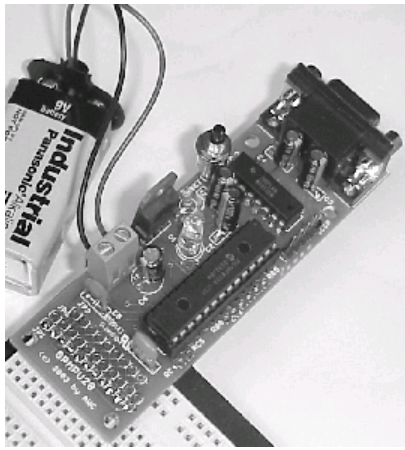
AWC

# GP4 PC Servo Control Kit






AWC
310 Ivy Glen
League City, TX 77573
(281) 334-4341
http://www.al-williams.com/awce.htm
V1.0 30 Aug 2003

## Table of Contents

Overview ..... 1
If You Need Help ..... 1
Building ..... 1
Operation ..... 3
Reference ..... 4
Command Reference ..... 5
Quick Reference ..... 8
  Methods/Functions ..... 8
  Properties ..... 8
  Events ..... 8
Example Visual Basic Code ..... 9
Tips ..... 10
Notes ..... 11
Schematic ..... 12

## Overview

The GP4 is the easiest way to interface servos to PC-based software. This kit provides a hardware interface that allows you to easily generate pulses designed to drive common servo motors. Features include:

- 8 outputs can drive 8 servos at once
- Servos are centered with a 1.5mS pulse
- Adjust servo position between -50 (1mS) and 50 (2mS) – over 100 discrete positions
- Works with standard servos or continuous rotation mods
- True RS232 from onboard 5V supply
- Disable each servo from software or via hardware control line
- All 8 pulses start at the time
- ActiveX control (OCX) supplied; or control via serial protocol (19200 baud)

## If You Need Help

If you require assistance, please feel free to contact us. The best way to get support is via e-mail (stamp@al-williams.com). However, you may also call between 9AM - 4PM Central Time at (281) 334-4341. You can also fax to (281) 754-4462. Be sure to check out our Web page for updates at www.al-williams.com/awce.

## Building

Please follow the directions included for building the GPMPU kit. The board requires one modification to work with the GP4. In

addition, you may want to consider any special power supply connections or serial connections you'd like to make. First, carefully cut pin 9 from IC2 (the MAX232) before soldering the chip to the PC board. The PCB hole that corresponds to this pin is the output for servo #1. The hole for servo #2 is RB2 and is connected by a short trace to the CTS output of the board. You may leave this connection intact with no ill effect (in fact, the software can read the input from servo #2 to verify the device is operating). However, if you don't want the output fed back to the PC, simply use a sharp knife to carefully remove the trace between RB2 and the adjacent hole marked CTS.

If you are prototyping, you may want to install the header at JP1 and use it to connect to a solderless breadboard. You will also need to solder wires to the holes along the long edge of the board near IC1 to connect to your servos.

For a permanent installation, you'll want to omit the header at JP1 and simply make the connections to fit your installation. If you don't plan on using the hardware enable features, consider hardwiring JP1-6 to +5V. On the other hand, if you want to always use the hardware enable for each channel, connect JP1-6 to ground. You can also leave JP1-6 open and drive it from the host microprocessor if you prefer.

One final customization you may want to consider: The onboard LED (D1) lights when the processor is running (which should be all the time). This is a useful diagnostic when you first complete the board, but once it is working you may wish to remove the LED to conserve power.

