TI-Innovator™ Technology with Python Guidebook

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.

Learning More with the TI-Innovator™ Technology with Python eGuide

Parts of this document refer you to the TI-Innovator™ Technology with Python eGuide for more details. The eGuide is a Web-based source of TI-Innovator™ information, including:

- Programming with the TI CE Family of Graphing Calculators and TI-Nspire™ Technology, including sample programs.
- Available TI-RGB Array and its commands.
- Available TI-Innovator™ Rover and its commands.
- Link to update the TI-Innovator™ Sketch software.
- Free classroom activities for TI-Innovator™ Hub.

Actual products may vary slightly from provided images.

もくじ

Pythonインターフェース

このガイドブックは，電卓ベースのPythonモジュールについて説明します。そのモジュールは次の2つで，TI-Innovator™エコシステムをサポートします。

- ti_hub_module
- ti_rover module

Pythonは，次のTIグラフ電卓で利用できます。

- TI-83 Premium CE Edition Python
- TI-84 Plus CE Python
- TI-84 Plus CE-T Python Edition

Pythonは，次のTI-Nspire™製品で利用できます。

ハンドヘルド

- TI-Nspire™ CX II
- TI-Nspire™ CX II CAS
- TI-Nspire™ CX II-T
- TI-Nspire™ CX II-T CAS
- TI-Nspire™ CX II-C
- TI-Nspire™ CX II-C CAS

ディスクトップソフトウェア

- TI-Nspire™ CX Premium Teacher Software
- TI-Nspire™ CX CAS Premium Teacher Software
- TI-Nspire™ CX Student Software
- TI-Nspire™ CX CAS Student Software

PythonコマンドをTI-Innovator™ Hub

TI Hubモジュールti_hubのメニューを使って，プログラムを作成・編集します。コマンド作成には時間を節約し，正確なコマンドのスペルと構文を支援します。

コマンド表にPythonコード例がある場合，これをそのままコピーして貼り付け，グラフ電卓に送信して使うことができます。

例

Python コード例	`light.on()`
	`light.off()`
	`sound.tone(440,4)`
	`b = brightness.measurement()`
	`print(b)`

Note: Hubメニューからコマンドを作成するには，次のことを知っておく必要があります。

- オンボードスピーカーのSOUNDなど，アドレス可能コンポーネントの一意の名前。
- 音の周波数や時間など，コンポーネントに適用されるコマンドパラメータ。一部のパラメータはオプションで，パラメータ値の範囲を知る必要がある場合があります。

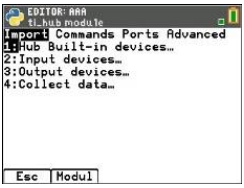
Note: Pythonコマンドは，CE電卓とTI-Nspire™CXIIでは同じです。ただし，CE電卓ではセンサと外部デバイスのそれぞれに独自のモジュールがあります。

スクリーンショットは参照用に提供されています。実際のメニューは，提供されている画像と若干異なる場合があります。

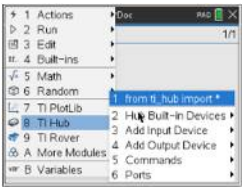
TI-HUBメニュー

- Hub Built-in devices

CE Calculators



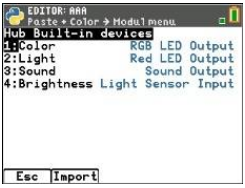
TI-Nspire™CX



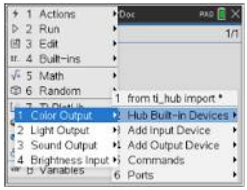
Built-inメニュー

- Color
- Light
- Sound
- Brightness

CE Calculators



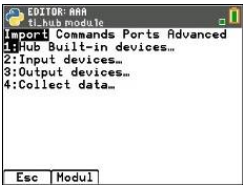
TI-Nspire™CX



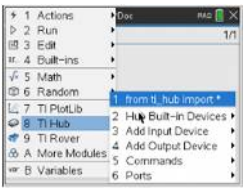
Inputメニュー

- Hub Input devices

CE Calculators

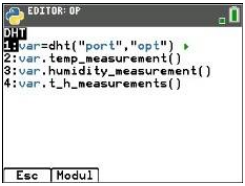


TI-Nspire™CX

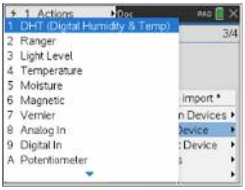


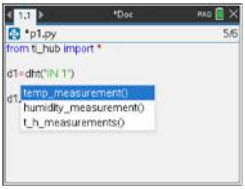
Input DHT

CE Calculators



TI-Nspire™CX





- var = dht("port", "opt")
- var.temp_measurement()
- var.humidity_measurement()
- var.t_h_measurement()

Input Ranger

- var=ranger("port")
- var.measurement()

Input LightLevel

- var=ranger("port")
- var.measurement()
- var.range(min,max)

Input Temp

- var=temperature("port")
- var.measurement()

Input Moisture

- var=moisture("port")
- var.measurement()
- var.range(min,max)

Input Magnetic

- var=magnetic("port")
- var=magnetic_close()
- var.measurement()

- var.trigger(level)

Input Vernier

- var=vernier("port","type")
- var.measurement()
- var.calibrate(a,b)
- var.calibrate(a,b,c,r)

Input Analog In

- var=analogin("port")
- var.measurement()
- var.range(min,max)

Input Digital In/Out

- var=digital("port")
- var.measurement()
- var.set(val)
- var.off()
- var.on()

Input Potentiometer

- var=potentiometer("port")
- var.measurement()
- var.range(min,max)

Input Thermistor

- var=thermistor("port")
- var.measurement()
- var.calibrate(c1,c2,c3,r)

Input Loudness

- var=loudness("port")
- var.measurement()
- var.range(min,max)

Input Color Input

- var=color_input("bb_port")
- var.color_number()
- var.red()
- var.green()
- var.blue()
- var.gray()

Input BB Port (Breadboard Port)

- var=bb_port("mask")
- var.read_port()
- var.read_port(mask)
- var.write_port(value)
- var.write_port(value, mask)

Input Hub Time

- var=hub_time()
- var.measurement()
- var.reset_time()

Input TI-RGB Array

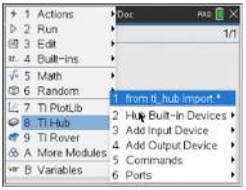
- var=rgb_array()
- var.set(led_position, red, green, blue
- var.set_all(red, green, blue
- var.all_off()
- var.pattern(value)
- var.measurement()

Outputメニュー

• Hub Output devices

CE Calculators

TI-Nspire™ CX

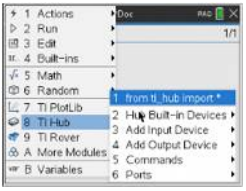


Output LED

- var=led("port")
- var.off()
- var.on()
- var.blink(freq,time)

CE Calculators

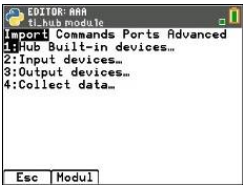
TI-Nspire™ CX



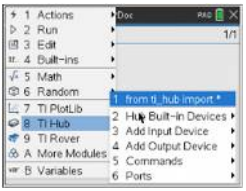
Output RGB

- var=rgb("port")
- var.rgb(r,g,b)
- var.blink(freq,time)
- var.off()

CE Calculators



TI-Nspire™ CX

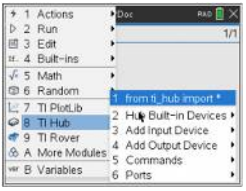


Output Speaker

- var=speaker("port")
- var.tone(freq,time)
- var.note(“note”,time))

CE Calculators

TI-Nspire™ CX

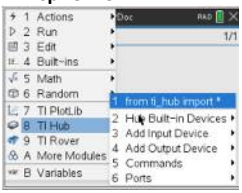


Output Power

- var=power("port")
- var.set(value)
- var.off()
- var.on()

CE Calculators

TI-Nspire™ CX

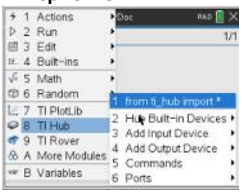


Output Continuous Servo

- var=continuous_servo("OUT 3")
- var.set_cw(speed,time)
- var.set_ccw(speed,time)
- var.stop()

CE Calculators

TI-Nspire™ CX

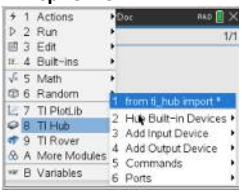


Output Analog Out

- var =analog_out ("port")
- var.set(value)
- var.off()
- var.on()

CE Calculators

TI-Nspire™ CX

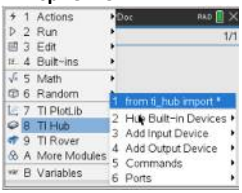


Output Vibration Motor

- var=vibration_motor("OUT 3")
- var.set(value)
- var.off()
- var.on()

CE Calculators

TI-Nspire™ CX

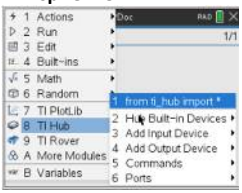


Output Relay

- var=relay("OUT 3")
- var.off()
- var.on()

CE Calculators

TI-Nspire™ CX

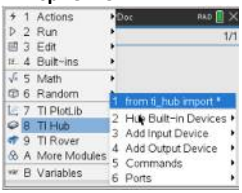


Output Servo

- var=servo("OUT 3")
- var.set_position(pos)
- var.zero()

CE Calculators

TI-Nspire™ CX

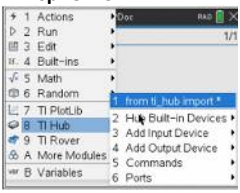


Output Squarewave

- var=squarewave("port")
- var.set(freq,duty,time)
- var.off()

CE Calculators

TI-Nspire™ CX



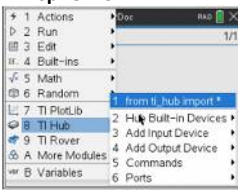
Output Digital in|out

var=digital("port")

- var.measurement()
- var.set(value)
- var.off()
- var.on()

CE Calculators

TI-Nspire™ CX



Built-in(内蔵)

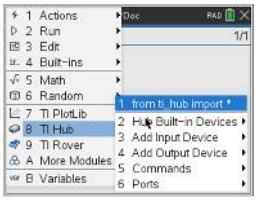
内蔵デバイス

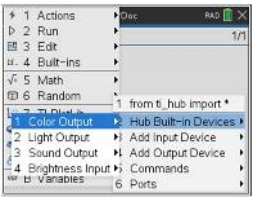
TI-Innovator™ Hubには，LED，RGB LED，スピーカー，輝度センサの4つのデバイスが組み込まれています。
追加のデバイスは，INポートとOUTポートを介してHubに接続できます。

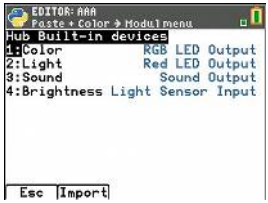
CE Calculators

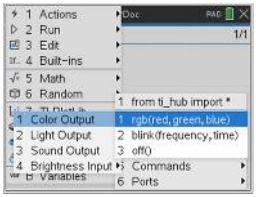
TI-Nspire™ CX II

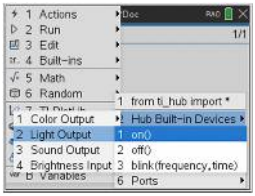
Menu

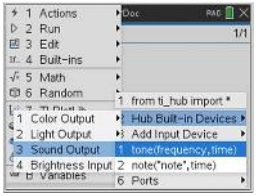


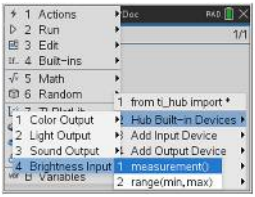








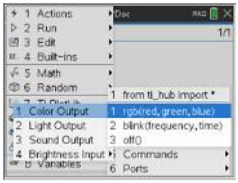




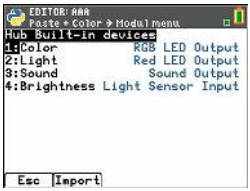
Color(色)

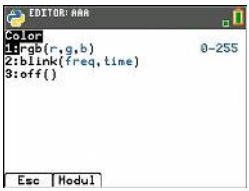
このオブジェクトは，TI-Innovator™ HubのオンボードRGB LEDを制御します。

TI-Nspire CX II: `from ti_hub import *`



CE products: `import color`





color.rgb()

コマンド	color.rgb()
構文	color.rgb(red, green, blue)
範囲	red = [0, 255] green = [0, 255] blue = [0, 255]
説明	オンボード**RGB LED**を制御します。 **例**: `color.rgb(255,0,255)` – RGB LEDの色を紫(赤と青)に設定 **Note:** 3つすべてを255に設定すると，RGB LEDが明るい白色に変わります。すべて0はそれをオフにします。
結果	オンボードRGB LEDを値で指定された色に設定します。
タイプまたはアドレス可能コンポーネント	制御

color.blink()

コマンド	color.blink()
構文	color.blink(frequency, time)
範囲	frequency = [0.1, 20] time = [0.1, 100]

説明	オンボード RGB LED の点滅頻度と持続時間を設定します。
	例:
	`color.rgb(255,0,255) # Set LED to purple`
	`color.blink(4, 5) # Make it blink 4 times a second for 5 seconds`
	Note: 点滅機能を呼び出す前に，色を指定します。

結果	該当なし
タイプまたはアドレス可能コンポーネント	制御

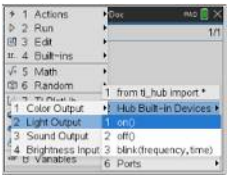
color.off()

コマンド	color.off
構文	color.off()
範囲	
説明	オンボード RGB LED をオフにします。 これは color.rgb(0,0,0)と同じです。
結果	LED が消灯します。
タイプまたはアドレス可能コンポーネント	制御

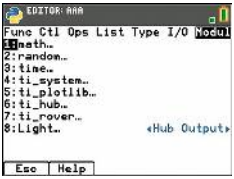
光(Light)

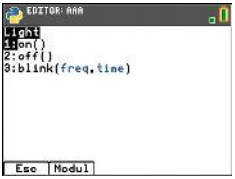
このオブジェクトは，TI-Innovator™ Hubのオンボード LEDを制御します。

TI-Nspire CX II: from ti_hub import *



CE products: `import light`





light.on()

コマンド	light.on()
構文	light.on()
範囲	
説明	オンボードのデジタルRED LEDをオンにします。
結果	LIGHTをオンにします。
タイプまたはアドレス可能コンポーネント	制御

light.off()

コマンド	light.off()
構文	light.off
範囲	
説明	オンボードのデジタルRED LEDをオフにします。
結果	LIGHTをオフにします。
タイプまたはアドレス可能コンポーネント	制御

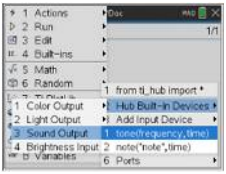
light.blink()

コマンド	light.blink()
構文	light.blink(frequency, seconds)
範囲	Frequency: 0.1〜20Hz Time: 0.1〜100s
説明	オンボードデジタルLEDの点滅頻度と持続時間を設定します。 **例**: `light.blink(2, 5) # Blink the LED 2 times a second for 5 seconds`
結果	
タイプまたはアドレス可能コンポーネント	制御

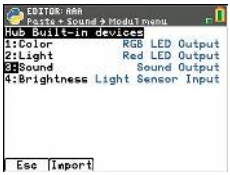
Sound(音)

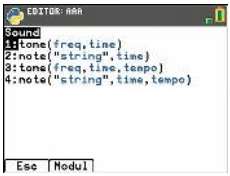
このオブジェクトは，TI-Innovator™ Hubのオンボードスピーカーを制御します。

TI-Nspire CX II: `from ti_hub import *`



CE products: `import sound`





sound.tone()

コマンド	sound.tone()
構文	sound.tone(frequency, [time])
範囲	frequency = [0, 8000] time = [0.1100]
説明	**SOUND**はオンボードスピーカーであり，指定された周波数のサウンドを生成できます。時間が指定されていない場合，既定値で1秒間再生されます。
結果	オンボードスピーカーからトーンを再生します。
タイプまたはアドレス可能コンポーネント	制御

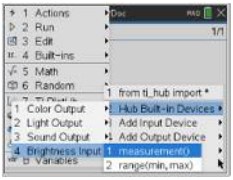
sound.note()

コマンド	sound.note()
構文	sound.note("note", [time])
範囲	note: 再生する音を指定する弦です。 note名は, "C", "CS", "D", "DS", "E", "F", "FS", "G", "GS", "A", "AS", "B"。 そしてオクターブ数は1から9までの範囲です(両端を含む)。 time = [0.1100]
説明	**SOUND**はオンボードスピーカーであり，指定された音を出すことができます。時間が指定されていない場合，サウンドは既定値で1秒間再生されます。 **例**: sound.note(“C5”, 2)
結果	オンボードスピーカーからnoteを再生します。
タイプまたはアドレス可能コンポーネント	制御

Brightness sensor(輝度センサ)

このオブジェクトはTI-Innovator™ Hubの輝度センサへのインターフェースです。
これにより，プログラムは輝度センサから値を読み取ることができます。

TI-Nspire CX II: `from ti_hub import *`



CE products: `import brightns`

brightness.measurement()

コマンド	TI-Nspire CX II: brightness.measurement() CE products: brightns.measurement()
構文	brightness.measurement()
範囲	0 - 100

説明	オンボードの周囲の光センサから現在の内部読み取り値を返します。
	例 TI-Nspire CX IIプログラム:
	`from ti_hub import *`
	`from time import *`
	`while get_key() != "esc":`
	`  b = brightness.measurement()`
	`  print(b)`
	`  sleep(1)`

結果	オンボードの光センサレベルを読み取ります。
タイプまたはアドレス可能コンポーネント	制御

brightness.range()

