GeckoMotion MANUAL

Shipping package contents: G-Rex G100 motion controller unit, 12VDC wall-plug power supply and GeckoMotion software CD. Not included: USB cable.

Please plug in the wall-plug 12VDC power supply into a 115VAC receptacle and plug the power cord into the G100 power jack. The red 'RUN' indicator on the G100 should light up. This means the G100 firmware is booted and is operating normally.

This installation assumes you are using Windows XP and all the following installation displays are taken during a WinXP install. Other operating systems (MacOS, Linux, Windows 98, Windows XP64) will have different installation sequences. Please see http://www.ftdichip.com/ for USB drives for their FT245BM controller drivers if they are not on the supplied GeckoMotion software CD.

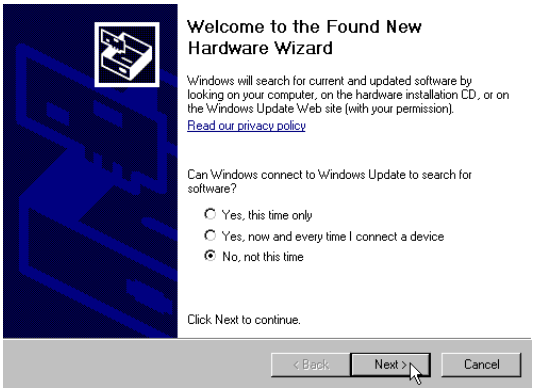
Turn on your PC and place the GeckoMotion software CD into your PC's CDROM drive.

Connect a USB cable to a USB port on your PC and connect the other end of the cable to the G100 USB port.

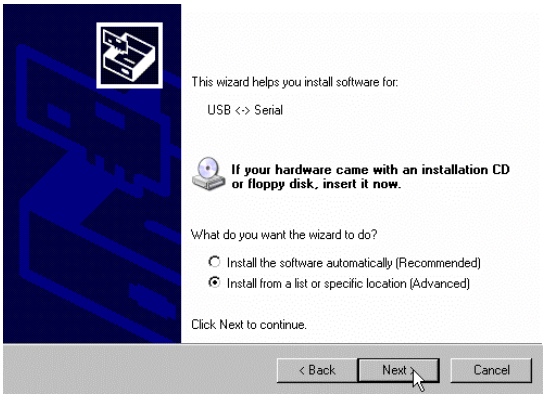
WindowsXP should show the 'Found New Hardware Wizard' when you plug in the USB cable.

The G-Rex USB VCP drivers provided by FTDI that work with the FT245BM USB controller used in the G100 must be installed on your PC.

Select 'No, not this time' and hit 'Next'.



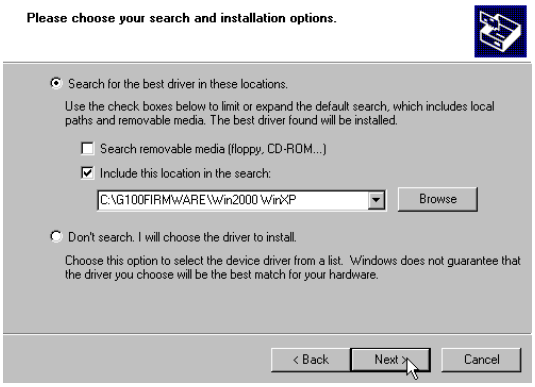
Select 'Install from a list or specific location' and hit 'Next'.



Hit ‘Browse’, then select the CDROM drive you placed the GeckoMotion CD in.

Select your operating system (Win98, Win2000, WinXP or WinXP 64) from the folders on the CD and hit ‘Next’.

This will install the USB drivers used by the G100.

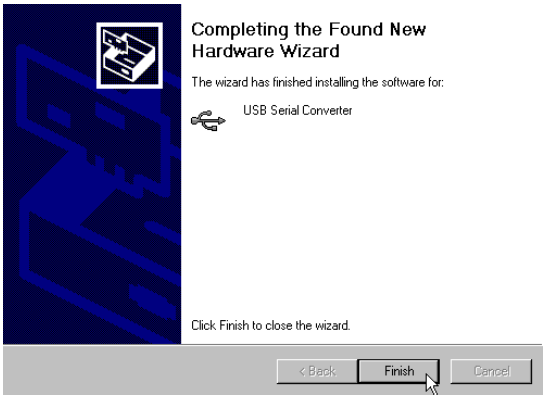


Click ‘Finish’.

This process must be repeated twice; the ‘Found New Hardware Wizard’ will appear a second time in WinXP.

Repeat the same keyboard entries as the first time.

This completes installing the USB drivers.



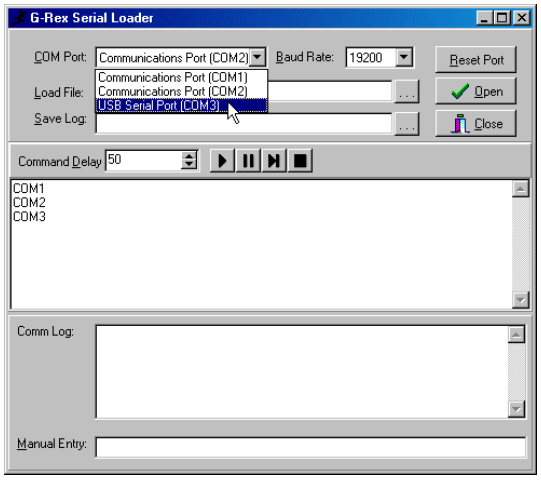
The next step for Windows users is to use ‘Windows Explorer’ to copy **‘GRex Tester.exe’** from the GeckoMotion CD onto their hard drive. Once it is copied over, it is suggested to right-click on **‘GRex Tester.exe’**, select ‘Create Shortcut’ and then drag the created shorcut onto the desktop.

GRex Tester.exe was written and kindly provided by Stephen Wille Padnos. It is simply the very best USB app to communicate with the G-Rex.

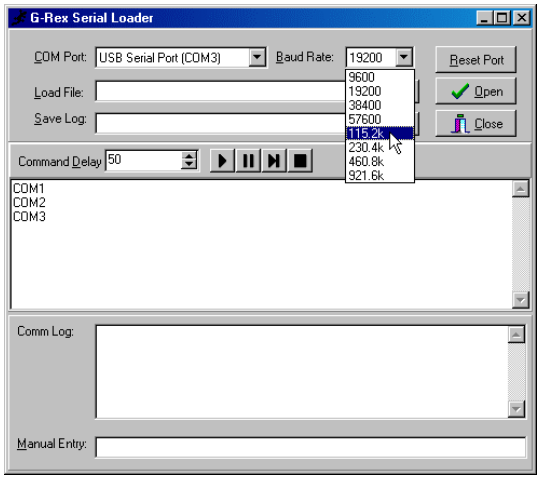
It allows you to write CNC motion control applications using a text editor like Windows Notepad and send the resulting .txt file via a USB port to the G100 for execution. It gives an interactive view of what instructions get sent to the G100 and what information the G100 sends back in response.

