Servosila G-code Reference

Servosila Motion Controller

Revision D (December 2023)

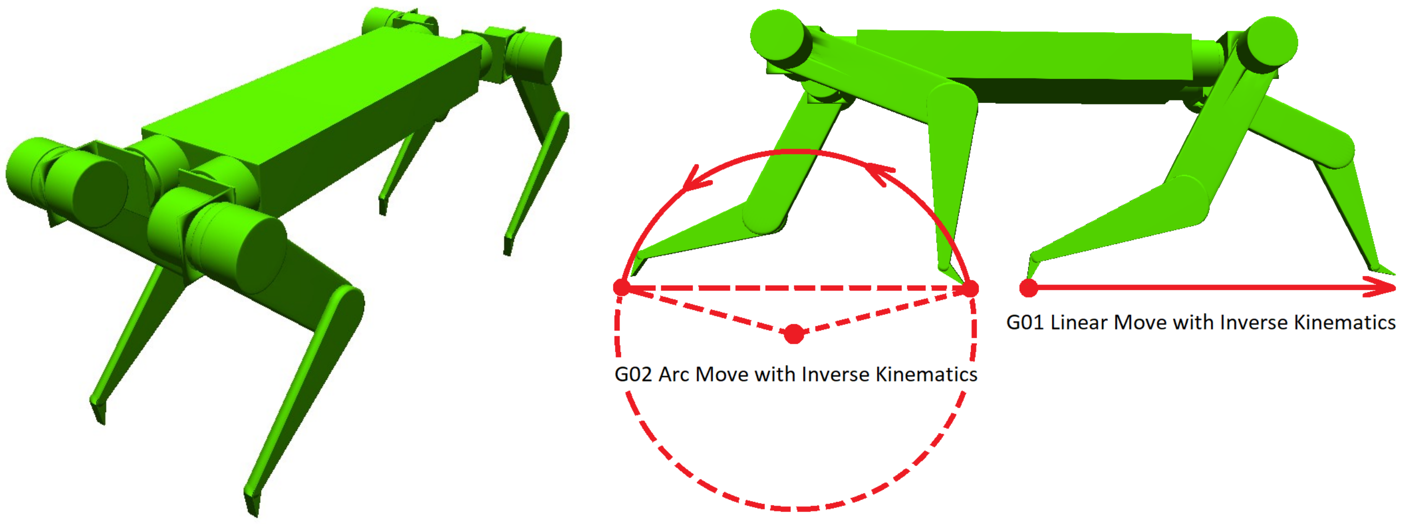
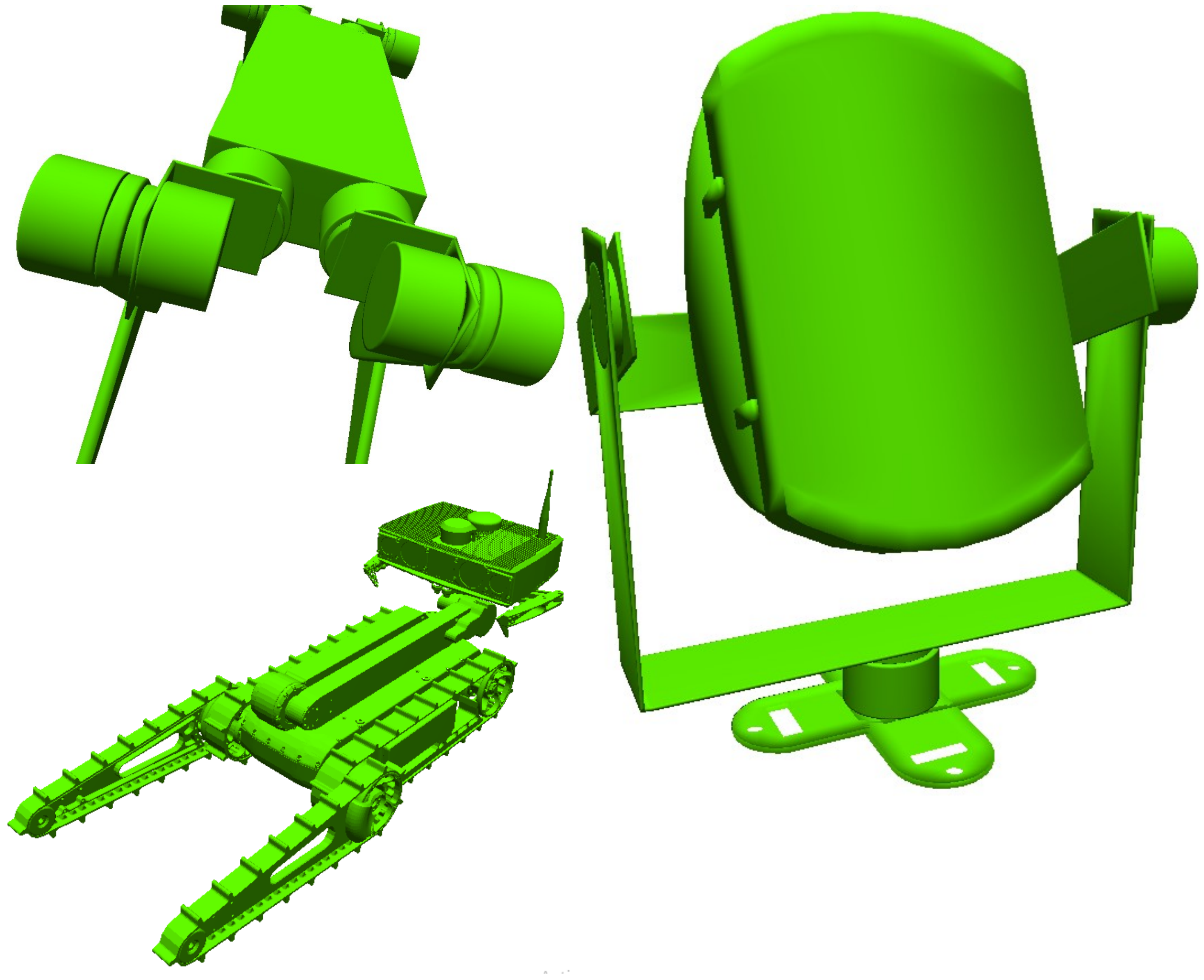