In any event, make your connections according to this chart:

| *Pin* | *Description* |
|---|---|
| JP1-1 | Ground |
| JP1-2-5 | If JP1-6 is low, set JP1-2 high to enable servo #7, JP1-3 enables servo #6, JP1-4 enables servo #5, and JP1-5 enables servo #4 |
| JP1-6 | Set high to enable all channels; low to read individual enables |
| JP1-7-10 | If JP1-6 is low, set JP1-7 high to enable servo #3, JP1-8 enable servo #2, JP1-9 enables servo #1, and JP1-10 enables servo #0 |
| JP1-11 | +5V (supplied by board if 7805 installed) |
| RB0 | Connector for Servo #0 |
| IC2-9 | Connector for Servo #1 |
| RB2 | Connector for Servo #2 |
| RB3 | Connector for Servo #3 |
| RB4 | Connector for Servo #4 |
| RB5 | Connector for Servo #5 |
| ICSP-5 | Connector for Servo #6 |
| ICSP-4 | Connector for Servo #7 |

## Operation

The GP4 connects to a PC or other host device via the RS-232 port. The GP4 is a DCE device, so you can use a straight cable to connect directly to a PC. The PC sends commands to the board using 9600 baud serial data. To make things simpler, you can download libraries that interface your programming language to

the GP4 seamlessly. Most Windows-based languages can use ActiveX controls or DLLs, and both are supplied.

The ActiveX control appears as a scroll bar that can control one servo. Simply set the Channel property and the scroll bar will move the indicated servo. If you are controlling more than one servo at a time, you will probably want to hide the control and use the methods to directly control each servo so that your user interface is consistent.

## Reference

The GP4 has a simple command structure. When using the libraries, the commands are even simpler. Each section below describes a command and how to execute it either directly or via the standard libraries. Unless you are trying to write your own library for a different platform, you probably won't care about the direct commands.

When using either ActiveX library, you must set the commport parameter to match the port the GP4 is using (e.g., set to 1 for COM1).

If you are directly controlling the GP3 (and not using the library), you need to know a little bit about how the protocol works. To prevent synchronization errors, each command byte starts with a 0 bit. Each data byte starts with a 1 bit. In general a command has the binary format (the raw commands are specified in binary):

0 C C C P P P L

Where C C C is the command code (0 to 7), P P P is typically a channel number (0-7) and L is an option bit. If additional data is required, it is sent **before** the command and will have the format:

1 N N N N N N N

All communications are at 9600 baud, 8 bits, 1 stop bit, and no parity.

By default, the GP4 is disabled and each channel is set to output 1.5mS pulses (that is, each channel is set to 0).

For the purposes of this manual, a byte has 8 bits and a word has 16 bits. Raw commands are shown in binary. Each raw command is a single byte, not 8 ASCII characters.

When sending each byte of a command you must wait for a response from the GP4. All commands return a 1 except for the ReadChannel command which returns the value of the requested channel in the same format you would use to set it. No other handshaking is supported or required.

## Command Reference

These commands are provided in the standard ActiveX library. In addition, each command discusses the raw command bytes you would send to duplicate the command if you are writing your own libraries.

**Reset –** This command resets the GP4 to its default state (all channels enabled, each set to 1mS, all outputs disabled). *Raw command: 0 0 0 0 0 0 0 0.*

**SetPosition(channel, position) –** Sets a servo's (from 0 to 7) to a position from -50 to 50. The position -50 corresponds to 1mS, 0 to 1.5mS, and 50 to 2mS. *Raw command: 1 S P P P P P P, 0 0 1 0 C C C 0 where C C C is the servo number (0-7), S is the sign (1 is negative), and P P P P P P P is the magnitude of the position (0-50).*

**EnableChannel(chan, enable) –** Enables or disables a particular channel. Note, the entire chip is disabled by default until you issue an *Enable* command. *Raw command: 0 0 0 1 C C C E where C C C is the servo number (0-7) and E is 1 to enable or 0 to disable.*

**SetMask(mask)** – This advanced command allows you to disable multiple servos at once. For each servo you want to disable, put a 0 in the corresponding mask bit (mask ranges from 0 to 255). This works like *EnableChannel* but it sets all 8 enables at once. The chip enable and any hardware enables must still be active. *Raw command: 1 E E E E E E E, 0 0 0 0 0 1 0 L where E E E E E E E are the enable bits for servos 7 to 1 and L is the enable bit for servo 0.*

**Enable(enable) –** This function enables or disables all outputs. Disabled outputs go low and stay low. Note: this is not the same as the *Enabled* property which enables or disables the scroll bar user interface. *Raw command: 0 0 0 0 0 0 1 E where E is 1 to enable or 0 to disable.*

**ReadChannel(channel) –** This function reads the value from a given channel. *Raw command: 0 0 1 1 C C C 0 where C C C is the servo number.*

**Comport –** This property sets the port that the control uses to talk to the GP4. For example, if you are using COM1, set this property to 1. The ActiveX control only opens the port when required and keeps it closed at other times unless *HoldPort* is true.

