LightMachinery

GADGETMASTER ™

User's Guide

MC-4501 rev 6.1

Introduction

The GADGETMASTER is designed to allow simple, safe and inexpensive control of motors and other devices. The GADGETMASTER connects to the printer port of any PC, by sending commands to the printer port the user controls motion and can receive information from sensors and switches. The GADGETMASTER is normally configured to have eleven output lines which can be used to turn on lights, DC motors, buzzers, speakers or to run stepper motors and eight input lines to provide a means to track position or the status of other switches and sensors. The GADGETMASTER can also be configured to have 12 outputs and 4 inputs. This allows the GADGETMASTER to run up to 3 stepper motors.

Caution

Your computer could be damaged by misusing this board. Your computer and the board should be turned off whenever connections are being made to the board. The protective cover on the back of the board is there to prevent inadvertent connection of the PC printer port to 12 volts or other voltage sources, do not remove this cover. Under no circumstances should the board be connected to 115 volts from an electrical outlet. Use only the power supply provided with the unit.
The GADGETMASTER is designed to control motors and other similar devices, when these motors are connected to mechanical devices such as gears and pulleys enough torque is generated to cause serious damage to people and property. Please think about how you are using the GADGETMASTER and use it safely.

Note: Do not connect the outputs to devices that will draw more than 0.24 Amps continuously. This means that external devices should have a resistance of at least 50 Ohms. Otherwise overheating of the driver chips will result. Care should be taken to ensure that this rule is followed whenever devices not supplied by LightMachinery are being connected.

General Description

Power Outputs

The outputs, numbered 1 through 12, are either *high* (12 volts) or *low* (grounded). The Outputs will conduct electricity from the high output through a motor or light bulb or other device to a low output (or to ground). If both outputs are set to high or both outputs are set to low then no current will flow and the device will not be turned on.

Sensor Inputs

The input lines, numbered 1 through 8, are 'pulled up' to 5 volts by the board through a 10 kohm resister. This means that when the input lines are grounded the lines change states and that a small amount of current will flow to ground. The input lines on the printer port are usually used for things like "out of paper" and "printer error". You can use these input lines for monitoring sensors to measure position, light/dark or switch open/closed.

Configuring the GADGETMASTER

The GADGETMASTER can be configured to either have 12 outputs and 4 inputs OR 11 outputs and 8 inputs. To change from one configuration to the other simply move the 'jumper' next to Input 1. Position A will enable 11 outputs and 8 inputs while position B will enable 12 outputs and 4 inputs (inputs 5 to 8).

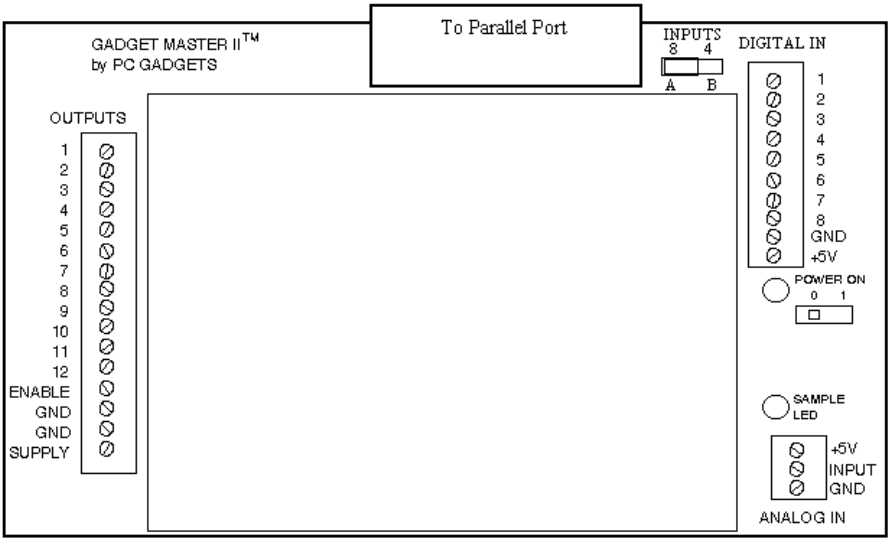
Connecting the GADGETMASTER to your Parallel Port

Connecting the GADGETMASTER to your PC is easy, just plug one end of the 25 pin cable into your printer port on the back of your PC and the other end into the same connector on the GADGETMASTER . The 12 Volt power supply plugs into the DC connector. Motors and lights etc. are connected between any two of the outputs or one of the outputs and ground (or supply).

GADGETMASTER Software

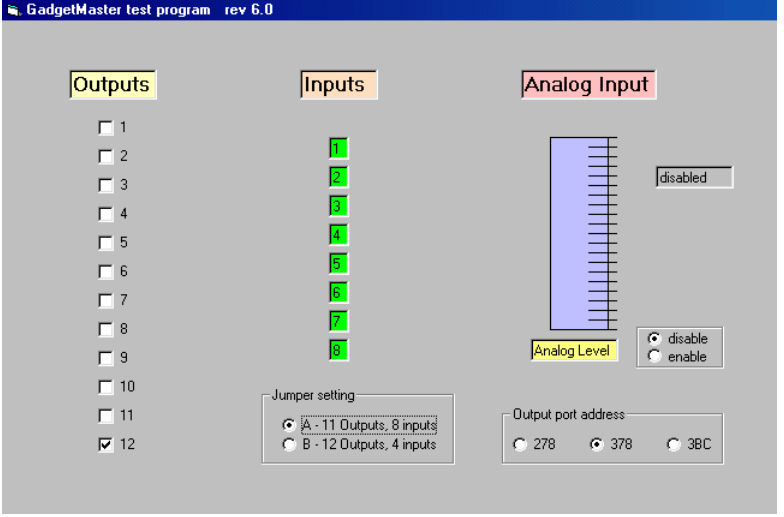
All of the software in this User's Guide, as well as new revisions to this guide, are available free of charge from our web site at www.lightmachinery.com in the *download* section.