G02 Arc Move with Inverse Kinematics
G01 Linear Move with Inverse Kinematics

TABLE OF CONTENTS

About Servosila Motion Controller

Servosila Motion Controller is embedded software for controlling motion of modern multi-axis robotic systems. The software runs on Linux, Windows or as a firmware on embedded MCUs.



Servosila Motion Controller uses **G-code** for the following **purposes**:

- as a way to define **geometry** of coordinated motions in a text format,
- as a high-level communications **protocol** between the Motion Controller and higher-level user applications,
- as a simple **scripting language** for programming multi-axis robotic systems,
- as a target language for generative AI and LLMs.

G-code Commands

Overview

The following sections list G-code commands supported by Servosila Motion Controller. Several code listings are embedded throughout the document for illustration purposes.

G00 Rapid Move

The G00 command moves one or many axes to specified target positions. The command is applicable to both rotary and linear axes. The motion is performed at the axes' maximum speeds defined by their servo drives' configuration and their motors' performance. The motion is not coordinated among the axes. This means that the axes may arrive at their destinations independently of each other and at different times.

The command has an implicit "Exact Stop" feature. This means that the command's execution ends only once all participating axes have reached their destinations and stopped within a preconfigured tolerance distance from their targets. Consider using an alternative G1020 Servo Move if the "Exact Stop" feature is not required, or an alternative G01 Linear Move if a motion coordination among participating axes is needed.

`G00 X-100 [5]100 [6]200 [7]0 [8]-90`	Move axes X, 5, 6, 7, 8 to the prescribed positions. This could be rotary or linear axes or a combination of both types. Note that X,Y,Z,A,B,C is the same as [0],[1],[2],[3],[4],[5]. In other words, axes 0 to 5 have dedicated names and can be referred to either by their indices or by the names. All other axes starting from 6 to 63 do not have dedicated names and have to be referred to by their indices such as [8]. For rotary axes, the measurement units of axis travel is "degrees". For linear axes, the measurement units of axis travel are either "millimeters" or "inches" depending on the configuration settings.

G01 Linear Move

The G01 command moves axes to prescribed target positions in a *coordinated way*. This means that all participating axes arrive at their destinations together at the same time while moving at synchronized speeds. The synchronization means, for example, that if one of the axes completes 40% of its travel to a target position, then all the other axes have completed exactly as much of their travels by that time. The command is applicable to both linear and rotary axes, including mixes of axes of both types assuming a proper speed measure is used to specify a combined motion speed (see G94 Speed in Millimeters per Minute, Inches per Minute, Degrees per Minute).

If all the axes participating in the move are linear Cartesian ones, then such a motion produces a straight line trajectory in the Cartesian space similar to XYZ-type CNC machines or typical 3D printers. If any of the axes are rotary, or the linear axes are not Cartesian ones, then the trajectory is not necessarily a straight line. The axes could even belong to disjoined mechanisms, but still move in a coordinated way using the G01 command. What matters is that the axes move together in a coordinated way and arrive to their destinations at the same time. This is a "workhorse" command.

In contrast to G00 Rapid Move, the G01 command does not enforce "Exact Stop" at the end of the travel by default. Since axes have physical inertia, this might lead to rounded corners between interconnected trajectories whenever the axes are not capable to keep up with desired speed of motion. If this effect needs to be avoided, use G09 Exact Stop or G61 Enable Exact Stop commands to enable "Exact Stop" function for the G01 command.

The G01 command can be used with *Inverse Kinematics function* that translates a linear motion of an end-effector in a Cartesian space into a motion of non-Cartesian axes. See section Inverse Kinematics.

