8/16 Channel

Servo Driver-Controller

Technical Manual Rev 1r0

e-Gizmo Servo Driver-Controller shown in full 16 channel configuartion.

Hobby analog servo motors are special type of motors that can be controlled to rotate at a speed and stop at a precise location. And once in position, it will stay in position as best as it could- it will automatically compensate for any force that tries to get it out of position. Hobby servos are not built for speed; these are usually equipped with gear head assemblies to obtain the strongest rotational force possible- the highest torque, at a specified rotational speed. These two main characteristics, controllable position and high torque, makes it a favorite actuator for robotics applications.

e-Gizmo 8/16 Channel Servo Controller is a universal servo controller that can control individual movements of up to 16 servo motors in a single board. You can use any number of boards (limited by your system controller or sequencer) to add servos as many as you like.

The servo driver-controller will work with any microcontroller running on a supply voltage of 3.3V to 5VDC.

1. Pin Description

The servo driver controller communicates with a host controller via a synchronous serial data interface. Your host controller does not need to have a hardware serial I/O interface; the fact is, the interface was designed for reliable data transfer using just ordinary I/O and a simple program routine. This will be discussed in detail in section 3.

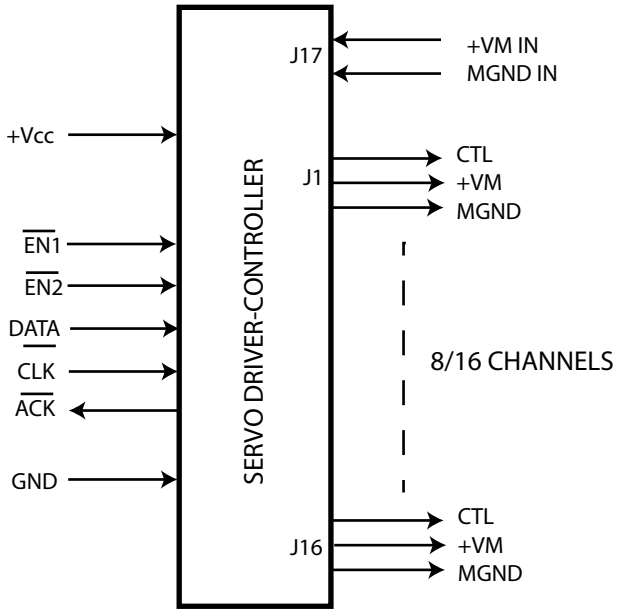


Figure 1.1 Servo Driver-Controller functional block diagram. Synchronous serial interface provides for a reliable data transfer with minimal I/O usage.

A simplified block diagram of the servo controller is shown in figure 1.1. Serial commands are sent to the servo controllers via the following pin interface:

$\overline{\text{EN1}}$ – Activates the chip controller for motor 1 to 8 to receive the serial command.

$\overline{\text{EN2}}$ – Activates the chip controller (if present) for motor 9 to 16 to receive the serial command.

DATA – Serial Data Input. Command input in serial format.