GADGETMASTER Connection Diagram



Using the GadgetMaster Test Program – gmtestv6.exe

Gmtestv6.exe is a simple Windows interface that enables you to turn on any output and to monitor all 8 inputs to the GADGETMASTER . The program requires the file *inpout32.dll* to be saved in the same directory as gmtestv6.exe.



To activate one of the outputs just click in box next to the output number that you want to activate. When a sensor input is connected to ground the input number will change from red to green. The analog input thermometer is a volt meter that displays the voltage level connected to the Analog Input. Measuring the voltage takes about ½ a second so when the Analog input is enabled the 8 sensor inputs are monitored every second or so. To speed up the response time of the program just disable the analog measurement.

Sensor Inputs

Switches

Each of the inputs to the GADGETMASTER is connected to 5 volts through a 10k ohm resistor. When one of the input connections is connected to ground, your PC will see the input change states. This can be quickly seen by connecting a wire to one of the input lines and touching the wire to one of the 3 ground connections on the board. You will see the checkbox for that sensor on your screen change states and change from red to green. Simple mechanical switches can be connected between any of the inputs and ground, when the switch is closed the check box will change color.

Photosensors

Variable resistance photosensors or photocells can also be connected between any of the inputs and ground. These devices change their

electrical resistance depending on the amount of light that is shining on them. You can connect a photocell between any of the inputs and ground. When the resistance to ground is lower than the resistance to 5 Volts (10k ohms) then the input will switch states and the checkbox will change from red to green. The sensors can be very useful not just because they can sense light and dark but because they are non-contact, they never wear out. Try connecting a photosensor between the analog input and 5 volts and a resistor (5k ohms or so) between the analog input and ground, the meter will now read the brightness that the photosensor is detecting.

Inputs 5,6,7 & 8

Parallel ports were originally designed with 4 inputs, the GADGETMASTER provides 8 inputs by multiplexing two sets of 4 inputs. When the Jumper on the board is set to the 'A' position then the 8 inputs are enabled. To read inputs 1,2,3 & 4 the user must output a 0 value for output 12. To read input 5,6,7 & 8 the user must output a 1 for output number 12. This will be explained further in the section on writing your own programs.

Power Outputs

The outputs are numbered from 1 to 12. The outputs are either high (12 Volts) or low (grounded). The outputs will conduct electricity from a high output through a motor or light to a low output. If both outputs are set to high or both outputs are set to low then no current will flow and the device will not be turned on.

Note: Do not connect the outputs to devices that will draw more than 0.24 Amps continuously. This means that external devices should have a resistance of at least 50 Ohms. Otherwise overheating of the driver chips will result. Care should be taken to ensure that this rule is followed whenever devices not supplied by LightMachinery are being connected.

Stepper Motors

The GADGETMASTER was designed with stepper motors in mind. Stepper motors are very useful in automation projects. Four phase stepper motors require 4 outputs, so the GADGETMASTER was designed with the option of enabling 12 outputs so that three stepper motors could be run at the same time.

DC Motors

DC motors are inexpensive and are ideally suited for high speed rotation. The GADGETMASTER was designed to enable the outputs to both source and sink current so that up to 12 DC motors can be powered in both forward and reverse.

Solenoids

Solenoids are great for projects that require a movement between only 2 positions, like pen-up and pen-down on a plotter. When a solenoid is energized the plunger extends (or contracts) a fixed amount usually less than 1 inch. There are also rotary solenoids which behave something like stepper motors. Rotary Solenoids rotate a fixed amount every time they are energized. They only require two connections but they are not usually reversible (reversable rotary solenoids have 3 wire connections).

Lights

12 volt lights are easy to power with the GADGETMASTER . Just connect the bulb between the one of the outputs and ground.

LED's

LED's a little more complicated because they need to be current limited. This means that a resistor is required in series with the LED so that too much current does not flow through the LED and burn it out. Most LED's will run on about 20 milliAmps so at 12 volts this would require a resistor of 600 ohms (V=IR) in series with the LED.

