Texas Instruments

TI-Innovator™ Rover Commands

Learn more about TI Technology through the online help at education.ti.com/eguide.

Important Information

Except as otherwise expressly stated in the License that accompanies a program, Texas Instruments makes no warranty, either express or implied, including but not limited to any implied warranties of merchantability and fitness for a particular purpose, regarding any programs or book materials and makes such materials available solely on an "as-is" basis. In no event shall Texas Instruments be liable to anyone for special, collateral, incidental, or consequential damages in connection with or arising out of the purchase or use of these materials, and the sole and exclusive liability of Texas Instruments, regardless of the form of action, shall not exceed the amount set forth in the license for the program. Moreover, Texas Instruments shall not be liable for any claim of any kind whatsoever against the use of these materials by any other party.

Contents

TI-Innovator™ Rover Commands Version 1.4

Prerequisite: Use the Send "Connect RV" Command First

The "**CONNECT RV**" command needs to be used first when using the Rover. The "**CONNECT RV**" command configures the TI-Innovator™ Hub software to work with the TI-Innovator™ Rover.

It establishes the connections to the various devices on the Rover – two motors, two encoders, one gyroscope, one RGB LED and one color sensor. It also clears the various counters and sensor values. The optional 'MOTORS' parameter configures only the motors and allows direct control of motors without the additional peripherals.

CONNECT RV - initializes the hardware connections.

- Connects RV and inputs and outputs built into the RV.
- Resets the Path and the Grid Origin.
- Sets the units per meter to default value of 10. Default Grid unit = 10cm.

Named RV Subsystems

The RV object contains several subsystems that are directly addressed by name. These subsystems consist of the wheels, and sensors that let the Rover sense the world.

The subsystems are listed by name in the following table.

Subsystem Name	Description of Subsystem
RV	The RV object as a whole.
RV.COLOR	The tri-color RGB LED on the top surface of the Rover can be controlled through user programs to display any color combination.
RV.COLORINPUT	The color sensor is on the bottom of the Rover and is used to detect the color of the surface.
RV.RANGER	The front-facing ultrasonic distance sensor. Returns measurements in meters. ~10.00 meters means no obstacle was detected.
RV.ENCODERGYRO	The rotary encoders – one on each motor – measure the distance traveled by the Rover. The left and right encoder, coupled with the gyroscope and operating time information.
RV.GYRO	The gyroscope is used to maintain the heading of Rover while it's in motion. It can also be used to measure the change in angle during turns.
RV.MOTOR.L	Left wheel motor and control for direct control (advanced) use.

Subsystem Name	Description of Subsystem
RV.MOTOR.R	Right wheel motor and control for direct control (advanced) use.
RV.MOTORS	Both the LEFT and RIGHT motor, managed as a single object for direct control (advanced) use.

Rover Command Categories

The Rover commands fall into two categories:

1. Queued execution: All of the Rover motion commands – FORWARD, BACKWARD, LEFT, RIGHT, ANGLE – are queued on the TI-Innovator Hub. They may execute at a future time.
2. Immediate execution: Other commands – like the ones to read the sensors or set the RGB LED on the Rover – are executed immediately.

This means that certain statements in your program will execute before statements that appear earlier in the program especially if the latter commands are part of the queued family.

For example, in the program below, the RGB LED will turn RED before the Rover stops moving:

```
Send "SET RV.COLOR 255 0 255" – immediately executed

Send "RV FORWARD 5" – queued command

Send "RV LEFT 45" – queued command

Send "RV RIGHT 90" – queued command

Send "SET RV.COLOR 255 0 0" – immediately executed
```

Example:

To change color after a "FORWARD" movement, use "TIME" parameter with "WAIT".

```
Send "RV FORWARD TIME 5"

WAIT 5

Send "SET RV.COLOR 255 0 255"
```

RV Commands, Code Samples, and Syntax

The following examples show how various commands for the RV are used. Anywhere a **SET** command is used, the **SET** may be left off (optional use).

Code Samples

When you see "**Code Sample**" in a command table, this "**Code Sample**" may be copied and pasted *as is* to send to your graphing calculator to use in your calculations.

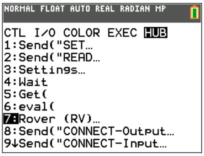
Example:

Code Sample:	`Send ("RV FORWARD 5")`
	`Send ("RV FORWARD SPEED 0.2 M/S TIME 10")`

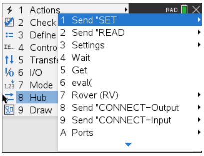
TI-Innovator™ Rover Menu

Rover (RV)…

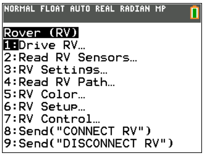
CE Calculators

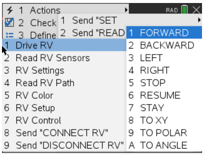


TI-Nspire™ CX



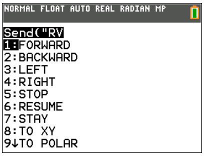
- Drive RV…
- Read RV Sensors…
- RV Settings…
- Read RV Path…
- RV Color…
- RV Setup…
- RV Control…
- Send("CONNECT RV")
- Send("DISCONNECT RV")



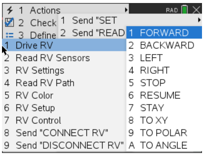


- **Drive RV…**
  - Send("RV
    - FORWARD
    - BACKWARD
    - LEFT
    - RIGHT
    - STOP
    - RESUME
    - STAY
    - TO XY
    - TO POLAR
    - TO ANGLE

CE Calculators

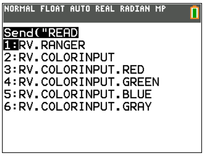


TI-Nspire™ CX

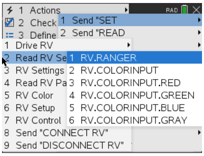


- **Read RV Sensors…**
  - Send"READ"
    - RV.RANGER
    - RV.COLORINPUT
    - RV.COLORINPUT.RED
    - RV.COLORINPUT.GREEN
    - RV.COLORINPUT.BLUE
    - RV.COLORINPUT.GRAY

CE Calculators

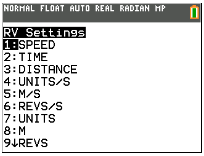


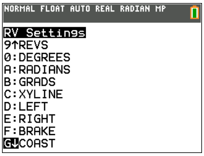
TI-Nspire™ CX



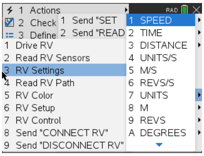
- **RV Settings…**
  - RV Settings
    - SPEED
    - TIME
    - DISTANCE
    - UNIT/S
    - M/S
    - REV/S
    - UNITS
    - M
    - REVS
    - DEGREES
    - RADIANS

