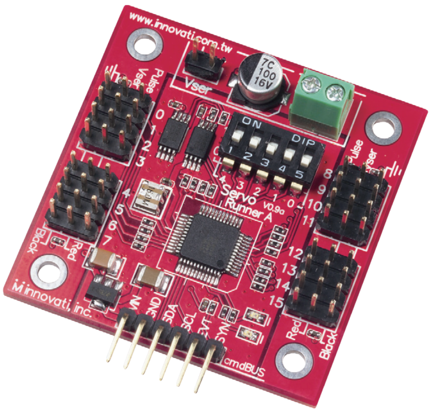
innovati



Servo Runner A User's Guide

Version: 1.2

Innovati's Servo Runner A module is capable of controlling up to 16 servos simultaneously. By providing integrated commands, this allows the user to determine the servo motion performance by directly setting the speed or time. Up to 250 frames are provided for storing the positions and motion configurations (speed or time), allowing various styles of motions to be achieved through the combinations of actions.

Trademark

Disclaimer

Errata

We hope that our users will find this user's guide a useful, easy to use and interesting publication, as our efforts to do this have been considerable. Additionally, a substantial amount of effort has been put into this user's guide to ensure accuracy and complete and error free content, however it is almost inevitable that certain errors may have remained undetected. As Innovati will continue to improve the accuracy of its user's guide, any detected errors will be published on its website. If you find any errors in the user's guide please contact us via email service@innovati.com.tw. For the most up-to-date information, please visit our web site at http://www.innovati.com.tw.

Table Of Contents

Product Overview

Innovati’s Servo Runner A module is capable of controlling up to 16 servos simultaneously. By providing integrated commands, this allows the user to determine the servo motion performance by directly setting the speed or time. Up to 250 frames are provided for storing the positions and motion configurations (speed or time), allowing various styles of motions to be achieved through the combinations of actions.Please use “ServoRunnerA” as the module object name in program.

Application

The operation and application of various servos which could include applications such as robotic arms, robotic joints, etc.

Product Features

- 16 servo control output interfaces for controlling 16 servos simultaneously.
- Capable of controlling the servo position from 0.5 ms to 2.5 ms.
- Software fine-tuning commands allow the fine adjustment of each servo rotation angle in the range of -128~127 µs without machine disassembly.
- The program allows the user to set the servo rotation speed. The user can set multiple levels for the servo rotation speed according to their specific requirements.
- The user can set a common time for every servo to reach different rotation angles at the same time.
- Up to 250 servo motion frames available for storing the positions, speeds or the time parameters of the 16 servos. These can be retrieved at a later time avoiding repeated setting operations which allows the user to combine the actions for various operations.
- 4 event notifications allowing the user to proceed to the next operation once the completion of the action is detected. The event can be configured based on detection the state of any one of the 16 servos.
- Various state inquiry commands allows the user to confirm whether the action of the servo has completed or not at any time, to acquire the current position and the target position and to fine adjuwst the parameters or the preset time and speed values.
- Resolution can be as small as 2µs.

Connection

Directly setup the ID switches to the required number, and then connect the cmdBUS™ cable to the corresponding pins on the BASIC Commander® (shown in the following figure). Then the required operations can be performed through the BASIC Commander®. DC power (6~12V) and ground should be connected to VIN and GND pin.