Click on the **GRex Tester.exe** after shortcut is dragged onto the desktop. You should see:

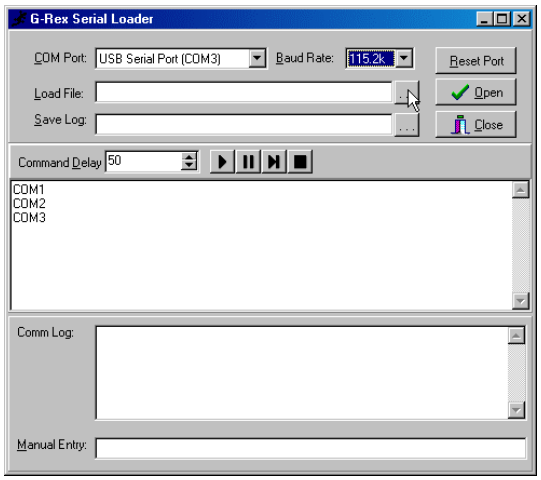
Select the USB comm port that is displayed.



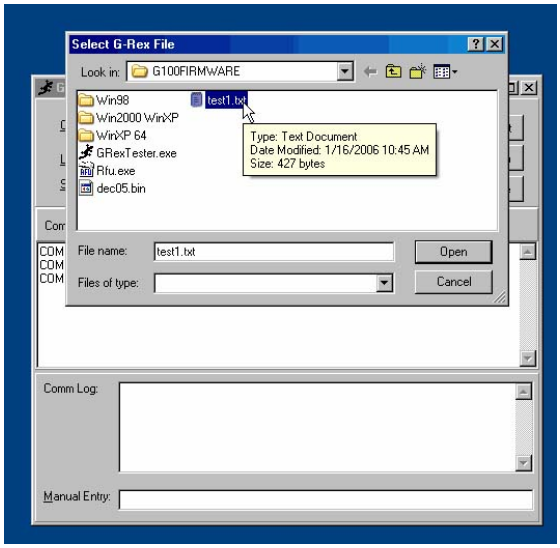
Select a reasonably high baud rate. 115.2K is a good choice.



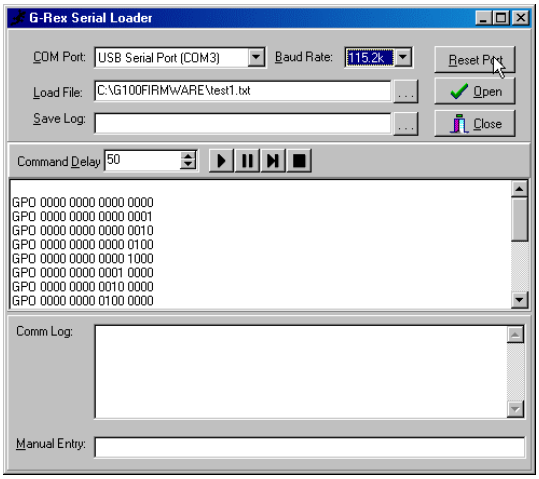
Now select ‘Load File’



Next, select ‘test1.txt’ from the ‘G100 FIRMWARE’ folder on your GeckoMotion software CD.



The following **test1.txt** program should appear Command Window.



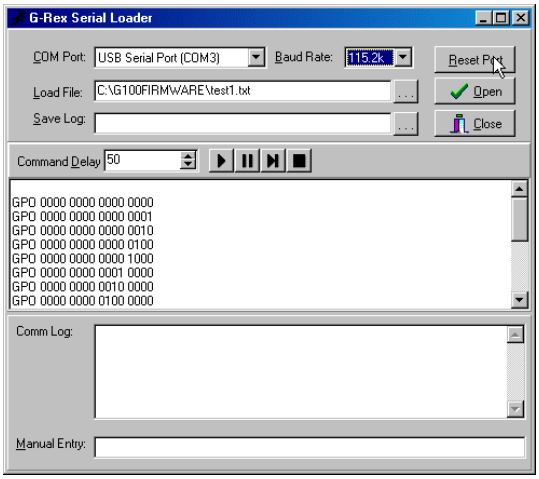
As you will learn later the pneumonic **‘GPO’** means ‘General Purpose Output’ and the format **‘xxxx xxxx xxxx xxxx’** (x = 0 or 1) means ‘1’ for an output being ‘on’ and ‘0’ means an output is ‘off’.
The left-hand most output is ‘16’, the right-hand most output is ‘1’ as marked on the cover of the G100.

The first line when sent to the G100 will turn output ‘1’ on while all the other 15 outputs will be off.

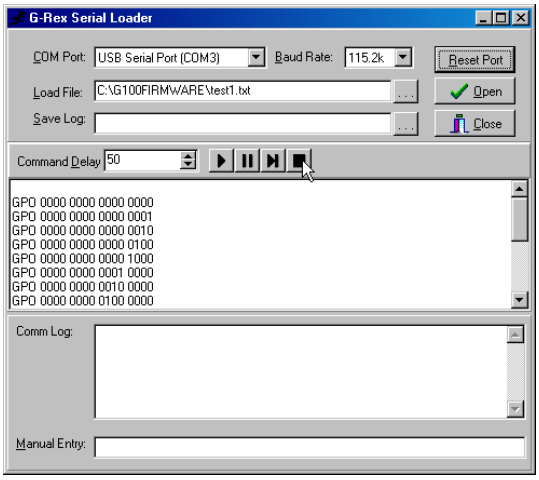
The next line, when sent, will turn output ‘1’ off and turn output ‘2’ on and so on. **Test1.txt** will turn the next output ‘on’ while turning all other outputs ‘off’. It is a very simple first program to test the G100.

Next select ‘Reset Port’. This connects the PC to the G-Rex.

If an ‘Unable to open serial port’ pop-up appears, select ‘Reset Port’ again. This is a harmless bug in the G-Rex Serial Loader.



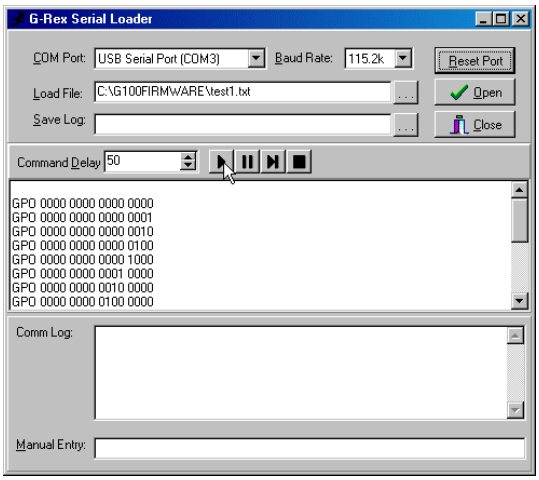
Next, hit the ‘Reset’ button in the G-Rex Serial Loader. This returns the send queue to the first line of the **test1.txt** application program.



Now you can run the **test1.txt** program. This program will turn each general-purpose output ‘on’ in turn while turning the previous (and all other) outputs ‘off’.

Each line will execute in sequence, producing a marquee-like flashing sequence of the G100 red output LED indicators as each output turns ‘on’, then ‘off’.

Kind of cool to look at especially if you repeatedly hit the **>** arrow key. Seeing this means the G100 is responding to instructions from the PC.

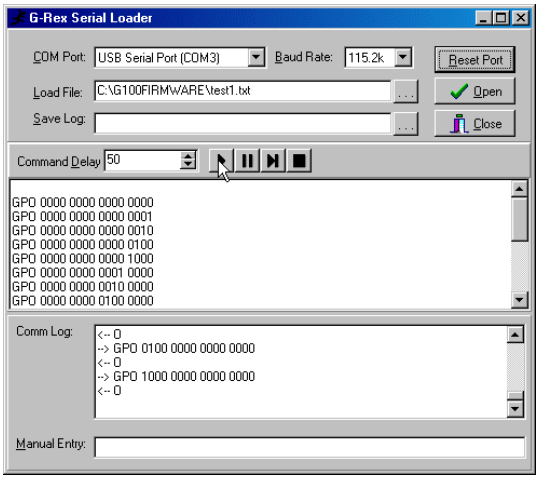


The G100 responds to commands by returning an acknowledgement of the command sent.