CE Calculators





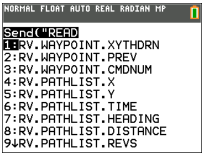
TI-Nspire™ CX

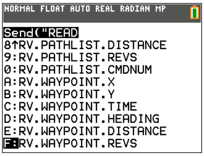


        - GRADS
        - XYLINE
        - LEFT
        - RIGHT
        - BRAKE
        - COAST
        - CW
        - CCW

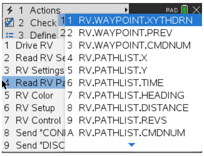
- **Read RV Path…**
    - Send "READ
        - RV.WAYPOINT.XYTHDRN
        - RV.WAYPOINT.PREV
        - RV.WAYPOINT.CMDNUM
        - RV.PATHLIST.X
        - RV.PATHLIST.Y
        - RV.PATHLIST.TIME
        - RV.PATHLIST.HEADING
        - RV.PATHLIST.DISTANCE
        - RV.PATHLIST.REVS
        - RV.PATHLIST.CMDNUM
        - RV.WAYPOINT.X
        - RV.WAYPOINT.Y
        - RV.WAYPOINT.TIME
        - RV.WAYPOINT.HEADING
        - RV.WAYPOINT.DISTANCE
        - RV.WAYPOINT.REVS

CE Calculators



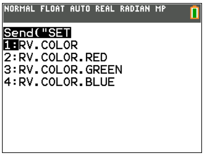


TI-Nspire™ CX

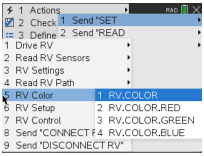


- **RV Color…**
    - Send "SET
        - RV.COLOR
        - RV.COLOR.RED
        - RV.COLOR.GREEN
        - RV.COLOR.BLUE

CE Calculators

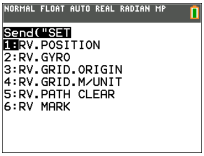


TI-Nspire™ CX

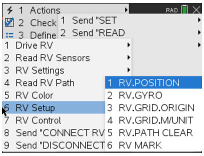


- **RV Setup…**
  - Send "SET
    - RV.POSITION
    - RV.GYRO
    - RV.GRID.ORIGIN
    - RV.GRID.M/UNIT
    - RV.PATH CLEAR
    - RV MARK

CE Calculators

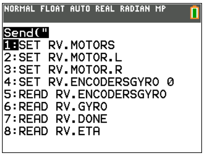


TI-Nspire™ CX

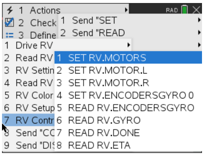


- **RV Control…**
  - Send "
    - SET RV.MOTORS
    - SET RV.MOTOR.L
    - SET RV.MOTOR.R
    - SET RV.ENCODERSGYRO 0
    - READ RV.ENCODERSGYRO
    - READ RV.GYRO
    - READ RV.DONE
    - READ RV.ETA

CE Calculators

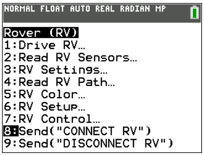


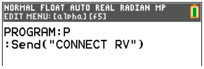
TI-Nspire™ CX



- **Send "CONNECT RV"**
  - Send "CONNECT RV"
    - CONNECT RV

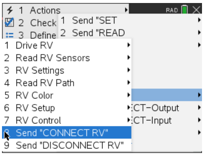
CE Calculators





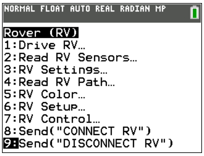
```
PROGRAM:P
:Send("CONNECT RV")
```

TI-Nspire™ CX

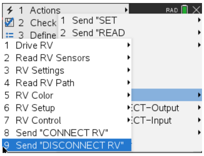


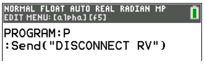
- **Send "DISCONNECT RV"**
  - Send "DISCONNECT RV"
    - DISCONNECT RV

CE Calculators



TI-Nspire™ CX





```
PROGRAM:P
:Send("DISCONNECT RV")
```

Drive RV...

RV Drive Command Families

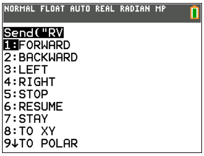
- Base Drive Commands (in the spirit of Turtle Graphics)
    - FORWARD, BACKWARD, RIGHT, LEFT, STOP, STAY
- Math Coordinate Drive Commands
    - Turn to Angle

Note: Drive commands have options for Speed, Time and Distance as appropriate

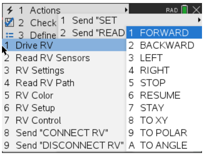
- See RV Settings for Machine-Level Control Commands
    - Set Left and Right Motor values for direction (CW/CCW) and level (0-255,Coast)
    - Read accumulated values for wheel encoder edges and gyro heading change.

- **Drive RV…**
    - Send("RV
        - FORWARD
        - BACKWARD
        - LEFT
        - RIGHT
        - STOP
        - RESUME
        - STAY
        - TO XY
        - TO POLAR
        - TO ANGLE

CE Calculators



TI-Nspire™ CX



RV FORWARD

Command:	**RV FORWARD**
Command Syntax:	**RV FORWARD [[SPEED s] [DISTANCE d] [TIME t]]**

Code Samples:	`Send ("RV FORWARD 0.5 M")`
	`Send ("RV FORWARD SPEED 0.22 M/S TIME 10")`
	`[SET] RV FORWARD`
	`[SET] RV FORWARD [DISTANCE] d [M|UNIT|REV]`
	`[SET] RV FORWARD [DISTANCE] d [M|UNIT|REV]`
	`      SPEED s.ss [M/S|[UNIT/S]|REV/S]`
	`[SET] RV FORWARD [DISTANCE] d [M|UNIT|REV]`
	`      TIME t`
	`[SET] RV FORWARD SPEED s`
	`[M/S|UNIT/S|REV/S]`
	`      [TIME t]`
	`[SET] RV FORWARD TIME t [SPEED s.ss`
	`      [M/S|[UNIT/S]|REV/S]]`

Range:	N/A
Describe:	RV moves forward a given distance (default 0.75 m). Default distance if specified is in UNIT (grid units). Optional M=meters, UNIT=grid-unit, REV=wheel-revolution. Default speed is 0.20 m/sec, max is 0.23 m/sec, min is 0.14 m/sec. Speed may be given and specified in meters/second, unit/second, revolutions/second.
Result:	Action to make the RV move in a forward direction
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV BACKWARD

Command:	**RV BACKWARD**
Command Syntax:	**RV BACKWARD**

Code Sample:	`Send("RV BACKWARD 0.5 M")`
	`Send("RV BACKWARD SPEED 0.22 M/S TIME 10")`
	`[SET] RV BACKWARD`
	`[SET] RV BACKWARD [DISTANCE] d [M|UNIT|REV]`
	`[SET] RV BACKWARD [DISTANCE] d [M|UNIT|REV]`
	`      SPEED s.ss [M/S|[UNIT/S]|REV/S]`
	`[SET] RV BACKWARD [DISTANCE] d [M|UNIT|REV]`
	`      TIME t`
	`[SET] RV BACKWARD SPEED s.ss`
	`      [M/S|UNIT/S|REV/S] [TIME t]`
	`[SET] RV BACKWARD TIME t`
	`      [SPEED s.ss [M/S|UNIT/S|REV/S]]`

