GM215 MANUAL

STEP MOTOR DRIVE

1540 SOUTH GRAND AVE - SANTA ANA - CA 92705
PH: (714) 832-8874 FAX: (714) 832-8082
WWW.GECKODRIVE.COM

Table of Contents

The GM215 is a 7A, 80VDC step motor drive with an integrated motion controller. It can be used as a step and direction input motor drive or as motion control enabled drive. The operating mode and mode related functions are set via the 10-position slide switch on side of the drive. The GM215 has 3 opto-isolated inputs and 3 opto-isolated outputs. The function of the input/outputs is also determined by the selected operating mode. The features of these operating modes are described below.

STEP MOTOR DRIVE FEATURES

The GM215 can operate as a conventional STEP and DIRECTION input step motor drive.

MICROSTEPPING: The GM215 has a 10 microstep native resolution; each full step angle of the motor is divided into 10 equally spaced microsteps so a 200 step per revolution motor has 2000 stopping locations per revolution.

STEP PULSE MULTIPLIER: The GM215 synthesizes 10 microstep pulses for every full-step pulse and 5 microstep pulses for half-step pulse. The GM215 acts like a full or half-step drive but motor has the smoothness of a 10-microstep drive.

SUB-MICROSTEPPING: For the 10 microstep resolution, each input step pulse is divided into 32 sub-microsteps resulting in a motor smoothness equal to a 320 microstep drive.

MORPHING: The GM215 morphs from sine-cosine motor currents at low speeds to square-wave currents at high speeds. This technique extracts the maximum possible power from the motor at higher speeds. Morphing begins at 240RPM and ends at 360RPM.

MID-BAND RESONANCE COMPENSATION: The GM215 uses active second-order damping to completely suppress a step motor's tendency to resonate and stall at medium speeds (300RPM – 900RPM). This results in stable motor operation in this region.

LOW SPEED RESONANCE COMPENSATION: Low speed resonances are motor vibrations at speeds below 120 RPM caused by motor non-linearity. The PROFILE and ADJUST trimpots settings nulls these vibrations.

PROTECTION: The GM215 is protected against motor to ground and motor to motor output short-circuits. It is also protected against reversed power supply polarity and power supply over-voltage. The internal fuse blows on polarity reversal and over-voltage.

AUTOMATIC STANDBY CURRENT: If enabled, the motor phase current is reduced to 70% of the set value and the GM215 motor switching topology is changed to a low heating recirculating mode. This happens 1 second after the last step pulse is received.

MOTION CONTROLLER FEATURES

The GM215 motion controller core is a 16-bit MCU, FPGA, Flash ROM and an RS485 interface transceiver. The motion controller executes ASCII format commands sent from an external PC or from its own non-volatile memory.

'ON THE FLY' MOTION CONTROL: Acceleration, velocity and destination can be changed even while the motor is in motion. The new values apply immediately.

MULTIPLE AXIS MOTION: Up to 4 GM215 drives can communicate with each other to execute coordinated multi-axis motion.

RUN FROM STORED PROGRAM: The GM215 can run stored motion programs from its non-volatile Flash memory without a computer connected to it. The memory can store over 65,000 coordinates or program lines.

POWERFUL COMMAND-SET: The command-set includes 'if-then-else' conditional branching, looping and macros (subroutines). It also includes canned high-speed HOME routines, JOG routines, SPEED control routines and more.

ANALOG INPUTS: The GM215 has on-board 3 trimpots to allow setting acceleration, velocity and destination instead of using digital settings.

GENERAL PURPOSE I/O: The GM215 has three opto-isolated user defined inputs and three opto-isolated user defined outputs.

RS-485 INTERFACE: The GM215 has RS485 interfaces available. The default baud rate is 115,200.

BOOT-LOADER: The user can update the GM215 firmware using the built in boot-loader function.

CN1 CONNECTOR ASSIGNMENTS

Terminal	Name	Function
1	GND	DC Power supply (-)
2	VDC+	DC Power supply (+)
3	A	Motor winding A
4	/A	Motor winding A
5	B	Motor winding B
6	/B	Motor winding B

GND (TERMINAL 1)
Connect the power supply '-' to this terminal. This connection must be hard-wired to the power supply.

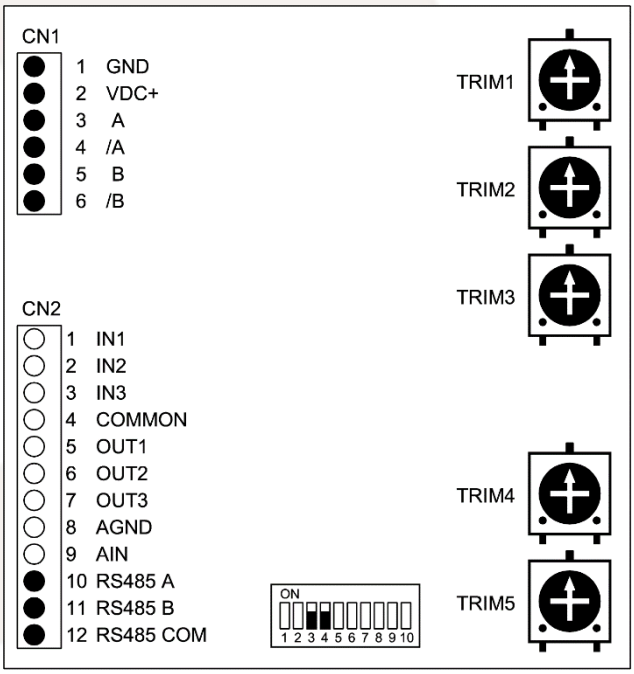
VDC+ (TERMINAL 2)
Connect the power supply '+' to this terminal. The power supply voltage must be between 18 VDC and 80 VDC and this connection must be hard-wired to the power supply. Do not use a switch, relay contact or any other device in series with this wired connection.

A (TERMINAL 3)
/A (TERMINAL 4)
Connect one motor phase winding to these terminals.

B (TERMINAL 5)
/B (TERMINAL 6)
Connect the other motor phase winding to these terminals.

TRIMPOT ADJUSTMENTS
TRIM1 and TRIM2 are used to maximize motor smoothness at speeds below 50 RPM. The Digital Self-Test feature can useful in making these adjustments. While the motor is turning, adjust TRIM1 for minimum motor vibration.

The rest of this user's manual is divided into two sections. Use the **GM215 STEP MOTOR DRIVE MANUAL** if the GM215 is used as a conventional step motor drive. Use the **GM215 MOTION CONTROLLER MANUAL** if the GM215 motion controller is used.

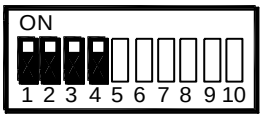


QUICK SWITCH SETTING GUIDE

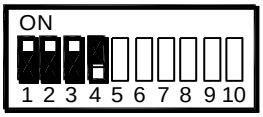
For SW5 to SW10, see NOTE.
For more details, see Motor Drive Mode and Motion Controller Mode Switch Settings.

= SW is "ON"

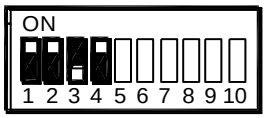
= SW is "OFF"



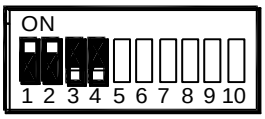
STEP/DIR full-step (Motor Drive Mode)



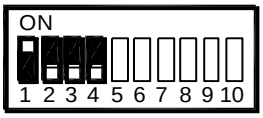
STEP/DIR half-step (Motor Drive Mode)



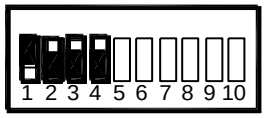
STEP/DIR 10-microstep (Motor Drive Mode)



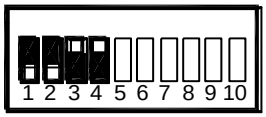
Digital SELF-TEST



Analog SELF-TEST



RUN Program-X axis (Motion Controller Mode)
Set SW3-SW4 for axis type



EDIT Program-X axis (Motion Controller Mode)
Set SW3-SW4 for axis type

NOTE:
1) SW5 is for motor size setting:
SW5 = ON for NEMA-23 or smaller
SW5 = OFF for NEMA-34 or larger
2) SW6-SW10 are for current setting

For current setting, see Fig 2 on page 7; for axis setting, see Fig 4 on page 11.

Figure 1: **QUICK SWITCH SETTING**

GM215 STEP MOTOR DRIVE MANUAL

This manual covers the GM215 when it's used as a conventional STP/DIR input step motor drive. Go to the **GM215 MOTION CONTROLLER MANUAL** if the GM215 is used as a motion controller and motor drive.

MOTOR CURRENT SETTING

= SW is 'ON'

= SW is 'OFF'

= SW is not used to set current

WARNING! THESE SWITCHES MUST BE SET CORRECTLY BEFORE RUNNING THE GM215 WITH A MOTOR WHEN IN MOTOR DRIVE MODE OR SELF-TEST MODE
SET CURRENT BEFORE TURNING POWER ON

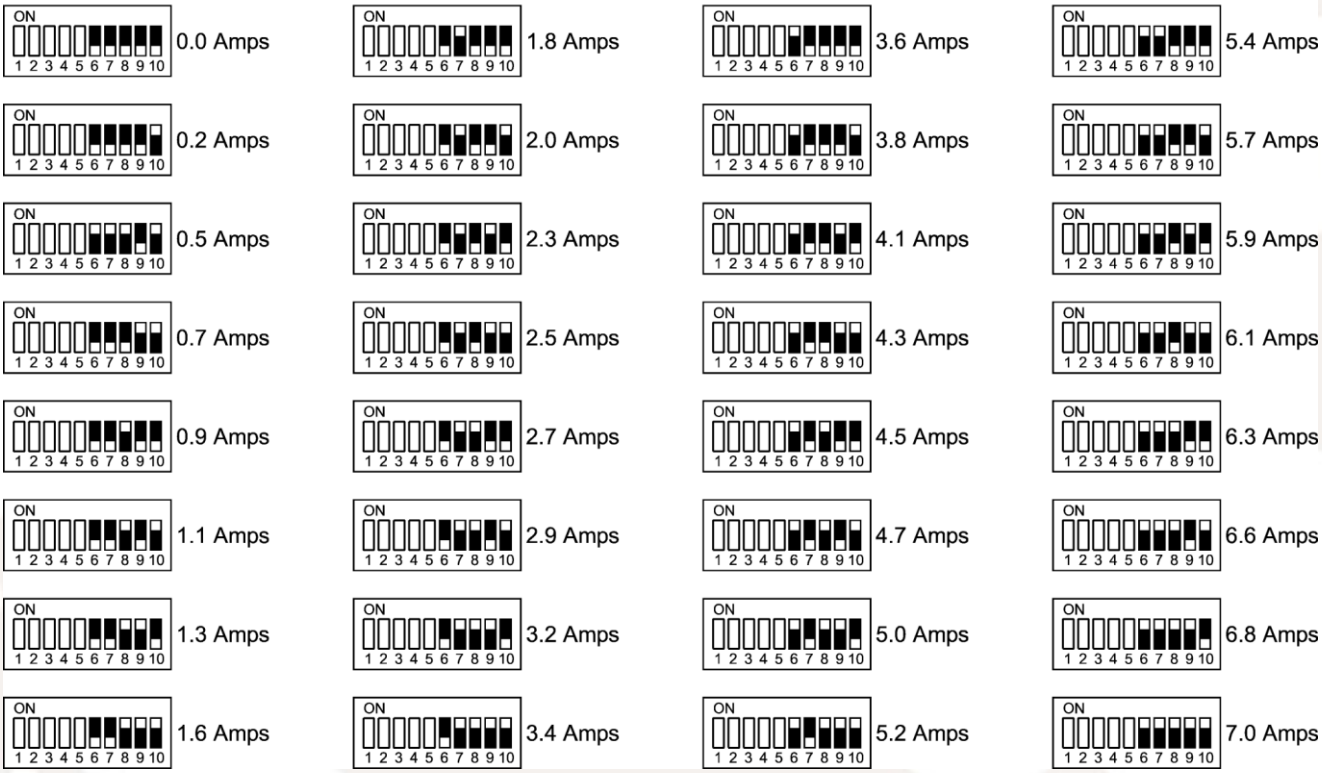


Figure 2**: MOTOR CURRENT SETTING**

SWITCH SETTINGS AND CONNECTOR WIRING

CAUTION:
Perform the following steps with the power supply turned 'OFF' until a step says it is OK to turn the power supply 'ON'. Do not change the current set switches after setup while the motor is powered.

The following switch settings and their function only applies to the Motor Drive Mode. The same switches have completely different functions in the Motion Controller Mode.

STEP 1: SELECT MOTOR DRIVE MODE
SW1 'ON' Select step motor drive mode operation. In this mode the GM215 acts like a STEP / DIRECTION motor drive and the internal motion controller isn't used.

STEP 2: SELECT MOTOR CURRENT
Use the following chart to set the GM215 to the motor's phase current rating. If a motor has a rated current that isn't listed in the table below, set the current to the first setting that is greater than the motor's rated current.

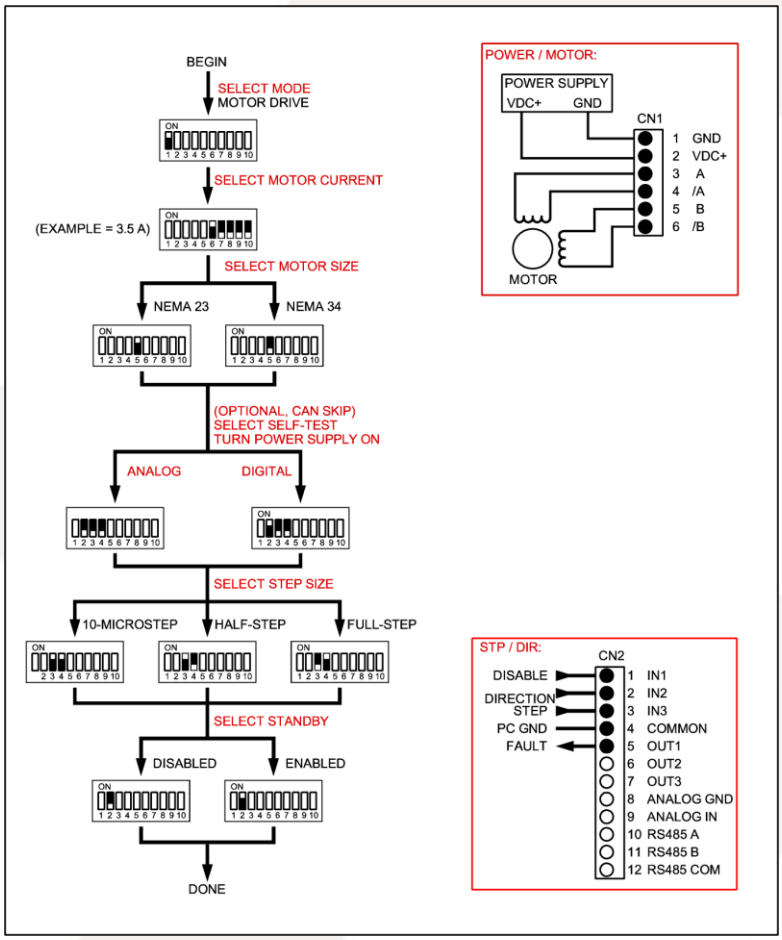


Figure 3: **MOTOR DRIVE MODE SETUP**

STEP 3: SELECT MOTOR SIZE
Switch 5 must be set to the motor frame size.
SW5 'ON' The motor is a NEMA-23 size or smaller.
SW5 'OFF' The motor is a NEMA-34 size or larger.

STEP 4: SELECT SELF-TEST

This step is optional and it can be skipped. Self-Test can also be used after setup is completed or at any other time when it is necessary to verify the GM215 is working correctly. It can be useful when debugging a controller to GM215 interface. When finished with Self-Test, restore switches SW2, SW3, and SW4 to their previous settings to exit the Self-Test routine.

Two Self-Test routines available, Digital and Analog:

DIGITAL SELF-TEST

SW2 'ON'
SW3 'OFF'
SW4 'OFF'
In this setting, a 1.8 degree motor continuously turns 5 revolution clockwise and counter-clockwise at low, medium, and high speed.

ANALOG SELF-TEST

SW2 'OFF'
SW3 'OFF'
SW4 'OFF'
In this setting, the motor's rate of acceleration is set by TRIM3, its velocity is set by TRIM5. The motor will move 5 revolutions in either direction before reversing.

STEP 5: SELECT STEP SIZE

The GM215 can move in full-step, half-step or 10-microstep increments.

FULL-STEP

SW3 'ON'
SW4 'ON'
Every step pulse will move a 1.8 degree motor 1.8 degrees. The motor will move with 10-microstep smoothness which greatly limits low speed vibration.

HALF-STEP

SW3 'ON'
SW4 'OFF'
Every step pulse will move a 1.8 degree motor 0.9 degrees. The motor will move with 10-microstep smoothness to limit low speed vibration.

10-MICROSTEP

SW3 'OFF'
SW4 'ON'
Every step pulse will move a 1.8 degree motor 0.18 degrees. The motor will move with 320-microstep smoothness which makes the motor's motion continuous instead of step-wise even at very low speeds (less than 1 full-step per second).

STEP 6: SELECT STANDBY CURRENT REDUCTION

Standby Current Reduction is a method of reducing motor heating while a motor is idle and not moving. This is accomplished reducing motor current to 70% of the set value if this setting is enabled. The only adverse effect is the motor's holding torque is reduced to about 75% of its nominal value. If the motor has a back-driving load such as holding up a weight against gravity, this selection may not be advisable.
When enabled, the current reduction goes into effect 1 second after the last step pulse is received. The motor stays in this state until a new step pulse sent, at which time full current is restored very quickly.

SW2 'ON' enables current standby. The current is reduced while a motor is idle.
SW2 'OFF' disables current standby. An idle motor will stay at full current.

THIS COMPLETES THE SWITCH SETUP

CN2 CONNECTOR ASSIGNMENTS

Connector CN2 terminals 6 through 12 are unused in the Step Motor Drive mode. Terminals 1, 2 and 3 are common cathode opto-isolated inputs with 200 Ohm current limit resistors. The inputs work with 2.5V, 3.3V and 5V logic levels. Logic 1 input current is 2 mA. The output is an open collector opto-isolator that has a 10 mA current sink rating (see **Appendix G** for more details).

Terminal	Function	I/O
1	DISABLE	INPUT
2	DIRECTION	INPUT
3	STEP	INPUT
4	COMMON	EXTERNAL GROUND
5	FAULT	OUTPUT

DISABLE (TERMINAL 1)

The motor drive is ENABLED when this input is unused or has a logic 0 applied. The motor drive is DISABLED when this input has a logic 1 applied. When DISABLED, the motor current goes to zero, there is no switching activity on the motor outputs and the motor free-wheels (detent torque). The motor position is restored if no step pulses have been sent while disabled.

DIRECTION (TERMINAL 2)

The state of this input determines the direction a motor will move when a step pulse is received. The DIRECTION logic level must be stable 250ns before the active edge of the step pulse.

STEP (TERMINAL 3)

A positive edge on this input (logic 0 to logic 1) causes the motor to move one increment of motion. The minimum logic 1 time is 1 microsecond and the minimum logic 0 time is 3 microseconds.

COMMON (TERMINAL 4)

This is the ground terminal for the DISABLE, DIRECTION and STEP inputs and the FAULT output. It must go to the ground terminal of the controller that generates these inputs.

FAULT (TERMINAL 5)

This output goes to a logic 1 when the drive goes into protective shutdown because of over-voltage, over-current or over-temperature. Once the cause for the FAULT is corrected, the FAULT output can be cleared by cycling the power supply or the DISABLE input.

When finish switch setup and wiring, it is OK to turn on power supply.

END OF GM215 STEP MOTOR DRIVE MANUAL

GM215 MOTION CONTROLLER MANUAL

This manual covers the GM215 when it's used as a motion controller and motor drive. Go to the **GM215 STEP MOTOR DRIVE MANUAL** if the GM215 is used as a conventional STP/DIR input step motor drive.

SWITCH SETTINGS AND CONNECTOR WIRING

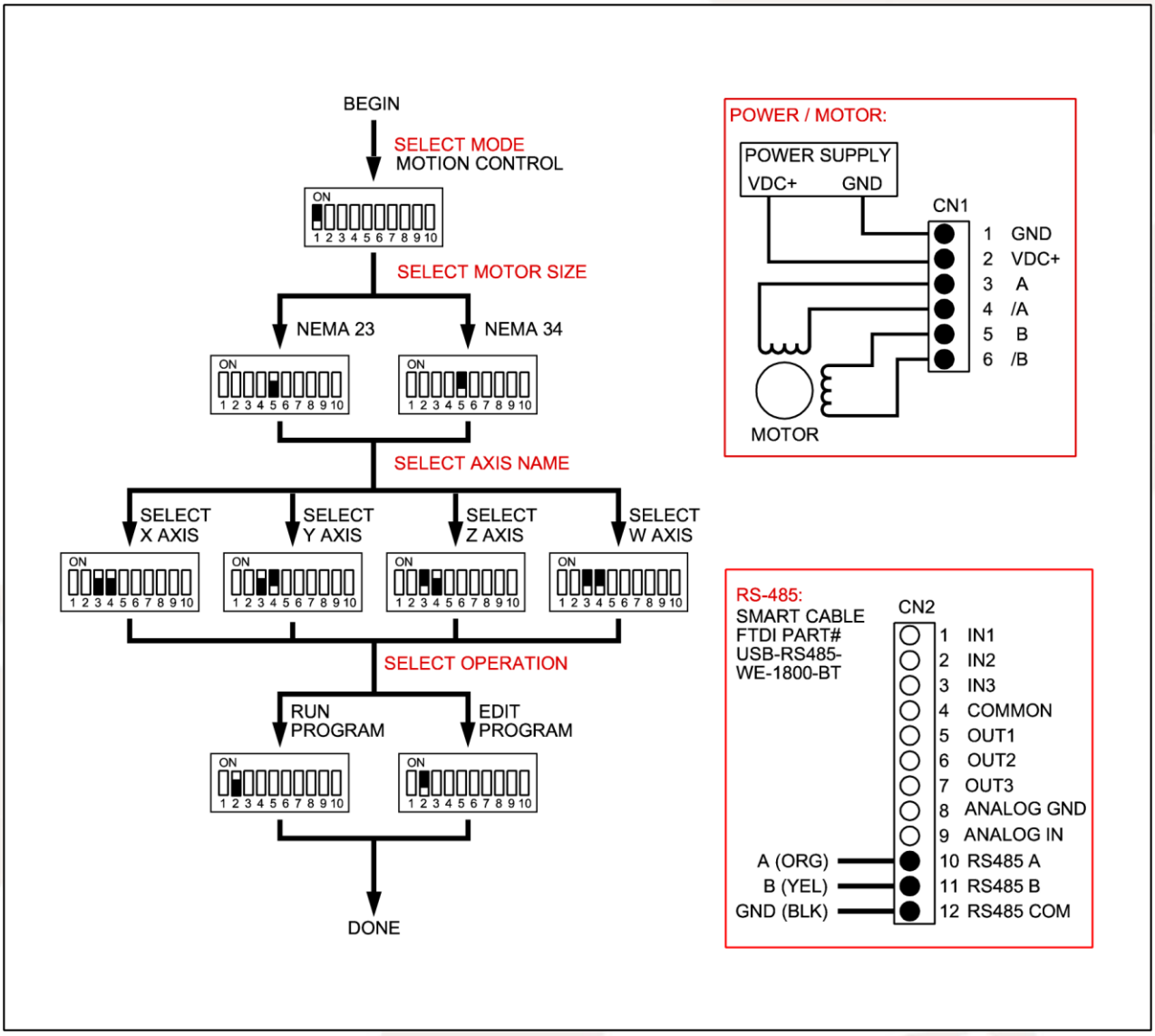


Figure 4: **MOTION CONTROLLER MODE SETUP**

GM215 SETUP FOR MOTION CONTROLLER MODE

CAUTION: The following switch settings and their function only applies to the Motion Controller Mode. The same switches have completely different functions in the Step Motor Drive mode. Turn the power supply 'OFF' and take care to set the switches to their correct settings when changing the operating mode.

STEP 1: SELECT MOTOR DRIVE MODE SW1 'OFF' Select motion controller mode operation.

STEP 2: SELECT MOTOR SIZE SW5 'ON' The motor is a NEMA-23 size or smaller.
SW5 'OFF' The motor is a NEMA-34 size or larger.

STEP 3: SELECT AXIS NAME
The axis name tells a GM215 which commands and communications apply to it and which it can ignore.

X AXIS:	**Y AXIS:**	**Z AXIS:**	**W AXIS:**
SW3 'ON' + SW4 'ON'	SW3 'ON' + SW4 'OFF'	SW3 'OFF' + SW4 'ON'	SW3 'OFF' + SW4 'OFF'

STEP 4: SELECT OPERATION The Motion Controller has 2 operating modes: EDIT and RUN.

EDIT MODE SW2 'OFF'

The EDIT mode is used to edit an existing user program or to write a new user program. This mode requires a serial connection (RS-485) between PC and GM215. The code can be flash to GM215 on board ROM and be stored permanently.

Edit mode is used as a debug tool. Each line of code can be single stepped. **For high performance motion, please use RUN mode.**

RUN MODE SW2 'ON'

In RUN mode, one or more GM215 can execute a user program stored in Flash ROM without requiring a computer connection.

NOTE: A RESET/START SWITCH OR BUTTON IS REQUIRED TO USE RUN MODE. See figure 7 – page 16 for more detail.

CN2 CONNECTOR ASSIGNMENTS

Terminals 1, 2 and 3 are opto-isolator LED anode inputs in series with 200 Ohm current limit resistors (see **Appendix G** for more details). The LED cathodes go to the COMMON terminal. The inputs work with 2.5V, 3.3V and 5V logic levels. Minimum logic 1 input current is 2 mA. Terminals 5, 6 and 7 are three opto-isolated collector outputs that have a 10 mA current sink rating and their emitters go to the COMMON terminal (see **Appendix G** for more details).

Terminal	Name	Function
1	IN1	INPUT
2	IN2	INPUT
3	IN3	INPUT
4	COMMON	
5	OUT1	OUTPUT
6	OUT2	OUTPUT
7	OUT3	OUTPUT
8	ANALOG GND	ANALOG GND
9	ANALOG IN	ANALOG INPUT
10	RS485 A	SERIAL PORT
11	RS485 B	SERIAL PORT
12	RS485 COM	SERIAL PORT GROUND

IN 1 (TERMINAL 1)
This is a low-speed opto-isolated general purpose input. A logic '1' signal turns the input 'ON'. The signal ground is the COMMON terminal (term 4). At RUN mode its dedicated function is **RESET**. (See figure 7)

IN 2 (TERMINAL 2)
This is a high-speed opto-isolated general purpose input. A logic '1' signal turns the input 'ON'. The signal ground is the COMMON terminal (term 4). Its dedicated function is the home switch input if the **HOME** command is used. (See figure 6)

IN 3 (TERMINAL 3)
This is a high-speed opto-isolated general purpose input. A logic '1' signal turns the input 'ON'. The signal ground is the COMMON terminal (term 4). It has no dedicated function.

COMMON (TERMINAL 4)
This terminal is the signal ground for IN 1, IN 2, IN 3, OUT 1, OUT 2 and OUT 3 opto-isolators.

OUTPUT 1 (TERMINAL 5) GENERAL PURPOSE OUTPUT
OUTPUT 2 (TERMINAL 6) GENERAL PURPOSE OUTPUT
OUTPUT 3 (TERMINAL 7) GENERAL PURPOSE OUTPUT

ANALOG INPUT (TERMINAL 8 & 9) Used with command IF_THEN_ELSE and COMPARE.

For example **IF x VIN IS > GOTO label1.**

Notes: This analog input is designed for a voltage feedback from external system, and it is a source of flow control command. It is NOT designed for using this analog value to control motor speed.
Analog speed control can be done by using on board trimpots or replacing trimpots with analog voltage source (0-3.3v).

RS485 (TERMINALS 10, 11, 12)
These terminals are used by the RS-485 transceiver. The RS-485 COM terminal connects to the GM215 circuit ground through a 33 Ohm resistor used to limit ground loop currents. All GM215s using the RS-485 serial interface must share a common power supply ground. The A input has a 3K pull-up resistor to 3.3V while the B input has a pull-down 3K resistor to ground. These terminals are NOT opto-isolated.

MULTI-AXIS POINT-TO-POINT MOTION

Point-to-point multi-axis motion is used when it does not matter what path is taken to a coordinate location. This is perfectly acceptable in many application because point-to-point motion takes the shortest possible time moving to a new location.

Each axis can use its own programmed accelerate, velocity and destination values for motion. The path taken to the destination probably won't be in a straight line because each axis' acceleration, velocity and destination can be different so each axis will take a different amount of time moving to the new location. All of the axis must finish moving before they can move to the next programmed location.