コマンド	TI-Nspire CX II: brightness.range() CE products: brightns.range() 上級者用
構文	brightness.range([min, max])
範囲	
説明	ADC入力値のADC 0-16383の範囲からユーザーが選択した範囲へのマッピングを変更/設定します。結果のセンサ読み取り値はこれにマップされ，浮動小数点の結果が返されます。既定値では，オンボードのBRIGHTNESSセンサは0～100の範囲です。
結果	オンボードの輝度/光センサのマッピングを設定します。
タイプまたはアドレス可能コンポーネント	センサ

入力

入力デバイス

センサへのPythonインターフェースでは，センサを表すオブジェクトを作成する必要があります。これには，センサが接続されているTI-Innovator™ Hubのポートの指定が含まれます。

センサの読み取り値は，各センサの.measurement()関数を介して読み取ることができます。

入力デバイスとそれに関連する関数は次のとおりです。

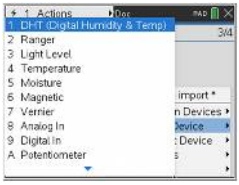
DHT(デジタル湿度と温度)

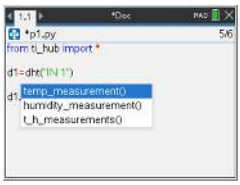
DHT(Digital Humidity & Temp)は，デジタル湿度と温度センサのインターフェースをサポートします。

入力デバイスの追加

項目	説明
DHT	現在の温度，湿度，センサタイプ，最後にキャッシュされた読み取りステータスで構成されるリストを返します。

TI-Nspire CX II: `from ti_hub import *`





オブジェクトを作成する関数をメニューから貼り付けます。

var = dht("port")

コマンド	var = dht("port")
構文	var = dht("port")
範囲	
説明	オブジェクトで使用可能な関数は，センサオブジェクトの変数名の後にピリオド(.)を追加することで表示されます。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.temp_measurement()

コマンド	var.temp_measurement()
構文	var.temp_measurement()

Pythonコード例	**TI Nspire CX II:**
	`from ti hub import *`
	`d1 = dht("IN 1")`
	`t = d1.temp_measurement()`
	`print("Temp is: ", t)`
	CE products:
	`from dht import *`
	`d1 = dht("IN 1", "DHT11")`
	`t = d1.temp measurement()`
	`print("Temp is: ", t)`

範囲	
説明	温度値を摂氏(°C)で返す関数
結果	該当なし
タイプまたはアドレス可能コンポーネント	制御

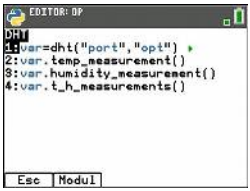
var.humidity_measurement()

コマンド	var.humidity_measurement()
構文	var.humidity_measurement()
範囲	
説明	相対湿度値を返す関数
結果	
タイプまたはアドレス可能コンポーネント	制御

var.t_h_measurements()

コマンド	var.t_h_measurements()
構文	var.t_h_measurements()
範囲	
説明	温度値を摂氏(°C)で返し，相対湿度値を返す関数。2つの浮動小数点値を返します。
結果	
タイプまたはアドレス可能コンポーネント	制御

CE products: `from dht import *`



var=dht("port","opt")

コマンド	var=dht("port","opt")
構文	var=dht("port","opt")
範囲	
説明	この関数は，DHTセンサの変数varで指定されたオブジェクトを作成します。 “port” – DHTセンサが接続されているポート “opt” – 2つの異なるタイプのDHTセンサ，DHT11(既定値)とDHT22, の使用を可能にするオプション
結果	
タイプまたはアドレス可能コンポーネント	制御

var.temp_measurement()

コマンド	var.temp_measurement()
構文	var.temp_measurement()

Pythonコード例	**TI Nspire CX II:**
	`from ti_hub import *`
	`d1 = dht("IN 1")`
	`t = d1.temp_measurement()`
	`print("Temp is: ", t)`
	CE products:
	`from dht import *`
	`d1 = dht("IN 1", "DHT11")`
	`t = d1.temp_measurement()`
	`print("Temp is: ", t)`

範囲	
説明	温度値を摂氏(°C)で返す関数
結果	該当なし
タイプまたはアドレス可能コンポーネント	制御

var.humidity_measurement()

コマンド	var.humidity_measurement()
構文	var.humidity_measurement()
範囲	
説明	相対湿度値を返す関数
結果	
タイプまたはアドレス可能コンポーネント	制御

var.t_h_measurements()

コマンド	var.t_h_measurements()
構文	var.t_h_measurements()
範囲	
説明	温度値を摂氏(°C)で返し，相対湿度値を返す関数。2つの浮動小数点値を返します。
結果	
タイプまたはアドレス可能コンポーネント	制御

Ranger

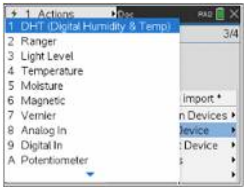
RANGERは超音波距離センサをサポートするモジュールです。

入力デバイスの追加

項目	説明
Ranger	指定された超音波距離センサRangerからの現在の距離の測定値を返します。

CE products: `from ranger import *`

TI-Nspire CX II: `from ti_hub import *`



オブジェクトを作成する関数をメニューから貼り付けます。

var=ranger("port")

コマンド	var=ranger("port")
構文	var=ranger("port")
範囲	
説明	**“port”**に接続されたRangerのオブジェクトを作成します。.
結果	
タイプまたはアドレス可能コンポーネント	制御

var.measurement()

コマンド	var.measurement()
構文	var.measurement()

Pythonコード例	**TI Nspire CX II:**
	`from ti_hub import *`
	`r1 = dht("IN 2")`
	`d = r1.measurement()`
	`print("Distance is: ", d)`
	CE products:
	`from ranger import *`
	`r1 = dht("IN 2")`
	`d = r1.measurement()`
	`print("Distance is: ", d)`

範囲	
説明	センサによって測定された距離をメートル単位で返します。
結果	
タイプまたはアドレス可能コンポーネント	制御

LightLevel (光センサ)

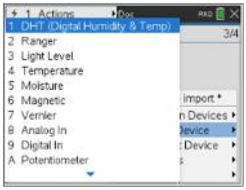
LIGHTLEVEL は，外部光レベルセンサの読み取りをサポートします。

入力デバイスの追加

項目	説明
Light Level	外光レベル(輝度)センサから明るさレベルを返します。

CE products: `from lightlvl import *`

TI-Nspire CX II: `from ti_hub import *`



オブジェクトを作成する関数をメニューから貼り付けます。

var=light_level("port")

コマンド	var=light_level("port")
構文	var=light_level("port")
範囲	
説明	**“port”**に接続された光レベルセンサのオブジェクトを作成します。.
結果	
タイプまたはアドレス可能コンポーネント	制御

var.measurement()

コマンド	var.measurement()
構文	var.measurement()

Pythonコード例	**TI Nspire CX II:**
	`from ti_hub import *`
	`l1 = light level("IN 1")`
	`b = l1.measurement()`
	`print("Light level is: ", b)`
	CE products:
	`from lightlvl import *`
	`l1 = light level("IN 1")`
	`b = l1.measurement()`
	`print("Light level is: ", b)`

範囲	
説明	センサによって測定された0～100の相対的な輝度を返します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.range(min,max)

コマンド	var.range(min,max)
構文	var.range(min,max)
範囲	
説明	センサの戻り値の範囲を設定します。これを有効にするには, var.measurement()関数の前に呼び出す必要があります。
結果	
タイプまたはアドレス可能コンポーネント	制御

Moisture(水分)

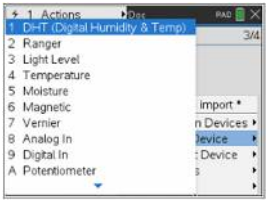
MOISTUREは，土壌水分センサをサポートするモジュールです。

入力デバイスの追加

項目	説明
Moisture	水分センサの読み取り値を返します。

CE products: `from moisture import *`

TI-Nspire CX II: `from ti_hub import *`



オブジェクトを作成する関数をメニューから貼り付けます。

var=moisture("port")

コマンド	**var=moisture("port")**
構文	var=moisture("port")
範囲	
説明	**“port”**に取り付けられた水分センサのオブジェクトを作成します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.measurement()

コマンド	var.measurement()
構文	var.measurement()

Pythonコード例	**TI Nspire CX II:**
	`from ti_hub import *`
	`m1 = moisture("IN 1")`
	`m = m1.measurement()`
	`print("Moisture is: ", m)`
	`m1 = moisture("IN 1")`
	`m = m1.measurement()`
	`print("Moisture is: ", m)`
	CE products:
	`from moisture import *`
	`m1 = moisture("IN 1")`
	`m = m1.measurement()`
	`print("Moisture is: ", m)`

範囲	
説明	センサによって測定された水分値を返します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.range(min,max)

コマンド	var.range(min,max)
構文	var.range(min,max)
範囲	既定値範囲: 0- 16383
説明	センサの戻り値の範囲を設定します。これを有効にするには, var.measurement()関数の前に呼び出す必要があります。
結果	
タイプまたはアドレス可能コンポーネント	制御

Magnetic(磁気)

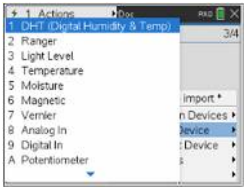
磁気センサは，磁石がセンサに近づいたことを検出するため使います。
MAGNETICは，磁気ホール効果センサをサポートするモジュールです。

入力デバイスの追加

項目	説明
Magnetic	磁場の存在を検出します。 フィールドの存在を判別するしきい値は，trigger()関数を介して設定されます。 しきい値の既定値は150です。

CE products: `from magnetic import *`

TI-Nspire CX II: `from ti_hub import *`



オブジェクトを作成する関数をメニューから貼り付けます。

var=magnetic("port")

コマンド	var=magnetic("port")
構文	var=magnetic("port")
範囲	
説明	“port”に取り付けられた磁気センサのオブジェクトを作成します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var=magnetic_close()

コマンド	var=magnetic_close()
構文	var=magnetic_close()
範囲	

説明	センサの近くで磁石が検出された場合は，0または1を返します。 0 =磁石が検出されました 1 =磁石が検出されません トリガー値は，var.trigger()関数を使って構成できます。 トリガーの既定値は150です。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.measurement()

コマンド	var.measurement()
構文	var.measurement()

Pythonコード例	**TI Nspire CX II:**
	`from ti_hub import *`
	`m1 = magnetic("IN 1")`
	`mg = m1.measurement()`
	`print("Magnetic strength mc = is: ", mg)`
	`m1.magnet_close()`
	`if (mc == 0):`
	`print("Magnet detected")`
	`else`
	`print("No magnet detected")`
	CE products:
	`from magnetic import *`
	`m1 = magnetic("IN 1")`
	`mg = m1.measurement()`
	`print("Magnetic strength is: ", mg)`
	`mc = m1.magnet_close()`
	`if (mc == 0):`
	`print("Magnet detected")`
	`else`
	`print("No magnet detected")`

範囲	範囲: 0～16383
説明	センサ付近の磁場の強さを表す値を返します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.trigger(level)

コマンド	var.trigger(level)
構文	var.trigger(level)
範囲	**level**の範囲は0～16383です。 既定値は150です。
説明	var.magnet_close()関数のトリガーレベルを設定するため使います。
結果	
タイプまたはアドレス可能コンポーネント	制御

Vernier(TI-SensorLinkで使用)

Vernierオブジェクトは，Vernierセンサからデータを収集するためTI-SensorLinkアダプタで使用されます。

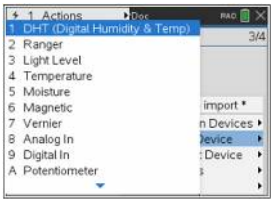
VERNIERは，TI-SensorLinkを介してVernierアナログセンサをセットアップ，そして読み取ることができるインターフェースをサポートしています。

入力デバイスの追加

項目	説明
Vernier	コマンドで指定されたVernierアナログセンサから値を読み取ります。 このコマンドは，次のVernierセンサをサポートしています。 • **temperature** – ステンレススチール温度センサ • **lightlevel** – TI光センサ • **pressure** – 旧ガス圧力センサ • **pressure** – 新ガス圧力センサ. • **pH** – pHセンサ • **force10** – ±10N設定，力センサ • **force50** – ±50N設定，力センサ • **accelerometer** – 低加速度計 • **generic** – 上記でサポートされていない他のセンサの設定，および方程式の係数を設定するため上記のcalibrate()APIの使用を許可します。

CE products: `from vernier import *`

TI-Nspire CX II:`from ti_hub import *`



オブジェクトを作成する関数をメニューから貼り付けます。

var=vernier("port","type")

コマンド	var=vernier("port","type")
構文	var=vernier("port","type")
範囲	

説明	この関数は，portに接続されたtypeで指定されたセンサを表すVernirオブジェクトを作成します。サポートされているVernierセンサは次のとおりです。 • ステンレススチール温度センサ(“temperature”) • pHセンサ(“pH”) • ガス圧力センサ(“pressure”，“pressure2”) • 力センサ(“force10”，“force50”) • 光センサ(“lightlevel”) • 低加速度計(“accelerometer”) これらすべてのセンサについて，var.measurement()関数はキャリブレーションされたセンサ値を返します。 一般のVernierセンサもサポートされています。これは，上記のリストにないVernierアナログセンサからデータを取得するため使用できます。センサオブジェクトは，適切な値を取得するため適切な係数でキャリブレーションする必要があります。キャリブレーションがない場合，var.measurement()関数は0～16383の，加工されていない電圧値を返します。
結果	
タイプまたはアドレス可能コンポーネント	

var.measurement()

コマンド	**var.measurement()**
構文	var.measurement()

Pythonコード例	**TI Nspire CX II:**
	`from ti_hub import *`
	`v1 = vernier("IN 2", "pH")`
	`pHval = v1.measurement()`
	`print("pH is: ", pHval)`
	CE products:
	`from vernier import *`
	`v1 = vernier("IN 2", "pH")`
	`pHval = v1.measurement()`
	`print("pH is: ", pHval)`

範囲	
説明	センサデータを返します。
結果	

タイプまたはアドレス可能コンポーネント	制御

var.calibrate(a,b)

コマンド	var.calibrate(a,b)
構文	var.calibrate(a,b)
範囲	
説明	この関数，上記のリストにないgeneric(汎用)Vernierセンサに必要になります。線形ax＋bセンサをキャリブレーションします。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.calibrate(a,b,c,r)

コマンド	var.calibrate(a,b,c,r)
構文	var.calibrate(a,b,c,r)
範囲	
説明	この関数はサーミスタスタイルのSteinhartセンサをキャリブレーションするために必要です。
結果	
タイプまたはアドレス可能コンポーネント	制御

Analog In(アナログ入力)

このオブジェクトは，はっきりとサポートされていないアナログセンサとのインターフェースに使用できます。

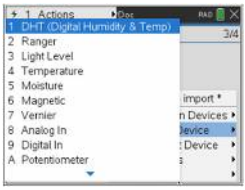
ANALOG.INは，アナログ入力汎用デバイスの使用をサポートしています。

入力デバイスの追加

項目	説明
Analog In	アナログ入力汎用デバイスの使用をサポートします。

CE products: `from analogin import *`

TI-Nspire CX II:`from ti_hub import *`



オブジェクトを作成する関数をメニューから貼り付けます。

var=analog_in("port")

コマンド	**var=analog_in("port")**
構文	var=analog_in("port")
範囲	
説明	**“port”**に接続されたアナログ入力センサのオブジェクトを作成します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.measurement()

コマンド	var.measurement()
構文	var.measurement()

Pythonコード例	**TI Nspire CX II:**
	`from ti_hub import *`
	`an1 = analog_in("BB 5")`
	`val = an1.measurement()`
	`print("Analog In value: ", val)`
	CE products:
	`from analogin import *`
	`an1 = analog_in("BB 5")`
	`val = an1.measurement()`
	`print("Analog In value: ", val)`

範囲	
説明	センサデータ値を返します。既定値の範囲は0～16383です。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.range(min,max)

コマンド	var.range(min,max)
構文	var.range(min,max)
範囲	
説明	センサの戻り値の範囲を設定します。これを有効にするには，関数var.measurement()の前に呼び出す必要があります。
結果	
タイプまたはアドレス可能コンポーネント	制御

Digital In/Out(デジタル入力/出力)

このオブジェクトは，はっきりとサポートされていないデジタルデバイスとのインターフェースに使用できます。同じオブジェクトがデジタル入力とデジタル出力で機能します。

DIGITALは，デジタル入力/出力ピンを制御するためのインターフェースを提供します。

入力デバイスの追加

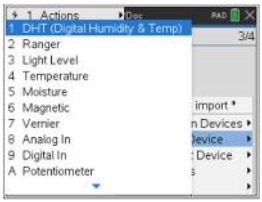
項目	説明
Digital In	DIGITALオブジェクトに接続されているデジタルピンの現在の状態，またはオブジェクトに最後に設定されたデジタル出力値のキャッシュされた状態を返します。

出力デバイスの追加

項目	説明
Digital Out	デジタル出力を制御するためのインターフェース • **set(val)**：デジタル出力をval(0または1)で指定された値に設定します。 • **on()**：デジタル出力の状態をhigh(1)に設定します。 • **off()**：デジタル出力の状態をlow(0)に設定します。

CE products: `from digital import *`

TI-Nspire CX II: `from ti_hub import *`



オブジェクトを作成する関数をメニューから貼り付けます。

var=digital("port")

コマンド	var=digital("port")
構文	var=digital("port")
範囲	
説明	このオブジェクトは，はっきりとサポートされていないデジタルデバイスとのインターフェースに使用できます。

結果	
タイプまたはアドレス可能コンポーネント	制御

var.measurement()

コマンド	var.measurement()
構文	var.measurement()

Pythonコード例	**TI Nspire CX II:**
	`from ti_hub import *`
	`d1 = digital("IN 1")`
	`val = d1.measurement()`
	`print("Value is: ", val)`
	CE products:
	`from digital import *`
	`d1 = digital("IN 1")`
	`val = d1.measurement()`
	`print("Value is: ", val)`

範囲	
説明	デジタルセンサを読み取り，0.0または1.0を返します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.set(val)

コマンド	var.set(val)
構文	var.set(val)
範囲	
説明	デジタルデバイスを0または1に設定します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.off()