Range:	N/A
Describe:	RV moves backward a given distance (default 0.75 m). Default distance if specified is in UNIT (grid units). Optional M=meters, UNIT=grid-unit, REV=wheel-revolution. Default speed is 0.20 m/sec, max is 0.23 m/sec, min is 0.14 m/sec. Speed may be given and specified in meters/second, unit/second, revolutions/second.
Result:	Action to make the RV move in a backward direction.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV LEFT

Command:	**RV LEFT**
Command Syntax:	**RV LEFT**

Code Sample:	`Send "RV LEFT"`
	`[SET] RV LEFT [ddd [DEGREES]]`
	`[SET] RV LEFT [rrr RADIANS]`
	`[SET] RV LEFT [ggg GRADIANS]`

Range:	N/A
Describe:	Default turn is 90 degrees unless DEGREES, RADIANS, or GRADIANS keyword is present, and then the value is converted internally to degrees format from the specified units. Value given is ranged to a value between 0.0 and 360.0 degrees. The turn will be executed as a SPIN motion.
Result:	Turn Rover to the LEFT.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV RIGHT

Command:	**RV RIGHT**
Command Syntax:	**RV RIGHT**

Code Sample:	`Send "RV RIGHT"`
	`[SET] RV RIGHT [ddd [DEGREES]]`
	`[SET] RV RIGHT [rrr RADIANS]`
	`[SET] RV RIGHT [ggg GRADIANS]`

Range:	N/A
Describe:	Default turn is 90 degrees unless DEGREES, RADIANS, or GRADIANS keyword is present, and then the value is converted internally to degrees format from the specified units. Value given is ranged to a value between 0.0 and 360.0 degrees. The turn will be executed as a SPIN motion.

Command:	**RV RIGHT**
Result:	Turn Rover to the RIGHT.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV STOP

Command:	**RV STOP**
Command Syntax:	**RV STOP**

Code Sample:	`Send "RV STOP"`
	`[SET] RV STOP`
	`[SET] RV STOP CLEAR`

Range:	N/A
Describe:	The **RV** will stop any current movement immediately. That movement can be resumed from where it left off with a **RESUME** operation. Any movement commands will cause the queue to flush immediately, and begin the just-posted new movement operation
Result:	Stop processing Rover commands from the command queue, and leave pending operations in the queue. (immediate action). Queue can be resumed by **RESUME**.The **RV** will stop any current movement immediately. That movement can be resumed from where it left off with a **RESUME** operation. Any movement commands will cause the queue to flush immediately, and begin the just-posted new movement operation. Stop processing Rover commands from the command queue, and flush any pending operations left in the queue. (immediate action).
Type or Addressable Component:	Control **Note:** This Rover control command is executed immediately.

RV RESUME

Command:	**RV RESUME**
Command Syntax:	**RV RESUME**

Code Sample:	`Send "RV RESUME"`
	`[SET] RV RESUME`

Range:	N/A
Describe:	Enable processing of Rover commands from the command queue. (immediate action), or resume (see RV STAY) operation.
Result:	Resume operation.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV STAY

Command:	**RV STAY**
Command Syntax:	**RV STAY**

Code Sample:	`Send "RV STAY"`
	`[SET] RV STAY [[TIME] s.ss]`

Range:	N/A
Describe:	Tells RV to "stay" in place for an optionally specified amount of time in seconds. Default is 30.0 seconds.
Result:	RV stays in position.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV TO XY

Command:	**RV TO XY**
Command Syntax:	**RV TO XY** x-coordinate y-coordinate [[SPEED] s.ss [UNIT/S] |M/S|REV/S] [XYLINE]

Code Sample:	`Send "RV TO XY 1 1"`
	`Send "RV TO XY eval(X) eval(Y)"`
	`Send "RV TO XY 2 2 SPEED 0.23 M/S"`

Range:	-327 to +327 for X and Y coordinates
Describe:	This command controls the movement of Rover on a virtual grid. Default location at start of program execution is (0,0) with Rover facing the positive x-axis. The x and y coordinates match the current grid size (default: 0.1 M/grid unit). Grid size can be changed through “SET RV.GRID.M/UNIT“ command The speed parameter is optional.
Result:	Moves Rover from current grid location to the specified grid location.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV TO POLAR

Command:	**RV TO POLAR**
Command Syntax:	**RV TO POLAR** R-coordinate Theta-coordinate [[DEGREES] |RADIANS|GRADS] [[SPEED] s.ss [UNIT/S]|M/S|REV/S] [XYLINE]

Code Sample:	`Send("RV TO POLAR 5 30") - r = 5 units, theta = 30 degrees`
	`Send("RV TO POLAR 5 2 RADIANS")`
	`Send("RV TO POLAR eval(sqrt(3^2+4^2)) eval (tan-1(4/3) DEGREES ")`

Range:	Theta-coordinate: -360 to +360 degrees R-coordinate: -327 to +327
Describe:	Moves the RV from its current position to the specified polar position relative to that position. The RV's X/Y position will be updated to reflect the new position. The “r” coordinate matches the current grid size (default: 0.1 M/grid

Command:	**RV TO POLAR**
	unit). Default location at start of program execution is (0,0) with Rover facing the positive x-axis. Default unit of theta is Degrees. The speed parameter is optional.
Result:	Moves Rover from current grid location to the specified grid location.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV TO ANGLE

Command:	**RV TO ANGLE**
Command Syntax:	**RV TO ANGLE**

Code Sample:	`Send "RV TO ANGLE"`
	
	`[SET] RV TO ANGLE rr.rr`
	`      [[DEGREES]|RADIANS|GRADIANS]`

Range:	N/A
Describe:	
Result:	Spins the RV to the specified angle from current heading.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

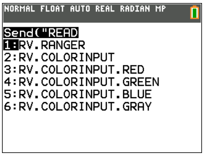
READ RV Sensors...

SEND("Read Sensor Commands

- Reading of low level sensors for learning foundations of robotics.

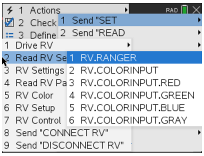
- **Read RV Sensors...**
  - Send("READ
    - RV.RANGER
    - RV.COLORINPUT
    - RV.COLORINPUT.RED
    - RV.COLORINPUT.GREEN
    - RV.COLORINPUT.BLUE
    - RV.COLORINPUT.GRAY

CE Calculators



```
NORMAL FLOAT AUTO REAL RADIAN MP
Send("READ
1:RV.RANGER
2:RV.COLORINPUT
3:RV.COLORINPUT.RED
4:RV.COLORINPUT.GREEN
5:RV.COLORINPUT.BLUE
6:RV.COLORINPUT.GRAY
```

TI-Nspire™ CX



- **RV.RANGER:** Returns value in Meters.
- **RV.COLORINPUT:** Reads color sensor that is built into the RV.

