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in this publication regarding controller applications and the like is provided only for your convenience and may be superseded by updates.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

[UIM2842 Ordering Information]
In order to serve you quicker and better, please provide the product number in following format.

**UIM2842 PART NUMBERING SYSTEM**

![Ordering Information Diagram]
Miniature Integral Design
- Miniature size 23mm*28.1mm*15mm
- Fit onto motors seamlessly
- Die-cast aluminum enclosure, improving heat dissipation and durability

Motor Driving Characteristics
- Wide supply voltage range 12 ~ 28VDC
- Output current 1A, instruction adjustable
- Full to 16th micro-step resolution
- Dual full H-bridge with PWM constant current control

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

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

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

Advanced I / O Control (without host)
- 3 sensor input ports, 1 analog input (12bit)
- 1 TTL output
- 3 trigger mode (continuous / intermittent / single)
- 8 independent motion parameter group
- Pre-set action controlled by I/O
- I/O triggers real-time event-based change notification(RTCN)
- 12 real-time event-based change notifications
- 14 programmable actions triggered by 8 sensor events

Others
- Initial status configure
- Auto-lock when emergency
- User program
UIM2842

GENERAL DESCRIPTION

UIM2842 is a kind of miniature stepper motor controller with CAN network interface. UIM2842 controller can be mounted onto NEMA11 series stepper motor through adapting flanges. Total thickness of the controller is less than 15mm.

With UIM2842 controller, it is simple to construct a control system. Users can control the whole “motor-sensor-third party actuator” system through their own CAN based host by using “SimpleCAN” protocol. Users also can control the system through a gateway produced by UIrobot, such as UIM2501, USBC9100 and PCI120, by using RS232 based string or “SimpleCAN” protocol. One gateway can network with up to 100 UIM242 controllers.

UIM2842 can realize encoder-based closed-loop control. Its architecture includes communication system, basic motion control system, advanced motion control module (linear/non-linear acceleration/deceleration, S-curve PT/PVT displacement control), sensor input control module, TTL output control module and user programming module.

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

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

**Figure0-1: Connection Terminal - Front**

**Figure0-2: Connection Terminal – Back**

**Control Terminal**

<table>
<thead>
<tr>
<th>Terminal No.</th>
<th>Designator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+5V</td>
<td>5V voltage output (80mA)</td>
</tr>
<tr>
<td>2</td>
<td>AG</td>
<td>Analog ground for sensors</td>
</tr>
<tr>
<td>3</td>
<td>S3</td>
<td>Sensor input port 3</td>
</tr>
<tr>
<td>4</td>
<td>P4</td>
<td>TTL signal output port</td>
</tr>
<tr>
<td>5</td>
<td>S1</td>
<td>Sensor input port 1</td>
</tr>
<tr>
<td>6</td>
<td>S2</td>
<td>Sensor input port 2</td>
</tr>
<tr>
<td>7</td>
<td>CANH</td>
<td>CAN signal dominant high</td>
</tr>
<tr>
<td>8</td>
<td>CANL</td>
<td>CAN signal dominant low</td>
</tr>
<tr>
<td>9,10</td>
<td>GND</td>
<td>Supply voltage ground</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(positive and negative can not be connected wrong)</td>
</tr>
<tr>
<td>11,12</td>
<td>VCC</td>
<td>Supply voltage, 12~28VDC</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 measured by a multimeter.

Encoder Terminal

<table>
<thead>
<tr>
<th>Terminal No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+5V</td>
<td>Positive input of 5V power (For encoder, max. current 80mA)</td>
</tr>
<tr>
<td>GND</td>
<td>Ground of 5V power (For encoder, max. current 80mA)</td>
</tr>
<tr>
<td>A</td>
<td>A phase input of encoder (50KΩ pull-up resistor in the controller)</td>
</tr>
<tr>
<td>B</td>
<td>B phase input of encoder (50KΩ pull-up resistor in the controller)</td>
</tr>
<tr>
<td>Z</td>
<td>Z phase input of encoder</td>
</tr>
</tbody>
</table>

**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.
Miniature Integrated Stepper Motor Controller (Closed-loop)

TYPICAL APPLICATION

UIM2842 controllers can work standalone or within a CAN network. Working standalone means only one UIM2842 controller is linked to the CAN based host (such as UIM2501). When working in a CAN network, up to 100 UIM242 controllers can be linked together.

Under both scenarios, sensor input S1/S2/S3 should be connected to terminal 5/6/3, and signal ground should be connected to terminal 2. Furthermore, please be aware:

• User is responsible for the power supply for sensors.
• Voltage on terminal 5/6/3 must be kept between -0.3V and 5.3V
• Signal line of TTL output port P4 should be connected to terminal 4, and signal ground should be connected to AG port (terminal 2)
• For TTL output, the max sourcing / sinking current must be kept in 0~20mA.
• Output voltage of P4 is 0~5 V (Relative to terminal 2)

Furthermore, users must note:

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

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

When working stand-alone, user can use the wiring scheme shown in figure 0-3. Please note that, this wiring scheme should be used for setting the ID of a UIM2842 controller.

For long distance transfer, both ends of the CAN bus should be terminated with 120Ω terminating resistors. As UIM2501 converter has a build-in terminating resistor, user only needs to attach a resistor at the other end of the bus. Please refer to the UIM2501 user manual for how to enable the UIM2501 converter’s terminating resistor. CANH and CANL should use a twisted wire pair.

Figure 0-3: Wiring Scheme for Standalone Operation

Warning: Hot plugging is forbidden.

Warning: All controllers, gateways and subscriber equipments must be common-grounded.
Miniature Integrated Stepper Motor Controller (Closed-loop)

Network Operation

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

In figure 0-4, a wiring scheme is presented for such network operation with one RS232/CAN converter connected with multiple UIM2842 controllers. For detailed terminal wiring on each controller, please refer to figure 0-2.

Note:

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

Figure 0-4: Wiring Scheme for Network Operation

Warning: Hot plugging is forbidden.

Warning: All controllers, gateways and subscriber equipments must be common-grounded.
There is another wiring scheme of network in Figure 0-5. When wiring in this way, the length of stub does not need be shorter than 2CM, it is more flexible:

**Figure 0-5: Wiring Scheme for Network Operation-2**

![Wiring Scheme for Network Operation-2](image)

**Warning:** Hot plugging is forbidden.

**Warning:** All controllers, gateways and subscriber equipments must be common-grounded.
## INSTRUCTION SET SUMMARY

### Network Communication

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

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Feedback Header</th>
<th>Message ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>BTRn;</td>
<td>Set CAN network communication bit rate index</td>
<td>AA</td>
<td>BC</td>
</tr>
<tr>
<td>BTR;</td>
<td>Check current CAN network bit rate index</td>
<td>AA</td>
<td>BC</td>
</tr>
<tr>
<td>SETn;</td>
<td>Assign an to UIM242 controller</td>
<td>AA</td>
<td>DD</td>
</tr>
<tr>
<td>gOFF;</td>
<td>Disable H-bridge circuit</td>
<td>AA</td>
<td>AD</td>
</tr>
<tr>
<td>gCURn;</td>
<td>Set output phase current</td>
<td>AA</td>
<td>AD</td>
</tr>
<tr>
<td>gACRn;</td>
<td>Enable/disable automatic current reduction</td>
<td>AA</td>
<td>AD</td>
</tr>
<tr>
<td>gMCSn;</td>
<td>Set micro-stepping resolution</td>
<td>AA</td>
<td>AD</td>
</tr>
<tr>
<td>gORG;</td>
<td>Set zero/origin position</td>
<td>CC</td>
<td>AD</td>
</tr>
<tr>
<td>gSPDn;</td>
<td>Set the desired speed, the sign decides direction</td>
<td>AA</td>
<td>AD</td>
</tr>
<tr>
<td>gSTPn;</td>
<td>Set relative position, the sign decides direction</td>
<td>AA</td>
<td>AD</td>
</tr>
<tr>
<td>gPOSn;</td>
<td>Set desired position, the sign decides direction</td>
<td>AA</td>
<td>AD</td>
</tr>
<tr>
<td>gQECn;</td>
<td>Set encoder based position, the sign decides direction</td>
<td>AA</td>
<td>AD</td>
</tr>
<tr>
<td>gDOUTn;</td>
<td>Set output TTL level</td>
<td>AA</td>
<td>AD</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>MDLn;</td>
<td>Check the model of controller</td>
<td>CC</td>
<td>DE</td>
<td>82</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>ENAn;</td>
<td>Set enable time, boot time after n ms enable</td>
<td>AA</td>
<td>A0</td>
<td>70</td>
</tr>
<tr>
<td>ENAFFFF;</td>
<td>Check enable time</td>
<td>AA</td>
<td>A0</td>
<td>71</td>
</tr>
<tr>
<td>ICFn;</td>
<td>Set initial configuration register</td>
<td>AA</td>
<td>DA</td>
<td>73</td>
</tr>
<tr>
<td>ICF;</td>
<td>Check initial configuration register</td>
<td>AA</td>
<td>DA</td>
<td>74</td>
</tr>
<tr>
<td>MCFn;</td>
<td>Set master configuration register</td>
<td>AA</td>
<td>B0</td>
<td>77</td>
</tr>
<tr>
<td>MCF;</td>
<td>Check master configuration register</td>
<td>AA</td>
<td>B0</td>
<td>78</td>
</tr>
<tr>
<td>SCFn;</td>
<td>Set sensor control configuration register n</td>
<td>AA</td>
<td>C0</td>
<td>94</td>
</tr>
<tr>
<td>SCF;</td>
<td>Check sensor control configuration register</td>
<td>AA</td>
<td>C0</td>
<td>96</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>61</td>
</tr>
<tr>
<td>FBK;</td>
<td>Check current motor status</td>
<td>CC</td>
<td>-</td>
<td>72</td>
</tr>
<tr>
<td>SFB;</td>
<td>Check sensor status</td>
<td>CC</td>
<td>C1</td>
<td>97</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;</td>
<td>Set auto-current reduction ratio ( \eta )</td>
<td>AA</td>
<td>-</td>
<td>62</td>
</tr>
<tr>
<td>ACR;</td>
<td>Check auto-current reduction ratio</td>
<td>AA</td>
<td>BA</td>
<td>63</td>
</tr>
<tr>
<td>CUR;</td>
<td>Set output phase current ( \eta )</td>
<td>AA</td>
<td>-</td>
<td>66</td>
</tr>
<tr>
<td>ENA;</td>
<td>Enable H-bridge circuit</td>
<td>AA</td>
<td>-</td>
<td>69</td>
</tr>
<tr>
<td>MCS;</td>
<td>Set micro-stepping resolution</td>
<td>AA</td>
<td>-</td>
<td>79</td>
</tr>
<tr>
<td>OFF;</td>
<td>Disable H-bridge circuit</td>
<td>AA</td>
<td>-</td>
<td>87</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;</td>
<td>Set backlash compensation value ( \eta )</td>
<td>AA</td>
<td>DE</td>
<td>64</td>
</tr>
<tr>
<td>BLC;</td>
<td>Check backlash compensation value</td>
<td>AA</td>
<td>DE</td>
<td>65</td>
</tr>
<tr>
<td>MAC;</td>
<td>Set acceleration rate ( \eta )</td>
<td>AA</td>
<td>B1</td>
<td>75</td>
</tr>
<tr>
<td>MAC;</td>
<td>Check acceleration rate</td>
<td>AA</td>
<td>B1</td>
<td>76</td>
</tr>
<tr>
<td>MDE;</td>
<td>Set deceleration rate ( \eta )</td>
<td>AA</td>
<td>B2</td>
<td>80</td>
</tr>
<tr>
<td>MDE;</td>
<td>Check deceleration rate</td>
<td>AA</td>
<td>B2</td>
<td>81</td>
</tr>
<tr>
<td>MMD;</td>
<td>Set maximum cessation speed ( \eta )</td>
<td>AA</td>
<td>B4</td>
<td>83</td>
</tr>
<tr>
<td>MMD;</td>
<td>Check maximum cessation speed</td>
<td>AA</td>
<td>B4</td>
<td>84</td>
</tr>
<tr>
<td>MMS;</td>
<td>Set maximum starting speed ( \eta )</td>
<td>AA</td>
<td>B3</td>
<td>85</td>
</tr>
<tr>
<td>MMS;</td>
<td>Check maximum starting speed</td>
<td>AA</td>
<td>B3</td>
<td>86</td>
</tr>
<tr>
<td>ORG;</td>
<td>Set zero/origin position</td>
<td>AA</td>
<td>B7</td>
<td>88</td>
</tr>
<tr>
<td>ORG;</td>
<td>Reset the position to a given value ( \eta )</td>
<td>AA</td>
<td>B7</td>
<td>89</td>
</tr>
<tr>
<td>SPD;</td>
<td>Set the desired speed ( \eta )</td>
<td>AA</td>
<td>B5</td>
<td>98</td>
</tr>
<tr>
<td>SPD;</td>
<td>Check current speed</td>
<td>CC</td>
<td>B2</td>
<td>99</td>
</tr>
<tr>
<td>STO;</td>
<td>Bind motion control parameters to sensor edge</td>
<td>AA</td>
<td>D1</td>
<td>103</td>
</tr>
<tr>
<td>STP;</td>
<td>Set desired incremental displacement ( \eta )</td>
<td>AA</td>
<td>B6</td>
<td>105</td>
</tr>
<tr>
<td>STP;</td>
<td>Check current incremental displacement</td>
<td>CC</td>
<td>B3</td>
<td>106</td>
</tr>
</tbody>
</table>