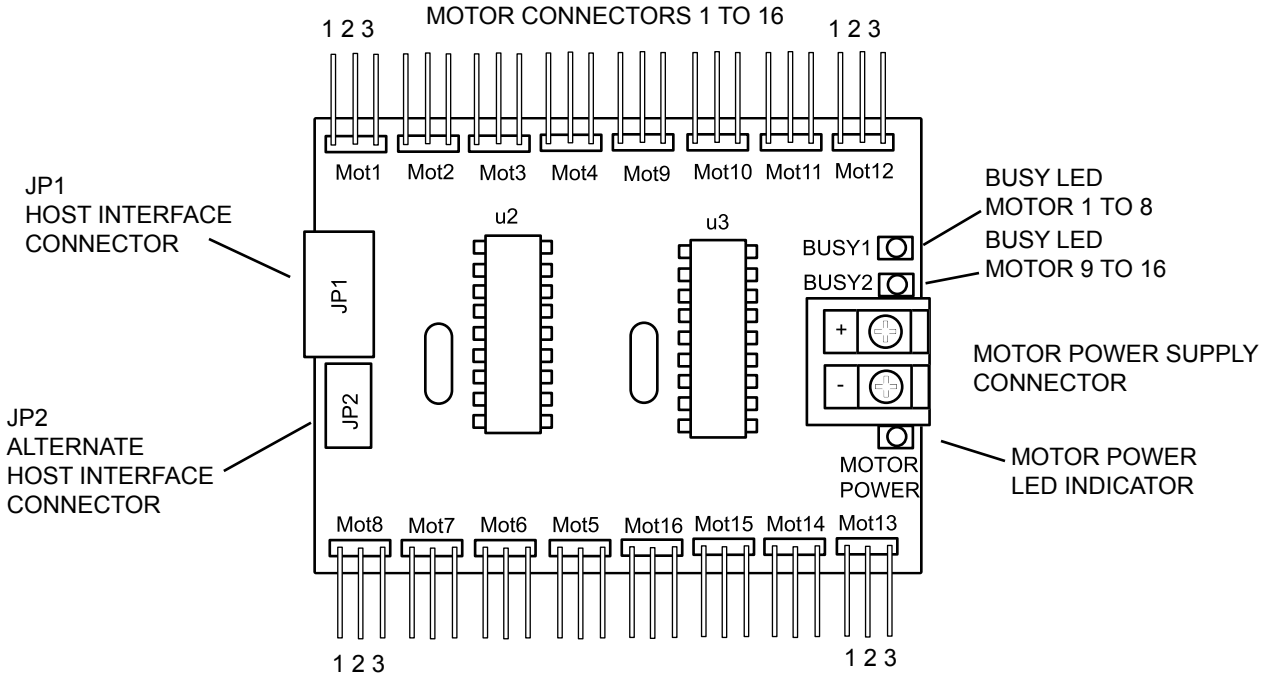
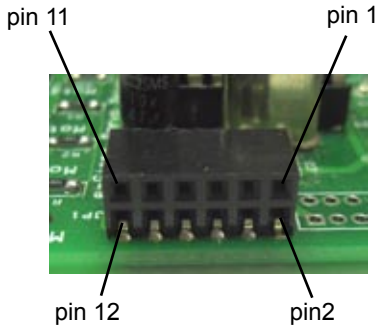


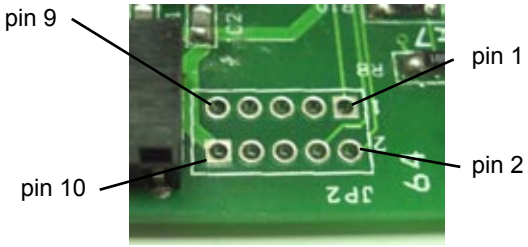
Figure 1.2. Layout of servo driver-controller with major parts and connectors identified.



pin 1 - EN1
pin 2 - ACK
pin 3 - EN2
pin 4 - CLK

pin 5,7,9,11 : +Vcc
pin 6 - DATA
pin 8,10,12 - GND

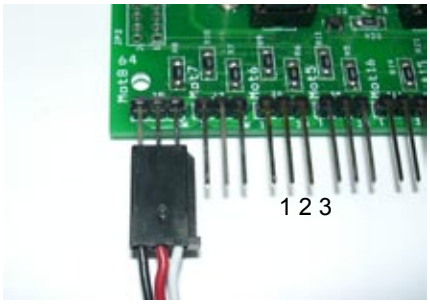
Figure 1.3. JP1 Host Interface connector pin identification.



pin 1 - EN1
pin 2 - ACK
pin 3 - EN2
pin 4 - CLK

pin 5,7,9 : +Vcc
pin 6 - DATA
pin 8,10 - GND

Figure 1.4. JP2 Alternate Host Interface connector pin identification.



pin 1 - MGND
pin 2 - +VM
pin 3 - CTL

e-Gizmo Servo motor

Black - MGND
Red - +VM
White - CTL

Figure 1.5. Servo driver-controller motor connectors are configured to readily accept e-Gizmo servo motor connector. This style follows the wiring arrangement most used by other servo motor manufacturers. Not all manufacturers follow this convention, however. It is your resposibility to determine the suitability of servos other than those recommended by e-Gizmo. Damage to circuit or servo motor may occur with improper connection.

$\overline{CLK}$ – Serial Clock input. Driving this pin High to low will cause the servo controller to read a bit of data from DATA input.