MULTI-AXIS VECTOR MOTION

Unlike point-to-point motion, a vector motion path is along a straight line from the last coordinate location (previous x, y, and z) connecting to the next coordinate (next x, y, and z). The rate of acceleration and velocity is independent of the vector direction. Oftentimes the coordinates are short line segments that, when concatenated, linearly approximate arbitrary 2D or 3D curves.

Vector motion requires all GM215s begin and finish executing each vector segment at precisely the same time and this requires that all GM215's microprocessor clocks be phase locked to the master x-axis clock. Besides communication tasks, the master x-axis GM215 uses the RS-485 serial bus to transmit a synchronizing signal to all slave axis GM215s.

For reason of simplicity, each axis is programmed with exactly the same user program. Each GM215 calculates the vector length from the coordinates stored in the program but only extracts the vector component that matches that motion controller's designated name. The vector component is then processed by a motion control algorithm and the results are sent to the motor drive section of the GM215. Each GM215 then drives its attached axis motor and all the axis motor's movements combine to reconstitute the vector as a 2D or 3D motion in the motor-driven mechanism.

I/O SCHEMATIC

The opto-isolator LED series current limit resistors are 200 Ohms on the IN1, IN2, and IN3 inputs. The input current is 17mA at +5A, 9mA at +3.3V and 2mA at +2.5V. The minimum operating input current for all inputs is 1.6mA (see **Appendix G** for more details). IN1 is a low-speed opto-isolator having a 50uS propagation delay. IN2 and IN3 are high-speed opto-isolators and have a 1uS propagation delay. OUT1, OUT2, and OUT3 are low-speed opto-isolators having a 50uS propagation delay (see **Appendix G** for more details).

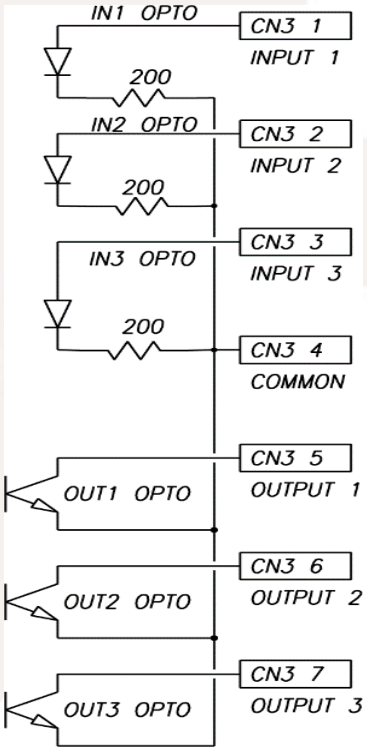


Figure 5: **I/O SCHEMATIC**

HOME/LIMIT SWITCH WIRING

When using **command HOME**, **a hardware connection is a must**. Default home switch input is IN2.
When using command SPEED CONTROL, it is possible to use two limit switches. Default limit switches input are IN2 and IN3.

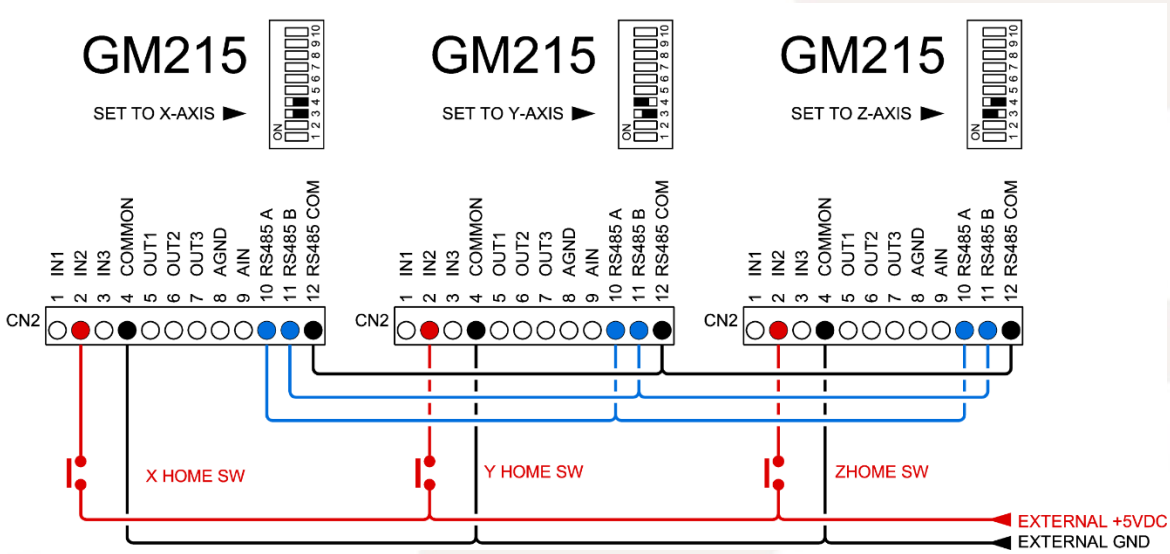


Figure 6: **How to wire the HOME switches**

RUN MODE START/RESET WIRING

After flash the on board ROM, GM215 can run at RUN MODE.
RUN mode requires a START/RESET button or switch to control the progress. Without it, master and slave drives in a multiple drives system cannot be synchronized well. Default start/reset input is IN1. Once applied a 5v, all drives will be reset and hold the reset status until 5v is released.

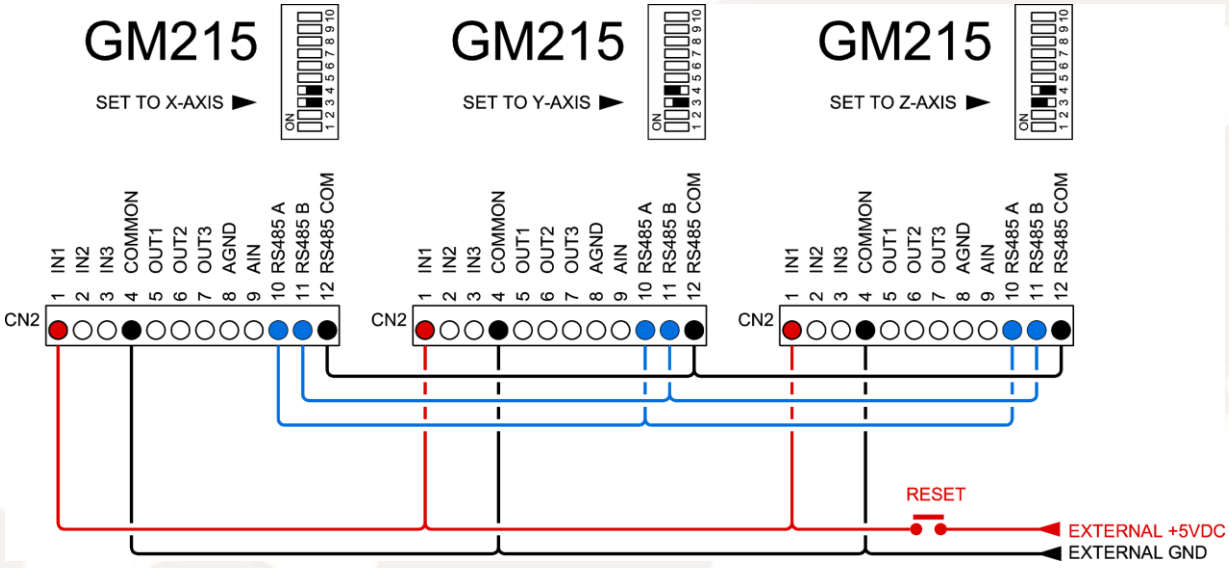


Figure 7: **How to wire the RESET switch for RUN mode**

COMMAND SET

The GM215 works as a multiple-axis motion controller when 2 or more drives are connected via its RS-485 interface. When used this way, **one GM215 must be named as the X drive (SW3 = ON, SW4 = ON) to make it the master drive**. Each of other drives must be set to a unique axis name (Y, Z, or W) using SW3 and SW4.

Once connected and named, the other slaved drives automatically synchronize themselves to the X-drive's microprocessor clock via sync pulse sent over the RS-485 interface. The drives communicate with each other and behave as if they were a single, 4-axis motion controller. All the drive's Flash ROMs are programmed with same user's application program; each drive executes only those commands that match the drive's name. This simplifies programming, editing and program maintenance because the drives are interchangeable. A single .bin file sent from the GeckoMotion application is written to all drives' Flash ROMs simultaneously.

Once a user's program is debugged and flashed to the drives, the drives can be put into RUN mode (SW1 = OFF, SW2 = ON) and the PC interface can be disconnected. The drives will execute the user's program from its non-volatile memory without a PC connection.

The drive's Flash ROM stores up to 65,536 lines of commands and all commands have fixed 2-word (32-bit) length. Appendix A shows how these commands are coded to this 2-word binary format used by the GM215. This is to allow a user to write their own GUI if they wish to do so. The RS-485 uses a standard UART set to 115,200 baud, 8-bit data, no parity and 1 stop bit.

Commands can be entered and debugged while the GM215 is in the EDIT PROGRAM mode (SW1, SW2 OFF). The GM215 must be connected to PC via its RS-485 interface while in the EDIT PROGRAM mode and the PC must be running the GeckoMotion application.

The commands have 3 main groups; Configuration commands, Motion commands and Program Flow commands:

CONFIGURATION COMMANDS:

CONFIGURE AXIS	Set the axis motor current and other parameters
ACCELERATION	Set axis rate of acceleration
VELOCITY	Set axis target velocity
ANALOG INPUTS	Switch the axis between digital and analog inputs
VECTOR AXIS	Combine 2 or more axis for vector motion
LIMIT CW	Set the axis CW travel limit
ZERO OFFSET	Move axis CW from home a distance equal to offset value
MOVING AVERAGE	Filter the axis motion using a moving average filter (future)
RESPOS	Reset motor position

MOTION COMMANDS:

MOVE	Move axis to an absolute or relative destination
HOME	Home the axis to a hardware switch location
JOG	Move the axis using external inputs
SPEED CONTROL	Operate the axis in velocity mode
POSITION ADJUST	Adjust the axis position using external inputs

PROGRAM FLOW COMMANDS:

GOTO	Go to a program-line, also Loop (4 nested loops allowed)
CALL	Call a subroutine and return (4 nested calls allowed)
IF THEN ELSE	If a condition is met then go to program-line 'n', else go to next program-line
WAIT	Wait a period of time, then go to next program-line
RETURN	This command is used to end a CALL function

MISCELLANEOUS COMMANDS:

OUTPUT	Turn a hardware output ON or OFF	
COMPARE	Stores a variable used by the IF-THEN-ELSE command	
CHANGE SCALE	Change the motion coordinate scale	(future)
ROTATE	Rotate the axis coordinates by an angular value	(future)
TANGENT	Rotate an axis to point in the motion direction	(future)
MIRROR	Mirror the axis coordinates vertically or horizontally	(future)
ENCODER	Axis tracks a quadrature encoder	(future)

The GeckoMotion host program uses labels for program flow commands. This greatly eases the burden of writing and debugging the user program:

00067	**PREVIOUS COMMAND**	
00068	**IF Z INPUT 2 IS OFF GOTO label_1**	An IF command tests if the Z axis input 2 is off. If true, the program jumps to 'label_1:'
00069	**NEXT COMMAND**	If false, the program goes to the next command
----	**------**	
----	**------**	
----	**------**	
00234	**label_1:**	
00235	**SOME OTHER COMMAND**	Program jumped here because the Z axis input 2 is off.
----	**------**	

Note: please see Example code on pages 34-40 and Sample code (Appendix D) on pages 74-77.

CONFIGURE AXIS **CONFIGURATION COMMAND**

Syntax: **a C**ONFIGURE: **i** AMPS, IDLE AT **p** **%** AFTER **s** SECONDS**<ENTER>**
Operands: **a** = X, Y, Z, W
CONFIGURE = configure axis command
0.0 <= **i** <= 7.0
00 <= **p** < 100
00.0 <= **s** =< 25.5
Operation: Sets the motor's operating parameters.
Type: The command settings are global.

Description: This command sets:
1. The selected axis motor phase current from 0 Amps to 7 Amps per phase in 100mA increments. Refer to the motor data-sheet for the correct current setting for motor being used.

2. Sets the motor's standby idle current while the motor is stopped to reduce motor heating. The motor standby current is the set percentage times the motor's set phase current. The drive also goes into recirculating mode switching while in idle to reduce motor eddy-current heating.

3. Sets the time delay before going into the idle mode after the motor stops. Motor current is restored to its full set value immediately after the motor must run again.

Example:

X CONFIGURE: 1.8 AMPS, IDLE AT 71% AFTER 2.5 SECONDS <ENTER>
X means the CONFIG command settings apply to the X axis motor.
CONFIGURE selects the CONFIG command.
1.8 AMPS is the motor phase current setting.
71 means reduce the motor phase current to 71% of the set AMPS value while the motor is stopped.
2.5 SECONDS is the delay time before current is reduced after the motor stops.
<ENTER> indicates the command is finished.

Note: please see Example code on pages 34-40 and Sample code (Appendix D) on pages 74-77.

ANALOG INPUTS **CONFIGURATION COMMAND**

Syntax: **ANALOG INPUTS TO a, a, a, a <ENTER>**
Operands: **a** = axis **X** or **Y** or **Z** or **W**
Operation: Sets inputs to analog for selected axis.
Type: The command settings are global.

Description:
This command sets analog voltage inputs values for acceleration, velocity and position settings for the selected axis.

Example:

ANALOG INPUTS TO Z, X <ENTER>
ANALOG INPUTS is the ANALOG INPUT command
Z includes the Z axis for analog inputs for Acceleration, Velocity and Position values.
, means another axis is to be included.
X includes the X axis for analog inputs for Acceleration, Velocity and Position values.
<ENTER> completes the command. The Y and W axis weren't included so will use digital values for Acceleration,
Velocity and Position in commands that require these settings.

Note: please see Example code on pages 34-40 and Sample code (Appendix D) on pages 74-77.

VECTOR AXIS **CONFIGURATION COMMAND**

Syntax: **VECTOR AXIS ARE a, a, a, a <ENTER>**
Operands: **a** = axis **X** or **Y** or **Z** or **W**
Operation: Associates which axis are to be combined for vector motion.
Type: The command settings are global.

Description:
This command selects which axis will be combined for vector motion. Vector velocity is the same regardless of the vector's direction, all associated axis motion begins and ends at the same time and the path taken will be a straight line connecting the origin and destination coordinates.

Example:

VECTOR AXIS ARE X, Y, W <ENTER>
VECTOR AXIS ARE is the VECTOR AXIS command
X means include the X axis.
, is the delimiter indicating another axis is to be included.
Y means also include the Y axis.
, means another axis is to be included.
W means also include the W axis.
<ENTER> indicates the command is finished.

Note: please see Example code on pages 34-40.

LIMIT CW **CONFIGURATION COMMAND**

Syntax: **a LIMIT CW n <ENTER>**
Operands: **a** = axis **X** or **Y** or **Z** or **W**
0 =< **n** <= 16,777,215
Operation: Sets the axis clockwise travel limit from the home position.
Type: The command settings are global.

Description:
This command sets the maximum distance for any motion away from the HOME position. Its function is to prevent damage to a mechanism if the axis is inadvertently commanded to move beyond the mechanism's safe limits. This can occur if a programming error is made or an axis is jogged past this limit if the limit is exceeded. The motor is decelerated to a stop and locked. Use software reset (press E-STOP button in GUI) or hardware reset (IN1 in RUN mode) to unlock the motor.

This command is optional. If the command isn't used, the default value is 16,777,215.

Example:

Z LIMIT CW 987654 <ENTER>
Z means the Z axis is selected
LIMIT CW means the LIMIT CW command
987654 sets the axis travel limit 987,654 increments of motion away from the home location.
<ENTER> indicates the command is complete.

Note: please see Example code on pages 34-40.

ZERO OFFSET **CONFIGURATION COMMAND**

Syntax: **a ZERO OFFSET n <ENTER>**
Operands: **a** = axis **X** or **Y** or **Z** or **W**
0 <= **n** <= 8,388,607
Operation: Sets the axis 'zero' position CW from the HOME switch.
Type: The command settings are global.

Description:
The command value is used by the HOME command to automatically move the axis CW from the HOME switch location a distance equal to the ZERO OFFSET value. If this command isn't used, the axis will be located at the HOME switch after the HOME command.

Example:

Y ZERO OFFSET 12345 <ENTER>
Y means the Y axis is selected
ZERO OFFSET means the ZERO OFFSET command
12345 means move 12,345 increments of motion CW from the HOME switch.
<ENTER> indicates the command is complete.

ACCELERATION **CONFIGURATION COMMAND**

Syntax: **a ACCELERATION n <ENTER>**
Operands: **a** = axis **X** or **Y** or **Z** or **W**
0 <= **n** <= 32767
Operation: Sets the axis rate of acceleration.
Type: The command settings are global.

Description:
This command sets the motor's rate of acceleration for every command that causes the motor move. This setting can be changed at any time, even while the axis is in motion without affecting the axis destination. If the ANALOG INPUT command includes this axis, then this set value is ignored and the analog value is used during axis motion. If the ANALOG INPUT command later excludes this axis, then the set value for acceleration in this command will apply.

Example:

Y ACCELERATION 12345 <ENTER>
Y means the Y axis is selected
ACCELERATION means the ACCELERATION command
12345 sets the acceleration rate to 12,345.
<ENTER> indicates the command is complete.

During acceleration the acceleration value is added to the axis current velocity register and the sum is output to the axis speed generator. Once the sum equals or exceeds the axis set velocity value, the set velocity is output to the speed generator instead. The sum is updated 1,000 times a second. Time to speed in milliseconds = VELOCITY value / ACCELERATION value

Note: please see Example code on pages 34-40 and Sample code (Appendix D) on pages 74-77.

VELOCITY **CONFIGURATION COMMAND**

Syntax: **a VEL n<ENTER> (or a VELOCITY n<ENTER>)**
Operands: **a** = axis **X** or **Y** or **Z** or **W**
VEL = VELOCITY command
0 <= **n** <= 32767
Operation: Sets the axis velocity limit.
Type: The command settings are global.

Description:
The motor will accelerate at a rate set by the ACCELERATE command to a speed set by the VELOCITY command. If the VELOCITY value is changed while the motor is running, the motor will accelerate or decelerate to the new VELOCITY value. If VELOCITY or ACCELERATION is changed while the axis is in motion, the axis will still reach the programmed destination.

Example:

X VELOCITY 12345 <ENTER> (or X VEL 12345 <ENTER>)
X means the X axis is selected
VEL means the VELOCITY command
12345 sets the velocity limit to 12,345.
<ENTER> indicates the command is complete.

How to calculate motor max speed: VEL (full_step/sec) = 0.3815 * n (0 <= n <= 32767)
Example: x velocity 2000 motor max speed will be: 0.3815 * 2000 = 763 full steps per second
0.3815 comes from 12.5 KHz / 32768

Note: please see Example code on pages 34-40 and Sample code (Appendix D) on pages 74-77.

RESPOS **CONFIGURATION COMMAND**

Syntax: **RESPOS a, a, a, a <ENTER>**
Operands: **a** = axis **X** or **Y** or **Z** or **W**
Operation: Reset motor position.
Dependence: None
Type: The command value is local.

Description:
This command is used to reset motor position with a purpose to let the motor run without hitting limit. Motor position will be reset to its initial value, and therefore, it is under the limit.

Example:

RESPOS X <ENTER>

X means the X axis is selected
RESPOS is the RESPOS command
<ENTER> means the command entry is complete

Note: please see Example code on pages 34-40 and Sample code (Appendix D) on pages 74-77.

MOVE **MOTION COMMAND**

Syntax: **asn, asn, asn, asn<ENTER>**
Operands: **a** = axis **X** or **Y** or **Z** or **W**
s = **+** or **–** or **<SP>**
0 =< **n** <=16,777,215 for **s** = **<SP>**
0 =< n <=8,388,607 for s = + or –
Operation: Moves the listed axis to positions set by the n value.
Type: The command settings are local.

Description:
Because this command is likely to be used far more frequently than any other command, the number of keystrokes to enter this command are kept to an absolute minimum.

SINGLE AXIS MOVE
This command causes the axis to accelerate at the ACCELERATION value rate, reach a speed set by the VELOCITY value and decelerate to a stop at a position set by the **n** destination value. These values can be changed even while the axis is in motion, the axis will stop at the expected destination.

Example:

Z 10000 <ENTER>
Z means the Z axis is selected
<SP> means this is an absolute move
10000 means move to a location that is 10,000 increments of motion from the zero position.
<ENTER> indicates the command is complete.

X+1234 <ENTER>
X means the X axis is selected
+ means this is a relative move in the clockwise direction
1234 means move 1234 increments of motion CW from the present location.
<ENTER> indicates the command is complete.

Y-10 <ENTER>
Y means the Y axis is selected
- means this is a relative move in the counter-clockwise direction
10 means move 10 increments of motion CCW from the present location.
<ENTER> indicates the command is complete.

The axis position is continuously compared to the limit value set by the **LIMIT CW** command. Any move that is beyond that limit results in the axis decelerating to a stop. No further motion command is permitted. Use software reset (press E-STOP button in GUI) or hardware reset (IN1 in RUN mode) to unlock the motor.
Warning: User should use an appropriate value of n when using this command.
For example: X+1000 (move 1000 steps CW)
X-2000 (THEN move 2000 steps CCW) is illegal and should be strictly forbidden. (x -1000 is OK)

MULT-AXIS MOVES
All axis start to move at the same time. If an axis is included in **VECTOR AXIS** command, it will combine with the other included axis to generate a straight-line vector path to the destination. If the axis is not included, it will act independently and use its own ACCELERATE and VELOCITY values to reach the destination set by the **n** value.

X+4000, Y-3000, W 5000 <ENTER>
X means the X axis is selected
+ means this is a relative move in the clockwise direction
4000 means move the X axis 4000 increments of motion CW from the present location.
, indicates another axis is to be included
Y means the Y axis is selected
- means this is a relative move in the clockwise direction
3000 means move the Y axis 3000 increments of motion CCW from the present location.
<ENTER> indicates the command is complete.

Note: please see Example code on pages 34-40 and Sample code (Appendix D) on pages 74-77.

HOME **MOTION COMMAND**

Syntax: **HOME a, a, a, a <ENTER>**
Operands: **HOME** = HOME command
a = axis **X** or **Y** or **Z** or **W**
Operation: Moves the listed axis to the HOME switch location.
Type: The command's OFFSET value is global.

Description:
The **HOME** command moves the selected axis CCW towards a HOME switch connected to **IN2**. The axis acceleration rate is the **ACELERATE** command value and velocity set by the **VELOCITY** command value. When the HOME switch closes, the axis decelerates to a stop, reverses direction to CW and runs at a slow speed while the HOME switch is closed. When the HOME switch opens, an automatic CW move from the HOME switch position is then made. The distance moved is the axis' **ZERO OFFSET** value. When this automatic movement finishes, the axis position register is cleared and current motor position is the origin of your system.

NOTES:

1) **Any point below the origin should be strictly avoided.** For example, from the origin, command such as "x+1000 then x-2000" should be avoided.
2) It is suggested to use a relative high acceleration and a relative low velocity as "go home acceleration and velocity". (See Example code for reference)

Example:

HOME X <ENTER>

HOME is the HOME command
X is the selected axis
<ENTER> means the command entry is finished

Multiple axis can be homed simultaneously to save time spent in this command. The X, Y and Z axis will home at the same time if the command was written as:

HOME X, Y, Z <ENTER>

HOME is the HOME command
X is the selected axis
, means more axis will be included
Y means the Y axis is included
, means more axis will be included
Z means the Z axis is included
<ENTER> means the command entry is finished. The previously set axis OFFSET values will be used.

Note: please see Example code on pages 34-40.

JOG **MOTION COMMAND**

Syntax: **JOG a, a, a, a <ENTER>**
Operands: **a** = axis **X** or **Y** or **Z** or **W**
Operation: Move the axis using switches or trimpots.
Dependence: **ANALOG INPUT** command.
Type: The command requires a hardware exit.

Description:
If the **ANALOG INPUT** command doesn't include the axis, the **JOG** command uses **IN2** as the JOG CW switch input and **IN3** as the JOG CCW switch input. Pushing the CW or CCW switch causes the axis to accelerate at the ACCELERATE command value to a velocity set by the VELOCITY command value. The axis decelerates to a stop when the jog switches are released. The axis position is continuously compared to the **LIMIT CW** command value and the **ZERO OFFSET** position value limits. Any jog switch command that moves the axis out-of-bounds results in the axis decelerating to a stop. Then only jog commands that move the axis back into bounds are allowed.

If the **ANALOG INPUT** command includes the axis, then **TRIM4** is the **JOG** command input. When **TRIM4** is less than 25% the motor will travel CCW. When **TRIM4** is more than 75% the motor will travel CW. When **TRIM4** is between 25% and 75% the motor will stop. The axis out-of-bounds response is the same as for jog switch inputs. **TRIM3** is used to set JOG acceleration, and **TRIM5** is used to set JOG velocity.

Example:

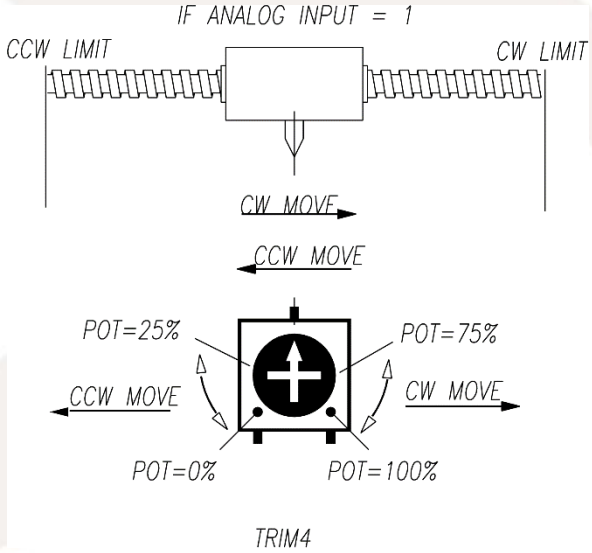
JOG X

X means the X axis is selected
JOG is the JOG command
<ENTER> means the command entry is complete

Once invoked, the **JOG** command cannot terminate on its own because it is an open ended command. A momentary logic '1' on **IN1** exits the **JOG** command. A simultaneous momentary logic '1' on **IN2** and **IN3** (by pressing SW2 and SW3 at the same time) can also exit the **JOG** command. The user program then advances to the next command.

If **JOG** is used in a 2 axis system, a switch-type or potentiometer-type joystick can be used. Apply hardware exit on both drive to exit this command.

Note: please see Example code on pages 34-40 and Sample code (Appendix D) on pages 74-77.



SPEED CONTROL **MOTION COMMAND**

Syntax: **a SPEED CONTROL sn <ENTER>**
Operands: **a** = **X** or **Y** or **Z** or **W**
s = **+** or **-**
0 <= **n** <= 32767
Operation: The axis runs continuously at set speeds.
Dependence: **ANALOG INPUT**, **ACCELERATE** and **VELOCITY** command values.
Type: The command requires a hardware exit.

Description:
This is special 'canned' command that runs the axis continuously at a set speed. **IN1** can be connected to a Stop/Run button, and CW/CCW travel distance limit switches can be used with **IN2** and **IN3** to change motor direction. The **ANALOG INPUT** command value determines the source of acceleration and velocity values if the axis is included. **TRIM5** sets the CW velocity and **TRIM4** sets the CCW velocity.

Example:

Z SPEED CONTROL +12345 <ENTER>
SPEED CONTROL is the SPEED CONTROL command
Z means Z is the selected axis
+ means the motor direction is CW.
12345 means move the axis at a speed of 12345
<ENTER> means the command entry is finished

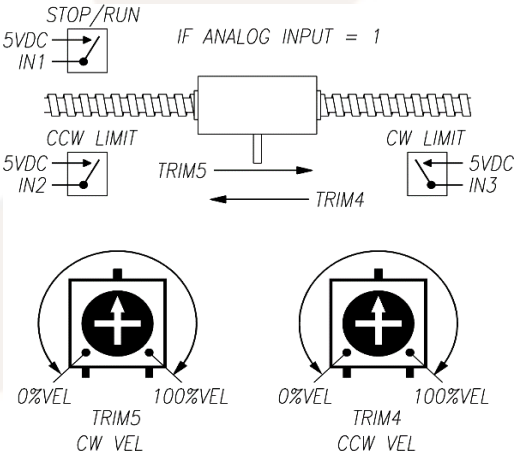
TRIM5 and **TRIM4** multiply the velocity value from 0 to 1 to set the CW and CCW velocities.

OPTIONAL SWITCH INPUTS (use momentary switches only)
If limit switches are used, **IN2** is the CCW travel distance limit, **IN3** is the CW travel distance limit and **IN1** is the Run/Stop switch input.
The initial motor direction is set by the **SPEED CONTROL** command sign operand. The motor will accelerate to speed in the CW direction if the sign is '**+**'. The axis moves in the initial direction until it encounters a limit switch. The motor then decelerates to a stop and then accelerates to speed in the opposite direction. This process continues indefinitely until the command is terminated.