Buzzers

12 Volt buzzers can be hooked up in the same way as lights or DC motors. When the output is activated; Buzzzzzzzzz.

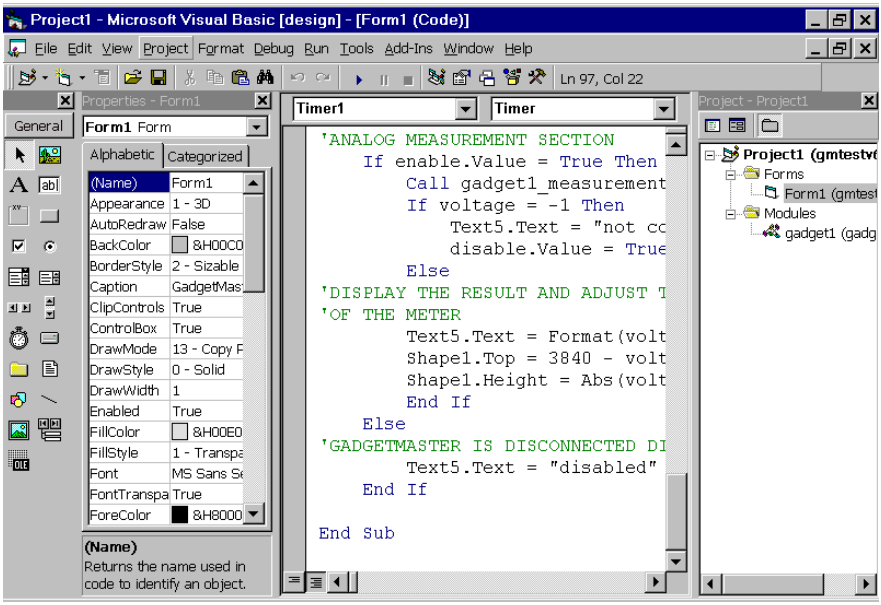
Speakers (or other coils)

While the GADGETMASTER will never make music hooking up speakers can be very interesting. When a speaker is energized (has current running through it) the cone is pushed forward. When the current is stopped, the cone relaxes back. By turning the output 'on and off' the speaker can be made to oscillate at different frequencies and create different tones. It can also be used as a way to position things by attaching them to the speakers cone...

Note: Speakers will have a low impedance, typically 8 ohms, and will draw too much current if connected directly to the outputs of the GADGETMASTER . Use either a resistor in series with the speaker or use a capacitor in series with the speaker. The capacitor will prevent any DC current from flowing through the speaker but will allow the speaker to jump whenever the output is energized or de-energized. If a series resistor is used you must ensure that it has a high power rating (several watts).

Writing your own programs in Visual Basic

OK, so you have run gmtestv6.exe and verified that the GADGETMASTER is working, now it is time to start writing your own programs. You can use any language that supports writing to the parallel port addresses, such as Basic or C++. In this User's Guide we will demonstrate the programming using Visual Basic. Since it is always easier to start your programming with a program that already works we suggest downloading the software code for the gmtestv6.exe program. Open the VB editor and open the file gmtestv6, it should look like this…



This program contains most of the elements to control simple inputs and outputs.

To make programming the GADGETMASTER even easier we have included the **gadget.bas** module. This module has subroutines that enable the use of simple commands turn the outputs off and on and read the state of the inputs. Inpout32.dll is still required to be in the same directory as your program (but inpout32.bas is not required). You can also modify these subroutines or write your own.

The following commands are enabled when gadget.bas is included in your program.

General Commands

Call gadget_output (a,b,c,d,e,f,g,h,i,j,k,l) - This command is used to turn the 12 outputs on or off. If a=1 and b=0 then output 1 is turned on and output 2 is turned off.

Example

'TURN OFF ALL THE OUTPUTS
Call gadget_output(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

Call gadget_input (m,n,o,p) - Thus command returns the status of the 8 input lines. If p=0 then input number 4 is off. If the jumper is on the B position then only the first four inputs are read. If the jumper is in the 'A' position and output 12 is 0 then inputs 1 – 4 are read, if output 12 is 1 then inputs 5 – 8 are read.

Example

```
'CHECK THE INPUTS 1 THROUGH 4
   Call gadget_input(in1, in2, in3, in4)
```

Example

```
'CHECK THE INPUTS 5 THROUGH 8
   Call gadget_output (a,b,c,d,e,f,g,h,i,j,k,1) 'output 12 must = 1
Call gadget_input(in5, in6, in7, in8)
```

Call gadget_measure (v, s) - This command measures the voltage on the analog input. 'v' is the voltage value on the analog input between 0 and 5 volts. If the GADGETMASTER is not connected then the value -1 is returned. The value of 's' is the number of loops performed by the computer during the 1/2 second cycle time required to measure the voltage on the analog input. It is a good representation of the computer's speed.

Example

```
'MEASURE THE ANALOG INPUT
      Call gadget_measurement(voltage, speed)
```

Call gadget_address("378") - This command sets the parallel port address. **It must be done once during the program**. Note that the address,;378, 278 or 3BC must be in quotations, since it is passed as text.

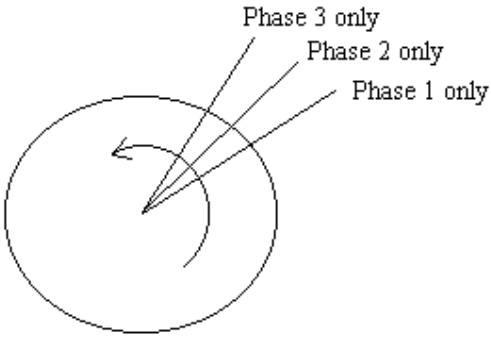
Example

```
'SET PORT ADDRESS TO 378 AS A DEFAULT
   Call gadget_address("378")
```

