVANCED MOTION CONTROL

UIM241XX has an optional Advanced Motion Control Module (sold separately) to perform linear/non-linear acceleration/deceleration and S-curve displacement and position control. User can specify corresponding motion control parameters through instructions. Instructions for the advanced motion control includes all the basic motion instructions and 6 additional instructions.

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

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

7.1 Linear Acceleration

Linear acceleration is defined as acceleration at constant rate. The relationship between the speed and time is shown in figure 7-1. After the acceleration rate and desired speed is set (MAC and SPD), UIM241 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 (MDE and SPD), UIM241 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 UIM241XX's non-linear acceleration function. Experiments show that through non-linear acceleration,
UIM241XX can make NEMA17/23 4000RPM (quad step) in 0.25 seconds. UIM241XX 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.

**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.

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

If the current speed is higher than the Max Starting Speed, the UIM241 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), UIM241XX 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, 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 20 micro-seconds with 64bit accuracy. In practice, when the desired displacement is small and the desired speed is high, deceleration starts before the desired speed is achieved to ensure that the speed decelerate to right zero when desired displacement is completed. The process is shown in figure 7-10.
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 in figure 7-11.

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.

UIM241 has two types of position counters: absolute position counter and displacement counter.
Absolute position counter is for recording the absolute position of motor. The actual angular displacement is also relative to micro stepping. The value recorded in absolute position counter will be stored automatically on Power Failure situation and can only be cleared on user instruction or preset sensor event. The counter will increase or decrease according to ICFG.CWW and the actual direction of motor. Absolute position counter value can be read through POS instruction.

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

7.7 Backlash Compensation

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

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

UIM241 controllers provide the function of backlash compensation to reduce the influence on mechanical transmission accuracy.

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

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

7.8 Advanced Motion Control Instructions

There are 6 additional instructions added as listed below.

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

2) Set acceleration: MAC; There are two ways to set the acceleration rate:(Figure 7-12):

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

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

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

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

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

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

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

![Diagram showing two modes to set acceleration](image)

#### 7.9 Enable/disable Advanced Motion Control Module (MCFG)

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

#### 7.10 Instruction List

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

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLC\eta;</td>
<td>Set backlash compensation value \eta</td>
<td>54</td>
</tr>
<tr>
<td>BLC;</td>
<td>Check backlash compensation value</td>
<td>55</td>
</tr>
<tr>
<td>MAC\eta;</td>
<td>Set acceleration rate \eta</td>
<td>63</td>
</tr>
<tr>
<td>MAC;</td>
<td>Check acceleration rate</td>
<td>64</td>
</tr>
<tr>
<td>MDE\eta;</td>
<td>Set deceleration rate \eta</td>
<td>68</td>
</tr>
</tbody>
</table>
### UIM241XX Miniature Integrated Stepper Motor Controller

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MDE;</td>
<td>Check deceleration rate</td>
<td>69</td>
</tr>
<tr>
<td>MMD_1;</td>
<td>Set maximum cessation speed ( \eta )</td>
<td>71</td>
</tr>
<tr>
<td>MMD;</td>
<td>Check maximum cessation speed</td>
<td>72</td>
</tr>
<tr>
<td>MMS_1;</td>
<td>Set maximum starting speed ( \eta )</td>
<td>73</td>
</tr>
<tr>
<td>MMS;</td>
<td>Check maximum starting speed</td>
<td>74</td>
</tr>
</tbody>
</table>
8.0 SENSOR INPUT CONTROL

UIM241XX Motion Controller has an optional (sold separately) Sensor Control Module which supports two sensor input ports: S1, S2. Port S2 can be configured for digital input (0-5V). 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 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, UIM241 can perform motion controls without the user device.

There are 6 sensor events that can be configured, as listed below:

<table>
<thead>
<tr>
<th>No.</th>
<th>Sensor Events</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S1 Falling Edge</td>
<td>S1 Voltage Level Change, High &gt;&gt;&gt;Low</td>
</tr>
<tr>
<td>2</td>
<td>S1 Rising Edge</td>
<td>S1 Voltage Level Change, Low &gt;&gt;&gt;High</td>
</tr>
<tr>
<td>3</td>
<td>S2 Falling Edge</td>
<td>S2 Voltage Level Change, High &gt;&gt;&gt;Low</td>
</tr>
<tr>
<td>4</td>
<td>S2 Rising Edge</td>
<td>S2 Voltage Level Change, Low &gt;&gt;&gt;High</td>
</tr>
<tr>
<td>5</td>
<td>Exceeding the Upper Limit</td>
<td>Analog input voltage is higher than user defined upper limit</td>
</tr>
<tr>
<td>6</td>
<td>Exceeding the Lower Limit</td>
<td>Analog input voltage is lower than user defined lower limit</td>
</tr>
</tbody>
</table>

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

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

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

**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 SFB.

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

Digital input of UIM241 has three sampling mode:

1) Continuous sampling
2) Intermittent sampling
3) Single sampling

Continuous Sampling

In continuous sampling mode, UIM241 controllers detect level fluctuation at port S1/S2 uninterruptedly. Once a fluctuation happens, controllers will call corresponding program, execute pre-set actions, and (or) send a message to user device.

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

Intermittent Sampling

In intermittent sampling mode, user needs to set sampling interval T (1~60000ms) at first. Once a fluctuation is detected at one port, UIM241 controllers will not detect the level fluctuation at this port until (T+1) ms later.

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

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

Single Sampling

In single sampling mode, once a fluctuation is detected at one port, UIM241 controllers will not detect the level fluctuation at this port until user configures the corresponding control bit of S12CON again.

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

8.4 Sensor Event, Action and Binding

UIM241XXs support 6 sensor events as listed in section 8.0. There are 13 actions that can be bound to those 6 sensor events. Binding means assigning a sensor action to a sensor event. The binding between events and actions are realized through the configuration of the Sensor Control Register S12CON. An action-code is needed when configuring sensor registers.