$\overline{ACK}$ – Acknowledge Output. The servo driver-controller momentarily drive this pin low to tell the host controller of a successful bit read.

+VCC – Servo driver positive power supply input. Connect to the +VCC of the host controller. The servo controller will work with Vcc voltage of 3.3V to 5VDC.

GND – Common ground. Connect to the host controller ground.

The Motor side has the following simple interface:

Motor Power Supply:

+VM in – Motor system positive power supply source.
MGND in – Motor system power supply ground.

Motor Connector (8 or 16 channels):

CTL – Servo Motor control output (White wire).
+VM – Motor positive supply (Red Wire).
MGND – Motor ground (Black Wire).

2. Circuit Description

The complete schematic diagram of the Servo driver-controller is shown in figure 2.1. The servo driver controller, in fact, contains two independent 8 channels servo controllers U2 and U3. These servo controller ICs , pre-programmed z8 microcontrollers, are each capable of controlling 8 motors (channels) simultaneously. The host controller picks the servo controller it wants to talk to by way of ENable lines EN1 and EN2. EN1 selects U2, sending control to motors 1 to 8. In a similar manner, EN2 selects U3, controlling motors 9 to 16.

LED D3 and D4 provide visual indication of the servo controller status. This LED lights up whenever the corresponding servo controller IC is busy doing a task. Resistor R1 to R11 provides limited protection to U2 against outside faults. R12 to R22 does identical job for U3.

The servo driver controller is available in 8 channel and full 16 channel version. One servo controller IC is simply omitted for the 8 channel version.

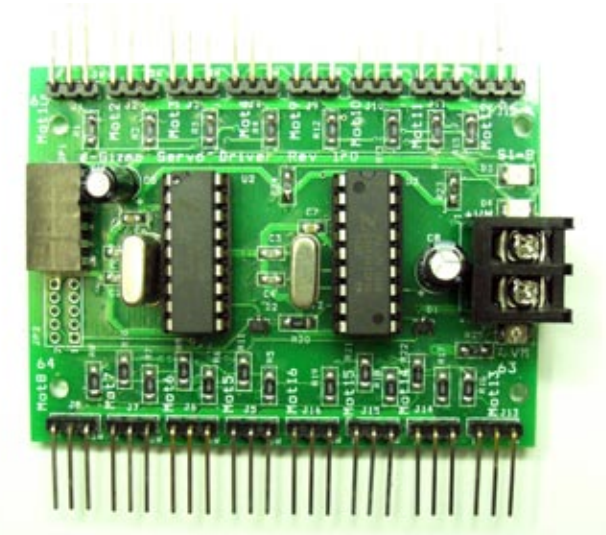


Figure 1.6. Photo detailing the layout and parts of Servo Driver-Controller.

Figure 1.7. Motor power supply connector J17 and LED indicators.

Figure 1.8. Host interface connector JP1 allows the host controller board to link directly with the servo driver-controller without the use of wires.