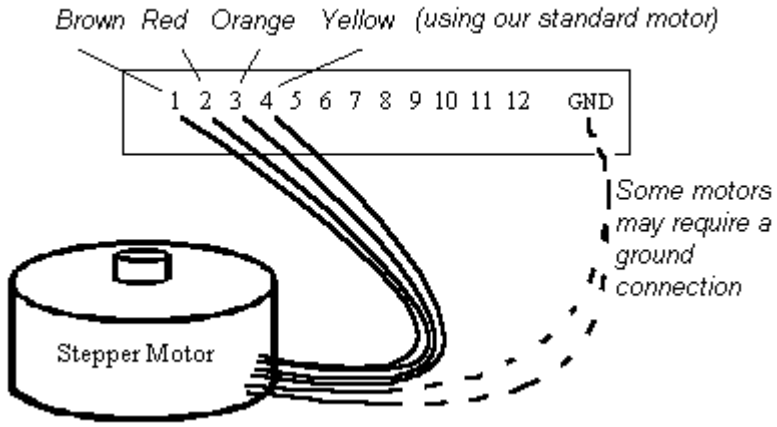
Controlling Stepper Motors

Stepper motors do not 'spin' when current is supplied to their windings, instead they 'step'. The angle of the step depends on the motor, common step angles are 1.8, 5, 7.5 and 15 degrees. Stepper motors are very useful in automation because their position can be precisely controlled. If you want a 15 degree stepper motor to turn 3 revolutions then simply step the motor 72 times (one revolution, 360 degrees, will require 24 steps). Stepper motors usually have 4 'phases' and these phases need to be energized one at a time to make the motor rotate. To make the stepper rotate backwards you simply energize the phases in the reverse order.

To connect the AIRPAX 15 degree stepper motor or the Eastern Air



Devices 1.8 degree stepper motor to the GADGETMASTER just follow the connection diagram below.

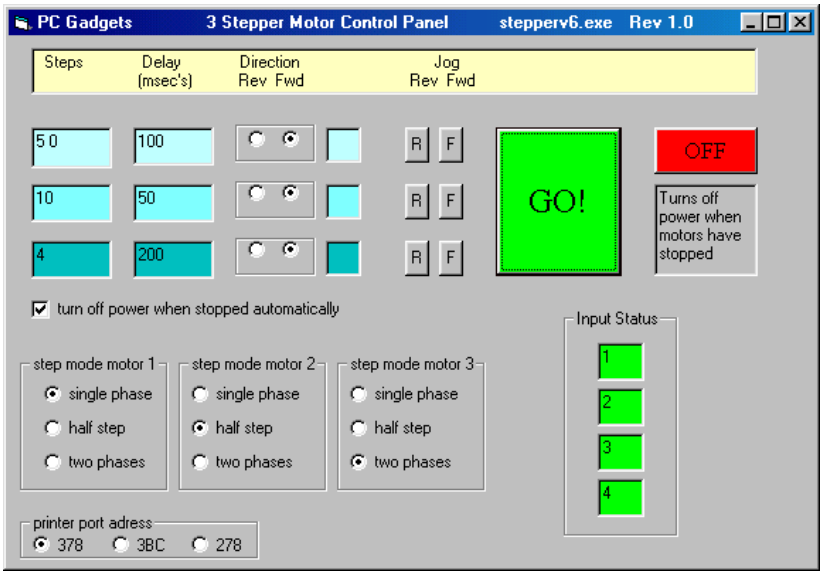


'Airpax' and 'Eastern Air Device' stepper motor connection diagram

By energizing output 1 then 2 then 3 then 4 then 1 then 2... The stepper motor will revolve. If at any time the order is reversed, the stepper motor will reverse. Stepper motors do not respond with lightning speed and care should be taken not to step them too fast or they will get out of sync with the outputs and lose their position. To determine how fast they can run under a given load just experiment with faster and faster speeds until they don't respond properly.

Using the Stepper Motor Control Program STEPPERV6.EXE

The program stepperv6.exe is included with the GADGETMASTER . When you run the program the screen will look something like this…



Just enter the number of steps you want the motor to step and press the 'Go Forward' or 'Go reverse' button. The delay value can be changed, the amount of the delay will depend on your computer. If you are using a Pentium computer you will need a delay that is 100 times longer than for older models of PC. Try entering higher and lower numbers. The pointer

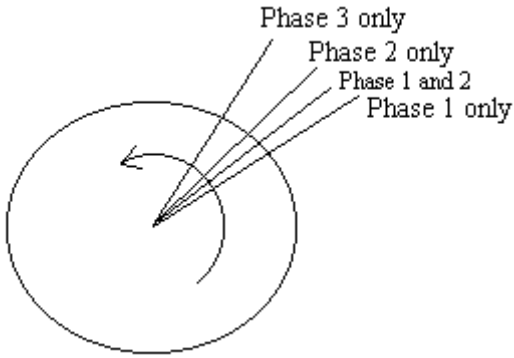
	One at a time step sequence				Two at a time step sequence			
	1	2	3	4	1	2	3	4
Output 1	X				X			X
Output 2		X			X	X		
Output 3			X			X	X	
Output 4				X			X	X

value is displayed to demonstrate the principle of stepping up and down through the four phases of the motor (and to help with debugging).

Stepper motors can either be run 'one phase on at a time' or 'two phases on at a time'. By energizing two phases at a time more current is supplied to the motor and therefore more torque is generated. When running with two phases on, adjacent phases are powered at the same time, as shown in the following diagram.