At any time the axis can be decelerated to a stop by applying a logic '1' to **IN1**. Releasing **IN1** (logic '0') results in the axis accelerating back to speed in the same direction it was going before.

Once invoked, the **SPEED CONTROL** command cannot terminate on its own because it is an open ended command. A simultaneous momentary logic '1' on **IN2** and **IN3** (by pressing SW2 and SW3 at the same time) exits the **SPEED CONTROL** command. The user program then advances to the command.

Note: please see Example code on pages 34-40 and Sample code (Appendix D) on pages 74-77.



POSITION ADJUST **MOTION COMMAND**

Syntax: **a POSITION ADJUST +/- n <ENTER>**
Operands: **a** = **X** or **Y** or **Z** or **W**
0 =< **n** <= 32,767
Operation: Adjusts the axis position using an analog voltage.
Dependence: None
Type: The command requires a hardware exit.

Description:

This command uses **TRIM5** to adjust the axis position within a CW / CCW range set by the **n** value. The TRIM5 setting adjusts the axis position over a +/- 100% range of the **n** value; the adjustment is 0% when TRIM5 is at the mid-point position.

Warning: Home (CCW limit) and CW limit will not be checked. User should use appropriate values of n when using this command.

Example:

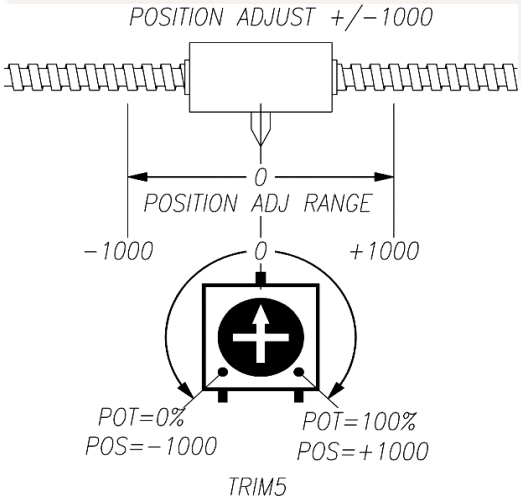
X POSITION ADJUST +/- 1000 <ENTER>

X means X is the selected axis.
POSITION ADJUST +/- means the POSITION ADJUST command.
1000 means set the full-scale range to +/- 1,000 increments of motion.
<ENTER> indicates the command entry is complete.

In this example, 2,000 is the number of steps required to turn the motor 1 full revolution. Turning TRIM5 from zero to full scale will proportionately adjust the motor from 1 revolution CCW of its commanded position to 1 revolution CW of its commanded position.

If POSITION ADJUST is used in a 2 axis system, a potentiometer-type joystick can be used to adjust the x,y position. An example of its utility would be to index a sample underneath a microscope, use joystick to examine the sample.
A momentary logic '1' must be applied to IN1 on the axis to exit this command.

Note: please see Example code on pages 34-40 and Sample code (Appendix D) on pages 74-77.



PROGRAM FLOW COMMANDS:

Program flow commands break the normal sequential order of program flow by forcing an out of sequence location of the next command. Commands then execute sequentially again starting at the new location. The main program flow commands are unconditional jumps and loops (**GOTO**), calls and returns from subroutines (**CALL**), If-Then-Else conditional jumps (**IF**) and wait loops (**PAUSE**).

GOTO **PROGRAM FLOW COMMAND**

Syntax: **GOTO k, LOOP n TIMES <ENTER>**
Operands: k = **<LABEL>**
0 <= **n** <= 255
Operation: Go to label name location in the user program.
Dependence: None
Type: The command value is local.

Description:
This command jumps to the label name location and resumes executing the program from there. Additionally, the command can jump to a label location a set number of times before the command is ignored (LOOP **n** TIMES).

Example:
GOTO abcd <ENTER>
GOTO is the GOTO command
abcd means jump to program a location named 'abcd:'
<ENTER> indicates the command entry is done

Example:
GOTO abcd, LOOP 8 TIMES <ENTER>
GOTO is the GOTO command
abcd means jump to program a location named 'abcd:'
, indicates looping is required
8 means LOOP 8 TIMES through this command before going to the next program line after this command
<ENTER> indicates the command entry is done

In the below example, **SOME COMMAND** executes at line 01007 and then the GOTO command at line 01008. The GOTO loop count decrements and if it's not zero, the program jumps to '**abcd:**'. This repeats 5 times until the LOOP count is zero. On the zero count the LOOP counter is set to 5 again, the GOTO jump is ignored and the program advances to **NEXT COMMAND** on line 01009.

```
01006    abcd:
01007       SOME COMMAND
01008       GOTO abcd, LOOP 5 TIMES
01009       NEXT COMMAND
```

Nested loops:
Nested loops are loops within loops. An example would be some process that has to be repeated for 'y' rows and 'x' columns. In the example below, there are 5 rows and 10 columns. The program loops through line 01004 to line 01010 ten times. Lines 01006 and 01008 are repeated five times for every pass through the first loop. Lines 01007 and 01008 get repeated 50 times.
Note: Up to 4 nested loops are allowed.

```
01004    outer_loop:
01005       SOME OTHER COMMAND
01006    inner_loop:
01007       SOME COMMAND
01008       GOTO inner_loop, LOOP 5 TIMES
01009       NEXT COMMAND
01010       GOTO outer_loop, LOOP 10 TIMES
01011       ANOTHER COMMAND
```

Note: please see Example code on pages 34-40 and Sample code (Appendix D) on pages 74-77.

CALL **PROGRAM FLOW COMMAND**

Syntax: **CALL k <ENTER>**
Operands: k = <LABEL>
Operation: Calls a function. Returns to the program line after this one when finished.
Dependence: None
Type: The command value is local.

Description:
This command jumps from the current command line to a command line specified by the value **n**. The program continues from that line until the function is finished. The last command line in the function must be **CALL** which causes a return to the command line immediately after the one which called the function.
Note: Up to 4 nested function calls are allowed.

Example:

CALL function_name <ENTER>
CALL is the CALL command
function_name means the CALL function begins at a program location labeled as '**function_name:**'
<ENTER> indicates the command entry is done

The following example shows how nested CALLs are used:

00198	**COMMAND**	
00199	**CALL** func_1	Call a subroutine beginning at line 758.
00200	**COMMAND**	←Return from **func_1** here.
00201	**COMMAND**	
-------	-------	
00311	**func_2:**	
00312	**COMMAND**	← **CALL func_2** from line 00760 from inside **func_1** calls another function starting on this command line.
00313	**COMMAND**	
00314	**RETURN**	Return. Subroutine **func_2** is finished.
--------	-------	
00757	**func_1:**	
00758	**COMMAND**	← **CALL func_1** from line 00199 calls a subroutine starting on this command line.
00759	**COMMAND**	
00760	**CALL** func_2	Call another subroutine beginning at line 312. This call is from inside the CALL 758 function.
00761	**COMMAND**	←Return from **func_2** here.
00762	**RETURN**	Return. Subroutine **func_1** is finished

Note: please see Example code on pages 34-40 and Sample code (Appendix D) on pages 74-77.

RETURN **PROGRAM FLOW COMMAND**

Syntax: **RETURN <ENTER>**
Operands: None
Operation: Returns to the program line after the CALL command line.
Dependence: None
Type: The command value is local.

Description:
This command is used to end a CALL function. The program counter jumps to the next program line after the CALL command line.

Example:

RETURN <ENTER>
RETURN means this a RETURN command.
<ENTER> indicates the command entry is completed

Note: please see Example code on pages 34-40 and Sample code (Appendix D) on pages 74-77.

WAIT **PROGRAM FLOW COMMAND**

Syntax: **WAIT nn.nnn SECONDS <ENTER>**
Operands: 0 <= **nn.nnn** <= 65.535
Operation: Pauses program execution for a period of time
Dependence: None
Type: The command value is local.

Description:
This command inserts a time delay on **n** milliseconds to pause the user program.

Example:

WAIT 2.500 SECONDS <ENTER>
WAIT means this is a PAUSE command.
2500 means wait for 2,500 milliseconds (2.5 seconds)
<ENTER> indicates the command entry is done

Note: please see Example code on pages 34-40 and Sample code (Appendix D) on pages 74-77.

IF-THEN-ELSE **PROGRAM FLOW COMMAND**

Syntax: **IF a t IS c GOTO k <ENTER>**
Operands: **a** = X or **Y** or **Z** or **W** axis name
t = **IN1** or **IN2** or **IN3** or **RDY** or **ERR** single-bit variable test or **VEL** or **POS** or **VIN** multiple-bit variable test
c = **ON** or **OFF** single-bit test or **>** or **=** or **<** multiple-bit variable test against COMPARE
k = **<LABEL>**
Operation: IF a test is true, THEN go to the label name program location, ELSE go to the next program line.
Dependence: None
Type: The command value is local.

Description:
This command is a conditional jump. A single bit variable **i** is tested for ON or OFF. If true, then the next command line location is at the labeled name. Else the next command line is executed. Multiple-bit variables VELOCITY, POSITION and VIN are compared against the COMPARE value (see COMPARE command). VIN is a 1-byte analog to digital conversion of a 0V to 5V input to the GM215. VEL is a 2-byte value of the axis's current velocity and POS is a 3-byte value of the axis' current position.

Examples:

IF X IN 3 IS ON GOTO label <ENTER>
IF means this is an IF-THEN-ELSE command.
X means test an X axis input or variable.
IN3 means test general purpose input number 3.
ON means the test is for an ON state.
Label is the GOTO label if the test is true.

IF Z VEL IS > GOTO vel_is_bigger <ENTER>
IF means this is an IF-THEN-ELSE command.
Z means test a Z axis input or variable.
VEL means the axis current velocity.
> Means test if velocity is greater than the COMPARE value.
vel_is_bigger is the GOTO label if the test is true.

IF W VIN IS < GOTO low_voltage <ENTER>
IF means this is an IF-THEN-ELSE command.
W means test a W axis input or variable.
VIN means test the axis analog input voltage VIN.
< Means test VIN to determine if it is less than the COMPARE value.
low_voltage is the GOTO label if the test is true.

An input can be continuously polled (tested) for an input state. If the input test is false, the input test is repeated until the result is true. When true, the user program advances to the next program line.

A HOME command is required when a switch connected to the X axis IN1 turns ON and the program waits until this is true. Assume the code to do this starts at command line 00123.

PGM LINE	COMMAND	COMMENT
00122	**wait_for_switch:**	Label name
00123	**IF X IN1 IS OFF GOTO wait_for_switch**	while the X axis IN1 is OFF the command THEN keeps retesting IN1
00124	**HOME X**	← ELSE if you are here, IN1 was ON. Home the axis.
00125	**NEXT COMMAND**	Do something after the axis has homed

If it's undesirable to stall the program while waiting for IN1 to be ON and it can be tested later, the IF command can be written as:

PGM LINE	COMMAND	COMMENT
00122	**IF X IN1 IS OFF GOTO switch_is_off**	Test the X axis for IN1 is OFF
00123	**HOME X**	← ELSE if you are here, IN1 was ON. Home the axis.
00124	**switch_is_off:**	Label name
00125	**NEXT COMMAND**	Continue with the program but have it loop through PGM LINE 00122 again.

Assume X axis output 3 must be turned 'on' if the Z axis voltage on VIN is between 2.5V and 3.0V.

PGM LINE	COMMAND	COMMENT
00455	Z COMPARE VALUE 154	256 times 3.0V divided by 5.0V
00456	IF Z VIN > GOTO out3_off	GOTO label **out3_off** if VIN is greater than 3V.
00457	Z COMPARE VALUE 128	256 times 2.5V divided by 5.0V
00458	IF Z VIN > GOTO out3_on	GOTO label **out3_on** if VIN is greater than 2.5V.
00459	out3_off:	Label name
00460	X OUT 3 OFF	Turn X axis output 3 off.
00461	X GOTO skip_line	GOTO label **skip_line**
00462	out3_on:	Label name
00463	X OUT 3 ON	Turn X axis output 3 on.
00464	skip_line:	Label name
00465	NEXT COMMAND	Continue with the program

The following is a partial list of Boolean operations on 2 and 3 inputs using the **IF** command. They are offered as a template to the user from which other, not listed logical operations can be formed:

AND2

```
IF X IN1 IS OFF GOTO false
IF X IN2 IS OFF GOTO false
SOME COMMAND (true)
false:
SOME OTHER COMMAND
```

AND3

```
IF X IN1 IS OFF GOTO false
IF X IN2 IS OFF GOTO false
IF X IN3 IS OFF GOTO false
SOME COMMAND (true)
false:
SOME OTHER COMMAND
```

XOR2

```
IF X IN1 IS ON GOTO next
IF X IN2 IS ON GOTO true
false:
SOME COMMAND
next:
IF X IN2 IS ON GOTO false
true:
SOME OTHER COMMAND
```

OR2

```
IF X IN1 IS ON GOTO true
IF X IN1 IS ON GOTO true
false:
SOME COMMAND
true:
SOME OTHER COMMAND
```

OR3

```
IF X IN1 IS ON GOTO true
IF X IN2 IS ON GOTO true
IF X IN3 IS ON GOTO true
false:
SOME COMMAND
true:
SOME OTHER COMMAND
```

OUTPUT **MISCELLANEOUS COMMAND**

Syntax: **a OUT n c <ENTER>**
Operands: **a** = X or **Y** or **Z** or **W** axis name
n = **1** or **2** or **3**
c = **ON**, **OFF, BR, RS, ER**
Operation: Outputs OUT1 or OUT2 or OUT3
Dependence: None
Type: The command value is local.

Description: Turns a hardware output ON or OFF

Examples:

X OUT 3 ON <ENTER>
X means the X axis is selected.
OUT is the output command.
3 is the hardware output number 3.
ON means turn this output ON.
<ENTER> means the command entry is complete.

Y OUT 1 BR <ENTER>
BR is Busy/Ready status.
Output1 of Y axis is set to reflect Busy/Ready status.

Z OUT 2 RS <ENTER>
RS is the R/S status.
Output2 of Z axis is set to reflect R/S status.

W OUT 3 ER <ENTER>
ER is the Error status.
Output3 of W axis is set to reflect ER status.

COMPARE **MISCELLANEOUS COMMAND**

Syntax: **a COMPARE VALUE n <ENTER>**
Operands: **a** = X or **Y** or **Z** or **W** axis name
0 =< **n** <= 16,777,215
Operation: Stores the value n to the COMPARE register
Dependence: None
Type: The command value is global.

Description: This command is used in conjunction with the IF-THEN-ELSE command when a variable has to be compared against a stored COMPARE value

Example:

X COMPARE VALUE 32767 <ENTER>
X means the X axis is selected.
COMPARE VALUE is the COMPARE command.
32767 is the value to be stored in the COMPARE register.
<ENTER> means the command entry is complete.

Note: please see Example code on pages 34-40.

EXAMPLE CODE:

EXAMPLE 0: MOVE A MOTOR

```
        x configure: 1 amps, idle at 50% after 1 seconds
        x limit cw 100000
        x offset 1000
        analog inputs to {0}            ; NO AXIS USING ANALOG
        vector axis are {0}             ; NO AXIS USING VECTOR
        x acceleration 64               ; RUN ACCELERATION
        x velocity 200                  ; RUN VELOCITY
motion:
        x+1000
        x-1000
        goto motion                     ; REPEAT. INFINITE LOOP
```

EXAMPLE 1: BASIC POINT TO POINT MOTION

```
x_config:
        x configure: 1 amps, idle at 50% after 1 seconds
        x limit cw 12000000
        x offset 1000

y_config:
        y configure: 1.5 amps, idle at 50% after 1 seconds
        y limit cw 12000000
        y offset 1000

start:
        x acceleration 512              ; GO HOME ACCELERATION
        x velocity 1000                 ; GO HOME VELOCITY
        y acceleration 512
        y velocity 1000
        home x, y                       ; X & Y GO HOME SIMULTANEOUSLY

        analog inputs to {0}            ; NO AXIS USING ANALOG
        vector axis are {0}             ; NO AXIS USING VECTOR
        x acceleration 128              ; RUN ACCELERATION
        x velocity 8000                 ; RUN VELOCITY
        y acceleration 128
        y velocity 8000

motion1:
        x+10000, y+10000
        x-10000, y-10000
        goto motion1                    ; REPEAT. INFINITE LOOP
```

EXAMPLE 2: BASIC VECTOR MOTION

```
x_config:
        x configure: 1 amps, idle at 50% after 1 seconds
        x limit cw 12000000
        x offset 1000

y_config:
        y configure: 1 amps, idle at 50% after 1 seconds
        y limit cw 12000000
        y offset 1000

start:
        x acceleration 512          ; GO HOME ACCELERATION
        x velocity 1000             ; GO HOME VELOCITY
        y acceleration 512
        y velocity 1000
        home x, y                   ; X & Y GO HOME SIMULTANEOUSLY

        vector axis are x, y        ; X & Y AXIS VECTOR MOTION
        x acceleration 128
        y acceleration 128
        x velocity 8000             ; SET DIFFERENT VELOCITY FOR TESTING
        y velocity 4000

motion1:
        x+10000, y+10000
        x-10000, y-10000
        goto motion1                ; INFINITE LOOP
```

EXAMPLE 3: POINT TO POINT MOTION, SPEED CONTROLLED BY TRIMPOT (ANALOG)

```
x_config:
        x configure: 1 amps, idle at 50% after 1 seconds
        x limit cw 12000000
        x offset 1000

y_config:
        y configure: 1 amps, idle at 50% after 1 seconds
        y limit cw 12000000
        y offset 1000

start:
        x acceleration 512          ; GO HOME ACCELERATION
        x velocity 1000             ; GO HOME VELOCITY
        y acceleration 512
        y velocity 1000
        home x, y                   ; X & Y GO HOME SIMULTANEOUSLY

        analog inputs to x, y       ; X & Y AXIS USING ANALOG CONTROL
        vector axis are {0}         ; NO AXIS USING VECTOR

motion1:
        x+10000, y+10000            ;
        x-10000, y-10000
        goto motion1                ; INFINITE LOOP
```

EXAMPLE 4: RUN MOTION UNDER ABSOLUTE POSITION SYSTEM

```
x_config:
        x configure: 1 amps, idle at 50% after 1 seconds
        x limit cw 12000000
        x offset 1000

y_config:
        y configure: 1 amps, idle at 50% after 1 seconds
        y limit cw 12000000
        y offset 1000

start:
        x acceleration 512          ; GO HOME ACCELERATION
        x velocity 1000             ; GO HOME VELOCITY
        y acceleration 512
        y velocity 1000
        home x, y                   ; X & Y GO HOME SIMULTANEOUSLY

        vector axis are x, y        ; X & Y AXIS VECTOR MOTION
        x acceleration 128          ; RUN ACCELERATION
        y acceleration 128
        x velocity 8000              ; RUN VELOCITY
        y velocity 8000

motion1:
        x 2000, y 2000              ; ABS POSITION. USEFUL FOR DRAWING, CUTTING, ETC.
        x 1500, y 2500
        x 2500, y 1500
        x 2000, y 2000
        x 0, y 0
        goto motion1                ; INFINITE LOOP
```

EXAMPLE 5: USING **JOG** COMMAND

```
x_config:
        x configure: 1 amps, idle at 50% after 1 seconds
        x limit cw 12000000
        x offset 1000

start:
        x acceleration 512          ; GO HOME ACCELERATION
        x velocity 1000             ; GO HOME VELOCITY
        home x                      ; X AXIS GO HOME
        x acceleration 128          ; RUN ACCELERATION
        x velocity 8000             ; RUN VELOCITY

        jog x                       ; JOG MODE. NEED HARDWARE LOGIC TO EXIT
        goto start                  ; INFINITE LOOP
```

Please see more Sample code (Appendix D) on pages 74-77.

EXAMPLE 6: USING **SPD CONTROL** COMMAND

```
        x configure: 1 amps, idle at 50% after 1 seconds
        x limit cw 12000000
        x offset 1000
        x acceleration 128
        x velocity 8000
        x speed control +1000       ; SPEED CONTROL MODE. NEED HARDWARE LOGIC TO EXIT
end:
        goto end                    ; INFINITE LOOP
```

Please see more Sample code (Appendix D) on pages 74-77.

EXAMPLE 7: USING **SPD CONTROL** COMMAND (ANALOG)

```
        x configure: 1 amps, idle at 50% after 1 seconds
        x limit cw 12000000
        x offset 1000
        x acceleration 128
        x velocity 8000

        analog inputs to x          ; X AXIS ANALOG. TRIM4 &TRIM5 CAN CHANGE SPD
        x speed control +1000       ; SPEED CONTROL MODE. NEED HARDWARE LOGIC TO EXIT
end:
        goto end                    ; INFINITE LOOP
```

Please see more Sample code (Appendix D) on pages 74-77.

EXAMPLE 8: USING **POSITION ADJUST** COMMAND

```
x_config:
        x configure: 1 amps, idle at 50% after 1 seconds
        x limit cw 12000000
        x offset 1000

start:
        x acceleration 512          ; GO HOME ACCELERATION
        x velocity 1000             ; GO HOME VELOCITY
        home x                      ; X AXIS GO HOME
        x acceleration 128          ; RUN ACCELERATION
        x velocity 8000             ; RUN VELOCITY
        x+3000
        x position adjust +/-2000   ; TRIM5 CAN CHANGE MOTOR POSITION
end:
       goto end                     ; INFINITE LOOP
```

Please see more Sample code (Appendix D) on pages 74-77.

EXAMPLE 9: USING **RESPOS** COMMAND (IMPLEMENT AN INDEX FUNCTION)

```
        x configure: 1 amps, idle at 50% after 1 seconds
        x limit cw 12000000
        x offset 1000
        x acceleration 128          ; RUN ACCELERATION
        x velocity 8000             ; RUN VELOCITY
        goto LOOP1
INDEX1:
        respos x
        x-2000
        goto LOOP1
INDEX2:
        respos x
        x+2000
LOOP1:
        if x in3 is on goto INDEX1
        if x in2 is on goto INDEX2
        goto LOOP1                  ; INFINITE LOOP
```

Please see more Sample code (Appendix D) on pages 74-77.

EXAMPLE 10: USING **ANALOG INPUT** AND **COMPARE** COMMAND

```
        x configure: 1 amps, idle at 50% after 1 seconds
        x limit cw 100000
        x offset 1000
        analog inputs to {0}            ; NO AXIS USING ANALOG
        vector axis are {0}             ; NO AXIS USING VECTOR
        x acceleration 64               ; RUN ACCELERATION
        x velocity 200                  ; RUN VELOCITY

        x compare value 511             ; ABOUT 2.5V
check:
        if x vin is > goto motion
        goto check
motion:
        x+1000
        x-1000
        goto check                      ; REPEAT. INFINITE LOOP
```

NOTE: THIS IS JUST AN EXPLORATION OF POSSIBLE NEW COMMANDS.

MOVING AVERAGE FILTER **CONFIGURATION COMMAND**

Syntax: **MOVING AVERAGE a, a, a, a n SAMPLES <ENTER>**
Operands: **a** = axis **X** or **Y** or **Z** or **W**
0 < **n** <=127
Operation: Associates which axis are moving average filtered.
Type: The command settings are global.

Description:
1) Moving average filtering dampens the 'jerk factor' (makes the velocity 2nd derivative finite) when an axis uses ramped acceleration.
2) Rounds abrupt vector angle changes during vector motion. This rounding allows higher vector velocities and it smooths the vector path along piece-wise linear approximations of curves.

Example:

MOVING AVERAGE X, Y, 100 SAMPLES <ENTER>
MOVING AVERAGE is the MOVING AVERAGE command
X includes the X axis
, indicates another axis is to be included
Y includes the Y axis
<SP> means all included axis are listed
100 means the MOVING AVERAGE filter will use 100 samples
<ENTER> means the command is complete

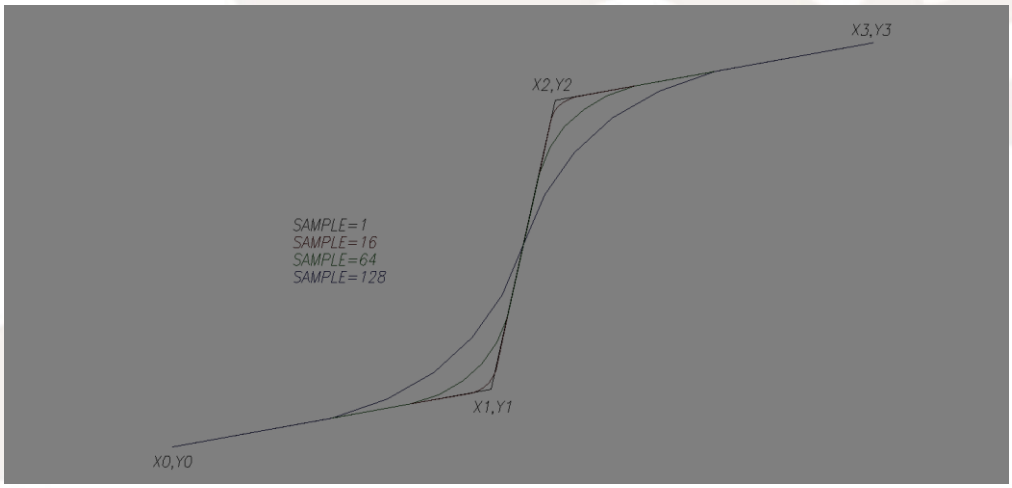
The moving average filter can be turned ON or OFF for selected axis. Using the previous example, the following turns filtering OFF for the X and Z axis:

MOVING AVERAGE Y <ENTER>

The X and Z axis have been removed from the included list. The SAMPLES value is unchanged and continues to be used for the Y axis. To turn the moving average filter back ON for the just the X and Y axis:

MOVING AVERAGE X, Y <ENTER> To turn the filter OFF for all axis: **MOVING AVERAGE <ENTER>**

The figure below shows the effect of different SAMPLE values. The black path shows the motion path has a SAMPLE value of 1 (no moving average filtering). The red path is for a SAMPLE value of 16, the green path is for a SAMPLE value of 64 and the blue path is for a SAMPLE value of 128. The vector motion path becomes progressively more rounded as the number of SAMPLE values increase.



GROUP AXIS **CONFIGURATION COMMAND**

Syntax: GROUP A IS ad ad ad a AXIS <ENTER>
Operands: **G** = GROUP command
a = axis **X** and **Y** and **Z** and **W** in that order
d = **,** or **<SP>**
Operation: Assigns axis to groups

Description:
The **GROUP** command assigns the axis to groups that have common function and each member axis in a group runs from the same user program. A different group can run a different user program. A group has 1 to 4 member drives and up to 4 groups can be active if each group has a single axis as a member.

The first group is called GROUP A and it must have the X axis as its first member. Additional members (if any) must be the Y axis, then the Z axis and then the W axis in that order. Additional groups (if any) are called GROUP B, GROUP C and GROUP D.

All axis, no matter which group they are in, know the status of all the other attached axis.

The purpose of grouping the axis allows multiple drives to perform tasks independently of other grouped axis.

EXAMPLE:

GROUP A IS X, Y AXIS, GROUP B IS Z, W AXIS <ENTER>

In this example the GROUP A user program might command the X, Y axis motors to operate an X, Y table while the GROUP B user program might command the Z, W axis motors to operate a lathe. Both programs would run independently of each other.

EXTERNAL INTERRUPT **PROGRAM FLOW COMMAND**

Syntax: EXTERNAL INTERRUPT GOTO n IF a AXIS i TURNS c <ENTER>
Operands: **a** = **X** or **Y** or **Z** or **W** axis name
i = **IN1** or **IN2** or **IN3** or **RDY** or **ERR**
c = **ON** or **OFF**
0 =< **n** < 65,536
Operation: If axis **a** input **I** condition **c** is true, go to program line **n.**

EXAMPLE:

EXTRERNAL INTERRUPT GOTO 12345 IF Z AXIS IN3 TURNS ON <ENTER>

In this example maybe the user wishes to pause the X, Y axis motion when IN3 turns on. The interrupt service code located at 12345 would have commands that see if the axis velocity value is zero. If not, the value would be made zero. If it was zero, the original velocity value would be restored. For that example IN3 would act as push-on, push-off switch pause and resume axis motion.

CHANGE SCALE	**MISCELLANEOUS COMMAND**
ROTATE	**MISCELLANEOUS COMMAND**
ENCODER	**MISCELLANEOUS COMMAND**
TANGENT	**MOTION COMMAND**

The EDIT MODE is used to write, debug or read back the user program to and from the GM215. To use the EDIT mode, user needs to do the followings:

1) **Download GeckoMotion Controller and Configuration Software** by using this link:

http://www.geckodrive.com/support/geckomotion.html

Click at "GeckoMotion Program" to start to download Setup file. Run Setup to install Geckomotion on your computer. After the installation, user will get:

- C:\Python2710\python.exe (Python executable file)
- C:\Python2710\Scripts\gmotion.py (GeckoMotion application file)
- C:\Python2710\Lib\site-packages\geckomotion__init__.py, assemble.py, devices.py, gmgui.py, gm.glade (GeckoMotion source files)

2) **Run GeckoMotion Controller and Configuration Software**:

Set the device in Edit mode (SW1 & SW2 off) and set axis name properly (if only one axis is used, it should be set as X-axis. If 2 axis are used, one is set as X-axis and the other one will be Y-axis). In this example, only X-axis will be used. GeckoMotion can be run directly from the bin director.