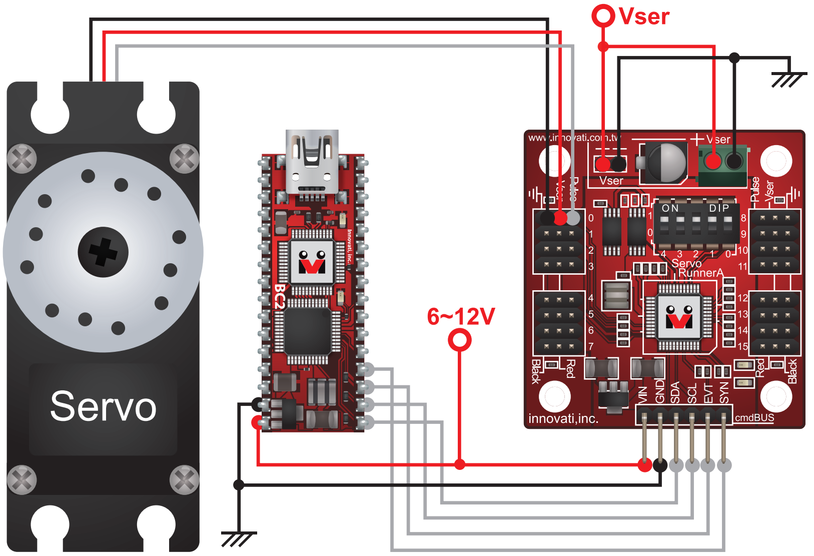


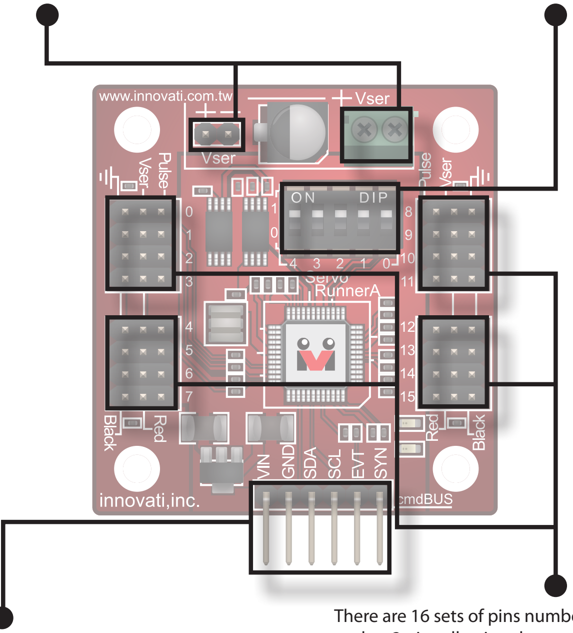
Figure 1: Connect Servo Runner A with BASIC Commander® and Servo

There are 16 servo connectors. Each connector has 3 pins, one to provide the control signal and two for the servo power. The servos should be connected according to the pin assignment configurations shown in figure on the right. In addition, the power required for the servos should be supplied at the power input position as shown in the figure. Take care to ensure that the current and voltage supply to the servos is correct to prevent the servos from damage.

Product Specifications

The input pins for the power supply of the servo controller module. The user can select any one from the two sets of input pins. Care should be taken to ensure correct pin polarity. Reversed connections may cause damage.

Module number setting switches. Set the number of the Servo module in binary format in right to left order. The module number is used by the BASIC Commander® to determine the required module to be controlled.



cmdBUS™ connection pins. Connect these pins to the corresponding pins of the BASIC Commander® to allow the BASIC Commander® to control the Servo module. (When connecting, please pay attention to the pin assignment. Connect Vin to Vin on the BASIC Commander®. Incorrect pin connection may cause damage to the module)

There are 16 sets of pins numbered 0~15. Each set has 3 pins allowing the user to connect up to 16 servos. The servo number printed on the circuit board corresponds to the number assigned in the controller program. The Black pins in the same column should be connected to the ground pins of the servos. Please pay attention to the pin assignment. Incorrect pin connection may cause damage to the module.

Figure 2: Pin Assignment And Switches Of The Module

Precautions For Operations

Please ensure that the voltage and current ranges are correct for the connected servos. Select a suitable power supply and connect correctly to Vser.

The Pulse pins of the servo should be connected to the module to comply with the requirements shown in Table 1.

