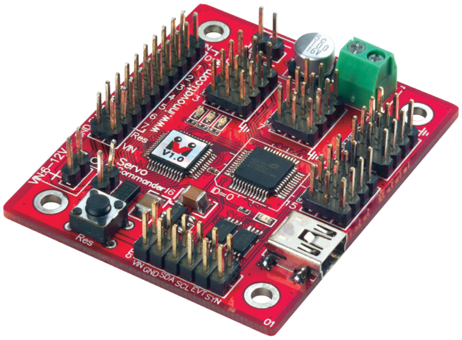


# Servo Commander 16 User's Guide

16 Servo Control Outputs

Version: 1.2

Innovati's Servo Commander 16 module incorporates BASIC Commander® – BC1 and one Servo Runner A module. It saves area occupied by the control modules and connection wires while keeping all the functions of Servo Runner A, controls 16 servos simultaneously, and the simple and integrated software functions enabling users to directly control the servo movement by fixed speed or common time. There are up to 250 frames for storing the positions and motion configurations (speed or time), thus various ways of motions can be achieved through the combinations of actions. Please use "ServoRunnerA" as the module object name with address ID 0 in program.

## 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 Content

# Product Overview

Innovati’s Servo Commander 16 module incorporates BASIC Commander® – BC1 and one Servo Runner A module. It saves area occupied by the control modules and connection wires while keeping all the functions of Servo Runner A, controls 16 servos simultaneously, and the simple and integrated software functions enabling users to directly control the servo movement by fixed speed or common time. There are up to 250 frames for storing the positions and motion configurations (speed or time), thus various ways of motions can be achieved through the combinations of actions. Please use “ServoRunnerA” as the module object name with address ID 0 in program.

# Application

- The operation and application of various servos including the robotic arms, robotic joints, etc.
- Various applications of small servos.

# Product Features

- Complete functions and hardware interfaces of BC1 and Servo Runner A.
- Built-in cmdBUS™ connection between BC1 and Servo Runner A: External connection is unnecessary.
- Shared power jumpers of the servos and control electronics: A single power supply is sufficient for servos and control electronics.
- 16 servo control output interfaces for controlling 16 servos simultaneously.
- Capable of controlling the position of the servo from 0.5 ms to 2.5 ms.
- Software fine-tune commands allow the user to fine adjust the rotation angle of each servo in the range of -128~127 µs only by software setting without the disassembly of the machine.
- Program allows user to set the rotation speed of the servo. The user can set multiple levels of the rotation speed of the servo according to the requirements.
- User can set a common time for every servo to reach different rotation angle at the same time.
- Built-in 250 frames in the Servo Commander board. Each frame can store the current target positions, speeds or the time parameters of the 16 servos which can be restored directly on demand, and thus avoid repeated setting operations and 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 the any one of the 16 servos.
- Various state inquiry commands allows the user to confirm whether the action of the servo is completed or not at any time, to acquire the current position and the target position, to fine adjust the parameters or the preset time and speed values.
- Resolution can be as small as 2µs.

**Note:** This manual mainly discusses the servo control. For commands and system configurations of BASIC Commander®, please refer to "BASIC Commander & innoBASIC Workshop Reference Manual."
To execute functions related to servo control, please set the module number as 0 in the program.

# Connection

The module has 16 servo connectors with 3 pins for each connector. The servo connectors provide servos with power and control signals, and are divided into two groups. To control servos, connect proper pins of servos to these connectors (as shown in the right figure). Two power supply connections are available, as shown in Figures 1 and 2. Before connecting the power, please check the current and voltage of the servo to avoid motor damages due to abnormal operations.