2.1 WINDOWS:

- Double click **gmotion.py** from Explorer (in C:\Python2710\Scripts\). A shortcut can be created for gmotion.py on the desktop.
- Use run on the start menu, browse to **python.exe>gmotion.py**
- Start from command prompt: **C:\Python2710\python.exe C:\Python2710\Scripts\gmotion.py**
  Or **python.exe C:\Python2710\Scripts\gmotion.py**

2.2 LINUX:

Do the followings:

- Use a shebang line for the Python interpreter
  `#!/usr/bin/env python`
- Make the script executable
  `chmod +x gmotion.py`
- If the script resides in a directory that appears in the PATH variable, user can simply type
  `$ gmotion.py`
- Otherwise, user needs to provide the full path (either absolute or relative). This includes the working directory, which should not be in your PATH
  `$ ./gmotion.py`

3) **Setup the GeckoMotion**:

Python window (Figure 9) and GeckoMotion window (Figure 10) should be displayed after the software is run, otherwise, Device Error window will be displayed (Figure 8) and user needs to do step 3.1 to establish the connection first.

- 3.1 Verify:

- Power supply should be on.
- The device should work properly. It will blink green green green if there is an attached motor, otherwise, make sure there is no error code between code 5 and code 9 (see Appendix E for more details in error code).
- The device should be set in Edit mode (SW1 & SW2 are off).
- Axis name should be set properly (should have at least one X-axis in the system).
- Should have correct RS485 connection.
- Turn power off and on again if necessary.
- Click **Continue** in Device Error window to advance to next step.

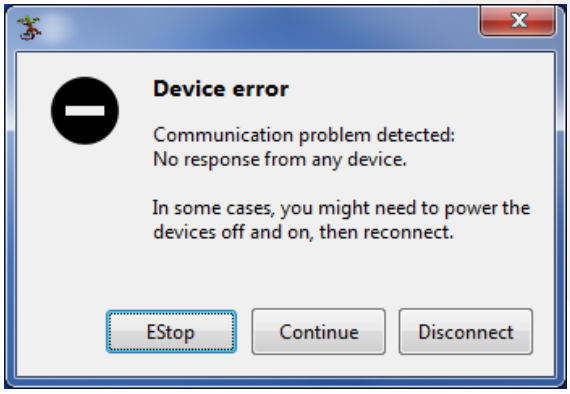


Figure 8: Device Error window

- 3.2 Python window and GeckoMotion window description:

Figure 9: Python window

Python window is used to display communication activities and status such as sent command, feedback data, error message, etc. To close this window will terminate GeckoMotion program.

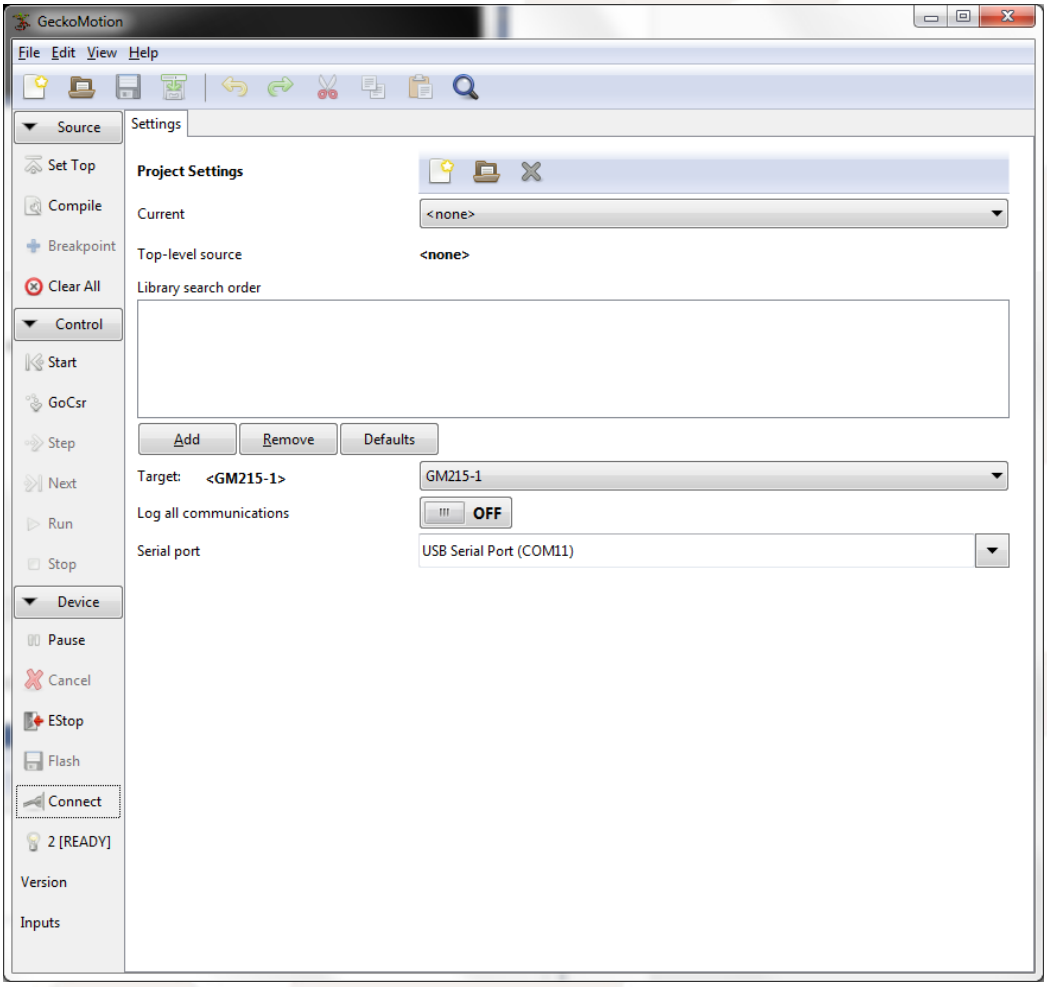


Figure 10: GeckoMotion window

The GeckoMotion window consists of 3 main parts: the menu, the toolbar, the tab pages.
The menu contains File, Edit, View, Help.
The tool bar contains Create new file, Open file, Save file, Etc.
The tab pages contain
- **Source** has 4 buttons: Set Top, Compile, Breakpoint, Clear All.
- **Control** has 6 buttons: Start, GoCsr, Step, Next, Run, Stop.
- **Device** has 6 buttons: Pause, Cancel, EStop, Flash, Connect, READY.
- **Settings** has 3 tabs: Target, Log all communications, Serial port.
- Extra buttons: Version, Inputs.

- 3.3 GeckoMotion window setup:

  Set the **Settings** in tab pages as followings:
  - Target: Select **GM215** or **GM215-1** for 3in-3out model, select **GM215-2** for 4in-2out model.
  - Log all communications: Set to **OFF** for this moment.
  - Serial port: Select the available port for this RS485 interface.

  Click Connect button under **Device** tab to establish the connection. If the connection is good, **1 [READY]** will be displayed (under the Connect button) and the following messages will be displayed on Python window:

  Detected axis 0 = X

  Otherwise, Device Error window will be displayed and user should go back to step 3.1.

4) **Create a program using GeckoMotion**:

4.1 Create a new program:
- In the menu, click File and then select **New**. A new tab will be inserted next to **Settings** tab with the name **<untitled>.**
- Click at **<untitled>** tab to open it and write a program (or copy-paste from other text editor).

For example:

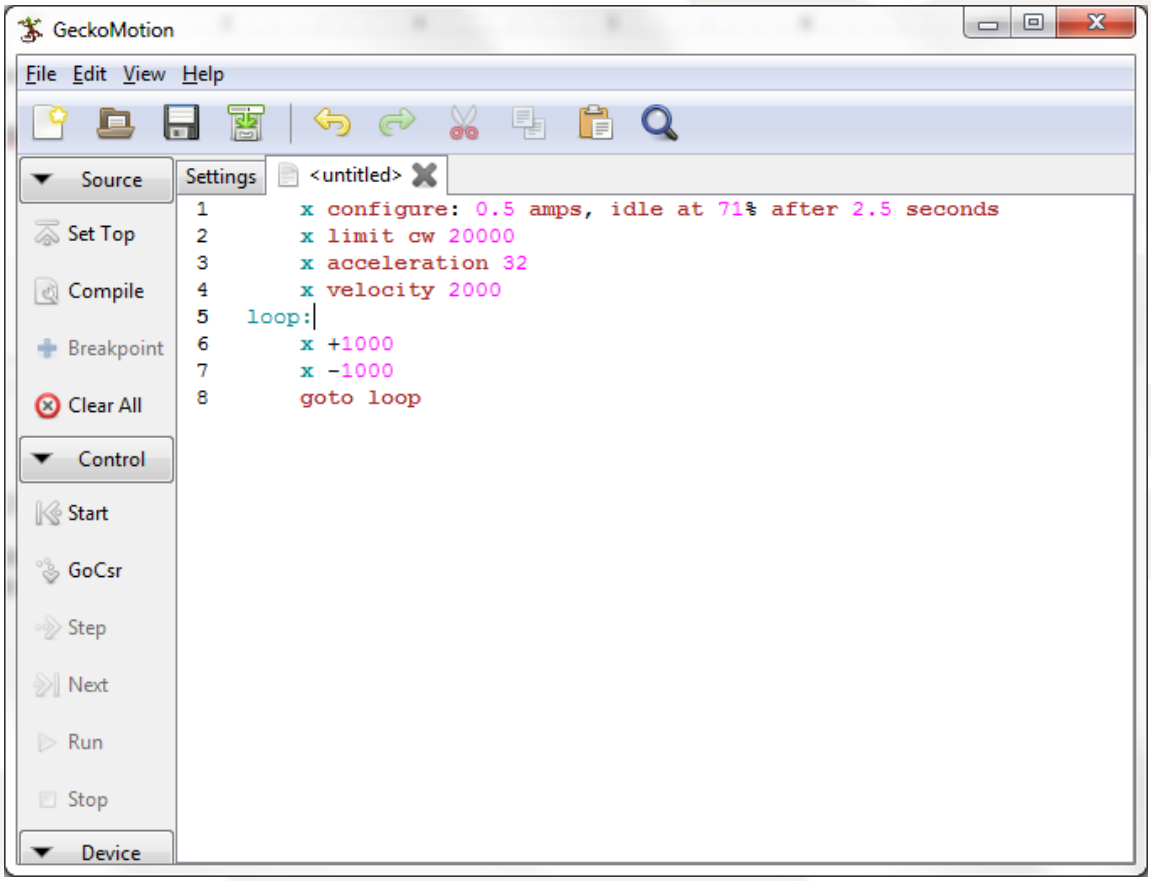


Figure 11: How to create a new file.

4.2 To save and rename a file: In the menu, click File and then select **Save**. This will open Save GeckoMotion Source File window (see Figure 12).
- Type in the new file name, e.g., **Name**: **Testfile** (see Figure 12).
- Select the folder to store this file.
  For example: **Save in folder**: **Users>…..>Gmotion Test Files>2016**.
- Select file type: keep this file type default setting: **GeckoMotion files** (see Figure 12).
- Click OK.

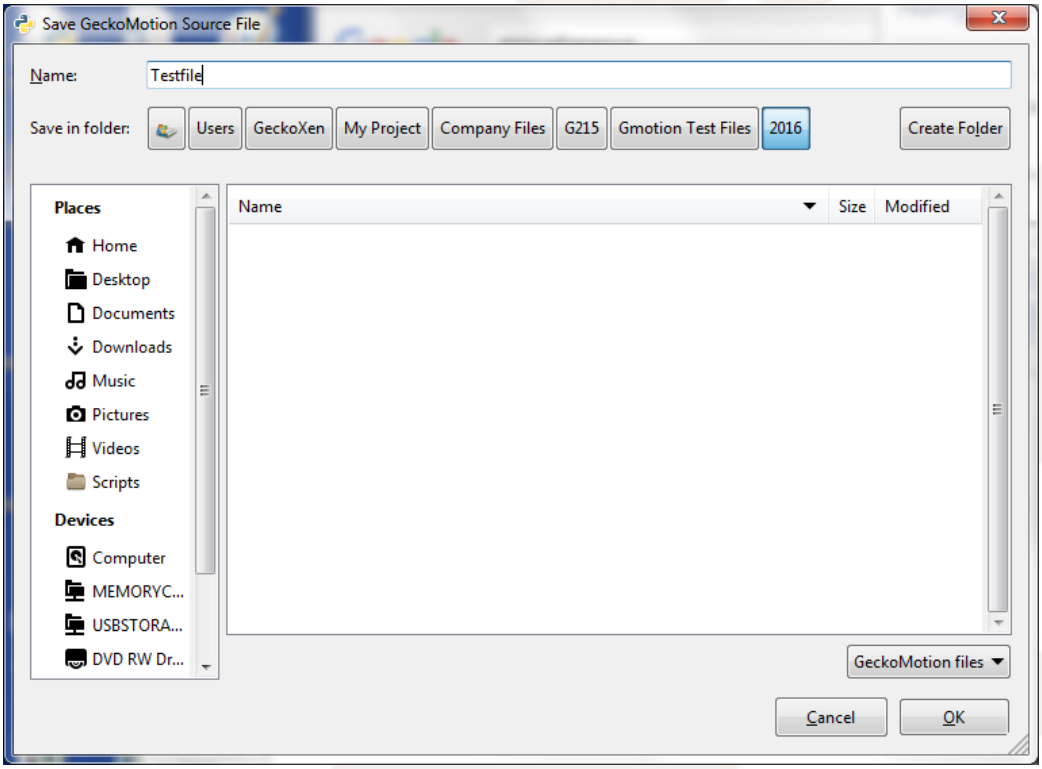


Figure 12: How to save and rename a file.

File name **<untitled>** will be replaced by **Testfile**.

4.3 To close a file: Click at **X** after the file name to close the file.
Example: In Figure 11, click at **X** after <untitled> will close <untitled>.

4.4 Reopen the file: In the menu, click File and then select **Open**. This will open Open GeckoMotion Source File window (Figure 13).

- Select the folder that contains the file. In this example, it is the folder "…>Gmotion Test Files>2016>".
- Select file type: **Any file**s (see Figure 13). "Testfile" should be displayed under **Name** tab in this window.
- Select "Testfile" and click OK. Testfile will be added to the list of opened files. Click at "Testfile" to open for edit, compile, or run.

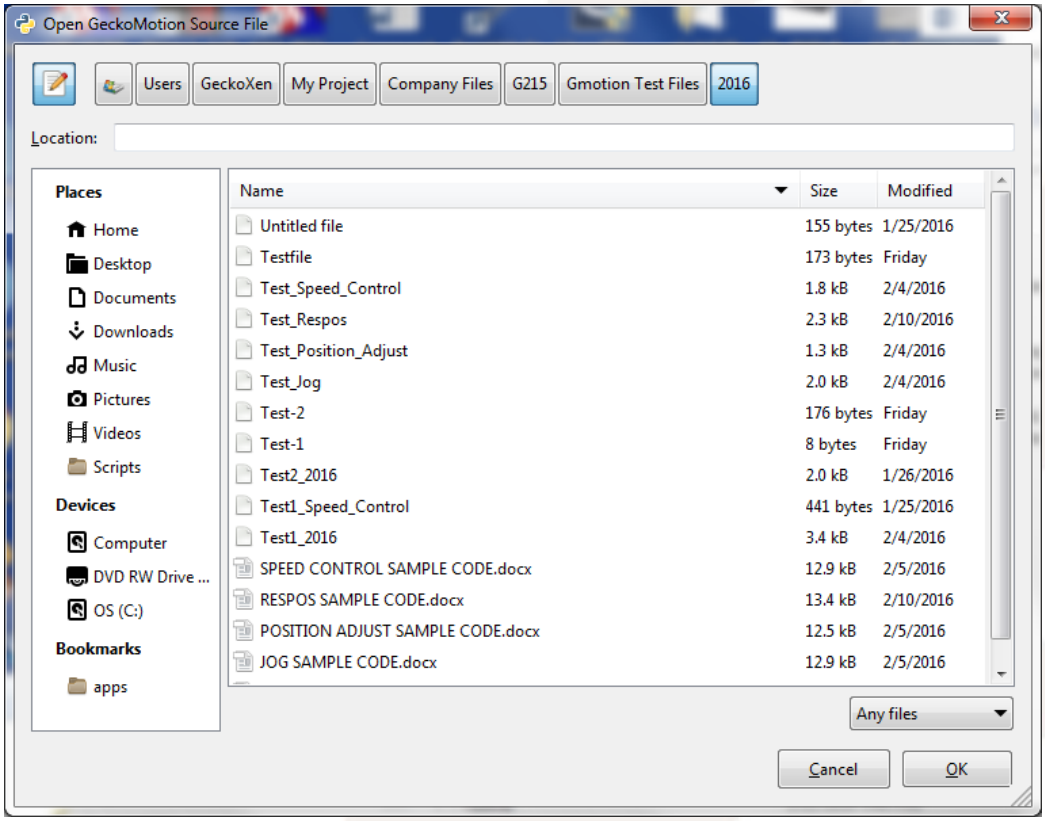


Figure 13: How to open a file.

5) **Run a program**:

Use the program in Figure 11 for this section.

5.1 Compile a program:

To run a program, user needs to compile it first.

- Open the file (see section 4.4).
- Click at file name to open the file. File name should be added to the right of **Settings** tab in GeckoMotion window.
- Click Compile button (under **Source** tab). If Compile button is greyed out (disabled), user should make a dummy change on the program to make Compile button clickable (enabled). A dummy change is an edit that has no effect on the program (add a character and then delete it).
  If the compilation is OK, user will see "**Compile**, **Assembling**, **Listing**" printed on Python window, otherwise, an error message will be printed. Make sure the compilation is OK before going to the next steps.

5.2 Run a program:

- Double check the connection by clicking at Connect button (under **Device** tap). If the connection is good, **1[READY]** will be displayed (under Connect button), and the following message will be displayed on Python window:
  Detected axis 0 = X
- Set Poll time: In GeckoMotion menu, click View and select **Log** to open GeckoMotion Log window (Figure 14).
  In GeckoMotion Log, check (yes) "**Poll every"** checkbox and set poll time to 100ms by using **+** or **–** buttons.
  Poll time can be changed per different application. In this example, it is 100ms.
- In GeckoMotion window, click EStop button (under **Device** tap) to reset program counter to the beginning (line 1) of the program. Line 1 of the program should be highlighted: x configure: 0.5 amps, idle at 71% after 2.5 seconds
- Click Run button (under **Control** tap) to run the program. Click Stop button to stop.

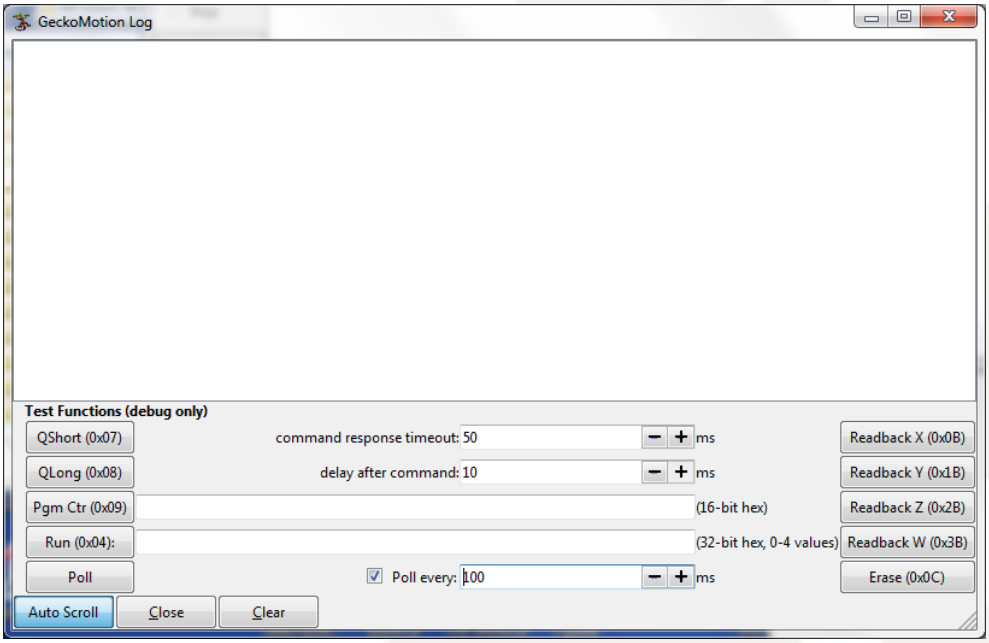


Figure 14: GeckoMotion Log window.

5.3 Step a program:

- Double check the connection by clicking at Connect button (under **Device** tap). If the connection is good, **1[READY]** will be displayed (under Connect button), and the following messages will be displayed on Python window:
  Detected axis 0 = X
- Disable Polling: In GeckoMotion menu, click View and select **Log** to open GeckoMotion Log window (Figure 14).
  In GeckoMotion Log window, uncheck (no) "**Poll every"** checkbox.
- Click **Settings** in GeckoMotion window and set Log all communications to **ON.** This will enable the feedback and print all sent commands and its feedback from GM215 on Python window.
- Click Estop button to reset program counter to the beginning of the program. Line 1 of the program should be highlighted: x configure: 0.5 amps, idle at 71% after 2.5 seconds
- Click Step or Next button (under **Control** tap) to execute the program one step at a time. This method should be used for troubleshooting purposes. See Figure 15 for more details.

  Example program content for this test:

```
1        x configure: 0.5 amps, idle at 71% after 2.5 seconds
2        x limit cw 20000
3        x acceleration 32
4        x velocity 2000
5    loop:
6        x +1000
7        x -1000
8        Goto loop
```

  Note: after executing a move command, such as x +1000 (line 6-7), user needs to click "**QLong (ox08)"** button in GeckoMotion Log window to advance to the next program line. **QLong** will feedback busy/ready status for the motor. It will advance to the next program line if it is not busy (the motor has stopped).
- Click Estop button to exit and reset program counter to the beginning of the program.

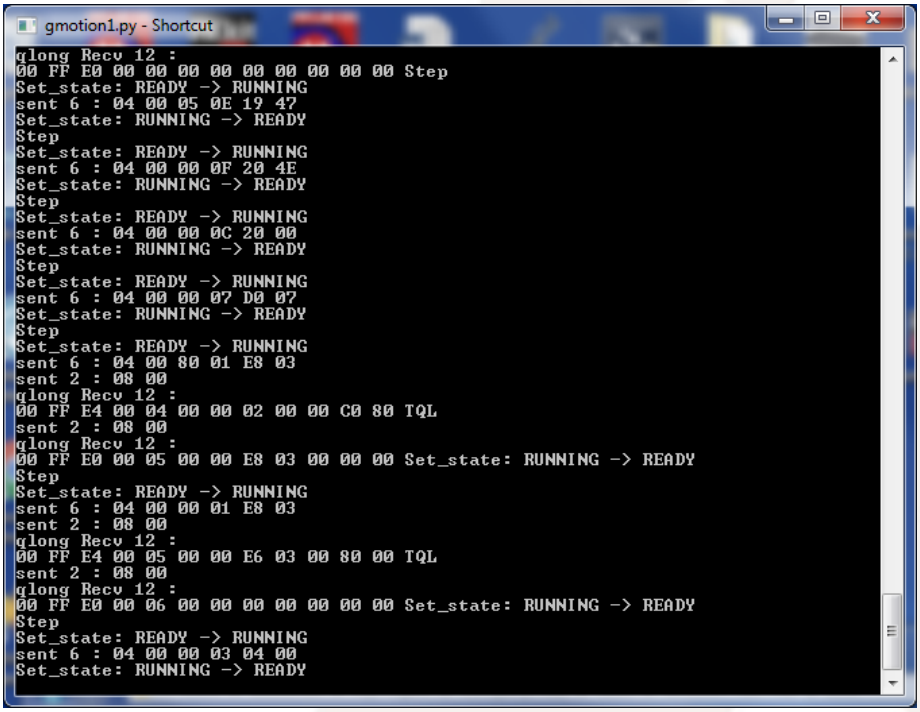


Figure 15: Python window displays Step control of the above program.

6) **Flash a program**:

After successfully compiled, the program can be downloaded to the flash memory on the board. Make sure the connection is still good (**1[READY]** is still displayed under Connect button).

6.1 Program Flash memory:

To program a flash memory, do the followings:

- Click Estop button to stop and reset the program counter.
- Disable Polling: In GeckoMotion Log, uncheck (no) "**Poll every**" checkbox.
- Click **Settings** in GeckoMotion window and set Log all communications to **ON.** This will enable the feedback and print all sent commands and its feedback from GM215 on Python window.
- Click Flash button to download the program to the flash memory of the GM215. The following message will be displayed:

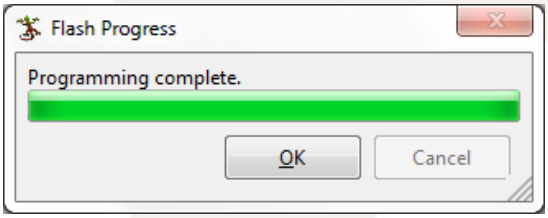


Figure 16: Flash Progress window

Program content will be printed on Python window with "Programming complete" message at the end.

6.2 Read Flash memory:

With the same settings as in 6.1, user can read the content of the flash memory from each axis by clicking at Readback X, or Readback Y, or Readback Z, or Readback W button in the GeckoMotion Log window. Flash content will be printed on Python window.

7) **Device Status**:

User can view device status such as inputs, outputs, motor position, velocity, ready/busy status by opening the GM215 Device Status window.
In GeckoMotion menu, click View and select **Status** to open GM215 Device Status window as below:

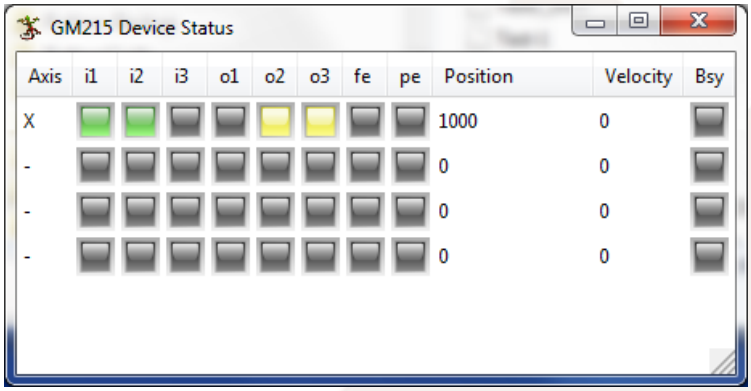


Figure 17: Device Status window

In this example, X axis has its input 1 and 2 are on, output 2 and 3 are on, motor position is 1000, not busy.

8) **Input Simulation**:

User can simulate an input by opening Input Simulation window and click at the corresponding input of an axis to toggle on/off that input.
In GeckoMotion menu, click View and select **Input overrides** to open Input Simulation window as below:

Figure 18: Input Simulation window

In this example, X axis has its input 1 & 2 are turned on.

9) **Assembler Listing and Communication Protocol**:

Assembler listing contains binary and source line of the program. The binary is the hex numbers that will be used to transmit to the GM215 via RS485 after the modification.
As in Figure 19, the first line of binary listing of the program is 0E05 4719 (x configure: 0.5 amps, idle at 71% after 2.5 seconds). This binary number has 2 words that will be modified like this:

- Reverse data bytes in each word:
  0E05 4719 => 050E 1947 (can be written as 05 0E 19 47)
- Add prefix 04 00 in front of the reversed command line:
  04 00 05 0E 19 47

This line can be sent to the GM215 to execute commands "x configure: 0.5 amps, idle at 71% after 2.5 seconds".
User should add 1ms delay after each line to avoid buffer overflow if the program is longer than 32 bytes.