- Start and run forwardly at preset-speed and acceleration (code: 10)
- Start and run reversely at preset-speed and acceleration (code: 2)
- Change direction and run at preset-speed and acceleration (code: 14)
- Forward displacement control follow the preset motion parameters (speed, displacement, acceleration) (code: 13)
- Reverse displacement control follow the preset motion parameters (speed, displacement, acceleration) (code: 5)
- Direction-change displacement control follow the preset motion parameters (speed, displacement, acceleration) (code: 9)
- Decelerate at preset deceleration until stop (code: 3)
- Emergency stop (code: 4)
8.5 Introduction to Sensor Input Control Instructions

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

1. MCF (Master Configuration Register)
   The ANE bit (MCFG<15>) and CHS bit (MCFG<14>) of the master configuration register define the digital/analog input of the sensor port. The S1IE bit (MCFG<0>) and S2IE bit (MCFG<1>) enable/disable the sensor real-time change notification (RTCN). See section 5.4 for details.

2. SCF (Sensor Configuration Register)
   SCF is used to configure following sensor input control registers: S12CON, ATCONH and ATCONL.

3. STG (Sensor Trigger Configuration)
   STG is used to configure sensor trigger mode, UIM241 has three trigger modes: Single Trigger, Continuous Trigger and N ms Intermittent Trigger.

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

5. SFB (Sensor Status Feedback)
   At any time and under any scenario, using the instruction SFB can always read back the logic value of S1 and S2 as well as the analog measurement (given MCFG<ANE>=1, MCFG<CHS> =0).
8.6 Sensor Input Control Register S12CON

S12CON (Sensor 1/2 Control) defines the binding relationship between S1 and S2 sensor events and actions, as well as the activation of corresponding RTCNs. It is a 16bits register inside the controller, and can be configured using the instruction SCF. When writing to it user needs to affix a 4bits suffix-code to point to this register. For details of SCF, please refer to chapter 11.

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

<table>
<thead>
<tr>
<th>Bit</th>
<th>S2RACT</th>
<th>S2FACT</th>
<th>S1RACT</th>
<th>S1FACT</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

8.7 Analog Threshold Control Register ATCONH & ATCONL

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

ATCONH structure is as follows:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Reserved</th>
<th>AH &lt;11:0&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 15-12 Unimplemented, read as 0.
Bit 11-0 AH<11:0> Upper limit of analog threshold.

ATCONL structure is as follows:
**UIM241XX Miniature Integrated Stepper Motor Controller**

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defination</td>
<td>Reserved</td>
<td>AL &lt;11:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

位 15-12  Unimplemented, read as 0.

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

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

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

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCFη;</td>
<td>Set sensor control configuration register η</td>
<td>80</td>
</tr>
<tr>
<td>SCF;</td>
<td>Check sensor control configuration register</td>
<td>81</td>
</tr>
<tr>
<td>SFB;</td>
<td>Check sensor status</td>
<td>82</td>
</tr>
<tr>
<td>STGη;</td>
<td>Set digital input sampling mode</td>
<td>85</td>
</tr>
<tr>
<td>STG;</td>
<td>Check digital input sampling mode</td>
<td>86</td>
</tr>
<tr>
<td>STO;</td>
<td>Store motion control parameters</td>
<td>87</td>
</tr>
<tr>
<td>STOη;</td>
<td>Bind motion control parameters to sensor edge</td>
<td>88</td>
</tr>
</tbody>
</table>

8.9 Example of S12CON Configuration

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

System Description

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

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

Realization:
1. First stop the motor by sending: OFF;
2. We are not interested in the rising edge of S2, so set S2RACT<3:0> = 0000
3. It is required Start and Run Reversely on S2 failing edge, so, set S2FACT<3:0> =0010
4. Same as 2, set S1RACT<3:0> = 0000
5. It is required Start and Run Forwardly on S1 failing edge, so, set S1FACT<3:0> =1010
6. Fill the S12CON with above bits, get: S12CON = 0000 0010 0000 1010 (binary)
7. Affix the suffix-code 0000 to S12CON, get:
   SCFG = 0000 0010 0000 1010 0000 (binary)=0x20A0 (hex)=8352 (decimal)
8. Send instruction:SCFx 20A0; or SCF 8352;
9. Set up desired speed, by sending instruction: SPD 5000;
10. Burn parameters into EEPROM, by sending: STO;
11. Press any one of the limit sensors, the mobile platform will work.
12. Disconnect the user device, and restart the UIM241 controller, the system will automatically run.
13. If enable auto-feedback, once the motor touches limit switch, user device will receive a feedback message of falling-edge on port S1/S2.

8.10 Example of ATCONH, ATCONL Configuration

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

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

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

Realization:
1. First stop the motor by sending: OFF;
2. Set MCFG<ANE>=1, MCFG<CHS>=0, MCFG<S1IE>=1, get:
   MCFG = 1000 0000 0000 0001 (binary) = 0x8001 (hex) = 32769 (decimal)
3. Send instruction: MCF x8001; or MCF 32769;
4. It is required Start and Run Forwardly on S1 falling edge (when analog input < 0.6V), therefore, S1FACT<3:0> = 1010
5. It is required Start and Run Reversely on S1 rising edge (when analog input > 4V), therefore, S1RACT<3:0> = 0010
6. Fill the S12CON with above bits, get: S12CON = 0000 0000 0010 1010 (binary)
7. Add suffix-code 0000 (for S12CON), get:
   SCFG = 0000 0000 0010 1010 0000 (binary)= 0x2A0 (hex)= 672 (decimal)
8. Send instruction: SCF x2A0; or SCF 672;
9. Calculate the upper limit: (4V/5V)*4095 = 3276 = 0000 1100 1100 1100 (binary)
10. Add suffix-code 0011 (for ATCONH), get:
    SCFG= 0000 1100 1100 1100 0011 (binary)= 0xCCC3 (hex)= 52419 (decimal)
11. Send instruction SCF xCCC3; or SCF 52419;
12. Calculate the lower limit: (0.6V/5V) * 4095 = 491 (value is rounded)= 0000 0001 1110 1011 (binary)
13. Add suffix-code 0010 (for ATCONL), get:
    SCFG= 0000 0001 1110 1011 0010 (binary)= 0x1EB2 (hex)= 7858 (decimal)
14. Send instruction: SCF x1EB2; or SCF 7858;
15. Set desired speed, by sending instruction: SPD 5000;
16. Burn parameters into EEPROM, by sending: STO;
17. Send instruction: ENA;
18. The system starts to work continuously.
19. Disconnect the user device, and restart the UIM241 controller, the system will automatically run.
9.0 INSTRUCTION