## Closed-Loop 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>QEC;</td>
<td>Set desired quadrature encoder’s position ( \eta )</td>
<td>AA</td>
<td>B8</td>
<td>90</td>
</tr>
<tr>
<td>QEC;</td>
<td>Check current quadrature encoder’s position</td>
<td>CC</td>
<td>B1</td>
<td>91</td>
</tr>
<tr>
<td>QER;</td>
<td>Set quadrature encoder’s resolution ( \eta )</td>
<td>AA</td>
<td>C2</td>
<td>92</td>
</tr>
<tr>
<td>QER;</td>
<td>Check quadrature encoder’s resolution</td>
<td>AA</td>
<td>C2</td>
<td>93</td>
</tr>
<tr>
<td>SQT;</td>
<td>Set tolerance ( \eta ) of blocked alarm</td>
<td>AA</td>
<td>B8</td>
<td>100</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>DOU;</td>
<td>Set output TTL level</td>
<td>AA</td>
<td>C1</td>
<td>67</td>
</tr>
<tr>
<td>DOU;</td>
<td>Check current output TTL level</td>
<td>AA</td>
<td>C1</td>
<td>68</td>
</tr>
<tr>
<td>STG;</td>
<td>Set digital input sampling mode</td>
<td>AA</td>
<td>C9</td>
<td>101</td>
</tr>
<tr>
<td>STG;</td>
<td>Check digital input sampling mode</td>
<td>AA</td>
<td>C9</td>
<td>102</td>
</tr>
</tbody>
</table>
Miniature Integrated Stepper Motor Controller (Closed-loop)

CHARACTERISTICS

Absolute Maximum Ratings

<table>
<thead>
<tr>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply Voltage</td>
</tr>
<tr>
<td>Voltage on S1/S2/S3/P4 with respect to GND</td>
</tr>
<tr>
<td>Maximum output current sunk by S1/S2/S3/P4</td>
</tr>
<tr>
<td>Maximum output current sourced by S1/S2/S3/P4</td>
</tr>
<tr>
<td>Ambient temperature under bias</td>
</tr>
<tr>
<td>Storage temperature</td>
</tr>
</tbody>
</table>

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>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply Power Voltage</td>
</tr>
<tr>
<td>Motor Output Current</td>
</tr>
<tr>
<td>Driving Mode</td>
</tr>
<tr>
<td>Stepping Resolution</td>
</tr>
</tbody>
</table>

Communication (Ambient Temperature 25°C)

<table>
<thead>
<tr>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
</tr>
<tr>
<td>Wiring method</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CAN bus drive</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Supports 1 Mb/s operation</td>
</tr>
<tr>
<td>• ISO-11898 standard physical layer requirements</td>
</tr>
<tr>
<td>• Short-circuit protection</td>
</tr>
<tr>
<td>• High voltage transient protection</td>
</tr>
<tr>
<td>• Auto-thermal shutdown protection</td>
</tr>
<tr>
<td>• Up to 100 nodes can be connected</td>
</tr>
<tr>
<td>• Differential bus, high noise immunity</td>
</tr>
</tbody>
</table>

Environment Requirements

<table>
<thead>
<tr>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cooling</td>
</tr>
<tr>
<td>Working environment</td>
</tr>
<tr>
<td>Working temperature</td>
</tr>
<tr>
<td>Humidity</td>
</tr>
<tr>
<td>Vibration</td>
</tr>
<tr>
<td>Storage temperature</td>
</tr>
</tbody>
</table>
Size and Weight

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>23mm x 28.1mm x 15mm</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 .................................................................................................................... 11
Characteristics .................................................................................................................................... 13

1.0 Overview ....................................................................................................................................... 18
  1.1 Basic Control System ................................................................................................................... 18
  1.2 Advanced Motion Control Module ............................................................................................... 19
  1.3 Sensor Input Control Module ....................................................................................................... 19
  1.4 TTL Output Control Module ........................................................................................................ 20
  1.5 Encoder-based Closed-loop Control Module ............................................................................. 20
  1.6 Instructions and Interface ........................................................................................................... 20

2.0 Instruction and Feedback Structure .............................................................................................. 21
  2.1 UIM2842 Message Communication Mode ................................................................................ 21
  2.2 Instruction Structure ................................................................................................................... 22
  2.3 Macro Operator and Null Instruction ........................................................................................ 22
  2.4 Instruction List ........................................................................................................................... 23

3.0 CAN2.0 Communication .............................................................................................................. 24
  3.1 Controller ID Assignment .......................................................................................................... 24
  3.2 Instruction List .......................................................................................................................... 24

4.0 Real-time Change Notification ...................................................................................................... 25
  4.1 RTCN Structure ........................................................................................................................ 25
  4.2 Enable/Disable RTCN ................................................................................................................. 26

5.0 Initial and Hardware/Firmware Configuration .............................................................................. 27
  5.1 Initial Configuration Register ..................................................................................................... 27
  5.2 Auto-enable ................................................................................................................................ 28
  5.3 User Program ............................................................................................................................. 28
  5.4 Master Configuration Register .................................................................................................. 28
  5.5 Instruction List .......................................................................................................................... 29

6.0 Basic Control Instructions ........................................................................................................... 30
  6.1 General Introduction of Motion Control Modes .......................................................................... 30
  6.2 Basic Instruction Acknowledgment (ACK) ................................................................................ 32
  6.3 Motor Status Feedback Message ............................................................................................... 33
  6.4 Instruction List .......................................................................................................................... 34

7.0 Advanced Motion Control ........................................................................................................... 35
  7.1 Linear Acceleration ..................................................................................................................... 35
  7.2 Linear Deceleration ..................................................................................................................... 35
  7.3 Nonlinear Acceleration ................................................................................................................ 35
  7.4 Nonlinear Deceleration ................................................................................................................. 36
  7.5 S-curve Displacement Control .................................................................................................. 38
  7.6 Direction Control and Position Counter ..................................................................................... 39
  7.7 Backlash Compensation ............................................................................................................. 39
  7.8 Advanced Motion Control Instructions ....................................................................................... 40
  7.9 Enable/disable Advanced Motion Control Module (MCFG) ....................................................... 41
  7.10 Instruction List ........................................................................................................................ 41

8.0 Sensor Input Control ..................................................................................................................... 42
  8.1 Rising and Falling Edge ............................................................................................................. 45
  8.2 Analog Input and Thresholds ..................................................................................................... 45
  8.3 Digital Input Sampling Mode .................................................................................................... 46
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4</td>
<td>Sensor Event, Action and Binding</td>
<td>46</td>
</tr>
<tr>
<td>8.5</td>
<td>Introduction to Sensor Input Control Instructions</td>
<td>47</td>
</tr>
<tr>
<td>8.6</td>
<td>Sensor Input Control Register S12CON</td>
<td>48</td>
</tr>
<tr>
<td>8.7</td>
<td>Sensor Input Control Register S34CON</td>
<td>48</td>
</tr>
<tr>
<td>8.8</td>
<td>Analog Threshold Control Register ATCONH &amp; ATCONL</td>
<td>50</td>
</tr>
<tr>
<td>8.9</td>
<td>Instruction List</td>
<td>50</td>
</tr>
<tr>
<td>8.10</td>
<td>Example of S12CON Configuration</td>
<td>50</td>
</tr>
<tr>
<td>8.11</td>
<td>Example of ATCONH, ATCONL Configuration</td>
<td>51</td>
</tr>
<tr>
<td>9.0</td>
<td>Encoder and closed-loop control</td>
<td>53</td>
</tr>
<tr>
<td>9.1</td>
<td>Enable/ Disable Encoder and Closed-loop Module (MCFG)</td>
<td>53</td>
</tr>
<tr>
<td>9.2</td>
<td>Instruction List</td>
<td>54</td>
</tr>
<tr>
<td>10.0</td>
<td>TTL Output control</td>
<td>55</td>
</tr>
<tr>
<td>10.1</td>
<td>Introduction to TTL Output Control Instructions</td>
<td>55</td>
</tr>
<tr>
<td>10.2</td>
<td>TTL Output Control Register S34CON</td>
<td>55</td>
</tr>
<tr>
<td>10.3</td>
<td>Output Control Configuration Instruction (SCF)</td>
<td>56</td>
</tr>
<tr>
<td>10.4</td>
<td>Instruction List</td>
<td>56</td>
</tr>
<tr>
<td>10.5</td>
<td>Example of TTL Output Control and S34CON Configuration</td>
<td>57</td>
</tr>
<tr>
<td>11.0</td>
<td>Instruction</td>
<td>58</td>
</tr>
<tr>
<td>11.1</td>
<td>Instruction Structure</td>
<td>58</td>
</tr>
<tr>
<td>11.2</td>
<td>Feedback Message Structure</td>
<td>58</td>
</tr>
<tr>
<td>11.3</td>
<td>Instruction Description</td>
<td>61</td>
</tr>
<tr>
<td>1.</td>
<td>; Check desired motor status</td>
<td>61</td>
</tr>
<tr>
<td>2.</td>
<td>ACRη Set auto-current reduction ratio</td>
<td>62</td>
</tr>
<tr>
<td>3.</td>
<td>ACR Check auto-current reduction ratio</td>
<td>63</td>
</tr>
<tr>
<td>4.</td>
<td>BLCη Backlash compensation</td>
<td>64</td>
</tr>
<tr>
<td>5.</td>
<td>BLC Check backlash compensation</td>
<td>65</td>
</tr>
<tr>
<td>6.</td>
<td>CURη Motor Current Adjusting</td>
<td>66</td>
</tr>
<tr>
<td>7.</td>
<td>DOUη Set TTL Output</td>
<td>67</td>
</tr>
<tr>
<td>8.</td>
<td>DOU Check TTL Output Level</td>
<td>68</td>
</tr>
<tr>
<td>9.</td>
<td>ENA H-Bridge Enable</td>
<td>69</td>
</tr>
<tr>
<td>10.</td>
<td>ENAxxFFFF Check enable time</td>
<td>70</td>
</tr>
<tr>
<td>11.</td>
<td>ENAxFFFF Check enable time</td>
<td>71</td>
</tr>
<tr>
<td>12.</td>
<td>FBK Motor Status Feedback Inquiry</td>
<td>72</td>
</tr>
<tr>
<td>13.</td>
<td>ICFη Initial Configuration Register Instruction</td>
<td>73</td>
</tr>
<tr>
<td>14.</td>
<td>ICF Check Initial Configuration Register</td>
<td>74</td>
</tr>
<tr>
<td>15.</td>
<td>MACη Set Acceleration Rate</td>
<td>75</td>
</tr>
<tr>
<td>16.</td>
<td>MAC Check Current Acceleration Rate</td>
<td>76</td>
</tr>
<tr>
<td>17.</td>
<td>MCFη / MCFxη Master Configuration Register Instruction</td>
<td>77</td>
</tr>
<tr>
<td>18.</td>
<td>MCF Check Master Configuration Register</td>
<td>78</td>
</tr>
<tr>
<td>19.</td>
<td>MCSη Setup Micro Stepping</td>
<td>79</td>
</tr>
<tr>
<td>20.</td>
<td>MDEη Set Deceleration Rate</td>
<td>80</td>
</tr>
<tr>
<td>21.</td>
<td>MMDη Check Current Deceleration Rate</td>
<td>81</td>
</tr>
<tr>
<td>22.</td>
<td>MDLη Check Controller Model</td>
<td>82</td>
</tr>
<tr>
<td>23.</td>
<td>MMDη Set Maximum Cessation Speed</td>
<td>83</td>
</tr>
<tr>
<td>24.</td>
<td>MMD Check current Maximum Cessation Speed</td>
<td>84</td>
</tr>
<tr>
<td>25.</td>
<td>MMSη Set Maximum Starting Speed</td>
<td>85</td>
</tr>
<tr>
<td>26.</td>
<td>MMS Check current Maximum Starting Speed</td>
<td>86</td>
</tr>
<tr>
<td>27.</td>
<td>OFF H- Bridge Disable</td>
<td>87</td>
</tr>
<tr>
<td>28.</td>
<td>ORG Reset Position Counter</td>
<td>88</td>
</tr>
<tr>
<td>29.</td>
<td>ORGη Reset Position Counter</td>
<td>89</td>
</tr>
<tr>
<td>30.</td>
<td>QECη Set desired quadrature encoder’s position</td>
<td>90</td>
</tr>
<tr>
<td>31.</td>
<td>QEC Check Current Encoder Position</td>
<td>91</td>
</tr>
<tr>
<td>32.</td>
<td>QERη Set Quadrature Encoder Resolution</td>
<td>92</td>
</tr>
<tr>
<td>33.</td>
<td>QER Check Quadrature Encoder Resolution</td>
<td>93</td>
</tr>
</tbody>
</table>
Miniature Integrated Stepper Motor Controller (Closed-loop)