This can be view in the **Comm Log:** pane. An issued command is preceded by a **-- >** prefix. A reply is preceded by a **< --** prefix followed by an acknowledgement character or text string.

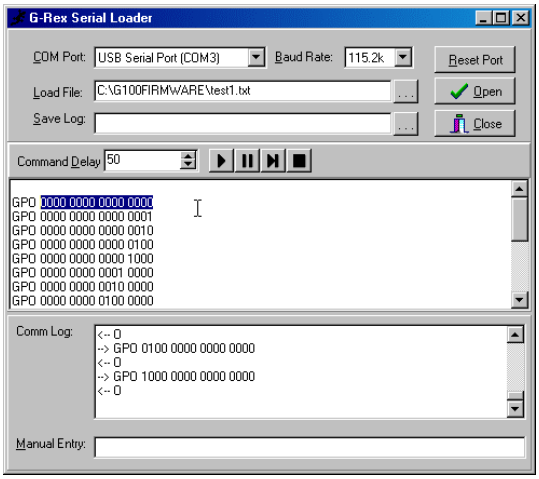
For a **GPO** command (General Purpose Output), the acknowledgement character is ‘**O**’. This indicates the instruction was received as being valid and was executed. A bad instruction would return a ‘**?**’ and would be discarded.

Much more on this later in this manual..



The **G-Rex Serial Loader** is editable. This is an extremely useful feature.

Say you want to change the first command line. Use any of the familiar Notepad editing commands, in this case the highlight command.



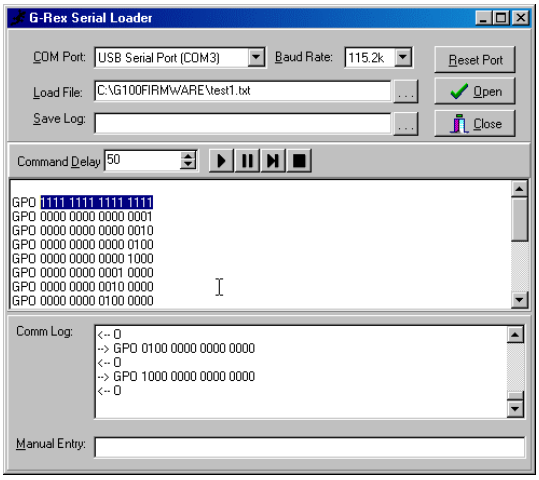
Replace the highlighted line from '0000 0000 0000 0000' with '1111 1111 1111 1111'.

This changes the G-Rex command from all outputs 'off' to all outputs 'on'.

Try it. Sending the changed command line will result in all indicator LEDs turning 'on'.

All other Notepad commands apply. Use CTR + INSERT to paste, CTR + SHIFT to paste into a file.

These features will allow you to fine-tune a program and save it to file.



This concludes the 'Check to see if the G100 is alive on my PC' part of the user manual. If you were able to get the marquee blinking LED lights on the G100, then the USB drivers are installed properly and the G-Rex Serial Loader works as it should. The **test1.txt** program running properly is proof of that.

The next part of the manual will deal with the GeckoMotion command-set in detail.

AXIS SETUP INSTRUCTIONS:

The G-Rex x, y, z, a, b and c axis must be programmed with setup values before any motion commands can be sent. These parameters are normally loaded first and usually are not changed after being loaded.

The G-Rex instructions generally follow an 'instruction > target > value <CR>' format where applicable. An example of this format is **Ax2500<CR>** and is interpreted by the G-Rex as " Set the rate of **A**cceleration for the **x** axis to **2500** increments of motion per time unit squared". The G-Rex will use this value for accelerating and decelerating the x axis in all subsequent motion until it is changed. Normally this is set only once in a program.

FREQUENCY RANGE:

Syntax: **R (axis name) (value) <CR>**
Input: **axis name = x, y, z, a, b, c**
value = 00 thru 07
Example: **Ry07<CR>**
Reply: **R** if valid, **?** if invalid

The G-Rex has 8 frequency ranges for each of the 6 axis. Each frequency range has 32,767 evenly spaced CW speeds and 32,767 evenly spaced CCW speeds. Each frequency range has twice the maximum frequency of the previous range.

The below table shows the step pulse frequency resolution, maximum frequency and suggested drive type for each of the 8 selectable frequency ranges. The **value** is the number used in the instruction to select the desired axis frequency range.

Value	Resolution	Maximum Frequency	Drive
00	128 Hz	4.194304 MHz	Servo
01	64 Hz	2.097152 MHz	Servo
02	32 Hz	1.048576 MHz	125-microstep
03	16 Hz	524.288 kHz	125-microstep
04	8 Hz	262.144 kHz	10-microstep
05	4 Hz	131.072 kHz	10-Microstep
06	2 Hz	65.536 kHz	10-Microstep
07	1 Hz	32.768 kHz	Full, Half-Step

Table 1

PULSE CONTROL MODE:

Syntax: **C (axis name) (value) <CR>**
Input: **axis name = x, y, z, a, b, c**
value = 00 thru 05
Example: **Cz00<CR>**
Reply: **C** if valid, **?** if invalid

Step pulse mode selects the step pulse duty cycle and active edge when the outputs are configured as step and direction. The outputs can also be configured as quadrature outputs or even as general purpose outputs if motion control is not required from the axis. The following table shows the axis step and direction mode for each valid **value** number used in the instruction.

Value	Output	Mode
00	Step/Dir	50% duty cycle, rising edge is active
01	Step/Dir	50% duty cycle, falling edge is active
02	Step/Dir	25% duty cycle
03	Step/Dir	75% duty cycle
04	Quadrature	Step = CH_A, Dir = CH_B
05	GP-outputs	Step and Direction become general purpose outputs

Table 2

ACCELERATION RATE:

Syntax: **A (axis name) (value) <CR>**
Input: **axis name = x, y, z, a, b, c**
value = 1 thru 32767
Example: **Ac300<CR>**
Reply: **A** if valid, **?** if invalid

This instruction sets the axis rate of acceleration. The **value** gets added to the axis velocity 1,024 times a second during acceleration and is subtracted from the axis velocity 1,024 times a second during deceleration.

32,767 different rates of acceleration can be selected but not all of them are practical values. The rate of acceleration is a constant (linear ramp) when used with the moving average filter switched off. If the moving average filter is switched on then the acceleration profile is 'S-shaped'.

The acceleration rate is 'persistent', meaning it will apply to the axis until it is changed.

If the acceleration **value** is **1**, an axis will take 32 seconds to accelerate from zero velocity to the maximum velocity for the axis selected frequency range. If the **value** is **32767**, the axis will take 1/1,024th of a second (976 microseconds) to reach the same speed.

The below table shows rates of acceleration in steps per second squared for each of the G-Rex 8 selectable frequency ranges. Assuming 15,000 radians per second squared as a maximum practical rate of acceleration, the maximum **value** matrix is also shown for 3 popular step motor drive resolutions. You can use **values** higher than suggested if your motor can accelerate very rapidly.