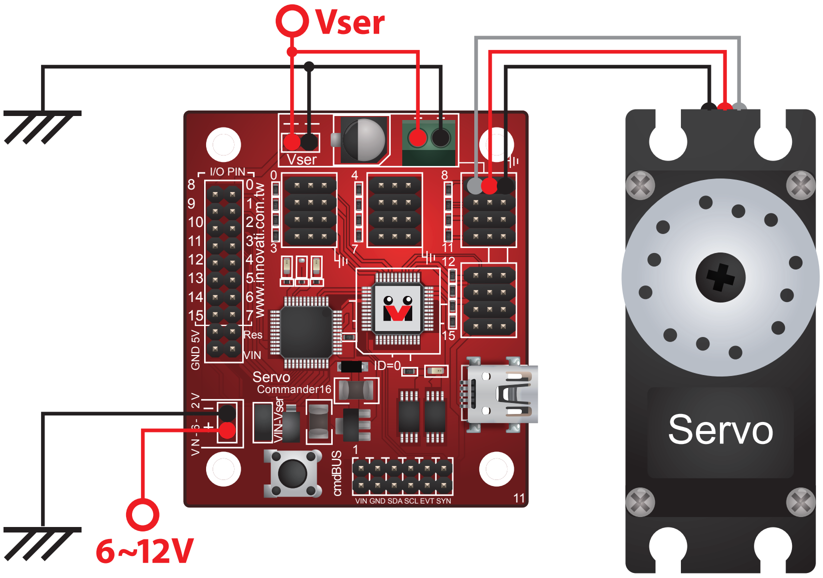


**Figure 1: The Servo And The Control Electronics Use Different Power Supplies.**

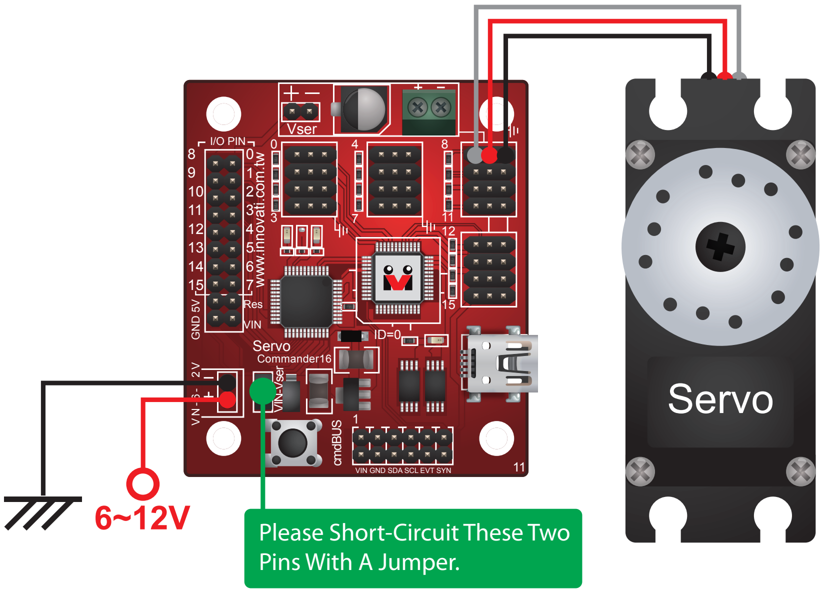


Figure 2: The Servo And The Control Electronics Use The Same Power Supply.

Before using the connection shown in the figure, please make sure the voltage of the power supply is within the servo's voltage tolerances.

# Product Specifications

Servo power input: two input power connectors are available. Either can be used. Please mind the positive and negative pin assignment. Incorrect connection may damage the devices.

16 programmable I/O pins with IDs are shown in the figure. These pins may be defined by the software programs as outputs or inputs.

GND, 5V, VIN and Res: for special applications that require these pins.

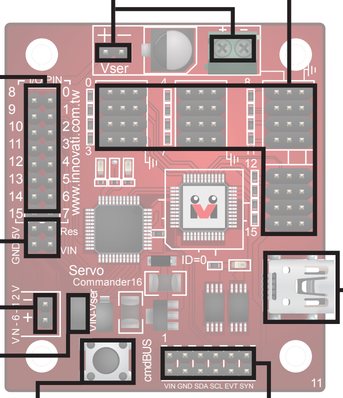
Please attach a 6-12V power supply here to power the electronics.

With the two pins shorted with a jumper, VIN and Vser share the same power supply. Please do not connect VIN and Vser to different power supplies when the jumper is installed.

Reset button: Resets the system when pressed.

There are 16 sets of pins numbered 0~15, allowing for up to 16 servo connections. Please write the control program with the proper servo module ID. The pins with ground labels and those in the same column as the previous pins should be connected to the ground pins of the servos (the rightmost far-right pins in the figure). Please mind the pin assignment. Incorrect pin connection may cause device damages.

USB connector: please connect the module to the PC with a USB cable to download control programs.



Servo Commander 16 provides two cmdBUS™ connectors that can be connected to other modules if necessary. When connecting other modules, please mind the pin assignments. Vin has to be connected to the Vin on other modules. Incorrect connection may damage the devices.

**Figure 3: Pin Assignment And Device Description.**

Current consumption: 32.3 mA (the current consumed at VIN when the Servo Commander 16 module is not connected to any servo.)

Module Dimensions: 46.8 x 54.8 (mm)

# Precautions For Operations

Please make sure of the voltage and current ranges required for the connected servos. Select a suitable power supply and connect correctly to the Vser.