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

9.1 Instruction Structure

An instruction is a message sent from the user device to motion controller to command certain operation. Instructions of UIM241 follow the rules listed below:

1. Length of an instruction (including the ending semicolon ";") should be within 20 characters.
2. Coded with standard 7 bits ASCII code (1-127). Expended ASCII code is NOT accepted.
3. Instruction structure is as follow:
   
   \[
   \text{INS } \eta ; \\
   \text{or } \text{INSx } \eta ; \\
   \text{or } \text{INS } ;
   \]

Where,

- **INS** Instruction Symbol
  - Comprises three letters with no space between them, and is not case sensitive.
  - If there is an x (INSx), then it means the value is hexadecimal.
  - Please note, if \( \eta \) is hexadecimal, then the data must have an even number of digits, such as 00, 01, 0A. A data has an odd number of digits will cause errors, for example, 001, 10A are illegal input.

- **\( \eta \)** Value
  - Comprises set of numbers, with no other characters between them.
  - Some instruction have no value, such as “SPD;” “STP;” etc.

- **;** Terminator
  - Each instruction must end with semicolon (;)
  - Note: Instruction without terminator will cause unpredictable results.

9.2 Feedback Message Structure

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

Structure of feedback message from UIM241XX is as follow:

\[
[\text{Header}] \ [\text{Controller ID}] \ [\text{Message ID}] \ [\text{Data}] \ [\text{Terminator}]
\]

**Header**
- The start of a feedback message
- There are 3 kinds of headers:
AA represents the ACK message, which is a repeat of the received instruction.
CC represents the status feedback, which is a description of current working status.
EE represents the error message.

Controller ID
The identification number of current controller in a network (also known as Node ID)
For UIM 241, Node ID is always 00.

Message ID
The property of the current message
For example, CC 00 A0 FF, where A0 denotes that there is a low level on sensor 1. For details, please refer to following sections.

Data
Has a 7bits data structure. High is in front, and low is in the back.
In figure 9-1 and 9-2, examples are shown on how to convert a set of 7bits data into 16bit data and 32bit data.
Obviously, one 16bit data takes three 7bit data to represent, and one 32bit data takes five 7bit data.

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

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

Figure 9-1: Conversion from three 7bits message data to a 16bits data
Figure 9-2: Conversion from five 7bits message data to a 32bits data

Data received (Hex):

<table>
<thead>
<tr>
<th>1st Byte</th>
<th>2nd Byte</th>
<th>3rd Byte</th>
<th>4th Byte</th>
<th>5th Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>D0</td>
<td>D1</td>
<td>D2</td>
<td>D3</td>
<td>D4</td>
</tr>
<tr>
<td>D5</td>
<td>D6</td>
<td>D7</td>
<td>D8</td>
<td>D9</td>
</tr>
<tr>
<td>D10</td>
<td>D11</td>
<td>D12</td>
<td>D13</td>
<td>D14</td>
</tr>
<tr>
<td>D15</td>
<td>D16</td>
<td>D17</td>
<td>D18</td>
<td>D19</td>
</tr>
<tr>
<td>D20</td>
<td>D21</td>
<td>D22</td>
<td>D23</td>
<td>D24</td>
</tr>
<tr>
<td>D25</td>
<td>D26</td>
<td>D27</td>
<td>D28</td>
<td>D29</td>
</tr>
</tbody>
</table>

Received sequence: earlier → later

16bit Binary Data
9.3 Instruction Description

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

1. ; Check desired motor status

Format: ;

Description: Check desired motor status

ACK: AA 00 [ASB] [CUR] [V0] [V1] [V2] [P0] [P1] [P2] [P3] [P4] FF

Comment: [ASB] >> Received data 0
[CUR] >> Received data 1
[V0] ~ [P4] >> Received data 2 ~ 9

[ASB] structure:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>N/A(=0)</td>
</tr>
<tr>
<td>6</td>
<td>ACR</td>
</tr>
<tr>
<td>5</td>
<td>ENA / OFF</td>
</tr>
<tr>
<td>4</td>
<td>DIR</td>
</tr>
<tr>
<td>3</td>
<td>MCS – 1(0 = full step, 15 = 1/16 step)</td>
</tr>
</tbody>
</table>

[CUR] structure:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>N/A(=0)</td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Phase Current (e.g. 27 = 2.7 Amp)</td>
</tr>
</tbody>
</table>

[V0] ~ [V2] is the converted value for desired speed (16 bits) (Figure 9-1)
[P0] ~ [P4] is the converted value for desired displacement (32 bits) (Figure 9-2)
2. ACR\(\eta\)  
Set auto-current reduction ratio

**Format:**  
ACR\(\eta\);

**Description:**  
set auto-current reduction ratio \(\eta\)
\[\eta = 0,1,\ldots,99.\]
\(\eta = 0,\) disable auto-current reduction. Standby-CUR = working current.
\(\eta = 1,\) in standby mode, current reduces to 50\%. Standby-CUR = working current / 2.
\(\eta = 2,3,\ldots,99,\) in standby mode, current reduces to 2,3,\ldots,99\%. Standby-CUR = working current * \(\eta / 100.\)

**ACK:**  
\(\eta = 0\) or \(\eta = 1,\) ACK is the same as ACK of “6. ENA”

\(\eta = 2,3,\ldots,99,\) ACK is as follow:
AA 00 BA [A0] FF

**Comment:**  
BA  \(\gg\) Message ID of instruction ACR\(\eta\);
[A0]  \(\gg\) Received data 0, A0 = \(\eta\)

**Note:**  
ACR is short for Automatic Current Reduce.
When ACR is enabled, the current will be reduced after motor stop, which means a decrease of holding torque. Value of this instruction will be stored in EEPROM.
\(\eta = 2,3,\ldots,99\) require controller Firmware version being 1232 or higher.
3. **ACR**  
**Check auto-current reduction ratio**

**Format:**  
ACR;

**Description:**  
Check auto-current reduction ratio

**ACK:**  
AA 00 BA [A0] FF

**Comment:**  
Refer to ACK comment of instruction ACRŋ;

**Note:**  
Require controller Firmware version being 1232 or higher.
4. **BDR**\(\eta\)  Set RS232 Baud Rate

**Format:**  
BDR\(\eta\);

**Description:**  
Set RS232 Baud Rate  
\(\eta = 9600, 19200, 38400, 56000, 100000, 115200, 250000;\)  
Other value of baud rate is also available, but it must be integral multiple of 100.