Symbol	Parameter	Test Conditions (VIN=7.5V)	Min	Typ	Max	Unit
V_{OH}	I/O Port output high voltage	No load	-	5	-	V
V_{OL}	I/O Port output low voltage	No load	-	0	-	V
I_{OL}	I/O Port Sink Current	Vload=0.1V_{OH}	10	20	-	mA
I_{OH}	I/O Port Source Current	Vload=0.9V_{OH}	-5	-10	-	mA
I_L	Current consumption	No load(connected to cmdBUS)	-	7	-	mA

Table 1: Servo Runner A module D.C. Characteristics (Ambient Temperature = 25 ℃)

Absolute Maximum Ratings:

Operating Temperature : 0 ℃ ~ 70 ℃ (excluding the servos)
Storage Temperature : -50 ℃ ~ 125℃

Commands And Events

The following tables list all the unique commands and events provided with the Servo Runner A Module. Note that essential words in the commands will be written in **bold** type and ***italics*** in bold type. The bold type word must be written exactly as shown, whereas the italic bold type words must be replaced with the user values. Note that the innoBASIC™ language is case-insensitive.

Command Format	Description
Servo Position Commands	
SetPos (***ID***, ***Pos***)	Set the servo with ***ID***, ranging from 0 to 15, for operation. The target position is set by ***Pos***. Note that the allowed range for ***Pos*** is 499~2500 in the unit of μs. If the given value is out of this range, the command will not be executed.
SetPosAndRun(***ID***, ***Pos***)	Same as command above. Except after settings are done, the servo starts to operate.
SetPosSpd(***ID***, ***Pos***, ***Spd***)	Set the servo with ***ID***, ranging from 0 to 15, for operation. The target position is set by ***Pos*** and traveling at a speed of ***Spd***. The larger the ***Spd*** value is, the faster the servo travels. Note that the ***Spd*** with value 0 means the full speed. (***Spd*** unit is μs/s)
SetPosSpdAndRun(***ID***, ***Pos***, ***Spd***)	Same as command above. Except after settings are done, the servo starts to operate.
SetPosTime(***ID***, ***Pos***, ***Time***)	Set the servo with ***ID***, ranging from 0 to 15, for operation. The target position is set by ***Pos*** and traveling to the target position in ***Time*** milliseconds. The allowable range of ***Time*** is 0~65535. Note that the ***Time*** with value 0 means full speed. If the value of ***Time*** is too short, the servo will travel at full speed.
SetPosTimeAndRun(***ID***, ***Pos***, ***Time***)	Same as command above. Except after settings are done, the servo starts to operate.
Servo Start Commands	
Run1Servo(***ID***) : **Run15Servo**(***ID1***, ···, ***ID15***) **RunAllServo**()	According to the set value of servo ***ID***(s), ranging from 0 to 15, each corresponding servo will perform the preset operation. If the servo starts without the speed or time settings but only the position setting, the servo will travel at the maximum speed. If any ***ID*** value out of its range, this command will not be executed.

Command Format	Description

Run1ServoWithEventA(*ID*)
:
Run15ServoWithEventA(*ID1*, ⋯,*ID15*)
RunAllServoWithEventA()
	Same as above, except that the event **A** will be triggered when all the indicated servos reach their target positions.

Run1ServoWithEventB(*ID*)
:
Run15ServoWithEventB(*ID1*, ⋯, *ID15*)
RunAllServoWithEventB()
	Same as above, except that the event **B** will be triggered when all the indicated servos reach their target positions.

Run1ServoWithEventC(*ID*)
:
Run15ServoWithEventC(*ID1*, ⋯, *ID15*)
RunAllServoWithEventC()
	Same as above, except that the event **C** will be triggered when all the indicated servos reach their target positions.

Run1ServoWithEventD(*ID*)
:
Run15ServoWithEventD(*ID1*, ⋯, *ID15*)
RunAllServoWithEventD()
	Same as above, except that the event **D** will be triggered when all the indicated servos reach their target positions.

Servo Stop Commands	

Pause1Servo(*ID*)
:
Pause15Servo(*ID1*, ⋯, *ID15*)
PauseAllServo()
	According to the set value of servo ***ID***(s), ranging from 0 to 15, each corresponding servo will stop at the preset operation. If any ***ID*** value out of its range, this command will not be executed.