Freq Range	Acceleration Rate	Full-Step	10-uStep	125-uStep
00	(**value**) * 131,072 steps/sec^2	3	36	455
01	(**value**) * 65,536 steps/sec^2	7	73	911
02	(**value**) * 32,768 steps/sec^2	14	146	1,822
03	(**value**) * 16,384 steps/sec^2	29	292	3,645
04	(**value**) * 8,192 steps/sec^2	58	583	7,290
05	(**value**) * 4,096 steps/sec^2	117	1,166	14,580
06	(**value**) * 2,048 steps/sec^2	233	2,333	29,160
07	(**value**) * 1,024 steps/sec^2	466	4,666	32,767

Table 3

MAXIMUM AXIS VELOCITY:

Syntax: **V (axis name) (value) <CR>**
Input: **axis name = x, y, z, a, b, c**
value = 1 thru 32767
Example: **Vx32000<CR>**
Reply: **V** if valid, **?** if invalid

This instruction sets the maximum velocity limit for an axis. The velocity limit equals the **value** times the selected axis frequency range resolution. Refer to Table 1, select the resolution (Hz) for the frequency range being used and divide the desired velocity by that resolution to generate the desired **value** for this instruction.

The velocity limit is 'persistent', meaning it will apply to the axis until it is changed.

Example 1: Assume the desired velocity is 3,000 RPM on a servomotor equipped with a 500-line encoder. So equipped, the servomotor drive requires 2,000 step pulses to take one revolution. Also assume the selected frequency range for this axis is 131.072 kHz (**value** = **05**) and the resolution is 4 Hz.

3,000 RPM is 50 revolutions per second. At 2,000 steps per revolution, 100 kHz is the required step pulse frequency. Divide 100 kHz by 4Hz and you get 25000 as the velocity **value**.

Example 2: Assume you have an 8-microstep drive running a 1.8-degree step motor connected to a 5 turns per inch lead-screw. You want to program the x-axis to have a feed-rate of 20 inches per minute.

8-microsteps per step results in 1,600 steps per motor revolution. Five revolutions per inch results in 8,000 steps per inch. 20 inches per minute is 160,000 steps per minute or 2,666.66 steps per second. Assuming the same frequency range selection as for Example 1, divide 2,666.66 by 4Hz and you get 666.66. Round it to 667 and use it for the velocity **value** in this instruction. The actual feed-rate will be 20.01 inches per minute due to rounding of the **value**.

MOVING AVERAGE CONTROL:

Syntax: **M (axis name) (value) <CR>**
Input: **axis name = x, y, z, a, b, c**
value = 1 thru 3
Example: **Mb2<CR>**
Reply: **M** if valid, **?** if invalid

This instruction controls the axis moving average filter. The moving average filter is used by the G-Rex to implement constant contouring and to generate 'S-shaped' acceleration profiles. This instruction is an exception to normally being used only during the setup; it is used extensively during constant contouring. See the Constant Contouring section of this manual for a detailed description of its use.

M(axis name)1 Constant contouring is on.

M(axis name)2 The moving average filter is off. By default, constant contouring is off as well.

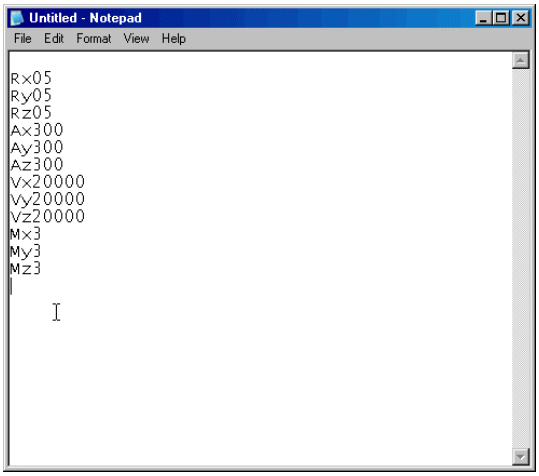
M(axis name)3 Constant contouring is off.

USING NOTEPAD TO WRITE A PROGRAM:

Assume you want to initialize three axis identically. Open Notepad and hit 'Enter' for the first line. This will insure the G-Rex will not ignore the first line of text.

After you are finished, save your work as a .txt file.

Open the G-Rex Serial Loader as before and open your saved file.



Press the 'Play' button on the G-Rex Serial Loader.

The 'Com Log' window begins to scroll showing the communication between the G-Rex and your PC.

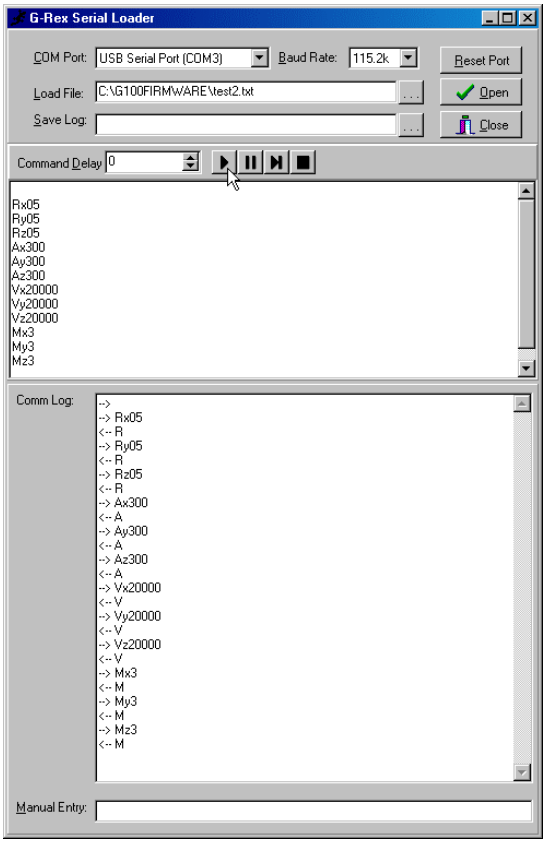
→ Indicates the first <CR> sent which syncs the G-Rex.

→ Rx05 indicates the frequency range instruction has been sent to set the x-axis to the 131.072 kHz range.

← R Indicates the G-Rex accepted and executed the instruction. If there had been a problem with the syntax or value range of the instruction, the G-Rex would have sent ← ? and ignored the instruction.

This proceeds automatically the same way for all the rest of the lines in the program 'test2.txt' until the last instruction is acknowledged. The communication then ends.

Pressing the 'Play' button again will repeat this program from the beginning again. Double-clicking the left mouse button while the cursor is in the 'Comm Log' window will clear the window contents.



AXIS MOTION INSTRUCTIONS:

AXIS COORDINATE DATA MOTION:

Syntax:	[**(axis name) (value)] [repeat up to 5 more times] <CR>**
Input:	**axis name = x, y, z, a, b, c**
	value = 0 thru 99999999 +, -
Example:	**x12345678y-987z+87654321<CR>**
Reply:	first **(axis name)** if valid, **?** if invalid

This is a fairly complex instruction when multiple axis are involved. Let's first cover a single axis implementation of this instruction. The attributes for a single axis will carry over to the more complex multiple axis implementations. A single axis instruction is:

(axis name) (value) <CR>

The **value** is any integer between and including 0 thru 99999999. This integer represents increments of motion, or steps. This integer may be preceded by the optional '+' or '-' operator.