34. SCFη / SCFxη Set Sensor Configuration ............................................................... 94
35. SCF Check the value of Sensor Configuration ....................................................... 96
36. SFB Check Sensor Data .................................................................................... 97
37. SPDη Speed Adjusting ...................................................................................... 98
38. SPD Check Current Speed .................................................................................. 99
39. SQIη Set Tolerance of Blocked Alarm ................................................................. 100
40. STGη Set Digital Input Sampling Mode ............................................................... 101
41. STG Check Digital Input Sampling Mode ............................................................ 102
42. STOη Parameter Banding .................................................................................. 103
43. STPη Displacement Control ............................................................................. 105
44. STP Check Displacement .................................................................................. 106

APPENDIX A  Dimensions .......................................................................................... 107
APPENDIX B  Installation instruction ......................................................................... 108
1.0 OVERVIEW

UIM2842 is a kind of miniature integrated stepper motor controller with CAN2.0B Active bus communication capability.

UIM2842 has a size of 23mm*28.1mm*15mm and is designed to mount onto NEMA28 stepper motor seamlessly. UIM2842 can provide 0-1A output current, current value is adjustable within the range through instructions. Once set, the value is stored in EEPROM. UIM2842 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. UIM2842 series of controllers work with 12 ~ 28VDC power supply.

UIM2842 can perform encoder-based closed-loop motion control. The control system comprises communication system, basic motion control system, absolute position counter, quadrature encoder interface and real-time event-based change notification system. There are also four optional modules to be added on customer request: Advanced Motion Module (linear/non-linear acceleration/deceleration, S-curve PV/PVT displacement control), Encoder-based Closed-loop Control Module, Sensor Input control Module and User Program 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.

UIM2842 controller applies CAN2.0B communication protocol, which, due to its high-speed (1 million bit rate) long-distance (10km) transference and high noise immunity, is widely used in applications with serious signal interference and yet requiring high reliability, such as automobile industry, automated manufacturing and traffic control. The whole CAN bus network is based on a twisted wire pair. Similar to the network of home appliances, multiple UIM2842 controllers are connected to the twisted pair in parallel just like multiple pulps connected to the two-wire power cord. CAN bus network boosts many advantages, one of them is controllers never compete for bus transference.

A UIM2501 CAN-R232 converter is used to connect UIM2842 controller(s) to user device through serial port. Meanwhile, ASCII-coded instructions from user device are converted and transfers in CAN protocol in high speed to long distance reliably to control stepper motor(s)’ motion parameters such as direction, speed, steps, micro-steps, current, enable and disable the H-bridge. For network operation, each controller should be set a unique ID and up to 100 UIM2842 controllers can be controlled through this UIM2501 converter.

1.1 Basic Control System

UIM2842 controller’s basic control system comprises communication system, basic motion control system, absolute position counter, and real-time event-based change notification system.

Communication System

CAN bus protocol communication is used to realize the control to UIM2842. Through one CAN-RS232 converter (the UIM2501), user device can command multiple UIM2842 controllers through RS232 using ASCII coded instructions. The CAN bit rate can be changed through instruction.

Basic Motion Control
Miniature Integrated Stepper Motor Controller(Closed-loop)

UIM2842 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 / Quadrature Encoder Interface**

UIM2842 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.

UIM2842 controller has Quadrature Encoder Interface and can work with quadrature encoder when sensor input module is installed. Furthermore, with the encoder-based closed-loop control module, the UIM2842 can perform self closed-loop control.

**Real-time Change Notification (RTCN)**

Similar to CPU’s interrupters, UIM2842 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 115200. UIM2842’s RTCN system supports 12 events: displacement control done absolution position reset; sensor 1/2/3 rising edge and falling edge; analog input beyond upper threshold, analog input lower than lower threshold; and TTL status, etc. All RTCNs can be enabled or disabled by instructions.

1.2 Advanced Motion Control Module

With advanced motion control module installed, UIM2842 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

UIM2842’s Sensor Input Control Module supports 3 channels of sensor input. They can accept a TTL level input of 0~5V. There is 1 channel can be configured as analog input (Precision: 12bit; Sample frequency: 50K; mean of 16 calculation; Update frequency: 1000Hz). User can configure the desired automatic action triggered by sensor status change. There are 14 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 + 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
• Execute preset interrupt program (controller firmware version requires 1302)
• Off

1.4 TTL Output Control Module

UIM2842's TTL Output Control Module supports 1 channel of TTL voltage level output. The output port P4 is capable of providing +/-20mA sourcing or sinking current. In practice, please keep the current consumption as low as possible to avoid overheating the controller. Port P4 also can output setting level when detects events list below (pre-configuration):
• Run/Stop status. The output voltage level is determined by if the speed is zero or not.
• Direction change. The output voltage level is determined by if the current motor direction is forward or reverse.
• Origin point hit. The output voltage level is determined by if current position is zero point or just crosses over the zero point.

1.5 Encoder-based Closed-loop Control Module

With the encoder-based closed-loop control module, UIM2842 controller can perform self closed-loop motion control. Without this module, UIM2842 can still interface with a quadrature encoder and provide reading to user device, but the self closed-loop is not available.

1.6 Instructions and Interface

Instructions for UIM2842 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 UIM2842 receives a message (instructions) from the user device, it will first ACK back (repeat) the received instruction, and then execute the instruction. UIM2842 will further send back a message to inform the user device of the completion of the instruction. Before a new instruction is received, UIM2842 will keep current working status (e.g. running, stop, etc.)

2.1 UIM2842 Message Communication Mode

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

Message of UIM has two forms listing below:
1. String based on RS232 (Figure 2-1), and
2. CAN message based on UI simpleCAN (Figure 2-2).

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

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

Host sends string message to UI gateway (such as UIM2501) through RS232 serial port, then the gateway converses the message into CAN message based on SimpleCAN, and sends it to specified UIM2842 controller. Similarly, feedback message sent by UIM2842 is based on SimpleCAN, the gateway converts it to string based on RS232, and sends it to host.
2.2 Instruction Structure

An instruction is a message sent from the user device to UIM2842 to Comment certain operation. Instructions of UIM2842 follow the rules listed below:

\[ \text{INS} \eta; \text{ or INSx} \eta; \text{ or INS}; \]

Instruction symbol \text{INS} comprises three letters with no space between them, and is not case sensitive. If there is an \text{x} (\text{INSx}), then it means the value is hexadecimal. Value \eta 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 UIM2842 controller. The maximum length of feedback message is 13 bytes.

Feedback messages from UIM2842 (through UIM2501/2502) 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 CAN network (also known as Node ID). Scope: 5 – 125.

Message ID denotes the property of the current message.

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

Terminator denotes the end of a feedback message. UIM2842 controller utilizes “FF” or “FE” as the terminator. If terminator is “FF”, it means there is no follow-up message; If terminator is “FE”, it means there has follow-up messages.

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

2.3 Macro Operator and Null Instruction

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

\[ \text{CUR 20; MCS 16; SPD 5000; ENA;} \]

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; \};

UIM2842 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 UIM2842 to feedback all the inquired parameters of the basic motion control. (i.e. Enable/disable, Current, Micro-stepping, Auto current reduction, Direction, Speed, and Displacement) Actually, user can simply send the null instruction “;” alone to check the status of the above parameters. If there is no null instruction “;” after the “}” in the above example, there will be no ACK message at all.

### 2.4 Instruction List

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

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>;</td>
<td>Query desired motor parameters</td>
<td>61</td>
</tr>
</tbody>
</table>
3.0 CAN2.0 COMMUNICATION

In order to communicate with UIM2842 controller, a UIM2501 CAN-RS232 Converting Controller is required between the user device and the UIM2842. The user device sends ASCII coded instructions through RS232 port to the UIM2501 converter. Inside UIM2501, the RS232 based instructions are translated into CAN messages and sent to UIM2842 controllers.

With this UIM2501 converter, the user does not have to understand and deal with CAN bus operations but still enjoy the advantages of CAN bus, such as high speed, long distance, interference immunity, network, and easy wiring. UIM2501 is small in size, and is set up near the host, so the communication is quick and efficient. UIM2501 supports 115200 bps RS232 baud rate. The instruction takes about 1ms (0.001s) to transfer from user machine to UIM2842. At the same time, it only takes 50~100 us to transfer a message through SimpleCAN. This ensures the real-time of the system.

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

3.1 Controller ID Assignment

Before operation, a unique identification number (i.e., ID or address) is assigned to every UIM2842 controller needs to be. ID is used to identify which object is the instructions send to, and where the ACK is from.

Every UIM2842 controller has a factory default ID of 5. User can change the ID through instruction. Before assign an ID to a UIM2842 controller, please make sure the UIM2501 controller and the UIM2842 controller are connected together using the standalone operation scheme (Figure 0-3). A motor is not necessary.

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

Please Note: If there are two or more UIM2842 controllers with the same ID in a network, the network may not work properly. Before assign an ID to a UIM2842 controller, please make sure the UIM2501 controller and the UIM2842 controller are connected together using the standalone operation scheme.

3.2 Instruction List

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

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDLη;</td>
<td>Check the model of controller,function module and version</td>
<td>82</td>
</tr>
</tbody>
</table>

For details of CAN2.0 bit rate setting and global instructions, please refer to the UIM2501 user manual.

Note: Incorrect bit rate will result in communication failure or unstable.
4.0 REAL-TIME CHANGE NOTIFICATION

UIM2842 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 less than 0.5ms when the baud rate is 115200. 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 (except event 11):

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

The structure of RTCN message for event 11:

```
CC [Controller ID] [Message ID] [Closed-loop ID] [P0] [P1] [P2] [P3] [P4] FF
```

Where, Closed-loop ID = 0, means open-loop control;
Closed-loop ID = 0, means open-loop control.