Examples of G01 Linear Moves	
G1010	Energize All Axes
G94	Units of Speed set to "Millimeters per Minute"
F6000	Speed set to 6000 Millimeters per Minute (=100 mm/sec)
G01 X-100 Y200 Z-100 ;Linear Cartesian axes	Move axes X,Y,Z to the prescribed positions in a coordinated way.
	Note: Using axis names X,Y,Z is same as using axes indices [0],[1],[2].
G01 U-10 V-50 W-100 ;Incremental move	Move axes X,Y,Z in an incremental and coordinated way.
	Note: U,V,W is an incremental way to specify motion of axes X,Y,Z.
G93	Changing Units of Speed to "Moves per Minute" before a coordinated rotary move.
F120	Speed set to 120 Moves per Minute (=2 moves/sec)
G01 [3]-45 [4]90 [5]-180 ;Rotary axes	A coordinated move of three rotary axes 3, 4, 5 by -45, 90, -180 degrees.
	The rotary axes' motion completes in 0.5 seconds (=2 moves/sec).
G93 F240	Speed set to 240 Motions per Second (=4 moves/sec).
	Combining G93 and F in one line just for clarity.
G01 X100 Y90 [3]45 [4]90 ;Linear+Rotary axes	Coordinated move of 2 linear X,Y axes and 2 rotary axes [3],[4].
	The move is completed in 0.25sec (=4 moves/sec).
G01 T3 X-159 Y-400 Z800 ;Inverse Kinematics	Move end effector/tool T3 to position X,Y,Z using Inverse Kinematics feature.
	The move is completed in 0.25sec (=4 moves/sec) as the speed is kept the same.
G1000	Reset All Axes
G1011	De-energize All Axes

	Notes:
	- Using axes names X,Y,Z is same as using axes indices [0],[1],[2].
	- Using axes names A,B,C is same as using axes indices [3],[4],[5].
	- U,V,W is an incremental way to specify motion of axes X,Y,Z.

G02 Arc/Circle Clockwise Move

See G03 Arc/Circle Counter-Clockwise Move.

G03 Arc/Circle Counter-Clockwise Move

The G02 and G03 commands cause a motion of **2 axes** along a circular trajectory or **3 axes** along a helix trajectory in a coordinated way. Depending on the provided parameters, the trajectories may be a *full circle*, *a minor arc*, *a major arc*, *a helix*, *a multi-turn circle*, *a multi-turn helix*, or a *multi-turn arc*. The trajectories could be clockwise ones (G02) or counter-clockwise ones (G03).

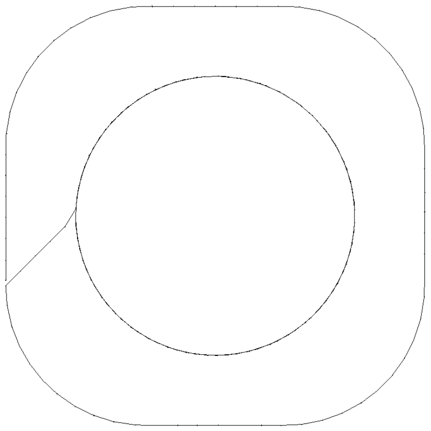
The command is applicable to linear Cartesian axes only, or otherwise must be used with *Inverse Kinematics* function to translate an arc motion in a Cartesian space into a motion of non-Cartesian axes.

By default, the G02 and G03 commands do not enforce “Exact Stop” at the end of the travel. Since axes have physical inertia, this might lead to rounded corners at the end of interconnected trajectories whenever the axes are not capable to keep up with desired speed of motion. In this case, use G09 Exact Stop or G61 Enable Exact Stop to enable “Exact Stop” function for the G02 and G03 commands.

There are two variations of the G02 and G03 commands:

- Variation #1: **Radius** is Provided (via parameter R). Positive R means minor arc. Negative R means major arc.
- Variation #2: **Arc Center** is Provided (via parameters I,J,K). This variation can be used to produce a full circle or a full helix. Note that the center of the arc/circle is defined in incremental way, so I,J,K coordinates are actually increments to a current position.

Examples of G02/G03 Arc Moves with some G01 Linear Moves	
G1010	Energize All Axes
G94 F30000	Units of Speed set to “Millimeters per Minute”
	Speed set to 30000 Millimeters per Minute (=500 mm/sec)
G00 X0 Y0	Rapid Move to the Origin point.
G01 V1000	Linear Move along axis Y. Note that V is an incremental way to refer to axis Y.
G02 X1000 Y2000 R1000	Clockwise arc move to [1000;2000] with radius 1000.
G01 U1000	Linear Move along axis X. Note that U is an incremental way to refer to axis X.
G02 X3000 Y1000 R1000	Clockwise arc move to [3000;1000] with radius 1000
G01 V-1000	Linear Move along axis Y. Note that V is an incremental way to refer to axis Y.
G02 X2000 Y-1000 I-1000 J0	Clockwise arc move to [2000;-1000] with arc center at incremental [-1000;0].
;G02 X2000 Y-1000 R1000	This is an alternative version of the previous command, but defined with a radius rather than with an arc center. This command has the same effect as the command above.
G01 U-1000	Linear Move along axis X. Note that U is an incremental way to refer to axis X.
G02 X0 Y0 I0 J1000	Clockwise arc move to [0;0] with arc center at incremental [0;1000].
;G02 X0 Y0 R1000	This is an alternative version of the previous command, but defined with a radius rather than with an arc center. This command has the same effect as the command above.
G01 U500 V500	Linear Move of axes X and Y. The axes are referred to in an incremental way as U, V.
;Full Circle	