Example: **x1234<CR>** moves the x-axis to the absolute coordinate location 1234. The motion may be CW or CCW depending on the current location of this axis. The distance moved will be the difference between the current location of the axis and the coordinate location 1234.

Example: **x+1234<CR>** moves the x-axis 1234 increments of motion CW relative to its current location.

Example: **x-1234<CR>** moves the x-axis 1234 increments of motion CCW relative to its current location.

The axis rate of acceleration and velocity depends on the previously set axis-associated values for **R (axis name) (value) <CR>**, **A (axis name) (value) <CR>** and **V (axis name) (value) <CR>**.

The **M (axis name) (value) <CR>** modifies motion by passing the axis velocity word stream (1,024 16-bit words per second) through a 128-sample moving average filter. This filter smoothes abrupt changes in velocity by averaging them.

M(axis name)1 Constant contouring is on. A new move (if one is pending) begins immediately after the axis velocity decelerates to zero at the **input** to the moving average filter.

M(axis name)2 Motion not modified. Moving average filter is disabled.

M(axis name)3 Constant contouring is off. The actual acceleration profile is the first integral of the **value** set by the axis associated acceleration instruction. This causes an abrupt change in velocity (acceleration **value** greater than maximum velocity **value**) to become a ramp; a ramped velocity becomes an S-shape profile. A new move (if one is pending) begins immediately after the move in progress axis velocity decelerates to zero on the **output** of the moving average filter.

When the axis begins a move, it sends "**(axis name) <CR>**" back to the PC's USB port to let the PC know new data for that axis may be sent. If new data is sent, then the new move begins as soon as the move in progress completes. If the PC sends no new data, the axis stops when the move in progress completes.

MULTIPLE AXIS MOTION

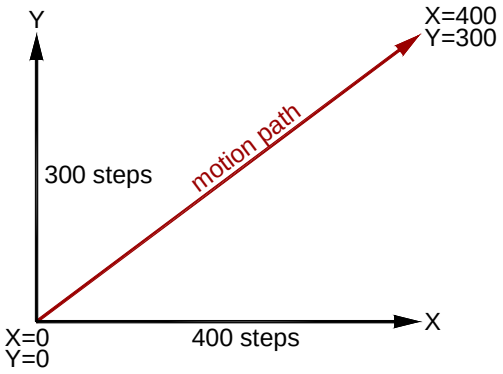
Multiple axis moves have 2 to 6 **(axis name)** and **(value)** listed in the instruction line.

The G-Rex handles all multi-axis moves as vectors. All the named axis in a multi-axis move begin moving and stop moving at the same time. If the axis control a mechanism that moves in 2 or 3 dimensions (X,Y or X,Y,Z), then the path traced for the move will be a straight line connecting the start coordinate location to the destination coordinates. Figure 1 illustrates a 2-dimensional move.

Example: Let's assume you have written the following program:

Ax30
Vx1000
Ay3
Vy100
x+400y+300

The first axis named becomes the master, all other mentioned axis in the instruction line become slaves. The master axis **A**, **V** and **M** instruction settings will apply for the pending move; the slave axis settings will be ignored.



The G-Rex calculates the length of the motion path (500 steps long) and uses that dimension in a 'virtual axis' as the distance to travel. It uses the master axis (X in this case) acceleration and max velocity settings for this 'virtual axis'. The resulting 'virtual output velocity' as it accelerates, runs and then decelerates to a stop is multiplied the X and Y axis vector component values (X = 0.8, Y = 0.6). These are then output to the actual X and Y step and direction outputs.

The result is the X axis runs at velocity of 800 and the Y axis runs at a velocity of 600. The velocity along the motion path is 1000 though, the value specified in **Vx1000** in the above program. In fact the velocity along the motion path will always be the master axis velocity setting regardless of the direction of the motion path.

Now let's say you had written **y+300x+400** instead of **x+400y+300**. The motion path would have been exactly the same but because Y is the master axis now, its settings would apply. The velocity along the motion path would have been 10 times slower because **Vy100** applies instead of **Vx1000**. This is a simple way of having a federate velocity or a rapid velocity selected by the order in which the axis are mentioned in an instruction line.

The above example was for 2-dimensional (X,Y) motion. The same principles apply for 3-dimensional moves (X,Y,Z) and for higher dimensional moves as well. A virtual axis is formed and virtual vector velocity is generated by the motion algorithm. The following is the math for a 6-D move:

MPL (Motion path length) = SQRT (x^2 + y^2 + z^2 + a^2 + b^2 + c^2)

Xvel = Vector Vel * x / MPL
Yvel = Vector Vel * y / MPL
Zvel = Vector Vel * z / MPL
Avel = Vector Vel * a / MPL
Bvel = Vector Vel * b / MPL
Cvel = Vector Vel * c / MPL

POINT-TO-POINT MOTION

Example: If you had written:

x+400
y+300

Then the result would be a point-to-point move instead of a vector move. Both axis would start at the same time but finish at different times. Both axis would reach the same coordinates (X = 400, Y = 300) as the vector move did but the path would not be a straight line.

Point-to-point moves have a velocity advantage over a vector move depending on the angle of the motion path. A 45-degree motion path gives the maximum advantage, 1.414 times greater velocity than a constant vector. Point-to-point is the choice if you want to get from here to there the fastest and you don't care about the path taken.

MULTIPLE INDEPENDENT VECTOR MOVES

The G-Rex can run multiple independent vector motion tasks. This means you can have three independent X,Y axis performing three independent tasks or 2 independent X,Y,Z axis doing two unrelated tasks or any other combination so long as the total number of axis do not exceed six.

x+0y+10000
z+7071a+7071
b+0c+20000
x+10000y+0
z+7071a-7071
b+20000c+0
x+0y-10000
z-7071a-7071
b+0c-20000
x-10000y+0
z-7071a+7071
b-20000c+0

The above program would trace out three squares, the first aligned at 0-degrees, the second square rotated 45-degrees and the third square like the first one except twice as large.

The G-Rex will reply to the G-Rex Serial Loader with **← (master axis name)** as each X,Y axis pair is ready for new data. The G-Rex Serial Loader doesn't use this reply; it only displays the reply in the Comm Log window. A different interface would use the reply to send the next requested instruction line:

→ **x+0y+10000**	Send move data to X,Y pair
← **x**	X,Y replies asking for next move data
→ **x+10000y+0**	Send next (pending) move data to X,Y pair
→ **z+7071a+7071**	Send move data to Z,A pair
← **z**	Z,A replies asking for next move data
→ **z+7071a-7071**	Send next (pending) move data to Z,A pair

etc.

This way data queuing is done by the PC. Data is sent on request, each axis pair in this example has pending data and the motion on all 3 independent tasks is continuous. Using the G-Rex Serial Loader will result in pauses between motion paths because data is sent in the order it was written.

CONTINUOUS PATH MOTION (M instruction)

The G-Rex is designed to generate continuous path motion in up to 6 dimensions. A continuous path in 2-D or 3-D might be a series of concatenated (connected end-to-end) line segments. Oftentimes these line segments are used to approximate an arbitrarily shaped curve. If the line segments are numerous enough, a piecewise linear approximation of a curve can be very good.