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>falling edge of S3</td>
<td>A4</td>
<td>Voltage on S3 port: High &gt;&gt;&gt; Low</td>
</tr>
<tr>
<td>6</td>
<td>rising edge of S3</td>
<td>A5</td>
<td>Voltage on S3 port: Low &gt;&gt;&gt; High</td>
</tr>
<tr>
<td>7</td>
<td>TTL output P4 low</td>
<td>A6</td>
<td>Voltage on P4 port: High &gt;&gt;&gt; Low</td>
</tr>
<tr>
<td>8</td>
<td>TTL output P4 high</td>
<td>A7</td>
<td>Voltage on P4 port: Low &gt;&gt;&gt; High</td>
</tr>
<tr>
<td>9</td>
<td>exceed upper limits</td>
<td>A1/A5*</td>
<td>Analog input &gt; user preset upper limit</td>
</tr>
<tr>
<td>10</td>
<td>below lower limit</td>
<td>A0/A4**</td>
<td>Analog input &lt; user preset lower limit</td>
</tr>
<tr>
<td>11</td>
<td>displacement control complete</td>
<td>A8</td>
<td>The desired position is reached</td>
</tr>
<tr>
<td>12</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 S3 is configured as analog, A5 denotes event 9, otherwise A5 denotes event 6.
** When S1 is configured as analog, A0 denotes event 10, otherwise A0 denotes event 1.
  When S3 is configured as analog, A4 denotes event 10, otherwise A4 denotes event 5.
4.2 Enable/Disable RTCN

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

Please note, to realize the sensor event control, user needs to further configure the sensor control registers S34CON and S12CON. Please refer to Chapter 8.0 and Chapter 10.0 for details.
5.0 INITIAL AND HARDWARE/FIRMWARE CONFIGURATION

UIM2842’s hardware and firmware can be configured through user instructions. There are 5 configuration registers for UIM2842: Initial Configuration Register, Master Configuration Register, S12CON, S34CON and Analog Threshold Register. In this chapter, only the Initial Configuration Register and Mater Configuration Register are described. User can find details about the other registers in their corresponding chapters.

5.1 Initial Configuration Register

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 16-bit register with following structure:

<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>Value</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>E</td>
<td>Lock</td>
<td>PROG</td>
<td>CCW</td>
<td>ENA</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 Unimplemented. Read as 0.

Execute user program after power-on
(if the version of controller firmware is 1302)
- 0 = Not Excute user program after power-on
- 1 = Excute user program after power-on

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
5.2 Auto-enable

Once ICFG.ENA is set to 1, UIM2842 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 11.0.

5.3 User Program

User can program on UIM2842. Once ICFG.PROG is set to 1, UIM2842 will execute user program after the power is on. For details, please refer to “UIM Programming Manual”. UIM2842 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>MCFG</th>
<th>bit 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>value</td>
<td>ANE</td>
<td>CHS</td>
<td>QEI</td>
<td>X</td>
<td>QEM</td>
<td>CM</td>
<td>AM</td>
<td>DM</td>
<td>X</td>
<td>ST1IE</td>
<td>ORGIE</td>
<td>STPIE</td>
<td>P4IE</td>
<td>S3IE</td>
<td>S2IE</td>
<td>S1IE</td>
</tr>
</tbody>
</table>

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

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

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

Bit12 Unimplemented. Read as 0.

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

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

Bit9 AM Acceleration Mode
0 = Value mode. Unit is pps/sec, or pulse/ (square second)
Miniature Integrated Stepper Motor Controller (Closed-loop)

1 = Period mode. Unit is millisecond.

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

Bit7  Unimplemented. Read as 0.

Bit6  STLIE Locked-rotor Detection Variation Notification
  0 =  Disable locked-rotor detection variation notification (only for closed-loop)
  1 =  Enable locked-rotor detection variation notification. Once the error between pulsing counter and encoder counter is overstep, a message will be send to user device automatically.

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

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

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

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

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

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

5.5 Instruction List

The following table shows the instructions mentioned in this chapter, the detail of those instructions is 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&lt;sub&gt;n&lt;/sub&gt;;</td>
<td>Set initial configuration register</td>
<td>73</td>
</tr>
<tr>
<td>ICF;</td>
<td>Check initial configuration register</td>
<td>74</td>
</tr>
<tr>
<td>MCF&lt;sub&gt;n&lt;/sub&gt;;</td>
<td>Set master configuration register</td>
<td>77</td>
</tr>
<tr>
<td>MCF;</td>
<td>Check master configuration register</td>
<td>78</td>
</tr>
</tbody>
</table>
6.0 BASIC CONTROL INSTRUCTIONS

UIM2842 controllers support abundant motion control instructions. The instructions of UIM2842 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 UIM2842 motion control modes is provided.

6.1 General Introduction of Motion Control Modes

There are three motion control modes for UIM2842 controller: Velocity Tracking (VT), Position Tracking (PT) and Position Velocity Tracking (PVT).

Velocity Tracking (VT)

In the Velocity Tracking (VT) mode, UIM2842 controller controls the motor speed to track desired speed.

**Figure 6-1 Velocity Tracking**

```
<table>
<thead>
<tr>
<th>Speed</th>
<th>Instruction “SPD 1000;” received at this point</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Basic motion control, speed rises without acceleration process</td>
</tr>
<tr>
<td>1000</td>
<td>Advanced motion control, linear/non-linear acceleration</td>
</tr>
<tr>
<td>-1000</td>
<td>Basic motion control, speed falls without deceleration process</td>
</tr>
<tr>
<td>-1000</td>
<td>Advanced motion control, linear/non-linear deceleration</td>
</tr>
</tbody>
</table>
```

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

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

Position Tracking (PT)

In the Position Tracking (PT) mode, UIM2842 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. UIM2842 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, UI-M2842 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, UI-M2842 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.

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

UIM2842 controller will instruct motor to run at the desired speed until it reaches the desired position and then stop. User can enter, successively or discontinuously, both desired speed and desired position. Shifting between the three modes is displayed in the following chart:

Figure 6-3 Shifting between Motion Control Modes

6.2 Basic Instruction Acknowledgment (ACK)

Upon receiving an instruction, the UIM2842 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, UIM2842 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 UIM2842 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>Controller ID</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 10-1 for how to convert to a signed 16bit integer.
5. STP4 – STP0 denotes the desired motor displacement. See figure 10-2 for how to convert to a signed 32bit integer.

**6.3 Motor Status Feedback Message**

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

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

<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>Controller ID</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 10-1 for how to convert to a signed 16bit integer.
5. STP4 – STP0 denotes the current motor displacement. See figure 10-2 for how to convert to a signed 32bit integer.
For more details on above conversion, please refer to the source code of the provided demo software. These software and related source code are VC++/VB based and free.

6.4 Instruction List

The following table shows the instructions mentioned in this chapter, the detail of those instructions is 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>62</td>
</tr>
<tr>
<td>ACR;</td>
<td>Check auto-current attenuation ratio</td>
<td>63</td>
</tr>
<tr>
<td>CURη;</td>
<td>Set output phase current η</td>
<td>66</td>
</tr>
<tr>
<td>ENA;</td>
<td>Enable H-bridge circuit</td>
<td>69</td>
</tr>
<tr>
<td>ENAη;</td>
<td>Set enable time, boot after η ms enable</td>
<td>70</td>
</tr>
<tr>
<td>ENAxFFFF;</td>
<td>Check enable time</td>
<td>71</td>
</tr>
<tr>
<td>FBK;</td>
<td>Check current motor status</td>
<td>72</td>
</tr>
<tr>
<td>MCSη;</td>
<td>Set micro-stepping resolution</td>
<td>79</td>
</tr>
<tr>
<td>OFF;</td>
<td>Disable H-bridge circuit</td>
<td>87</td>
</tr>
<tr>
<td>ORG;</td>
<td>Set zero/origin position</td>
<td>88</td>
</tr>
<tr>
<td>ORGη;</td>
<td>Reset the position to a given value η</td>
<td>89</td>
</tr>
<tr>
<td>SPDη;</td>
<td>Set the desired speed η</td>
<td>98</td>
</tr>
<tr>
<td>SPD;</td>
<td>Check current speed</td>
<td>99</td>
</tr>
<tr>
<td>STPη;</td>
<td>Set desired incremental displacement η</td>
<td>105</td>
</tr>
<tr>
<td>STP;</td>
<td>Check current incremental displacement</td>
<td>106</td>
</tr>
</tbody>
</table>
7.0 ADVANCED MOTION CONTROL

UIM2482 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), UIM2482 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), UIM2482 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 UIM2482’s non-linear acceleration function. Experiments show that through non-linear acceleration,
UIM2842 can make NEMA17/23 4000RPM (quad step) in 0.25 seconds. UIM2842 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 UIM2842 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), UIM2842 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, UIM2842’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.

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

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

**Figure 7-11: S-curve Displacement Control (case 3)**

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.

UIM2842 has two types of position counters: absolute position counter and displacement counter.

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

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

7.7 Backlash Compensation

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

UIM2842 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
Miniature Integrated Stepper Motor Controller (Closed-loop)

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

In addition, notice that, if you want to store the value of set acceleration, deceleration, maximum start speed and maximum instantaneous stopping speed in the EEPROM, you need to send an instruction STO0; after sending the above instructions. Otherwise, after power failure and restarted, the parameter values will be restored to the factory value. In addition, the instruction STO0; affects the real-time performance of the control. See the Chapter 11.0 for details for more details.

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

7.9 Enable/disable Advanced Motion Control Module (MCFG)

Advanced Motion Control Module can be enabled or disabled by setting the CM bit of MCFG (MCFG<10>). Setting the CM bit (MCFG<CM>=1) will enable the module and clearing the CM bit (MCFG<CM>=0) will disable the advanced motion control module. (For details of setting, please refer to Section 5.4 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>64</td>
</tr>
<tr>
<td>BLC;</td>
<td>Check backlash compensation value</td>
<td>65</td>
</tr>
<tr>
<td>MAC(\eta);</td>
<td>Set acceleration rate (\eta)</td>
<td>75</td>
</tr>
<tr>
<td>MAC;</td>
<td>Check acceleration rate</td>
<td>76</td>
</tr>
<tr>
<td>MDE(\eta);</td>
<td>Set deceleration rate (\eta)</td>
<td>80</td>
</tr>
<tr>
<td>MDE;</td>
<td>Check deceleration rate</td>
<td>81</td>
</tr>
<tr>
<td>MMD(\eta);</td>
<td>Set maximum cessation speed (\eta)</td>
<td>83</td>
</tr>
<tr>
<td>MMD;</td>
<td>Check maximum cessation speed</td>
<td>84</td>
</tr>
<tr>
<td>MMS(\eta);</td>
<td>Set maximum starting speed (\eta)</td>
<td>85</td>
</tr>
<tr>
<td>MMS;</td>
<td>Check maximum starting speed</td>
<td>86</td>
</tr>
</tbody>
</table>
8.0 SENSOR INPUT CONTROL

UIM2842 Motion Controller has an optional (sold separately) Sensor Control Module which supports three sensor input ports: S1, S2 and S3. Port S2 can be configured for digital input (0-5V). Port S1 and S3 can be configured for either digital input or analog input.

Besides digital input condition circuit, UIM2842 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 50KHz. The analog feedback value is a mathematic average of 16 samples, and the update rate is 1000 Hz.

A clamping circuit (shown in the Figure 8-1) is internal connected to the sensor port, so regardless of whether it's digital or analog, the input voltage cannot exceed -0.3V ~ 5.3V, otherwise permanent damage can be done. If the TTL level at the sensor port is too low (LTTL sensor), the clamp circuit will not be aware of the TTL changes, and the controller will not be able to perform the action correctly on the edge of the TTL.

Therefore, once the level of the TTL sensor is not 5V/0V, you must take the corresponding measures between the sensor and the controller port:

![Figure 8-1 Clamping Circuit](image1)

Clamping Circuit (X=1,2,3)

1) Current-limiting Resistance

If the TTL level of sensor is slightly higher than 5V (12V/0V or 24V/0V), you can connect a resist ance for 1KΩ ~ 2KΩ in series between the sensor and the controller Sx port, to ensure that the Sx port voltage is less than 5V. (X=1,2,3)

![Figure 8-2 Current-limiting Resistance](image2)
2) NPN-type Triode

If the LTTL-type sensor (3.3V/0V) is used, you can add a NPN-type triode between the sensor and the controller $S_X$ port. Please note to adjust the resistance of $R$, to ensure that the $S_X$ port voltage is not higher than 5V. (X=1,2,3)

**Figure 8-3 NPN-type Triode**

3) Photoelectric Isolation Module

No matter which type of TTL-level sensor you use, connect a photoelectric isolation module between the sensor and the controller $S_X$ port, which can ensure the controller accurately and effectively identify the edge of TTL, and make the corresponding action. The photoelectric isolation module has strong anti-interference, and it is recommended to be used in factories, interference occasions. Please note that the resistance of $R$ should be adjusted to ensure that the current flowing through the photoelectric isolation module is 10mA. (X=1,2,3)

**Figure 8-4 Photoelectric Isolation Module**

Note: You can choose the photoelectric isolation module of UIROBOT company or you can also configure by yourself.