**HoldPort** – If this property is set to True, the library opens the serial port and holds it open until you change the property to False. When this property is False, each command that communiates with the GP4 opens the serial ports, performs a transaction, and closes it again. This property must be True if you want to monitor the state of channel 2 using *Servo2FB*.

**Channel –** This property sets the default channel that the built-in scroll bar uses. This also changes the *Value* property's meaning to reflect the selected channel.

**Value** – This property reflects the actual value of the selected channel, and allows you to change the value of the selected channel without calling *SetPosition*. Note that if you don't reset the GP4 when your program starts, you should read *Value* to prime the built-in scroll bar. This prevents the control from reading the GP4 before you set the port. Of course, if you are hiding the control, you don't need to worry about this.

**Reverse –** Normally, the built-in scroll bar is set to -50 when it is all the way to the left and to 50 when it is full to the right. If this property is True, the scroll bar reverses so that 50 is on the left and -50 is on the right.

**SmallChange –** This property controls how many units a small change to the built-in scroll bar causes.

**LargeChange** – This property controls how many units a large change to the built-in scroll bar causes.

**Servo2FB –** This property (which is only accurate while *HoldPort* is true) reflects the state of servo #2's output.

**Change** – The ActiveX control raises this event when the built-in scroll bar changes.

**Scroll** – The ActiveX control raises this event when the user drags the built-in scroll bar.

**S2FBChange** – This event (which only regularly occurs when *HoldPort* is true) indicates that the *Servo2FB* property has changed. Note that this generates nearly 50 events per second, so the event handler should be fast.

# Quick Reference

## *Methods/Functions*

| Name | Parameters | Return | Description and Notes |
|---|---|---|---|
| Reset | none | none | Resets GP4 |
| SetPosition | servo, position | none | Sets position of specified servo |
| EnableChannel | servo, enable | none | Enables or disables indicated servo |
| SetMask | Mask | none | Sets output mask |
| Enable | enable | none | Enables or disables entire GP4 |
| ReadChannel | channel | position | Reads channel position |

Notes:

Servo numbers are from 0 to 7

Position information is from -50 to 50

Enable is True or False

## *Properties*

| Property | Notes |
|---|---|
| Comport | Sets the port used to talk to the GP4 |
| HoldPort | Set to True if you want the control to open the port and hold it open |
| Channel | Selects the channel for the built-in scroll bar |
| Value | Reflects the value of the channel associated with the built-in scroll bar |
| Reverse | Set to true to reverse the sense of the built-in scroll bar (left edge is 50, right edge is -50) |
| SmallChange | The value to add to the position for a small scroll bar change |
| LargeChange | The value to add to the position for a large scroll bar change |
| Servo2FB | The current state of the servo 2 feedback channel (if connected); only accurate when HoldPort is true. |

## *Events*

| Event | Notes |
|---|---|
| Scroll | Fires when the user moves the scroll bar thumb |
| Change | Fires when the user changes the scroll bar value |
| S2FBChange | Fires when Servo2FB changes (passes the value of Servo2FB) |

## Example Visual Basic Code

The available libraries include an ActiveX control and an ActiveX DLL. The control is easy to use when you have a form-based program. The DLL is easier to use in cases where you don't have a form. Here is a simple example of using a control named Servo that references a GP3 ActiveX control.

```
Private Sub Button1_Click()

Servo.commport=7        ' set comm port

Servo.SetPosition 2,10  ' set servo 2 pos

End Sub
```

To use the ActiveX DLL, use AWCGP4DLL.GP4DLL as the PROGID (or add a reference to the DLL in your project).

