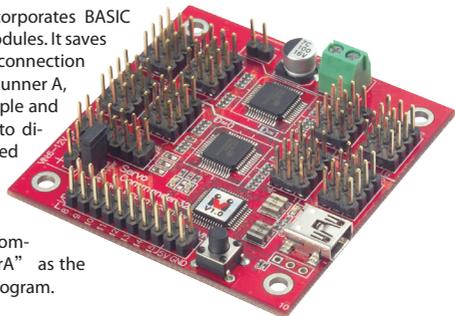


Servo Commander 32 User's Guide

32 Servo Control Outputs

Version: 1.2

Innovati's Servo Commander 32 module incorporates BASIC Commander® – BC1 and two Servo Runner A modules. It saves area occupied by the control modules and connection wires while keeping all the functions of Servo Runner A, controls 32 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 memory blocks for storing the addresses 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 0 and 1 in program.



Trademark

Innovati®, , and BASIC Commander® are registered trademarks of Innovati, Inc.

InnoBASIC™ and cmdBUS™ are trademarks of Innovati, Inc.

Copyright © 2008-2009 by Innovati, Inc. All Rights Reserved.

Due to continual product improvements, Innovati reserves the right to make modifications to its products without prior notice. Innovati does not recommend the use of its products for application that may present a risk to human life due to malfunction or otherwise.

No part of this publication may be reproduced or transmitted in any form or by any means without the expressed written permission of Innovati, Inc.

Disclaimer

Full responsibility for any applications using Innovati products rests firmly with the user and as such Innovati will not be held responsible for any damages that may occur when using Innovati products. This includes damage to equipment or property, personal damage to life or health, damage caused by loss of profits, goodwill or otherwise. Innovati products should not be used for any life saving applications as Innovati's products are designed for experimental or prototyping purposes only. Innovati is not responsible for any safety, communication or other related regulations. It is advised that children under the age of 14 should only conduct experiments under parental or adult supervision.

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	1
Application	1
Product Features	2
Connection	3
Product Specifications	5
Precautions For Operations	6
Commands and Events	7
Example Program	10
Appendix	13

Product Overview

Innovati's Servo Commander 32 module incorporates BASIC Commander® – BC1 and two Servo Runner A modules. It saves area occupied by the control modules and connection wires while keeping all the functions of Servo Runner A, controls 32 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 memory blocks 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 0 and 1 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 two Servo Runner A modules.
- Built-in cmdBUS connection between BC1 and two Servo Runner A modules: External connection is unnecessary.
- Shared power jumpers of the servos and control electronics: A single power supply is sufficient for servos and control electronics.
- 32 servo control output interfaces for controlling 32 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 $-128\sim 127\ \mu\text{s}$ 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 memory blocks in the Servo Commander board. Each memory block can store the current target positions, speeds or the time parameters of the 32 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 32 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 $2\ \mu\text{s}$.

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 and 1 in the program.

Connection

The module has 32 servo connectors with 3 pins for each connector. The servo connectors provide servos with power and control signals, and are divided into two groups. The upper 16 connectors belong to the servo module 1 while the lower 16 connectors the servo module 0. 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.