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, UIM2842 can perform motion controls without the user device.
There are 8 sensor events that can be configured, as listed below:

Table 8-1: Sensor Events

<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>S3 Falling Edge</td>
<td>S3 Voltage Level Change, High &gt;&gt;&gt; Low</td>
</tr>
<tr>
<td>6</td>
<td>S3 Rising Edge</td>
<td>S3 Voltage Level Change, Low &gt;&gt;&gt; High</td>
</tr>
<tr>
<td>7</td>
<td>Exceeding the Upper Limit</td>
<td>Analog input voltage is higher than user defined upper limit</td>
</tr>
<tr>
<td>8</td>
<td>Exceeding the Lower Limit</td>
<td>Analog input voltage is lower than user defined lower limit</td>
</tr>
</tbody>
</table>

There are 14 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
- Execute the preset interrupt program
- 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 8-5: Rising and Falling Edge of a Digital Sensor Input](image)

8.2 Analog Input and Thresholds

![Figure 8-6: Analog Input and Thresholds](image)

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

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

Digital input of UIM2842 has three sampling modes:

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

In different sampling mode is respectively following three trigger modes: zero-interval continuous triggering, N millisecond interval triggering and single triggering.

Continuous Sampling

In continuous sampling mode, UIM2842 controllers detect level fluctuation at port S1/S2/S3 uninterrupted. 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, UIM2842 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, UIM2842 controllers will not detect the level fluctuation at this port until user configures the corresponding control bit of S12CON (or S34CON) again.

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

8.4 Sensor Event, Action and Binding

UIM2842 supports 8 sensor events as listed in section 8.0. There are 14 actions that can be bound to those 8 sensor events. Binding means assigning a sensor action to a sensor event. The binding between events and actions are realized through the configuration of the Sensor Control Register S12CON. An action-code is needed when configuring sensor registers.

- Start and run forwardly at preset-speed and acceleration (code: 10)
- Start and run reversely at preset-speed and acceleration (code: 2)
- Change direction and run at preset-speed and acceleration (code: 14)
- 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)
Miniature Integrated Stepper Motor Controller (Closed-loop)

- Decelerate at preset deceleration until stop (code: 3)
- Emergency stop (code: 4)
- Reset position and encoder counter (code: 6)
- Reset position and encoder counter + Reverse displacement control follow the preset motion parameters (speed, displacement, acceleration) (code: 7)
- Reset position and encoder counter + Decelerate at preset deceleration until stop (code: 11)
- Reset position and encoder counter + Emergency stop (code: 12)
- Execute the preset interrupt program (code: 8)*
- Off (code: 15)

Note: *The version of controller firmware is required 1302.

8.5 Introduction to Sensor Input Control Instructions

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

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

2. SCF (Sensor Configuration Register)
   SCF is used to configure following sensor input control registers: S12CON, S34CON, ATCONH and ATCONL.

3. STG (Sensor Trigger Configuration)
   STG is used to configure sensor trigger mode, UIM242 has three trigger mode: 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:

| Bit 15-12 | S2RACT<3:0> | S2 Rising-edge Action |
| Bit 11-8  | S2FACT<3:0> | S2 Falling-edge Action |
| Bit 7-4   | S1RACT<3:0> | S1 Rising-edge Action |
| Bit 3-0   | S1FACT<3:0> | S1 Falling-edge Action |

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

<table>
<thead>
<tr>
<th>ACT Code (binary)</th>
<th>Action</th>
<th>RTCN or Not</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>N/A</td>
<td>No RTCN (Ignore MCFG&lt;S2IE&gt;&lt;S1IE&gt;)</td>
</tr>
<tr>
<td>0001</td>
<td>N/A</td>
<td>Depends on MCFG&lt;S2IE&gt;&lt;S1IE&gt;</td>
</tr>
<tr>
<td>0010</td>
<td>Start and Run Reversely</td>
<td>Depends on MCFG&lt;S2IE&gt;&lt;S1IE&gt;</td>
</tr>
<tr>
<td>0011</td>
<td>Decelerate until Stop</td>
<td>Depends on MCFG&lt;S2IE&gt;&lt;S1IE&gt;</td>
</tr>
<tr>
<td>0100</td>
<td>Emergency Stop</td>
<td>Depends on MCFG&lt;S2IE&gt;&lt;S1IE&gt;</td>
</tr>
<tr>
<td>0101</td>
<td>Reverse Displacement Control</td>
<td>Depends on MCFG&lt;S2IE&gt;&lt;S1IE&gt;</td>
</tr>
<tr>
<td>0110</td>
<td>Reset position</td>
<td>Depends on MCFG&lt;S2IE&gt;&lt;S1IE&gt;</td>
</tr>
<tr>
<td>0111</td>
<td>Reset position + Displacement Control</td>
<td>Depends on MCFG&lt;S2IE&gt;&lt;S1IE&gt;</td>
</tr>
<tr>
<td>1000</td>
<td>Execute preset interrupt program*</td>
<td>Depends on MCFG&lt;S2IE&gt;&lt;S1IE&gt;</td>
</tr>
<tr>
<td>1001</td>
<td>Direction-change displacement control</td>
<td>Depends on MCFG&lt;S2IE&gt;&lt;S1IE&gt;</td>
</tr>
<tr>
<td>1010</td>
<td>Start and Run Forwardly</td>
<td>Depends on MCFG&lt;S2IE&gt;&lt;S1IE&gt;</td>
</tr>
<tr>
<td>1011</td>
<td>Reset position + Decelerate until Stop</td>
<td>Depends on MCFG&lt;S2IE&gt;&lt;S1IE&gt;</td>
</tr>
<tr>
<td>1100</td>
<td>Reset position + Emergency Stop</td>
<td>Depends on MCFG&lt;S2IE&gt;&lt;S1IE&gt;</td>
</tr>
<tr>
<td>1101</td>
<td>Forward Displacement Control</td>
<td>Depends on MCFG&lt;S2IE&gt;&lt;S1IE&gt;</td>
</tr>
<tr>
<td>1110</td>
<td>Change direction and run</td>
<td>Depends on MCFG&lt;S2IE&gt;&lt;S1IE&gt;</td>
</tr>
<tr>
<td>1111</td>
<td>OFF</td>
<td>Depends on MCFG&lt;S2IE&gt;&lt;S1IE&gt;</td>
</tr>
</tbody>
</table>

Note: *The version of controller firmware is required 1302.

8.7 Sensor Input Control Register S34CON

S34CON（Sensor3 / Port4 Control） defines the binding relationship between S3 sensor events and actions, as well as the activation of corresponding RTCNs. It is a 16bits register inside the controller, and can be configured using the instruction SCF. When writing to it user needs to affix a 4bits suffix-code to point to this register. For details of SCF, please refer to chapter 11.

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

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

| Bit 15-12 | S3RACT | S3 Rising-edge Action |
| Bit 11-8  | P4LVL  | P4 Level |
| Bit 7-4   | P4EVENT | P4 Event |
| Bit 3-0   | S3FACT | S3 Falling-edge Action |
Bit 15-12 **Blocked alarm**

Action code after blocked is as follows:

<table>
<thead>
<tr>
<th>ACT (binary)</th>
<th>Action</th>
<th>RTCN or Not</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>N/A</td>
<td>No RTCN (Ignore MCFG&lt;S2IE&gt;)</td>
</tr>
<tr>
<td>0001</td>
<td>N/A</td>
<td>Depends on MCFG&lt;S2IE&gt;</td>
</tr>
<tr>
<td>0010</td>
<td>N/A</td>
<td>Depends on MCFG&lt;S2IE&gt;</td>
</tr>
<tr>
<td>0011</td>
<td>Emergency Stop</td>
<td>Depends on MCFG&lt;S2IE&gt;</td>
</tr>
<tr>
<td>0100</td>
<td>N/A</td>
<td>Depends on MCFG&lt;S2IE&gt;</td>
</tr>
<tr>
<td>0101</td>
<td>N/A</td>
<td>Depends on MCFG&lt;S2IE&gt;</td>
</tr>
<tr>
<td>1000</td>
<td>Execute preset interrupt program*</td>
<td>Depends on MCFG&lt;S2IE&gt;</td>
</tr>
<tr>
<td>1001</td>
<td>Direction-change displacement control</td>
<td>Depends on MCFG&lt;S2IE&gt;</td>
</tr>
<tr>
<td>1010</td>
<td>N/A</td>
<td>Depends on MCFG&lt;S2IE&gt;</td>
</tr>
<tr>
<td>1011</td>
<td>N/A</td>
<td>Depends on MCFG&lt;S2IE&gt;</td>
</tr>
<tr>
<td>1100</td>
<td>Reset position + Emergency Stop</td>
<td>Depends on MCFG&lt;S2IE&gt;</td>
</tr>
<tr>
<td>1101</td>
<td>N/A</td>
<td>Depends on MCFG&lt;S2IE&gt;</td>
</tr>
<tr>
<td>1110</td>
<td>Change direction and run</td>
<td>Depends on MCFG&lt;S2IE&gt;</td>
</tr>
<tr>
<td>1111</td>
<td>OFF</td>
<td>Depends on MCFG&lt;S2IE&gt;</td>
</tr>
</tbody>
</table>

Note: * The version of controller firmware is required 1302.

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

Please refer to chapter 9.0 for details.

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

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

The action code caused by S3 level edge (the same as the S1/S2):

<table>
<thead>
<tr>
<th>ACT Code (binary)</th>
<th>Action</th>
<th>RTCN or Not</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>N/A</td>
<td>No RTCN (Ignore MCFG&lt;S2IE&gt;&lt;S1IE&gt;)</td>
</tr>
<tr>
<td>0001</td>
<td>N/A</td>
<td>Depends on MCFG&lt;S2IE&gt;&lt;S1IE&gt;</td>
</tr>
<tr>
<td>0010</td>
<td>Start and Run Reversely</td>
<td>Depends on MCFG&lt;S2IE&gt;&lt;S1IE&gt;</td>
</tr>
<tr>
<td>0011</td>
<td>Decelerate until Stop</td>
<td>Depends on MCFG&lt;S2IE&gt;&lt;S1IE&gt;</td>
</tr>
<tr>
<td>0100</td>
<td>Emergency Stop</td>
<td>Depends on MCFG&lt;S2IE&gt;&lt;S1IE&gt;</td>
</tr>
<tr>
<td>0101</td>
<td>Reverse Displacement Control</td>
<td>Depends on MCFG&lt;S2IE&gt;&lt;S1IE&gt;</td>
</tr>
<tr>
<td>0110</td>
<td>Reset position</td>
<td>Depends on MCFG&lt;S2IE&gt;&lt;S1IE&gt;</td>
</tr>
<tr>
<td>0111</td>
<td>Reset position + Displacement Control</td>
<td>Depends on MCFG&lt;S2IE&gt;&lt;S1IE&gt;</td>
</tr>
<tr>
<td>1000</td>
<td>Execute preset interrupt program*</td>
<td>Depends on MCFG&lt;S2IE&gt;&lt;S1IE&gt;</td>
</tr>
<tr>
<td>1001</td>
<td>Direction-change displacement control</td>
<td>Depends on MCFG&lt;S2IE&gt;&lt;S1IE&gt;</td>
</tr>
<tr>
<td>1010</td>
<td>Start and Run Forwardly</td>
<td>Depends on MCFG&lt;S2IE&gt;&lt;S1IE&gt;</td>
</tr>
<tr>
<td>1011</td>
<td>Reset position + Decelerate until Stop</td>
<td>Depends on MCFG&lt;S2IE&gt;&lt;S1IE&gt;</td>
</tr>
<tr>
<td>1100</td>
<td>Reset position + Emergency Stop</td>
<td>Depends on MCFG&lt;S2IE&gt;&lt;S1IE&gt;</td>
</tr>
<tr>
<td>1101</td>
<td>Forward Displacement Control</td>
<td>Depends on MCFG&lt;S2IE&gt;&lt;S1IE&gt;</td>
</tr>
<tr>
<td>1110</td>
<td>Change direction and run</td>
<td>Depends on MCFG&lt;S2IE&gt;&lt;S1IE&gt;</td>
</tr>
<tr>
<td>1111</td>
<td>OFF</td>
<td>Depends on MCFG&lt;S2IE&gt;&lt;S1IE&gt;</td>
</tr>
</tbody>
</table>

Note: * The version of controller firmware is required 1302.
8.8 Analog Threshold Control Register ATCONH & ATCONL

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

ATCONH structure is as follows:

<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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AH &lt;11:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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:

<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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
</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).