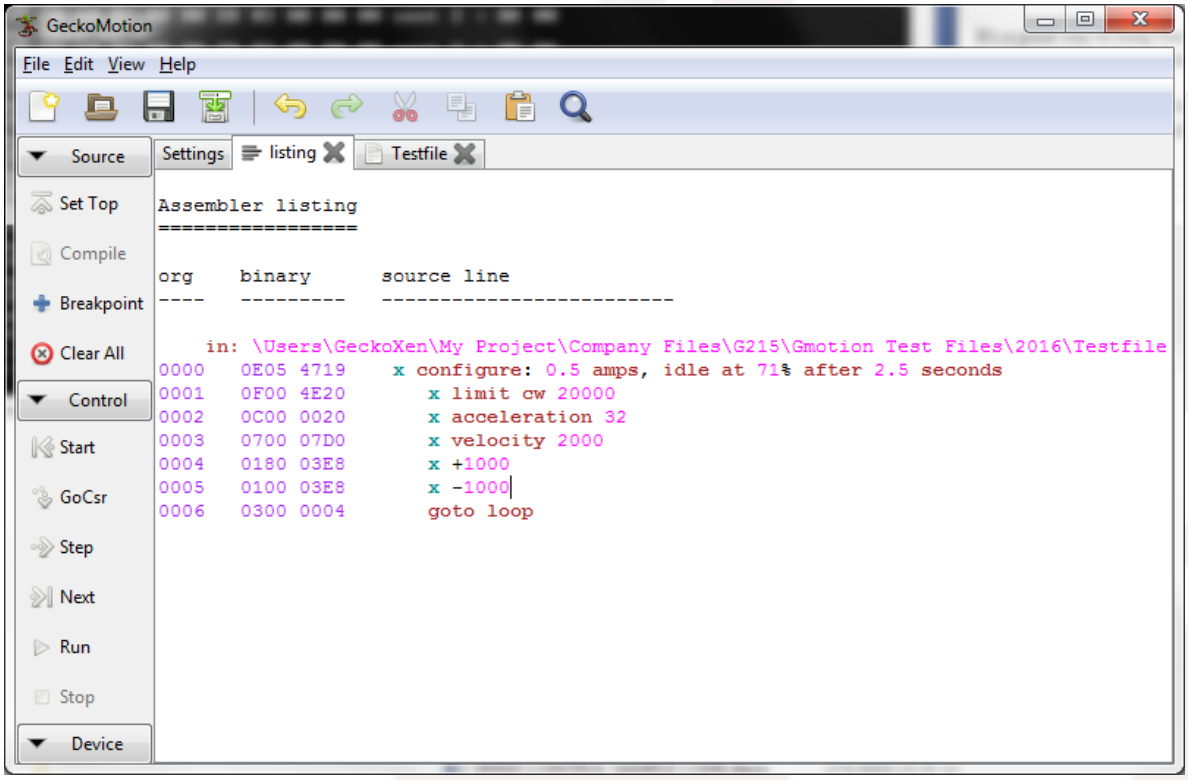


Figure 19: Assembler Listing of a program.

Binary	RS485 data	Source line	
0E05 4719	04 00 05 0E 19 47	x configure: 0.5 amps, idle at 71% after 2.5 seconds	
0F00 4E20	04 00 00 0F 20 4E	x limit cw 20000	
0C00 0020	04 00 00 0C 20 00	x acceleration 32	
0700 07D0	04 00 00 07 D0 07	x velocity 2000	
0180 03E8	04 00 80 01 E8 03	x +1000	Motion command: should wait for busy cleared
Extra Command	08 00	Query Long	Looping and check for busy bit cleared?
0100 03E8	04 00 00 01 E8 03	x -1000	Motion command: should wait for busy cleared
Extra Command	08 00	Query Long	Looping and check for busy bit cleared?
0300 0004	04 00 00 03 04 00	goto loop	

10) **Using other Serial Communication Programs (not GeckoMotion)**:

All other serial communication programs such as Hyperterminal, RealTerm, Serial Port Unitlity, etc., or user's own program can be used to control the GM215 with the following setup and protocol:

- Setup:
  - Set Port to available COM port for this RS485 interface.
  - Set baudrate to 115200, 8 bits, no parity, 1 stop bit, no flow control.
- Protocol:
  - Get program hex file from Appendix C or from GeckoMotion binary listing (see Figure 20).

For ease of operation, user can write a program using GeckoMotion, compile and then extract data from its binary listing (see Figure 19). Otherwise, user needs to calculate command data from Appendix C. The result is the program hex file in figure 20.

For example, with these commands:

x velocity 1000

x +5000

Calculated command data (from Appendix C) will be 0700 03E8 and 0180 1388.

x velocity 1000: 0700 03E8 4 bytes of memory: High word= 0700 (VELOCITY command)
Low word = 03E8 (1000=03E8 Hex)
x +5000: 0180 1388 4 bytes of memory: High word = 0180 (MOVE command)
Low word = 1388 (5000=1388 Hex)

There are 2 words for each command.

- Reverse data bytes in each word like this:
  0700 03E8 => 0007 E803 (can be written as 00 07 E8 03)
  0180 1388 => 8001 8813 (can be written as 80 01 88 13)
- Add prefix 04 00 in front of every reversed command line:
  04 00 00 07 E8 03 (for x velocity 1000)
  04 00 80 01 88 13 (for x +5000)

These 2 lines can be sent to the GM215 to execute commands "x velocity 1000" and "x +5000". User should add 1ms delay after each line to avoid buffer overflow if the program is longer than 32 bytes.

Per this conversion, the program in figure 19 will become program hex file in figure 20. This hex file can be executed by the GM215:

```
04 00 05 0E 19 47
04 00 00 0F 20 4E
04 00 00 0C 20 00
04 00 00 07 D0 07
04 00 80 01 E8 03 (1)
04 00 00 01 E8 03 (2)
04 00 00 03 04 00
```

Figure 20: Program hex file

Note: (1) and (2) are motion commands (x +1000 and x -1000). After the GM215 executes these commands, its busy bit will be set. When the motor stops, its busy bit will be cleared. User should wait for busy bit cleared before sending the next command. Busy bit can be polled via query feedback. Query Long command (08 00) should be sent continuously in a loop until its busy bit is cleared (see QUERY_SHORT and QUERY_LONG commands in Appendix B)
For test purpose, a delay time can be used after each motion command instead of sending query request.

- Example: Using Serial Port Utility program (see Figure 21):
  In this example, the first 5 lines of the program hex file (in Figure 20) will be sent. The fifth line is the move command (x+1000). The motor should move CW after these commands were sent.
  Procedure:
  - Download and run Serial Port Utility program.
  - Set Serial Port Setting as described above (COM11 is for this example only).
  - Set Receive Setting and Send Setting to **Hex**
  - Make sure **COM11 OPENED, 115200, 8, NONE, 1, OFF** displayed at the bottom (COM11 is for this example only).
  - Copy and paste the first 5 lines of program hex file onto "Send window" as in figure 21. Add 08 00 for query long request at the end. Command string will be like this:
    04 00 05 0E 19 47 04 00 00 0F 20 4E 04 00 00 0C 20 00 04 00 00 07 D0 07 04 00 80 01 E8 03 08 00
  - Click **Send** button to send this string.

  Query feedback will be displayed on the top window ("Receive window") like this:
  00 FF E4 00 04 00 00 00 EE 05 60 80 (12 bytes)
  The first 2 bytes are not used. The third byte (E4) contains busy bit (bit 2) which is set in this case (it is busy because the motor is running).
  If query request (08 00) is sent after the motor stops, user will receive query feedback like this:
  00 FF E0 00 05 00 00 E8 F1 05 00 00 (12 bytes)
  E0: Busy bit is cleared (bit 2 = 0)

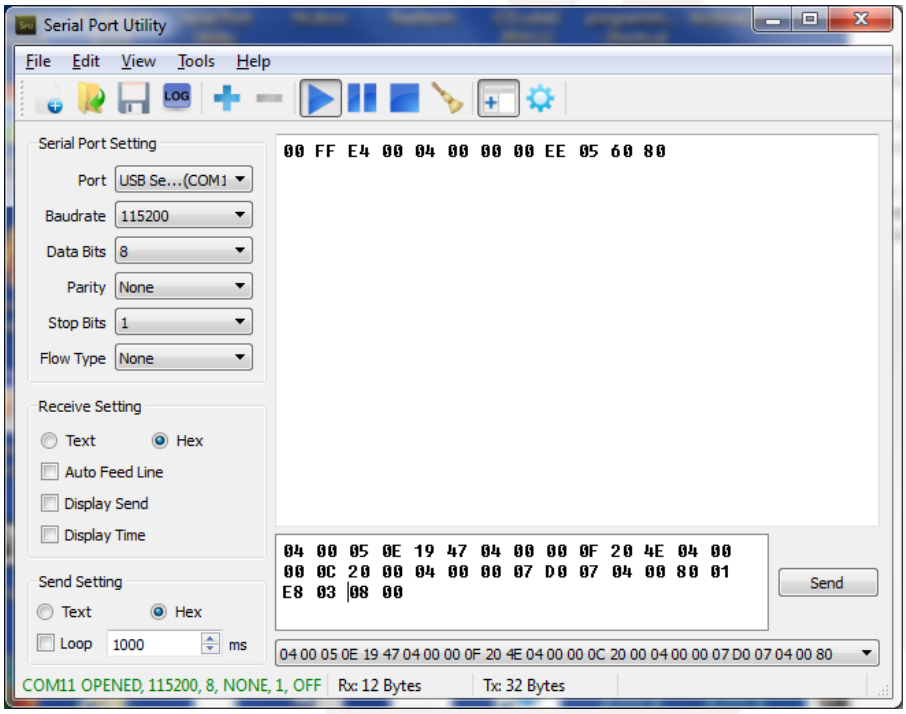


Figure 21: Using Serial Port Utility for motor control.

APPENDIX:

This section gives a brief description of the GM215 motion controller is organized and describes how to convert a text based command into the 2-word binary format command used by the GM215. The GM215 can have up to 32 commands in its command-set. Presently there are about 25 defined commands which leaves expansion room for additional commands in the future.

The GM215 uses a 4 chip-set to perform its motion control functions. They are:

1) A Microchip PIC24F16KL402 MCU for data processing, 4 channel ADC conversion, 2 SPI interfaces, UART and bootloader
2) An Actel ProAsic3 A3P030 FPGA for motor drive logic, step frequency generator, time base, interrupt generator and I/O
3) A 25PE20 SPI interface 2MB flash memory for non-volatile user program storage
4) A 3072 RS-485 transceiver

The GM215 can operate as a standard STEP/DIRECTION input microstep motor drive or as a motion-control enabled microstep motor drive. The motion controller has 2 modes of operation; EDIT MODE or RUN MODE. The EDIT mode requires PC USB connection to an RS-485 'Smart Cable' and GeckoMotion.exe GUI running on the PC.

APPENDIX A: RUN MODE

In the RUN MODE the drives read commands from non-volatile flash memory and execute the commands without a PC connection. The RS-485 connection must be used when two or more drives are expected to coordinate their activity. It isn't necessary if only a single drive is used or if multiple drives have completely independent tasks.

MASTER AND SLAVE DRIVE DESIGNATION:
The GM215 is designed to operate as multiple-axis motion controller when two or more drives (maximum of 4 drives) are connected via the RS-485 interface. Each drive reserves two DIP switch settings (SW3 and SW4) to set the drive's axis name (X, Y, Z or W) and one drive must be named 'X' designating it as the MASTER axis while the other drives are set as SLAVE axis (Y, Z or W).

PHASE-LOCK SLAVES TO MASTER:
The MASTER drive's controller runs from its crystal controlled oscillator while the SLAVE drives run from their tunable (+/- 2%) oscillators. The MASTER drive transmits a SYNC pulse over the RS-485 interface which is used by the SLAVE drives to phase-lock their clock oscillators to the MASTER drive's crystal controlled oscillator frequency. This insures all SLAVE drives synchronize themselves to the MASTER drive's clock and execute the user's application program exactly at the same place and time as the MASTER drive.

ALL DRIVES USE THE SAME USER PROGRAM:
The flash memories of all drives are programmed identically. Many commands such as MOVE has coordinates that apply only to a specific drive. The drive named X will only process the X coordinates for its motor while the Y drive will only process the Y coordinates and a Z drive will ignore X, Y coordinates for its motor altogether. All drives process the data for the commanded axis motions but only output to the motors the data that matches their name. Other commands such as WAIT are global and apply to all axis. All drives process this command simultaneously and all finish at the same time.

The advantage is the same program can be flashed to all the drives simultaneously which saves time, only a single program has to be written, debugged and maintained and the drives become interchangeable by just renaming them.

ALL DRIVES PASS INFORMATION TO EACH OTHER:
Some commands like HOME for multiple axis can take an undetermined amount of time to finish and the next command cannot start until all drives finish the HOME routine. Each connected drive in turn transmits its STATUS to all the other drives, in this case it's each drive's BUSY/READY status. The BUSY/READY status indicates if a drive is executing a command or if the command is finished. When all drives transmit a READY then the next command can be executed.

Each drive has 3 opto-isolated inputs and 3 opto-isolated outputs. Normally switches are connected to the inputs and all drives need to know the state of its own inputs and the inputs of all the connected drives. This is because the IF-THEN-ELSE command may be used to change the program flow based on inputs. All drives have to make the same program flow decision. The transmitted STATUS from each drive provides this information. Finally, the STATUS also informs all drives of an ERROR condition in any drive.

THE USER PROGRAM MUST LOOP-BACK ON ITSELF:
The GM215 drive or drives are attached to a mechanism that is driven by a motor or motors. In most cases the mechanism

performs a repetitive sequence of events that has a beginning and an end. At the end of this sequence the user program must loop back to the beginning to repeat the cycle. This may an IF-THEN-ELSE command that polls a 'start' switch connected a drive's input.

APPENDIX B: EDIT MODE

The EDIT MODE is used for:

1) Writing, testing and debugging a user program
2) Programming the flash memory with the user program
3) Loading GM215 firmware updates.

In the EDIT mode all drives switch to their crystal oscillators and the host PC UART becomes the master. As master, only the host can initiate transmissions over the RS-485 bus. The host can query the drives and the queried drive will reply with an 8-byte data block containing current information about the drive. The default UART settings are 115,200 Baud, 8-bit date, no parity, 1 stop bit.

The host PC communicates with the drives using SPECIAL COMMANDS in the EDIT mode. These are distinct from the user program commands used in the RUN mode. These are:

0) E_STOP	0x0000	Stops command execution immediately
1) STOP	0x0001	Stops when current command is finished
2) PAUSE	0x0002	Decelerates all axis to a stop
3) RESUME	0x0003	Resumes a command from PAUSE
4) RUN	0x0004	Execute the specified command line and stop
5) PROGRAM_FLASH_ROM	0x0005	Flash the entire memory from a file
6) UPDATE_FIRMWARE	0x0006	Allows updating the PIC firmware from a file
7) QUERY_SHORT	0x0007	Returns all attached axis status and X axis program counter only
8) QUERY_LONG	0x0008	Returns all attached axis full information.
9) LOAD_PGM_COUNTER	0x0009	Loads the drive's program counter with a base value
A) TBD	0x000A	To be determined
B) READ_BACK	0x000B	Read back from flash memory
C) BULK_ERASE	0x000C	Erase the whole flash memory
D) TEST_IN	0x000D	Software simulate input
E) VERSION	0x000E	PIC and FPGA firmware version number

E_STOP and **STOP** **EDIT MODE COMMAND**

Syntax: 0x0000 for E_STOP
0x0001 for STOP

Description:
The E_STOP command will immediately shut-off the motors without deceleration and turn-off the motor current. Send: 0x0000
The STOP command will terminate user command execution after the current command finishes. Send: 0x0001

PAUSE **EDIT MODE COMMAND**

Syntax: 0x0002 for PAUSE

Description: Immediately decelerates all attached axis to a stop at their programmed rates of acceleration. This command can be sent at any time, even when axis are in motion moving towards their programmed destination coordinates.

RESUME **EDIT MODE COMMAND**

Syntax: 0x0003 for RESUME

Description: RESUME accelerates all attached axis to their previous velocities at their programmed rates of acceleration. This command can be sent at any time, all axis will resume moving towards their original destinations.

RUN **EDIT MODE COMMAND**

Syntax: 0x0004 for RUN followed by USER COMMAND

Description:
RUN_COMMAND_LINE will execute a single user command sent from the host GUI and then stop. All attached drives read the sent command, if the command names drives then only those drives will execute the command and the unnamed drives will ignore it. The sent user command is not stored to the attached GM215 flash memories.

The ADDRESS is used by the guest axis to calculate the next user command address. If executing the user command doesn't change program flow then the calculated address will be the host-sent ADDRESS + 1. Program flow user commands use the host-sent ADDRESS
to calculate relative next user command addresses; absolute addresses decode directly from the sent user command and don't use the host-sent ADDRESS.

All attached drives need to know all other drives' status to properly execute program flow commands. The Busy/Ready flags of all the involved drives indicate Ready when user command is completed.

Example 1:
A possible host RUN command line is **IF Z IN 2 OFF GOTO <label>**. To execute this command line (assume label = 614), send:

0x0004	the host UART sends 0x04, 0x00	this is the special command RUN
0x0266	the host UART sends 0x66, 0x02	this is the lower word for the user command in hex.
0x85A2	the host UART sends 0xA2, 0x85	this is the upper word for the user command in hex.

The guest axis sends a SHORT_QUERY reply to the host:

X STATUS low byte, then X STATUS high byte,
X PC ADDRESS low byte, then X PC ADDRESS high byte,
Y STATUS low byte, then Y STATUS high byte,
Z STATUS low byte, then Z STATUS high byte,
W STATUS low byte, then W STATUS high byte,

See the SHORT_QUERY reply format on page 60.

The host needs to reply when a sent user command is completed. The sequence is:

1) Host waits until it receives the entire reply.
2) Host sends a SORT_QUERY or LONG_QUERY command. Guests reply accordingly.
3) Host waits until it receives the entire reply.
4) Host inspects message for 'Ready' on all involved guest axis.
5) If all aren't 'Ready', host goes to step (2).
6) All are 'Ready'. The reply PC ADDRESS is the next user command address.

Example 2:
The host wishes to RUN a user command **X+4000, Y-3000, W 5000**. The host sends:

0x0004		RUN command
0x0FA0	0x2180	User command **X+4000,**
0x0BB8	0x6100	User command **Y-3000,**
0x1388	0xC000	User command W **5000**

PROGRAM_FLASH_ROM **EDIT MODE COMMAND**

Syntax: 0x0005 for PROGRAM_FLASH_ROM

Description: After successfully compiled, user can download the program to the flash memory on board. The PC should send 0x0005 first and then start to send the data stream. See Appendix C or contact Geckodrive for more details.

UPDATE_FIRMWARE **EDIT MODE COMMAND**

See Appendix F for details.

QUERY_SHORT **EDIT MODE COMMAND**
QUERY_LONG **EDIT MODE COMMAND**

Syntax: 0x0007 for QUERY_SHORT
0x0008 for QUERY_LONG

Description:
The QUERY_LONG command updates each axis' inputs, outputs, position, velocity, errors and Busy/Ready status. This can be used as a near real-time update if the host sends the LONG_QUERY command repeatedly. Each axis will immediately reply with a 10-byte block of data for each axis. The order of reply will be X, Y, Z and W, each 10 bytes long. If an axis is not attached or malfunctioning, it will be skipped. It is important to wait for the axis to reply before issuing another command, otherwise Tx collisions may result. The LONG_QUERY reply takes 3.48 milliseconds. The reply format is:

Byte 1: Drive status byte 1. Busy/Ready flag, input states, Error flag and axis name.
Byte 2: Drive status byte 2. Output states, group number.
Byte 3: LSB of the axis program counter (PC [7:0]).
Byte 4: MSB of the axis program counter (PC [15:8]).
Byte 5: ANALOG INPUT register (ANA [7:0]).
Byte 6: LSB of the motor position register (POS [7:0]).
Byte 7: Middle byte of the motor position register (POS [15:8]).
Byte 8: MSB of the motor position register (POS [23:16]).
Byte 9: LSB of the axis velocity (VEL [7:0]).
Byte 10: MSB of the axis velocity (VEL [15:8]).

Byte 1, 11, 21, 31	Byte 2, 12, 22, 32	Byte 3, 13, 23, 33	Byte 4, 14, 24, 34	Byte 5, 15, 25, 35
bit 7 = IN1	bit 7 = TBD	bit 7 = PC [7]	bit 7 = PC [15]	bit 7 = ANA[7]
bit 6 = IN2	bit 6 = OUT3	bit 6 = PC [6]	bit 6 = PC [14]	bit 6 = ANA[6]
bit 5 = IN3	bit 5 = OUT2	bit 5 = PC [5]	bit 5 = PC [13]	bit 5 = ANA[5]
bit 4 = ERR1	bit 4 = OUT1	bit 4 = PC [4]	bit 4 = PC [12]	bit 4 = ANA[4]
bit 3 = ERR2	bit 3 = TBD	bit 3 = PC [3]	bit 3 = PC [11]	bit 3 = ANA[3]
bit 2 = BSY/RDY	bit 2 = TBD	bit 2 = PC [2]	bit 2 = PC [10]	bit 2 = ANA[2]
bit 1 = AXIS1	bit 1 = GP1	bit 1 = PC [1]	bit 1 = PC [9]	bit 1 = ANA[1]
bit 0 = AXIS0	bit 0 = GP0	bit 0 = PC [0]	bit 0 = PC [8]	bit 0 = ANA[0]

Byte 6, 16, 26, 36	Byte 7, 17, 27, 37	Byte 8, 18, 28, 38	Byte 9, 19, 29, 39	Byte 10, 20, 30, 40
bit 7 = POS [7]	bit 7 = POS[15]	bit 7 = POS[23]	bit 7 = VEL[7]	bit 7 = VEL[15]
bit 6 = POS [6]	bit 6 = POS[14]	bit 6 = POS[22]	bit 6 = VEL[6]	bit 6 = VEL[14]
bit 5 = POS [5]	bit 5 = POS[13]	bit 5 = POS[21]	bit 5 = VEL[5]	bit 5 = VEL[13]
bit 4 = POS [4]	bit 4 = POS[12]	bit 4 = POS[20]	bit 4 = VEL[4]	bit 4 = VEL[12]
bit 3 = POS [3]	bit 3 = POS[11]	bit 3 = POS[19]	bit 3 = VEL[3]	bit 3 = VEL[11]
bit 2 = POS [2]	bit 2 = POS[10]	bit 2 = POS[18]	bit 2 = VEL[2]	bit 2 = VEL[10]
bit 1 = POS [1]	bit 1 = POS[9]	bit 1 = POS[17]	bit 1 = VEL[1]	bit 1 = VEL[9]
bit 0 = POS [0]	biit 0 = POS[8]	bit 0 = POS[16]	bit 0 = VEL[0]	bit 0 = VEL[8]

The QUERY_SHORT command requests the X axis to reply with two STATUS bytes and its two PC ADDRESS bytes. All the other attached axis to reply with just their two STATUS bytes each. The order of reply will be X, Y, Z and W. If an axis is not attached or malfunctioning, it reply time slot is empty. The QUERY_SHORT reply takes 0.87 milliseconds. The reply format is:

Byte 1=X, 5=Y, 7=Z, 9=W	Byte 2=X, 6=Y, 8=Z, 10=W	Byte 3=X	Byte 4=X
AXIS STATUS1 byte	AXIS STATUS2 byte	PC ADDRESS LOW byte	PC ADDRESS HIGH byte
bit 7 = IN1	bit 7 = TBD	bit 7 = PC [7]	bit 7 = PC [15]
bit 6 = IN2	bit 6 = OUT3	bit 6 = PC [6]	bit 6 = PC [14]
bit 5 = IN3	bit 5 = OUT2	bit 5 = PC [5]	bit 5 = PC [13]
bit 4 = ERR1	bit 4 = OUT1	bit 4 = PC [4]	bit 4 = PC [12]
bit 3 = ERR2	bit 3 = TBD	bit 3 = PC [3]	bit 3 = PC [11]
bit 2 = BSY/RDY	bit 2 = TBD	bit 2 = PC [2]	bit 2 = PC [10]
bit 1 = AXIS1	bit 1 = GP1	bit 1 = PC [1]	bit 1 = PC [9]
bit 0 = AXIS0	bit 0 = GP0	bit 0 = PC [0]	bit 0 = PC [8]

LOAD_PROGRAM_COUNTER **EDIT MODE COMMAND**

Syntax: 0x0009 for LOAD_PROGRAM_COUNTER followed by PROGRAM COUNTER VALUE

Description:
The LOAD_PROGRAM_COUNTER command initializes the GM215 program counter to a value sent by the host. This allows the host to start running the user program from any address location inside the user program instead of only from the beginning. This facilitates debugging the user program by not having to run time-consuming commands that lead up to the command that requires debugging.

This command is used in conjunction with the RUN special command. The host sends this special command followed by a 2-byte program counter value, that being the address of the subsequent RUN command. The LOAD_PROGRAM_COUNTER command is only needed when the host intends to send an out of sequence RUN command.

Example:
The host wishes to RUN an out of sequence user command **X+4000, Y-3000, W 5000**. The host sends:

0x0009		LOAD_PROGRAM_COUNTER command
0xA93C		Program address for the following user command
0x0004		RUN command
0x0FA0	0x2180	User command **X+4000,**
0x0BB8	0x6100	User command **Y-3000,**
0x1388	0xC000	User command W **5000**

When this motion command finishes execution, the X axis GM215 updates its program counter to the next user program address, in this case its 0xA93D. The next QUERY shows this update and the host uses this address to fetch the next user command.

VERSION **EDIT MODE COMMAND**

Syntax: 0x000E for VERSION

Description: the VERSION command will return firmware version for the PIC and FPGA. The first 3 numbers are PIC version, and the last number is FPGA version. In order to get version feedback, user needs to turn on “Log all communications” on GeckoMotion.

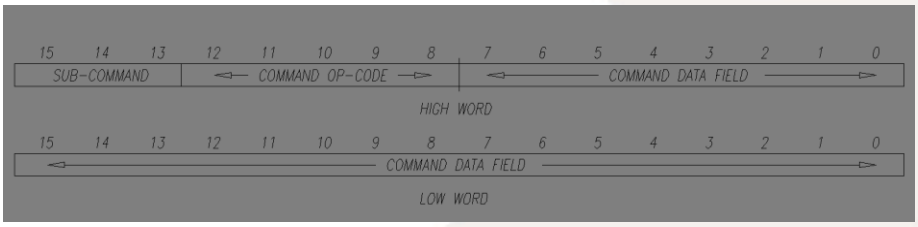
Example: If the feedback is “X axis version: 07 05 01 02”, the first 3 numbers are for PIC version: 07 05 01, and the last number is for FPGA version: 02

07 05 01: PIC version 7.5.1
02: FPGA version 2

APPENDIX C: COMMAND FORMAT

The rest of this section describes how the user commands are converted from text to binary machine language format.

COMMAND FORMAT:
2 words are used to form a command, here called HIGH WORD and LOW WORD. The upper byte of the HIGH WORD contains a 5-bit command op-code and a 3-bit sub-command. The lower 8-bits of the word is always a data field as is the entire LOW WORD.



The G215 serial flash memory is byte oriented so 1 command is equal to 4 bytes of memory. Little-endian format is used meaning the least significant byte is located at the lowest address and the most significant byte is located at the highest address. This 4-byte block constitutes a single command and the program counter increments on each 4-byte read from memory.

Some commands like MOVE can be concatenated on the same command line and up to 4 axis are supported. This means up to 16 bytes are read from memory for the command line. The command line counter must be incremented 2, 3 or 4 times for 2, 3 or 4 concatenated axis respectively to show the address of the next command. Presently the flash memory supports a maximum of 65,536 command lines. This can be expanded to 524,288 lines if a larger memory is used should the need arise.

Almost any standard format such as g-code or Gerber can be a source text file if the user wishes to write a text to binary converter. In the following command translation descriptions GeckoMotion text is used as the source.