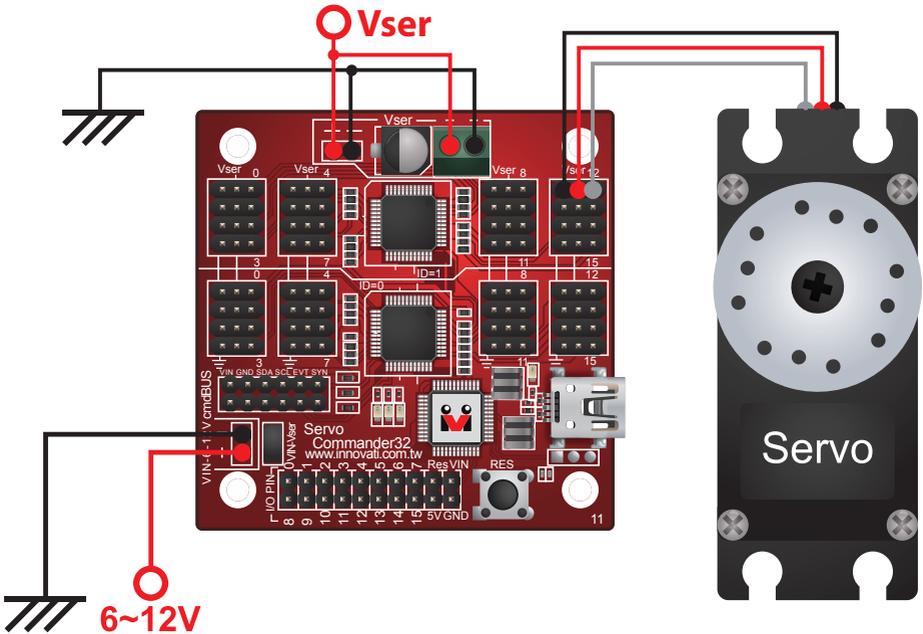


Figure 1: The servo and the control electronics use different power supplies.

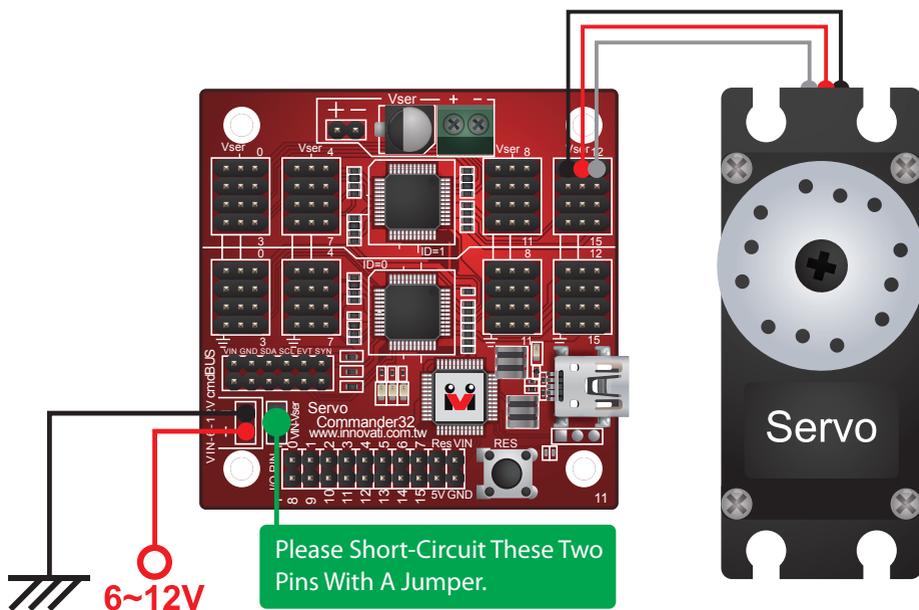


Figure 2: The servo and the control electronics use the same power supply.