So, to move to position 1, using 'one at a time', output number 1 is powered. To move to position 1 using 'two at a time', energize both phase 1 and phase 2.

Note: Because twice the current is being supplied to the motor, the motor will run twice as hot.



Half stepping

Powering two phases at the same time moves the motor to a position half way in between the position of either phase, as shown in the diagram below. The motor can be 'half stepped by alternating single and double phase positions. When the motor is powered down it will naturally remain in one of the single phase positions.

Heat Management

You will notice that stepper motors can run quite hot. This is to be expected, however there are ways to reduce the heating of the motors. When current is supplied to the stepper motor, the motor heats up. When

the current is stopped, the motor cools down. When the motor is run with 2 phases on at a time the motor will run twice as hot. When a motor is sitting still with the current on, the motor will continue to heat up. **Stepper motors are normally turned off when they are not turning.** If the motor is powered off while running 'one phase on at a time', it will remain in the same position unless something turns the motor. If the motor is powered off while running 'two phases at a time' then it will slip over to one of the single phase positions. This is because the single phase positions correspond to magnetic detents in the motor. So, one way to cut down on the heat to the motor is to run in single phase mode and to turn off the motors when they are going to be left in a fixed position. This can also be done if the motor is run with two phases on provided the motor is allowed to move a half step when it is powered down. **Remember to turn off the motor at the end of your program with statements such as**

```
'TURN OFF ALL THE OUTPUTS
   Call gadget_output(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
```

Writing Stepper Motor programs in Visual Basic

Stepperv6.exe is written using gadget.bas and the complete source code can be downloaded from our web site. The following is a description of the stepper motor commands available in gadget.bas and the examples from the stepperv6.exe program. All of the source code software for both stepperv6.exe and gadget.bas can be modified for your application.

Note: Since 3 steppers motors can be connected to the GADGETMASTER , the motor needs to be specified in the command.
Stepper motor 1 must be connected to outputs 1,2,3 and 4.
Stepper motor 2 must be connected to outputs 5,6,7 and 8.
Stepper motor 3 must be connected to outputs 9,10,11 and 12.

Call stepper_move(motor, total_steps) - This command moves a stepper motor by the number of steps specified by 'total_steps'.

Example

```
'MOVE MOTOR 1 THE  NUMBER OF STEPS SPECIFIED IN
step1.Text IN FORWARD OR REVERSE
If rev1.Value = False Then                    'REVERSE OR FORWARD?
        Call stepper_move(1, Val(steps1.Text))   'FORWARD
Else
        Call stepper_move(1, -Val(steps1.Text))   'REVERSE
End If
```

Example
'JOG MOTOR 1, ONE STEP THEN POWER OFF ALL THE OUTPUTS
Call stepper_move(1, 1)
If power_off.Value = 1 Then Call gadget_output(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

Call step_mode(motor, mode As String) - This command specifies the 'mode' of the stepper motor. Mode can be specified in three ways;
"single" indicates that the motor will be run with a single motor phase on at one time.
"two" indicates that the motor will be run with two phases on at a time.
"half" indicates that the motor will be run by alternating 'two phases on' and 'one phase on'. In the half step mode the movements of the motor are half the normal step size.

The mode must be listed as a string (enclosed in quotes).
NOTE: The mode must be set before a stepper is moved.

Example
'SET THE STEPPER MODE FOR MOTOR NUMBER 1
If single_phase_1.Value = True Then Call Step_mode(1, "single")
If two_phases_1.Value = True Then Call Step_mode(1, "two")
If half_step_1.Value = True Then Call Step_mode(1, "half")

Call step_speed(motor, delay_time) - This command sets the speed of motor 1,2, or 3. The delay time is the amount of time in milliseconds between each step.

Example
'SET THE DELAY BETWEEN STEPS FOR MOTOR NUMBER 2
Call Step_speed(2, Val(delay2))

Call gadget_read_outputs(a,b,c,d,e,f,g,h,i,j,k,l) - This command reads the current value of the motor outputs. This can be useful if you are running stepper motors in combination with DC motors or solenoids. Since you may want to output direct commands to the DC motors but preserve the value of the outputs to the stepper motors.

Home Sensors

The concept of the 'home sensor' is essential to modern robotics and automation. Stepper motors will maintain their positions as long as they are not run too fast or are under powered for the job. But usually when a system is first powered up, the computer does not know where the device

or motor is starting from. A home sensor solves this problem (and demonstrates the inputs side and the output side of the GADGETMASTER working together). A home switch can be a simple mechanical switch. The stepper motor is stepped in one direction until the switch is triggered, then the motor is stopped and ready to start from a known position.

Sample program
This program is the same as the one above with the addition of the single line to check for the home sensor.

```
For x = 1 to 200                          'DO THIS LOOP UP 200 TIMES
Call gadget_input(in1, in2, in3, in4)     'READ THE INPUTS
If in1 = 0 then Call stepper_move(1, 1)   'IF INPUT IS OFF MOVE 1
STEP
Next x                                    'KEEP GOING…
```

Analog Input

Description
Sometimes measuring whether something is 'off' or 'on' isn't enough. Sometimes it's nice to know 'how bright' or 'how hot' something is. The Analog input provides you the capability of digitally measuring a voltage value between 0V and 5V. It's most accurate if the input is kept between 1V and 4V. It converts the steady voltage level into a rectangular pulse on pin 15. By measuring the width (duration) of the pulse you can determine the voltage value present on terminal 2 of the Analog Input. We think you may find this useful in lots of ways.