RV.RANGER

<table>
<tr><th>Command:</th><th>RV.RANGER</th></tr>
<tr><td>Command Syntax:</td><td><b>RV.RANGER</b></td></tr>
<tr><td><b>Code Sample:</b></td><td>

```
Send("READ RV.RANGER")
Get(R)
```

<table>
<tr><td>Connects the Rover Vehicle to the TI-Innovator™ Hub. This establishes connections with the motor driver, color sensor, gyroscope, ultrasonic ranger, and proximity sensors.</td><td>CONNECT RV</td></tr>
<tr><td>Returns the current distance from the front of the RV to an obstacle. If there is no obstacle detected, a range of 10.00 meters is reported</td><td>READ RV.RANGER
Get(R)</td></tr>
</table>
</td></tr>
</table>

Command:	**RV.RANGER**
Range:	N/A
Describe:	The front-facing ultrasonic distance sensor. Returns measurements in meters. ~10.00 meters means no obstacle was detected.
Result:	Returns value in Meters.
Type or Addressable Component:	Sensor **Note:** This Rover sensor command is executed immediately.

RV.COLORINPUT

Command:	**RV.COLORINPUT**
Command Syntax:	**RV.COLORINPUT**

Code Sample:	`Send("READ RV.COLORINPUT")`
	`Get(C)`

Range:	1 thru 9
Describe:	Bottom-mounted color sensor detects the color of the surface. Can also detect gray-level scale of black (0) to white (255).
Result:	Returns current color sensor information. The return value is in the 1 – 9 range which maps to the colors below: **Color** **Return value** Red 1 Green 2 Blue 3 Cyan 4 Magenta 5 Yellow 6 Black 7 White 8 Gray 9
Type or	Sensor

Command:	**RV.COLORINPUT**
Addressable Component:	**Note:** This Rover sensor command is executed immediately.

RV.COLORINPUT.RED

Command:	**RV.COLORINPUT.RED**
Command Syntax:	**RV.COLORINPUT.RED**

Code Sample:	`Send("READ RV.COLORINPUT.RED")`
	`Get(R)`

Range:	0 - 255
Describe:	Detect intensity of individual red components of surface. The results are in 0-255 range.
Result:	Returns current color sensor "red value".
Type or Addressable Component:	Sensor **Note:** This Rover sensor command is executed immediately.

RV.COLORINPUT.GREEN

Command:	**RV.COLORINPUT.GREEN**
Command Syntax:	**RV.COLORINPUT.GREEN**

Code Sample:	`Send("READ RV.COLORINPUT.GREEN")`
	`Get(G)`

Range:	0 - 255
Describe:	Detect intensity of individual green components of surface. The results are in 0-255 range.
Result:	Returns current color sensor "green" value.
Type or Addressable Component:	Sensor **Note:** This Rover sensor command is executed immediately.

RV.COLORINPUT.BLUE

Command:	**RV.COLORINPUT.BLUE**
Command Syntax:	**RV.COLORINPUT.BLUE**

Code Sample:	`Send("READ RV.COLORINPUT.BLUE")`
	`Get(B)`

Range:	0 - 255
Describe:	Detect intensity of individual blue components of surface. The results are in 0-255 range.
Result:	Returns current color sensor "blue" value.
Type or Addressable Component:	Sensor **Note:** This Rover sensor command is executed immediately.

RV.COLORINPUT.GRAY

Command:	**RV.COLORINPUT.GRAY**
Command Syntax:	**RV.COLORINPUT.GRAY**

Code Sample:	`Send("READ RV.COLORINPUT.GRAY")`
	`Get(G)`

Range:	0 - 255
Describe:	Detect grayness of surface. The result will be in 0-255 range.
Result:	Returns an interpolated "grayscale" value based on 0.3*red + 0.59*green + 0.11*blue 0-black, 255 - white.
Type or Addressable Component:	Sensor **Note:** This Rover sensor command is executed immediately.

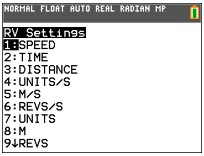
RV Settings...

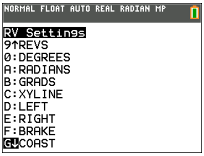
RV Settings Commands

Settings menu for Rover contains other commands that support RV commands such as FORWARD or BACKWARD.

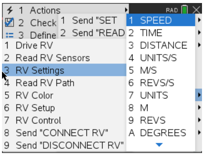
- **RV Settings...**
  - RV Settings
    - SPEED
    - TIME
    - DISTANCE
    - UNIT/S
    - M/S
    - REV/S
    - UNITS
    - M
    - REVS
    - DEGREES
    - RADIANS
    - GRADS
    - XYLINE
    - LEFT
    - RIGHT
    - BRAKE
    - COAST
    - CW
    - CCW

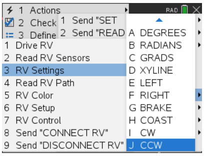
CE Calculators





TI-Nspire™ CX





Read RV Path…

Reading WAYPOINT and PATH

Tracking the RV’s Path

In order to support analysis of the Rover during and after a run, the sketch will automatically measure the following information for each Drive command:

- X Coordinate on virtual grid
- Y Coordinate on virtual grid
- Time in seconds that the current command has been executing.
- Distance in coordinate units for the path segment.
- Heading in degrees (absolute terms measured Counter Clockwise with the X-axis as 0 degrees.
- Revolutions by the wheel in executing the current command
- Command number, tracks the number of commands executed, begins with 0.

The Path values will be stored in lists, starting with the segments associated with the earliest commands and going to the segments associated with the latest commands.

The drive command in progress, the **WAYPOINT**, will repeatedly update the last element in the Path lists as the Rover progresses toward the last waypoint.

When a drive command is completed a new waypoint is initiated and the dimension of the Path lists are incremented.

> **Note:** This implies that when all the drive commands in the queue are completed that another waypoint for the stopped state is automatically started. This is similar to the initial position where the RV is stationary and counting time.

Max number of waypoints: 80

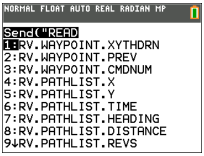
RV Position and Path

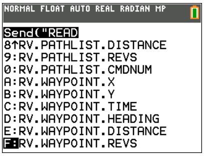
- Ability to read X,Y coordinate, Heading, Time and Distance for each drive command in execution.
- Will store path history in lists for plotting and analysis

Note: Coordinate grid scale can be set by the user, default is 10cm per unit. The user will have options to set the origin of the grid.