**ACK:**  
AA [Reserved]  BD  FF

**Comment:**  
[Reserved]  >>  Factory use;  
BD  >>  Message ID of instruction BDR.

**Note:**  
The 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. **BLC\(\eta\)**  
**Backlash compensation**

**Format:**  
BLC\(\eta\);

**Description:**  
Set value of backlash compensation in reciprocating motion  
\(\eta = 0,1,\ldots,65535\) (Unsigned integer)  
Units: pps (open-loop)

**ACK:**  
AA 00 DE [B0] [B1] [B2] FF

**Comment:**  
DE \(\gg\) Message ID of instruction BLC\(\eta\);  
[B0] ~ [B2] \(\gg\) Received data 0 ~ 2

[B0] ~ [B2] is the converted value for the value of backlash compensation (16 bits) (Figure 9-1)
<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
<th>ACK</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLC;</td>
<td>Check the value of backlash compensation in reciprocating motion</td>
<td>AA 00 DE [B0] [B1] [B2] FF</td>
<td>Refer to ACK comment of instruction BLC;</td>
</tr>
</tbody>
</table>
7. CURη  Motor Current Adjusting

Format:  CURη;

Description:  Set the output phase current to η.
η = 0,1,⋯,80 (unsigned integer)
0…80 represent 0…8.0 amps.

ACK:  AA 00 [ASB] [CUR] [V0] [V1] [V2] [P0] [P1] [P2] [P3] [P4] FF

Comment:  [ASB]  >> Received data 0
[CUR]  >> Received data 1
[V0] ~ [P4]  >> Received data 2 ~ 9

[ASB] structure:

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>N/A(=0)</td>
<td>ACR</td>
<td>ENA / OFF</td>
<td>DIR</td>
<td>MCS – 1(0 = full step. 15 = 1/16 step)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[CUR] structure:

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>N/A(=0)</td>
<td>Phase Current (e.g. 27 = 2.7 Amp)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[V0] ~ [V2] is the converted value for desired speed (16 bits) (Figure 9-1)
[P0] ~ [P4] is the converted value for desired displacement (32 bits) (Figure 9-2)

Note:  Value of this instruction will be stored in EEPROM.
If the received current value is not one of the above integers, an Error ACK will be sent to the user device through RS232. Incorrect instructions will be discarded without being executed.
8. ENA  H-Bridge Enable

**Format:**  
ENA;

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

**ACK:**  
AA 00 [ASB] [CUR] [V0] [V1] [V2] [P0] [P1] [P2] [P3] [P4] FF

**Comment:**  
- [ASB]  >> Received data 0
- [CUR]  >> Received data 1
- [V0] ~ [P4]  >> Received data 2 ~ 9

**[ASB] structure:**

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>N/A(=0)</td>
<td>ACR</td>
<td>ENA / OFF</td>
<td>DIR</td>
<td>MCS – 1(0 = full step, 15 = 1/16 step)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**[CUR] structure:**

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>N/A(=0)</td>
<td>Phase Current (e.g. 27 = 2.7 Amp)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[V0] ~ [V2] is the converted value for desired speed (16 bits) (Figure 9-1)
[P0] ~ [P4] is the converted value for desired displacement (32 bits) (Figure 9-2)

**Note:**  
Only after the H-bridge enabled, can the controller drive the motor.
9. ENAη  Set enable time

Format:   ENAη;

Description:  Set auto-enable time register ENAtimer.
Controller auto-enable the H-bridge circuit after power is on for η ms.
η = 1, 2, ..., 60000;

ACK:  AA 00 A0 [E0] [E1] [E2] FF

Comment:  A0 >> Message ID of instruction ENAη;
[E0] ~ [E2] >> Received data 0 ~ 2

Note:  This instruction sets ENAtimer only, can not enable controller.
In order to enable controller after pre-set time, user needs to configure initial configuration register by using instruction ICF after ENAtimer is set.
Require controller Firmware version being 1232 or higher.
10. **ENAxFFFF**  
Check enable time

**Format:** ENAxFFFF;

**Description:** Check auto-enable time

**ACK:** AA 00 A0 [E0] [E1] [E2] FF

**Comment:** Refer to ACK comment of instruction ENAη;

**Note:** This instruction checks ENAtimer only, can not enable controller. Require controller Firmware version being 1232 or higher.
11. **FBK**  
**Motor Status Feedback Inquiry**

**Format:**  
FBK;

**Description:**  
Check the current motor status

**ACK:**  
AA 00 [ASB] [CUR] [V0] [V1] [V2] [P0] [P1] [P2] [P3] [P4] FF

**Comment:**  
- [ASB] >> Received data 0
- [CUR] >> Received data 1
- [V0] ~ [P4] >> Received data 2 ~ 9

Structure of [ASB] is as follow:

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition</td>
<td>N/A (=0)</td>
<td>ACR</td>
<td>ENA / OFF</td>
<td>DIR</td>
<td>MCS – 1 (0 = full step, 15 = 1/16 step)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Structure of [CUR] is as follow:

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition</td>
<td>N/A (=0)</td>
<td>Phase Current (e.g. 27 = 2.7 Amp)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[V0] ~ [V2] is the converted value for the current motor speed. (16 bits) (Figure 9-1)