G02 X500 Y500 I1000 J0 ;**G03** X500 Y500 I1000 J0 **G04** P2.5 **G1011**	Clockwise full circle move to [500;500] with arc center at incremental [1000;0]. Alternative: Counter-Clockwise full circle move to [500;500] with arc center at incremental [1000;0]. Sleep for 2.5 seconds De-energize All Axes **The resulting trajectory:** 

Examples of 3D Helix Moves	
G1010 P0	Energize Axis 0
G1010 P1	Energize Axis 1
G1010 P2	Energize Axis 2
G01 X0 Y0 Z0 F30000	Linear Move Axis 0 and Axis 1 to position [0;0] with speed 30000 Millimeters per Minute
;Version 1: Helix via Circular Move	
G03 X0 Y0 Z2000 I1000 J0 P12	Counter-Clockwise Arc Move of Axes X, Y with Z axis linear. P2 means "two full circles”,
;Version 2: Helix via Two Coordinated Moves	
;**G03** X0 Y0 I1000 J0 P2 **G01** Z2000	G03 Circle Move coordinated with G01 Linear Move
G1011	De-energize All Axes

G04 Dwell / Sleep

The G04 command causes a temporary suspension of execution of any further G-code commands for a prescribed period of time. The program just idles for this time period.

However, the axes are allowed to continue moving during the program’s sleep time. The sleep time may be used to allow the axes to catch up with previously issued G-code commands. Although execution of *new* G-code commands is suspended, the execution of *previous* commands may still be ongoing as the axes have certain physical inertia. Another way to achieve the same result is by using the “Exact Stop” feature.

G04 P2.5	Sleep for 2.5 seconds

G09 Exact Stop

The G09 command enables “Exact Stop” function for a single G-code command that immediately follows the G09 command. This means that the following command’s execution ends only once all participating axes have reached their destinations and stopped within a preconfigured tolerance distance from their targets. The G09 command affects the next G-code command only and does not affect any commands after. For a modal version of the G09 command, look at G61 Enable Exact Stop and G64 Disable Exact Stop.

`G09 G01 X100.50`	Linear Move: axis 0 to a new position with “Exact Stop” function enabled.
`G01 Y-100`	The next motion command is not affected by the G09 command. The second move is done without “Exact Stop”.

G20 Use Inches

The G20 command switches the system from using millimeters to using inches for all motion commands such as Linear Move or Arc/Circle Move. This change includes switching the speed measure from *Millimeters per Minute* to *Inches per Minute*. As a good programming style, the G20 command should be issued at the very beginning of the program to avoid mixing up units in the program.

G21 Use Millimeters

The G21 command reverses the G20 command. The default distance units are millimeters, so this command is only needed whenever G20 is used.

G61 Enable Exact Stop

The G61 command enables “Exact Stop” function in a modal way. This means that all subsequent motion commands are executed with “Exact Stop” until the function is explicitly disabled by a G64 command. This means that execution of each motion command ends only once all participating axes have reached their destinations and stopped within a preconfigured tolerance distance from their targets. This behavior slows down the motion, but improves precision of the trajectory tracking by the axes. By default, the “Exact Stop” feature is disabled for all moves except G00 Rapid Move. Enable the “Exact Stop” feature when it is really needed only.

G64 Disable Exact Stop

The G64 command reverses the G61 command. By default, the “Exact Stop” feature is disabled for all moves with an exception of G00 Rapid Move that always uses “Exact Stop”.

G90 Absolute Positioning

The G90 command reverses the G91 command. The “Incremental Positioning” feature is disabled by default.

G91 Incremental Positioning

The G91 enables “Incremental Positioning” function, also known as relative positioning. When “Incremental Positioning” is activated, the system treats parameters of motion commands such as X, Y, Z, A, B, C, [6], [7],…. etc as relative to the current position of the corresponding axes.

The G91 command is a modal one. This means that all motion commands that follow a G91 command are automatically assumed to have their parameters defined in an incremental way. An alternative to the modal approach is to use U, V, W parameters instead of X, Y, Z when defining motion parameters. The U, V, W parameters are incremental versions of X, Y, Z and allow providing incremental parameters on per-command basis rather than in a modal way.