## Tips

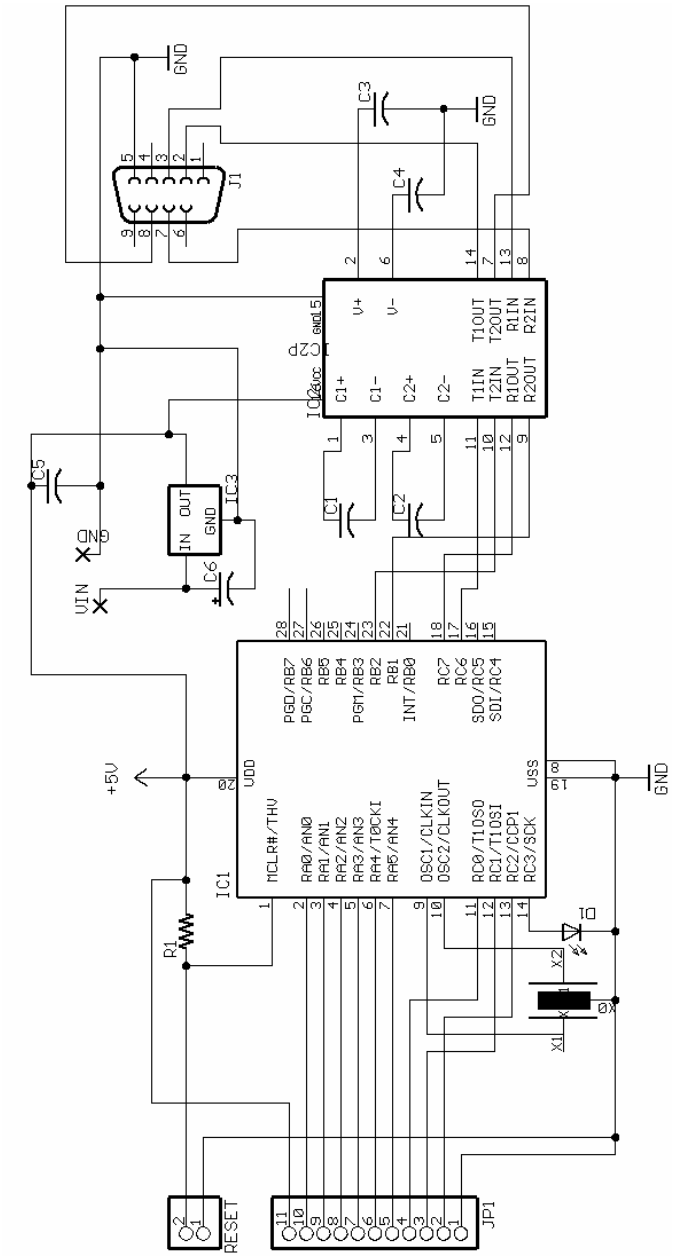
- You can wire a Molex or other similar connector to the I/O holes to make a neat, removable assembly.

- The ActiveX control has a built-in scroll bar that is useful when you want to control a single servo. If you want to control multiple servos (or want to provide your own user interface) set the control's Visible property to False and simply use the control's methods to perform the required tasks.

- Check our Web site for sample projects and the latest library files. See http://www.al-williams.com/gp4.htm.

## Notes

## Schematic


GND
C3
GND
C4
J1
C5
C6
IC3
IN OUT GND
VIN
C1
C2
IC2
V+
V-
C1+
C1-
C2+
C2-
T1IN
T2IN
R1OUT
R2OUT
T1OUT
T2OUT
R1IN
R2IN
IC1
MCLR#/THV
VDD
VSS
RA0/AN0
RA1/AN1
RA2/AN2
RA3/AN3
RA4/T0CKI
RA5/AN4
RB0/INT
RB1
RB2
RB3/PGM
RB4
RB5
RB6/PGC
RB7/PGD
RC0/T1OSO
RC1/T1OSI
RC2/CCP1
RC3/SCK
RC4/SDI
RC5/SDO
RC6
RC7
OSC1/CLKIN
OSC2/CLKOUT
+5V
R1
X1
X2
RESET
JP1