For example, with these commands:

x velocity 1000
x +5000

In the binary listing of the Geckomotion, it will be seen as:

0700 03E8 4 bytes of memory: High word = 0700 (VELOCITY command)
Low word = 03E8 (1000=03E8 Hex)

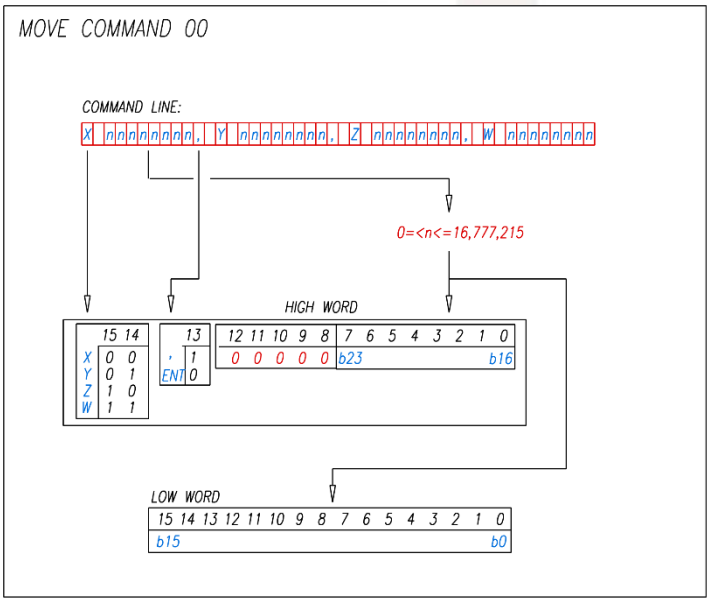
0180 1388 4 bytes of memory: High word = 0180 (MOVE command)
Low word = 1388 (5000=1388 Hex)

RS485 transmit will be: 00 07 E8 03 80 01 88 13

RS485 transmit for flashing memory will be: 05 00 00 07 E8 03 80 01 88 13 FF FF
(05 00 and FF FF are prefix and suffix for flash command)

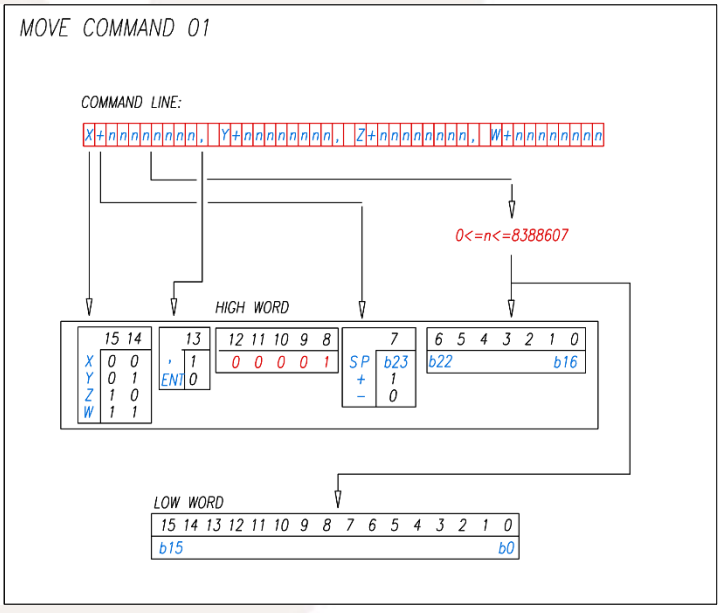
RS485 Receive for flash memory read back: 00 07 E8 03 80 01 88 13 FF FF
(0B 00 should be sent first in order to receive this read back)

MOVE **op-code 0x00** **MOTION COMMAND**



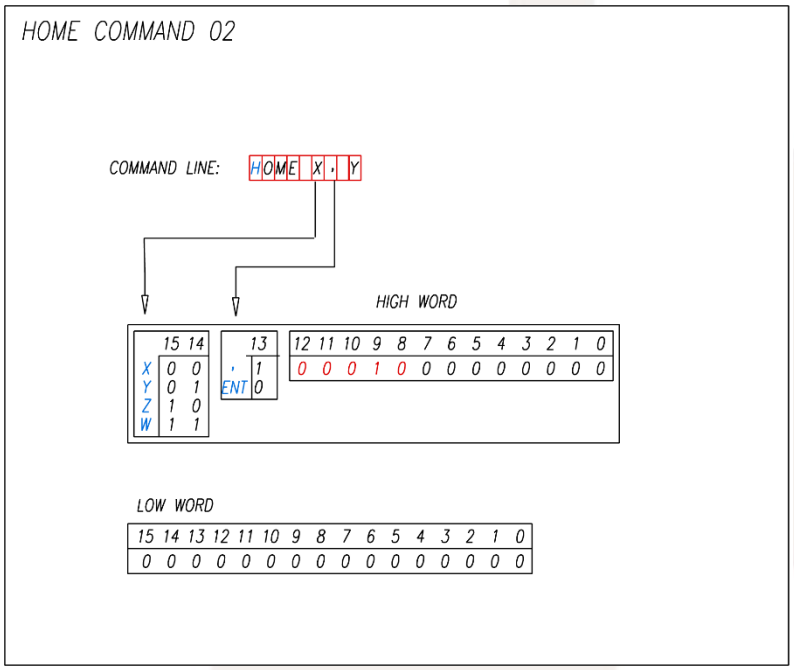
EXAMPLE: **X 255 <ENTER>** **H=0000, L=00FF** **(hex value)**
Y 65535 <ENTER> **H=4000, L=FFFF**
Z 16777215 <ENTER> **H=80FF, L=FFFF**
X 15, Y 255, Z 4095, W 65535 <ENTER> **H=2000, L=000F (X COORDINATE)**
H=6000, L=00FF (Y COORDINATE)
H=A000, L=0FFF (Z COORDINATE)
H=C000, L=FFFF (W COORDINATE)

MOVE **op-code 0x01** **MOTION COMMAND**



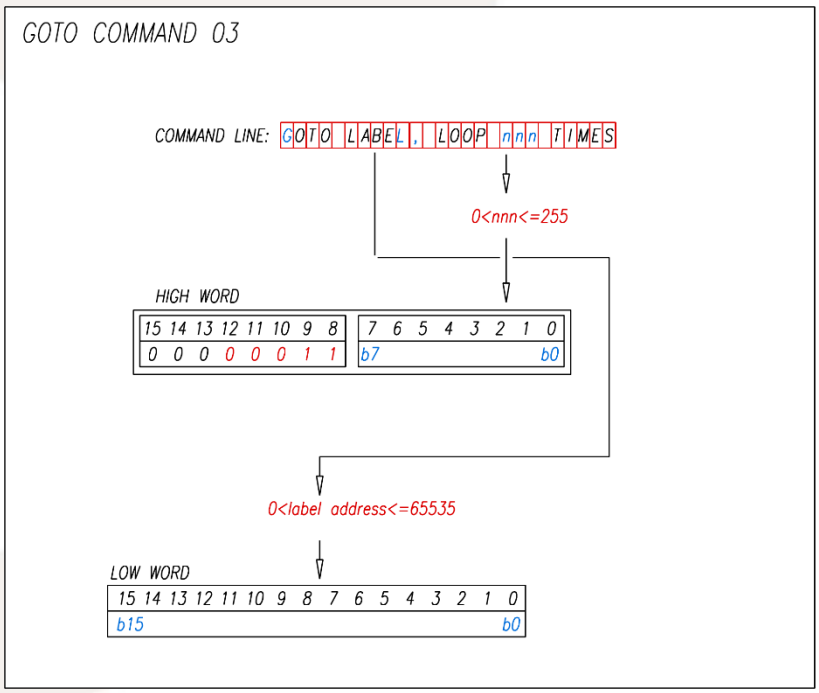
EXAMPLE: **X +255 <ENTER>** **H=0180, L=00FF**
Y -65535 <ENTER> **H=4100, L=FFFF**
Z +8388607 <ENTER> **H=80FF, L=FFFF**
X +15, Y -255, Z +4095, W -65535 <ENTER> **H=2180, L=000F (X COORDINATE)**
H=6100, L=00FF (Y COORDINATE)
H=A180, L=0FFF (Z COORDINATE)
H=C100, L=FFFF (W COORDINATE)

HOME **op-code 0x02** **MOTION COMMAND**



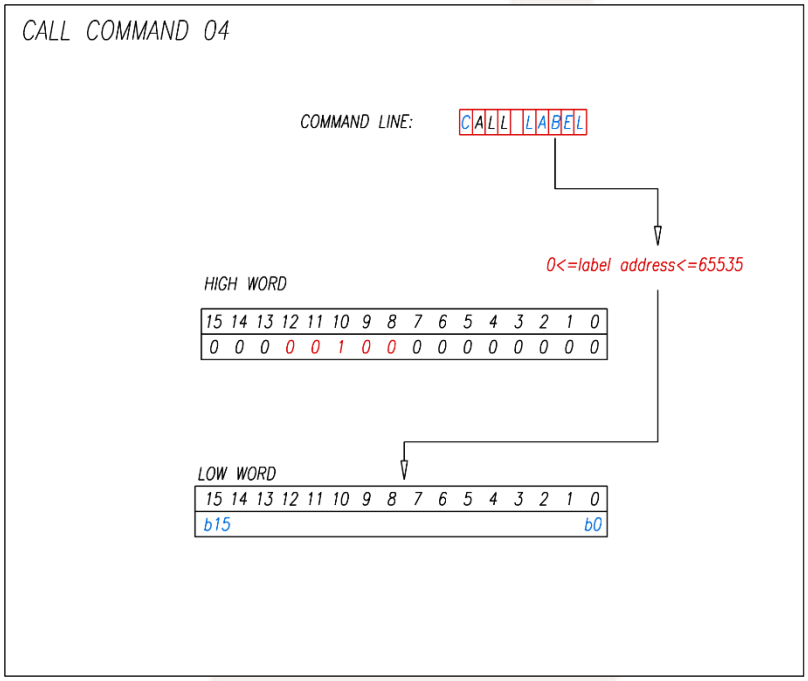
EXAMPLE: **HOME X <ENTER>** **H=0200, L=0000**
HOME X, Y, Z, W <ENTER> **H=2200, L=0000** **(X COORDINATE)**
H=6200, L=0000 **(Y COORDINATE)**
H=A200, L=0000 **(Z COORDINATE)**
H=C200, L=0000 **(W COORDINATE)**

GOTO **op-code 0x03** **PROGRAM FLOW COMMAND**



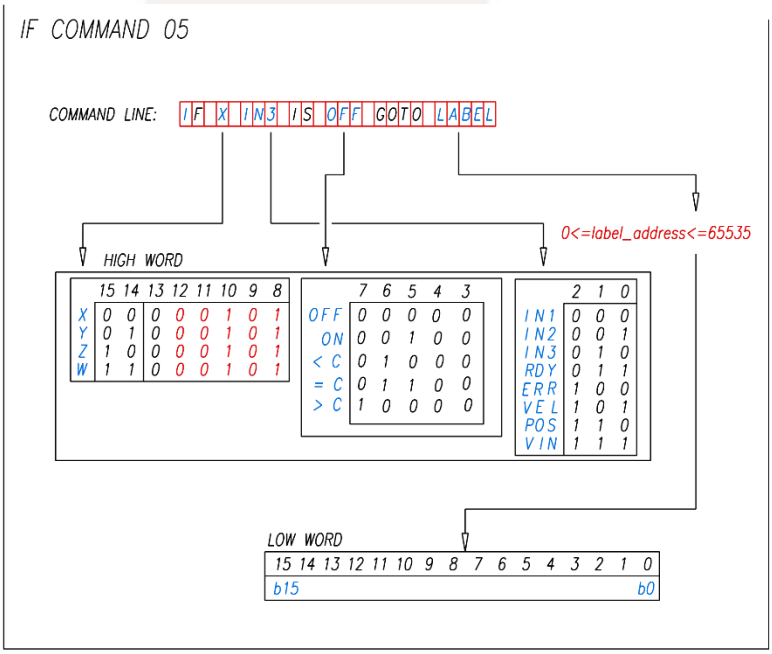
EXAMPLE: **GOTO CW_MOTION, LOOP 8 TIMES <ENTER>** **H=0308, L=000D (if CW_MOTION address is at 000D)**

CALL **op-code 0x04** **PROGRAM FLOW COMMAND**



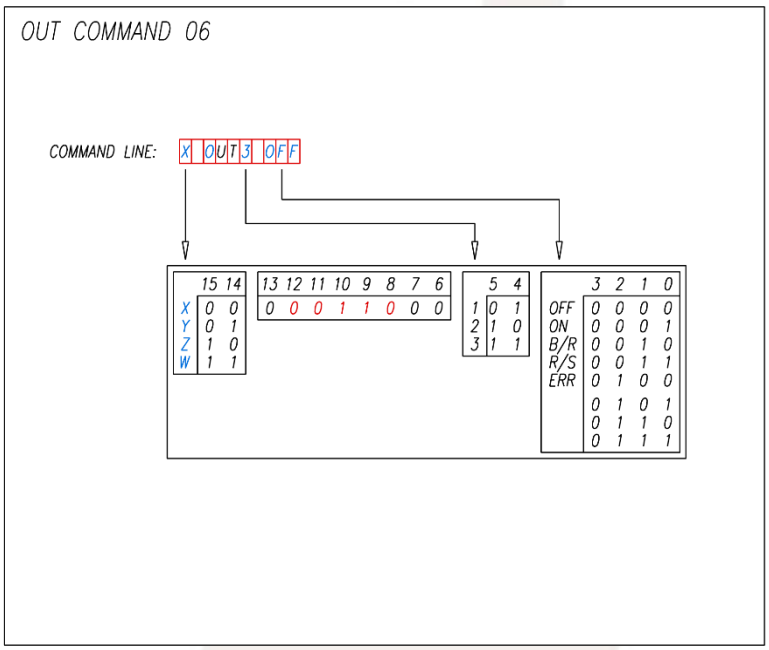
EXAMPLE: **CALL X_CONFIG <ENTER>** **H=0400, L=000C (if X_CONFIG address is at 000C)**

IF-THEN-ELSE **op-code 0x05** **PROGRAM FLOW COMMAND**



EXAMPLE:
IF X IN1 IS ON GOTO LOOP1 <ENTER> **H=0520, L=000D (if LOOP1 address is at 000D)**
IF Y RDY IS OFF GOTO LOOP1 <ENTER> **H=4503, L=000D**
IF Z ERR IS ON GOTO LOOP1 <ENTER> **H=8524, L=000D**
IF W VEL IS < GOTO LOOP1 <ENTER> **H=C545, L=000D**
IF X POS IS = GOTO LOOP1 <ENTER> **H=0566, L=000D**
IF X VIN IS > GOTO LOOP1 <ENTER> **H=0587, L=000D**

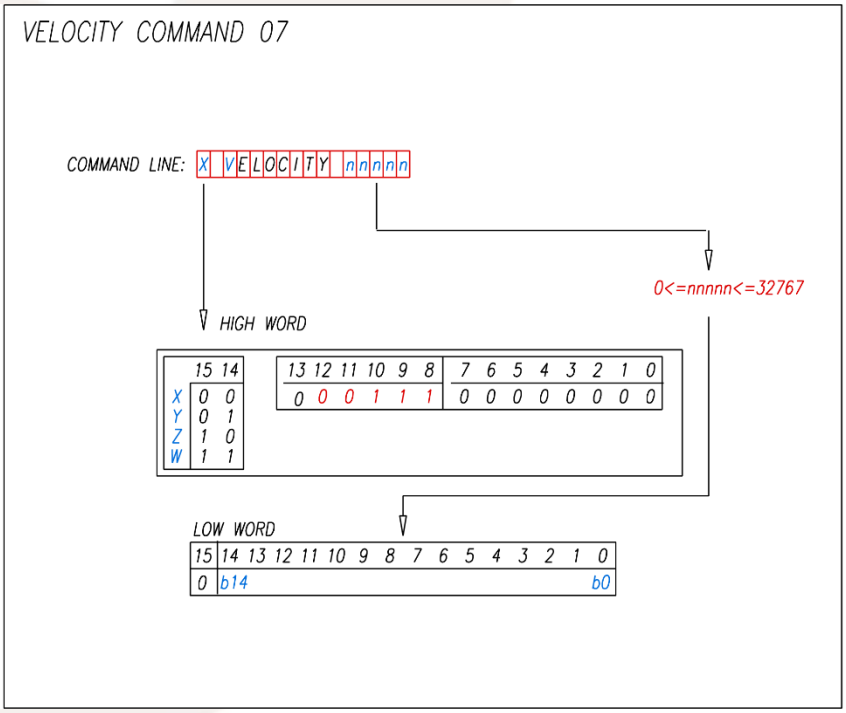
OUTPUT op-code 0x06 **MISCELLANEOUS COMMAND**



EXAMPLE:

X OUT 1 ON <ENTER>	**H=0611, L=0000**
Y OUT 2 BR <ENTER>	**H=4622, L=0000**
Z OUT 3 RS <ENTER>	**H=8633, L=0000**
W OUT 1 ER <ENTER>	**H=C614, L=0000**

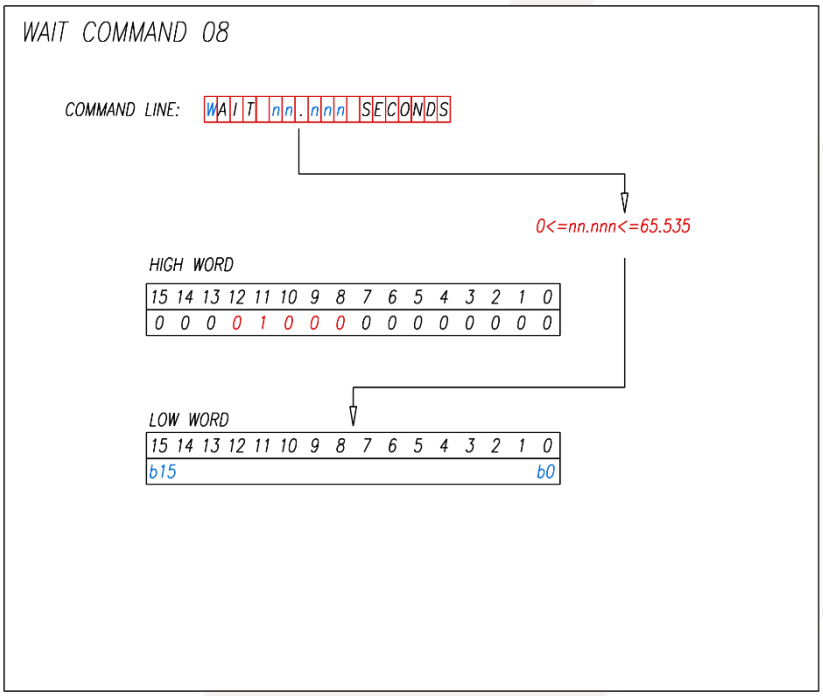
VELOCITY op-code 0x07 **CONFIGURATION COMMAND**



EXAMPLE:

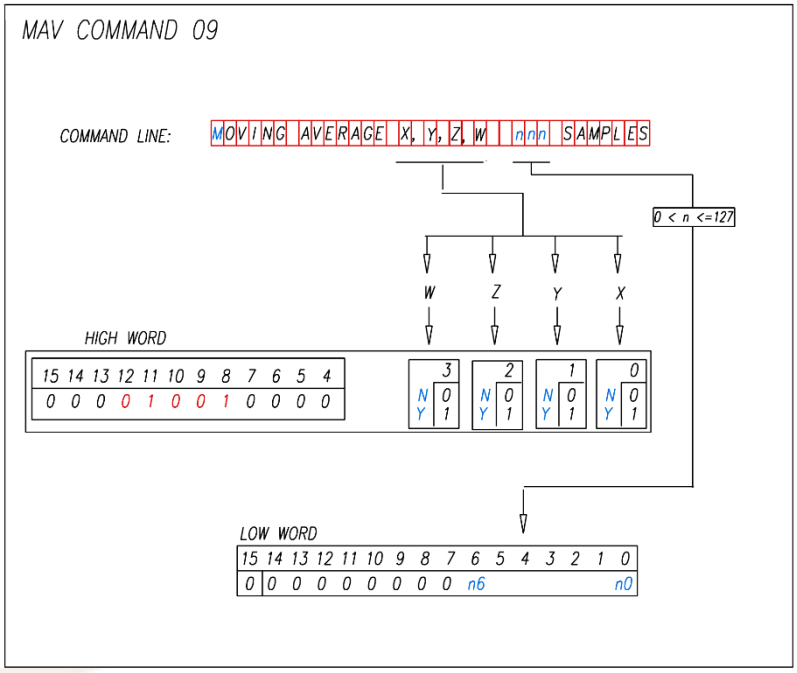
X VEL 255 <ENTER> OR X VELOCITY 255 <ENTER>	**H=0700, L= 00FF**
Y VEL 32767 <ENTER> OR X VELOCITY 255 <ENTER>	**H=4700, L= 7FFF**

WAIT **op-code 0x08** **PROGRAM FLOW COMMAND**



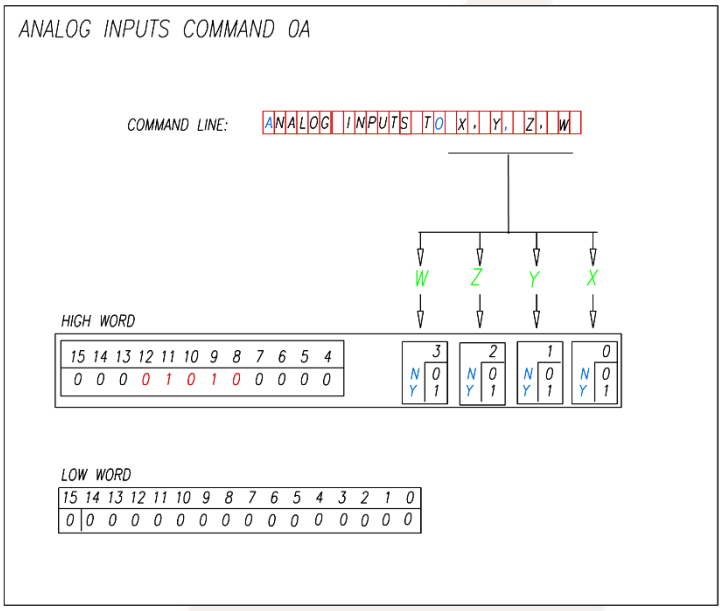
EXAMPLE:
WAIT 0.255 SECONDS <ENTER> **H=0800, L=00FF**
WAIT 1.234 SECONDS <ENTER> **H=0800, L=04D2**
WAIT 65.535 SECONDS <ENTER> **H=0800, L=FFFF**

MOVING AVERAGE FILTER **op-code 0x09** **CONFIGURATION COMMAND**



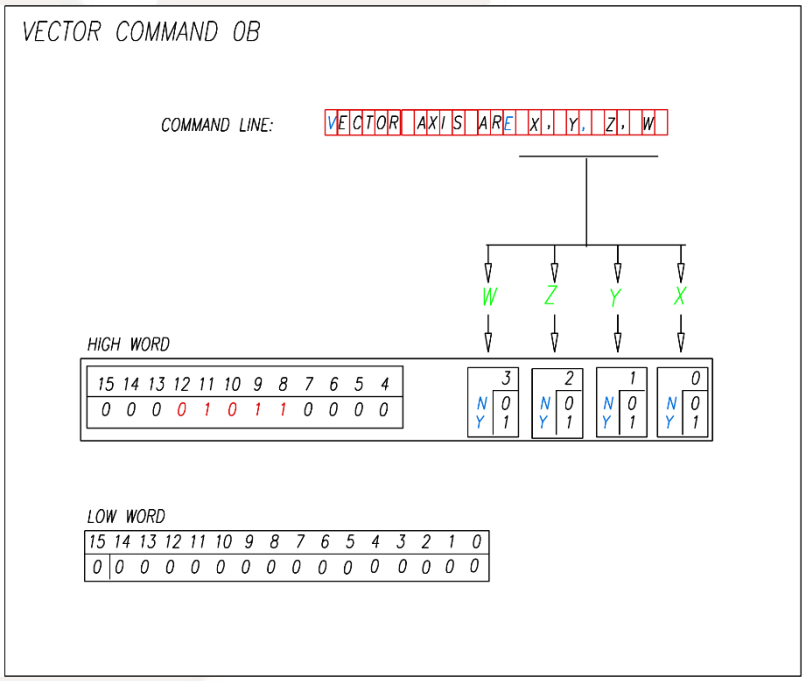
EXAMPLE: **MOVING AVERAGE X, Y, Z 100 SAMPLES <ENTER>** **H=090E, L=0064**

ANALOG INPUTS **op-code 0x0A** **CONFIGURATION COMMAND**



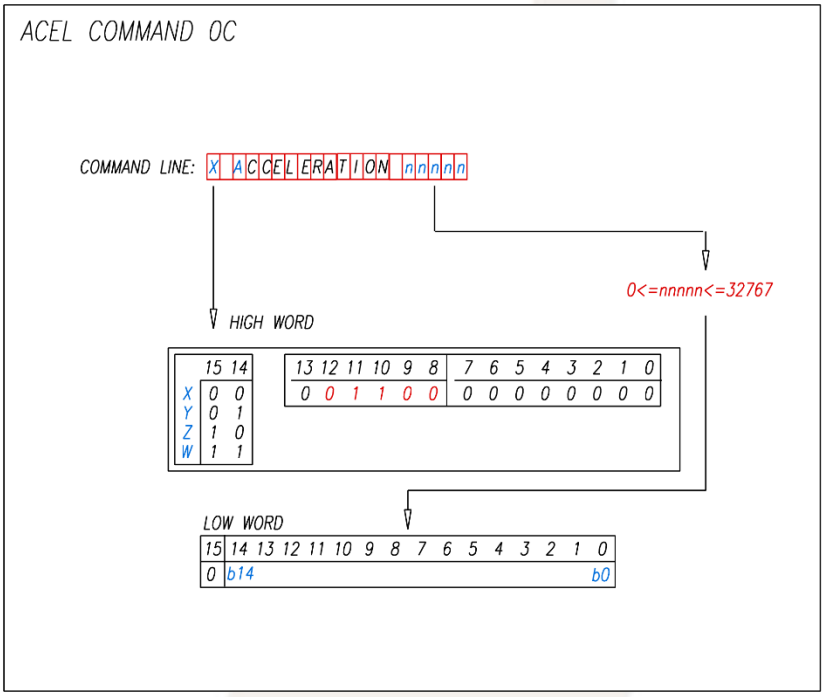
EXAMPLE:
ANALOG INPUT X <ENTER> **H=0A01, L=0000**
ANALOG INPUT X, Y <ENTER> **H=0A03, L=0000**
ANALOG INPUT X, Y, Z, W <ENTER> **H=0A0F, L=0000**

VECTOR AXIS **op-code 0x0B** **CONFIGURATION COMMAND**



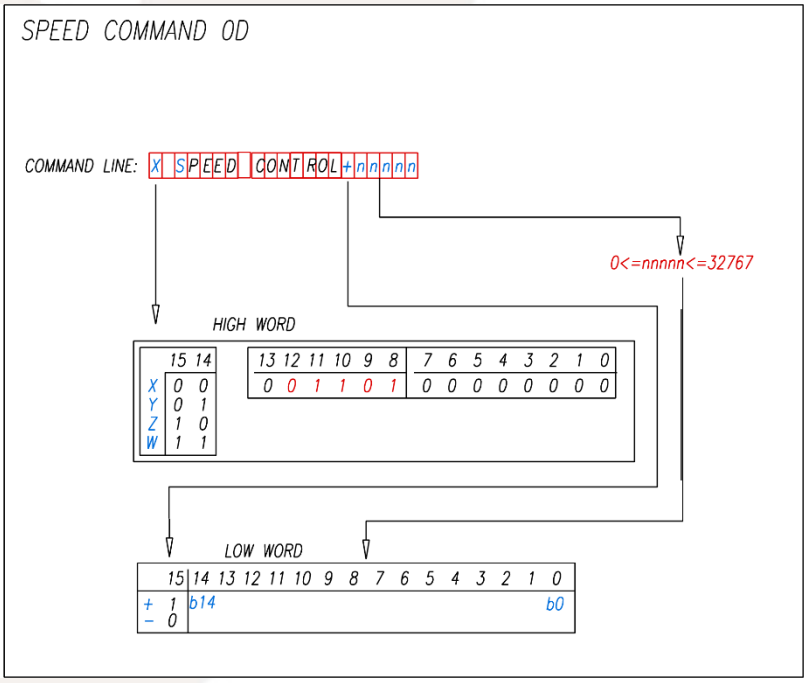
EXAMPLE:
VECTOR AXIS ARE X <ENTER> **H=0B01, L=0000**
VECTOR AXIS ARE X, Y <ENTER> **H=0B03, L=0000**
VECTOR AXIS ARE X, Y, Z, W <ENTER> **H=0B0F, L=0000**

ACCELERATION **op-code 0x0C** **CONFIGURATION COMMAND**



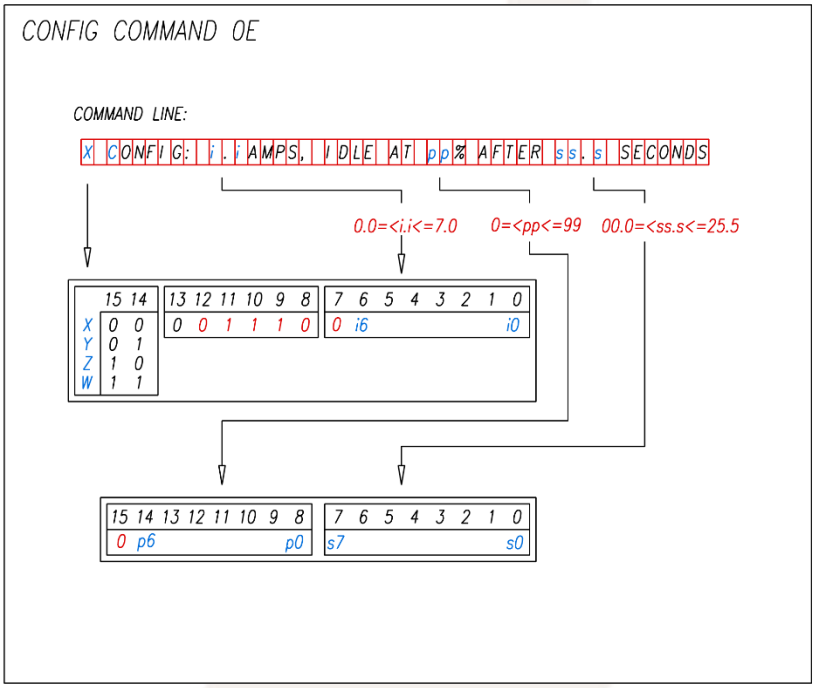
EXAMPLE: **X ACCELERATION 255 <ENTER>** **H=0C00, L=00FF**
Y ACCELERATION 32767 <ENTER> **H=4C00, L=7FFF**