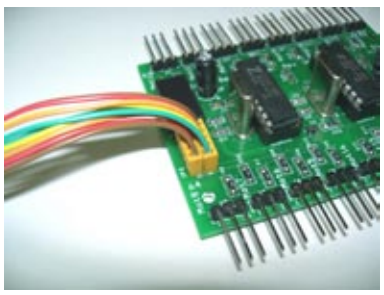
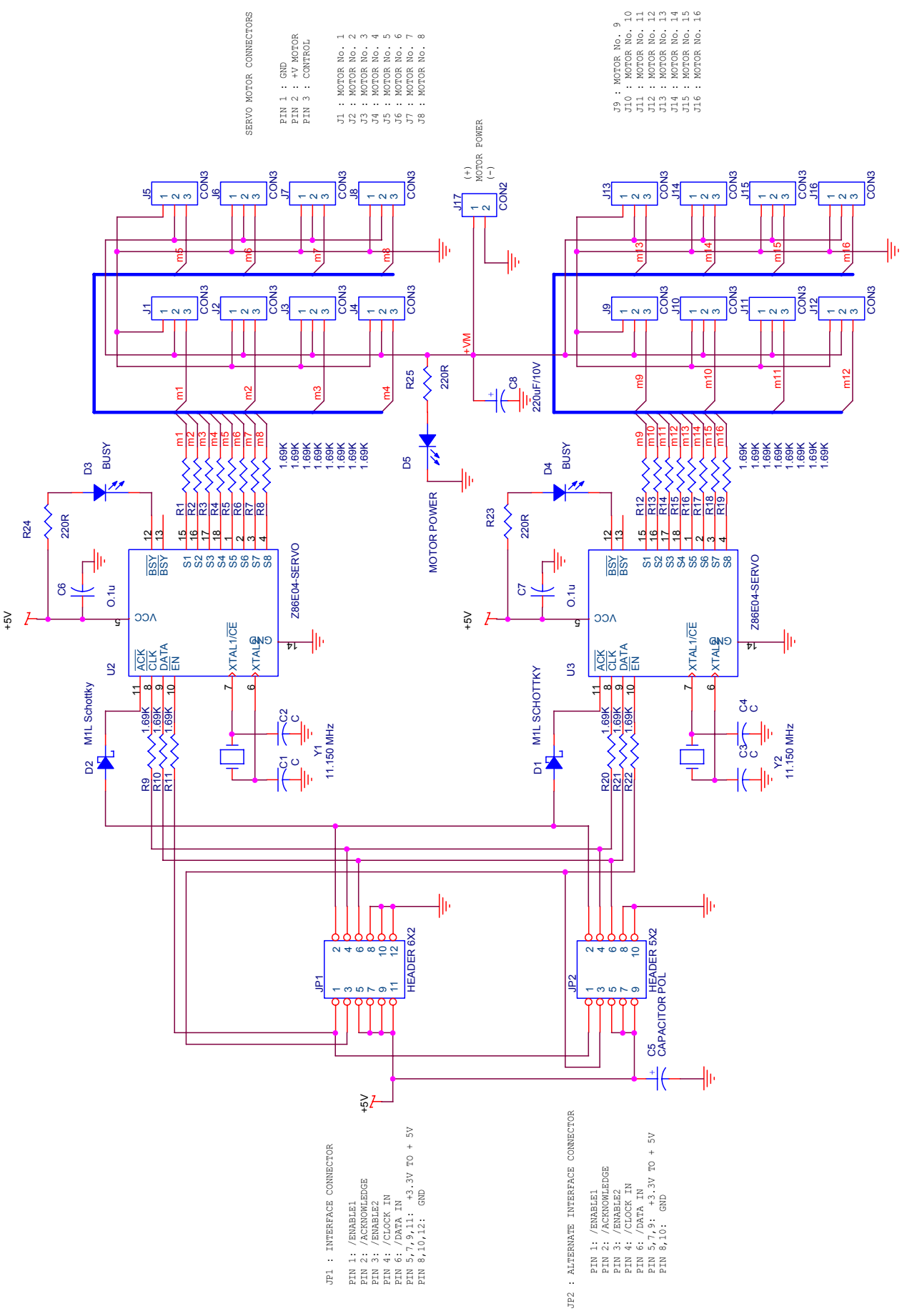


Figure 1.9. Use the alternate host connector JP2 if your host controller location or configuration does not allow for a direct mating connection.



3. Serial Data (Command) Transfer

This section describes in detail the serial data transfer process. This sequence must be strictly followed by your host controller to ensure a reliable data transfer. Figure 3.1 illustrates the timing sequence. Each timing milestones are labeled with numbers indicating the order of occurrence. One command set consists of two bytes (16 bits) of binary stream.

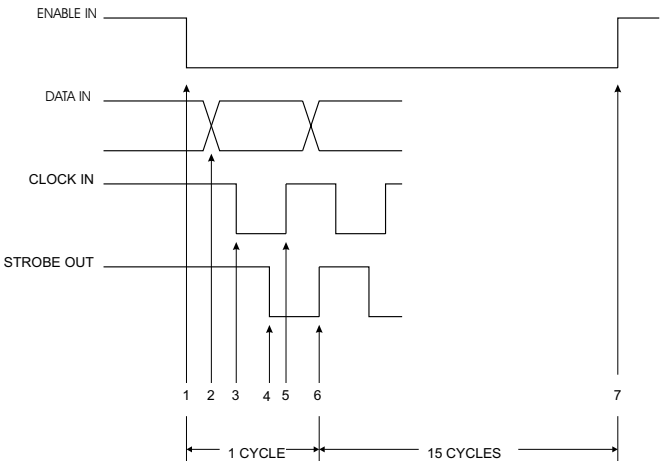


Figure 3.1. Synchronous serial transfer timing diagram.