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

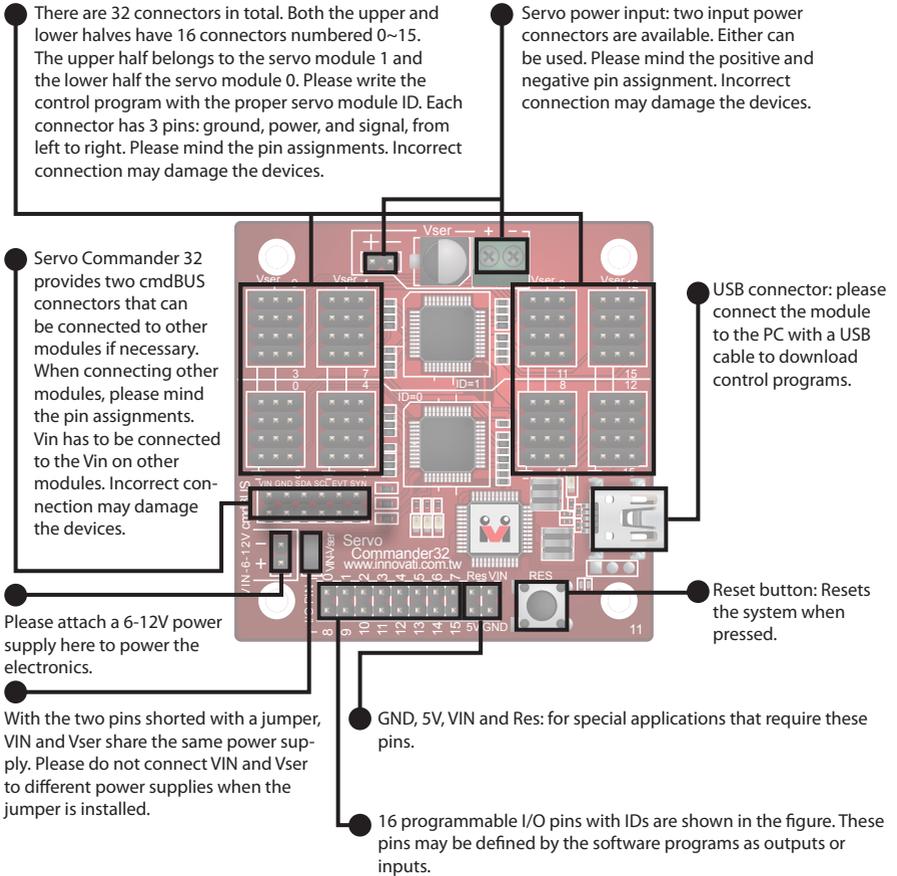


Figure 3: Pin Assignment And Device Description.

Current consumption: 37.5 mA (the current consumed at VIN when the Servo Commander 32 module is not connected to any servo.)

Module Dimensions: 57.5 x 58.6 (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} =0.1V _{OH}	10	20	-	mA
I _{OH}	I/O Port Source Current	-	V _{load} =0.9V _{OH}	-5	-10	-	mA

Table 1: Current Limits Of The Servo Commander 32 Module (Test Temperature = 25 °C)

Absolute Maximum Ratings:

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

Storage Temperature of the Module: -50 °C ~ 125 °C

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 (<i>ID</i> , <i>Pos</i>)	Set the servo with <i>ID</i> , ranging from 0 to 15, for operation. The target position is set by <i>Pos</i> . 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 (<i>ID</i> , <i>Pos</i>)	Same as command above. Except after settings are done, the servo starts to operate.
SetPosSpd (<i>ID</i> , <i>Pos</i> , <i>Spd</i>)	Set the servo with <i>ID</i> , ranging from 0 to 15, for operation. The target position is set by <i>Pos</i> and traveling at a speed of <i>Spd</i> . The larger the <i>Spd</i> value is, the faster the servo travels. Note that the <i>Spd</i> with value 0 means the full speed. (<i>Spd</i> unit is μs/s)
SetPosSpdAndRun (<i>ID</i> , <i>Pos</i> , <i>Spd</i>)	Same as command above. Except after settings are done, the servo starts to operate.
SetPosTime (<i>ID</i> , <i>Pos</i> , <i>Time</i>)	Set the servo with <i>ID</i> , ranging from 0 to 15, for operation. The target position is set by <i>Pos</i> and traveling to the target position in Time milliseconds. The allowable range of <i>Time</i> is 0~65535. Note that the <i>Time</i> with value 0 means full speed. If the value of <i>Time</i> is too short, the servo will travel at full speed.
SetPosTimeAndRun (<i>ID</i> , <i>Pos</i> , <i>Time</i>)	Same as command above. Except after settings are done, the servo starts to operate.
Servo Start Commands	
Run1Servo (<i>ID</i>) : Run15Servo (<i>IDI</i> , ..., <i>ID15</i>) RunAllServo ()	According to the set value of servo <i>ID</i> (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 <i>ID</i> value out of its range, this command will not be executed.
Run1ServoWithEventA (<i>ID</i>) : Run15ServoWithEventA (<i>IDI</i> , ..., <i>ID15</i>) RunAllServoWithEventA ()	Same as above, except that the event A will be triggered when all the indicated servo reach their target positions.
Run1ServoWithEventB (<i>ID</i>) : Run15ServoWithEventB (<i>IDI</i> , ..., <i>ID15</i>) RunAllServoWithEventB ()	Same as above, except that the event B will be triggered when all the indicated servo reach their target positions.