Stop1Servo(*ID*)
:
Stop15Servo(*ID1*, ⋯, *ID15*)
StopAllServo()
	Same as above, except that the control signal ceases to transmit to the servo(s). As a result, the servo will change its position by applying an external force.

Servo and Memory Status Commands	

Get1ServoReadyStatus(*ID*, *Status*)
:
Get15ServoReadyStatus(*ID1*, ⋯, *ID15*, *Status*)
GetAllServoReadyStatus(*Status*)
	Get the operation status of the servo(s) indicated by ***ID***(s), ranging from 0 to 15, and store the status in ***Status***. When all the servos reach their target positions, the returned status will be 1, otherwise value 0 will be returned.

GetNowPos (*ID*, *Pos*)	Get the current position of the servo indicated by ***ID***, ranging from 0 to 15, and then store it in the word variable ***Pos***.
GetPos(*ID*, *Pos*)	Get the target position of the servo indicated by ***ID***, ranging from 0 to 15, and then store it in the word variable ***Pos***.
GetPosOffset(*ID*, *Offset*)	Get the position offset of the servo indicated by ***ID***, ranging from 0 to 15, and then store it in the short variable ***Offset***, ranging form -128 to 127. The unit of ***Offset*** is microsecond (μs).

Command Format	Description
GetSpdAndTime(*ID*, *Type*, *Value*)	Get the motion type of the servo indicated by ***ID***, ranging from 0 to 15, and store the values in ***Type***. The corresponding setting values are stored in the word variable ***Value***. If the set servo travel type is speed, then the returned value for ***Type*** will be 1. If the set servo travel type is time, then the returned value for ***Type*** will be 0.
LoadFrame(*FrameID*)	Load the servos operation settings from the frame memory block indicated by ***FrameID***, ranging from 0 to 249, as the current target position and motion type of the servos.
SaveFrame(*FrameID*)	Store the current settings of servos operations into the frame memory blocks indicated by ***FrameID***, ranging from 0 to 249.
SetPosOffset(*ID*, *Offset*)	Set the offset of the servo indicated by ***ID*** with the value ***Offset***, ranging from -128 to 127.
LoadOffset()	Load the offset value from the EEPROM and replace the current settings.
SaveOffset()	Store current offset value into the EEPROM.

Table 2: Command Table

Event Name	Description
ServoPosReadyEventA	Execute the **Run*N*ServoWithEventA** command, where ***N*** can be literally ***1~15*** or ***All***. When all the indicated servos reach their target positions, this event will be triggered.
ServoPosReadyEventB	Execute the **Run*N*ServoWithEventB** command, where ***N*** can be literally ***1~15*** or ***All***. When all the indicated servos reach their target positions, this event will be triggered.
ServoPosReadyEventC	Execute the **Run*N*ServoWithEventC** command, where ***N*** can be literally ***1~15*** or ***All***. When all the indicated servos reach their target positions, this event will be triggered.
ServoPosReadyEventD	Execute the **Run*N*ServoWithEventD** command, where ***N*** can be literally ***1~15*** or ***All***. When all the indicated servos reach their target positions, this event will be triggered.