SPEED CONTROL **op-code 0x0D** **MOTION COMMAND**



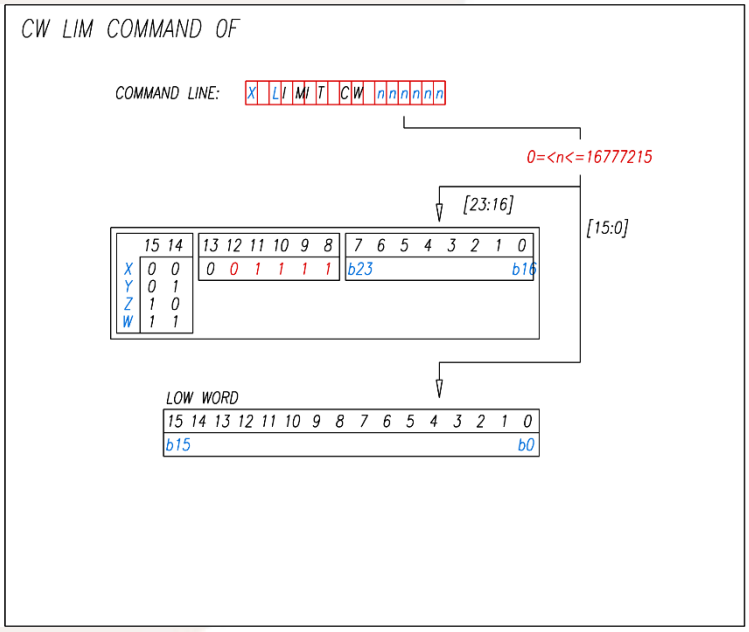
EXAMPLE: **X SPEED CONTROL 15 <ENTER>** **H=0D00, L=000F**
Y SPEED CONTROL +255 <ENTER> **H=4D00, L=00FF**
Z SPEED CONTROL +32767 <ENTER> **H=8D00, L=FFFF**
W SPEED CONTROL -32767 <ENTER> **H=CD00, L=7FFF**

CONFIGURE AXIS op-code 0x0E **CONFIGURATION COMMAND**



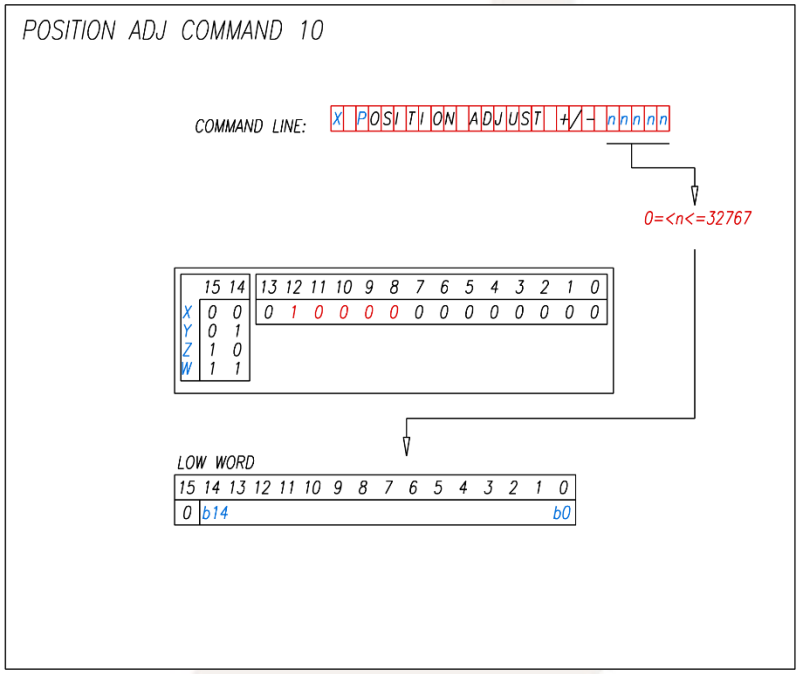
EXAMPLE: **X CONFIG: 0.1 AMPS, IDLE AT 0% AFTER 0.0 SECONDS <ENTER>** **H=0E01, L=0000**
Y CONFIG: 1.5 AMPS, IDLE AT 15% AFTER 1.5 SECONDS <ENTER> **H=4E0F, L=0F0F**
Z CONFIG: 7.0 AMPS, IDLE AT 99% AFTER 25.5 SECONDS <ENTER> **H=8E46, L=63FF**

LIMIT CWop-code 0x0F **CONFIGURATION COMMAND**



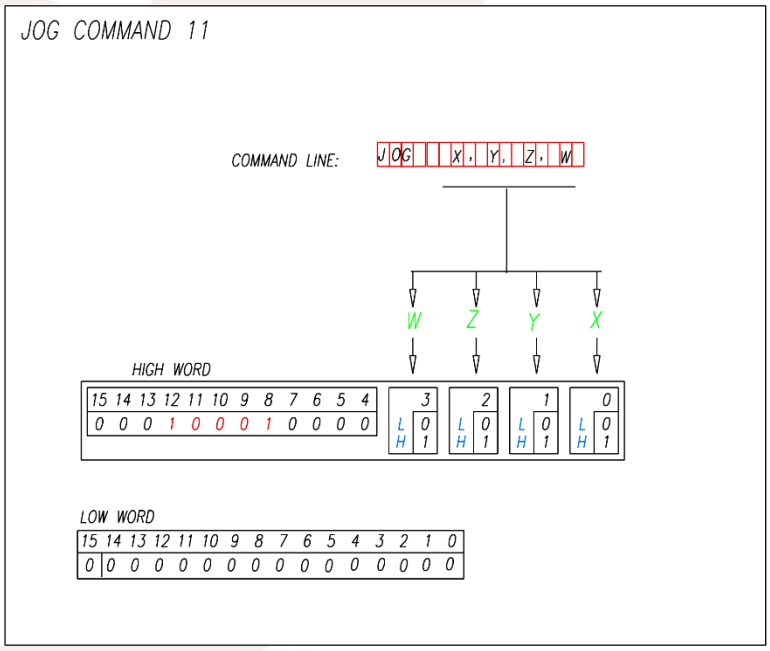
EXAMPLE: **X LIMIT CW 255 <ENTER>** **H=0F00, L=00FF**
Y LIMIT CW 65535 <ENTER> **H=4F00, L=FFFF**
Z LIMIT CW 16777215 <ENTER> **H=8FFF, L=FFFF**

POSITION ADJUST op-code 0x10 **CONFIGURATION COMMAND**



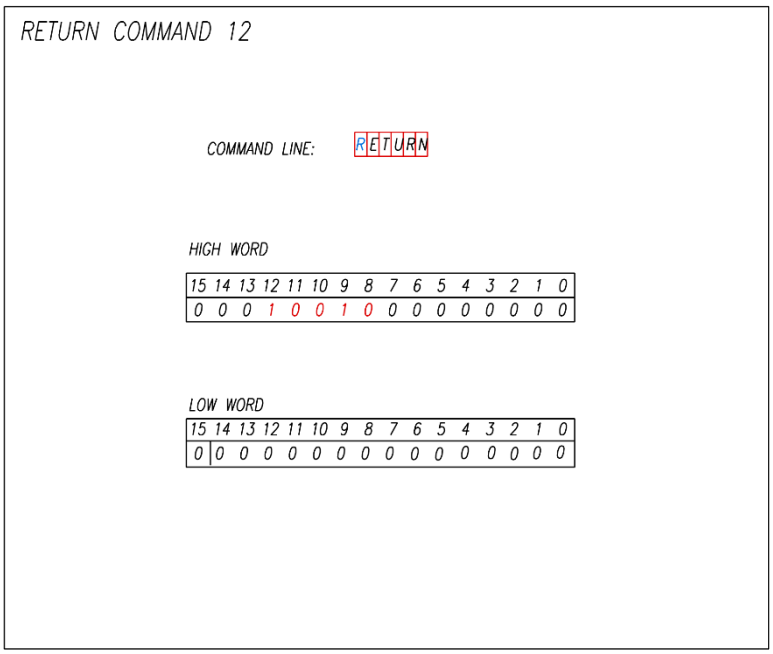
EXAMPLE: **X POSITION ADJUST +/- 255 <ENTER>** **H=1000, L=00FF**
Y POSITION ADJUST +/- 32767 <ENTER> **H=5000, L=7FFF**

JOG op-code 0x11 **MOTION COMMAND**



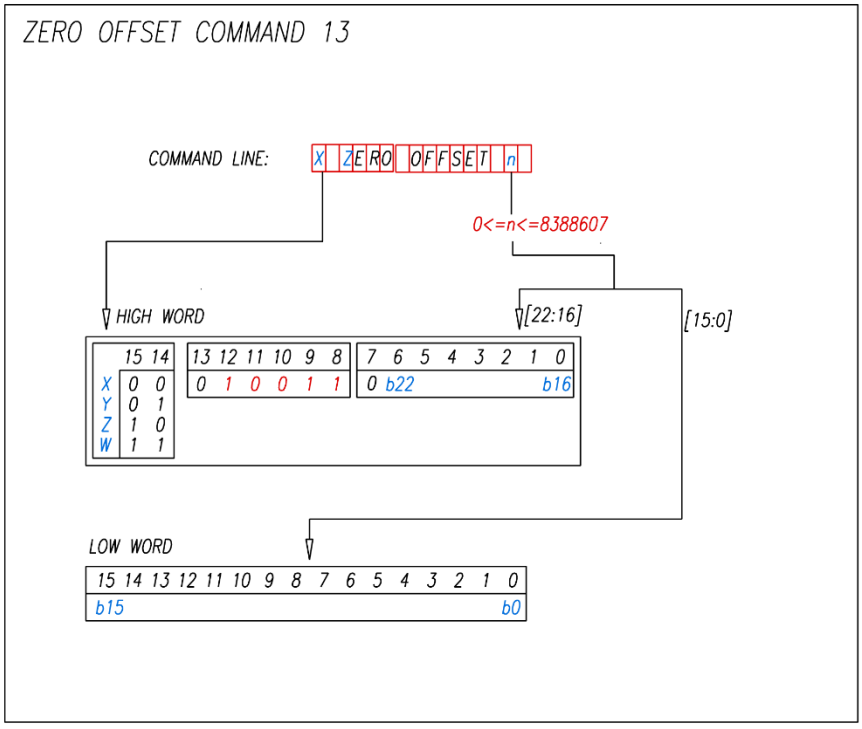
EXAMPLE: **JOG X <ENTER>** **H=1101, L=0000**
JOG X, Y <ENTER> **H=1103, L=0000**
JOG X, Y, Z, W <ENTER> **H=110F, L=0000**

RETURN op-code 0x12 **PROGRAM FLOW COMMAND**



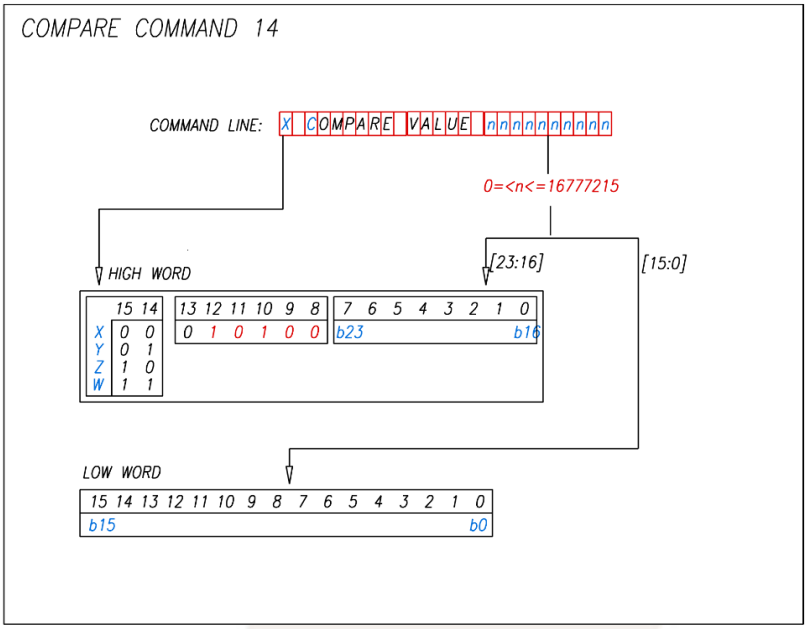
EXAMPLE: **RETURN <ENTER>** **H=1200, L=0000**

ZERO OFFSET op-code 0x13 **CONFIGURATION COMMAND**



EXAMPLE:
X ZERO OFFSET 255 <ENTER> **H=1300, L=00FF**
Y ZERO OFFSET 65535 <ENTER> **H=5300, L=FFFF**
Z ZERO OFFSET 16777215 <ENTER> **H=93FF, L=FFFF**

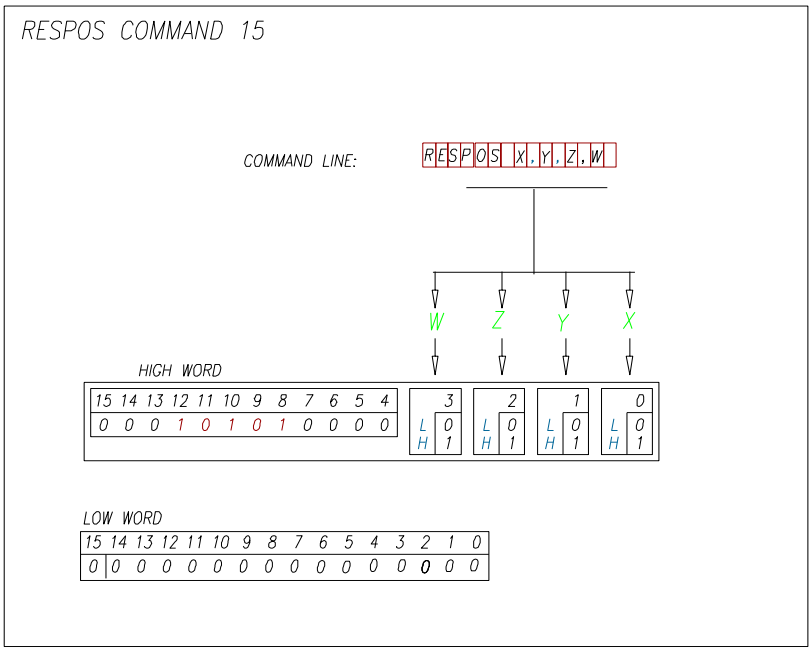
COMPARE **op-code 0x14** **MISCELLANEOUS COMMAND**



EXAMPLE:

X COMPARE VALUE 255 <ENTER>	**H=1400, L=00FF**
Y COMPARE VALUE 65535 <ENTER>	**H=5400, L=FFFF**
Z COMPARE VALUE 16777215 <ENTER>	**H=94FF, L=FFFF**

RESPOS **op-code 0x15** **MOTION COMMAND**



EXAMPLE:

RESPOS X <ENTER>	**H=1501, L=0000**
RESPOS X, Y <ENTER>	**H=1503, L=0000**
RESPOS X, Y, Z, W <ENTER>	**H=150F, L=0000**

APPENDIX D: MORE SAMPLE CODE

JOG COMMAND:

```
;SET UP:
;  1. TURN POWER OFF
;  2. CONNECT THE GM215 WITH A FREE-RUNNING MOTOR.
;  3. USE 3 MOMENTARY SWITCHES (SW1, SW2, SW3). CONNECT ONE SIDE OF THE SWITCHES
;     TO IN1, IN2, IN3 AND CONNECT THE OTHER SIDE OF THE SWITCHES TO AN EXTERNAL
;     +5VDC (WITH GROUND CONNECTED TO CN2-PIN4). CN2 IS THE 12-PIN CONNECTOR.
;  4. SET AXIS SWITCHES TO X-AXIS
;  5. TURN POWER ON AND EXECUTE THE FOLLOWING PROGRAM:
;TEST 1: DIGITAL JOG (NO TRIMPOTS ARE USED)
;       DIGITAL JOG WILL USE THE ACCELERATION, VELOCITY AND LIMIT VALUES SET IN
;       "xconfig" SUBROUTINE.
;
          call xconfig        ; call subroutine to set up motion parameters
          x 10000             ; move to mid-point (limit is 20000)
           jog x              ; execute Jog command in digital mode
;
; AT THIS STEP, THE MOTOR IS AT MID-POINT AND WAIT FOR SWITCHES PRESSING
; 1. IF SW2 IS PRESSED, THE MOTOR WILL JOG CW UNTIL ITS CW LIMIT (20000)
; 2. IF SW3 IS PRESSED, THE MOTOR WILL JOG CCW UNTIL CCW LIMIT (0)
; 3. IF SW1 IS PRERSSED, JOG COMMAND WILL BE TERMINATED AND NEXT COMMAND WILL BE
EXECUTED         ; (TEST2).
;
;TEST 2: ANALOG JOG (TRIM3, TRIM4 & TRIM5 WILL BE USED)
;       ANALOG JOG WILL USE TRIM3 TO SET ACCELERATION, TRIM5 TO SET VELOCITY
;       TRIM4 TO JOG CW OR CCW.
;       WHEN TRIM4 IS <25% THE MOTOR WILL TRAVEL CCW UNTIL CCW LIMIT (0)
;       WHEN TRIM4 IS >75% THE MOTOR WILL TRAVEL CW UNTIL CW LIMIT (20000)
;
          wait 2.0 seconds ; wait for SW2 & SW3 released
          analog inputs to x          ; set analog voltage inputs values for X-axis
           jog x                      ; execute Jog command in analog mode
;
; IF SW1 IS PRERSSED, THIS COMMAND WILL BE TERMINATED AND NEXT COMMAND WILL BE
; EXECUTED (LOOP).
;
 loop:
          goto loop

xconfig:
   x configure: 0.5 amps, idle at 71% after 2.5 seconds
   x limit cw 20000          ; SET JOG CW LIMIT
   x acceleration 32         ; SET MOTOR ACCELERATION
   x velocity 5000           ; SET MOTOR VELOCITY
   return
```

SPEED CONTROL COMMAND:

```
;SET UP:
;   1. TURN POWER OFF
;   2. CONNECT THE GM215 WITH A FREE-RUNNING MOTOR.
;   3. USE 2 MOMENTARY SWITCHES (SW2, SW3). CONNECT ONE SIDE OF THE SWITCHES
;      TO IN2 & IN3 AND CONNECT THE OTHER SIDE OF THE SWITCHES TO AN EXTERNAL
;      +5VDC (WITH GROUND CONNECTED TO CN2-PIN4). CN2 IS THE 12-PIN CONNECTOR.
;   4. SET AXIS SWITCHES TO X-AXIS
;   5. TURN POWER ON AND EXECUTE THE FOLLOWING PROGRAM:

;TEST 1: DIGITAL SPEED CONTROL (NO TRIMPOTS ARE USED)
        call xconfig            ; call subroutine to set up motion parameters
        x speed control +1000   ; execute Speed Control command in digital mode
;
; AT THIS STEP, THE MOTOR WILL RUN CW UNTIL:
; 1. IF SW3 IS PRESSED, THE MOTOR WILL CHANGE ITS DIRECTION TO CCW
; 2. IF SW2 IS PRESSED, THE MOTOR WILL CHANGE ITS DIRECTION TO CW
; 3. IF BOTH SW2 & SW3 ARE PRERSSED, THIS COMMAND WILL BE TERMINATED AND
;    NEXT COMMAND WILL BE EXECUTED (TEST2).
;
;TEST 2: ANALOG SPEED CONTROL (TRIM4 & TRIM5 WILL BE USED)
        wait 2.0 seconds        ; wait for SW2 & SW3 released
        analog inputs to x      ; set analog voltage inputs values for X-axis
        x speed control +1000   ; execute Speed Control command in analog mode
;
; AT THIS STEP, THE MOTOR WILL RUN CW UNTIL:
; 1. IF SW3 IS PRESSED, THE MOTOR WILL CHANGE ITS DIRECTION TO CCW (USE TRIM4
;    TO CONTROL THE SPEED FOR CCW MOTION)
; 2. IF SW2 IS PRESSED, THE MOTOR WILL CHANGE ITS DIRECTION TO CW (USE TRIM5
;    TO CONTROL THE SPEED FOR CW MOTION)
; 3. IF BOTH SW2 & SW3 ARE PRERSSED, THIS COMMAND WILL BE TERMINATED AND
;    NEXT COMMAND WILL BE EXECUTED (LOOP).
;
loop:
        goto loop

xconfig:
        x configure: 0.5 amps, idle at 71% after 2.5 seconds ; SET MOTOR CURRENT
        x acceleration 32       ; SET MOTOR ACCELERATION
        x velocity 2000         ; SET MOTOR VELOCITY
        return
```

POSITION ADJUST COMMAND:

```
;SET UP:
;  1. TURN POWER OFF
;  2. CONNECT THE GM215 WITH A FREE-RUNNING MOTOR.
;  3. USE 1 MOMENTARY SWITCH FOR SW1. CONNECT ONE SIDE OF SW1 TO IN1 AND
;     CONNECT THE OTHER SIDE OF THE SW1 TO AN EXTERNAL +5VDC (WITH GROUND
;     CONNECTED TO CN2-PIN4). CN2 IS THE 12-PIN CONNECTOR.
;  4. SET AXIS SWITCHES TO X-AXIS
;  5. EXECUTE THE FOLLOWING PROGRAM:

;TEST 1: POSITION ADJUST (TRIMPOT 5 WILL BE USED)
;       POSITION ADJUST WILL USE THE ACCELERATION, VELOCITY VALUES SET IN
;       "xconfig" SUBROUTINE.
          call xconfig                  ; call subroutine to set up motion parameters
          x position adjust +/-2000 ; execute position adjust command
;
; AT THIS STEP, THE MOTOR WILL MOVE TO A POSITION CORESSPONDING TO TRIM5 VALUE.
; USE TRIM5 TO ADJUST THE AXIS POSITION WITHIN A CW/CCW RANGE SET BY THE n VALUE
; (n=2000 IN THIS EXAMPLE). THE ADJUSTMENT IS O% WHEN TRIM5 IS AT THE MID-POINT
; IF SW1 IS PRERSSED, POSITION ADJUST COMMAND WILL BE TERMINATED AND NEXT COMMAND
; WILL BE EXECUTED (LOOP).
;
loop:
          goto loop
;
xconfig:
          x configure: 0.5 amps, idle at 71% after 2.5 seconds ; SET MOTOR CURRENT
          x acceleration 32             ; SET MOTOR ACCELERATION
          x velocity 5000               ; SET MOTOR VELOCITY
           return
```

RESPOS COMMAND:

```
;SET UP:
;  1. TURN POWER OFF
;  2. CONNECT THE GM215 WITH A FREE-RUNNING MOTOR.
;  3. USE 2 MOMENTARY SWITCHES (SW2, SW3). CONNECT ONE SIDE OF THE SWITCHES
;     TO IN2, IN3 AND CONNECT THE OTHER SIDE OF THE SWITCHES TO AN EXTERNAL
;     +5VDC (WITH GROUND CONNECTED TO CN2-PIN4). CN2 IS THE 12-PIN CONNECTOR.
;  4. SET AXIS SWITCHES TO X-AXIS
;  5. TURN POWER ON AND EXECUTE THE FOLLOWING PROGRAM:
;
;TEST 1: USING RESPOS COMMAND (IMPLEMENT AN INDEX FUNCTION)
        call xconfig        ; CALL SOUBROUTINE TO SETUP MOTION PARAMETERS
        x limit cw 10000    ; SET CW LIMIT
index1:
        x +5000
        goto exit
index2:
        x -5000
exit:
        respos x                    ; RESET MOTOR POSITION
loop:
        if x in2 is on goto index1  ;  TURN ON SW2 TO RUN THE MOTOR CW
        if x in3 is on goto index2  ;  TURN ON SW3 TO RUN THE MOTOR CCW
        goto loop

;TEST 2: MORE TESTS TO CLARIFY RESPOS COMMAND
;
        call xconfig        ; call subroutine to set up motion parameters
        x limit cw 10000    ; SET CW LIMIT
        x +9999             ; THIS COMMAND WILL BE EXECUTED BECAUSE MP IS UNDER THE LIMIT
                            ; MP = MOTOR POSITION
        respos x    ; MOTOR POSITION AND ITS CW LIMIT WILL BE ADDED AN OFFSET OF 3FFFFF
                    ; AFTER THIS COMMAND IS EXECUTED, THE MOTOR POSITION IS AT 3FFFFF OR
                    ; 4194303, AND ITS LIMIT IS 4194303+10000=4204303
  ; AFTER RESPOS, IF WE EXECUTE:
        x +9999     ; THIS COMMAND WILL BE EXECUTED BECAUSE MP IS UNDER THE LIMIT.
                    ; MOTOR POSITION=4194303, LIMIT=4204303
  ; AFTER RESPOS, IF WE EXECUTE:
        x limit cw 10000 ; SET THE LIMIT TO 10000
        x +9999     ; THIS COMMAND WON'T BE EXECUTED BECAUSE MP IS ABOVE THE LIMIT
                    ; MOTOR POSITION=4194303, LIMIT=100000
  ; AFTER RESPOS, IF LIMIT COMMAND IS USED, SHOULD ADD THE OFFSET (4194303)
        x limit cw 4204303          ; OFFSET+10000 (4194303+10000)
        x +9999                     ; THIS COMMAND WILL BE EXECUTED BECAUSE MP IS UNDER THE LIMIT

xconfig:
        x configure: 0.5 amps, idle at 71% after 2.5 seconds ; SET MOTOR CURRENT
        x acceleration 32           ; SET MOTOR ACCELERATION
        x velocity 5000   ; SET MOTOR VELOCITY
        return
```

APPENDIX E: ERROR CODE and FUSE REPLACEMENT (Error code is for GM215 with FPGA version 2 or later)

LED1 and LED2 are used to display error code for the system.
LED1 is the combination of 2 different LEDs for displaying 2 colors: red and green.

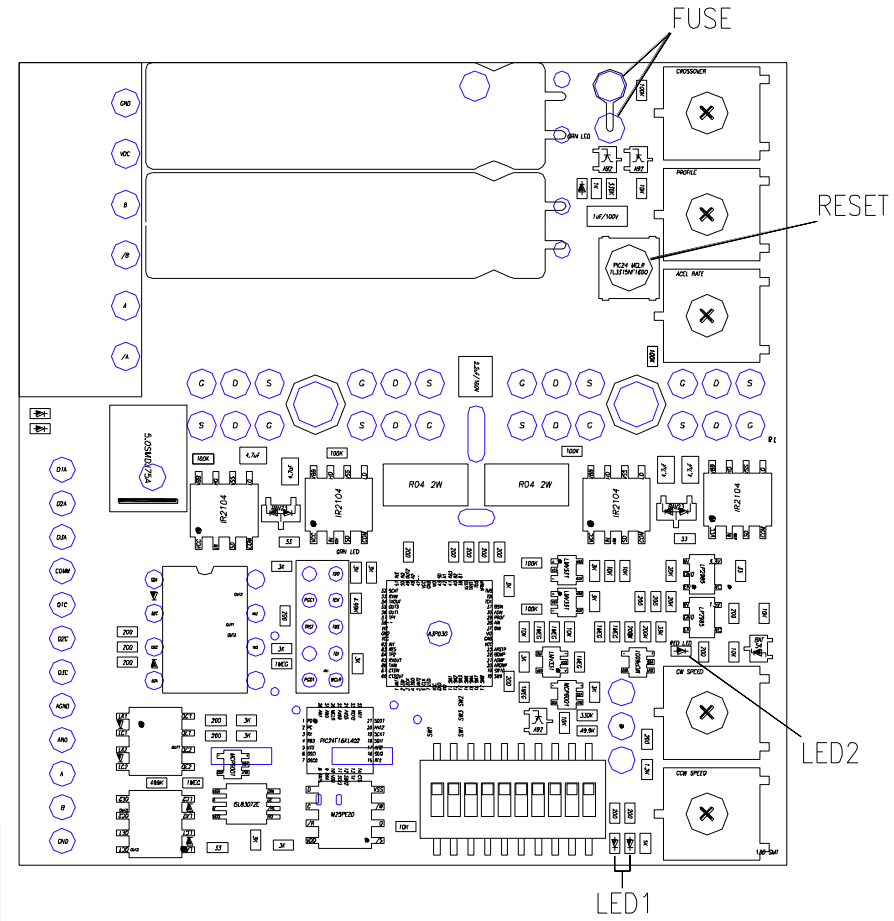


Figure 22: LEDs, Fuse, and Reset button location

Code 1: LED1 flashing GRN GRN GRN : IDLE (ready to use)

Code 2: . . . . . GRN GRN RED : NO PHASE A (check phase A connection)

Code 3: . . . . . GRN RED GRN : NO PHASE B (check phase B connection)

Code 4: . . . . . GRN RED RED : NO MOTOR (check motor connection)

Code 5: . . . . . RED GRN GRN : RUN MODE IDLE (wait for start signal at IN1)

Code 6: . . . . . RED GRN GRN : PIC/FPGA ERROR (if not in Run Mode Idle)

Code 7: LED1 is RED (no flashing) . . . . : DISABLED (Step/Dir or Motor Drive Mode)

Code 8: LED1 is RED (no flashing) . . . : FAULT (if not disabled in Step/Dir)

Code 9: LED2 is RED: BLOWN FUSE (replace the fuse, see Note 2)

Note 1: FAULT (error code 6b) happens when the motor draws too much current or the board has a failure if it is not disabled in motor drive mode. Check power supply, turn it off and then on again. If the problem persists after power is up, the board needs to be repaired.