Command Format	Description
Run1ServoWithEventC(<i>ID</i>) : Run15ServoWithEventC(<i>IDI1</i>, ..., <i>IDI15</i>) RunAllServoWithEventC()	Same as above, except that the event C will be triggered when all the indicated servo reach their target positions.
Run1ServoWithEventD(<i>ID</i>) : Run15ServoWithEventD(<i>IDI1</i>, ..., <i>IDI15</i>) RunAllServoWithEventD()	Same as above, except that the event D will be triggered when all the indicated servo reach their target positions.
Servo Stop Commands	
Pause1Servo(<i>ID</i>) : Pause15Servo(<i>IDI1</i>, ..., <i>IDI15</i>) PauseAllServo()	According to the set value of servo <i>ID</i> (s), ranging from 0 to 15, each corresponding servo will stop at the preset operation. If any <i>ID</i> value out of its range, this command will not be executed.
Stop1Servo(<i>ID</i>) : Stop15Servo(<i>IDI1</i>, ..., <i>IDI15</i>) 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(<i>ID</i>, <i>Status</i>) : Get15ServoReadyStatus(<i>IDI1</i>, ..., <i>IDI15</i>, <i>Status</i>) GetAllServoReadyStatus(<i>Status</i>)	Get the operation status of the servo(s) indicated by <i>ID</i> (s), ranging from 0 to 15, and store the status in <i>Status</i> . When all the servo reach their target positions, the returned status will be 1, otherwise value 0 will be returned.
GetNowPos (<i>ID</i>, <i>Pos</i>)	Get the current position of the servo indicated by <i>ID</i> , ranging from 0 to 15, and then store it in the word variable <i>Pos</i> .
GetPos(<i>ID</i>, <i>Pos</i>)	Get the target position of the servo indicated by <i>ID</i> , ranging from 0 to 15, and then store it in the word variable <i>Pos</i> .
GetPosOffset(<i>ID</i>, <i>Offset</i>)	Get the position offset of the servo indicated by <i>ID</i> , ranging from 0 to 15, and then store it in the short variable <i>Offset</i> , ranging form -128 to 127. The unit of <i>Offset</i> is microsecond (μ s).
GetSpdAndTime(<i>ID</i>, <i>Type</i>, <i>Value</i>)	Get the motion type of the servo indicated by <i>ID</i> , ranging from 0 to 15, and store the values in <i>Type</i> . The corresponding setting values are stored in the word variable <i>Value</i> . If the set servo travel type is speed, then the returned value for <i>Type</i> will be 1. If the set servo travel type is time, then the returned value for <i>Type</i> will be 0.
LoadFrame(<i>FrameID</i>)	Load the servo operation settings from the frame memory block indicated by <i>FrameID</i> , ranging from 0 to 249, as the current target position and motion type of the servo.
SaveFrame(<i>FrameID</i>)	Store the current settings of servo operations into the frame indicated by <i>FrameID</i> , ranging from 0 to 249.
SetPosOffset(<i>ID</i>, <i>Offset</i>)	Set the offset of the servo indicated by <i>ID</i> with the value <i>Offset</i> , 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

Servo Commander 32 User's Guide

Event Name	Description
ServoPosReadyEventA	Execute the <i>RunNServoWithEventA</i> command, where <i>N</i> can be literally 1~15 or All. When all the indicated servo reach their target positions, this event will be triggered.
ServoPosReadyEventB	Execute the <i>RunNServoWithEventB</i> command, where <i>N</i> can be literally 1~15 or All. When all the indicated servo reach their target positions, this event will be triggered.
ServoPosReadyEventC	Execute the <i>RunNServoWithEventC</i> command, where <i>N</i> can be literally 1~15 or All. When all the indicated servo reach their target positions, this event will be triggered.
ServoPosReadyEventD	Execute the <i>RunNServoWithEventD</i> command, where <i>N</i> can be literally 1~15 or All. When all the indicated servo 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 servo.
` Please adjust the allowed position range for the servo to avoid damage
` to the servo.
Peripheral mySer0 As ServoRunnerA @ 0 ` Set the module to be operated as 0.
Peripheral mySer1 As ServoRunnerA @ 1 ` Set the module ID as 0.
                                     ` Note: The module number must be set to
                                     ` 0 or 1 to use the servo related
                                     ` commands of the Servo Commander A.
Dim EventEnd0, EventEnd1 As Byte    ` Store the variable for determining the
                                     ` completeness of the event.
Dim i As Byte                        ` Store the loop variable.
Dim SerStatus0, SerStatus1 As Byte  ` Store the Status of the Servo0 and
                                     ` Server1 Motor.