Table 3: Event Table

Example Program

```
'In the example program, the position value is set according to the range of the
'majority of the servos. Please adjust the allowed position range for the servos
'to avoid damage to the servos.
Peripheral mySer As ServoRunnerA @ 0        'Set the module to be operated as 0

Dim EventEnd As Byte         'Store the variable for indicate the completion of
                             'the event
Dim i As Byte                'Store the loop variable

Dim SerStatus As Byte        'Store the Status of the Servo

Sub Main()                   'Main subroutine
    mySer.SetPosOffset(0, 0)          'Set the offset value of Servo0 as 0
    mySer.SetPosAndRun(0, 1500)       'Run Servo 0 to position 1500
    Pause 1000               'Pause for the servo to move to the target position
    mySer.SetPos(0, 2200)    'Set the target position of Servo 0 as 2200
    mySer.SaveFrame(0)       'Save the current motion as Frame 0
    mySer.Run1Servo(0)       'Allow Servo 0 to start the motion
    Pause 500
    mySer.SetPosSpdAndRun(0, 700, 1000)         'Servo 0 moves to the position
                                                '700 at speed 1000
    Pause 2000
    mySer.SetPosTimeAndRun(0, 2200, 1000)       'Servo 0 moves to the position
                                                '2200 in 1 second.
    Pause 1000
    EventEnd=0
    mySer.SetPosTime(0, 700, 1000)    'Set Servo 0 to position 700 in 1 second
    mySer.SaveFrame(1)                'Save the current motion as Frame1
    mySer.Run1ServoWithEventA(0)      'Run Servo 0 and generate EventA
                                      'when com pletes
    Do
       Pause 1
    Loop Until EventEnd=1

'Repeats to load values in Frame 0 and Frame1 and then activate Servo 0 for
'operation. The position value stored in Frame 0 is 2200. The position value
'stored in Frame1 is 700. Servo 0 will move between these two positions back
'and forth 4 times.

    For i=0 To 3
       mySer.LoadFrame(1)  'Read the setting value stored in Frame 1
```

```
        mySer.Run1Servo(0)
        Pause 1000
        mySer.LoadFrame(0)  'Read the setting value stored in Frame 0
        mySer.Run1Servo(0)
        Pause 1000
    Next
        mySer.SetPosAndRun(0, 1500)

'The following loop repeats to perform a Status reading operation. After
'completion of the operation is confirmed, the loop will stop
    Do
        mySer.Get1ServoReadyStatus(0, SerStatus)         'Read Servo 0 and
                                                         'store it in SerStatus
        Loop Until SerStatus>0
End Sub

Event mySer.ServoPosReadyEventA()
    mySer.SetPosAndRun(0, 2200)
    Pause 1000
    EventEnd=1
End Event
```

Appendix

Important Notes

At version 1.0, note that while using this command for obtaining the target position, the command should be performed more than 1 ms after the set-position command is executed. Otherwise, it may retrieve previous set position value.

At version 1.0, while using SaveFrame for storing the positions, the position value should be an even number. If it is an odd number, an error will occur while retrieving the value.

At version 1.0, note that while performing repeated SaveFrame storing operations in series, the storing operations should be separated by a time interval of at least 5 ms. Otherwise, it may not be possible to correctly perform all the store commands.

Module ID Setting Table

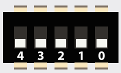
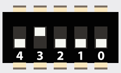
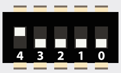
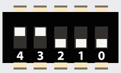
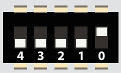
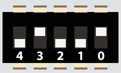
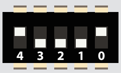
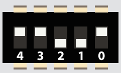
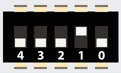
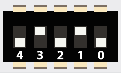
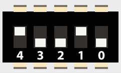
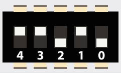
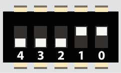
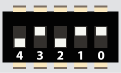
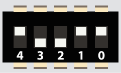
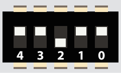
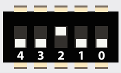
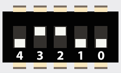
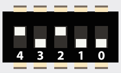
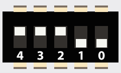
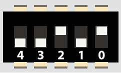
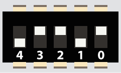
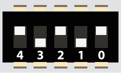
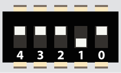
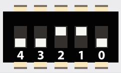
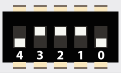
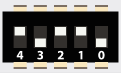
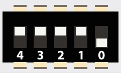
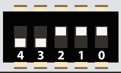
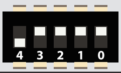
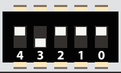
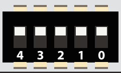
DIP Switch	ID	DIP Switch	ID	DIP Switch	ID	DIP Switch	ID
	0		8		16		24
	1		9		17		25
	2		10		18		26
	3		11		19		27
	4		12		20		28
	5		13		21		29
	6		14		22		30
	7		15		23		31

Table 4: Module ID Setting Table