G93 Speed in Moves per Minute

See G94 Speed in Millimeters per Minute, Inches per Minute, Degrees per Minute

G94 Speed in Millimeters per Minute, Inches per Minute, Degrees per Minute

The G93 and G94 commands define units of motion speed to be used with motion commands such as Linear Move or Arc/Circle Move. The speed units are set in a modal way meaning that the chosen measure of speed is used until the program’s end is reached or another command changes the speed units again. In the essence, these commands define units of speed in which F codes specify speeds for motion commands.

There are two principal ways to define a speed of a coordinated motion:

- ***Moves per Minute*** is a measure of speed applicable to both *rotary* and *linear* axes including complex multi-axes moves that involve axes of both types. This is in contrast with the second type of measures of speed (see *Millimeters per Minute)* that is applicable to either 2-3 Cartesian linear axes, or a *single* rotary axis or a *single* linear axis. The speed measure *Moves per Minute* is a generalized version of *Revolutions per Minute (RPM)*. While *RPM* is concerned with rotations, *Moves per Minute* is a more generic version that can be used to specify speeds of arbitrary linear motions as well as rotary motions including motions of disjoined mechanisms and motions of various combinations of linear and rotary axes. This caters for a unified programming approach.
- ***Millimeters per Minute*** is a “traditional” measure of speed that is primarily designed for linear Cartesian axes such as those found of XYZ-type CNC machines or typical 3D printers, where such speed is called *feed rate* (thus the F parameter). The distance for such speed measurements is computed using a *square root of sum of squares of travels* of the two or three Cartesian linear axes involved in the motion. Note that the formula produces an incorrect result if the axes are not Cartesian ones. This speed measure can be used with Linear Move as well as Arc/Circle

Move assuming that the participating axes are linear Cartesian ones. It is also used with Inverse Kinematics commands that may involve both rotary and linear axes.

There are important variations of the *Millimeters per Minute* measure of speed:

- **Inches per Minute**. It is possible to switch from using millimeters to using inches as measures of distances. In such a case, the speed unit automatically becomes *Inches per Minute* instead of *Millimeters per Minute.*
- ***Degrees per Minute.*** Motion speeds defined in *Millimeters per Minute* are automatically treated as defined in *Degrees per Minute* whenever applied to a ***single rotary axis***. Note that for moves that involved multiple rotary axes the motion speed must be defined using the other units (*Moves per Minute)*, or, otherwise, *inverse kinematics* commands must be used.
- When the *Millimeters per Minute* measure is used to move a *single linear* axis, the speed is defined as speed *along that linear axis*. In other words, a single linear axis is "always a Cartesian one".

The default speed measure is *Millimeters per Minute.*

G1000 Axis Reset

The G1000 command transmits "Reset" commands to the servos of the specified axes. The effects of the command are specified in the servo drive's reference document.

Typical effects are:

- The axes start "free-wheeling" if the mechanics permits.
- Fault Latches are cleared. This can be used as a way to restart after a hardware fault.
- The servo drives perform a "soft reset" routine.
- Firmware-controlled Multi-Turn Work Zone Counts are reset by the servo drives. This does not affect multi-turn encoders.

Refer to a servo drives' reference document for information about what happens with servos upon them receiving a "Reset" command.

`G1000`	`Reset All Axes`
`G1000 P5`	`Reset Axis 5 only`

G1001 Axis Off

The G1001 command transmits "Off" commands to the servos of the specified axes. The effects of the command are specified in the servo drives' reference document.

Typical effects are:

- The axes start “free-wheeling” if the mechanics permits.

`G1001`	`Turn Off All Axes`
`G1001 P5`	`Turn Off Axis 5 only`

G1002 Axis Stop

The G1001 command transmits “Stop” commands to the servos of the specified axes. The effects of the command are specified in the servo drives’ reference document.

Typical effects are:

- The axes’ motion is stopped in a controllable way. The motion is stopped not by friction of “free-wheeling”, but by a counteracting torque applied by the servo drive.
- Upon a controllable stop, the servo drive may apply a brake to the axis. This depends on the configuration of the servo drive. Alternatively, upon reaching a zero speed, the axis may start “free wheeling”.

`G1002`	`Stop All Axes`
`G1002 P5`	`Stop Axis 5 only`

G1010 Axis Capture

The G1010 command energizes servo actuators of specified axes. By default, axes are not energized and therefore are allowed to “free wheel” if the mechanics permits.

The G1010 commands works as the following:

- The system reads the axes’ current positions from the axes’ **encoders**.
- The encoders’ readings are converted into millimeters, inches or degrees depending of the axes’ types and G-code modal settings.
- The system then uses the converted values as references to initiate servo positioning of the axes. The axes’ servos energize and start holding their positions even under influence of external forces. The “free-wheeling” stops.
- If an axis has been already energized, nothing else is done to the axis.

Note that an axis must be “energized” this way before the axis is allowed to be used with motion commands such as G01 Linear Move or G02 Arc/Circle Clockwise Move. Otherwise, a runtime error is reported by the system upon attempting to move an axis that is not energized. Essentially, each G-code program should start from energizing selected axes that are to be used by the program. It is possible to energize all axes at once with a single G1010 command.