By connecting a fixed resistor between the analog input and ground and a variable resistor between the analog input and 5 volts a voltage divider is created. This means that if the variable resistance is small then the input will be close to 5 volts, if the variable resistance is large then the input will be closer to ground
An example is the measurement of temperature. A varistor is a resistor that has a resistance that varies with temperature. By connecting a suitable varistor between terminal 1 and 2 and a matching resistor between terminal 2 and 3 you can measure the temperature of the varistor.

Another example is the measurement of brightness using a photocell. As the light hitting the photocell is increased the resistance of the photocell decreases. By connecting a photsensor between the input and 5 volts and a

fixed resistor of a similar value between the input and ground and voltage divider is created. As the light level is increased the voltage will increase.

You can also measure the position of a potentiometer shaft by connecting the potentiometer across terminals 1 and 3 with the "wiper" connected to pin 2. Make sure to keep the 5V current drain low, we recommend using at least 1 Kohm.

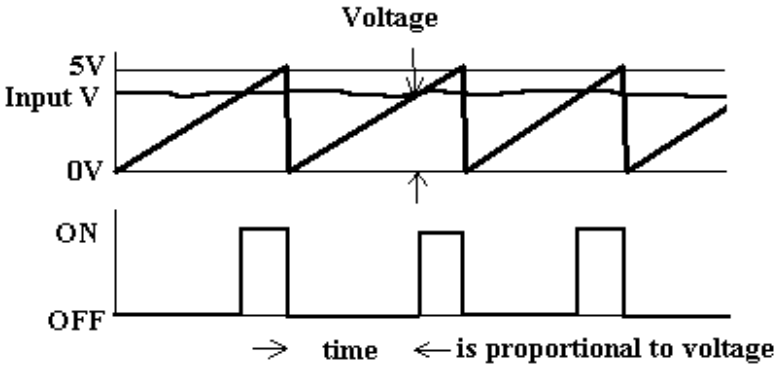
Connection

There are 3 terminal blocks installed on the GadgetMaster ; the 16 terminal "Output" block, the 10 terminal "Digital In" block and the 3 terminal "Analog In" terminal block.

The center terminal of the 3 is the actual input. Terminal 1 is connected to the 5V dc power supply, Terminal 3 is connected to the ground or 5V return. The voltage level present on the center terminal 2 is converted into a
square pulse and sent to the parallel connector on pin 15.

How it works

An oscillator on the GadgetMaster runs continually when the power is on. The output of the oscillator is converted into a ramp, or saw tooth shaped waveform. The saw tooth has a reasonably flat slope between 0V and 5V. The saw tooth is then fed into a comparitor. The comparitor 'compares' the ramp voltage to the analog input voltage. The comparitor output gives a 'low' output level when the analog input voltage value is higher than the ramp voltage. As soon as the ramp voltage goes above the input value the output goes high. This 'on' or 'off' signal is sent to the PC on the port status address with a value of 0 or 8. A longer 'off signal' means a higher



voltage.

When the output is off, the red LED is on. You will notice that the LED glows longer when the analog input voltage is higher. It provides you with a visual idea what the comparitor is sending back to the parallel port.

You will notice that the analog reading is not always consistent, this is because *computer interrupts* caused by modern PC software (Windows) will vary the number of loops that a program will perform. You will find that you can make the readings more accurate by averaging a few measurements.

Call gadget_measure (v) - This command measures the voltage on the analog input. v is the voltage value on the analog input between 0 and 5 volts. If the GadgetMaster is not connected then the value -1 is returned.

Example

```
'ANALOG MEASUREMENT
    Call gadget_measurement(voltage)
'DISPLAY THE RESULT
    Text5.Text = Format(voltage, "#0.000")
```

Output Enable

The 'output enable' connection provides a simple way to prevent the outputs from functioning. If this terminal, output 13, is connected to ground then all of the outputs are disabled. This feature is intended to provide a means powering off all of the outputs if a switch is closed. This terminal is pulled high to 5 volts and does not need to be connected to make the outputs work.

The High Power, 24 Volt Option

For applications that require a higher power drive the GadgetMaster can be fitted with the '24 volt Option'. This includes a 24 volt 1.25 amp power supply and several modifications to the GadgerMaster board including a large heat sink and fan. Do not attempt to use a large power supply on a normal GadgetMaster (it will overheat and self-destruct). It is fine to run most 12 volt motors at 24 volts but only for short periods of time, the 24 volt option is capable of overdriving small motors and care needs to be taken to avoid overheating motors that may be under-rated for your job. For example, stepper motors can be powered down when they are stationary.

Troubleshooting

Problem: I run gmtest.exe and nothing happens.
Check: Make sure everything is plugged in. Make sure the board is turned on. Make sure the GADGETMASTER is turned on (the yellow LED will light), red LED will pulse. Make sure the GADGETMASTER is connected to the correct printer port. Try the different output port selections; 278,378 and 3BC.

Problem: The board is overheating.
Check: Make sure the device that you are connecting has a resistance of at least 50 ohm or is drawing less that 240 milliamps continuously. Make sure that none of the outputs are connected directly to ground or a ground plane such as a metal object.