コマンド	var.off()
構文	var.off()
範囲	
説明	var.set(0)に相当します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.on()

コマンド	var.on()
構文	var.on()
範囲	
説明	var.set(1)に相当します。
結果	
タイプまたはアドレス可能コンポーネント	制御

Potentiometer(ポテンショメータ)

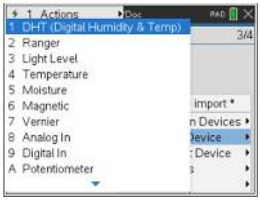
POTENTIOMETERは，汎用の可変抵抗/ポテンショメータをサポートするインターフェースです。

入力デバイスの追加

項目	説明
Potentiometer	ポテンショメータセンサをサポートします。 センサの範囲は，関数range()で変更できます。

CE products: `from potentio import *`

TI-Nspire CX II: `from ti_hub import *`



オブジェクトを作成する関数をメニューから貼り付けます。

var=potentiometer("port")

コマンド	var=potentiometer("port")
構文	var=potentiometer("port")
範囲	
説明	ポテンショメータセンサのオブジェクトを作成します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.measurement()

コマンド	var.measurement()
構文	var.measurement()

Pythonコード例	**TI Nspire CX II:**
	`from ti_hub import *`
	`p1 = potentiometer("IN 1")`
	`r = p1.measurement()`
	`print("Value is: ", m)`
	CE products:
	`from potentio import *`
	`p1 = potentiometer("IN 1")`
	`r = p1.measurement()`
	`print("Value is: ", r)`

範囲	
説明	ポテンショメータの測定値を返します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.range(min,max)

コマンド	var.range(min,max)
構文	var.range(min,max)
範囲	
説明	ポテンショメータの戻り値の範囲を設定します。これを有効にするには，関数var.measurement()の前に呼び出す必要があります。
結果	
タイプまたはアドレス可能コンポーネント	制御

Thermistor(サーミスタ)

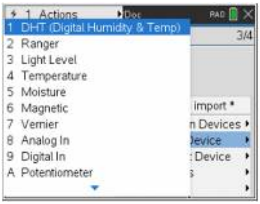
THERMISTORは，サーミスタセンサの読み取りをサポートします。

入力デバイスの追加

項目	説明
Thermistor	サーミスタセンサを読み取ります。 既定値の係数は，TI-Innovator™ Hubのブレッドボードパックに含まれているサーミスタと一致するように設計されており，10KΩの固定抵抗を使った場合に使用されます。 サーミスタの新しいキャリブレーション係数と基準抵抗のセットは，関数calibrate()を使って構成できます。

CE products: `from thermist import *`

TI-Nspire CX II: `from ti_hub import *`



オブジェクトを作成する関数をメニューから貼り付けます。

var=thermistor("port")

コマンド	var=thermistor("port")
構文	var=thermistor("port")
範囲	
説明	サーミスタセンサのオブジェクトを作成します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.measurement()

コマンド	var.measurement()
構文	var.measurement()

Pythonコード例	**TI Nspire CX II:**
	`from ti_hub import *`
	`t1 = thermistor("IN 1")`
	`val = t1.measurement()`
	`print("Thermistor value: ", val)`
	CE products:
	`from thermist import *`
	`t1 = thermistor("IN 1")`
	`val = t1.measurement()`
	`print("Thermistor value: ", val)`

範囲	
説明	サーミスタの測定値を返します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.calibrate(c1,c2,c3,r)

コマンド	var.calibrate(c1,c2,c3,r)
構文	var.calibrate(c1,c2,c3,r)
範囲	
説明	指定された値でサーミスタをキャリブレーションします。
結果	
タイプまたはアドレス可能コンポーネント	制御

Loudness(音の大きさ)

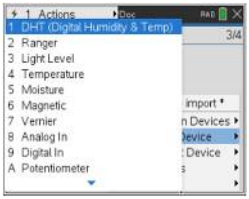
LOUNDESS入力センサは，サウンドラウドネスセンサのルーチンをサポートします。

入力デバイスの追加

項目	説明
Loudness	サウンドラウドネスセンサをサポートします。

CE products: `from loudness import *`

TI-Nspire CX II: `from ti_hub import *`



オブジェクトを作成する関数をメニューから貼り付けます。

var=loudness("port")

コマンド	var=loudness("port")
構文	var=loudness("port")
範囲	
説明	サウンドラウドネスセンサのオブジェクトを作成します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.measurement()

コマンド	var.measurement()
構文	var.measurement()

Pythonコード例	**TI Nspire CX II:**
	`from ti_hub import *`
	`l1 = loudness("IN 1")`
	`ld_val = l1.measurement()`
	`print("Loudness value is: ", ld_val)`
	CE products:
	`from loudness import *`
	`l1 = loudness("IN 1")`
	`ld_val = l1.measurement()`
	`print("Loudness value is: ", ld_val)`

範囲	
説明	ラウドネスセンサの測定値を返します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.range(min,max)

コマンド	var.range(min,max)
構文	var.range(min,max)
範囲	
説明	ラウドネスセンサの戻り値の範囲を設定します。これを有効にするには，関数var.measurement()の前に呼び出す必要があります。
結果	
タイプまたはアドレス可能コンポーネント	制御

Color Input(カラー入力)

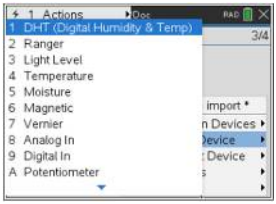
COLORINPUTは，I2C接続カラー入力センサへのインターフェースを提供します。

入力デバイスの追加

項目	説明
Color Input	I2C接続カラー入力センサへのインターフェースを提供します。 I2Cポートに加えてbb_portピンを使って，カラーセンサのLEDを制御します。 • **color_number():** センサが検出している色を表す1から9までの値を返します。 数字は次のマッピングごとの色を表しています。 1：Red(赤) 2：Green(緑) 3：Blue(青) 4：Cyan(シアン，青緑色) 5：Magenta(マゼンタ，赤紫色) 6：Yellow(黄色) 7：Black(黒) 8：White(白) 9：Gray(灰色) • **red():** 検出されている赤のカラーレベルの強度を表す0～255の値を返します。 • **green():** 検出されている緑のカラーレベルの強度を表す0～255の値を返します。 • **blue():** 検出されている青のカラーレベルの強度を表す0～255の値を返します。 • **gray():** 検出されている灰色のレベルを表す0～255の値を返します。ここで，0は黒，255は白です。

CE products: `from colorinp import *`

TI-Nspire CX II: `from ti_hub import *`



オブジェクトを作成する関数をメニューから貼り付けます。

var=color_input("bb_port")

コマンド	var=color_input("bb_port")
構文	var=color_input("bb_port")

Pythonコード例	**TI Nspire CX II:**
	`from ti_hub import *`
	`c1 = color_input("BB 2")`
	`value = c1.color_number()`
	`if (value == 1):`
	`··print("Red!")`
	`else:`
	`··print("Not red")`
	CE products:
	`from colorinp import *`
	`c1 = color_input("BB 2")`
	`value = c1.color_number()`
	`if (value == 1):`
	`··print("Red!")`
	`else:`
	`··print("Not red")`

範囲	
説明	color_inputセンサのオブジェクトを作成します。センサはI2Cポートに接続されており，bb_portはセンサのLEDを制御するピンのブレッドボードポートのピンです。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.color_number()

コマンド	var.color_number()
構文	var.color_number()
範囲	
説明	カラーセンサから1～9の値を返します。 1：Red(赤) 2：Green(緑) 3：Blue(青) 4：Cyan(シアン，青緑色) 5：Magenta(マゼンタ，赤紫色) 6：Yellow(黄色)

	7：Black(黒) 8：White(白) 9：Gray(灰色)
結果	
タイプまたはアドレス可能コンポーネント	制御

var.red()

コマンド	var.red()
構文	var.red()
範囲	
説明	センサの下の色の赤成分を表す0～255の値を返します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.green()

コマンド	var.green()
構文	var.green()
範囲	
説明	センサの下の色の緑成分を表す0～255の値を返します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.blue()

コマンド	var.blue()
構文	var.blue()
範囲	
説明	センサの下の色の青成分を表す0～255の値を返します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.gray()

コマンド	var.gray()
構文	var.gray()
範囲	
説明	センサの下の色の灰色成分を表す0～255の値を返します。
結果	
タイプまたはアドレス可能コンポーネント	制御

BB Port (ブレッドボードポート)

BBPORT(Breadboard Port)は，10個すべてのBBポートピンをデジタル入力/出力ポートの組み合わせとして使用するサポートを提供します。

入力デバイスの追加

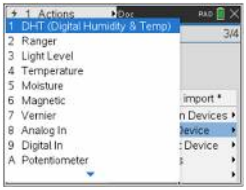
項目	説明
BBPort	10個のBBポートピンすべてをデジタル入力/出力ポートの組み合わせとして使うためのサポートを提供します。 初期化関数には，10ピンのサブセットの使用を可能にするオプションのマスク(mask)パラメータがあります。 • **read_port()**: BBポートの入力ピンの現在の値を読み取ります。 • **write_port(value)**: 出力ピンの値を指定された値に設定します。値は0～1023です。マスクが指定されている場合，値はvar = bbport(mask)操作のマスク値に対しても調整されることに注意します。

出力デバイスの追加

項目	説明
BB Port	TI-RGB Arrayをプログラミングするための関数を提供します。上記の詳細を参照してください。

CE products: `from bbport import *`

TI-Nspire CX II: `from ti_hub import *`



オブジェクトを作成する関数をメニューから貼り付けます。

var=bb_port("mask")

コマンド	var=bb_port("mask")
構文	var=bb_port("mask")
範囲	
説明	ブレッドボードポート(bb_port)のオブジェクトを作成します。 maskは，0～1023の値を持つオプションのパラメータであり，特定

	のピンを選択的に接続するために使われます。マスク値は，10進数，2進数，16進数の形式で指定できます。 たとえば，1023または0X3FFは10ピンすべてを選択し，既定値の内部マスク値です。これは，マスクが指定されていない場合にbb_port()オブジェクトによって使われます。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.read_port()

コマンド	var.read_port()
構文	var.read_port()

Pythonコード例	**TI Nspire CX II:**
	`from ti_hub import *`
	`bb = bbport()`
	`readval = bb.read_port()`
	`print("BB Port value is: ", readval)`
	`bb.write_port(0x55)`
	CE products:
	`from bbport import *`
	`bb = bbport()`
	`readval = bb.read_port()`
	`print("BB Port value is: ", readval)`
	`bb.write_port(0x55)`

範囲	
説明	BBポートの入力ピンの電流値を読み取ります。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.read_port(mask)

コマンド	var.read_port(mask)
構文	var.read_port(mask)
範囲	
説明	BBポートの入力ピンの電流値を読み取ります。 maskを使うと，入力として読み取るピンを指定できます。指定されていない場合の既定値：オブジェクトの初期化中に指定された接続ピ

	ンマスク。このマスクは有効範囲についても検証され，接続されたピンマスクに対して内部的に論理ANDされます。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.write_port(value)

コマンド	var.write_port(value)
構文	var.write_port(value)
範囲	
説明	出力ピンの値を指定された値に設定します。値は0～1023です。マスクが指定されている場合，値はvar=bbport(mask)操作のマスク値に対しても調整されることに注意します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.write_port(value, mask)

コマンド	var.write_port(value, mask)
構文	var.write_port(value, mask)
範囲	
説明	出力ピンの値を指定された値に設定します。値は0～1023です。マスクが指定されている場合，値はvar=bbport(mask)操作のマスク値に対しても調整されることに注意します。 オプションのマスクを使うと，出力として設定するピンのサブセットを指定できます。指定されていない場合の既定値：オブジェクトの初期化中に指定された接続ピンマスク。このマスクは有効範囲についても検証され，接続されたピンマスクに対して内部的に論理ANDされます。
結果	
タイプまたはアドレス可能コンポーネント	制御

Hub Time(ハブタイム)

このオブジェクトは，はっきりとサポートされていないアナログセンサとのインターフェースに使用できます。

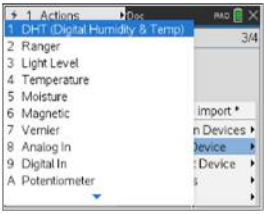
TIMERはTI-Innovator™TIMERオブジェクトへのインターフェースです。

入力デバイスの追加

項目	説明
Hub Time	内蔵ミリ秒タイマーへのアクセスを提供します。

CE products: `from analogin import *`

TI-Nspire CX II: `from ti_hub import *`



オブジェクトを作成する関数をメニューから貼り付けます。

var=hub_time()

コマンド	var=hub_time()
構文	var=hub_time()
範囲	
説明	TI-Innovator™ Hubにタイマーのオブジェクトを作成します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.measurement()

コマンド	var.measurement()
構文	var.measurement()

Pythonコード例	**TI Nspire CX II:**
	`from ti_hub import *`
	`from time import *`
	`t1=hub_time()`
	`t1.reset_time()`
	`print("Before: ", t1.measurement())`
	`sleep(.5)`
	`print("After: ", t1.measurement())`
	CE products:
	`from timer import *`
	`from time import *`
	`t1=hub_time()`
	`t1.reset_time()`
	`print("Before: ", t1.measurement())`
	`sleep(.5)`
	`print("After: ", t1.measurement())`

範囲	
説明	タイマーの現在の値をミリ秒単位で返します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.reset_time()

コマンド	var.reset_time()
構文	var.reset_time()
範囲	
説明	時間をゼロにリセットします。
結果	
タイプまたはアドレス可能コンポーネント	制御

TI-RGB Array

TI-RGB Arrayは，TI-RGB Arrayをプログラミングするインターフェースを制御します。簡単な操作から難しい操作まで扱います。

入力デバイスの追加

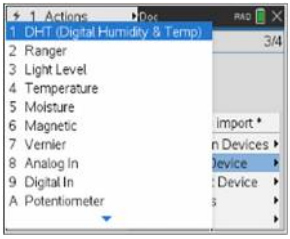
項目	説明
TI-RGB Array	TI-RGBアレイをプログラミングする関数を提供します。 初期化機能は，オプションのLAMPパラメータを受け入れて，外部電源を必要とするTI-RGB Arrayの高輝度モードを有効にします。 • **set(led_position, r,g,b)**：特定のled_position(0-159を指定されたr, g, b値に設定します。ここで，r, g, bは0から255までの値です。 • **set_all(r,g,b)**：Array内のすべてのRGB LEDを同じr, g, b値に設定します。 • **all_off()**：Array内のすべてのRGBをオフにします。 • **measurement()**：RGB arrayがTI-Innovator™から使っているおおよその電流引き込みをミリアンペアで返します。 • **pattern(pattern)**：引数の値を0～65535の範囲のバイナリ値として使って，表現の1の値が存在するピクセルをオンにします。LEDは255のpwmレベル値で赤としてオンになります。

出力デバイスの追加

項目	説明
TI-RGB Array	TI-RGB Arrayをプログラミングするための関数を提供します。

CE products: `from rgb_arr import *`

TI-Nspire CX II: `from ti_hub import *`



オブジェクトを作成する関数をメニューから貼り付けます。

var=rgb_array()

コマンド	var=rgb_array()
構文	var=rgb_array()

Pythonコード例	**TI Nspire CX II:**
	`from ti_hub import *`
	`r1 = rgb_array()`
	`r1.set_all(255,0,0)`
	`c_red = r1.measurement()`
	`r1.set_all(255,255,255)`
	`c_white = r1.measurement()`
	`print("Red LEDs current: ", c_red, "White LEDs current: ", c_white)`
	CE products:
	`from rgb_arr import *`
	`r1 = rgb_array()`
	`r1.set_all(255,0,0)`
	`c_red = r1.measurement()`
	`r1.set_all(255,255,255)`
	`c_white = r1.measurement()`
	`print("Red LEDs current: ", c_red, "White LEDs current: ", c_white)`

範囲	
説明	TI-RGB Arrayのオブジェクトを作成します。 初期化には，TI-RGB ArrayがTI-Innovator™ Hubに接続されていることの確認が含まれます。 TI-RGB Arrayは，入力デバイスと出力デバイスの両方であることに注意します。両方のメニューに含まれています。 オプションのパラメータlampでは，LEDの輝度を上げるために外部バッテリーが必要です。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.set(led_position, red, green, blue)

コマンド	var.set(led_position, red, green, blue)
構文	var.set(led_position, red, green, blue)
範囲	
説明	指定されたled_position(0-15)を，指定されたr, g, b値に設定します。 ここで，r, g, bは0から255までの値です。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.set_all(red, green, blue)

コマンド	var.set_all(red, green, blue)
構文	var.set_all(red, green, blue)
範囲	
説明	Array内のすべてのRGB LEDを同じr, g, b値に設定します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.all_off()

コマンド	var.all_off()
構文	var.all_off()
範囲	
説明	Array内のすべてのRGB LEDをオフにします。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.pattern(value)

コマンド	var.pattern(value)
構文	var.pattern(value)
範囲	
説明	引数の値を0～65535の範囲のバイナリ値として使うと，表現の1の値が存在するピクセルがオンになります。 LEDは赤で点灯します。 **例**：バイナリ0b 11001001である201の値は，LED 0, 3, 67をオンにします。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.measurement()

コマンド	var.measurement()
構文	var.measurement()
範囲	
説明	RGB arrayがTI-Innovatorから使っているおおよその電流引き込みをミリアンペアで返します。
結果	
タイプまたはアドレス可能コンポーネント	制御

出力

出力デバイス

出力デバイスの追加

このメニューには，ti_hubモジュールでサポートされている出力デバイスのリストがあります。すべてのメニュー項目はオブジェクトの名前を貼り付け，デバイスで使われる変数とポートを待ちます。

LED

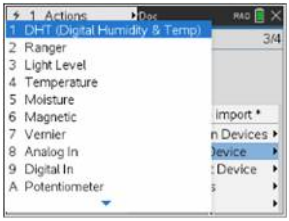
LEDは，外部接続されたLEDを制御するためのインターフェースです。

出力デバイスの追加

項目	説明
LED	外部接続されたLEDを制御するための関数

CE products: `from led import *`

TI-Nspire CX II: `from ti_hub import *`



オブジェクトを作成する関数をメニューから貼り付けます。

var=led("port")