The exceptions are the following commands that energize the axes automatically: G00 Rapid Move, G1020 Servo Move. These commands can be used at the beginning of the program instead of G1010 to simultaneously energize and move axes to initial positions. The commands G1030 Speed Command (RPM), G1040 Torque Command also do not require the axes to be energized.

G1010	Energize All Axes
G1010 P5	Energize Axis 5 only

G1011 Axis Release

The G1011 command reverses the effects of the G1010 Axis Capture command.

The G1011 command works as the following:

- The system stops regularly transmitting commands to the servos that actuate the axes referred to by the G1011 command.
- Since the servos stop receiving commands from the motion control system, the servos start to time out within a preconfigured heartbeat-tracking period of time. What happens next with each axis depends on the configuration of the servo drive that actuates the axis. For example, an axis might start "free-wheeling" or, alternatively, the servo might automatically apply a brake to the axis. This is governed by the configuration of the servo drives rather then by motion control system. In the essence, the G1011 command tells the system to "stop commanding" the servos, and the servo drives are left to cope with this on their own. It is recommended that a G1000 Axis Reset, a G1001 Axis Off , or a G1002 Axis Stop command is issued prior to a G1011 command to gracefully shutdown each axis.

G1011	De-Energize All Axes
G1011 P5	De-Energize Axis 5 only

G1020 Servo Move

The G1020 command causes the system to transmit "Servo" positioning commands to the servo drives that actuate specified axes. The axes start moving to the commanded target positions independently of each other at a maximum speed of each axis, not in a coordinated way. Furthermore, the system does not wait until the axes actually reach their destinations and just proceeds with executing the program's next G-code command.

The G1020 command differs from a similar G00 Rapid Move command in the way that "Exact Stop" is not enforced. No waiting for the axis to actually reach their destinations.

If compared with a similar G01 Linear Move command, the G1020 command does not do coordination of the axes' motion. Each of the axes moves at their own maximum speed without taking into account what other axes are doing.

G1020 [0]1000.5 [1]-40.18 [10]90	Move axes 0, 1, 10
G1020 X400.5 Y-119.785 Z100 A90 B-45 C0	Move axes 0, 1, 2, 3, 4, 5. Note that X,Y,Z,A,B,C means [0],[1],[2],[3],[4],[5]

G1030 Speed Command (RPM)

The G1030 command causes the system to transmit an “Electronic Speed Control (RPM)” command to each to the axes refereed to by the command. The axes’ motors start spinning at the specified RPM speed.

Note that servo drive might apply limits on the travels of the axes. This is governed by the servo drive’s own configuration. The system does not wait for the axes to reach their prescribed speeds before proceeding to the next G-code command. The axes may not actually manage to reach the commanded speed at all.

G1030 [0]3000 [1]-1000 [10]12000	Transmits “Speed” commands to axes 0, 1, 10. The speed references are defined in RPM.

G1040 Torque Command

The G1040 command causes the system to transmit an “Electronic Torque Control” command to each to the axes refereed to by the command. The axes’ actuators then apply the prescribes torques to the axes. The system does not wait for the axes to apply their prescribed torques before proceeding to the next G-code command. The axes may not actually manage to reach the commanded torque.

G1040 [0]0.14 [1]-1.2 [10]0.75	Transmits “Electronic Torque Control” commands to axes 0, 1, 10. The torque references are defined in Nm.

G1050 Disable Inverse Kinematics

The G1050 command reverses the effects of a previously issued Txx code and unselects a chosen end effector/tool, thus disabling the Inverse Kinematics function.

G1050	No tool is selected any longer. This disables Inverse Kinematics.
G01 T3 X-59.0 Y-417.5 Z404.5 **G1050 G01** [8]90	Note a G1050 command in the middle of the line. This is a concatenation of two primitive moves: 1. a Linear Move with Inverse Kinematics with tool T3... 2. ...that is coordinated with a second Linear Move of a single axis 8 without Inverse Kinematics. A G1050 command is employed here to disable Inverse Kinematics function for the second motion command within the coordinated move. This is needed because Txx code is modal and thus would have otherwise affected the second motion command.

Axes Naming Convention

In motion commands, axes are referred to either by names or by indices. Note that names X,Y,Z,A,B,C have an equivalent index notation [0],[1],[2],[3],[4],[5]. In other words, axes 0 to 5 have dedicated names and can be referred to either by their indices or by the names. However, all other axes starting from 6 upwards do not have dedicated names and thus have to be referred to by their indices such as [8].

For example, consider two equivalent motion commands:

```
G00 Y3.5 Z-1.2         ;using axes names

;is the same as

G00 [1]3.5 [2]-1.21    ;using axes indices
```

Under default settings, the X,Y,Z,A,B,C parameters simply refer to axes 0,1,2,3,4,5. However, when Inverse Kinematics function is enabled via a Txx code, the system starts treating X,Y,Z & A,B,C parameters as Cartesian coordinates & Euler angles of the selected end-effector tool.