Note that this transfer is implemented by softaware and not by any dedicated serial hardware peripherals, hence, no hardware timing setup and delays are specified. Simply write your host controller routine such that it strictly follows this timing sequence, and you should encounter no problem at all communicating with the servo driver-controller.

1. Start a transfer, switch the EN1 or EN2 input from high to low.

   The servo controller IC will respond to serial transfer only when its corresponding EN input is low. The initial HIGH to LOW transition of EN input initializes the servo controller to recognize the following data bit as bit 0. Pulling EN input back to logic HIGH at any time stops the transfer an puts the servo controller ready for a new transfer at the next HIGH to LOW transition.

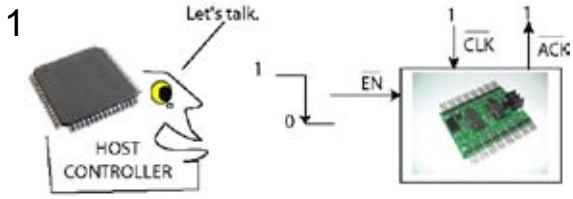
2. Set appropriate data bit at DATA input.

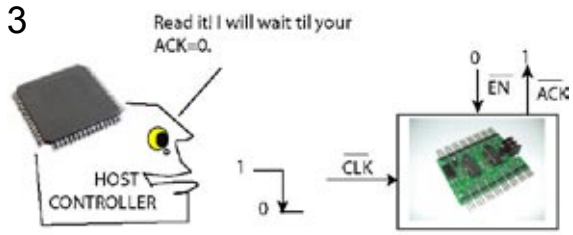
3. Switch clock input from high to low.

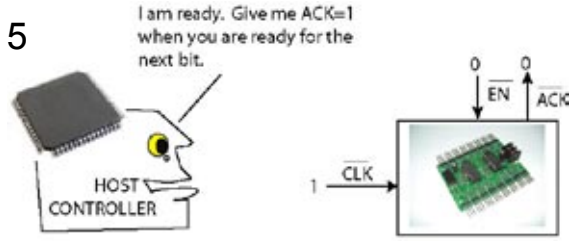
   CLK input High to Low transition signals the servo controller IC to read whatever data now appearing on DATA input. The servo controller

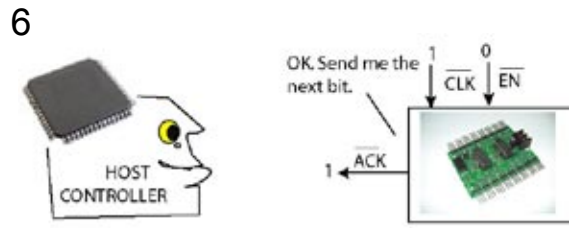
4. Wait until the ACK pin switch turns Low.

COMMUNICATION START









Repeat Steps 2 to 6 15 more times to complete 16 bit transfer

COMMUNICATION ENDS

5. Switch the clock input back to High.

6. Wait until ACK pin turns High.

7. Repeat the cycle starting from step 2 to send the remaining 15 bits.

8. Switch the Enable pin High to end the transfer, and ready the system for a new transfer.

4. Data Transmission Format

One servo driver-controller command packet consists of 16-bit (2 bytes) data organized as follows:

Bit 15 to bit 12 : Motor address
Bit 11 to bit 8 : Command
Bit 7 to bit 0: 8 bit command parameter

Command packet is transmitted with bit 0 first.

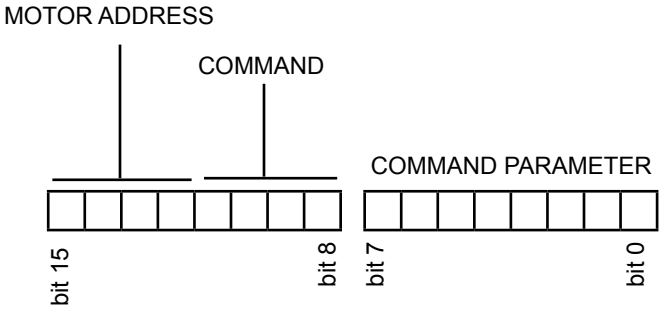


Figure 4.1. Servo driver-controller command transmission format.