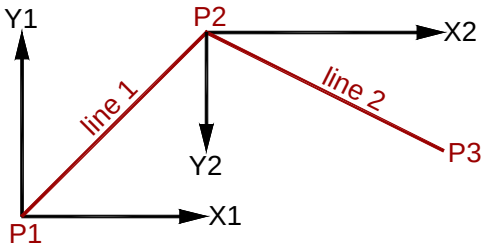
At other times the concatenated line segments do not form an approximation but rather are an exact representation of the desired shape. An example might be a right angle.

The G-Rex has two modes of continuous path motion: **velocity-honoring** or **path-honoring**.

Path-honoring means the exact path as programmed will be followed. If a right-angle forming line segments are programmed, a right-angle path will result. If a 100-sided polygon forming line segments are programmed, a 100-sided polygon path will result.

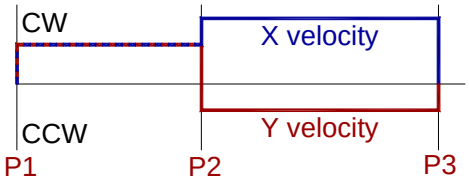
Velocity-honoring means the master axis velocity setting will be maintained along the path formed by the concatenated line segments.

Both methods have problems and advantages.

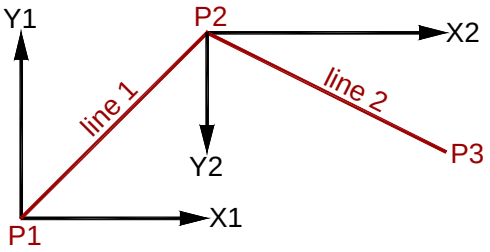


The graph to the right satisfies both requirements (at very low speeds). The red line 1 and line 2 are two line segments along a motion path. The path motion starts at P1, changes direction at P2 and concludes at P3.

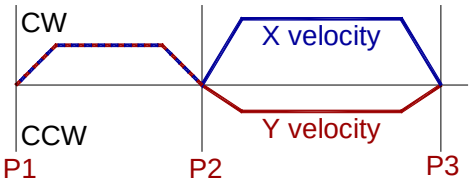
If it is desired to follow the path and maintain a constant velocity along the path, then the XY axis velocities must be as shown for the X velocity and Y velocity graphs.



Note that velocity must change instantaneously at P1, P2 and P3 to meet both requirements. Torque is the first derivative of velocity if a non-zero system inertia is present. The first derivative of a step change is an infinite-impulse function; meaning infinite torque is required from the motor. A motor 'fudges' this requirement at very low speeds by having some elasticity. At higher speeds this does not help and the motor must stall or the drive must error-fault.

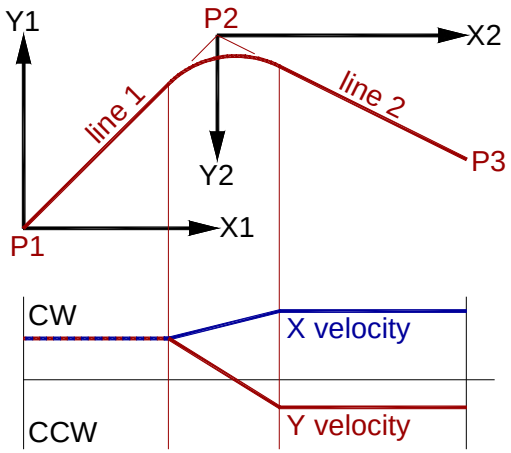


Path-honoring at higher speeds must include an acceleration component to the path motion. The graph to the right shows the same motion path as above but now the XY motors accelerate at the beginning of a line segment, run at speed, then decelerate to a stop at the end of a segment. This process repeats for every line segment in the path.



Use **M(axis name)2<CR>** for the master axis, also set **A (axis name) (value) <CR>** for the master axis along with **V (axis name) (value) <CR>**. Those will set the rate of acceleration and continuous path motion velocity between the acceleration and deceleration intervals. The generated path will be exactly as programmed but the motion may be very jerky (due to accel/decel) if there are numerous short line segments.

Velocity-honoring is another name for 'constant contouring'. Instead of following the programmed path exactly, the programmed master axis velocity is maintained. This requires placing a curved path that blends from line 1 into line 2 as shown in the graph below. It is required because any change in direction causes acceleration on both axis. Only a curved path keeps the axis acceleration rate finite.



Note that X velocity and Y velocity both begin and end changing simultaneously. The path that results is actually parabolic and is asymptotic to line 1 and line 2.

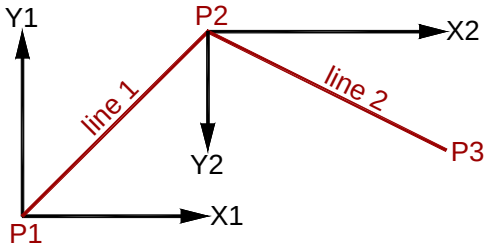
The velocity along the motion path stays constant and is equal to the master axis velocity instruction.

Set the master axis to **M(axis name)1<CR>** to cause the G-Rex to use the **velocity-honoring** mode. Set the master axis **A (axis name) (value) <CR>** to 32,767 (infinite acceleration) to use this mode; the moving average filter will convert the step-change in velocity to the acceleration ramp shown ramp in the graph above.

The final use of the 'M' instruction is for forming an 'S-shaped' acceleration, deceleration ramp.

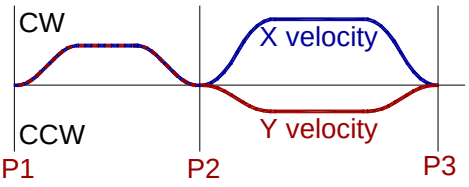
This uses a finite acceleration instruction value **A (axis name) (value) <CR>** in conjunction with **M(axis name)3<CR>**. The moving average filter gives the first integral of the acceleration slope. If the slope is a step function, it generates a slope, if the acceleration slope is a ramp, it delivers an 'S-shaped' profile. An 'S-shaped' acceleration profile keeps the second derivative of velocity finite. It is often called the 'jerk factor'.

What this does is round the start and end of the acceleration / deceleration slope profile. This minimizes load 'ringing' at the beginning and end of acceleration and deceleration.



The effect is fairly subtle but nonetheless can be noticed. The main effect is mechanism motion just sounds smoother.

It cannot be used in **velocity-honoring** mode and time to speed is less optimal than **M(axis name)2<CR>**. It is most effective with large inertial loads.



The graph to the right illustrates the behavior of this instruction on the rounding of X velocity and Y velocity profiles. The motion path is of the **path-honoring** type.

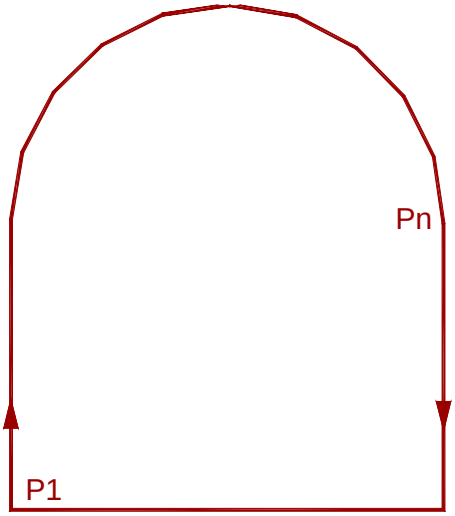
To summarize:

Use **M(axis name)1<CR>** for constant-contouring.
Use **M(axis name)2<CR>** for exact path following.
Use **M(axis name)3<CR>** for the smoothest possible exact path following at the expense of slightly longer execution time.