コマンド	var=led("port")
構文	var=led("port")
範囲	
説明	LEDのオブジェクトを作成します。 ポートはOUT 1，OUT 2，OUT 3にすることができます。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.off()

コマンド	var.off()
構文	var.off()
範囲	
説明	LEDをオフにします。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.on()

コマンド	var.on()
構文	var.on()
範囲	
説明	LEDをオンにします。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.blink(freq,time)

コマンド	var.blink(freq,time)
構文	var.blink(freq,time)

Pythonコード例	**TI Nspire CX II:**
	`from ti_hub import *`
	`from time import *`
	
	`l1=led("OUT 1")`
	`for i in range(5):`
	`··l1.on()`
	`··sleep(1)`
	`··l1.off()`
	
	`l1.blink(5,2)`
	CE products:
	`from led import *`
	`from time import *`
	
	`l1=led("OUT 1")`
	`for i in range(5):`
	`··l1.on()`
	`··sleep(1)`
	`··l1.off()`
	
	`l1.blink(5,2)`

範囲	
説明	指定された時間，指定された周波数でLEDを点滅させます。 周波数：0.1～20Hz 時間：0.1～100秒
結果	
タイプまたはアドレス可能コンポーネント	制御

RGB

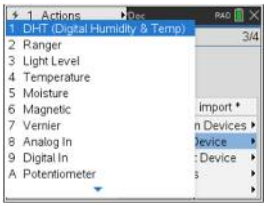
RGBは，外部RGB LEDを制御するためサポートします。

出力デバイスの追加

項目	説明
RGB	外部RGB LEDの制御のサポート

CE products: `from rgb import *`

TI-Nspire CX II: `from ti_hub import *`



オブジェクトを作成する関数をメニューから貼り付けます。

var=rgb("port")

コマンド	var.rgb("port")
構文	var.rgb("port")
範囲	
説明	RGB LEDのオブジェクトを作成します。 ポートはOUT 1，OUT 2，OUT 3にすることができます。 **CE module**：var=rgb("r", "g", "b") – RGB LEDを3つの異なるブレッドボードピンまたは同じOUTポートに接続できます。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.rgb(r,g,b)

コマンド	var.rgb(r,g,b)
構文	var.rgb(r,g,b)
範囲	
説明	RGB LEDの色を，r(赤)，g(緑)，b(青)の値で指定された色に設定します。r, g, bはすべて0から255の間です。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.blink(freq,time)

コマンド	var.blink(freq,time)
構文	var.blink(freq,time)

Pythonコード例	**TI Nspire CX II:**
	`from ti_hub import *`
	`rgb1=rgb("OUT 2")`
	`rgb1.rgb(255,0,255)`
	`rgb1.blink(4,4)`
	CE products:
	`from rgb import *`
	`rgb1=rgb("OUT 2", "OUT 2", "OUT 2")`
	`rgb1.rgb(255,0,255)`
	`rgb1.blink(4,4)`

範囲	
説明	指定された時間，指定された周波数でRGB LEDを点滅させます。 周波数：0.1～20Hz 時間：0.1～100秒 RGB LEDの色を設定するには，最初にtha.t var.rgb()を呼び出す必要があることに注意します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.off()

コマンド	var.off()
構文	var.off()
範囲	
説明	RGB LEDをオフにします。
結果	
タイプまたはアドレス可能コンポーネント	制御

Speaker(スピーカー)

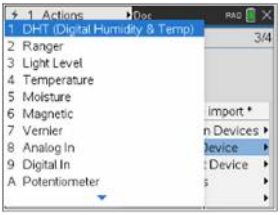
SPEAKERは，TI-Innovator™ Hubで外部スピーカーをサポートするモジュールです。

出力デバイスの追加

項目	説明
Speaker	TI-Innovator™ Hubで外部スピーカーをサポートする関数 機能は上記のsoundと同じです。

CE products: `from speaker import *`

TI-Nspire CX II: `from ti_hub import *`



オブジェクトを作成する関数をメニューから貼り付けます。

var=speaker("port")

コマンド	var=speaker("port")
構文	var=speaker("port")
範囲	
説明	外部スピーカー用のオブジェクトを作成します。 ポートは，OUT 1，OUT 2，OUT 3，または任意のBBポートピンにすることができます。 外部スピーカーには，内蔵スピーカーと同じAPIがあります。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.tone(freq,time)

コマンド	var.tone(freq,time)
構文	var.tone(freq,time)
範囲	
説明	time(時間：0.1～100秒)，freq(周波数：0～8000Hz)で指定されたトーンを再生します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.note(“note”,time)

コマンド	var.note(“note”,time)
構文	var.note(“note”,time)

Pythonコード例	**TI Nspire CX II:**
	`from ti_hub import *`
	`from time import *`
	`sp1=speaker("OUT 1")`
	`sp1.tone(440,2)`
	`sleep(2)`
	`sp1.note("A4",4)`
	CE products:
	`from speaker import *`
	`from time import *`
	`sp1=speaker("OUT 1")`
	`sp1.tone(440,2)`
	`sleep(2)`
	`sp1.note("A4",4)`

範囲	
説明	tme(0.1～100秒)の”note”で指定されたnoteを再生します。 noteには，"C", "CS", "D", "DS", "E", "F", "FS", "G", "GS", "A", "AS", "B"が指定きます。また，オクターブ数は1から9(端を含む)の範囲です。
結果	
タイプまたはアドレス可能コンポーネント	制御

Power(電力)

POWERは，TI-Innovator™ Hubで外部電源を制御するパワー/FET制御インターフェースです。

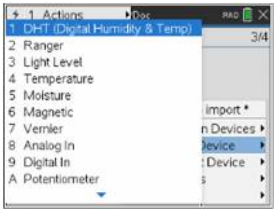
出力デバイスの追加

このメニューには，ti_hubモジュールでサポートされている出力デバイスのリストがあります。すべてのメニュー項目はオブジェクトの名前を貼り付け，デバイスで使われる変数とポートを待ちます。

項目	説明
Power	TI-Innovator™ Hubで外部電源を制御する関数 • **set(value)**：電力レベルを，指定された値(0～100)に設定します。 • **on()**：電力レベルを100に設定します。 • **off()**：電力レベルを0に設定します。

CE products: `from power import *`

TI-Nspire CX II: `from ti_hub import *`



オブジェクトを作成する関数をメニューから貼り付けます。

var=power("port")

コマンド	var=power("port")
構文	var=power("port")
範囲	
説明	TI-Innovatorで外部電源を制御するパワー/FET制御インターフェースのオブジェクトを作成します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.set(value)

コマンド	var.set(value)
構文	var.set(value)

Pythonコード例	**TI Nspire CX II:**
	`from ti_hub import *`
	`from time import *`
	
	`p1=power("OUT 3")`
	
	`# Set 50% power`
	`p1.set(50)`
	`sleep(2)`
	`# Turn off`
	`p1.off()`
	CE products:
	`from power import *`
	`from time import *`
	
	`p1=power("OUT 3")`
	
	`# Set 50% power`
	`p1.set(50)`
	`sleep(2)`
	`# Turn off`
	`p1.off()`

範囲	
説明	出力の状態を“value”(0から100の間)に設定します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.off()

コマンド	var.off()
構文	var.off()
範囲	
説明	電力レベルをゼロに設定します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.on()

コマンド	var.on()
構文	var.on()
範囲	
説明	電力レベルを100に設定します。
結果	
タイプまたはアドレス可能コンポーネント	制御

Continuous Servo(連続サーボ)

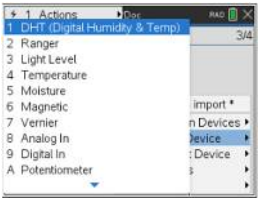
CONTINUOUS SERVOは，連続サーボモーターを制御するインターフェースを提供します。

出力デバイスの追加

項目	説明
Continuous Servo	連続サーボモーターを制御する関数 • **set_cw(speed,time)**：サーボは，指定された速度(0～255)で，秒単位の指定された時間，時計回りに回転します。 • **set_ccw(speed,time)**：サーボは，指定された速度(0～255)で，秒単位の指定された時間，反時計回りに回転します。 • **stop()**：連続サーボを停止します。

CE products: `from conservo import *`

TI-Nspire CX II: `from ti_hub import *`



オブジェクトを作成する関数をメニューから貼り付けます。

var=continuous_servo("OUT 3")

コマンド	var=continuous_servo("OUT 3")
構文	var=continuous_servo("OUT 3")

Pythonコード例	**TI Nspire CX II:**
	`from ti_hub import *`
	`from time import *`
	
	`cs1=continuous_servo("OUT 3")`
	
	`cs1.set_cw(100,5)`
	`sleep(5)`
	`cs1.set_ccw(100,5)`
	CE products:
	`from conservo import *`
	`from time import *`
	`cs1=continuous_servo("OUT 3")`

	cs1.set_cw(100,5) sleep(5) cs1.set_ccw(100,5)
範囲	
説明	連続サーボモーターのオブジェクトを作成します。 サーボモーターの動作にはOUT 3でのみ使用可能な5Vが必要なため, ポートはOUT 3に制限されています。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.set_cw(speed,time)

コマンド	var.set_cw(speed,time)
構文	var.set_cw(speed,time)
範囲	
説明	指定された**speed**(速度) (0～100)と，指定された**time**(時間) (秒単位)でモーターを時計回りに回します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.set_ccw(speed,time)

コマンド	var.set_ccw(speed,time)
構文	var.set_ccw(speed,time)
範囲	
説明	指定された**speed**(速度) (0～100)と，指定された**time**(時間) (秒単位)でモーターを反時計回りに回します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.stop()

コマンド	var.stop()
構文	var.stop()
範囲	
説明	モーターを停止します。
結果	
タイプまたはアドレス可能コンポーネント	制御

Analog Out(アナログ出力)

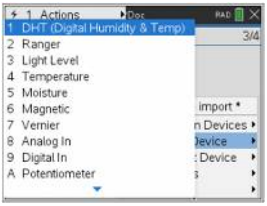
ANALOG OUTは，可変デューティサイクルを持つ，パルス幅変調出力から生成されたソフトウェアをサポートします。

出力デバイスの追加

項目	説明
Analog Out	アナログ出力汎用デバイスを使用するための関数

CE products: `from analogout import *`

TI-Nspire CX II: `from ti_hub import *`



オブジェクトを作成する関数をメニューから貼り付けます。

var=analog_out ("port")

コマンド	var=analog_out ("port")
構文	var=analog_out ("port")
範囲	
説明	**"port"**に接続されたアナログ出力デバイスのオブジェクトを作成します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.set(value)

コマンド	var.set(value)
構文	var.set(value)

Pythonコード例	**TI Nspire CX II:**
	`from ti_hub import *`
	
	`a1=analog_out("BB 5")`
	`a1.set(127)`
	CE products:
	`from analgout import *`
	
	`a1=analog_out("BB 5")`
	`a1.set(127)`

範囲	
説明	指定されたanalog_outオブジェクトの出力PWM(pulse width modulated，パルス幅変調)値を設定します。値は0から255です。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.off()

コマンド	var.off()
構文	var.off()
範囲	
説明	出力PWM値をゼロに設定し，出力ピンをデジタルローレベルに駆動します。var.set(0)に相当します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.on()

コマンド	var.on()
構文	var.on()
範囲	
説明	出力PWM値を255に設定します。これにより，デジタル高信号が継続的に出力されます。var.set(255)に相当します。
結果	
タイプまたはアドレス可能コンポーネント	制御

Vibration Motor(振動モーター)

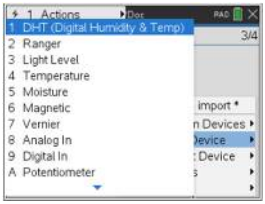
VIBRATION MOTORは，振動モーターを実行するためのインターフェースを制御します。

出力デバイスの追加

項目	説明
Vibration Motor	振動モーターを制御するための関数 • **set(val)**：振動モーターの強度をval (0～255)に設定します。 • **off()**：振動モーターをオフにします。 • **on()**：振動モーターを最高レベルでオンにします。

CE products: `from vibmotor import *`

TI-Nspire CX II: `from ti_hub import *`



オブジェクトを作成する関数をメニューから貼り付けます。

var=vibration_motor("OUT 3")

コマンド	**var=vibration_motor("OUT 3")**
構文	var=vibration_motor("OUT 3")
範囲	
説明	振動モーターのオブジェクトを作成します。 サーボモーターの動作にはOUT 3でのみ使用可能な5Vが必要なため，ポートはOUT 3に制限されています。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.set(value)

コマンド	var.set(value)
構文	var.set(value)

Pythonコード例	**TI Nspire CX II:**
	`from ti_hub import *`
	`from time import *`
	
	`vm1=vibration_motor("OUT 3")`
	`vm1.set(127)`
	`sleep(5)`
	`vm1.off()`
	CE products:
	`from vibmotor import *`
	`from time import *`
	
	`vm1=vibration_motor("OUT 3")`
	`vm1.set(127)`
	`sleep(5)`
	`vm1.off()`

範囲	
説明	valueで指定された電力で振動モーターをオンにします。値は0～255です。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.off()

コマンド	var.off()
構文	var.off()
範囲	
説明	振動モーターをオフにします。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.on()

コマンド	var.on()
構文	var.on()
範囲	
説明	振動モーターを最高レベルまでオンにします。
結果	
タイプまたはアドレス可能コンポーネント	制御

Relay(リレー)

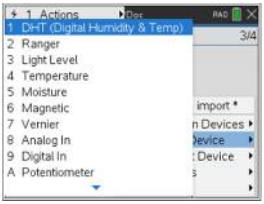
RELAYは，リレーを処理するためのインターフェースを制御します。

出力デバイスの追加

項目	説明
Relay	リレーを制御するためのインターフェースを制御します。 • **on**()：リレーをオン状態に設定します。 • **off**()：リレーをオフ状態に設定します。

CE products: `from relay import *`

TI-Nspire CX II: `from ti_hub import *`



オブジェクトを作成する関数をメニューから貼り付けます。

var=relay("OUT 3")

コマンド	var=relay("OUT 3")
構文	var=relay("OUT 3")

Pythonコード例	**TI Nspire CX II:**
	`from ti_hub import *`
	`from time import *`
	
	`r1=relay("OUT 3")`
	
	`r1.on()`
	`sleep(5)`
	`r1.off()`
	CE products:
	`from relay import *`
	`from time import *`
	
	`r1=relay("OUT 3")`
	
	`r1.on()`

	sleep(5) rl.off()
範囲	
説明	リレーのオブジェクトを作成します リレーが動作するにはOUT 3でのみ使用可能な5Vが必要なため，ポートはOUT 3に制限されます。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.off()

コマンド	var.off()
構文	var.off()
範囲	
説明	リレーをオフにします。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.on()

コマンド	var.on()
構文	var.on()
範囲	
説明	リレーをオンにします。
結果	
タイプまたはアドレス可能コンポーネント	制御

Servo(サーボ)

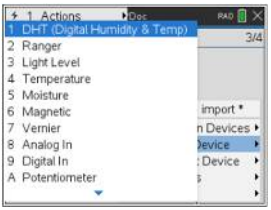
SERVOは，スイープサーボモーターのインターフェースを制御します。

出力デバイスの追加

項目	説明
Servo	サーボモーターを制御する関数 • **set_position(pos)**：スイープサーボの位置を－90～＋90の範囲で設定します。 • **zero()**：スイープサーボをゼロ位置に設定します。

CE products: `from servo import *`

TI-Nspire CX II: `from ti_hub import *`



オブジェクトを作成する関数をメニューから貼り付けます。

var=servo("OUT 3")

コマンド	var=servo("OUT 3")
構文	var=servo("OUT 3")

Pythonコード例	**TI Nspire CX II:**
	`from ti_hub import *`
	`from time import *`
	`servo1=servo("OUT 3")`
	
	`servo1.zero()`
	`sleep(2)`
	`servo1.set_position(45)`
	CE products:
	`from servo import *`
	`from time import *`

	servo1=servo("OUT 3") servo1.zero() sleep(2) servo1.set_position(45)
範囲	
説明	スイープサーボモーターのオブジェクトを作成します。 サーボモーターの動作にはOUT 3でのみ使用可能な5Vが必要なため，ポートはOUT 3に制限されています。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.set_position(pos)

コマンド	var.set_position(pos)
構文	var.set_position(pos)
範囲	
説明	スイープサーボの位置を－90～＋90の範囲内に設定し，位置を10分の1に丸めます。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.zero()

コマンド	var.zero()
構文	var.zero()
範囲	
説明	スイープサーボをゼロ位置に設定します。
結果	
タイプまたはアドレス可能コンポーネント	制御

Squarewave(方形波)

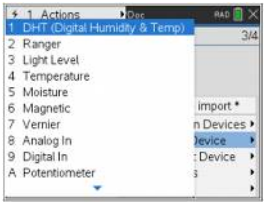
SQUAREWAVEは，デューティサイクル変動制御を備えた方形波生成インターフェースのソフトウェアです。

出力デバイスの追加

項目	説明
Squarewave	方形波を生成するための関数 • **set(frequency,duty,time)**：既定値のデューティサイクルが50%(デューティが指定されていない場合)およびfrequencyで指定された出力周波数で出力方形波を設定します。周波数は1～500Hzです。デューティサイクルは，指定されている場合，0～100%です。 • **off()**：方形波をオフにします。

CE products: `from squarewv import *`

TI-Nspire CX II: `from ti_hub import *`



オブジェクトを作成する関数をメニューから貼り付けます。

var=squarewave("port")

コマンド	var=squarewave("port")
構文	var=squarewave("port")

Pythonコード例	**TI Nspire CX II:**
	`from ti_hub import *`
	
	`sq1=squarewave("BB 7")`
	
	`# Generate squarewave of`
	`# 200Hz at 25% duty cycle`
	`# for 5 seconds`
	`sq1.set(200,25,5)`
	CE products:
	`from squarewv import *`

	sq1=squarewave("BB 7")
	