4.1 Motor address – Specifies one of 8 motor where the command will take effect. Note that each servo controller IC can control up to 8 motors only.

Valid values:

Binary	Hex	
0001	1	- Motor No. 1
0010	2	- Motor No. 2
0011	3	- Motor No. 3
0100	4	- Motor No. 4
0101	5	- Motor No. 5
0110	6	- Motor No. 6
0111	7	- Motor No. 7
1000	8	- Motor No. 8

4.2 Command- Specifies the command to be executed by the specified motor (address).

Valid values in Binary(Hex):

0001 (Hex 1) - Set Speed specified by command parameter.

0010 (Hex 2) – Set Position specified by command parameter

0011 (Hex 3) – Set all 8 motors in neutral position

1000 (Hex 8) – Servo ON/OFF specified by command parameter

4.3 Parameter – command parameter. This is discussed in greater detail in the next section.

5. Command Description.

5.1 **Set Speed** – set rate of rotation. The rate of rotation can be adjusted in increments of 2.5mS per degree of rotation.

Valid command parameter values:

128 – 255 (Fastest - Slowest)

Rotation rate can be approximated as follows:

Rate= (Parameter-128)* 2.5ms per degrees.

With Parameter=128, the rotation speed is limited by the maximum speed of the motor (It is, in reality, not 0 ms per degrees). All succeeding motion commands for the addressed motor will run at the last specified speed.

5.2 **Set position** – set the angular position of the motor.

Valid command parameter values:

50-250 (Hex 32 to FA).

Note that these values correspond to pulse widths of 0.5 to 2.5ms. 1.5ms (150) is the center (neutral) position, equivalent to 0 degree angular position. Almost all hobby servo complies with this “standard”. The actual angular position can be computed by the following simple formula:

Angular position = (Parameter -150) * 0.9 degrees

This formula tells us that the motor can be positioned +/-90 degrees from its centered position at 0.9 degrees step. Note that the resolution is twice that of standard 1.8 degrees per step stepping motor.

Note: Not all servo motor can travel this full +/-90 degrees range. Some motors are restricted to +/- 45 degrees rotation. Operating the motor outside its travel range can damage the motor. It is your responsibility to keep the motor drive within its turning range.

5.3 **Set all (eight) motors in neutral (center) position**. This command will position all eight motors of the selected servo controller to center position.

Valid command parameter values:
0 – 255 (Hexadecimal 00 – FF)

This command does not require a motor number and command parameter, but the host controller must transmit all bits just the same, since the servo controller always expects a 16 bit transfer.

5.4 **Servo ON/OFF** – Turn ON or OFF selected servos. Selection is done through command parameter as follows:

Command parameter:

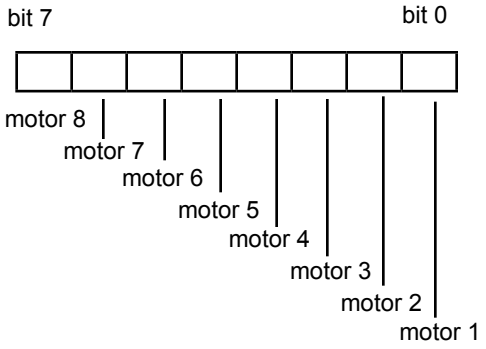


Figure 5.1. Servo ON/OFF parameter definition. Each bit represent a motor as shown. Set a bit to 1 corresponding to the motor you want to turn ON and use.

Set the corresponding bit of the servo you want to turn on to 1. See the usage example discussed next for more details. The servo controller defaults to all OFF during power ON- this will prevent your robot from making unexpected motions when switched ON. Among the things your host controller has to do first is to send the servo ON command before the controller is put into use. The motor number address is not used.

6. Command exercise.

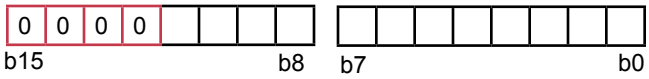
A step by step example on how to assemble the 16 bit command set will be shown in the following exercise.