USING THE (M instruction) IN A PROGRAM

Velocity-honoring and **Path-honoring** modes can be mixed as required in a program.

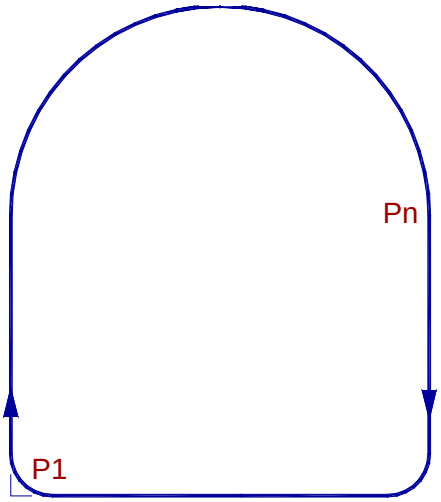
The graph to the right shows a typical example where mixing modes may be very useful. The graph is half of a 20-sided polygon grafted on top of a rectangle. It is a 'mouse-hole' shape.



If the master axis uses **M(axis name)2<CR>** then the path will be exactly as shown. However the actual path motion will have a start-stop jerkiness particularly along the polygon portion of the path.

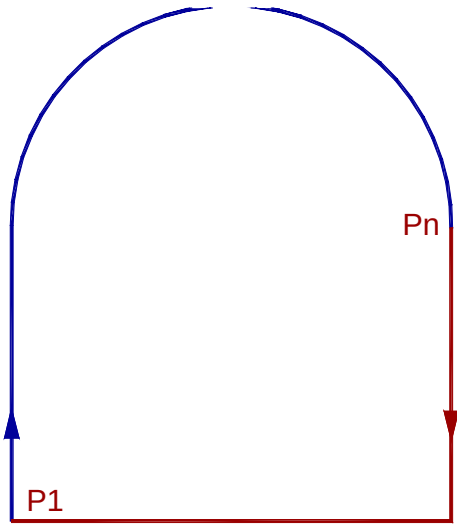
The starting point is P1 as is the ending point of this graph. Pn is the end segment of the polygon.

The path will be absolutely accurate which may not be a good thing. The polygon portion may be better if it approximated a semi-circle, as may have been the intent.



If the master used **M(axis name)1<CR>** throughout, the polygon at the top of the graph would have approximated a semi-circle very well but it would have been at the expense of having the right-angles rounded at the bottom. This would not be good.

The solution is to mix **velocity-honoring** and **path-honoring** modes as required in a program.



Preceding P1 have the master axis have **M(axis name)1<CR>**. After P2 change the master axis to use **M(axis name)1<CR>**.

The resulting path will be **velocity-honoring** over the blue portion of the path and switch to **path-honoring** over the red portion of the path. The curved parts will be curved, the sharp cornered parts will have sharp corners.

The **M** instruction can be switched in and out as many times as necessary in a program. Just precede each line segment series with the **M** instruction as required.

HOME:

Syntax: **H (value) (axis name) <CR>**
Input: **axis name = x, y, z, a, b, c**
value = + and **-**
Example: **H+x<CR>**
Reply: **(axis name)** when at HOME position, **?** if invalid

This instruction causes the named axis to execute a 'HOME' process. The named axis will move CW if the **value** is '+' and CCW if the **value** is '-'.

A home switch (SPST) must be present on the mechanism driven by the named axis and must be physically located where the mechanism is expected to stop. Normally this is at one travel extent of the mechanism. The home switch must connect to the named axis LIM input and GND. The switch must allow for axis over-travel without sustaining damage.

The named axis will use its **V** and **A** settings for velocity and acceleration towards the selected direction HOME switch.

The named axis will decelerate to a stop at the **A** setting rate when it closes the HOME limit switch. The axis will then reverse direction and run off of the limit switch at a low speed. The axis will stop the instant the limit switch opens. The named axis position register will be reset to zero and the axis will notify the PC with the **(axis name)** reply to indicate it is at the HOME location.

This instruction is homes only the named axis. Sending **H+x<CR>** homes only the X-axis. The other axis can be executing other instructions while the named axis is executing the **H** instruction. If it is desired to home all 6 axis, send:

H+x The X-axis will home CW
H-y The Y-axis will home CCW
H+z The Z-axis will home CW
H+a The A-axis will home CW
H-b The B-axis will home CCW
H-c The C-axis will home CCW

Do not home unused axis or axis without a HOME switch connected to the named axis LIM input! Doing so will cause those axis to execute an instruction that will never complete. A G-Rex hardware reset will then be required to clear those instructions.

SPEED CONTROL MODE:

Syntax: **S (axis name) (value) <CR>**
Input: **axis name = x, y, z, a, b, c**
value = + or **– 0** thru **32767**
Example: **Sz+15000<CR>**
Reply: **(axis name)** if valid, **?** if invalid

This instruction causes the named axis to emulate a speed control. The axis will use its **M** and **A** settings to accelerate the axis to the velocity **value** in the instruction. A plus sign preceding the **value** results in CW rotation, a minus sign preceding the **value** results in CCW rotation. The axis will run at the set velocity and direction until the axis is sent a new instruction.

The speed and direction can be changed by sending a new **S** instruction. The named axis will accelerate / decelerate to the new direction and/or velocity **value**.

GENERAL PURPOSE OUTPUTS:

Syntax: **GPO (value) <CR>**
Input: **value** = **nnnn nnnn nnnn nnnn** n = 0 or 1 to change all
Input: **value** = **output name** and **H** or **L** to change a single output
Example 1: **GPO 1101 0010 1001 0001<CR>**
Example 2: **GPO 14L<CR>**
Reply: **O** if valid, **?** if invalid

This instruction operates on the G-Rex's 16 general-purpose outputs. All outputs can be changed simultaneously or a single named output can be changed individually.

To change all outputs:

The **value** = **nnnn nnnn nnnn nnnn** reads as outputs 16 15 14 13 then 12 11 10 9 then 8 7 6 5 and 4 3 2 1. A '1' means the output is 'on' while a '0' means the output is 'off'. The outputs are open-collector transistors so 'on' is zero volts on the output. The indicator LED next to the output terminal will be lit when an output is 'on'.

There is no read-back of the outputs, so keep a copy if it's necessary to know the state of the outputs.

Example: **GPO 1101 0010 1001 0001** means outputs 16 15 13 10 8 5 and 1 are to be turned 'on' while all the other outputs are to be turned 'off'.

To change a named output:

The output names are 1 through 16. The **value** must contain the output name followed by either 'H' or 'L'. 'H' means the output is 'on' while 'L' means the output is 'off'.

Example: **GPO 14L** means output 14 is to be turned 'off'.

Example: **GPO 3H** means output 3 is to be turned 'on'.

GENERAL PURPOSE INPUTS:

Syntax: **GPI <CR>**
Reply: **I nnnn nnnn nnnn nnnn** n = 0 or 1 if valid, **?** if invalid
Example: **I 1101 0010 1001 0001**

This instruction reads the G-Rex's 16 general-purpose inputs. The returned **nnnn nnnn nnnn nnnn** reads as inputs numbered 16 15 14 13 then 12 11 10 9 then 8 7 6 5 and 4 3 2 1. A '1' means the input is 'off' while a '0' means the input is 'on'. The inputs are active low so 'on' means zero volts on the input. The indicator LED next to the input terminal will be lit when an input is 'on'.