The Pulse pins of the servo should be connected to the module in a way complying with the requirements shown in Table 1. (This is the range allowing the module to operate.)

| Symbol | Parameter | Test Conditions | | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| | | VIN=7.5V | Conditions | | | | |
| $V_{OH}$ | I/O Port output high voltage | - | No loading | - | 5 | - | V |
| $V_{OL}$ | I/O Port output low voltage | - | No loading | - | 0 | - | V |
| $I_{OL}$ | I/O Port Sink Current | - | $V_{load}$=0.1$V_{OH}$ | 10 | 20 | - | mA |
| $I_{OH}$ | I/O Port Source Current | - | $V_{load}$=0.9$V_{OH}$ | -5 | -10 | - | mA |

**Table 1: Current Limits Of The Servo Commander 16 Module (Test Temperature = 25 ℃)**

## Absolute Maximum Ratings:

Operating Temperature of the Module: 0 ℃ ~ 70 ℃ (Please confirm the operating temperature of the servos according to the specifications of the servos)

Storage Temperature of the Module: -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*)**<br>:<br>**Run15Servo(*ID1*, …, *ID15*)**<br>**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. |
| **Run1ServoWithEventA(*ID*)**<br>:<br>**Run15ServoWithEventA(*ID1*, …, *ID15*)**<br>**RunAllServoWithEventA()** | Same as above, except that the event A will be triggered when all the indicated servos reach their target positions. |
| **Run1ServoWithEventB(*ID*)**<br>:<br>**Run15ServoWithEventB(*ID1*, …, *ID15*)**<br>**RunAllServoWithEventB()** | Same as above, except that the event B will be triggered when all the indicated servos reach their target positions. |

| Command Format | Description |
|---|---|
| **Run1ServoWithEventC(*ID*)**<br>:<br>**Run15ServoWithEventC(*ID1*, ···, *ID15*)**<br>**RunAllServoWithEventC()** | Same as above, except that the event C will be triggered when all the indicated servos reach their target positions. |
| **Run1ServoWithEventD(*ID*)**<br>:<br>**Run15ServoWithEventD(*ID1*, ···, *ID15*)**<br>**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*)**<br>:<br>**Pause15Servo(*ID1*, ···, *ID15*)**<br>**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*)**<br>:<br>**Stop15Servo(*ID1*, ···, *ID15*)**<br>**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*)**<br>:<br>**Get15ServoReadyStatus(*ID1*, ···, *ID15*, *Status*)**<br>**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). |
| **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 servo 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 servo operations into the frame 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 ***RunNServoWithEventA*** 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 ***RunNServoWithEventB*** 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 ***RunNServoWithEventC*** 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 ***RunNServoWithEventD*** 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 Provided By The Module**

# 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.

'Set the module ID as 0. Note: The module number must be set to 0 or 1 to use
'the servo related command of the Servo Commander A.

Dim EventEnd As Byte          'Store the variable for determining the completeness
                              '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)         'Activate Servo0 to move to the position
                                        '1500.
    Pause 1000                'Pause a time interval for the servo to move
                              'to the target position.

    mySer.SetPos(0, 2200)     'Set the target position of Servo0 as 2200.
    mySer.SaveFrame(0)        'Store the motion of the currently indicated servo
                              'motor into Frame0.
    mySer.Run1Servo(0)        'Allow Servo0 to start the motion.
    Pause 500

    mySer.SetPosSpdAndRun(0, 700, 1000)
                              'Activate Servo0 and then move to the position 700
                              'at a speed of 1000.
    Pause 2000
    mySer.SetPosTimeAndRun(0, 2200, 1000)
                              'Activate Servo0 and move to the position 2200 for
                              'a time interval of 1 second.
    Pause 1000

    EventEnd=0
    mySer.SetPosTime(0, 700, 1000)
                              'Set Servo0 to move to the position 700 for a time
                              'interval of 1 second.
    mySer.SaveFrame(1)        'Store the motion of the currently indicated servo
                              'motor into Frame1.
    mySer.Run1ServoWithEventA(0)
                              'Activate Servo0 and generate EventA when
                              'it completes the operation.

    Do
        Pause 1
    Loop Until EventEnd=1

'The following loop repeats to read the setting values in Frame0 and then
'activate Servo0 for operation.
'The position value stored in Frame0 is 2200. The position value stored in
'Frame1 is 700.
'Servo0 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 Frame1.
        mySer.Run1Servo(0)
    Pause 1000
    mySer.LoadFrame(0)                'Read the setting value stored in Frame0.
    mySer.Run1Servo(0)
    Pause 1000

    Next
        mySer.SetPosAndRun(0, 1500)

'The following loop repeats to perform the operation of reading the Status.
'After the completion of the operation is confirmed, the loop will stop.

    Do
        mySer.Get1ServoReadyStatus(0, SerStatus)
'Read the status of Servo0 and then store it in SerStatus.
    Loop Until SerStatus>0

End Sub

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

# Appendix

## Known problem:

※The version is specified in the laser label on the module.