Exercise:

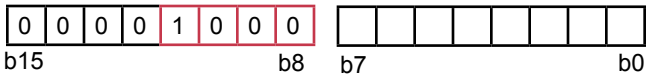
Build a sequence that will command servo motor 1 and 7 to rotate to +16 and -18 degrees position respectively.

1. Assemble command to turn ON servo.

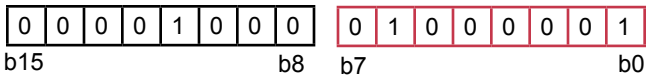
   1a. The motor address is not used in this command. Let us just put zeroes in this field.



   1b. Next, assemble the command field. Put 1000 in this field for the servo ON/OFF command.



   1c. To turn servo motor 1 and 7, we put a 1 in their corresponding field.

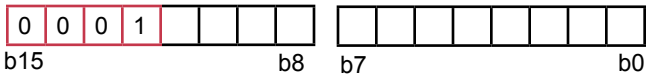


   1d. The 16 bit sequence for this command is completed. This binary number translate to Hex 0841. Your host controller should then send this sequence to turn ON the servo 1 and 7.

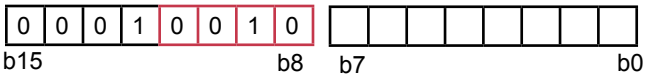
Note that once you turn ON a servo, it will stay in ON state until you turn if OFF. You do not have to send this command for every movement.

2. Assemble motor 1 position command.

   2a. Motor address = 0001, send this command to motor 1.



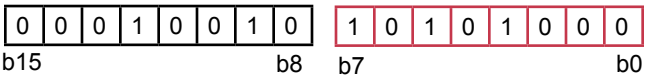
2b. Command = 0010, position command.



2c. From the formula given in section 5.2, compute the parameter corresponding to +17 degrees.

Parameter = (Angular position /0.9) + 150
Parameter = (17/0.9) + 150 = 167.777, rounded of to 168.

The binary equivalent of 168 is 10101000. Put this in the command field.



2d. The complete command set will then appear as below. This is equivalent to Hex 12A8.

The previous exercise has just shown you how to construct the 16 bit sequence in the most elementary way. You should have taken notice by now that it will be far more efficient to assemble this sequence using hexadecimal numbers directly. That we will do in the last sequence of this exercise that follows next.

3. Assemble motor 7 position command.

3a. Motor address = 0111 (hex 7) command corresponding to motor 7.

Command Packet in Hex : 7

3b. Command = 0010 (hex 2) specifying position command.

Command Packet in Hex : 72

3c. From the formula given in section 5.2, compute the parameter corresponding to -18 degrees.

Parameter = (Angular position /0.9) + 150
Parameter = (-18/0.9) + 150 = 130.

The Hex equivalent of 130 is 82. The complete command set will then appear as shown below:

Command Packet in Hex : 7282

In summary, to execute movements as described in the exercise, your host controller should send the following 16 bit command set in the following ordered sequence:

Hex 0881 – turns on servo 1 and 7
Hex 12A8 – rotate motor 1 to +16 degrees position
Hex 7282 – rotate motor 7 to -18 degrees position

The time that will take to send these commands to the servo-controller is so short compared to the speed of motion that it will appear to you all three commands are happening at the same time. In practice, if you want one motion precedes another, you should insert a delay loop between transmissions of command sets.

7. GENERAL SPECIFICATIONS

Operating Voltage Vcc :	3.3V to 5V
Operating Voltage Vm :	Motor Specified
Operating Current (Control circuit):	< 75mA
Motor Output Refresh rate:	50Hz
Pulse width output range:	0.50-2.5 ms

8. Bill Of Materials

Item	Quantity	Reference	Part
1	4	C1,C2,C3,C4	27pF SMD0805
2	1	C5	47uF/10V ECAP
3	2	C7,C6	0.1u SMD0805
4	1	C8	220uF/10V
5	2	D1,D2	M1L Schottky
6	2	D3,D4	LED GREEN SMD1210
7	1	D5	LED GREEN SMD1210
8	1	JP1	HEADER 6X2
9	1	JP2	HEADER 5X2 (optional)
10	16	J1,J2,J3,J4, J5,J6,J7,J8, J9,J10,J11, J12,J13,J14, J15,J16	
11	1	J17	Screw Terminal 2 way
12	22	R1,R2,R3,R4 ,R5,R6,R7,R8, R9,R10,R11, R12,R13,R14, R15,R16,R17 ,R18,R19,R20, R21,R22	1.69K SMD1206
13	3	R23,R24,R25	220R SMD1206
14	2	U2,U3	Z86E04-SERVO IC
15	2	Y1,Y2	11.150 MHz Crystal