Sub Main()
    mySer0.SetPosOffset(0, 0)        ` Main subroutine
    mySer1.SetPosOffset(0, 0)        ` Set the offset value of Servo0 as 0.
    mySer0.SetPosAndRun(0, 1500)    ` Set the offset value of Servo1 as 0.
    mySer1.SetPosAndRun(0, 1500)    ` Activate Servo0 to move to the
    mySer0.SetPosAndRun(0, 1500)    ` position 1500.
    mySer1.SetPosAndRun(0, 1500)    ` Activate Servo1 to move to the
    mySer0.SetPosAndRun(0, 1500)    ` position 1500.
    Pause 1000                      ` Pause a time interval for the servo
    mySer0.SetPos(0, 2200)           ` motor to move to the target position.
    mySer1.SetPos(0, 2200)           ` Set the target position of Servo0 as
    mySer0.SaveFrame(0)              ` 2200.
    mySer1.SaveFrame(0)              ` Set the target position of Servo1 as
    mySer0.Run1Servo(0)              ` 2200.
    mySer1.Run1Servo(0)              ` Store the motion of the currently
    Pause 500                        ` indicated servo0 motor into Frame0.
    mySer0.SetPosSpdAndRun(0, 700, 1000) ` Store the motion of the currently
    mySer1.SetPosSpdAndRun(0, 700, 1000) ` indicated servo1 motor into Frame0.
    mySer0.SetPosSpdAndRun(0, 700, 1000) ` Allow Servo0 to start the motion.
    mySer1.SetPosSpdAndRun(0, 700, 1000) ` Allow Servo1 to start the motion.
    mySer0.SetPosSpdAndRun(0, 700, 1000) ` Activate Servo0 and then move to the
    mySer1.SetPosSpdAndRun(0, 700, 1000) ` position 700 at a speed of 1000.
    mySer0.SetPosSpdAndRun(0, 700, 1000) ` Activate Servo1 and then move to the
    mySer1.SetPosSpdAndRun(0, 700, 1000) ` position 700 at a speed of 1000.
```

Servo Commander 32 User's Guide

```
Pause 2000
mySer0.SetPosTimeAndRun(0, 2200, 1000) ` Activate Servo0 and move to the
` position 2200 for a time interval of
` 1 second.
mySer1.SetPosTimeAndRun(0, 2200, 1000) ` Activate Servo1 and move to the
` position 2200 for a time interval of
` 1 second.

Pause 1000

EventEnd0=0
EventEnd1=0
mySer0.SetPosTime(0, 700, 1000) ` Set Servo0 to move to the position 700
` for a time interval of 1 second.
mySer1.SetPosTime(0, 700, 1000) ` Set Servo1 to move to the position 700
` for a time interval of 1 second.
mySer0.SaveFrame(1) ` Store the motion of the currently
` indicated servo into Frame1.
mySer1.SaveFrame(1) ` Store the motion of the currently
` indicated servo into Frame1.
mySer0.Run1ServoWithEventA(0) ` Activate Servo0 and generate EventA
` when it completes the operation.
mySer1.Run1ServoWithEventA(0) ` Activate Servo1 and generate EventA
` when it completes the operation.

Do
  Pause 1
Loop Until EventEnd0=1 and EventEnd1=1

` The following loop repeats to read the setting values in Frame0 and Frame1
` and then activate Servo0 and Servo1 for operation.
` The position value stored in Frame0 is 2200. The position value stored in
` Frame1 is 700.
` Servo0 and Servo1 will move between these two positions back and forth 4
` times.
For i=0 To 3
  mySer0.LoadFrame(1) ` Read the setting value stored in Frame1.
  mySer1.LoadFrame(1) ` Read the setting value stored in Frame1.
  mySer0.Run1Servo(0)
  mySer1.Run1Servo(0)
  Pause 1000
  mySer0.LoadFrame(0) ` Read the setting value stored in Frame0.
  mySer1.LoadFrame(0) ` Read the setting value stored in Frame0.
  mySer0.Run1Servo(0)
  mySer1.Run1Servo(0)
```

```
    Pause 1000
Next

mySer0. SetPosAndRun(0, 1500)
mySer1. 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
    mySer0.Get1ServoReadyStatus(0, SerStatus0)
        ` Read the status of Servo0 and then store it in SerStatus0.
    mySer1.Get1ServoReadyStatus(0, SerStatus1)
        ` Read the status of Servo1 and then store it in SerStatus1.
Loop Until SerStatus0>0 And SerStatus1>0
End Sub

Event mySer0.ServoPosReadyEventA()
mySer0.SetPosAndRun(0, 2200)
Pause 1000
EventEnd0=1
End Event

Event mySer1.ServoPosReadyEventA()
mySer1.SetPosAndRun(0, 2200)
Pause 1000
EventEnd1=1
End Event
```

Appendix

Known problem:

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