8.9 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>95</td>
</tr>
<tr>
<td>SCF;</td>
<td>Check sensor control configuration register</td>
<td>97</td>
</tr>
<tr>
<td>SFB;</td>
<td>Check sensor status</td>
<td>98</td>
</tr>
<tr>
<td>STGη;</td>
<td>Set digital input sampling mode</td>
<td>102</td>
</tr>
<tr>
<td>STG;</td>
<td>Check digital input sampling mode</td>
<td>104</td>
</tr>
<tr>
<td>STO;</td>
<td>Store motion control parameters</td>
<td></td>
</tr>
<tr>
<td>STOη;</td>
<td>Bind motion control parameters to sensor edge</td>
<td>105</td>
</tr>
</tbody>
</table>

8.10 Example of S12CON Configuration

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

**System Description**

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

**Requirements:**
Miniature Integrated Stepper Motor Controller (Closed-loop)

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

**Realization:**
1. First stop the motor by sending: OFF;
2. We are not interested in the rising edge, so set S2RACT<3:0> = 0000
3. It is required Start and Run Reversely on S2 failing edge, so, set S2FACT<3:0> =0010
4. Same as 2, set S1RACT<3:0> = 0000
5. It is required Start and Run Forwardly on S1 failing edge, so, set S1FACT<3:0> =1010
6. Fill the S12CON with above bits, get: S12CON = 0000 0010 0000 1010 (binary)
7. Affix the suffix-code 0000 to S12CON, get:
   SCFG = 0000 0010 0000 1010 0000 (binary)=0x20A0 (hex)=8352 (decimal)
8. Send instruction: SCFx 20A0; or SCF 8352;
9. Set up desired speed, by sending instruction: SPD 5000;
10. Burn parameters into EEPROM, by sending: STO;
11. Press any one of the limit sensors, the mobile platform will work.
12. Disconnect the user device, and restart the UIM242 controller, the system will automatically run.
13. If enable auto-feedback, once the motor touches limit switch, user device will receive a feedback message of falling-edge on port S1/S2.

**8.11 Example of ATCONH, ATCONL Configuration**

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

**System Description**

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

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

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

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

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

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

UIM2842 controller has a built-in quadrature encoder (hereinafter referred to as encoder) interface circuit, which is capable of decoding encoder signals of less than 200KHz input. Another option is user can connect external encoder of their own choice to UIM2842 controller, using S1 and S2 ports for channel A and B. In this case, however, INDEX decoding function is not available. S1/S2 supports 0-5V TTL input. The input range for S1 and S2 ports of UIM2842 controller is -0.3V ~ 5.3V. Any input beyond this range can result in permanent damage. Also, for this case, encoder power supply is to be provided by user.

For UIROBOT UIM2842 controller with internal encoder, the S1 and S2 ports are not occupied and therefore are available for sensors. Whether the encoder is built-in or external, the controlling mode and the instructions are the same.

Instructions relative to encoder control function are listed below:

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Function</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MCF enable encoder function</td>
<td>MCF 1792;</td>
</tr>
<tr>
<td>2</td>
<td>QEC encoder-based position control</td>
<td>QEC - 200000;</td>
</tr>
<tr>
<td>3</td>
<td>STP encoder-based displacement control</td>
<td>STP 500;</td>
</tr>
<tr>
<td>4</td>
<td>QER set encoder resolution</td>
<td>QER 500;</td>
</tr>
<tr>
<td>5</td>
<td>SQT Set tolerance of blocked alarm</td>
<td>SQT 150;</td>
</tr>
</tbody>
</table>

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

**Enable Encoder Interface**

The Encoder Decoding Module is enabled / disabled through configuring the QEI bit of MCFG (MCFG<13>). When MCFG<QEI>=0, the encoder decoding module is disabled; when MCFG<QEI>=1, the encoder decoding module is enabled. If external encoder is used, S1 and S2 ports must be used for channel A and channel B respectively. If user chooses UIROBOT internal encoder, S1 and S2 ports are available for sensors. Please
Enable Closed-loop Control Module

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

Please note, closed-loop control module is a must even if user uses external encoders. Otherwise, UIM2842 controller can only read the external encoder data, but cannot maintain closed-loop motion control with this data. However, if the internal encoder is installed, Closed-loop Control Module is automatically included.

For master configuration register (MCFG), please refer to Section 5.4.

9.2 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>QECη;</td>
<td>Set desired quadrature encoder’s position η</td>
<td>90</td>
</tr>
<tr>
<td>QEC;</td>
<td>Check current quadrature encoder’s position</td>
<td>91</td>
</tr>
<tr>
<td>QERη;</td>
<td>Set quadrature encoder’s resolution η</td>
<td>92</td>
</tr>
<tr>
<td>QER;</td>
<td>Check quadrature encoder’s resolution</td>
<td>93</td>
</tr>
<tr>
<td>SQTη;</td>
<td>Set tolerance η of blocked alarm</td>
<td>100</td>
</tr>
</tbody>
</table>
Miniature Integrated Stepper Motor Controller(Closed-loop)

10.0 TTL OUTPUT CONTROL

UIM2842 controller has an optional TTL Output Control Module (sold separately) that supports 1 channel of TTL level output. This output port (P4) is capable of providing 20mA sourcing or sinking current. In practice, please keep the current as low as possible to prevent overheating the controller. Port P4 also can output setting level when detects events list below (pre-configuration):

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

10.1 Introduction to TTL Output Control Instructions

There are 3 instructions related to the TTL output control.

1) **MCF** The P4IE bit (MCFG<3>) of the master configuration register enables/disables the P4 real-time change notification (RTCN). For details, please refer to section 5.1.

2) **SCF** is used to configure the register S34CON. S34CON is shared by Sensor 3 and TTL output. When it works as TTL output, it defines the relationship between events and output level.

3) **DOU** is used to directly control the TTL output voltage level as well as check current voltage level.

10.2 TTL Output Control Register S34CON

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

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

<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>STALL</td>
<td>P4LVL</td>
<td>P4EVENT</td>
<td>S3RACT</td>
<td>S3FACT</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>

Bit 15-12 **Blocked alarm**
Action code after blocked is as follow:

<table>
<thead>
<tr>
<th>ACT (binary)</th>
<th>Action</th>
<th>RTCN or Not</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>N/A</td>
<td>No RTCN (Ignore MCFG&lt;S2IE&gt;)</td>
</tr>
<tr>
<td>0001</td>
<td>N/A</td>
<td>Depends on MCFG&lt;S2IE&gt;</td>
</tr>
<tr>
<td>0010</td>
<td>N/A</td>
<td>Depends on MCFG&lt;S2IE&gt;</td>
</tr>
<tr>
<td>0011</td>
<td>N/A</td>
<td>Depends on MCFG&lt;S2IE&gt;</td>
</tr>
<tr>
<td>0100</td>
<td>Emergency Stop</td>
<td>Depends on MCFG&lt;S2IE&gt;</td>
</tr>
<tr>
<td>0101</td>
<td>N/A</td>
<td>Depends on MCFG&lt;S2IE&gt;</td>
</tr>
<tr>
<td>0110</td>
<td>N/A</td>
<td>Depends on MCFG&lt;S2IE&gt;</td>
</tr>
<tr>
<td>1000</td>
<td>Execute preset interrupt program*</td>
<td>Depends on MCFG&lt;S2IE&gt;</td>
</tr>
<tr>
<td>1001</td>
<td>Direction-change displacement control</td>
<td>Depends on MCFG&lt;S2IE&gt;</td>
</tr>
<tr>
<td>1010</td>
<td>N/A</td>
<td>Depends on MCFG&lt;S2IE&gt;</td>
</tr>
<tr>
<td>1011</td>
<td>N/A</td>
<td>Depends on MCFG&lt;S2IE&gt;</td>
</tr>
<tr>
<td>1100</td>
<td>Reset position + Emergency Stop</td>
<td>Depends on MCFG&lt;S2IE&gt;</td>
</tr>
<tr>
<td>1101</td>
<td>N/A</td>
<td>Depends on MCFG&lt;S2IE&gt;</td>
</tr>
<tr>
<td>1110</td>
<td>Change direction and run</td>
<td>Depends on MCFG&lt;S2IE&gt;</td>
</tr>
<tr>
<td>1111</td>
<td>OFF</td>
<td>Depends on MCFG&lt;S2IE&gt;</td>
</tr>
</tbody>
</table>

Note: * The version of controller firmware is required 1302.

Bit 11  P4LVL  Port P4 output voltage level

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

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

<table>
<thead>
<tr>
<th>P4EVENT (binary)</th>
<th>Action</th>
<th>RTCN or Not</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>No action. Output is controlled by instruction</td>
<td>Depends on MCFG&lt;P4IE&gt;</td>
</tr>
<tr>
<td>001</td>
<td>When SPD&gt;0, Output = P4LVL, vice versa.</td>
<td>Depends on MCFG&lt;P4IE&gt;</td>
</tr>
<tr>
<td>010</td>
<td>When move forward, Output = P4LVL, vice versa.</td>
<td>Depends on MCFG&lt;P4IE&gt;</td>
</tr>
<tr>
<td>111</td>
<td>When POS=0, Output = P4LVL, vice versa.</td>
<td>Depends on MCFG&lt;P4IE&gt;</td>
</tr>
</tbody>
</table>

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

10.3 Output Control Configuration Instruction (SCF)

Please refer to chapter 8 for detailed information.

10.4 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>DOUη;</td>
<td>Set output TTL level η</td>
<td>67</td>
</tr>
<tr>
<td>DOU;</td>
<td>Check current output TTL level</td>
<td>68</td>
</tr>
</tbody>
</table>
10.5 Example of TTL Output Control and S34CON Configuration

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

System Description
A reciprocating mobile platform

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

Realization
1. First stop the motor by sending OFF;
2. Set MCFG<P4IE>=1, get:
   MCFG = 0000 0000 0000 1000 (binary) = 0x8 (hex)= 8 (decimal)
3. Send instruction: MCF 8;
4. Set P4EVENT <2:0>=010, link to direction event
5. Set P4LVL=1, so when motor moves forward, P4 will output 5V
6. Fill the S34CON with above bits, get: S34CON = 0000 1010 0000 0000 (binary)
7. Affix the suffix-code 0001 to S34CON, get:
   SCFG =0000 1010 0000 0000 0001 (binary)=0xA001 (hex)=40961 (decimal)
8. Send instruction: SCF xA001; or SCF 40961;
9. Send instruction: ENA;
10. Run the motor. There are numerous ways to run the motor. The easiest way is using SPDη;

During the motion, once actual direction (NOT desired direction) is plus, P4 will output 5V and RTCN. Vice verse.
This chapter describes the detail of the instructions mentioned in this document.

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

11.1 Instruction Structure

An instruction is a message sent from the user device to motion controller to command certain operation. Instructions of UIM2842 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.

11.2 Feedback Message Structure

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

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

\[
\text{[Header]} \ [\text{Controller ID}] \ [\text{Message ID}] \ [\text{Data}] \ [\text{Terminator}]
\]
Miniature Integrated Stepper Motor Controller (Closed-loop)

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

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

Message ID
The property of the current message
For example, CC 05 A0 FF, where A0 denotes that there is a low level on sensor 1. For details, please refer to following sections.