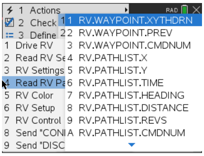
- **Read RV Path…**
  - Send("READ
    - RV.WAYPOINT.XYTHDRN
    - RV.WAYPOINT.PREV
    - RV.WAYPOINT.CMDNUM
    - RV.PATHLIST.X
    - RV.PATHLIST.Y
    - RV.PATHLIST.TIME
    - RV.PATHLIST.HEADING
    - RV.PATHLIST.DISTANCE
    - RV.PATHLIST.REVS
    - RV.PATHLIST.CMDNUM
    - RV.WAYPOINT.X
    - RV.WAYPOINT.Y
    - RV.WAYPOINT.TIME
    - RV.WAYPOINT.HEADING
    - RV.WAYPOINT.DISTANCE
    - RV.WAYPOINT.REVS

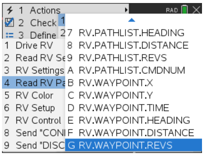
CE Calculators





TI-Nspire™ CX





See Also:

- RV.ETA
- RV.DONE

RV.WAYPOINT.XYTHDRN

Command:	**RV.WAYPOINT.XYTHDRN**
Command Syntax:	**RV.WAYPOINT.XYTHDRN**
Code Sample:	Send("READ RV.WAYPOINT.XYTHDRN")
Example:	Getting the distance traveled toward the current way-point from the last way-point

Code Sample:	Send("READ RV.WAYPOINT.XYTHDRN")
	Get(L_1)
	(L_1)(5)->D

Range:	N/A
Describe:	READ RV.WAYPOINT.XYTHDRN - read the x-coord, y-coord, time, heading, distance traveled, number of wheel revolutions, command number of the current waypoint. Returns a list with all these values as elements.
Result:	Return list of current way-point X, Y coordinates, Time, Heading, Distance, Revolutions, and command number.
Type or Addressable Component:	Returns Data

RV.WAYPOINT.PREV

Command:	**RV.WAYPOINT.PREV**
Command Syntax:	**RV.WAYPOINT.PREV**
Code Sample:	Send("READ RV.WAYPOINT.PREV")
Example:	Getting the distance traveled during the previous way-point.

Code Sample:	Send("READ RV.WAYPOINT.PREV")
	Get(L_1)
	(L_1)(5)->D

Command:	**RV.WAYPOINT.PREV**
Range:	N/A
Describe:	READ RV.WAYPOINT.PREV - read the x-coord, y-coord, time, heading, distance traveled, number of wheel revolutions, command number of the previous waypoint. Returns a list with all these values as elements.
Result:	Return list of the previous way-point X, Y coordinates, time, heading, distance, revolutions, and command number.
Type or Addressable Component:	Returns Data

RV.WAYPOINT.CMDNUM

Command:	**RV.WAYPOINT.CMDNUM**
Command Syntax:	**RV.WAYPOINT.CMDNUM**
Code Sample:	`Send("READ RV.WAYPOINT.CMDNUM")`
Example:	Program to determine if a drive command has completed without referring to a specific command number. **Note:** the **Wait** is intended to increase the probability of catching a difference in the Command Number.

Code Sample:	`Send("RV FORWARD 10")`
	`Send("READ RV.WAYPOINT.CMDNUM")`
	`Get(M)`
	`M->N`
	While M=N
	`Send("READ RV.WAYPOINT.CMDNUM")`
	`Get(N)`
	`End`
	Disp "Drive Command is completed"

Range:	N/A

Command:	**RV.WAYPOINT.CMDNUM**
Describe:	READ RV.WAYPOINT.CMDNUM - returns the last command number of the current waypoint.
Result:	Returns a value of 0 if the RV is currently "working" on a command and is either in motion, or running a STAY operation. This command will return a value of 1 when ALL queued operations are completed, nothing is remaining in the command queue, and the current operation has completed (and immediately after CONNECT RV).
Type or Addressable Component:	Returns Data

See Also: RV.DONE

RV.PATHLIST.X

Command:	**RV.PATHLIST.X**
Command Syntax:	**RV.PATHLIST.X**
Code Samples:	`Send("READ RV.PATHLIST.X")`
Example:	Program to plot the RV path on the graph screen

Code Samples:	Plot1(xyLine, L$_1$, L$_2$,▫,BLUE)
	Send("READ RV.PATHLIST.X")
	Get(L1)
	Send("READ RV.PATHLIST.Y")
	Get(L2)
	DispGraph

Range:	N/A
Describe:	READ RV.PATHLIST.X - returns a list of X values from the beginning to and including the current Waypoint X value.
Result:	Return list of X coordinates traversed since last **RV.PATH CLEAR** or initial **CONNECT RV**.

Command:	**RV.PATHLIST.X**
Type or Addressable Component:	Returns Data

RV.PATHLIST.Y

Command:	**RV.PATHLIST.Y**
Command Syntax:	**RV.PATHLIST.Y**
Code Sample:	`Send("READ RV.PATHLIST.Y")`
Example:	Program to plot the RV path on the graph screen

Code Sample:	Plot1(xyLine, L_1, L_2,▫,BLUE)
	Send("READ RV.PATHLIST.Y")
	Get(L1)
	Send("READ RV.PATHLIST.X")
	Get(L2)
	DispGraph

Range:	N/A
Describe:	READ RV.PATHLIST.Y - returns a list of Y values from the beginning to and including the current Waypoint Y value.
Result:	Return list of Y coordinates traversed since last **RV.PATH CLEAR** or initial **CONNECT RV**.
Type or Addressable Component:	Returns Data

RV.PATHLIST.TIME

Command:	**RV.PATHLIST.TIME**
Command Syntax:	**RV.PATHLIST.TIME**
Code	`Send "READ RV.PATHLIST.TIME"`

Command:	**RV.PATHLIST.TIME**
Sample:	
Range:	N/A
Describe:	READ RV.PATHLIST.TIME - returns a list of the time in seconds from the beginning to and including the current Waypoint time value.
Result:	Return list of cumulative travel times for each successive way-point.
Type or Addressable Component:	Returns Data

RV.PATHLIST.HEADING

Command:	**RV.PATHLIST.HEADING**
Command Syntax:	**RV.PATHLIST.HEADING**
Code Sample:	`Send "READ RV.PATHLIST.HEADING"`
Range:	N/A
Describe:	READ RV.PATHLIST.HEADING - returns a list of the headings from the beginning to and including the current Waypoint heading value.
Result:	Return list of cumulative angular headings taken.
Type or Addressable Component:	Returns Data

RV.PATHLIST.DISTANCE

Command:	**RV.PATHLIST.DISTANCE**
Command Syntax:	**RV.PATHLIST.DISTANCE**
Example:	Getting the cumulative distance traveled since the beginning of a journey by the RV
Code	`Send "READ RV.PATHLIST.DISTANCE"`

Command:	RV.PATHLIST.DISTANCE

Sample:	`Get(`L_1`)`
	`sum(`L_1`)`

Range:	N/A
Describe:	READ RV.PATHLIST.DISTANCE - returns a list of the distances traveled from the beginning to and including the current Waypoint distance value.
Result:	Return list of cumulative distances traveled.
Type or Addressable Component:	Returns Data