	# Generate squarewave of
	# 200Hz at 25% duty cycle
	# for 5 seconds
	sq1.set(200,25,5)

範囲	
説明	方形波ジェネレータのオブジェクトを作成します。 サーボモーターの動作にはOUT 3でのみ使用可能な5Vが必要なため，ポートはOUT 3に制限されています。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.set(freq,duty,time)

コマンド	var.set(freq,duty,time)
構文	var.set(freq,duty,time)
範囲	
説明	既定値のデューティサイクルが50%(デューティが指定されていない場合)で，出力周波数がfreqで指定された出力方形波を設定します。周波数は1～500Hzです。デューティサイクルは，指定されている場合，0～100%です。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.off()

コマンド	var.off()
構文	var.off()
範囲	
説明	生成されている方形波をオフにし，TI-Innovator Hubの出力ラインをデジタルロー状態に設定します。
結果	
タイプまたはアドレス可能コンポーネント	制御

Digital in/out(デジタル入力/出力)

DIGITALは，デジタル入力/出力ピンを制御するためのインターフェースを提供します。

入力デバイスの追加

項目	説明
Digital In	DIGITALオブジェクトに接続されているデジタルピンの現在の状態，またはオブジェクトに最後に設定されたデジタル出力値のキャッシュされた状態を返します。

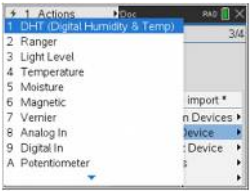
出力デバイスの追加

項目	説明
Digital Out	デジタル出力を制御するためのインターフェース。 • **set(val)**：デジタル出力をvalで指定された値(0または19に設定します。 • **on()**：デジタル出力の状態を高(1)に設定します。 • **off()**：デジタル出力の状態を低(0)に設定します。

Digital(入力/出力)モジュールはPythonクラスとして実装され，以下のように定義されたメソッドがあります。

CE products: `from digital import *`

TI-Nspire CX II: `from ti_hub import *`



オブジェクトを作成する関数をメニューから貼り付けます。

var=digital("port")

コマンド	var=digital("port")
構文	var=digital("port")
範囲	
説明	デジタルデバイスのオブジェクトを作成します。これは，入力デバイスまたは出力デバイスにすることができます。
結果	

タイプまたはアドレス可能コンポーネント	制御

var.measurement()

コマンド	var.measurement()
構文	var.measurement()

Pythonコード例	**TI Nspire CX II:**
	`from ti_hub import *`
	`from time import *`
	
	`d1=digital("BB 1")`
	`d1.on()`
	`sleep(5)`
	`d1.off()`
	CE products:
	`from digital import *`
	`from time import *`
	
	`d1=digital("BB 1")`
	`d1.on()`
	`sleep(5)`
	`d1.off()`

範囲	
説明	デジタル入力の状態を読み取り，1または0を呼び出し元に返します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.set(value)

コマンド	var.set(value)
構文	var.set(value)
範囲	
説明	デジタル出力の状態をvalue(0または1)に設定します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.off()

コマンド	var.off()
構文	var.off()
範囲	
説明	デジタル出力の状態を低(0)に設定します。
結果	
タイプまたはアドレス可能コンポーネント	制御

var.on()

コマンド	var.on()
構文	var.on()
範囲	
説明	デジタル出力の状態を高(1)に設定します。
結果	
タイプまたはアドレス可能コンポーネント	制御

Python ti_roverモジュール

CEグラフ電卓の表示は，下図のようになります。

[Fns…]>Modul: ti_rover module

ti_roverメソッドはカタログにリストされていないため，リファレンスガイドにはリストされていません。引数とその既定値，許可された値の詳細については，メニューの画面情報をご覧ください。TI-Innovator™ HubとTI-Innovator™ RoverのPythonプログラミングの詳細については，education.ti.comをご覧ください

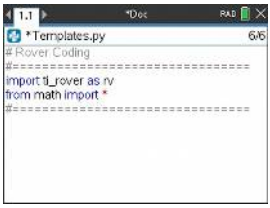
Notes:
TI-Pythonプログラミングでは，TI-Innovator™ Roverを接続・切断するメソッドを含める必要はありません。TI-Innovator™ Rover Pythonメソッドは，追加のメソッドなしで接続と切断を処理します。これは，TI-BasicでTI-Innovator™ Roverをプログラミングする場合とは少し異なります。

rv.stop()は一時停止として実行され，その後，キュー内のRoverの動きを再開します。後に別の移動コマンドを実行した場合です。

rv.stop()の場合，移動キューはクリアされます。これも，TI-BasicでTI-Innovator™ Roverをプログラミングする場合とは少し異なります。

CE Calculators

TI-Nspire™ CX



import ti_rover

Note: このモジュールを使う新規プログラムを作成するときは，Rover Codingプログラムタイプを使うことをお勧めします。これにより，関連するすべてのモジュールが確実にインポートされます。

項目	説明
import ti_rover as rv” (ti_roverをrvとしてインポート)	rv名前空間のti_roverモジュールからすべてのメソッド(関数)をインポートします。その結果，メニューから貼り付けられたすべての関数名の前にrv.が付きます。

Driveメニュー

I/O Inputメニュー

I/O Outputメニュー

I/O Pathメニュー

Settings

Commandsメニュー

---

ti_roverモジュールはPythonライブラリとして実装され，Pythonファームウェアディストリビューションに埋め込まれた凍結Pythonモジュールとして構築されます。

ti_roverモジュールをスクリプトにインポートすると，TI-Innovator™ Hubプレゼンスチェックと TI-Roverプレゼンスチェックの両方が実行されます。TI-Innovator™ Hubが存在しない場合，“TI-Innovator is not present.(TI-Innovatorは存在しません)”という例外が生成されます。Hubは存在するが，CONNECT RVプロセス中にTI-Roverが存在するものとして検出されない場合，”TI-Rover is not present.(TI-Roverは存在しません)”という例外が生成されます(これの最も一般的な原因は，TI-RoverのI2Cケーブルが正しく接続されていないことです)。

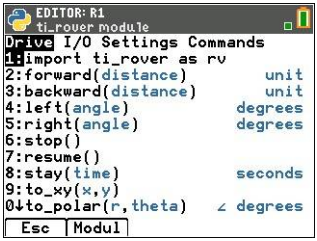
通常の使用例では，ti_roverモジュールは“**import ti_rover as rv**”“としてスクリプトにインポートされます。

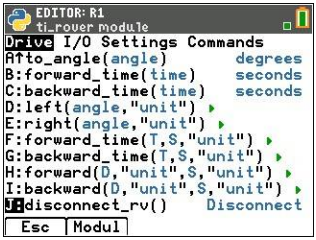
ライブラリは，以下の表で定義されている関数を実装しています。

Driveメニュー

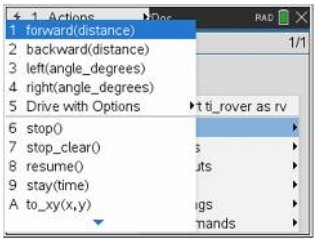
CEメニューとTI-Nspire™ CX II メニューは異なります。それぞれのスクリーンショットは，次のとおりです。

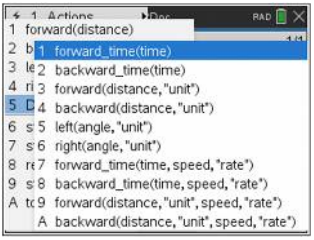
CEメニュー

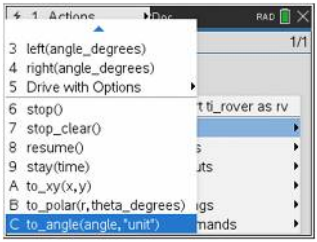




TI-Nspire™ CX IIメニュー







Driveコマンド

rv.forward

コマンド	rv.forward()
構文	rv.forward([distance]) rv.forward(distance, [rate]) rv.forward(distance, rate, speed, [rate])

Pythonコード例	`rv.forward(0.5)`
	`rv.forward(0.5, "m")`
	`rv.forward(0.5, "m", 0.22, "m/s")`
	--- 	
	`rv.forward()`
	`rv.forward([distance])`
	`rv.forward([distance],`
	`["m"|"units"|"revs"])`
	`rv.forward([distance],`
	`["m"|"units"|"revs"], s.ss)`
	`rv.forward([distance],`
	`["m"|"units"|"revs"], s.ss,`
	`["m/s"|"units/s"|"revs/s"])`

範囲	該当なし
説明	RVは所定の距離(既定値は0.75m)前方に移動します。指定されている場合の既定値の距離はUNIT(グリッド単位)です。オプションのM＝メートル，UNIT＝グリッド単位，REV＝ホイール回転。 既定値の速度は0.20m/秒，最大は0.23m/秒，最小は0.14m/秒です。速度は，メートル/秒，単位/秒，回転/秒で指定できます。
結果	RVを前進方向に動かすアクション
タイプまたはアドレス可能コンポーネント	制御 **Note:** このRover制御コマンドは，キューで送信，実行されます。

rv.forward_time

コマンド	.forward_time()
構文	rv.forward_time([time]) rv.forward_time (time, speed, [rate])

Pythonコード例	`rv.forward_time(10)`
	`rv.forward_time(10, 0.22, "m/s")`
	--- 	
	`rv.forward_time([TIME])`
	`rv.forward_time([TIME], [SPEED])`
	`rv.forward_time([TIME], [SPEED], ["m/s"|"units/s"|"revs/s"])`

範囲	該当なし
説明	RVは所定の距離(既定値は0.75m)前方に移動します。指定されている場合の既定値の距離はUNIT(グリッド単位)です。オプションのM＝メートル，UNIT＝グリッド単位，REV＝ホイール回転。 既定値の速度は0.20m/秒，最大は0.23m/秒，最小は0.14m/秒です。 速度は，メートル/秒，単位/秒，回転/秒で指定できます。既定値の速度単位はメートル/秒です。
結果	RVを前進方向に動かすアクション
タイプまたはアドレス可能コンポーネント	制御 **Note:** このRover制御コマンドは，キューで送信，実行されます。

rv.backward

コマンド	.backward()
構文	rv.backward([distance]) rv.backward(distance, [rate]) rv.backward(distance, rate, speed, [rate])

Pythonコード例	`rv.backward(0.5)`
	`rv.backward(0.5, "m")`
	`rv.backward(0.5, "m", 0.22, "m/s")`
	--- 	
	`rv.backward()`
	`rv.backward([distance])`
	`rv.backward([distance], ["m"|"units"|"revs"])`
	`rv.backward([distance], ["m"|"units"|"revs"], s.ss)`
	`rv.backward([distance], ["m"|"units"|"revs"], s.ss, ["m/s"|"units/s"|"revs/s"])`

範囲	該当なし
説明	RVは指定された距離(既定値は0.75m)後方に移動します。指定されている場合の既定値の距離はUNIT(グリッド単位)です。オプションのM＝メートル，UNIT＝グリッド単位，REV＝ホイール回転。 既定値の速度は0.20m/秒，最大は0.23m/秒，最小は0.14m/秒です。 速度は，メートル/秒，単位/秒，回転/秒で指定できます。既定値の速度単位はメートル/秒です。
結果	
タイプまたはアドレス可能コンポーネント	制御 **Note:** このRover制御コマンドは，キューで送信，実行されます。

rv.backward_time

コマンド	.backward_time()
構文	rv.backward_time([time]) rv.backward_time (time, speed, [rate])

Pythonコード例	`rv.backward_time(10)`
	`rv.backward_time(10, 0.22, "m/s")`
	--- 	
	`rv.backward_time([TIME])`
	`rv.backward_time([TIME], [SPEED])`
	`rv.backward_time([TIME], [SPEED], ["m/s"|"units/s"|"revs/s"])`

範囲	該当なし
説明	RVは指定された距離(既定値は0.75m)後方に移動します。指定されている場合の既定値の距離はUNIT(グリッド単位)です。オプションのM＝メートル，UNIT＝グリッド単位，REV＝ホイール回転。 既定値の速度は0.20m/秒，最大は0.23m/秒，最小は0.14m/秒です。 速度は，メートル/秒，単位/秒，回転/秒で指定できます。既定値の速度単位はメートル/秒です。
結果	RVを後方に動かすアクション
タイプまたはアドレス可能コンポーネント	制御 **Note:** このRover制御コマンドは，キューで送信，実行されます。

rv.left

コマンド	.left()
構文	rv.left([angle]) rv.left(angle, [unit])

Pythonコード例	`rv.left(90)`
	`rv.left(ddd, ["degrees"])`
	`rv.left(rrr, ["radians"])`
	`rv.left(ggg, ["grads"])`

範囲	該当なし
説明	DEGREES，RADIANS，GRADIANSのキーワードが存在しない限り，既定値の回転は90°です。その後，値は指定された単位から度形式に内部的に変換されます。与えられた値は0.0～360.0°の値の範囲です。ターンはSPINモーションとして実行されます。
結果	Roverを左に回転します。
タイプまたはアドレス可能コンポーネント	制御 **Note:** このRover制御コマンドは，キューで送信，実行されます。

rv.right

コマンド	.right()
構文	rv.right([angle]) rv.right(angle, [unit])

Pythonコード例	`rv.right(90)`
	`rv.right(ddd, ["degrees"])`
	`rv.right(rrr, ["radians"])`
	`rv.right(ggg, ["grads"])`

範囲	該当なし
説明	DEGREES，RADIANS，GRADIANSのキーワードが存在しない限り，既定値の回転は90°です。その後，値は指定された単位から度形式に内部的に変換されます。与えられた値は0.0～360.0°の値の範囲です。ターンはSPINモーションとして実行されます。
結果	Roverを右に回転します。
タイプまたはアドレス可能コンポーネント	制御 **Note:** このRover制御コマンドは，キューで送信，実行されます。

rv.stop

コマンド	.stop()
構文	rv.stop() rv.stop_clear()
Pythonコード例	`rv.stop()`
範囲	該当なし
説明	**RV**は現在の動きを即座に停止します。その移動は，**RESUME**操作で中断したところから再開できます。移動コマンドを実行すると，キューがすぐにフラッシュされ，投稿されたばかりの新しい移動操作が開始されます。
結果	コマンドキューからのRoverコマンドの処理を停止し，保留中の操作をキューに残します(即時アクション)。キューは**resume()**によって再開できます。**RV**は現在の移動を即座に停止します。その移動は，**resume()**操作で中断したところから再開できます。移動コマンドを実行すると，キューがすぐにフラッシュされ，投稿されたばかりの新しい移動操作が開始されます。 コマンドキューからのRoverコマンドの処理を停止し，キューに残っている保留中の操作をすべてフラッシュします(即時アクション)。
タイプまたはアドレス可能コンポーネント	制御 **Note:** このRover制御コマンドは，キューで送信，実行されます。

rv.resume()

コマンド	.resume()
構文	rv.resume()
Pythonコード例	`rv.resume()`
範囲	該当なし
説明	コマンドキューからのRoverコマンドの処理を有効にするか(即時アクション)，再開します(**rv.stay()**操作を参照)。
結果	操作を再開します。
タイプまたはアドレス可能コンポーネント	制御 **Note:** このRover制御コマンドは，キューで送信，実行されます。

rv.stay

コマンド	.stay()
構文	rv.stay([time])
Pythonコード例	`rv.stay(5)`
範囲	該当なし
説明	オプションで指定された時間(秒単位)の間，RVに留まるように指示します。 既定値は30.0秒です。
結果	RVは所定の位置に留まります。
タイプまたはアドレス可能コンポーネント	制御 **Note:** このRover制御コマンドは，キューで送信，実行されます。

rv.to_xy

コマンド	.to_xy()
構文	rv.to_xy([x,y])
Pythonコード例	`rv.to_xy(1, 1)`
範囲	x, y座標は－327～＋327。
説明	このコマンドは仮想グリッド上のRoverの動きを制御します。プログラム実行開始時の既定値の位置は(0, 0)で，Roverは正のx軸を向いています。 x, y座標は現在のグリッドサイズと一致します(既定値：0.1M/グリッド単位)。 グリッドサイズは**grid_m_unit()**コマンドで変更できます。速度パラメータはオプションです。
結果	Roverを現在のグリッド位置から指定されたグリッド位置に移動します。
タイプまたはアドレス可能コンポーネント	制御 **Note:** このRover制御コマンドは，キューで送信，実行されます。

rv.to_polar

コマンド	.to_polar()
構文	rv.to_polar([radius, theta])
Pythonコード例	`rv.to_polar(5, 30) - r = 5 units, theta = 30 degrees`
範囲	シータ座標：－360～＋360° r座標：－327～＋327
説明	RVを現在の位置からその位置を基準にして指定された極位置に移動します。 RVのX/Y位置は，新しい位置を反映するように更新されます。r座標は現在のグリッドサイズと一致します(既定値：0.1M/グリッド単位)。 プログラム実行開始時の既定値の位置は(0, 0)で，Roverは正のx軸を向いています。 シータの既定値の単位は度です。速度パラメータはオプションです。
結果	Roverを現在のグリッド位置から指定されたグリッド位置に移動します。
タイプまたはアドレス可能コンポーネント	制御 **Note:** このRover制御コマンドは，キューで送信，実行されます。

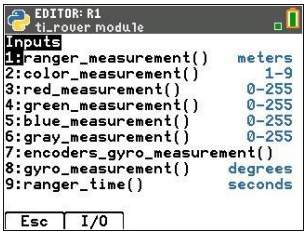
rv.to_angle

コマンド	.to_angle()
構文	.to_angle([angle]) .to_angle(angle, [unit])
Pythonコード例	`rv.to_angle(0)` `rv.to_angle(rr.rr, ["degrees"|"radians"|"gradians"])`
範囲	該当なし
説明	
結果	RVを現在の方位から指定された角度に回転させます。
タイプまたはアドレス可能コンポーネント	制御 **Note:** このRover制御コマンドは，キューで送信，実行されます。