Note 2: To replace the fuse, pull out the bad fuse from fuse sockets and insert the new fuse (5A rating).

APPENDIX F: UPDATE FIRMWARE

SET UP:

- Download or copy ds30LoaderGui.exe to a directory. This file can be downloaded from Geckodrive website.
- Connect the GM215 with host PC by using a RS-485 cable. Disconnect all devices except for the target device.
- Set SW1-SW4 to ON position

Firmware can be updated in two different ways with or without GeckoMotion.

A. WITH GECKOMOTION:

1. Start GeckoMotion to get the main menu. There is no need to establish a connection between GeckoMotion and the GM215 (do not click Connect). If “Device error” message is displayed, click “Continue” to advance.

2. On the main menu, select “File”, and then “Firmware”.

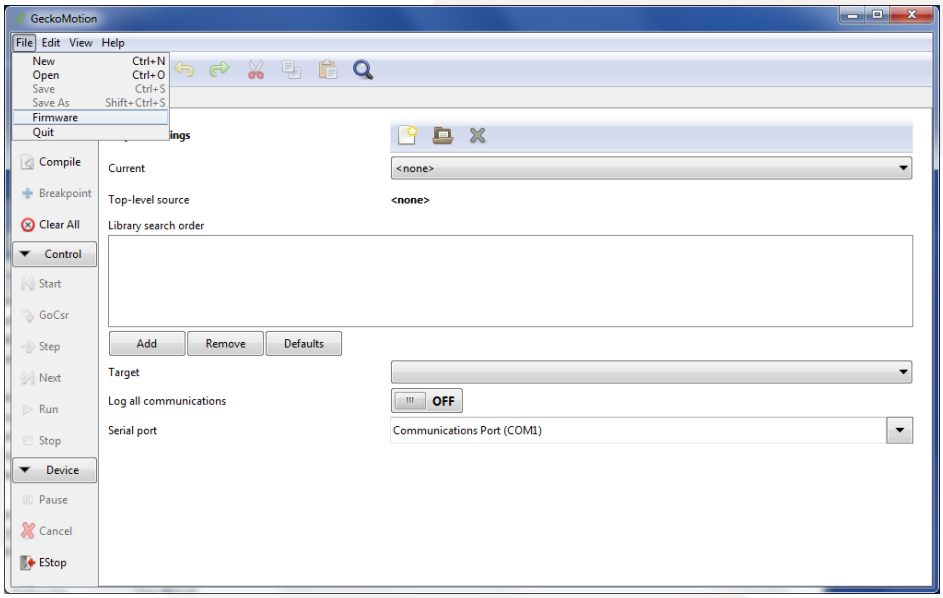


Figure 23: How to update firmware

3. Browse to the location of the ds30LoaderGui.exe, and click OK

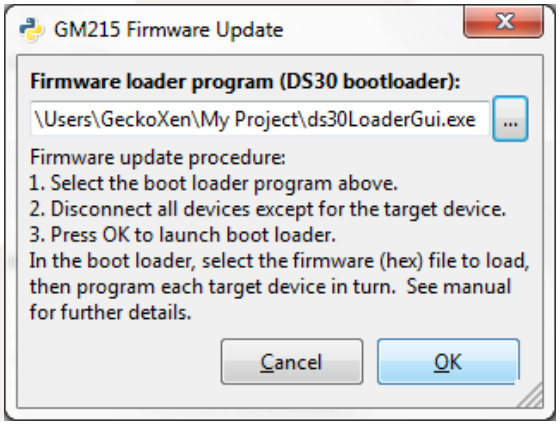


Figure 24: How to select ds30LoaderGui.exe

4. Go to section **C (DS30 LOADER GUI)** to continue (Figure 25).

B. WITHOUT GECKOMOTION:

The ds30 Loader GUI does not require to be installed, it can be run directly from the bin director.

1. WINDOWS:
   - Double click **ds30LoaderGui.exe** from Explorer
   - Use run on the start menu, browse to **ds30LoaderGui.exe**
   - Start from command prompt

   Go to section **C (DS30 LOADER GUI)** to continue (Figure 25).

2. LINUX:
   Run command: **mono ds30LoaderGui.exe**
   Go to section **C (DS30 LOADER GUI)** to continue (Figure 25).

3. MAX OS X:
   Run command: **mono ds30LoaderGui.exe**
   Go to section **C (DS30 LOADER GUI)** to continue (Figure 25).

C. DS30 LOADER GUI:

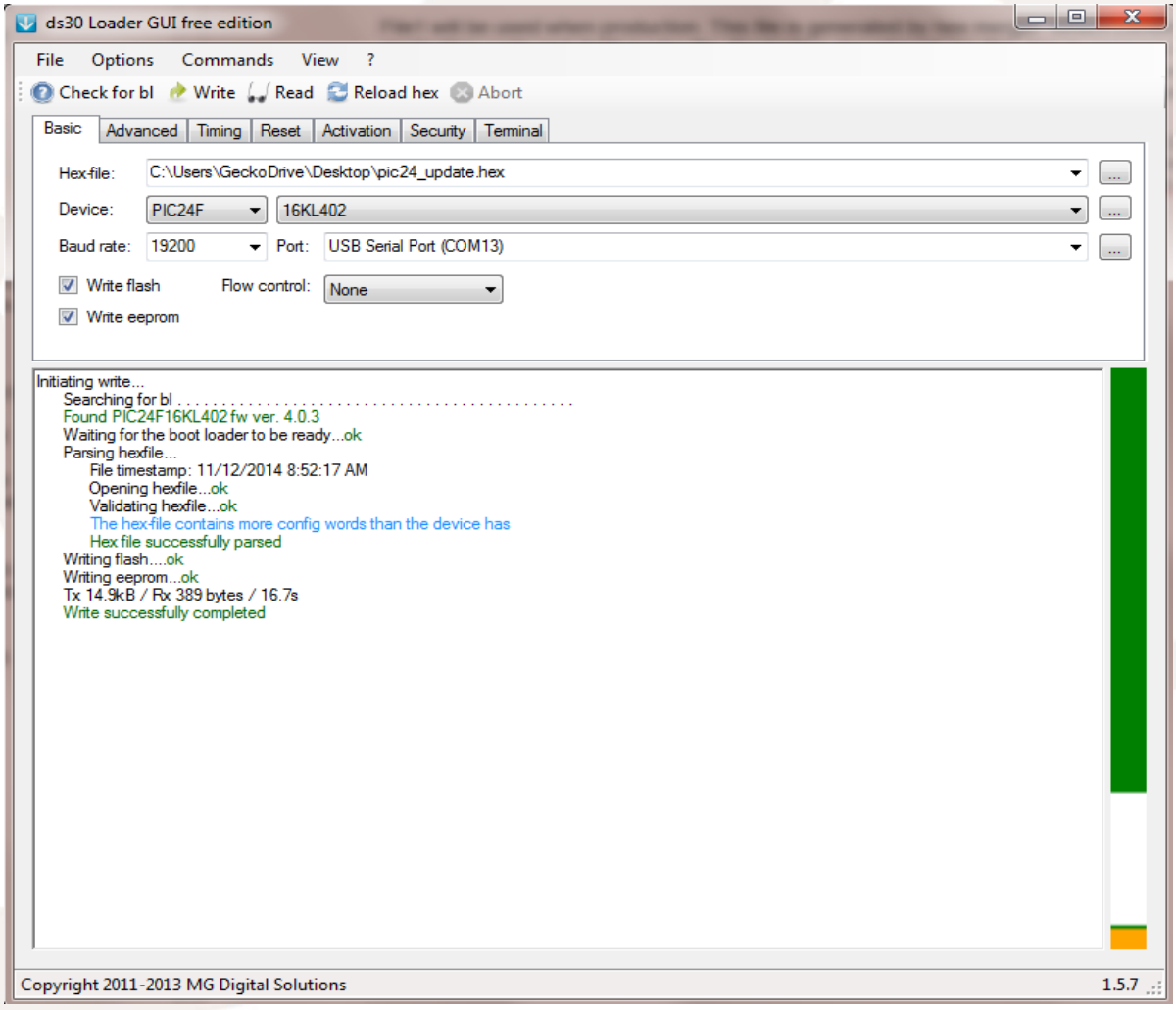


Figure 25: ds30 Loader GUI

- **SETUP:**

  The ds30 Loader GUI consists of 5 main parts; the menu, the toolbar, the tab pages, the output text box and the graphical hex file representation.
  The menu contains: File, Option, Commands, View.
  The tool bar contains: Check for bl, Write, Read, Reload hex, Abort.
  The tab pages contains: Basic, Advanced, Timing, Reset, Activation, Security, Terminal.

  1) If the tab pages contains 1 item only, such as Basic, user should get more items by clicking View in the menu and select Advanced-mode. This will bring more items to the tab pages such as Advanced, Timing, Reset, Activation, Security,Terminal.
  2) Click Timing and set its parameters like this:
     Hello timeout (ms): **10000**
     Poll time (ms): **80** (should be between 50 and 80)
     Timeout (ms): **10000**
     Delay after port open (ms): **0**
  3) Click Basic to go back the main page.
  4) Select Hex-file: in Hex-file box, select the file to be updated. This file can be downloaded from Geckodrive web. After the file is selected, "Hex file successfully parsed" will be displayed on the output text box (see Fig 20).
  5) Set Device to **PIC24F** and **16KL402** (see Fig 25).
  6) Set Baud rate to **19200**, Port: available port for RS485 communication with the GM215.
  7) Check (yes) Write flash checkbox and Write eeprom checkbox (see Fig 25).
  8) Set Flow control to **None**.

- **UPDATE:**

  Write command must be executed to update the firmware. When Write button on the tool bar is clicked, "Searching for bl….." will be displayed on output text box for an amount of time that was set in Timing (10000ms in this example). During this time, RESET button must be pressed down and released before the boot loader times out. See Fig 19 for the location of RESET button.

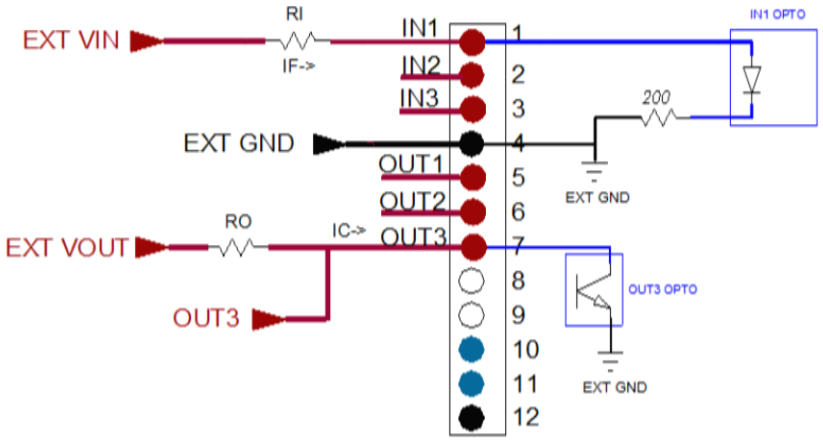
  There are two different ways to invoke the boot loader and perform the write:
  1) Procedure 1:
     a) Press and hold down RESET button. (see Fig 19 for RESET button location)
     b) Click the Write button in the GUI. When "Searching for bl…" is displaying on the output text box, release RESET button.
  2) Procedure 2:
     a) Click the Write button in the GUI. "Searching for bl…" will be displaying on the output text box.
     b) Press and hold down RESET button for 1 to 2 seconds, and then release.

  If the procedure is performed properly, it will execute Write command and "Write successfully completed" will be displayed at the end (see Fig 25).

  Note: Procedure 1 may be more reliable than procedure 2.

APPENDIX G: INPUTS and OUTPUTS INTERFACE

GM215 INPUTS/OUTPUTS INTERFACE



NOTE FOR INPUTS:

1) IF SHOULD BE BETWEEN 2MA TO 10MA, MAX 25MA.
2) VIN SHOULD BE BETWEEN 3.3VDC TO 60VDC (MAX)
3) RI CALCULATION WITH IF=5MA: RI(KOhm)=[(VIN-VF)/IF]-0.2 = [(VIN-1.45)/5]-0.2
4) EXAMPLE 1: IF VIN IS 3.3V, RI = [(3.3-1.45)/5]-0.2 = 0.170K (±10%)
5) EXAMPLE 2: IF VIN IS 5V, RI = (5-1.45)/5-0.2 = 0.510K (±10%)
6) EXAMPLE 3: IF VIN IS 12V, RI = (12-1.45)/5-0.2 = 1.92K (±10%)
7) EXAMPLE 4: IF VIN IS 24V, RI = (24-1.45)/5-0.2 = 4.51K (±10%)

NOTE FOR OUTPUTS:

1) IC SHOULD BE LESS THAN 150MA.
2) VOUT SHOULD BE BETWEEN 3.3VDC TO 60VDC (MAX)
3) RO CALCULATION WITH IC=5MA: RO(KOhm)=(VOUT-VCE)/IC = (VOUT-0.4)/5
4) EXAMPLE 1: IF VOUT IS 3.3V, RO = (3.3-0.4)/5 = 0.580K (±10%)
5) EXAMPLE 2: IF VOUT IS 5V, RO = (5-0.4)/5 = 0.920K (±10%)
6) EXAMPLE 3: IF VOUT IS 12V, RO = (12-0.4)/5 = 2.32K (±10%)
7) EXAMPLE 4: IF VOUT IS 24V, RO = (24-0.4)/5 = 4.72K (±10%)

Figure 26: Inputs/Outputs Interface

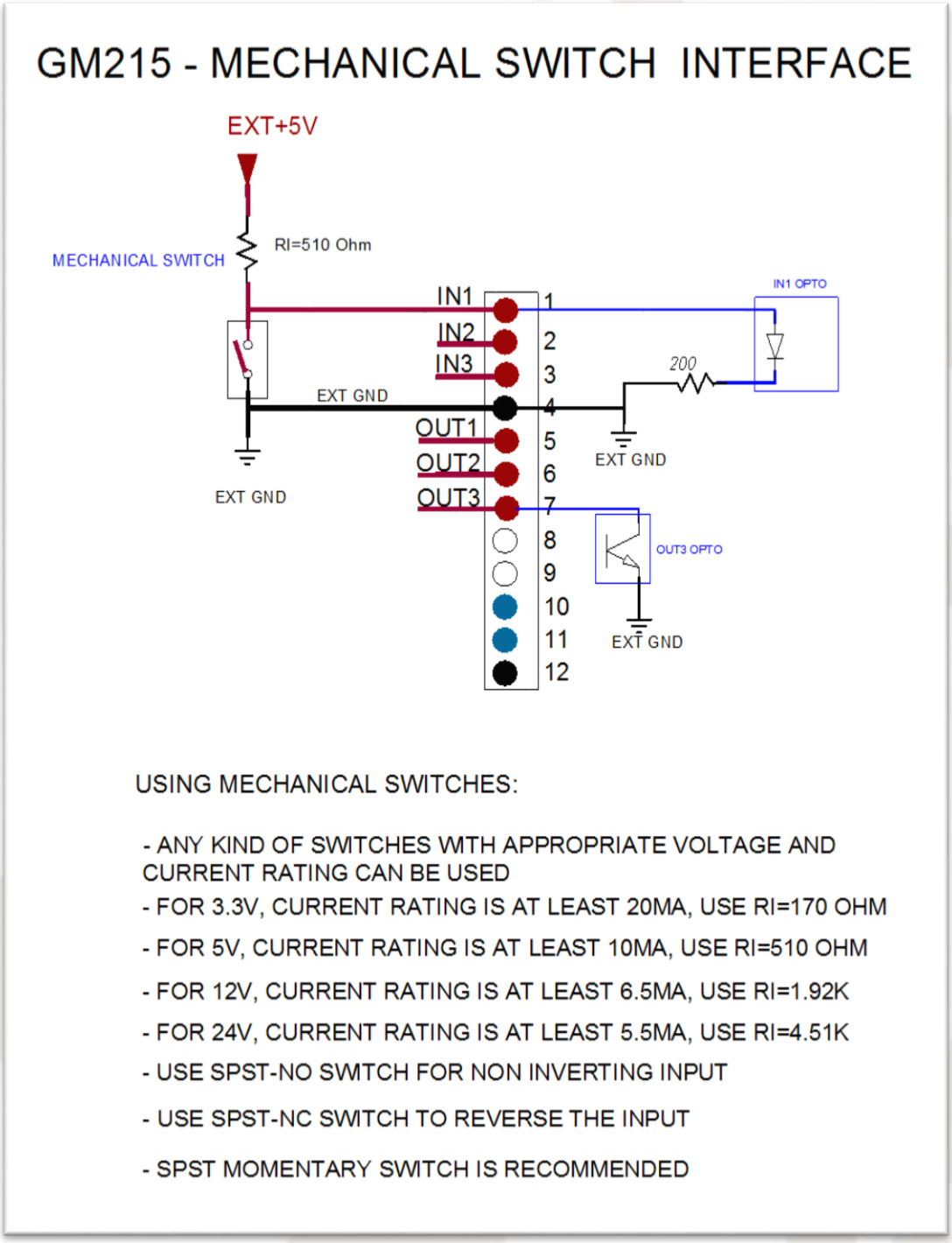
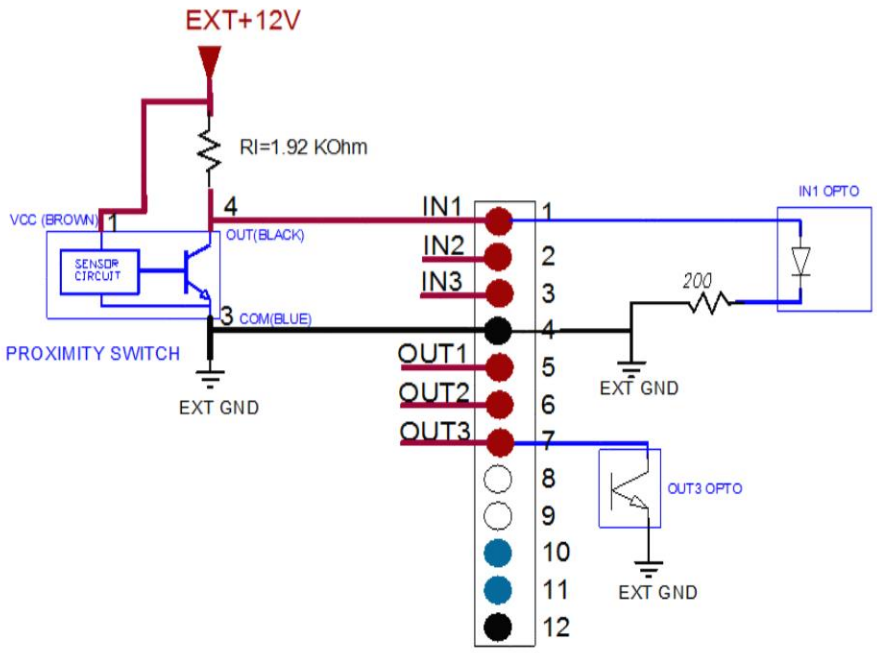


Figure 27: Mechanical Switch Interface

GM215 - INDUCTIVE SWITCH INTERFACE

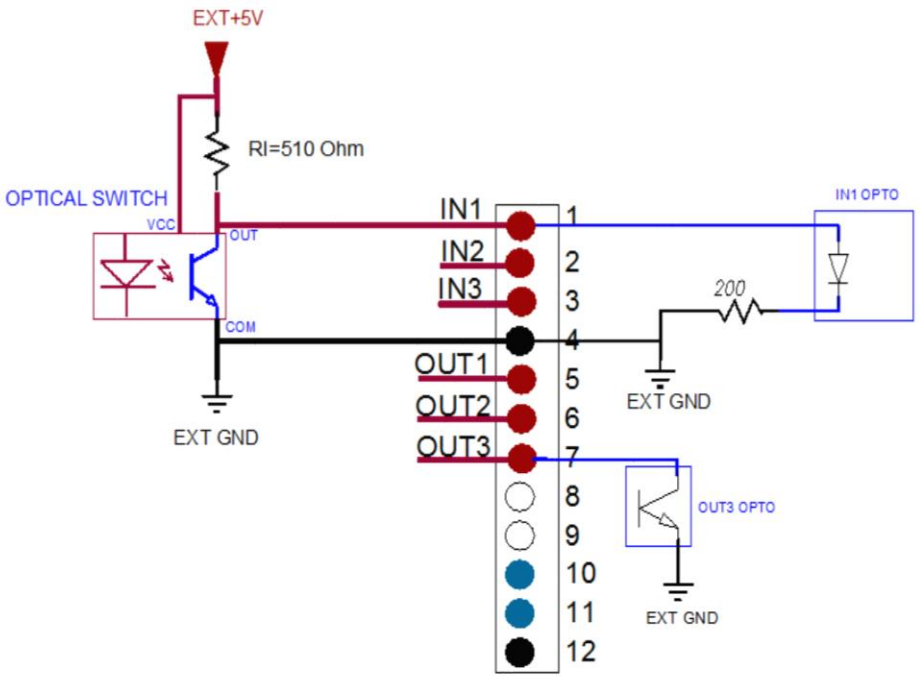


SOME INDUCTIVE SWITCHES:

- IMA224.8C, CROUZET-USA., 0.8MM, 10-30VDC, NPN - NORMALLY OPEN
- IMA2261C, CROUZET-USA., 1MM, 10-30VDC, NPN - NORMALLY OPEN
- IMA2281C, CROUZET-USA., 4MM, 10-30VDC, NPN - NORMALLY OPEN
- NBB1.5-8GM25-E0, PEPPERL+FUCHS INC., 1.5MM, 2KHZ, 10-30VDC, NPN-NORMALLY OPEN
- NBB2-8GM30-E2, PEPPERL+FUCHS INC., 2MM, 3KHZ, 10-30VDC, NPN - NORMALLY OPEN
- NBN3-8GM30-EO, PEPPERL+FUCHS INC., 3MM, 2.5KHZ, 10-30VDC, NPN - NORMALLY OPEN
- TL-M2ME1-3 2M, OMRON AUTOMATION AND SAFETY, 2MM, 5-24VDC, NPN - NORMALLY OPEN
- E57-18GS05-CDB, EATON, PROXIMITY SENSOR, 18MM, 5MM RANGE, DC, 3-WIRE, 10-30VDC, NPN OUTPUT
- INDUCTIVE 3 WIRE DC PROXIMITY SENSORS, 8, 12, 18 OR 30MM, AUTONICS PR SERIES, 10-30VDC, PART NUMBERS: PR08-1.5DN THROUGH PR30-15DN

Figure 28: Inductive Switch Interface

GM215 - OPTICAL SWITCH INTERFACE



SOME OPTICAL SWITCHES:

- OPL530..OPL563 (-OC FOR OPEN COLLECTOR) FROM TT ELECTRONICS / OPTEK TECHNOLOGY, VCC=4.5V TO 16V, 935NM WAVELENGTH, PHOTO DETECTOR, LOGIC OR OPEN COLLECTOR OUTPUT

- QSE159 FROM FAIRCHILD, 880NM WAVELENGTH, PHOTO DETECTOR / OPTICAL SENSOR, LOGIC OR OPEN COLLECTOR OUTPUT

- SDP8601-001 FROM HONEYWELL, 935NM WAVELENGTH, PHOTO DETECTOR, LOGIC OR OPEN COLLECTOR OUTPUT

- MLX75303KXD-EAA-000-RE FROM MELEXIS TECHNOLOGIES NV, 850NM WAVELENGTH, OPTICAL SWITCH, OPEN DRAIN OUTPUT

- SFH 5140F FROM OSRAM OPTO SEMICONDUCTORS, 950NM WAVELENGTH, PHOTO DETECTOR, LOGIC OUTPUT

Figure 29: Optical Switch Interface

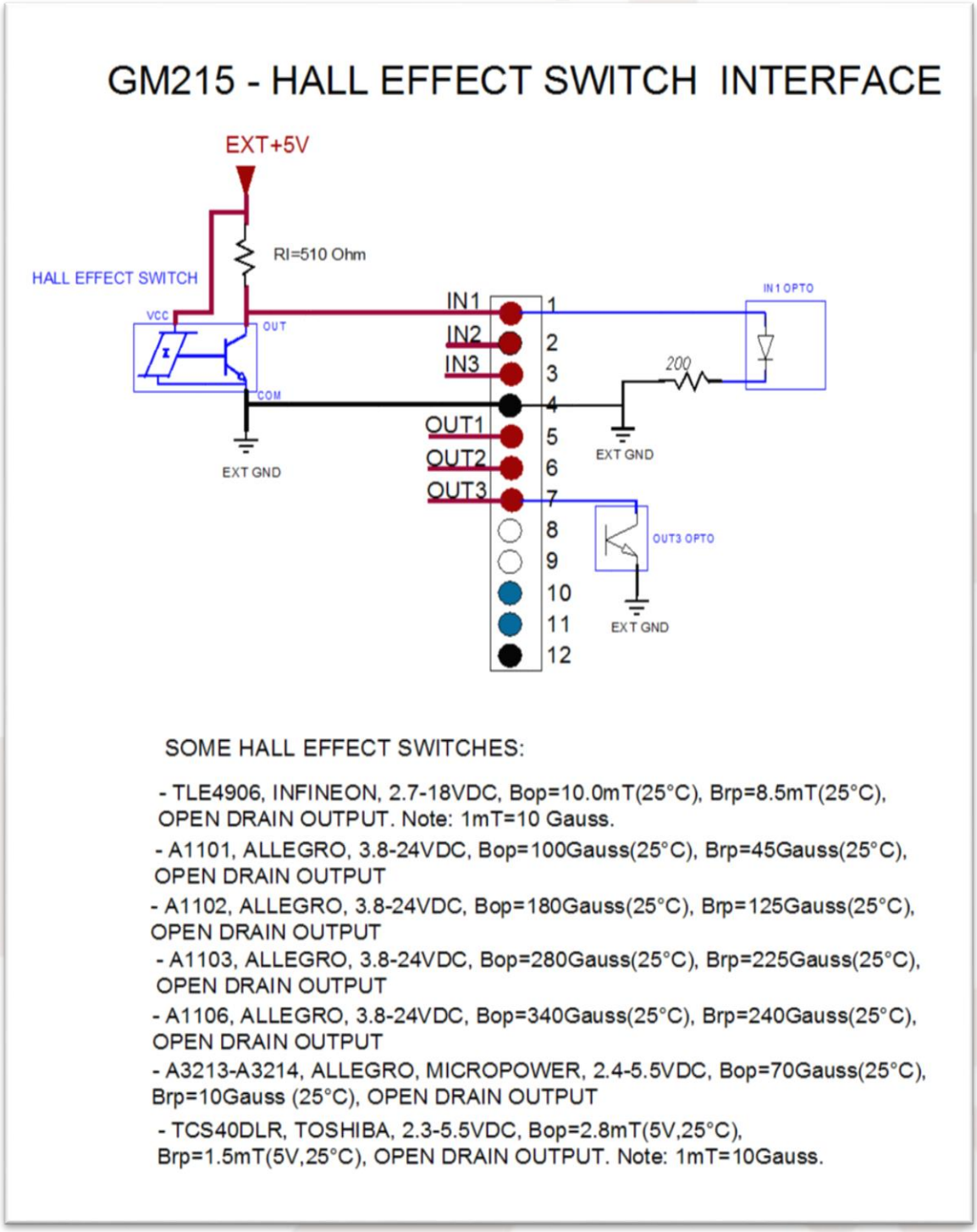


Figure 30: Hall Effect Switch Interface

DISCLAIMER

CERTAIN APPLICATIONS USING POWER PRODUCTS MAY INVOLVE POTENTIAL RISKS OF DEATH, PERSONAL INJURY OR SEVERE DAMAGE TO PROPERTY. GECKODRIVE INC. PRODUCTS ARE NOT DESIGNED, AUTHORIZED OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT DEVICES OR OTHER CRITICAL APPLICATIONS. INCLUSION OF GECKODRIVE INC. PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE PURCHASER'S OWN RISK.

In order to minimize risks associated with the purchaser's application, adequate design and operating safeguards must be provided by the purchaser to minimize inherent or procedural hazards. GECKODRIVE INC. assumes no liability for applications assistance or the purchaser's product design. GECKODRIVE INC. does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright or other intellectual property right of GECKODRIVE INC.

MANUAL CHANGE LOG

DATE	MAJOR CHANGES
10/09/2014	GM215 Rev7-A Manual Published
01/05/2015	GM215 Rev7-B Manual UPDATED
01/26/2015 02/06/2015	GM215 Rev7-C Manual UPDATED (add limit switch wiring and run mode wiring descriptions) GM215 Rev7-C Manual UPDATED (update Example code)
07/03/2015 07/06/2015 04/15/2016	GM215 REV7-D Manual UPDATED (add velocity calculation formula and one more command RESPOS) GM215 REV7-D Manual UPDATED (add analog input description) GM215 REV7-F Manual UPDATED (add RESPOS command, table of contents, error code, using GeckoMotion software, Sample code, update firmware, inputs and outputs interface)