Data
Has a 7bits data structure. High is in front, and low is in the back.
In figure 12-1 and 12-2, examples are shown on how to convert a set of 7-bit data into 16-bit data and 32-bit data.
Obviously, one 16-bit data takes three 7-bit data to represent, and one 32-bit data takes five 7-bit 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 12-1: Conversion from three 7bits message data to a 16bits data

Data received (Hex): [XX] [XX] … [1stByte] [2ndByte] [3rdByte] … [XX] [XX] [ FF ]
Receive sequence: earlier ➔ later
Figure 12-2: Conversion from five 7-bit message data to a 32-bit data

Data received (Hex): [XX] [XX] ... [1\text{st} Byte] [2\text{nd} Byte] [3\text{rd} Byte] [4\text{th} Byte] [5\text{th} Byte] ... [XX] [XX] [FF]

Received sequence: earlier \rightarrow later

1\text{st} Byte: 0 0 0 0 D31 D30 D29 D28
2\text{nd} Byte: 0 D27 D26 D25 D24 D23 D22 D21
3\text{rd} Byte: 0 D20 D19 D18 D17 D16 D15 D14
4\text{th} Byte: 0 D13 D12 D11 D10 D9 D8 D7
5\text{th} Byte: 0 D6 D5 D4 D3 D2 D1 D0

16-bit Binary Data: D31 D30 D29 D28 D27 D26 D25 D24 D23 D22 D21 D20 D19 D18 D17 D16 D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0
11.3 Instruction Description

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

1. ; Check desired motor status

Format: ;

Description: Check desired motor status

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

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

[ASB] structure:

<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 12-1)

[P0] ~ [P4] is the converted value for desired displacement (32 bits) (Figure 12-2)
2. **ACR\(\eta\)**  Set auto-current reduction ratio

**Format:**

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

**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 \(\times \eta / 100.\)

**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 [Controller ID] BA [A0] FF

**Comment:**

BA  >> Message ID of instruction ACR\(\eta;\)

[A0]  >> 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 hardware version being 1232 or higher.
### Miniature Integrated Stepper Motor Controller (Closed-loop)

#### 3. ACR  Check auto-current reduction ratio

<table>
<thead>
<tr>
<th>Format:</th>
<th>ACR;</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong></td>
<td>Check auto-current reduction ratio</td>
</tr>
<tr>
<td><strong>ACK:</strong></td>
<td>AA  [Controller ID]  BA  [A0]  FF</td>
</tr>
<tr>
<td><strong>Comment:</strong></td>
<td>Refer to ACK comment of instruction ACRη</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>Require controller hardware version being 1232 or higher.</td>
</tr>
</tbody>
</table>
### 4. BLC\(\eta\) Backlash compensation

**Format:**
\[ \text{BLC}\eta; \]

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

**ACK:**
\[ \text{AA [Controller ID] DE [B0] [B1] [B2] FF} \]

**Comment:**
\[ \text{DE} \quad >> \quad \text{Message ID of instruction BLC}\eta; \]
\[ \text{[B0] ~ [B2]} \quad >> \quad \text{Received data 0 ~ 2} \]

\[ \text{[B0] ~ [B2]} \text{ is the converted value for the value of backlash compensation (16 bits) (Figure 12-1)} \]
5. BLC  Check backlash compensation

Format: BLC;

Description: Check the value of backlash compensation in reciprocating motion

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

Comment: Refer to ACK comment of instruction BLC;
6. CUR$_\eta$  Motor Current Adjusting

**Format:**  
CUR$_\eta$;

**Description:**  
Set the output phase current to $\eta$.

$\eta = 0,1,\ldots,80$ (unsigned integer)

0…80 represent 0…8.0 amps.

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

**Comment:**

[ASB]  $>>$ Received data 0

[CUR]  $>>$ Received data 1

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

[ASB] structure:

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

[CUR] structure:

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

[V0] ~ [V2] is the converted value for desired speed (16 bits) (Figure 12-1)

[P0] ~ [P4] is the converted value for desired displacement (32 bits) (Figure 12-2)

**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.
Miniature Integrated Stepper Motor Controller (Closed-loop)

7. **DOU\(\eta\)**  Set TTL Output

<table>
<thead>
<tr>
<th>Format:</th>
<th>DOU(\eta);</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>Set TTL output level</td>
</tr>
<tr>
<td>(\eta = 0, 1)</td>
<td></td>
</tr>
<tr>
<td>Comment:</td>
<td>C1 &gt;&gt; The message ID of DOU(\eta);</td>
</tr>
<tr>
<td></td>
<td>P4 &gt;&gt; The logic level of the TTL output</td>
</tr>
<tr>
<td></td>
<td>P4 = 1 means the output is 5V</td>
</tr>
<tr>
<td></td>
<td>P4 = 0 means the output is 0V</td>
</tr>
<tr>
<td>Note:</td>
<td>Using DOU(\eta); will affect S34CON.</td>
</tr>
<tr>
<td></td>
<td>Once DOU(\eta); instruction is received, UIM242 controller will clear P4LVL and P4EVENT&lt;2:0&gt;. Therefore, if user wants to re-bind the events to the output control, user needs to reconfigure S34CON. This is to prevent potential confliction between user instruction and events controlled output.</td>
</tr>
</tbody>
</table>
8. **DOU**  Check TTL Output Level

**Format:**  DOU;

**Description:**  Check current TTL output level

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

**COMMENT:**  Refer to ACK comment of DOUη;

**Note:**  Using DOU; will NOT affect S34CON.
Miniature Integrated Stepper Motor Controller (Closed-loop)

9. ENA H-Bridge Enable

Format: ENA;

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

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

Comment:

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

[ASB] structure:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>N/A (=0)</th>
<th>ACR</th>
<th>ENA / OFF</th>
<th>DIR</th>
<th>MCS – 1(0 = full step. 15 = 1/16 step)</th>
</tr>
</thead>
</table>

[CUR] structure:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>N/A (=0)</th>
<th>Phase Current (e.g. 27 = 2.7 Amp)</th>
</tr>
</thead>
</table>

[V0] ~ [V2] is the converted value for desired speed (16 bits) (Figure 12-1)

[P0] ~ [P4] is the converted value for desired displacement (32 bits) (Figure 12-2)

Note: Only after the H-bridge enabled, can the controller drive the motor.
10. ENA$_\eta$  Set enable time

Format:  ENA$_\eta$;

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

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

Comment:  A0  >>  Message ID of instruction ENA$_\eta$;
[E0] ~ [E2]  >>  Received data 0 ~ 2

[E0] ~ [E2] is the converted value for auto-ENA time (16 bits) (Figure 12-1), units: ms.

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

**Format:**  
ENAFFFF;

**Description:**  
Check auto-enable time

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

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

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

Format:  FBK;

Description:  Check the current motor status

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

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

Structure of [ASB] is as follow:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Definition</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>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Structure of [CUR] is as follow:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Definition</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></td>
</tr>
</tbody>
</table>

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

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

Note:  User can get current motor status by using this instruction at any time.
Please note: current status is different from desired status.
13. **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>0</th>
<th>1</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 [Controller ID] 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 12-1)

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

**Format:**  
ICF;

**Description:**  
Check initial configuration register

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

**Comment:**  
Refer to ACK comment of ICFx

**Note:**  
Require controller hardware version being 1232 or higher.
15. MAC\eta Set Acceleration Rate

Format: \textit{MAC\eta;}

Description: Set the acceleration rate to \eta.
\eta = 1 \cdot 2 \text{...} 65,000,000; \text{(Pre-requiring MCFG<AM> = 0, value mode)}
\eta = 1 \cdot 2 \text{...} 60,000; \text{(Pre-requiring MCFG<AM> = 1, period mode)}

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

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

[A0] \sim [A4] \text{is the converted value for the value of the acceleration rate}
\text{(32 bits) (Figure 12-2).}
16. MAC  Check Current Acceleration Rate

Format:  MAC;

Description:  Check current acceleration rate

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

Comment:  Refer to ACK comment of MAC11;.
17. 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, \(\eta = 0,1,...,65535\) (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>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition</td>
<td>D0</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 [Controller ID] B0 [C0] [C1] [C2] FF

**Comment:**
B0 >> The Message ID of MCF\(\eta\);
[C0] ~ [C2] >> Received data 0 ~ 2

[C0] ~ [C2] is the converted value for the value of master configuration register. (16 bits) (Figure 12-1)

**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>No.</th>
<th>Command</th>
<th>Description</th>
<th>Format</th>
<th>Acknowledgment</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>MCF</td>
<td>Check Master Configuration Register</td>
<td>MCF;</td>
<td>AA [Controller ID] B0 [C0] [C1] [C2] FF</td>
<td>Refer to ACK comment of MCF;</td>
</tr>
</tbody>
</table>
Miniature Integrated Stepper Motor Controller(Closed-loop)

19. **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 [Controller ID] [ASB] [CUR] [V0] [V1] [V2] [P0] [P1] [P2] [P3] [P4] FF

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

[ASB] structure:

<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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Phase Current (e.g. 27 = 2.7 Amp)</td>
<td></td>
</tr>
</tbody>
</table>

[V0] ~ [V2] is the converted value for desired speed (16 bits) (Figure 12-1)  
[P0] ~ [P4] is the converted value for desired displacement (32 bits) (Figure 12-2)

**Note:**  
Real-time update micro-stepping. MCS is short for Microstepping.  
Once received, the MCS value will be stored in the controller’s EEPROM.
20. \textbf{MDE}_\eta \hspace{1cm} \textbf{Set Deceleration Rate} \\

\textbf{Format:} \hspace{1cm} \text{MDE}_\eta ; \\

\textbf{Description:} \hspace{1cm} Set the deceleration rate to $\eta$.  
\hspace{1cm} $\eta = 1 \cdot 2 \cdots 65,000,000$; (Pre-requiring $\text{MCFG}<\text{DM}> = 0$, value mode)  
\hspace{1cm} $\eta = 1 \cdot 2 \cdots 60,000$; (Pre-requiring $\text{MCFG}<\text{DM}> = 1$, period mode) \\

\textbf{ACK:} \hspace{1cm} AA \ [\text{Controller ID}] \ B2 \ [\text{FG}] \ [\text{D0}] \ [\text{D1}] \ [\text{D2}] \ [\text{D3}] \ [\text{D4}] \ FF \\

\textbf{Comment:} \hspace{1cm} B2 \hspace{1cm} >> \hspace{0.5cm} \text{The message ID of MDE}_\eta ;  
\hspace{1cm} [\text{FG}] \hspace{1cm} >> \hspace{0.5cm} \text{Equal to the DM bit of the MCFG DM}  
\hspace{1cm} \hspace{0.5cm} \text{Denote the input mode (value / period):}  
\hspace{1cm} \hspace{0.5cm} \hspace{0.5cm} \text{FG} = 1, \text{unit: ms};  
\hspace{1cm} \hspace{0.5cm} \hspace{0.5cm} \text{FG} = 0, \text{unit: pps/s};  
\hspace{1cm} [\text{D0}] \sim [\text{D4}] \hspace{1cm} >> \hspace{0.5cm} \text{Received data 0 \sim 4}  

[\text{D0}] \sim [\text{D4}] \hspace{0.5cm} \text{is the converted value for the value of the deceleration rate}  
\hspace{0.5cm} \text{(32 bits)} \hspace{0.5cm} (\text{Figure11-2}).
### Miniature Integrated Stepper Motor Controller(Closed-loop)

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>21.</td>
<td>MDE</td>
<td>Check Current Deceleration Rate</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Format:**  
MDE;

**Description:** Check current deceleration rate

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

**Comment:** Refer to ACK comment of MDEη;.
22. MDLη Check Controller Model

Format: MDLη;

Description: Check the Model, installed optional modules and firmware version of the UIM242 controller of ID =η.
η = 5, 6 … 125.

ACK: CC [Controller ID] DE 18 02 [CUR] [asb] [V0] [V1] [V2] FF
Note: [ ] denotes one byte, the data is hexadecimal.