ENCODER READ COMMANDS:

Syntax: **E (axis name) <CR>**
Input: **axis name = x, y, z, a, b, c**
Reply: **E (axis name) (sign) (value)**
sign = +, -
value = 0 to 2147483648, leading zeros suppressed
Example: **Ey+1234567890**

This instruction reads the contents of the named axis quadrature encoder counter and returns the value. The G-Rex quadrature decoders are the “times 4” type which means the counts per revolution is 4 times the encoder line count.

Syntax: **E* <CR>**
Reply: **Ex (sign) (value)**
Ey (sign) (value)
Ez (sign) (value)
Ea (sign) (value)
Eb (sign) (value)
Ec (sign) (value)
sign = +, -
value = 0 to 2147483648, leading zeros suppressed

Example: **Ex+0**
Ey-4567
Ez+12000000
Ea-20000
Eb+314159265
Ec+0

This instruction reads the contents of all 6 quadrature encoder counters and returns the values. It is a short cut around having to send individual instructions for each of the 6 axis.

ENCODER WRITE COMMANDS:

Syntax: **E (axis name) (sign) (value) <CR>**
Input: **axis name = x, y, z, a, b, c**
sign = +, -
value = 0 to 2147483648
Example: **Ex+1234567890<CR>**
Reply: **E (axis name) (sign) (value)** this echoes-back the command

This instruction loads the quadrature encoder counter with value. The encoder will increment or decrement from this value.

AXIS POSITION READ COMMANDS:

Syntax: **P (axis name) <CR>**
Input: **axis name = x, y, z, a, b, c**
Example: **Px<CR>**
Reply: **P (axis name) (value)**

This instruction reads back the named axis position. The returned integer is dimensioned in increments of motion (step pulses).

Syntax: **P* <CR>**
Reply: **Px (value)**
Py (value)
Pz (value)
Pa (value)
Pb (value)
Pc (value)
value = 0 to 2147483648, leading zeros suppressed

Example: **Px0**
Py4567
Pz12000000
Pa20000
Pb314159265
Pc0

This instruction reads the contents of all 6 axis position registers and returns the values. It is a short cut around having to send individual instructions for each of the 6 axis.

AUTONOMOUS OPERATION:

Syntax: **L <CR>**
(your program)
\<CR>
Reply: **L** if valid, **?** if invalid

This instruction loads the application program to the G-Rex battery-backed RAM memory. The USB cable can be disconnected after loading and the program can then be run without a PC attached.

The load program instruction **L** stores every instruction line by line that follows it until the end of file instruction **** is encountered. Loading a new program over-writes the previous program (if any) in the G-Rex battery-backed RAM memory.

The stored application program is started by momentarily shorting General Purpose Input 1 to ground (GND). The program will begin with executing the first instruction after the **L** instruction and finish when it executes the instruction preceding the **** instruction.

Example: Open G-Rex Serial Loader, select the USB comm-port and click the Reset Port button.

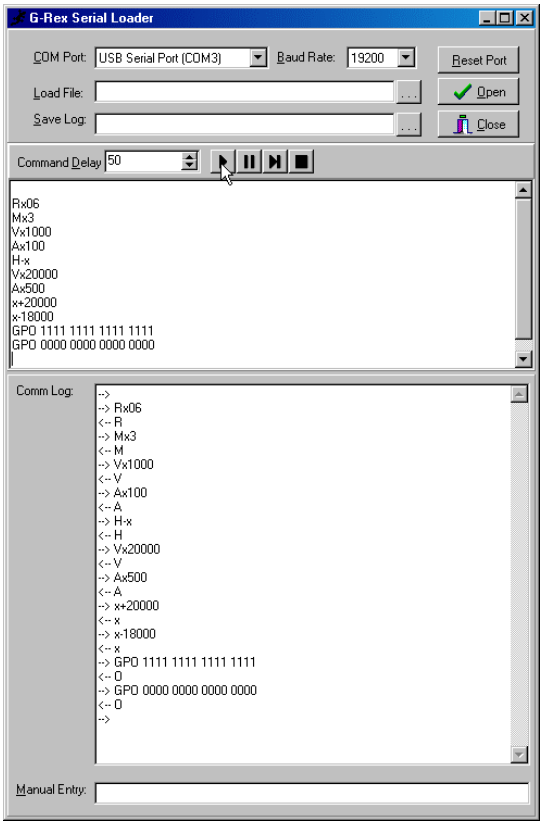
Write a program in the File Window. In this example the axis setup instructions **Rx06** and **Mx3** command frequency range 6 and an 'S-shaped' acceleration profile.

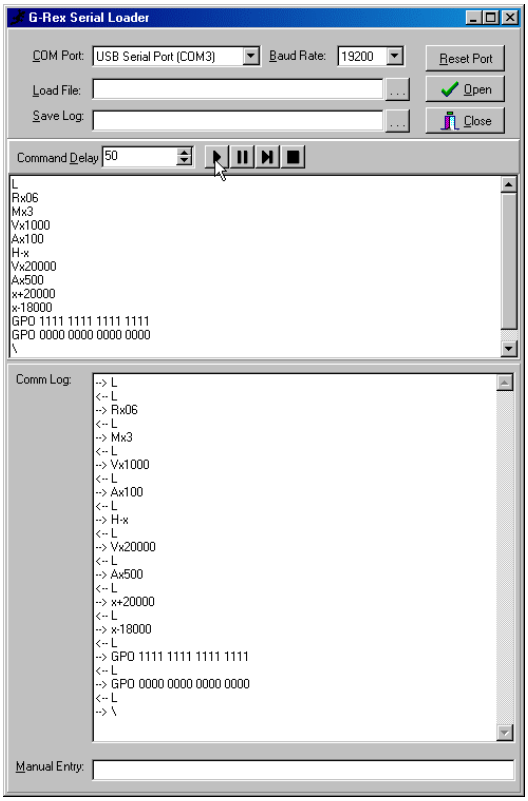
Vx1000 commands a velocity of 2,000 step pulses per second or 1 revolution per second when a 10-microstep drive is used.

H-x commands the X-axis to home to CCW limit switch.

The X-axis velocity is then changed to 20 revolutions per second and the axis is commanded to move 20,000 increments of motion (10 motor revolutions) CW. This is followed by a move of 9 revolutions CCW. Finally all 16 General-Purpose Outputs are turned on, then off.

This concludes the sample program. You can test it by clicking the Play button. The Comm Log window will scroll down as each instruction is sent to the G-Rex and executed.





Once the program is tested, precede the first instruction line with **L<CR>** and put **\<CR>** after the last instruction.

Click the Play button to load the program to the G-Rex. The Comm Log window will scroll down as each instruction is loaded by the G-Rex but this time the motor will not move.

At this point you can highlight and copy the program to Windows clipboard (Ctrl + C) if you wish to save it and then close the G-Rex Serial Loader.

The USB cable can now be disconnected and if need be, power to the G-Rex can be disconnected. The program will remain stored even if the G-Rex is un-powered.

To start the stored program, momentarily short the General-Purpose Input 1 to GND. The program will begin to execute the same as before but there will be no pauses between instructions caused by the USB latency time. These pauses were very short so you may not notice a difference.

The G-Rex can remain connected to the PC and the G-Rex Serial Loader can remain open. If the **L<CR>** and **\<CR>** instruction lines are removed, the G-Rex will respond to PC commands as before. The same program or another one can be run from the PC without affecting the stored program in any way.