The codes U,V,W provide a way to refer to axes X,Y,Z an incremental way. The U, V, W codes are incremental versions of X,Y,Z codes and allow providing incremental parameters on per-command basis rather than in a modal way. To refer to other axes in the incremental way, use the modal G91 Incremental Positioning code.

Motion Coordination Operators

There are three ways to define coordinated motion of multiple axes in a G-code program:

1. **Primitives.** Use G01 Linear Move, or G02 Arc/Circle Clockwise Move/ G03 Arc/Circle Counter-Clockwise Move commands, the primitives with built-in motion coordination capabilities.

2. **Concatenation Operator over Primitives**. Write multiple primitive motion commands in the same line of code. This causes the motions to start and finish together in a coordinated and synchronized way. The *first motion command* in the line is special as it is used to compute the time needed to complete the motion. All other motion commands from the same line are synchronized with this first command. This operator is a “workhorse” way of creating complex multi-mechanism and multi-axis coordinated motions.

3. **Comma Operator between Primitives.** Use a comma operator “,” to separate motion commands within the same line of code. This causes the motions to start together, but finish at their own time. The system waits until all motion commands within such a line of code finish before proceeding with executing the next line of code.

Consider the following examples illustrating the coordination operators:

G93 F60 **G01** [5]100 [6]200 [7]0 [8]-90	<u>**A Primitive Coordinated Motion Command**</u> The speed is 60.0 Moves per Minute (1.0 move/sec). Move axes 5, 6, 7, 8 to the prescribed positions in a coordinated way. The entire move is going to be completed in 1 second.
G93 F60 **G01** [5]100 [6]200 [7]0 [8]-90 G02 X100 Y100 R50	<u>**Concatenation Operator**</u> The speed is 60.0 Moves per Minute (1.0 move/sec). Note that now two primitive motion commands, a linear and an arc moves, are combined in the same line of code. Linear: Move axes 5, 6, 7, 8 to prescribed positions in a coordinated way. Circular: Move axes X, Y according to an arc trajectory with radius 50 to the prescribed positions. The system orchestrates both linear and circular motions so that the motions of all involved axes start and finish at the same time. The entire coordinated move (linear+circular) is going to be completed in 1 second as prescribed by the G93 F60 codes. The first motion command in the line is used to compute the time needed to complete the motion. All other motion commands from the same line are synchronized with the first command. It is possible to coordinate motions of multiple mechanisms this way, e.g. in a robotic cell or on a conveyor.
G93 **G01** [5]100 [6]200 [7]0 [8]-90 F60**,** **G02** X100 Y100 R50 F120	<u>**Comma Operator**</u> Note the comma “,” between the commands in the same line. Note F parameters in each of the primitive commands. The F codes define different speeds for each individual move. The difference with the previous example is that the primitive linear and circular motions start at the same time as before, but now finish at their own time. The second (circular) motion is going at its own (faster) speed of 120 Motions per Minute (2 motions/second) while the linear motion is going at 60 Motions per Minute (1 motion/second). Thus, the motions finish at different times. The comma “,” operator means that the motions start together, but finish at their own time. The system waits until all motion commands within a line of code finish before proceeding with executing of the next line of code. Without the comma operator, the first motion command in the line of code would have been used as a synchronization reference for the other command in the same line. So, both commands would have finished at the same time.

Inverse Kinematics

Inverse Kinematics is a function that translates G-code motion commands defined in a Cartesian tool space (X,Y,Z and angles A,B,C) into a motion of combination of non-Cartesian rotary and/or linear axes. The automatic translation generates the required Cartesian motion of a tool/end effector at the end of a kinematics chain such as a manipulator or a quadruped’s leg. This function simplifies motion programming since the program code can stay focused on defining a trajectory of end effector(s) in a