[P0] ~ [P4] is the converted value for the current motor displacement. (32 bits) (Figure 9-2)

**Note:**  
User can get current motor status by using this instruction at any time.  
Please note: current status is different from desired status.
12. ICFx\(\eta\)  Initial Configuration Register Instruction

**Format:**
ICFx\(\eta\);

**Description:** Configure the initial configuration register (InitCFG)
Parameter \(\eta\) has two bytes, structure is as follow:

<table>
<thead>
<tr>
<th>Byte</th>
<th><strong>0</strong></th>
<th><strong>1</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Definition</strong></td>
<td>D0</td>
<td>D1</td>
</tr>
</tbody>
</table>

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

Example:
Initial Configuration = 0x1234;
Then send instruction: ICFx 34 12;

**ACK:**
AA 00 DA [C0] [C1] [C2] FF

**Comment:**
DA >> Message ID of instruction ICFx\(\eta\);
[C0] ~ [C2] >> Received data 0 ~ 2

[C0] ~ [C2] is the converted value for the value of initial configuration register (16 bits) (Figure 9-1)

**Note:** Require controller Firmware version being 1232 or higher.
13. ICF  Check Initial Configuration Register

Format:  ICF;

Description:  Check initial configuration register

ACK:  AA 00 DA [C0] [C1] [C2] FF

Comment:  Refer to ACK comment of ICFx[1];

Note:  Require controller Firmware version being 1232 or higher.
14. **MAC\(\eta\) ** Set Acceleration Rate

**Format:**

MAC\(\eta\); 

**Description:**

Set the acceleration rate to \(\eta\).

\[ \eta = 1, 2 \ldots 65,000,000; \text{ (Pre-requiring MCFG<AM> = 0, value mode)} \]

\[ \eta = 1, 2 \ldots 60,000; \text{ (Pre-requiring MCFG<AM> = 1, period mode)} \]

**ACK:**

AA 00 B1 [FG] [A0] [A1] [A2] [A3] [A4] FF

**Comment:**

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

\([A0] ~ [A4]\) is the converted value for the value of the acceleration rate (32 bits) (Figure 9-2).
<table>
<thead>
<tr>
<th></th>
<th>MAC</th>
<th>Check Current Acceleration Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format:</td>
<td>MAC;</td>
<td></td>
</tr>
<tr>
<td>Description:</td>
<td>Check current acceleration rate</td>
<td></td>
</tr>
<tr>
<td>ACK:</td>
<td>AA 00 B1 [FG] [A0] [A1] [A2] [A3] [A4] FF</td>
<td>Refer to ACK comment of MAC11;</td>
</tr>
</tbody>
</table>
16. **MCF\( \eta \)/ MCFx\( \eta \) Master Configuration Register Instruction**

**Format:**

MCF\( \eta \); or MCFx\( \eta \);

**Description:**

Setup Master Configuration Register

1. If \( \eta \) is decimal,
   
   Use format: MCF\( \eta \);
   
   Where, \( \eta = 0,1,\ldots65535 \) (16 bits unsigned integer)

2. If \( \eta \) is hexadecimal,
   
   Use format MCFx\( \eta \);
   
   Where, \( \eta \) has 2 bytes, and the structure is as follow:

<table>
<thead>
<tr>
<th>Byte</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>D0</td>
</tr>
<tr>
<td>1</td>
<td>D1</td>
</tr>
</tbody>
</table>

Where,

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

For example:

Master Configuration = 0x1234;

Then send instruction MCFx 34 12;

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

**ACK:**

AA 00 B0 [C0] [C1] [C2] FF

**Comment:**

B0 \( \gg \) The Message ID of MCF\( \eta \);

[C0] ~ [C2] \( \gg \) Received data 0 ~ 2

[C0] ~ [C2] is the converted value for the value of master configuration register. (16 bits) (Figure 9-1)

**Note:**

If \( \eta \) using decimal, first convert the 16 bits binary number to a decimal based number.

**Example:**

Instruction : MCF34611; or MCFx8733;

ACK: AA 05 B0 02 0E 33 FF

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

0x8733 (34611 decimal). Here assume, Controller ID=5.
<table>
<thead>
<tr>
<th>17. MCF</th>
<th>Check Master Configuration Register</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Format:</strong></td>
<td>MCF;</td>
</tr>
<tr>
<td><strong>Description:</strong></td>
<td>Check the value of the Master Configuration Register</td>
</tr>
<tr>
<td><strong>ACK:</strong></td>
<td>AA 00 B0 [C0] [C1] [C2] FF</td>
</tr>
<tr>
<td><strong>Comment:</strong></td>
<td>Refer to ACK comment of MCF;</td>
</tr>
</tbody>
</table>
18. **MCS$\eta$ Setup Micro Stepping**

**Format:**

MCS$\eta$;

**Description:**

Set micro-stepping resolution.

$\eta = 1, 2, 4, 8, 16$ (unsigned integer);

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

**ACK:**

AA 00 [ASB] [CUR] [V0] [V1] [V2] [P0] [P1] [P2] [P3] [P4] FF

**Comment:**

[ASB] $>>$ Received data 0

[CUR] $>>$ Received data 1

[V0] $-$ [P4] $>>$ Received data 2 $-$ 9

[ASB] structure:

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>N/A (=0)</td>
<td>ACR</td>
<td>ENA / OFF</td>
<td>DIR</td>
<td>MCS $-1$ (0 = full step, 15 = 1/16 step)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[CUR] structure:

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>N/A (=0)</td>
<td>Phase Current (e.g. 27 = 2.7 Amp)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[V0] $-$ [V2] is the converted value for desired speed (16 bits) (Figure 9-1)

[P0] $-$ [P4] is the converted value for desired displacement (32 bits) (Figure 9-2)

**Note:**

Real-time update micro-stepping. MCS is short for Microstepping.

Once received, the MCS value will be stored in the controller’s EEPROM.
19. **MDE\(\eta\)** *Set Deceleration Rate*

**Format:**

\[ \text{MDE}\(\eta\); } \]

**Description:**

Set the deceleration rate to \(\eta\).