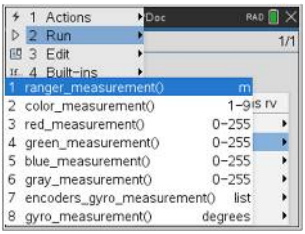
Inputsメニュー

CEメニューとTI-Nspire™ CX II メニューは異なります。それぞれのスクリーンショットは，以下のとおりです。

CEメニュー



TI-Nspire™ CX IIメニュー



Sensorコマンド

rv.ranger_measurement()

コマンド	**.ranger_measurment()**
構文	.ranger_measurment()
Pythonコード例	`r = rv.ranger_measurement()`
範囲	該当なし
説明	正面向きの超音波距離センサ。測定値をメートルで返します。約10.00メートルは，障害物が検出されなかったことを意味します。
結果	値をメートルで返します。
タイプまたはアドレス可能コンポーネント	センサ **Note:** このRoverセンサコマンドは，すぐに実行されます。

rv.color_measurement():

コマンド	**.color_measurement()**
構文	.color_measurement()
Pythonコード例	`c = rv.color_measurement()`
範囲	1～9
説明	底面に取り付けられたカラーセンサが表面の色を検出します。黒(0)から白(255)の灰色レベルスケールも検出できます。
結果	現在のカラーセンサ情報を返します。 戻り値は1～9の範囲で，以下の色にマップされます。

	色	戻り値
	Red(赤)	1
	Green(緑)	2
	Blue(青)	3
	Cyan(シアン)	4
	Magenta(マゼンタ)	5
	Yellow(イエロー)	6
	Black(黒)	7
	White(白)	8
	Gray(灰色)	9
タイプまたはアドレス可能コンポーネント	センサ **Note:** このRoverセンサコマンドは，すぐに実行されます。	

rv.red_measurement()

コマンド	**.red_measurement()**
構文	.red_measurement()
Pythonコード例	`r = rv.red_measurement()`
範囲	0～255
説明	表面の個々の赤の成分の強度を検出します。結果は0～255の範囲です。
結果	現在のカラーセンサのred(赤)の値を返します。
タイプまたはアドレス可能コンポーネント	センサ **Note:** このRoverセンサコマンドは，すぐに実行されます。

rv.green_measurement()

コマンド	**.green_measurement()**
構文	.green_measurement()
Pythonコード例	`g = rv.green_measurement()`
範囲	0～255
説明	表面の個々の緑の成分の強度を検出します。結果は0～255の範囲です。
結果	現在のカラーセンサのgreen(緑)の値を返します。
エラー	
タイプまたはアドレス可能コンポーネント	センサ **Note:** このRoverセンサコマンドは，すぐに実行されます。

rv.blue_measurement():

コマンド	.blue_measurement()
構文	.blue_measurement()
Pythonコード例	`b = rv.blue_measurement()`
範囲	0～255
説明	表面の個々の青の成分の強度を検出します。結果は0～255の範囲です。
結果	現在のカラーセンサのblue(青)の値を返します。
エラー	
タイプまたはアドレス可能コンポーネント	センサ **Note:** このRoverセンサコマンドは，すぐに実行されます。

rv.gray_measurement():

コマンド	.gray_measurement()
構文	.gray_measurement()
Pythonコード例	`gray = rv.gray_measurement()`
範囲	0～255
説明	表面の個々の灰色の成分の強度を検出します。結果は0～255の範囲です。
結果	現在のカラーセンサのgray(灰色)の値を返します。
エラー	
タイプまたはアドレス可能コンポーネント	センサ **Note:** このRoverセンサコマンドは，すぐに実行されます。

rv.encoders_gyro_measurement()

コマンド	.encoders_gyro_measurement()
構文	.encoders_gyro_measurement()
Pythonコード例	`L1 = rv.encoders_gyro_measurement()`
範囲	該当なし
説明	ジャイロと動作時間情報を組み合わせた左右のエンコーダ
結果	現在の左右のエンコーダの値のリストと，ジャイロと動作時間情報の組み合わせ。
タイプまたはアドレス可能コンポーネント	制御 **Note:** このRover READコマンドはすぐに実行されます。

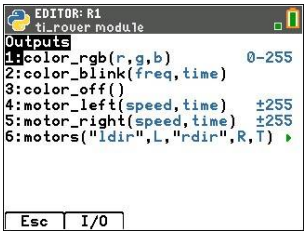
rv.gyro_measurement

コマンド	.gyro_measurement()
構文	.gyro_measurement()
Pythonコード例	`rv.gyro_measurement()`
範囲	該当なし
説明	ジャイロスコープは，Roverが動いている間，Roverの方向を維持するために使われます。また，ターン中の角度の変化を測定するために使うこともできます。 CONNECT RVコマンドが処理されると，ジャイロスコープを使用できるようになります。 GYROオブジェクトは，RVが動いていないときでも使用できます。
結果	現在のジャイロセンサの角度偏差を0.0から返し，部分的にドリフトオフセットを補正して読み取ります。
タイプまたはアドレス可能コンポーネント	制御 **Note:** このRover READコマンドはすぐに実行されます。

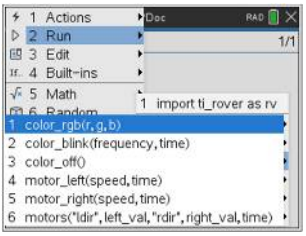
Outputsメニュー

CEメニューとTI-Nspire™ CX II メニューは異なります。それぞれのスクリーンショットは，以下のとおりです。

CEメニュー



TI-Nspire™ CX IIメニュー



Outputコマンド

rv.color_rgb()

コマンド	**.color_rgb()**
構文	.color_rgb(red, green, blue)
Pythonコード例	`rv.color_rgb(255, 255, 255)`
範囲	該当なし
説明	RoverのRGB LEDに表示されるRGBカラーを設定します。COLORなどを使ったすべてのRGB LED操作と同じ構文。
結果	RoverのRGB LEDに表示されている現在のRGBカラーを3要素リストとして返します。
タイプまたはアドレス可能コンポーネント	制御 **Note:** このRover制御コマンドは，すぐに実行されます。

rv.color_blink()

コマンド	**.color_blink()**
構文	.color_blink(frequency, time)
Pythonコード例	`rv.color_rgb(255, 255, 10)` `rv.color_blink(2, 10)`
範囲	該当なし
説明	rv.color_rgb()の補完として使用されます。現在のカラー設定で点滅します。 **frequency**(頻度)：範囲0.1～20 **time**(時間)：範囲0.1～100

結果	RoverのRGB LEDに表示されているLEDを点滅させます。
タイプまたはアドレス可能コンポーネント	制御 **Note:** このRover制御コマンドは，すぐに実行されます。

rv.color_off()

コマンド	**.color_off()**
構文	.color_off(frequency, time)
Pythonコード例	`rv.color_rgb(255, 255, 10)` `rv.color_off()`
範囲	該当なし
説明	RoverのRGB LEDに表示されているRGB LEDをオフにします。
結果	該当なし
タイプまたはアドレス可能コンポーネント	制御 **Note:** このRover制御コマンドは，すぐに実行されます。

rv.motor_left()

コマンド	**.motor_left()**
構文	.motor_left(power, time)
Pythonコード例	`rv.motor_left(l00, 2)`
範囲	left_wheel：－255～255 time(時間)：0.1～100
説明	左モーターの直接PWM値を設定します。**CCW**＝順方向，**CW**＝逆方向，pwm値 負=順方向，正=逆方向。**TIME**オプションはすべてのモードで利用できます。
結果	左ホイールモーターとダイレクトコントロール(高度)用の制御。
タイプまたはアドレス可能コンポーネント	制御 **Note:** このRover制御コマンドは，キューで送信，実行されます。

rv.motor_right()

コマンド	.motor_right()
構文	.motor_right(power, time)
Pythonコード例	`rv.motor_right(l00, 2)`
範囲	right_wheel：－255～255 time(時間)：0.1～100
説明	右モーターの直接PWM値を設定します。**CCW**＝順方向，**CW**＝逆方向，pwm値 負=順方向，正=逆方向。**time**オプションはすべてのモードで利用できます。
結果	右ホイールモーターとダイレクトコントロール(高度)用の制御。
エラー	**Error**：値が－255～255の範囲外です。 **Error**：時間が範囲外です。
タイプまたはアドレス可能コンポーネント	制御 **Note:** このRover制御コマンドは，キューで送信，実行されます。

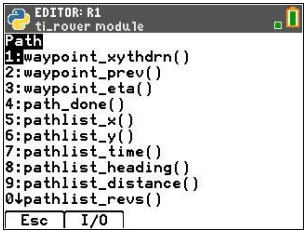
rv.motors()

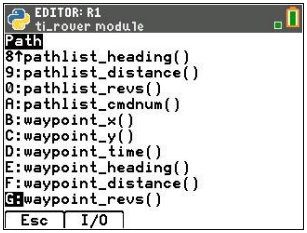
コマンド	.motors()
構文	.motors(“left_dir”, left_wheel, “right_dir”, “right_wheel”, time)
Pythonコード例	`rv.motors(“cw”|”ccw”, left_wheel,` `“cw”|”ccw”,right_wheel, time)`
範囲	left_wheel：－255～255 right_wheel：－255～255 time(時間)：0.1～100
説明	左または右，あるいは両方のモーターPWM値を設定します。 負の値はCCWを意味し，正の値はCWを意味します。 左CW＝後方への動き，左CCW＝前進運動。 右CW＝前進運動，右CCW＝後退運動。 PWM値は0～255の数値です。 値0は停止です。
結果	LEFTモーターとRIGHTモーターの両方が，直接制御(高度)の使用のため1つのオブジェクトとして管理されます。
タイプまたはアドレス可能コンポーネント	制御 **Note:** このRover制御コマンドは，キューで送信，実行されます。

Pathメニュー

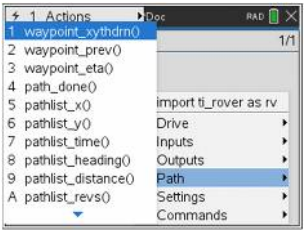
CEメニューとTI-Nspire™ CX II メニューは異なります。それぞれのスクリーンショットは，以下のとおりです。

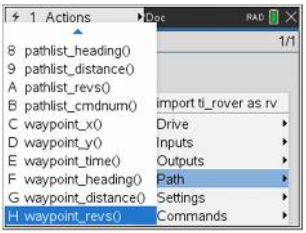
CEメニュー





TI-Nspire™ CX IIメニュー





Pathコマンド

rv.waypoint_xythdrn()

コマンド	waypoint_xythdrn()
構文	waypoint_xythdrn()
Pythonコード例	`rv.waypoint_xythdrn()`
例	最後のウェイポイント(経路上の地点)から現在のウェイポイントに向かって移動した距離を取得します。
Pythonコード例	`L1 = rv.waypoint_xythdrn()` `D = L1[5]`
範囲	該当なし
説明	x座標，y座標，時間，進行方向，移動距離，ホイールの回転数，現在のウェイポイントのコマンド番号を読み取ります。これらすべての値を要素として含むリストを返します。 **Note:** rv.left()やrv.right()など，回転コマンドによる回転数は測定されません。
結果	現在のウェイポイントのX，Y座標，時間，方位，距離，回転数，コマンド番号のリストを返します。

タイプまたはアドレス可能コンポーネント	データを返します。

rv.waypoint_prev()

コマンド	**.waypoint_prev()**
構文	.waypoint_prev()
Pythonコード例	`rv.waypoint_prev()`
例	前のウェイポイント(経路上の地点)で移動した距離を取得します。
Pythonコード例	`L1 = rv.waypoint_prev()` `D = L1[5]`
範囲	該当なし
説明	x座標，y座標，時間，進行方向，移動距離，ホイールの回転数，現在のウェイポイントのコマンド番号を読み取ります。これらすべての値を要素として含むリストを返します。
結果	前のウェイポイントのX，Y座標，時間，方位，距離，回転数，コマンド番号のリストを返します。
タイプまたはアドレス可能コンポーネント	データを返します。

rv.waypoint_eta()

コマンド	**.waypoint_eta()**
構文	.waypoint_eta()
Pythonコード例	`time = rv.waypoint_eta()`
範囲	該当なし
説明	ウェイポイントまで運転するための推定時間を返します。
結果	ウェイポイントまで運転するための推定時間を返します。
タイプまたはアドレス可能コンポーネント	データを返します。

rv.path_done()

コマンド	.path_done()
構文	.path_done()
Pythonコード例	`while not rv.path_done():` `  pass`
範囲	該当なし
説明	Roverが移動しているか(0)，すべての移動が終了しているか(1)に応じて，値0または1を返します。
結果	Roverが移動しているか(1)，またはすべてのパスコマンドが実行されたか(0)を示す，0または1を返します。
エラー	
タイプまたはアドレス可能コンポーネント	データを返します。

rv.pathlist_x()

コマンド	.pathlist_x()
構文	.pathlist_x()
Pythonコード例	`rv.pathlist_x()`
例	グラフ画面にRVパスをプロットするようにプログラムします。

Pythonコード例	`rv.to_xy(1, 2)`
	`x = rv.pathlist_x()`
	`y = rv.pathlist_y()`
	
	`print("x = {}".format(x))`
	`print("y = {}".format(y))`

範囲	該当なし
説明	現在のWaypoint Xの値を含むX値のリストを最初から最後まで返します。
結果	最後の**rv.path_clear()**または実行の開始以降に通過したX座標のリストを返します。
タイプまたはアドレス可能コンポーネント	データを返します。

rv.pathlist_y()

コマンド	.pathlist_y()
構文	.pathlist_y()
Pythonコード例	`rv.pathlist_y()`
例	グラフ画面にRVパスをプロットするようにプログラムします。

Pythonコード例	`rv.to_xy(1, 2)`
	`x = rv.pathlist_x()`
	`y = rv.pathlist_y()`
	
	`print("x = {}".format(x))`
	`print("y = {}".format(y))`

範囲	該当なし
説明	現在のWaypoint Yの値を含むY値のリストを最初から最後まで返します。
結果	最後の**rv.path_clear()**または実行の開始以降に通過したY座標のリストを返します。
タイプまたはアドレス可能コンポーネント	データを返します。

rv.pathlist_time()

コマンド	.pathlist_time()
構文	.pathlist_time()
Pythonコード例	`rv.pathlist_time()`
範囲	該当なし
説明	開始から現在のウェイポイント(経路上の地点)までの時間のリストを秒単位で返します。
結果	連続する各ウェイポイントの累積移動時間のリストを返します。
タイプまたはアドレス可能コンポーネント	データを返します。

rv.pathlist_heading()

コマンド	.pathlist_heading()
構文	.pathlist_heading()
Pythonコード例	`rv.pathlist_heading()`
範囲	該当なし
説明	最初から現在のウェイポイント(経路上の地点)までの方位のリストを返します。
結果	取得した累積角度方位のリストを返します。
タイプまたはアドレス可能コンポーネント	データを返します。

rv.pathlist_distance()

コマンド	.pathlist_distance()
構文	.pathlist_distance()
例	RVによる移動の開始以降の累積移動距離を取得します。
Pythonコード例	`d = rv.pathlist_distance()`
範囲	該当なし
説明	最初から現在のウェイポイントまでの移動距離のリストを返します。
結果	累積移動距離のリストを返します。
タイプまたはアドレス可能コンポーネント	データを返します。

rv.pathlist_revs()

コマンド	.pathlist_revs()
構文	.pathlist_revs()
Pythonコード例	`rv.pathlist_revs()`
範囲	該当なし
説明	最初から現在のウェイポイントまでに移動した回転数のリストを返します。 **Note:** rv.left()やrv.right()などの回転コマンドによる回転数は測定されません。
結果	走行したホイール回転のリストを返します。
タイプまたはアドレス可能コンポーネント	データを返します。

rv.pathlist_cmdnum()

コマンド	.pathlist_cmdnum()
構文	.pathlist_cmdnum()
Pythonコード例	`actions = rv.pathlist_cmdnum()`
範囲	該当なし
説明	rv.pathlist.cmdnum() - パスのコマンド番号のリストを返します。
結果	現在のウェイポイント(経路上の地点)エントリに移動するために使われるコマンドのリストを返します。 0 - ウェイポイントの開始(最初のアクションがSTAYの場合，START は指定されませんが代わりにSTAYが表示されます。) 1 - 前進 2 - 後進 3 - 左スピン 4 - 右スピン 5 - 左折 6 - 右折 7 - RVが現在の位置にとどまる時間(動きなし)は，時間リストに表示されます。 8 - RVは現在このウェイポイントで動いています。
タイプまたはアドレス可能コンポーネント	データを返します。

rv.waypoint_x()

コマンド	.waypoint_x()
構文	.waypoint_x()
Pythonコード例	`x = rv.waypoint_x()`
範囲	該当なし
説明	現在のウェイポイントのx座標を返します。
結果	現在のウェイポイントのx座標を返します。
タイプまたはアドレス可能コンポーネント	データを返します。

rv.waypoint_y()

コマンド	.waypoint_y()
構文	.waypoint_y()
Pythonコード例	`y = rv.waypoint_y()`
範囲	該当なし
説明	現在のウェイポイントのy座標を返します。
結果	現在のウェイポイントのy座標を返します。
タイプまたはアドレス可能コンポーネント	データを返します。

rv.waypoint_time()

コマンド	.waypoint_time()
構文	.waypoint_time()
Pythonコード例	`time = rv.waypoint_time()`
範囲	該当なし
説明	以前のウェイポイントから現在のウェイポイントまでの移動に費やされた時間を返します。
結果	累積ウェイポイント移動時間の合計値を秒単位で返します。
タイプまたはアドレス可能コンポーネント	データを返します。

rv.waypoint_heading ()

コマンド	.waypoint_heading ()
構文	.waypoint_heading ()
Pythonコード例	`heading = rv.waypoint_heading()`
範囲	該当なし
説明	現在のウェイポイントの絶対方位を返します。
結果	現在の絶対方位を度単位で返します。(＋h=反時計回り，－h＝時計回り)。
タイプまたはアドレス可能コンポーネント	データを返します。

rv.waypoint_distance()