RV.PATHLIST.REVS

Command:	RV.PATHLIST.REVS
Command Syntax:	**RV.PATHLIST.REVS**
Code Sample:	`Send "READ RV.PATHLIST.REVS"`
Range:	N/A
Describe:	READ RV.PATHLIST.REVS - returns a list of the number of revolutions traveled from the beginning to and including the current Waypoint revolutions value.
Result:	Return list of wheel revolutions traveled.
Type or Addressable Component:	Returns Data

RV.PATHLIST.CMDNUM

Command:	RV.PATHLIST.CMDNUM
Command Syntax:	**RV.PATHLIST.CMDNUM**

Command:	RV.PATHLIST.CMDNUM
Code Sample:	`Send "READ RV.PATHLIST.CMDNUM"`
Range:	N/A
Describe:	READ RV.PATHLIST.CMDNUM - returns a list of command numbers for the path
Result:	Return list of commands used to travel to the current way-point entry. 0 - Start of Way-points (if first action is a STAY, then no START is given, but a STAY will be shown instead.) 1 - Travel forward 2 - Travel backward 3 - Left spin motion 4 - Right spin motion 5 - Left turn motion 6 - Right turn motion 7 - Stay (no motion) the time the RV stays at the current position is given in the TIME list. 8 - RV is currently in motion on this way-point traversal.
Type or Addressable Component:	Returns Data

RV.WAYPOINT.X

Command:	RV.WAYPOINT.X
Command Syntax:	**RV.WAYPOINT.X**
Code Samples:	`Send("READ RV.WAYPOINT.X")`
Range:	N/A
Describe:	READ RV.WAYPOINT.X - returns x coordinate of current waypoint.
Result:	Return current way-point X coordinate.
Type or Addressable	Returns Data

Command:	RV.WAYPOINT.X
Component:	

RV.WAYPOINT.Y

Command:	RV.WAYPOINT.Y
Command Syntax:	**RV.WAYPOINT.Y**
Code Samples:	`Send("READ RV.WAYPOINT.Y")`
Range:	N/A
Describe:	READ RV.WAYPOINT.Y - returns x coordinate of current waypoint.
Result:	Return current way-point Y coordinate.
Type or Addressable Component:	Returns Data

RV.WAYPOINT.TIME

Command:	RV.WAYPOINT.TIME
Command Syntax:	**RV.WAYPOINT.TIME**
Code Sample:	`Send("READ RV.WAYPOINT.TIME")`
Range:	N/A
Describe:	READ RV.WAYPOINT.TIME - returns time spent traveling from previous to current waypoint
Result:	Return total cumulative way-point travel time value in seconds.
Type or Addressable Component:	Returns Data

RV.WAYPOINT.HEADING

Command:	**RV.WAYPOINT.HEADING**
Command Syntax:	**RV.WAYPOINT.HEADING**
Code Sample:	`Send("READ RV.WAYPOINT.HEADING")`
Range:	N/A
Describe:	READ RV.WAYPOINT.HEADING - returns absolute heading of current waypoint
Result:	Return current absolute heading in degrees. (+h = counter-clockwise, -h = clockwise.)
Type or Addressable Component:	Returns Data

RV.WAYPOINT.DISTANCE

Command:	**RV.WAYPOINT.DISTANCE**
Command Syntax:	**RV.WAYPOINT.DISTANCE**
Code Sample:	`Send("READ RV.WAYPOINT.DISTANCE")`
Range:	N/A
Describe:	READ RV.WAYPOINT.DISTANCE - returns distance traveled between previous and current waypoint
Result:	Return cumulative total distance traveled in meters.
Type or Addressable Component:	Returns Data

RV.WAYPOINT.REVS

Command:	**RV.WAYPOINT.REVS**
Command Syntax:	**RV.WAYPOINT.REVS**
Code Sample:	`Send("READ RV.WAYPOINT.REVS")`
Range:	N/A
Describe:	READ RV.WAYPOINT.REVS - returns number of revolutions needed to travel between previous and current waypoint
Result:	Return total revolutions of the wheels performed to travel the cumulative distance to the current way-point.
Type or Addressable Component:	Returns Data

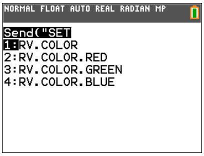
RV Color…

Send("SET Commands

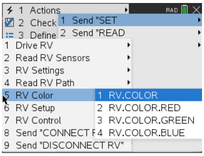
RGB LED on Rover - This supports the same commands and parameters as the RGB LED on the TI-Innovator™ Hub.

- **RV Color…**
  - Send("SET
    - RV.COLOR
    - RV.COLOR.RED
    - RV.COLOR.GREEN
    - RV.COLOR.BLUE

CE Calculators



TI-Nspire™ CX



RV.COLOR

Command:	**RV.COLOR**
Command Syntax:	**RV.COLOR**

Code Sample:	`Send "SET RV.COLOR`
	`[SET] RV.COLOR rr gg bb [[BLINK] b [[TIME] s.ss]]`

Range:	N/A
Describe:	Set the RGB color to be displayed on the Rover's RGB LED. Same syntax as for all RGB LED operations with COLOR, etc.
Result:	Return the current RGB color, as a three-element list, that is being displayed on the Rover's RGB LED
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV.COLOR.RED

Command:	**RV.COLOR.RED**
Command Syntax:	**RV.COLOR.RED**
Code	`Send "SET RV.COLOR.RED`

Command:	**RV.COLOR.RED**
Sample:	`[SET] RV.COLOR.RED rr [[BLINK] b [[TIME] s.ss]]`
Range:	N/A
Describe:	
Result:	Set the RED color to be displayed on the Rover's RGB LED.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV.COLOR.GREEN

Command:	**RV.COLOR.GREEN**
Command Syntax:	**RV.COLOR.GREEN**

Code Sample:	`Send "SET RV.COLOR.GREEN`
	`[SET] RV.COLOR.GREEN gg [[BLINK] b [[TIME] s.ss]]`

Range:	N/A
Describe:	
Result:	Set the GREEN color to be displayed on the Rover's RGB LED.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV.COLOR.BLUE

Command:	**RV.COLOR.BLUE**
Command Syntax:	**RV.COLOR.BLUE**

Command:	**RV.COLOR.BLUE**

Code Sample:	`Send "SET RV.COLOR.BLUE`
	`[SET] RV.COLOR.BLUE bb [[BLINK] b [[TIME] s.ss]]`

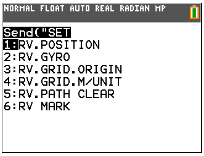
Range:	N/A
Describe:	
Result:	Set the BLUE color to be displayed on the Rover's RGB LED.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV Setup…

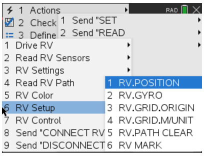
Send("SET Commands

- **RV Setup…**
  - Send("SET
    - RV.POSITION
    - RV.GYRO
    - RV.GRID.ORIGIN
    - RV.GRID.M/UNIT
    - RV.PATH CLEAR
    - RV MARK