- \(\eta = 1 \cdot 2 \ldots 65,000,000\); (Pre-requiring MCFG<DM> = 0, value mode)
- \(\eta = 1 \cdot 2 \ldots 60,000\); (Pre-requiring MCFG<DM> = 1, period mode)

**ACK:**

\[ \text{AA } 00 \text{ B2 } [\text{FG}] [\text{D0}] [\text{D1}] [\text{D2}] [\text{D3}] [\text{D4}] \text{ FF} \]

**Comment:**

- B2 >> The message ID of MDE\(\eta\);
- [FG] >> Equal to the DM bit of the MCFG highlight DELM;
- Denote the input mode (value / period):
  - FG = 1, unit: ms;
  - FG = 0, unit: pps/s;
- [D0] ~ [D4] >> Received data 0 ~ 4

[D0] ~ [D4] is the converted value for the value of the deceleration rate (32 bits) (Figure 9-2).
20. MDE Check Current Deceleration Rate

Format: MDE;

Description: Check current deceleration rate

ACK: AA 00 B2 [FG] [D0] [D1] [D2] [D3] [D4] FF

Comment: Refer to ACK comment of MDE\textsubscript{11}.;
21. MDL Check Controller Model

Format: MDL;

Description: Check the Model, installed optional modules and firmware version of current controller.

ACK: CC 00 DE 18 02 [CUR] [asb] [V0] [V1] [V2] FF

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

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

[V0] ~ [V2] is the converted value for the firmware version (12 bits) (Figure 9-1).

Structure of [asb] is as follow:

<table>
<thead>
<tr>
<th>Bit Definition</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Int. QE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Closed-loop</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adv. Motion</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No. of sensor port</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For example, if bit 4 is 1, the Advanced Motion Control module is installed.
22. MMD\(\eta\)  Set Maximum Cessation Speed

**Format:**
MMD\(\eta\);

**Description:**
Set the maximum cessation speed at \(\eta\).
\(\eta = 1, 2 \ldots 65,000,000\); (unsigned integer)

**ACK:**
AA 00 B4 [M0] [M1] [M2] FF

**COMMENT:**
B4  \(\gg\) The message ID of MMD\(\eta\);
[M0] ~ [M2]  \(\gg\) Received data 0 ~ 2

[M0] ~ [M2] is the converted value for the value of maximum cessation speed. (16 bits) (Figure 9-1).

Unit: pps (pulse/second)
23. **MMD**  
Check current Maximum Cessation Speed

**Format:**  
MMD;

**Description:**  
Check the maximum cessation speed

**ACK:**  
AA 00 B4 [M0] [M1] [M2] FF

**Comment:**  
Refer to ACK comment of MMDη;
24. **MMS_\eta**  Set Maximum Starting Speed

**Format:**

MMS_\eta;

**Description:**

Set the maximum starting speed at \( \eta \).

\( \eta = 1 \cdot 2 \ldots 65,000,000 \); (unsigned integer)

**ACK:**

AA 00 B3 [M0] [M1] [M2] FF

**Comment:**

B3 >> The message ID of MMS_\eta;

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

[M0] ~ [M2] is the converted value for the value of maximum starting speed. (16 bits) (Figure 9-1).

Unit: pps (pluse/second).
25. **MMS**  

**Check current Maximum Starting Speed**

**Format:**  
MMS;

**Description:**  
Check the maximum starting speed

**ACK:**  
AA 00 B3 [M0] [M1] [M2] FF

**Comment:**  
Refer to ACK comment of MMS;
26. **OFF**  
**H- Bridge Disable**

**Format:**  
OFF;

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

**ACK:**  
AA 00 [ASB] [CUR] [V0] [V1] [V2] [P0] [P1] [P2] [P3] [P4] FF

**Comment:**  

| [ASB] | >> Received data 0 |
| [CUR] | >> Received data 1 |
| [V0] ~ [P4] | >> Received data 2 ~ 9 |

**[ASB] structure:**

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>N/A(=0)</td>
<td>ACR</td>
<td>ENA / OFF</td>
<td>DIR</td>
<td>MCS – 1(0 = full step, 15 = 1/16 step)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**[CUR] structure:**

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>N/A(=0)</td>
<td>Phase Current (e.g. 27 = 2.7 Amp)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[V0] ~ [V2] is the converted value for desired speed (16 bits) (Figure 9-1)  
[V0] ~ [P4] is the converted value for desired displacement (32 bits) (Figure 9-2)

**Note:**  

 Turns off the dual H-bridge motor driving circuit.  
Once controller is OFF, most devices of controller (including MOSFET) are turn off, the motor is free, and the logical circuit can work.
<table>
<thead>
<tr>
<th>27.</th>
<th>ORG</th>
<th>Reset Position Counter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format:</td>
<td>ORG;</td>
<td></td>
</tr>
<tr>
<td>Description:</td>
<td>Reset the position/encoder counter to zero (0), is equivalent to instruction ORG=0;</td>
<td></td>
</tr>
<tr>
<td>ACK:</td>
<td>AA 00 B7 [P0] [P1] [P2] [P3] [P4] FF</td>
<td></td>
</tr>
<tr>
<td>Comment:</td>
<td>Please refer to “29. POS;”.</td>
<td></td>
</tr>
</tbody>
</table>
## 28. ORG\(\eta\)  Reset Position Counter

**Format:**  
\[ \text{ORG}\(\eta\); \]

**Description:**  
Reset the position/encoder counter to a given value \(\eta\).  
\(\eta = -2147483647 \sim +2147483647\) (signed integer)  
\(\eta = 0\), reset the position/encoder counter to zero (0)  
\(\eta \neq 0\), reset the position/encoder counter to a given value \(\eta\)

**ACK:**  
AA 00 B7 [P0] [P1] [P2] [P3] [P4] FF

**Comment:**  
Please refer to “29. POS;”.  

29. **POSη**  Position Control

**Format:**  
POSη;

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

\[ \eta = -2147483647 \sim +2147483647 \text{ (signed integer)} \]

**ACK:**  
AA 00 B7 [P0] [P1] [P2] [P3] [P4] FF

**Comment:**  
B7 >> The message ID of desired position
[P0] ~ [P4] >> Received data 0 ~ 4

[P0] ~ [P4] is the converted value for the desired absolute position (32 bits) (Figure 10-2)

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

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

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

- User issues the instruction ORG
- User pre-configured sensor ORG event takes place.