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

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

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>Defination</td>
<td>0</td>
<td>Int. QE</td>
<td>Closed-loop</td>
<td>Adv. Motion</td>
<td>No. of sensor port</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For example, if bit 4 is 1, the Advanced Motion Control module is installed.
23. 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 [Controller ID] B4 [M0] [M1] [M2] FF

COMMENT: B4 >> The message ID of MMD$\eta$;
[M0] ~ [M2] >> Received data 0 ~ 2

[M0] ~ [M2] is the converted value for the value of maximum cessation speed. (16 bits) (Figure 12-1).
Unit: pps (pulse/second)
24. **MMD**  
Check current Maximum Cessation Speed

**Format:**  
MMD;

**Description:**  
Check the maximum cessation speed

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

**Comment:**  
Refer to ACK comment of MMD
25. **MMSη**  Set Maximum Starting Speed

**Format:**  
MMSη;

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

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

**Comment:**  
B3 >> The message ID of MMSη;  
[M0] ~ [M2] >> Received data 0 ~ 2

[M0] ~ [M2] is the converted value for the value of maximum starting speed. (16 bits) (Figure11-1).  
Unit: pps (pluse/second).
26. MMS  Check current Maximum Starting Speed

Format:  MMS;

Description:  Check the maximum starting speed

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

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

Format:  OFF;

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

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

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

[ASB] structure:

<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>5</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

[VO] ~ [V2] is the converted value for desired speed (16 bits) (Figure 12-1)

[P0] ~ [P4] is the converted value for desired displacement (32 bits) (Figure 12-2)

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

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

Format: ORG;

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

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

Comment: Please refer to “31. POS;”. 
29. ORGη  Reset Position Counter

Format:  ORGη;

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

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

Comment:  Please refer to “31. POS;”.  

30. QEC\(\eta\)  Set desired quadrature encoder's position

Format:  
\[\text{QEC}\eta;\]

Description:  
Set desired encoder position to \(\eta\)
\[\eta = -2,000,000,000 \ldots -1, 0, 1 \ldots +2,000,000,000.\]

ACK:  
AA [Controller ID] B8 [Q0] [Q1] [Q2] [Q3] [Q4] FF

Comment:  
B8  >>  Message ID of desired encoder position;
[Q0] ~ [Q4]  >>  Received data 0 ~ 4

[Q0] ~ [Q4] is the converted value for desired encoder position. (32 bits) (Figure 12-2)

Note:  
The encoder counter records encoder pulses.

When the direction is positive, the counter increases; when the direction is negative, the counter decreases. When ICFG.CW = 0, consider clockwise as forward direction; when ICFG.CW = 1, consider anticlockwise as forward direction.

Encoder counter can only be reset/cleared under following situations:

-- Commanded by user instruction ORG
-- User preset sensor ORG event happens

Please also be aware:

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

-- For every slot, the encoder counter records 4 pulses. E.g., when QEC=500, the encoder counter records 500\*4 = 2000 pulses each turn.
31. **QEC**  
**Check Current Encoder Position**

**Format:**  
QEC;

**Description:**  
Check current encoder position

**ACK:**  
CC [Controller ID] B1 [Q0] [Q1] [Q2] [Q3] [Q4] FF

**Comment:**  
B1 >> Message ID of current encoder position;  
[Q0] ~ [Q4] >> Received data 0 ~ 4

[Q0] ~ [Q4] is the converted value for current encoder position. (32 bits)  
(Figure 12-2)
32. QERᵦ Set Quadrature Encoder Resolution

Format: QERᵦ;

Description: Set the quadrature encoder resolution at η
η = 0, 1 … 65000.

ACK: AA [Controller ID] C2 [R0] [R1] [R2] FF

Comment: C2 >> Message ID of QER;
[R0] ~ [R2] >> Received data 0 ~ 2

[R0] ~ [R2] is the converted value for encoder resolution. (16 bits)
(Figure 12-1)

Note: Incorrect QER value can result in unpredictable closed-loop control operations!
33. **QER**  Check Quadrature Encoder Resolution

<table>
<thead>
<tr>
<th>Format:</th>
<th>QER;</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong></td>
<td>check current quadrature encoder resolution</td>
</tr>
<tr>
<td><strong>ACK:</strong></td>
<td>AA [Controller ID] C2 [R0] [R1] [R2] FF</td>
</tr>
<tr>
<td><strong>Comment:</strong></td>
<td>Refer to ACK comment of instruction QERη;</td>
</tr>
</tbody>
</table>
34. **SCF_η / SCFx_η Set Sensor Configuration**

**Format:** SCF_η; or SCFx_η;

**Description:** Configure S12CON, S34CON, ATCONH and ATCONL.

1) When _η_ is decimal:
   - Instruction type is SCF_η;
   - Where, _η_ = 0, 1 ... 1048575 (24 bits unsigned integer)
   - Refer to Chapter 8.

2) When _η_ is hexadecimal:
   - Instruction is SCFx_η;
   - Where _η_ 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>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 = 0, 1, 2, 3 denotes configuration of S12CON, S34CON, ATCONH and ATCONL separately.

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

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

**Comment:**
- C0 >> The message ID of SCF_η;
- [S0] ~ [S4] >> Received data 0 ~ 4
- [AL0] ~ [AL1] >> Received data 5 ~ 6
- [AH0] ~ [AH1] >> Received data 7 ~ 8

[S0] ~ [S4] is the converted value for [S34CON : S12CON] (32 bits) (Figure 12-2), where, 16 bits high denotes S34CON, 16 bits low denotes S12CON.

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

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

**Note:** The suffix-code for S12CON is 00 (hexadecimal)
Miniature Integrated Stepper Motor Controller (Closed-loop)

The suffix-code for S34CON is 01 (hexadecimal)
The suffix-code for ATCONH is 02 (hexadecimal)
The suffix-code for ATCONL is 03 (hexadecimal)
35. SCF  Check the value of Sensor Configuration

Format: SCF;

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

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

Comment: Refer to ACK comment of SCF;
Miniature Integrated Stepper Motor Controller (Closed-loop)

36. SFB Check Sensor Data

Format: SFB;

Description: Check sensor readings and status

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

COMMENT: C1 >> The message ID of SFB;
[D1] ~ [D3] >> Received data 1 ~ 3
[AN0] ~ [AN1] >> Received data 4 ~ 5

[D1] ~ [D3] represent the logic level of S1, S2 and S3 respectively (0/1).
[AN0] [AN1] is the converted value for analog input (12 bits). (Figure 12-1)
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.
37. SPD\(\eta\)  Speed Adjusting

Format:  SPD\(\eta\);

Description:  Set the desired speed to \(\eta\).
\(\eta = -65535 \ldots -1, \ 0, \ 1 \ldots \ +65535; \) (signed integer)

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

Comment:  B5  >> The message ID for desired speed
[V0] ~ [V2]  >> Received data 0 ~ 2

[V0] ~ [V2] is the converted value for the value of desired speed. (16 bits)
(Figure11-1)
Unit: Pluse/ Second, PPS or Hz.
The sign of speed decides direction. If no “+” or “-” specified before “x”, it is taken as “+”.

Note:  Once H-bridge is enabled, motor starts running on receiving the instruction 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, motor speed = 1.8 * 100 / 16 = 11.25°/sec = 1.875 rpm
38. SPD  Check Current Speed

Format:  SPD;

Description:  Check current speed

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

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

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

Unit: Pluse/ Second, PPS or Hz.

The sign of speed decides direction. If no "+" or "-" specified before "x", it
is taken as "+".
39. SQT$\eta$ Set Tolerance of Blocked Alarm

Format: SQT$\eta$;

Description: Set tolerance of blocked alarm.
Alarm when deviation of reading between encoder and pulse counter is larger than $\eta$.
$\eta = 0, 1 \ldots + 65535$.

ACK: AA [Controller ID] B8 [Q0] [Q1] [Q2] [Q3] [Q4] FF

Comment: B8 >> Message ID of desired encoder position;
[Q0] ~ [Q4] >> Received data 0 ~ 4

[Q0] ~ [Q4] is the converted value for desired encoder position. (32 bits)
(Figure 12-2)
Miniature Integrated Stepper Motor Controller (Closed-loop)

40. **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><strong>Definition</strong></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 = 00,01,02 (hexadecimal) denotes configuring sensor 1,2,3;

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

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

**ACK:**

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

**Comment:**

C9 >> Message ID of instruction STGxη;

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

[S0] ~ [S2] is the converted value for sampling mode of sensor 1 (16 bits) (Figure 12-1)

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

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

**Example:**

Requirements:

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

Then:

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

Format: STG;

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

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

Comment: Refer to ACK comment of instruction STGx1;
42. **STO\(\eta\) Parameter Banding**

<table>
<thead>
<tr>
<th>Format:</th>
<th>STO(\eta);</th>
</tr>
</thead>
</table>
| Description: | Banding motion parameters to motions (Triggered by sensor edge or controlled by instruction)  
\(\eta = 0, 1, \ldots, 7\)
\(\eta = 0, \quad \Rightarrow \quad \text{Motion controlled by instruction}\)
\(\eta = 1, \quad \Rightarrow \quad \text{Only for close-loop}\)
\(\eta = 2, \quad \Rightarrow \quad \text{Motion triggered by rising edge of S1}\)
\(\eta = 3, \quad \Rightarrow \quad \text{Motion triggered by falling edge of S1}\)
\(\eta = 4, \quad \Rightarrow \quad \text{Motion triggered by rising edge of S2}\)
\(\eta = 5, \quad \Rightarrow \quad \text{Motion triggered by falling edge of S2}\)
\(\eta = 6, \quad \Rightarrow \quad \text{Motion triggered by rising edge of S3}\)
\(\eta = 7, \quad \Rightarrow \quad \text{Motion triggered by falling edge of S3}\)

The motion parameters include:

1) acceleration  
2) deceleration  
3) maximum starting speed  
4) maximum stopping speed

For sensors, they also include:

5) speed  
6) displacement increment

<table>
<thead>
<tr>
<th>ACK:</th>
<th>AA [Controller ID] D1 FF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comment:</td>
<td>D1 (\quad \Rightarrow \quad \text{Message ID of instruction STO};)</td>
</tr>
</tbody>
</table>
| Note: | Require controller hardware version being 1232 or higher.  
This instruction will affect real time performance. It takes around 15 ms for the instruction to be executed. It is recommended that sending this instruction when the motor is idle, and wait 20ms before sending other instructions. Before set parameters, disable the controller first.  
Default setting for STO0: 300/300/0/0/0/0, it can be configurated by instruction.  
Parameters for each edge can be different. Not all parameters are needed; the non-value parameter will be assigned as the value of parameters for STO0 |
| Example: | Disable the controller: OFF; |
Set 1st group of parameters: SPD₁; STP₁; MAC₁; MDE₁; MMS₁; MMD₁;
Banding it to rising edge of S1: STO2;
......

Set 6th group of parameters: SPD₆; STP₆; MAC₆; MDE₆; MMS₆; MMD₆;
Banding it to falling edge of S3: STO7;
Miniature Integrated Stepper Motor Controller (Closed-loop)

43. STP\(\eta\) Displacement Control

<table>
<thead>
<tr>
<th>Format:</th>
<th>STP(\eta);</th>
</tr>
</thead>
</table>

**Description:** Set the desired incremental displacement, i.e. the displacement relative to current position
\(\eta = -2147483647 \sim +2147483647\) (signed integer)

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

**Comment:**
- B6 >> The message ID of STP\(\eta\);
- [P0] ~ [P4] >> Received data 0 ~ 4

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

**Note:** Displacement is essentially defined as counts of the pulse or encoder counter.
Actual pulse 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, UIM242 will execute the current instruction and stop motor. The former STP instruction is regarded as being completed. Meanwhile, system will shift from PT mode to VT mode.
- If an STP instruction is received while the motor is already running, the former steps will not be counted in the displacement of current STP instruction.

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

**Check Displacement**

**Format:**  

STP;

**Description:**  

Check current incremental displacement.

**ACK:**  

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

**Comment:**  

B3 >> The message ID of current incremental displacement

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

[P0] ~ [P4] is the converted value for the current incremental displacement (32 bits) (Figure11-2)
Miniature Integrated Stepper Motor Controller (Closed-loop)

APPENDIX A DIMENSIONS

![Diagram of NEMA 28 Stepper Motor, Seal Sleeve, Encoder, and Closed-loop Motion Controller with dimensions labeled in millimeters.]

Unit: mm
APPENDIX B  INSTALLATION INSTRUCTION

Back Mounting

Side Mounting