Problem: The stepper motors are running very hot.
Check: Stepper motors run hot depending on how long current is left running through the motors. Read the section on Heat Management and modify your software, if possible, to reduce the 'on time' of the motors.

For other questions, comments, problems and help just email us at support@lightmachinery.com or direct from our web site at www.lightmachinery.com

Good Luck with *your* projects

Appendix A

Binary Numbers

Computers think in a very simple way, things are either 'on' or 'off', up or down, yes or no. So computers store all numbers as either 1 or 0. Big number are just lots of 1's and 0's. This kind of number representation is called binary. We normally use base 10 or decimal math. Using base 10 math we count from 0 to 9 and then we add a new digit and make 10. In binary math we add a new digit after we count to 1. Here are a list of numbers from 0 to 32 in each of the number bases

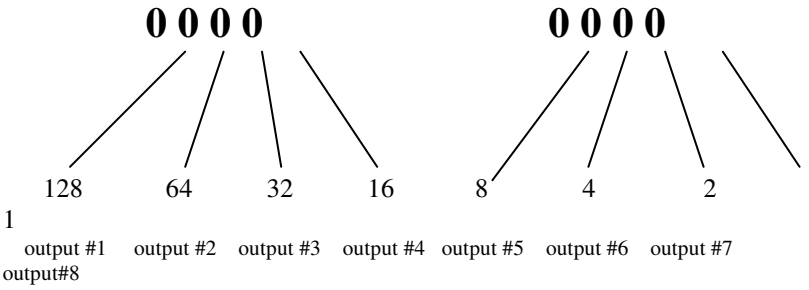
Decimal	Binary	2 Bytes		Powers of 2	Powers of 2 as decimals
0	0	0000	0000	0	0
1	1	0000	0001	2^0	1
2	10	0000	0010	2^1	2
3	11	0000	0011	2^1+2^0	2+1
4	100	0000	0100	2^2	4
5	101	0000	0101	2^2+2^0	4+1
6	110	0000	0110	2^2+2^1	4+2
7	111	0000	0111	$2^2+2^1+2^0$	4+2+1
8	1000	0000	1000	2^3	8
9	1001	0000	1001	2^3+2^0	8+1
10	1010	0000	1010	2^3+2^1	8+2
11	1011	0000	1011	$2^3+2^1+2^0$	8+2+1
12	1100	0000	1100	2^3+2^2	8+4
13	1101	0000	1101	$2^3+2^2+2^0$	8+4+1
14	1110	0000	1110	$2^3+2^2+2^1$	8+4+2
15	1111	0000	1111	$2^3+2^2+2^1+2^0$	8+4+2+1
16	10000	0001	0000	2^4	16
17	10001	0001	0001	2^4+2^0	16+1
18	10010	0001	0010	2^4+2^1	16+2
19	10011	0001	0011	$2^4+2^1+2^0$	16+2+1
20	10100	0001	0100	2^4+2^2	16+4
21	10101	0001	0101	$2^4+2^2+2^0$	16+2+1
22	10110	001	0110	$2^4+2^2+2^1$	16+4+2
23	10111	0001	0111	$2^4+2^2+2^1+2^0$	16+4+2+1
24	11000	0001	1000	2^4+2^3	16+8
25	11001	0001	1001	$2^4+2^3+2^0$	16+8+1
26	11010	0001	1010	$2^4+2^3+2^1$	16+8+1
27	11011	0001	1011	$2^4+2^3+2^1+2^0$	16+8+2+1
28	11100	0001	1100	$2^4+2^3+2^2$	16+8+4
29	11101	0001	1101	$2^4+2^3+2^2+2^0$	16+8+4+1
30	11110	0001	1110	$2^4+2^3+2^2+2^1$	16+8+4+2
31	11111	0001	1111	$2^4+2^3+2^2+2^1+2^0$	16+8+4+2+1
32	100000	0010	0000	2^5	32

Appendix B

Understanding Addresses

1's and 0's are 'what' the computer stores in memory, addresses are 'where' the numbers are stored. Each memory location in the computer has an address. At each address two bytes (8 bits) of information is stored. Each pin on the computer's parallel port is 'connected' to a single bit (or 1 or 0) in the computer's memory.

For example, each of the bits in the two bytes stored at 378 Hex corresponds to the first 8 outputs of the GADGETMASTER .



To send an output to output #1, you simply write a 1000 0000 or 128 to address 378 Hex. To run a motor or turn on lights at outputs #2, #3 and #7 then simply write 98 (=2+32+64) to 378 Hex. In a basic program this would be written as follows;

OUT(&378H) = 98

Once this was executed, boom! action! at outputs #2, #3 and # 7

Appendix C

IBM Printer Port Assignments

Pin	Address	Printer Name	GADGETMASTER Name	Value to Activate	Value when Grounded
1	37A Hex	Strobe	Output 12 or Input Select	2^0 or 1	
2	378 Hex	Data bit 0	Output 8	2^0 or 1	
3	"	Data bit 1	Output 7	2^1 or 2	
4	"	Data bit 2	Output 6	2^2 or 4	
5	"	Data bit 3	Output 5	2^3 or 8	
6	"	Data bit 4	Output 4	2^4 or 16	
7	"	Data bit 5	Output 3	2^5 or 32	
8	"	Data bit 6	Output 2	2^6 or 64	
9	"	Data bit 7	Output 1	2^7 or 128	
10	379 Hex	Acknowledge	Input 1 or input 5		2^6 or 64
11	"	Busy	Input 2 or Input 6		2^7 or 128
12	"	Out of paper	Input 3 or Input 7		2^5 or 32
13	"	Printer on line	Input 4 or Input 8		2^4 or 16
14	37A Hex	Auto line feed	Output 9	2^1 or 2	
15	"	Printer Error	Analog Input		2^3 or 8
16	"	Initialize Printer	Output 10	2^2 or 4	
17	"	Select	Output 11	2^3 or 8	
18-25		Ground	Ground		