9. PCB Layout

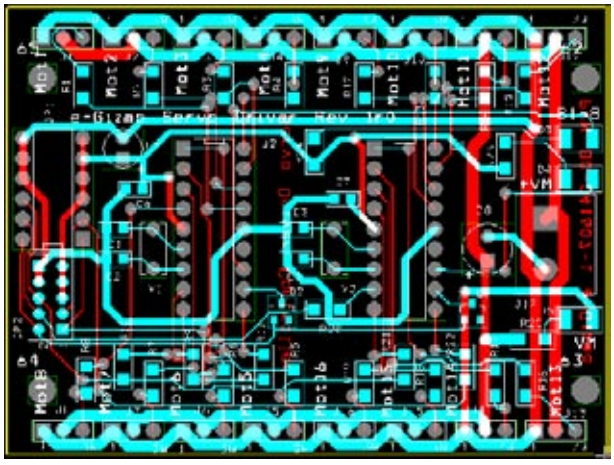


Figure 9.1. e-Gizmo Servo Driver-Controller composite layout.

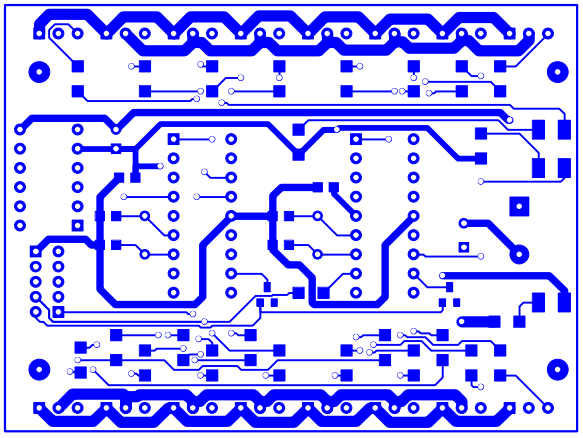


Figure 9.2. Component side copper layout shown actual size.

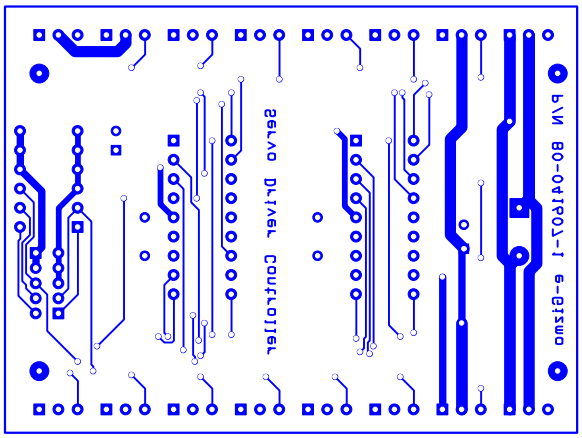


Figure 9.3. Bottom side copper layout.

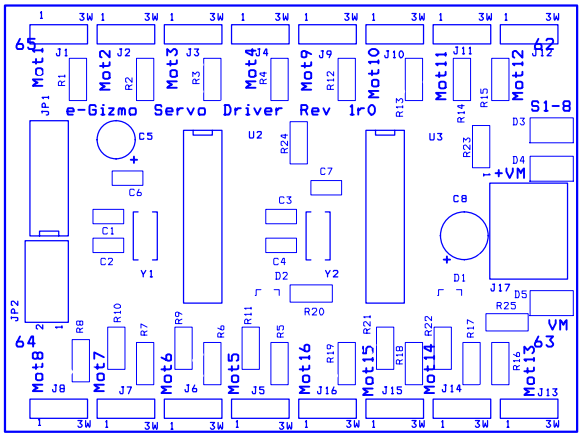


Figure 9.4. Component Silk Screen layout.