コマンド	.waypoint_distance()
構文	.waypoint_distance()
Pythonコード例	`distance = rv.waypoint_distance()`
範囲	該当なし
説明	以前のウェイポイントと現在のウェイポイントの間を移動した距離を返します。
結果	累積総移動距離をメートル単位で返します。
タイプまたはアドレス可能コンポーネント	データを返します。

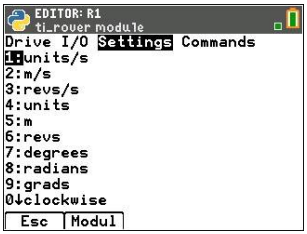
rv.waypoint_revs()

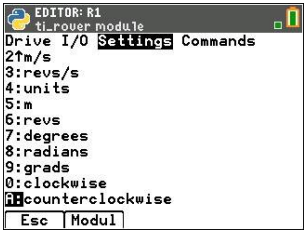
コマンド	.waypoint_revs()
構文	.waypoint_revs()
Pythonコード例	`revs = rv.waypoint_revs()`
範囲	
説明	以前のウェイポイントと現在のウェイポイントの間を移動するのに必要な回転数を返します。 **Note:** rv.left()やrv.right()などの回転コマンドによる回転数は測定されません。
結果	現在のウェイポイントまでの累積距離を移動するため実行されたホイールの総回転数を返します。
エラー	
タイプまたはアドレス可能コンポーネント	データを返します。

Settingsメニュー

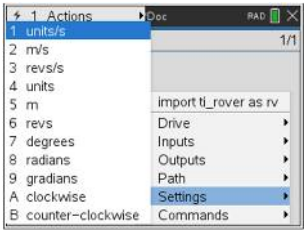
CEメニューとTI-Nspire™ CX II メニューは異なります。それぞれのスクリーンショットは，以下のとおりです。

CEメニュー





TI-Nspire™ CX IIメニュー



Settings

units/s

コマンド	units/s
構文	units/s
メソッド	rv.units/s
範囲	
説明	1秒あたりのグリッド単位での速度のオプション
結果	
エラー	
タイプまたはアドレス可能コンポーネント	設定

ms/s

コマンド	ms/s
構文	ms/s
メソッド	rv.ms/s
範囲	
説明	メートル/秒単位の速度のオプション
結果	
エラー	
タイプまたはアドレス可能コンポーネント	設定

revs/s

コマンド	revs/s
構文	revs/s
メソッド	rv.revs/s
範囲	
説明	1秒あたりのホイール回転数の速度のオプション
結果	
エラー	
タイプまたはアドレス可能コンポーネント	設定

units

コマンド	Units
構文	units
メソッド	rv.units
範囲	
説明	グリッド単位での距離のオプション
結果	
エラー	
タイプまたはアドレス可能コンポーネント	設定

m

コマンド	m
構文	m
メソッド	rv.m
範囲	
説明	メートル単位の距離のオプション
結果	
エラー	
タイプまたはアドレス可能コンポーネント	設定

revs

コマンド	revs
構文	revs
メソッド	rv.revs
範囲	
説明	ホイール回転数の距離のオプション
結果	
エラー	
タイプまたはアドレス可能コンポーネント	設定

degrees

コマンド	degrees
構文	degrees
メソッド	rv.degrees
範囲	
説明	度単位の回転のオプション
結果	
エラー	
タイプまたはアドレス可能コンポーネント	設定

radians

コマンド	radians
構文	radians
メソッド	rv.radians
範囲	
説明	ラジアン単位の回転のオプション
結果	
エラー	
タイプまたはアドレス可能コンポーネント	設定

gradians

コマンド	gradians
構文	gradians
メソッド	rv.gradians
範囲	
説明	グラード単位の回転のオプション
結果	
エラー	
タイプまたはアドレス可能コンポーネント	設定

clockwise

コマンド	clockwise
構文	clockwise
メソッド	rv.clockwise
範囲	
説明	ホイールの方向を指定するオプション
結果	
エラー	
タイプまたはアドレス可能コンポーネント	設定

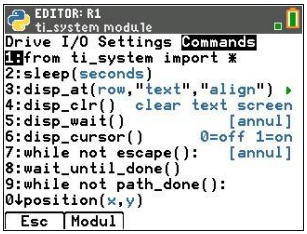
counter-clockwise

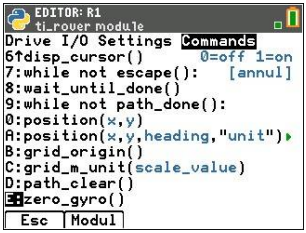
コマンド	counter-clockwise
構文	counter-clockwise
メソッド	rv.counter-clockwise
範囲	
説明	ホイールの方向を指定するオプション
結果	
エラー	
タイプまたはアドレス可能コンポーネント	設定

Commandsメニュー

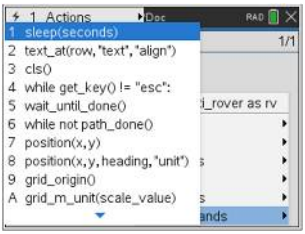
CEメニューとTI-Nspire™ CX II メニューは異なります。それぞれのスクリーンショットは，以下のとおりです。

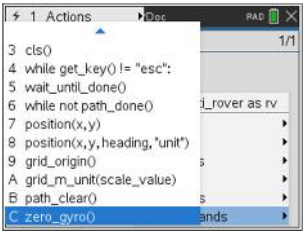
CEメニュー





TI-Nspire™ CX IIメニュー





Commands

以下のコマンドは，ti_system import *モジュールとTI Roverモジュールから集めた関数です。

sleep(seconds)

コマンド	sleep(seconds)
構文	sleep(seconds)
モジュール	ti_system
説明	指定の秒数，sleep(中断)します。引数secondsは浮動小数です。

Pythonコード例	`from time import *`
	`a=monotonic()`
	`sleep(15)`
	`b=monotonic()`
	`print(b-a)`
	
	Run the program TIME
	`>>>15.0`

結果	指定の秒数，sleep(中断)します。引数secondsは浮動小数です。
エラー	

タイプまたはアドレス可能コンポーネント	**from ti_system import***

disp_at(row,col,"text")

コマンド	**disp_at(row,col,"text")**
構文	disp_at(row,col,"text")
モジュール	ti_system
説明	プロット領域の行と列の位置からテキストを表示します。 プログラムの終了時に，カーソルを持つREPL>>>|がテキストの後に表示されます。disp_cursor()を使ってカーソルの表示を制御します。

Pythonコード例	`from ti_system import *`
	`disp_clr() #clears Shell screen`
	`disp_at(5,6,"hello")`
	`disp_cursor(0)`
	`disp_wait()`

結果	プロット領域の行と列の位置からテキストを表示します。
エラー	
タイプまたはアドレス可能コンポーネント	**from ti_system import***

disp_clr()

コマンド	**disp_clr()** **テキスト画面クリア**
構文	disp_clr() テキスト画面クリア
モジュール	ti_system
説明	シェル環境で画面をクリアします。行0～11，整数はシェル環境の表示行をクリアするためオプションの引数として使えます。

Pythonコード例	`from ti_system import *`
	`disp_clr() #clears Shell screen`
	`disp_at(5,"hello","left")`
	`disp_cursor(0)`
	`disp_wait()`

結果	シェル環境で画面をクリアします。行0～11，整数はシェル環境の表示行をクリアするためのオプションの引数として使えます。
エラー	
タイプまたはアドレス可能コンポーネント	**from ti_system import***

disp_wait()

コマンド	disp_wait() [clear]
構文	disp_wait() [clear]
モジュール	ti_system
説明	この時点でプログラムの実行を停止し，[clear]を押して画面がクリアされるまで画面の内容を表示します。

Pythonコード例	`from ti_system import *`
	`disp_clr() #clears Shell screen`
	`disp_at(5,"hello","left")`
	`disp_cursor(0)`
	`disp_wait()`

結果	この時点でプログラムの実行を停止し，[clear]を押して画面がクリアされるまで画面の内容を表示します。
エラー	
タイプまたはアドレス可能コンポーネント	**from ti_system import***

disp_cursor()

コマンド	disp_cursor() 0=オフ 1=オン
構文	disp_cursor() 0=オフ 1=オン
モジュール	ti_system
説明	プログラムの実行中のシェルでカーソルの表示を制御します。

Pythonコード例	`from ti_system import *`
	`disp_clr() #clears Shell screen`
	`disp_at(5,"hello","left")`
	`disp_cursor(0)`
	`disp_wait()`

結果	プログラムの実行中のシェルでカーソルの表示を制御します。
エラー	
タイプまたはアドレス可能コンポーネント	**from ti_system import***

while not escape(): クリア

コマンド	while not escape(): クリア
構文	while not escape(): クリア
モジュール	ti_system
説明	プログラムがループを終了することを提案するが，スクリプトを実行し続けるwhileループ。 escキーが押されるまで，whileループでコマンドを実行します。
Pythonコード例	
結果	
エラー	
タイプまたはアドレス可能コンポーネント	**from ti_system import***

rv.wait_until_done()

コマンド	.wait_until_done()
構文	.wait_until_done()

Pythonコード例	`rv.wait_until_done()`
	
	プログラム例
	`for i in range(4):`
	`··rv.forward(.5,"M")`
	`··rv.left()`
	`··rv.wait_until_done()`
	`··rv.color_rgb(255,0,0)`
	`··sleep(1)`
	`··rv.color_off()`
	
	Note: このプログラムは，RoverのLEDを1秒間赤にオンにしてから，正方形を左に曲がるたびにオフにするタスクに適しています。

範囲	該当なし
説明	プログラムはTI-Roverの移動が停止するまで待機します。
結果	プログラムはTI-Roverの移動が停止するまで待機します。
タイプまたはアドレス可能コンポーネント	設定

while not path_done()

コマンド	while not path_done()
構文	while not path_done()
メソッド	rv.while not path_done()
範囲	
説明	Roverがすべての動きを終えるまで，whileループでコマンドを実行します。
結果	関数path_done()は，Roverが移動しているか(0)，すべての移動を終了しているか(1)に応じて0または1の値を返します。
エラー	
タイプまたはアドレス可能コンポーネント	

rv.position()

コマンド	.position()
構文	.grid_origin()
Pythonコード例	`rv.position(xxx, yyy [, hhh, "degrees"|"radians"|"grads"])`
範囲	該当なし
説明	仮想グリッド上のRoverの座標位置とオプションで方位を設定します。
結果	Rover構成が更新されます。
タイプまたはアドレス可能コンポーネント	設定

rv.grid_origin()

コマンド	.grid_origin()
構文	.position(x,y) .position(x,y,heading,"unit")
Pythonコード例	`rv.grid_origin()`
範囲	該当なし
説明	RVを(0, 0)の現在のグリッド原点にあるように設定します。方位は0.0に設定されているため，RVの現在の位置は仮想x軸を正のx値に向けるように設定されています。
結果	
タイプまたはアドレス可能コンポーネント	設定

rv.grid_m_unit()

コマンド	.grid_m_unit()
構文	.grid_m_unit(scale_value)
Pythonコード例	`rv.grid_m_unit(nnn)`
範囲	該当なし
説明	仮想グリッド上のグリッド単位(grid unit)のサイズを設定します。既定値は，1単位グリッドあたり0.1mです。 scale_valueが0.2の場合，1mあたり5単位(1単位グリッドあたり20cm)を意味します。 scale_valueが0.05の場合，1mあたり20単位(1単位グリッドあたり5cm)を意味します。 仮想グリッド上のグリッド単位のサイズを設定します。既定値は，1mあたり10単位です(1単位グリッドあたり100mm/10cm)。 5の値は，1mあたり5単位(1単位グリッドあたり200mm/20cm)を意味します。 20の値は，1mあたり20単位(1単位グリッドあたり50mm/5cm)を意味します。
結果	
タイプまたはアドレス可能コンポーネント	設定

rv.path_clear()

コマンド	.path_clear()
構文	.path_clear()
Pythonコード例	`rv.path_clear()`
範囲	該当なし
説明	既存のパス/ウェイポイント情報をすべてクリアします。 ウェイポイント/パスリスト情報の一連の移動操作を実行する前に推奨されます。
結果	
タイプまたはアドレス可能コンポーネント	設定

rv.zero_gyro()

コマンド	.zero_gyro()
構文	.zero_gyro()
Pythonコード例	`rv.zero_gyro()`
範囲	該当なし
説明	Roverジャイロを0.0の角度にリセットし，左右のホイールエンコーダカウントをクリアします。
結果	
エラー	
タイプまたはアドレス可能コンポーネント	設定

PythonによるVERNIERコマンドを使った TI-SensorLinkアダプタ

ステンレススチール温度センサ（Python）

Pythonコマンド		
スケッチ オブジェクト	VERNIER	
構文		
Pythonコード例		
	望ましい行動	Pythonコード例
	取り付けたVernierセンサから温度を読み取ります。	`v1=vernier ("IN1","temperature") t1 = v1.measurement()`

pHセンサ(Python)

Pythonコマンド		
スケッチ オブジェクト	VERNIER	
構文		
Pythonコード例		
	望ましい行動	Pythonコード例
	取り付けたVernierセンサからpHを読み取ります。	`v1 =vernier("IN1","pH") ph = v1.measurement()`

ガス圧センサ(Python)

Pythonコマンド		
スケッチ オブジェクト	VERNIER	
構文		
Pythonコード例		
	望ましい行動	Pythonコード例
	取り付けたVernierセンサからガス圧を読み取ります。	`v1=vernier ("IN1","pressure") p = v1.measurement()`

カセンサ(Python)

Pythonコマンド		
スケッチ オブジェクト	VERNIER	
構文		
Pythonコード例		
	望ましい行動	Pythonコード例
	10N構成で取り付けたVernierセンサから力を読み取ります。	`v1=vernier ("IN1","force10") f1 = v1.measurement()`
	50N構成で取り付けたVernierセンサから力を読み取ります。	`v1=vernier ("IN1","force50") f1 = v1.measurement()`

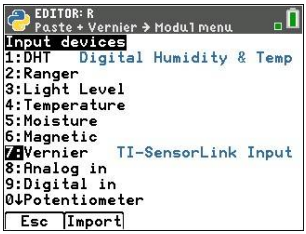
vernier()インターフェース

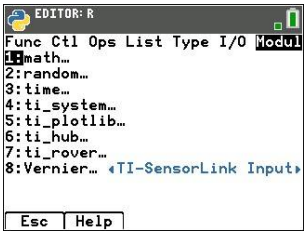
TIセンサーリンクに接続されているVernierセンサには，vernier()インターフェースからアクセスできます。

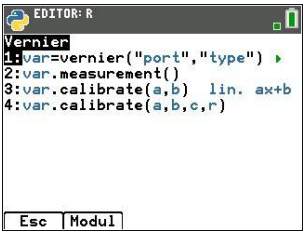
CEファミリー

ti_hubモジュールのinput deviceメニューグループの一部。

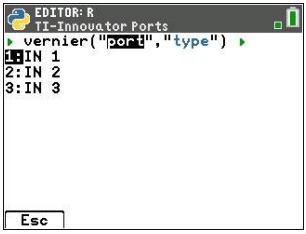
from vernier import *ステートメントがプログラムに含まれると，新しいメニューがModulesメニューに追加されます。このメニューにはVernierセンサで使用できるすべての関数があります。

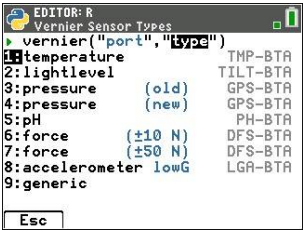






関数vernier()には，ポートと，接続されているセンサタイプの2つのヘルパープロンプトがあります。



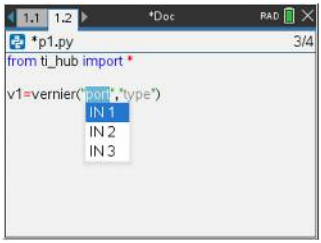


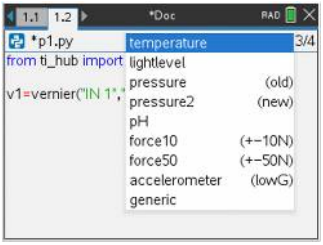
TI-Nspire CX II

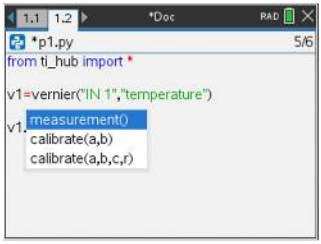
Vernierセンサへのインターフェースは，ti_hubモジュールの一部です。このインターフェースを使うには，ti_hubモジュールをインポートする必要があります。

```
from ti_hub import *
```

Vernier()オブジェクトが初期化されると，関連するすべての関数を変数名の後にピリオド(.)を入力して選択できます。







vernier()

コマンド	vernier()
構文	v1 = vernier("port", "type")

Pythonコード例	ポートIN1に接続された温度センサに接続します。
	`v1=vernier("IN1","temperature")`
	`t1 = v1.measurement()`

範囲	該当なし
説明	**VERNIER**センサオブジェクトは，接続されているVernierセンサのタイプを指定する手段を提供します(そして，電圧からセンサユニットへの適切な変換を自動的に適用します)。**VERNIER**センサは，TI-Innovator™ HubのIN1，IN2，IN3ポートに接続できます。 **Note:** 3つの**VERNIER**オブジェクトのサポートが提供されます。
結果	**VERNIER**センサをTI-Innovator™ Hubに接続します。これにより，センサとの接続が確立します。
エラー	
タイプまたはアドレス可能コンポーネント	

.measurement()

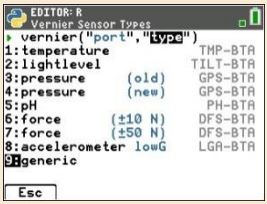
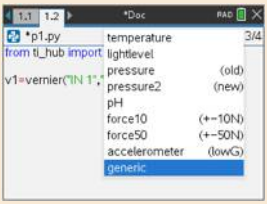
コマンド	.measurement()
構文	.measurement()

Pythonコード例	ポートIN1に接続された温度センサに接続し，温度を測定します。
	`v1=vernier("IN1","temperature")`
	`t1 = v1.measurement()`