User needs pay attention to the two notes list below:

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

The actual motor position is also relative to the micro-stepping resolution.
30. **POS**  
**Check Current Position**

**Format:** POS;

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

**ACK:** CC 00 B0 [P0] [P1] [P2] [P3] [P4] FF

**Comment:** B0 >> The message ID of current position  
[P0] ~ [P4] >> Received data 0 ~ 4

[P0] ~ [P4] is the converted value for the desired absolute position (32 bits) (Figure 9-2)

This position is relative to origin / zero position of counter.
31. **SCF\(\eta\) / SCFx\(\eta\) Set Sensor Configuration**

**Format:** SCF\(\eta\); or SCFx\(\eta\);

**Description:** Configure S12CON, ATCONH and ATCONL.

1) When \(\eta\) is decimal:
   - Instruction type is SCF\(\eta\);
   - Where, \(\eta = 0, 1 \ldots 1048575\) (24 bits unsigned integer)

2) When \(\eta\) is hexadecimal:
   - Instruction is SCFx\(\eta\);
   - Where \(\eta\) has 3 bytes, the structure is as follow:

<table>
<thead>
<tr>
<th>Byte</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition</td>
<td>D0</td>
<td>D1</td>
<td>IDX</td>
</tr>
</tbody>
</table>

   - Where,
     - [D1 D0] compose a hexadecimal 16bit data, D1 is high, D0 is low.
     - IDX = 0,1,2,3 denotes configuration of S12CON, ATCONH and ATCONL separately.

   Example:
   - Set S12CON as 0x1234;
   - Then send instruction SCFx 34 12 00; (00 is suffix)
   - Each Byte must have an even number of digits or letters.

**ACK:** AA 00 C0 [S0] [S1] [S2] [AL0] [AL1] [AH0] [AH1] FF

**Comment:**
- C0 \(>>\) The message ID of SCF\(\eta\);
- [S0] ~ [S2] \(>>\) Received data 0 ~ 2
- [AL0] ~ [AL1] \(>>\) Received data 3 ~ 4
- [AH0] ~ [AH1] \(>>\) Received data 5 ~ 6

- [S0] ~ [S2] is the converted value for [S12CON] (16 bits) (Figure 9-1),
- [AL0] [AL1] is the converted value for lower limit of analog threshold ATCONL (12 bits) (Figure 9-1)
- [AH0] [AH1] is the converted value for upper limit of analog threshold ATCONH (12 bits) (Figure 9-1)

**Note:**
- The suffix-code for S12CON is 00 (hexadecimal)
- The suffix-code for ATCONH is 02 (hexadecimal)
- The suffix-code for ATCONL is 03 (hexadecimal)
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>32. SCF</strong></td>
<td><strong>Check the value of Sensor Configuration</strong></td>
</tr>
<tr>
<td><strong>Format:</strong></td>
<td>SCF;</td>
</tr>
<tr>
<td><strong>Description:</strong></td>
<td>Check the current value of S12CON\ATCONH and ATCONL</td>
</tr>
<tr>
<td><strong>ACK:</strong></td>
<td>AA 00 C0 [S0] [S1] [S2] [AL0] [AL1] [AH0] [AH1] FF</td>
</tr>
<tr>
<td><strong>Comment:</strong></td>
<td>Refer to ACK comment of SCF;</td>
</tr>
</tbody>
</table>
33. **SFB**  

**Check Sensor Data**

- **Format:** SFB;

- **Description:** Check sensor readings and status

- **ACK:**  
  CC 00 C1 [D1] [D2] [AN0] [AN1] FF

- **Comment:**  
  C1 >> The message ID of SFB;  
  [D1] ~ [D2] >> Received data 1 ~ 2  
  [AN0] ~ [AN1] >> Received data 3 ~ 4

  [D1] ~ [D2] represent the logic level of S1, S2 respectively (0/1).  
  [AN0] [AN1] is the converted value for analog input (12 bits). (Figure 9-1)  
  AN1 and AN0 are 0 if no analog input port is configured.

- **Note:** This instruction can be used for sensor data inquiry at any time and under any condition.
34. **SPD_ \eta**  Speed Adjusting

<table>
<thead>
<tr>
<th>Format:</th>
<th>SPD_ \eta;</th>
</tr>
</thead>
</table>
| Description: | Set the desired speed to \eta.  
\eta = -65535...-1, 0, 1 ... + 65535; (signed integer) |
| ACK:         | AA 00 B5 [V0] [V1] [V2] FF |
| Comment:     | B5 >> The message ID for desired speed  
[V0] ~ [V2] >> Received data 0 ~ 2 |

[V0] ~ [V2] is the converted value for the value of desired speed. (16 bits)  
(Figure 9-1)

Unit: Pulse/ Second, PPS or Hz.

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

**Note:** Once H-bridge is enabled, motor starts running on receiving the instruction SPD_ \eta (\eta \neq 0) until another instruction "SPD0;" is given.

**Example:** For a 1.8° stepper motor, if the SPD = 100;
User sent: SPD100;
If MCS=1, motor speed = 1.8 * 100 = 180°/sec = 30 rpm  
If MCS=16, motr speed = 1.8 * 100 / 16 = 11.25°/sec = 1.875 rpm
35. SPD  Check Current Speed

Format: SPD;

Description: Check current speed

ACK: CC 00 B2 [V0] [V1] [V2] FF

Comment: B2 >> The message ID of current speed
[V0] ~ [V2] >> Received data 0 ~ 2

[V0] ~ [V2] is the converted value for the value of desired speed. (16 bits)
(Figure 9-1)

Unit: Pulse/ Second, PPS or Hz.