CE Calculators



TI-Nspire™ CX



RV.POSITION

Command:	**RV.POSITION**
Command Syntax:	**RV.POSITION**

Code Sample:	`Send "SET RV.POSITION"`
	`[SET] RV.POSITION xxx yyy`
	`[hhh [[DEGREES]|RADIANS|GRADIANS]]`

Range:	N/A
Describe:	Sets the coordinate position and optionally the heading of the Rover on the virtual grid.
Result:	Rover configuration is updated.
Type or Addressable Component:	Setting

RV.GYRO

Command:	**RV.GYRO**
Command Syntax:	**RV.GYRO**
Code Sample:	`Send "SET RV.GYRO"`

Command:	**RV.GYRO**
Range:	N/A
Describe:	Sets the on-board Gyroscope.
Result:	
Type or Addressable Component:	Control (for Gyroscope)

RV.GRID.ORIGIN

Command:	**RV.GRID.ORIGIN**
Command Syntax:	**RV.GRID.ORIGIN**

Code Sample:	`Send "SET RV.GRID.ORIGIN"`
	`[SET} RV.GRID.ORIGIN`

Range:	N/A
Describe:	Sets RV as being at current grid origin point of (0,0). The "heading" is set to 0.0 resulting in the current position of the RV now set to pointing down a virtual x-axis toward positive x values.
Result:	
Type or Addressable Component:	Setting

RV.GRID.M/UNIT

Command:	**RV.GRID.M/UNIT**
Command Syntax:	**RV.GRID.M/UNIT**

Code Sample:	`Send "SET RV.GRID.M/UNIT"`
	`[SET] RV.GRID.M/UNIT nnn`

Command:	**RV.GRID.M/UNIT**
Range:	N/A
Describe:	Set the size of a "grid unit" on the virtual grid. Default is 10 units per meter (100 mm / 10 cm per unit grid). A value of 5 means 5 units per meter or 200 mm / 20 cm per unit grid). A value of 20 means 20 units per meter, or 50 mm / 5 cm per unit grid.
Result:	
Type or Addressable Component:	Setting

RV.PATH CLEAR

Command:	**RV.PATH CLEAR**
Command Syntax:	**RV.PATH CLEAR**

Code Sample:	`Send "SET RV.PATH CLEAR"`
	`[SET] RV.PATH CLEAR`

Range:	N/A
Describe:	Clears any pre-existing path / waypoint information. Recommended before doing a sequence of movement operations where waypoint / path-list information is desired.
Result:	
Type or Addressable Component:	Setting

RV MARK

Command:	**RV MARK**
Command Syntax:	**RV MARK**

Code Sample:	`Send "SET RV MARK"`
	`[SET] RV MARK [[TIME] s.ss]`

Command:	**RV MARK**
Range:	N/A
Describe:	Enable RV to make a "mark" with a pen at the specified time interval (default is 1 second if not specified). A time value of 0.0 turns OFF marking. Marking ONLY happens if the Rover is moving in a forward direction.
Result:	
Type or Addressable Component:	Setting (for Rover)

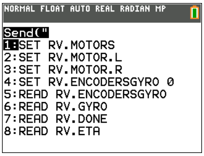
RV Control…

SEND(" Commands

Wheel commands and other commands relevant for learning foundations of the Rover vehicle.

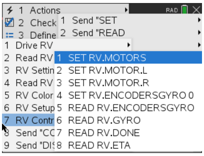
- **RV Control …**
  - Send("
    - SET RV.MOTORS
    - SET RV.MOTOR.L
    - SET RV.MOTOR.R
    - SET RV.ENCODERSGYRO 0
    - READ RV.ENCODERSGYRO
    - READ RV.GYRO
    - READ RV.DONE
    - READ RV.ETA

CE Calculators



```
NORMAL FLOAT AUTO REAL RADIAN MP
Send("
1:SET RV.MOTORS
2:SET RV.MOTOR.L
3:SET RV.MOTOR.R
4:SET RV.ENCODERSGYRO 0
5:READ RV.ENCODERSGYRO
6:READ RV.GYRO
7:READ RV.DONE
8:READ RV.ETA
```

TI-Nspire™ CX



SET RV.MOTORS

Command:	**SET RV.MOTORS**
Command Syntax:	**SET RV.MOTORS**

Code Sample:	`Send "SET RV.MOTORS"`
	`[SET] RV.MOTORS [LEFT][CW|CCW]`
	`<pwm value|BRAKE|COAST>`
	`[RIGHT][CW|CCW]`
	`<pwm value|BRAKE|COAST>`
	`[DISTANCE ddd [M|[UNITS]|REV|FT]]`
	`| [TIME s.ss]`

Range:	N/A
Describe:	Set left or right or both motor PWM values. Negative values imply **CCW** and Positive values imply **CW**. Left **CW**=backward motion. Left **CCW**=forward motion. Right CW=forward motion, Right **CCW**=backward motion. PWM values may be numeric from -255 to +255, or keywords "**COAST**" or "**BRAKE**". Value of 0 is stop (coast). Use of the **DISTANCE** option is only available if the **RV** is connected with all sensors. **CONNECT RV MOTORS** means no sensors are available to measure distance, so the **DISTANCE** option is an error in

Command:	**SET RV.MOTORS**
	this instance.
Result:	Both the LEFT and RIGHT motor, managed as a single object for direct control (advanced) use.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

SET RV.MOTOR.L

Command:	**SET RV.MOTOR.L**
Command Syntax:	**SET RV.MOTOR.L**

Code Sample:	`Send "SET RV.MOTOR.L"`
	`[SET] RV.MOTOR.L [CW|CCW] <+/-pwm value|BRAKE|COAST>`
	`[TIME s.ss] | [DISTANCE ddd [[UNITS] |M|REV|FT]]`

Range:	N/A
Describe:	Set left motor direct PWM value. **CCW** = forward, **CW** = backward, pwm value negative = forward, positive = backward. **TIME** option available in all modes, **DISTANCE** option available only when **RV** is fully connected (not the **RV MOTORS** option).
Result:	Left wheel motor and control for direct control (advanced) use.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

SET RV.MOTOR.R

Command:	**SET RV.MOTOR.R**
Command Syntax:	**SET RV.MOTOR.R**

Code Sample:	`Send "SET RV.MOTOR.R"`
	`[SET] RV.MOTOR.R [CW|CCW] <+/-pwm`

Command:	**SET RV.MOTOR.R**

	`value|BRAKE|COAST>`
	`[TIME s.ss] | [DISTANCE ddd [[UNITS]`
	`|M|REV|FT]]`

Range:	N/A
Describe:	Set right motor direct PWM value. **CW** = forward, **CCW** = backward, pwm value positive = forward, negative = backward. **TIME** option available in all modes, **DISTANCE** option available only when **RV** is fully connected (not the **RV MOTORS** option).
Result:	Right wheel motor and control for direct control (advanced) use.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

SET RV.ENCODERSGYRO 0

Command:	**SET RV.ENCODERSGYRO 0**
Command Syntax:	**SET RV.ENCODERSGYRO 0**
Code Sample:	`Send "SET RV.ENCODERSGYRO 0"`
Range:	N/A
Describe:	Reset the left and right encoder, coupled with the gyro and operating time information.
Result:	
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

READ RV.ENCODERSGYRO

Command:	**READ RV.ENCODERSGYRO**
Command Syntax:	**READ RV.ENCODERSGYRO**

Command:	**READ RV.ENCODERSGYRO**
Code Sample:	`Send "READ RV.ENCODERSGYRO"`
Range:	N/A
Describe:	The left and right encoder, coupled with the gyro and operating time information.
Result:	List of values of current left and right encoder, coupled with gyro and operating time information
Type or Addressable Component:	Control **Note:** This Rover READ command is executed immediately.