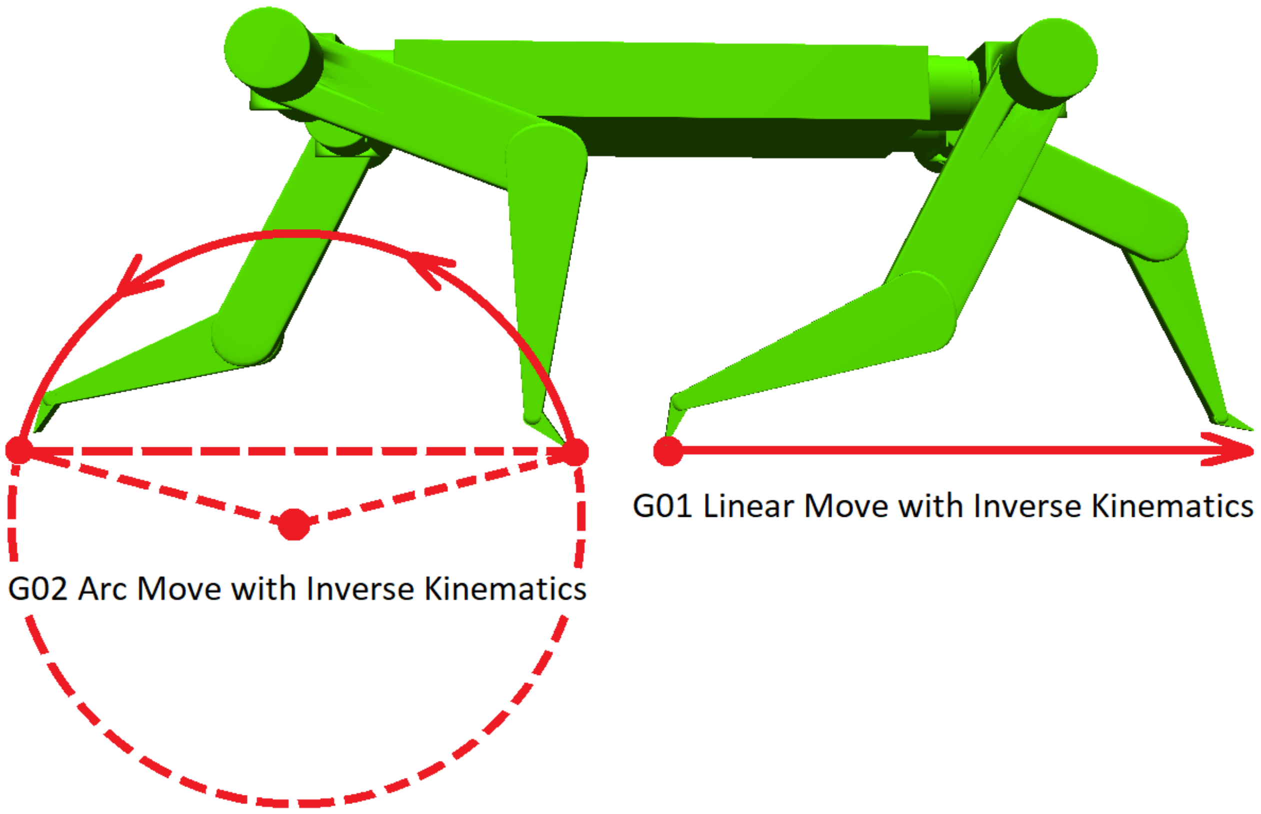
Cartesian space, while leaving it up to the motion control system to figure out how to achieve the desired trajectory(-ies) through a coordinated motion of non-Cartesian axes.

The inverse kinematics function is enabled by specifying a Txx parameter with a motion command such as a G00 Rapid Move, a G01 Linear Move or a G02 Arc/Circle Clockwise Move. The Txx code tells the system which of the end effectors/tools to move according to the motion commands. For example, T3 means “end effector/tool 3”. All end effectors as well as their kinematic chains must be configured in a kinematics model loaded into the motion control system.

A Txx parameter when added to a motion command causes a *significant change* in the meaning of X,Y,Z,A,B,C motion parameters. In normal circumstances, the X,Y,Z,A,B,C parameters simply refer to axes 0,1,2,3,4,5 according to a naming convention. However, upon seeing a Txx code, the system starts treating X,Y,Z & A,B,C parameters as Cartesian coordinates & Euler angles of the selected end-effector tool. The inverse kinematics function then automatically converts the tool’s coordinates into axes positions when executing the command. See Example: Robot Dog Gait Generation via Inverse Kinematics.

The Txx code is a modal one. The T code simultaneously selects a tool/end-effector and enables the Inverse Kinematics function. The Inverse Kinematics function stays enabled for subsequent commands until it is explicitly disabled by a G1050 Disable Inverse Kinematics command or until an end of program is reached. The G1050 command reverses the effects of a previously issued Txx code and unselects a chosen end effector/tool, thus disabling the Inverse Kinematics function. It is possible to change selection of the tool dynamically by issuing a new Txx code that supersedes the previous Txx code.

Example: Robot Dog Gait Generation via Inverse Kinematics


G01 Linear Move with Inverse Kinematics
G02 Arc Move with Inverse Kinematics

Example of Inverse Kinematics Moves: Robot Dog Gait Generation	
G1010 ;G94 F180000 ;speed in mm/min G93 F60 ;moves/min or steps/min ;Robot Dog 4-leg Gait Generation ;The 4 legs correspond to end-effectors T3,T6,T9,T12 defined in a kinematics model of the Robot Dog. G01 T3 X-159 Y-400 Z800 G01 T12 X159 Y-400 Z-100 G02 T6 Y-400 Z0 R500 G02 T9 Y-400 Z-800 R500 G01 T6 X159 Y-400 Z800 G01 T9 X-159 Y-400 Z-100 G02 T3 Y-400 Z0 R500 G02 T12 Y-400 Z-800 R500	Energize All Axes Forward leg motion is an arc motion above ground. The leg is not in contact with the ground. Backward leg motion is a linear motion parallel to the ground. The leg is in the contact with the ground. Synchronization of all the linear and arc motions is done using the first motion in each line.

Visit us at

www.servosila.com/en/motion-control

and

YouTube: http://www.youtube.com/user/servosila