IBM Printer Port Pin Assignments

Appendix D

Detailed Programming Notes

For some users programming using the Gadget Commands my not provide enough flexibility to achieve all of their requirements. All of the inputs and outputs can be controlled directly using commands in Basic and the following section provides the details for this kind of programming.

Visual Basic does not have the ability to read or write directly to an address in memory, such as the parallel port address. In other words it does not have the old INP or OUT commands from old fashioned Basic. Fortunately, the INP and OUT commands can be restored by including a dynamic link library (.dll) in the same directory as your VB5 program and adding a small basic module to each program that you write. The files Inpout32.dll and inpout32.bas work very well and can be downloaded from the LightMachinery web site.

Using Basic to track 'inputs'

The inputs are monitored by reading the data at the address of the printer port. A full list of the printer port addresses is given in the appendix. Here is the Visual Basic DOS code used in gmtest.exe to determine which of the inputs are grounded.

```
IF (INP(&H379) AND 64) = 64 THEN     ' 379Hex is the printer status address
        check1.value = 1             'Input 1 is grounded
ELSE check1.value = 0                'Input 1 is not grounded
END IF

IF (INP(&H379) AND 128) = 128 THEN
        check2.value = 1             'Input 2 is grounded
ELSE check2.value = 0                'Input 2 is not grounded
END IF

IF (INP(&H379) AND 32) = 32 THEN
        check3.value = 1             'Input 3 is grounded
ELSE check3.value = 0                'Input 3 is grounded
END IF

IF (INP(&H379) AND 16) = 16 THEN
        check4.value = 1             'Input 4 is grounded
```

```
   ELSE check4.value = 0                    'Input 4 is grounded
   END IF
```

The AND statement compares the 2 bytes stored at memory location 379 Hex to each of the values that we are looking for. 'AND' performs a 'bitwise' comparison which means that it compare each bit stored at 379 Hex with the number we are comparing.

For example;
The statement **INP(&H379) AND 16** compares the value stored at 379 Hex which may be
1011 0100 to 16 which is 0001 0000. The AND statement first compares the first bit in 379 *and* the first bit in 16. If *both* are 1 then a 1 is returned in that position. In this case the result would be a zero in all the positions except the position where both are 1's which is at the last position in the first byte, so the result is 0001 0000 or 16.

The *check1.value* statements simply display the bullet under the inputs heading on the display. If check1.value is set to 1 then the bullet for the first input is shown, if check1.value is set equal to 0 then the bullet is not shown.

then only inputs 5 - 8 can be used, inputs 1 - 4 cannot be read.

Using Basic to program motion

DC motors, solenoids, lights, LED's, buzzers, speakers; attach any of these devices as shown in the connection diagram. To change the state of the outputs simply write or 'output' numbers to the correct address. The first 8 outputs are addressed at 378 Hex, the last 3 outputs are at 37A Hex, as shown in the **IBM Printer Port Pin Assignments** in the appendix. So, to activate a motor connected to output number 1 simply output the number 128 to address 378 Hex .

```
OUT &H378, 128
or
motor_output = 128
OUT &H378, motor_output
```

To activate a motors, lights or whatever connected to outputs 1, 3 ,4, 7 simply output the number 128, 32, 16 & 2 to address 378 Hex.

```
motor_output = 128 + 32 + 16 + 2
```

```
OUT &H378, motor_output
or
motor_output = 178
OUT &H378, motor_output
or simply,
OUT &H378, 178
```

Using Basic to Control Stepper Motors

This is the software for the 'Go Reverse' button the DOS version of Stepper.exe.

```
SUB gobtnrev_Click ()                'This subroutine is activated by
pressing                                    'the 'Go Reverse' button

 SHARED p                            'p is a pointer that will keep track of
                                            'which phase is energized
                                     'p needs to be shared between the
                                            'forward and reverse routines
step_out(1) = 128                    'when p is equal to 1 the program will
                                            'output the number 128
step_out(2) = 64                     'when p is equal to 2 the program will
                                            'output the number 64
step_out(3) = 32                     'when p is equal to 3 the program will
                                            'output the number 32
step_out(4) = 16                     'when p is equal to 4 the program will
                                            'output the number 16


FOR x = 1 TO VAL(steps.text)         'do this loop for every step

  IF p > 1 THEN p = p - 1 ELSE p = 4        ' step pointer backward
        pointer.text = STR$(p)              ' displays pointer value on the
                                                    'screen
        OUT &H378, step_out(p)              'outputs new position
                FOR y = 1 TO VAL(delay.text)      'delay loop
                y = y + 1
                NEXT y
         NEXT x
        OUT &H378, 0                        'turn power off
```

Index