範囲	使っているセンサによって異なります。
説明	センサデータを返します。 データの単位は，使っているセンサによって異なります。
結果	
エラー	
タイプまたはアドレス可能コンポーネント	TI-SensorLinkアダプタを介したVernierセンサ

.calibrate(a,b)

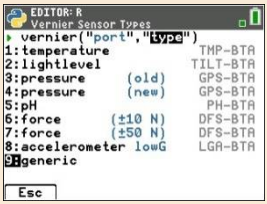
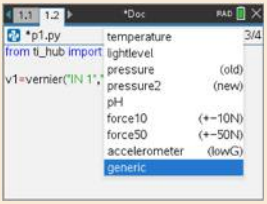
コマンド	.calibrate(a,b)
構文	.calibrate(a,b)

Pythonコード例	`v1 = vernier("IN2"," ")`
	`v1.calibrate(16.325,0.0)`
	キャリブレーション機能を使うには，generic(汎用)センサを選択します。これらのキャリブレーション値は，塩分センサ(SAL-BTA)用です。
	CE family:
	 
	TI-Nspire CX II:
	 

範囲	
説明	指定されたセンサにax＋bの線形キャリブレーションの式を設定します。 これは，サポートされているセンサのリストにないセンサを使う場合のみ必要であることに注意してください。
結果	Vernier()インターフェースは，新しい係数でキャリブレーションされます。
タイプまたはアドレス可能コンポーネント	TI-SensorLink

.calibrate(a,b,c,r)

コマンド	.calibrate(a,b,c,r)
構文	.calibrate(a,b,c,r)

Pythonコード例	`v1 = vernier("IN2"," ")`
	`v1.calibrate(1.333342e-7, 2.22468e-4,1.02119e-3)`
	
	キャリブレーション機能を使うには，generic(汎用)センサを選択します。
	CE family:
	 
	TI-Nspire CX II:
	 

範囲	
説明	係数a, b, c, 抵抗rとして使ってSteinhartキャリブレーション方程式を設定します。 通常，サーミスタとともに使われます。 これは，サポートされているセンサのリストにないセンサを使う場合のみ必要であることに注意してください。
結果	Vernier()インターフェースは，新しい係数でキャリブレーションされます。
タイプまたはアドレス可能コンポーネント	TI-SensorLink

Pythonプログラムを使ったTI-RGB Arrayコマンド

PythonプログラムでTI-RGB Arrayを使うには，適切なモジュールをインポートする必要があります。

- TI-Nspire CX II: from `ti_hub import *`
- CE family: from `rgb_arr import *`

これらのモジュールは，TI-RGB Arrayへのオブジェクト指向インターフェースを提供します。

プログラムは，最初にTI-RGB Arrayのオブジェクトを作成し，使用可能な関数を使ってそれとやり取りする必要があります。
追加のコマンドについては，education.ti.com/eguideを参照してください。

rgb_array()

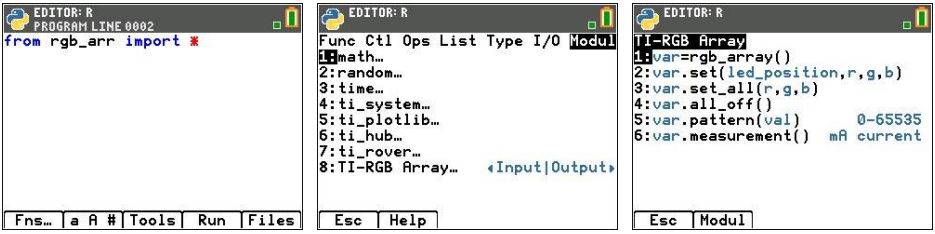
コマンド	rgb_array()
構文	var = rgb_array()
Pythonコード例	`a1 = rgb_array()`
範囲	該当なし
説明	このコマンドは，TI-Innovator™ HubソフトウェアがTI-RGB Arrayと連動するように設定します。
結果	TI-RGB ArrayをTI-Innovator™ Hubに接続します。これで，TI-RGB Arrayをプログラムする準備が整いました。
タイプまたはアドレス可能コンポーネント	TI-RGB Arrayのすべてのコンポーネント

rgb_array("lamp")

コマンド	rgb_array("lamp")
構文	var = rgb_array("lamp")
Pythonコード例	`a1 = rgb_array("lamp")`
範囲	該当なし
説明	このコマンドは，外部電源(USBバッテリーなど)が**PWR**ポートに接続されている限り，TI-RGB Arrayのhigh brightness(高輝度)モードを有効にします。 **Note: lamp**を入力する必要があります。
結果	TI-RGB Arrayが高輝度モードになるように構成されました。外部電源が接続されていない場合，**lamp**は効果がありません。つまり，明るさは既定値レベルになります。また，エラーはビープ音で示されることに注意します。
タイプまたはアドレス可能コンポーネント	TI-RGB Arrayのすべてのコンポーネント

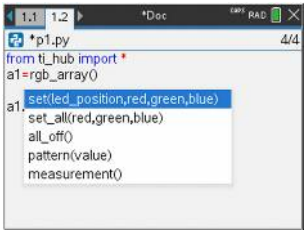
CEファミリー

fromrgb_arr import *ステートメントがプログラムに含まれると，TI-RGB Arrayの新しいメニューがモジュールメニューに追加されます。



TI-Nspire CX II

rgb_array()オブジェクトが初期化されると，関連するすべての関数を”.”(ピリオド)を変数名の後に入力することにより選択できます。



.set(n,r,g,b)

コマンド	.set(led_position, red, green, blue)
構文	var = rgb_array("lamp")
Pythonコード例	`.set(n, r, g, b)` `.set( eval(n), r, g, b)`
範囲	nの場合は0-15，r, g, bの場合は0-255
説明	.set()コマンドは，TI-RGB Arrayの各RGB LEDの明るさと色を制御します。
結果	指定されたLEDが指定された色で点灯します。
タイプまたはアドレス可能コンポーネント	TI-RGB Arrayのすべてのコンポーネント 関連項目：set_all()

.set_all(r,g,b)

コマンド	.set_all(r,g,b)
構文	.set_all(r,g,b)
Pythonコード例	`a1 = rgb_array()` `a1.set_all(255, 0, 255)`
	`a1.set_all(0,0,0)`
	`a1.set_all( eval(r), eval(g), eval(b))`

all_off()

コマンド	all_off()
構文	all_off()
Pythonコード例	`a1.all_off()`
範囲	
説明	1つのコマンドですべてのLEDを制御するには，set_all(r, g, b)を使います。
結果	1つのコマンドですべてのLEDを制御します。
タイプまたはアドレス可能コンポーネント	TI-RGB Arrayのすべてのコンポーネント 関連項目：set_all()

.pattern()

コマンド	.pattern()
構文	.pattern(value)
Pythonコード例	`a1 = rgb_array()` `a1.pattern(value)`
範囲	value: 0– 65535
説明	TI-RGB Arrayのvalueで指定された数値を表示します。値は10進数または16進数で指定できます。
結果	TI-RGB Arrayのすべてのコンポーネント
タイプまたはアドレス可能コンポーネント	TI-RGB Arrayのすべてのコンポーネント 関連項目：set_all()

.measurement()

コマンド	.measurement()
構文	.measurement()
Pythonコード例	`a1 = rgb_array()` `current = a1.measurement()`
範囲	
説明	TI-RGB Arrayが消費する電流の値をmAで返します。
結果	TI-RGB Arrayのすべてのコンポーネント
タイプまたはアドレス可能コンポーネント	TI-RGB Arrayのすべてのコンポーネント

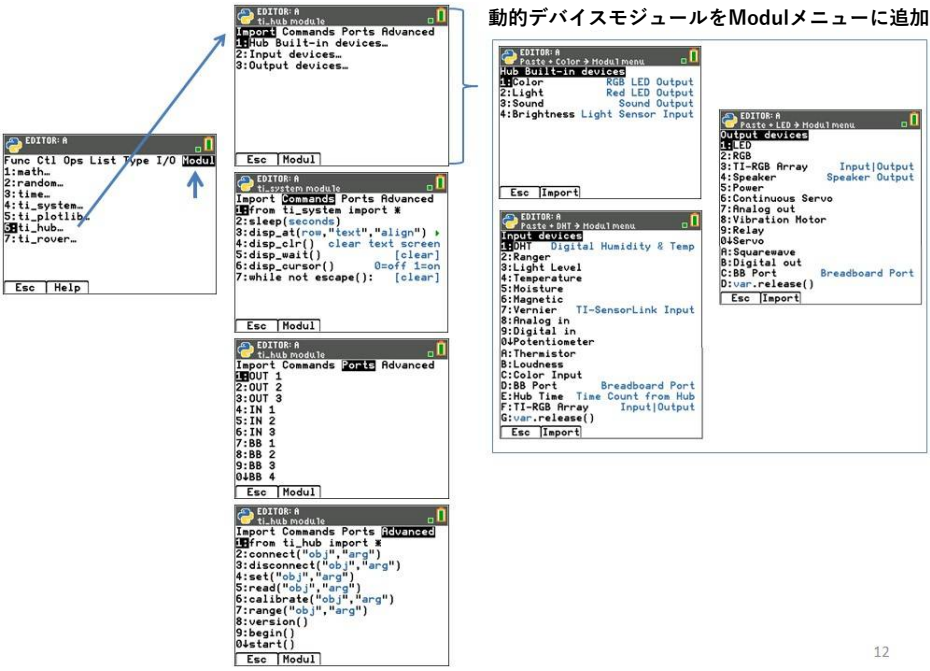
付録

浮動小数点値と小数点以下の桁数

Python ti_hubモジュール

CEグラフ電卓は，次のように表示されます。

[Fns…]>Modul: ti_hub module



ti_hubモジュールには，コマンドをTI-Innovator™ Hubに直接送信する低いレベルのコマンドインターフェイスが含まれています。以下の表は，このモジュールによって提供される関数とコマンドインターフェイスを示しています。

コマンド	from ti_hub import *
構文	from ti_hubimport *
メソッド	
説明	ti_hubモジュールからメソッドをすべてインポートします。 **Note:** このモジュールを使う新規プログラムを作成するときは，**HubProject**プログラムタイプを使うことをお勧めします。これにより，関連するすべてのモジュールが確実にインポートされます。
結果	
エラー	

タイプまたはアドレス可能コンポーネント	

CE Calculators

TI-Nspire™ CX

参照：TI-Hub Pythonコマンド

connect()

コマンド	**connect("obj","arg")**
構文	connect("obj","arg")
メソッド	connect("obj","arg")
Object(オブジェクト)	接続するTI-Innovator™オブジェクトを説明する文字列
Argument(引数)	可能なインデックス番号と，ポートなどの必要な追加の接続パラメータを含む文字列が含まれます。 追加情報が必要ない場合，このパラメータは空の文字列であることがあります。
範囲	
説明	以前接続したセンサを接続するか，またはTI-Innovator™から制御します。
結果	
エラー	
タイプまたはアドレス可能コンポーネント	

disconnect()

コマンド	disconnect("obj","arg")
構文	disconnect("obj","arg")
メソッド	disconnect("obj","arg")
Object(オブジェクト)	切断するTI-Innovator™オブジェクトを説明する文字列
Argument(引数)	可能なインデックス番号を持つ文字列が含まれます。 追加情報が必要ない場合，このパラメータは空の文字列であることがあります。
範囲	
説明	以前接続したセンサを切断するか，またはTI-Innovator™から制御します。
結果	
エラー	
タイプまたはアドレス可能コンポーネント	

set()

コマンド	set("obj","arg")
構文	set("obj","arg")
メソッド	set("obj","arg")
Object(オブジェクト)	**SET**コマンドの送信先となるTI-Innovator™オブジェクトを説明する文字列
Argument(引数)	オブジェクトのインデックス番号，設定する値，実行する操作に基づいて必要なその他のパラメータ値など，**SET**操作に必要な追加情報を含む文字列が含まれます。
範囲	
説明	ピン値の高低の設定，LEDのオンなど，接続されたセンサまたはコントロールに値または値のセットを送信します。
結果	
エラー	
タイプまたはアドレス可能コンポーネント	

read()

コマンド	read("obj","arg")
構文	read("obj","arg")
メソッド	read("obj","arg")
Object(オブジェクト)	**READ**コマンドの送信先となるTI-Innovator™オブジェクトを説明する文字列
Argument(引数)	オブジェクトのインデックス番号や，読み取る値に基づいて必要なその他のパラメータ値など，**READ**操作に必要な追加情報を含む文字列が含まれます。
範囲	
説明	センサ値，または他の接続されたセンサ，または制御値，またはプロパティを読み取ります。TI-Innovator™固有の情報を読み取るためにも使用できます。
結果	
エラー	
タイプまたはアドレス可能コンポーネント	

calibrate()

コマンド	calibrate("obj", "arg")
構文	calibrate("obj", "arg")
メソッド	calibrate("obj", "arg")
Object(オブジェクト)	**CALIBRATE**コマンドの送信先となるTI-Innovator™オブジェクトを説明する文字列
Argument(引数)	オブジェクトのインデックス番号や，キャリブレーション操作に設定する係数値など，**CALIBRATE**操作に必要な追加情報を含む文字列が含まれます。
範囲	
説明	キャリブレーション値を必要とするセンサ，または既定値のキャリブレーション値を調整できるセンサにキャリブレーション係数を送信します。
結果	
エラー	
タイプまたはアドレス可能コンポーネント	

range()

コマンド	range("obj","arg")
構文	range("obj","arg")
メソッド	range("obj","arg")
Object(オブジェクト)	**RANGE**コマンドの送信先となるTI-Innovator™オブジェクトを説明する文字列
Argument(引数)	オブジェクトのインデックス番号，設定する最小・最大範囲値など，**RANGE**操作に必要な追加情報を含む文字列が含まれます。
範囲	
説明	入力センサのADC読み取り値のマッピングを別の範囲に設定します。
結果	
エラー	
タイプまたはアドレス可能コンポーネント	

version()

コマンド	version get(string) return string
構文	**version**get(string) return string
メソッド	version()
Object(オブジェクト)	
Argument(引数)	
範囲	
説明	現在のTI-Innovator™ Hubのsketchファームウェアバージョン情報を返します。
結果	
エラー	
タイプまたはアドレス可能コンポーネント	

begin()

コマンド	**begin Wait 0.5 seconds get(string) "READY" return string**
構文	**begin**Wait 0.5seconds get(string) "READY" return string
メソッド	begin()
Object(オブジェクト)	
Argument(引数)	
範囲	
説明	TI-Innovator™ Hubのsoft-reset(ソフトリセット)を実行し，すべてのユーザーセンサとコントロールを切断し，すべての値を既定値にリセットします。
結果	
エラー	
タイプまたはアドレス可能コンポーネント	

about()

コマンド	**about get(string) return string**
構文	**about** get(string) return string
メソッド	about()
Object(オブジェクト)	
Argument(引数)	
範囲	
説明	TI-Innovator™ Hubファームウェアの著作権情報を文字列として取得します。
結果	
エラー	
タイプまたはアドレス可能コンポーネント	

isti()

コマンド	**isti get(string): "TISTEM" return string**
構文	**isti**get(string): "TISTEM" return string
メソッド	isti()
Object(オブジェクト)	
Argument(引数)	
範囲	
説明	TI-Innovator™が存在することを検出するために使用できます。**ISTI** コマンドに対する**TISTEM**の有効な応答は，TI-Innovator™が接続されていることを示します。
結果	
エラー	
タイプまたはアドレス可能コンポーネント	

what()

コマンド	**what get(string) return string**
構文	**what** get(string) return string
メソッド	what()
Object(オブジェクト)	
Argument(引数)	
範囲	
説明	TI-Innovator™であるプラットフォームに関連する情報を返します。
結果	
エラー	
タイプまたはアドレス可能コンポーネント	

who()

コマンド	**who get(string) return string**
構文	**who**get(string) return string
メソッド	who()
Object(オブジェクト)	
Argument(引数)	
範囲	
説明	TI-Innovator™であるプラットフォームに関連する情報を返します。
結果	
エラー	
タイプまたはアドレス可能コンポーネント	

last_error()

コマンド	**last_error()**
構文	**readlast error** get(string) return string
メソッド	last_error()
Object(オブジェクト)	
Argument(引数)	
範囲	
説明	TI-Innovator™のREADLAST ERRORプロパティへの簡単なアクセスを提供します。便利なプログラムまたはデバッグプログラム。
結果	
エラー	
タイプまたはアドレス可能コンポーネント	

sleep()

コマンド	sleep(seconds)
構文	sleep(seconds)
メソッド	sleep(seconds)
Object(オブジェクト)	秒単位の時間
Argument(引数)	
範囲	
説明	Python timeタイムモジュールのsleep()関数と同じ操作を実行します。
結果	
エラー	
タイプまたはアドレス可能コンポーネント	

小数点以下の浮動小数点値と桁数

ti_roverモジュールとTI-Innovatorオブジェクトでは，種々のパラメータが浮動小数点値で表されます。多数のルーチンも浮動小数点値を返します。これらの値はTI-Innovator™ Hubに送信するため特定の方法でフォーマットされ，センサの読み取り値やその他の浮動小数点の戻り値は，小数点以下の特定の桁数に丸められて返されます。

次の情報は，これらの制約の詳細です。

浮動小数点の戻り値

センサの読み取り値を返すモジュールや浮動小数点の値で表されるその他の値は，小数点の右側の小数点以下3桁に丸められます。

次の表は，種々のパラメータに対してモジュールに送信される浮動小数点の値のフォーマットを示しています。

小数点の右側の桁	パラメータ	Pythonフォーマット
1	速度，位置(serbos)	**:.1f**
1	抵抗(キャリブレーション)	**:.1f**
2	秒(時間)	**:.2f**
2	頻度(点滅，音)	**:.2f**
2	グリッドユニット(TI-Rover)	**:.2f**
2	位置(TI-Rover)	**:.2f**
3	距離(TI-Rover)	**:.3f**
3	速度(TI-Rover)	**:.3f**
3	角度(TI-Rover)	**:.3f**
3	範囲(最小値，最大値)	**:.3f**
9	キャリブレーション係数	**:.9e**