The sign of speed decides direction. If no “+” or “-” specified before “x”, it is taken as “+”.
36. **STGxη** Set Digital Input Sampling Mode

**Format:**

```
STGxη;
```

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

Structure of η:

<table>
<thead>
<tr>
<th>Byte</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defination</td>
<td>D0</td>
<td>D1</td>
<td>IDX</td>
</tr>
</tbody>
</table>

Where,

[D1 D0] compose a hexadecimal 16bit data, D1 is high, D0 is low.
IDX = 00,01 (hexadecimal) denotes configuring sensor 1,2;

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

[D1 D0] > EA60, denotes single sampling, once triggered, the S12CON must be configured again.

**ACK:**

```
AA 00 C9 [S0] [S1] [S2] [S3] [S4] [S5] FF
```

**Comment:**

C9 >> Message ID of instruction STGxη;
[S0] ~ [S5] >> Received data 0 ~ 5

[S0] ~ [S2] is the converted value for sampling mode of sensor 1 (16 bits) (Figure 9-1)
[S3] ~ [S5] is the converted value for sampling mode of sensor 2 (16 bits)

**Example:**

Requirements:

1) Sensor 1;
2) Intermittent mode, interval is 200ms.

Then:

1) IDX = 00 (hexadecimal)
2) [D1 D0] = 200 (decimal) = 00C8 (hexadecimal),
   So, D0 = C8, D1=00; (hexadecimal)
3) Then send instruction STGx C8 00 00;
37. STG Check Digital Input Sampling Mode

Format: STG;

Description: Check digital input sampling mode of S1, S2

ACK: AA 00 C9 [S0] [S1] [S2] [S3] [S4] [S5] FF

Comment: Refer to ACK comment of instruction STGx1;
38. STO EEPROM Store

Format: STO;

Description: Banding motion parameters to motions (Triggered by sensor edge or controlled by instruction)
Motion parameters include:
1) Acceleration
2) Deceleration
3) Max. starting speed
4) Max. cessation speed
For sensor, there also has:
5) Speed
6) Displacement

ACK: AA 00 D1 FF

Comment: D1  >> The message ID of STO;

Note: This instruction will affect real time performance.
It takes around 20 ms for the instruction to be executed. It is recommended that sending this instruction when the motor is idle, and wait 20ms before sending other instructions.
39. STO\(\eta\) Parameter Banding

**Format:**

\[ \text{STO}\eta; \]

**Description:**

Band motion parameters to motions (Triggered by sensor edge or controlled by instruction)

\[ \eta = 0, 1, \ldots, 5 \]

\(\eta = 0, \quad >> \text{Motion controlled by instruction} \)

\(\eta = 1, \quad >> \text{Only for close-loop} \)

\(\eta = 2, \quad >> \text{Motion triggered by rising edge of S1} \)

\(\eta = 3, \quad >> \text{Motion triggered by falling edge of S1} \)

\(\eta = 4, \quad >> \text{Motion triggered by rising edge of S2} \)

\(\eta = 5, \quad >> \text{Motion triggered by falling edge of S2} \)

**ACK:**

AA 00 D1 FF

**Comment:**

D1  \(>>\) Message ID of instruction STO;

**Note:**

Require controller Firmware version being 1232 or higher.

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

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

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

**Example:**

Disable the controller: OFF;

Set 1st group of parameters: SPD\(\eta\); STP\(\eta\); MAC\(\eta\); MDE\(\eta\); MMS\(\eta\); MMD\(\eta\);

Banding it to rising edge of S1: STO2;

......

Set 4th group of parameters: SPD\(\eta\); STP\(\eta\); MAC\(\eta\); MDE\(\eta\); MMS\(\eta\); MMD\(\eta\);

Banding it to falling edge of S2: STO5;
40. **STP\(\eta\) Displacement Control**

<table>
<thead>
<tr>
<th>Format:</th>
<th>STP(\eta);</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>Set the desired incremental displacement, i.e. the displacement relative to current position(\eta = -2,000,000,000\ldots-1, 0, 1 \ldots +2,000,000,000;) (signed integer)</td>
</tr>
<tr>
<td>ACK:</td>
<td>AA 00 B6 [P0] [P1] [P2] [P3] [P4] FF</td>
</tr>
<tr>
<td>Comment:</td>
<td>B6 &gt;&gt; The message ID of STP(\eta); [P0] ~ [P4] &gt;&gt; Received data 0 ~ 4</td>
</tr>
</tbody>
</table>

\([P0] ~ [P4]\) is the converted value for the desired motor displacement (32 bits) (Figure 9-2)

| Note:           | Displacement is essentially defined as counts of the pulse or encoder counter. Actual pulse sent to motor is controlled by displacement counter. The actual motor displacement is also relative to the micro-stepping resolution or encoder resolution. If an STP0; instruction is received before the former STP instruction is completed, UIM241 will execute the current instruction and stop motor. The former STP instruction is regarded as being completed. Meanwhile, system will shift from PT mode to VT mode. If an STP instruction is received while the motor is already running, the former steps will not be counted in the displacement of current STP instruction. |

| Example:        | For a 1.8° stepper motor, if STP = 200 pulse; User sent: STP200; If MCS=1, motor rotation angle = 1.8 * 200 = 360° If MCS=16, motor rotation angle = 1.8 * 200 / 16 = 22.5° |
41. **STP** Check Displacement

**Format:**

```
STP;
```

**Description:**

Check current incremental displacement.

**ACK:**

```
CC 00 B3 [P0] [P1] [P2] [P3] [P4] FF
```

**Comment:**

B3 >> The message ID of current incremental displacement

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

[P0] ~ [P4] is the converted value for the current incremental displacement (32 bits) (Figure 9-2)
APPENDIX A  DIMENSIONS

Units: mm
APPENDIX B  INSTALLATION INSTRUCTION

NEMA 17 (do not need flange)
1. Fix the UIM controller on motor with screw (2 or 4)
2. Connect the motor pin to motor terminal of UIM controller

NEMA 23/34/42 (need flange)
1. Fix flange on motor
2. Fix the UIM controller on flange with screw
3. Connect the motor pin to motor terminal of UIM controller