READ RV.GYRO

Command:	**READ RV.GYRO**
Command Syntax:	**READ RV.GYRO**

Code Sample:	`Send "READ RV.GYRO"`
	`READ RV.GYRO [[DEGREES]|RADIANS|GRADIANS]`

Range:	N/A
Describe:	The gyroscope is used to maintain the heading of Rover while it’s in motion. It can also be used to measure the change in angle during turns. The gyroscope is ready to use after the **CONNECT RV** command is processed. The GYRO object shall be usable even when the RV is not in motion.
Result:	Returns current gyro sensor angular deviation from 0.0, reading partially drift-offset compensated.
Type or Addressable Component:	Control **Note:** This Rover READ command is executed immediately.

READ RV.DONE

Command:	**READ RV.DONE**
Command Syntax:	**READ RV.DONE**
Code Sample:	`Send("READ RV.DONE")`
Example:	**RV.DONE** as an alias for **RV.WAYPOINT.CMDNUM**

Code Sample:	`For n,1,16`
	`Send "RV FORWARD 0.1"`
	`Send "RV LEFT"`
	`EndFor`
	`@ Wait for Rover to finish driving`
	`Send "READ RV.DONE"`
	`Get d`
	`While d=0`
	`Send "READ RV.DONE"`
	`Get d`
	`Wait 0.1`
	`EndWhile`
	`Send "READ RV.PATHLIST"`
	`Get L`

Range:	N/A
Describe:	**RV.DONE** as an alias for **RV.WAYPOINT.CMDNUM** To improve usability a new state variable was created called **RV.DONE**. This is an alias of **RV.WAYPOINT.CMDNUM**.
Result:	
Type or Addressable Component:	Returns Data

See Also: RV.WAYPOINT.CMDNUM

READ RV.ETA

Command:	**READ RV.ETA**
Command Syntax:	**READ READ RV.ETA**
Code Sample:	`Send("READ RV.ETA")`
Example:	The code sample below returns the estimated time to drive to coordinate (4,4)

Code Sample:	`Send "RV TO XY 4 4"`
	`Send "READ RV.ETA"`
	`Get eta`
	`Disp eta`

	Note: This value will not be exact. It will depend on the surface for one, but it will be a close enough estimate for the expected applications. The value will be time in seconds with a minimum unit of 100 ms.
Example	If a different **READ** command is issued, the value of the variable is overwritten with the information that was requested.

Code Sample:	`Send "RV TO XY 3 4"`
	`Send "READ BRIGHTNESS"`
	`Get eta`

	Note: eta - will contain the value of the **BRIGHTNESS** sensor, not the **RV.ETA** variable
Range:	N/A
Describe:	Calculate the estimated time to complete each Rover command.
Result:	
Type or Addressable Component:	Returns Data

Sample program:

Set **RGB** to red while moving forward, green when turning.

Code Sample:

```
For n, 1, 4
Send "RV FORWARD"
Send "READ RV.ETA"
Get eta
Send "SET COLOR 255 0 0"
Wait eta
Send "RV LEFT"
Send "READ RV.ETA"
Get eta
Send "SET COLOR 0 255 0"
Wait eta
EndFor
```

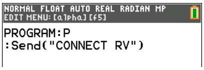
Send "CONNECT RV"

SEND("CONNECT RV") Commands

CONNECT RV - initializes the hardware connections.

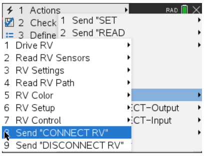
- Connects RV and inputs and outputs built into the RV.
- Resets the Path and the Grid Origin.
- Sets the units per meter to default value.
- **Send("CONNECT RV")**

CE Calculators



```
NORMAL FLOAT AUTO REAL RADIAN MP
EDIT MENU: [alpha] [f5]
PROGRAM:P
:Send("CONNECT RV")
```

TI-Nspire™ CX



CONNECT RV

Command:	**CONNECT RV**
Command Syntax:	CONNECT RV [MOTORS]

Code Sample:	`Send "CONNECT RV"`
	`Send "CONNECT RV MOTORS"`

Range:	N/A
Describe:	The "**CONNECT RV**" command configures the TI-Innovator™ Hub software to work with the TI-Innovator™ Rover. It establishes the connections to the various devices on the Rover – two motors, two encoders, one gyroscope, one RGB LED and one color sensor. It also clears the various counters and sensor values. The optional 'MOTORS' parameter configures only the motors and allows direct control of motors without the additional peripherals.
Result:	Connects the Rover Vehicle to the TI-Innovator™ Hub. This establishes connections with the motor driver, color sensor, gyroscope, ultrasonic ranger, and RGB LED. The Rover is now ready to be programmed
Type or Addressable Component:	All components of the Rover - two motors, two encoders, one gyroscope, one RGB LED and one color sensor.

Send "DISCONNECT RV"

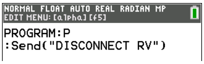
SEND("DISCONNECT RV") Commands

DISCONNECT RV - disconnects all the hardware peripherals from the Hub.

Format: Send("DISCONNECT RV")

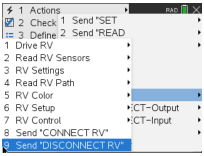
- **Send("DISCONNECT RV")**

CE Calculators



```
PROGRAM:P
:Send("DISCONNECT RV")
```

TI-Nspire™ CX



DISCONNECT RV

Command:	**DISCONNECT RV**
Command Syntax:	DISCONNECT RV

Code Sample:	`Send "DISCONNECT RV"`
	`DISCONNECT RV`

Range:	N/A
Describe:	The "**DISCONNECT RV**" command removes the logical connections between the TI-Innovator™ Hub and the TI-Innovator™ Rover. It also clears the counters and sensor values. It allows the use of the breadboard port of the TI-Innovator™ Hub with other devices.
Result:	The TI-Innovator™ Hub is now logically disconnected from the TI-Innovator™ Rover
Type or Addressable Component:	N/A