

## 【Qumcum BLE版】Python用インターフェース一覧

関数名	機能	説明	分類
qumcum.connect(address)	Qumcumと接続し通信を開始する	Args: address (str): 接続するQumcumのアドレス(4桁の英数字) Returns: int: 処理結果(0:成功, 0以外:エラー)	必須
qumcum.end()	Qumcumとの通信を終了する	Args: なし Returns: int: 処理結果(0:成功, 0以外:エラー)	必須
qumcum.get_battery_value()	電池残量の計測値を取得する 直前のコマンド実行結果から保持した値です。	Args: なし Returns: int: 処理結果(0:成功, 0以外:エラー)	データ取得
qumcum.get_info()	ロボットの基本情報を取得する	Args: なし Returns: int: 処理結果(0:成功, 0以外:エラー) Note: 戻り値はタプルで返され、先頭が処理結果、2つ目が結果文字列となり、成功した場合は下記の文字列が返ります。エラーの時、結果文字列は"NG"となります。 'INF,Qumcum,CRETARIA,2.01.03A,001EC066DE60' 左記のようなフォーマットで格納され INF,Qumcum,CRETARIA,<ファームウェアバージョン>,<ロボットのMACアドレス>がbuffに格納されます	データ取得
qumcum.get_init()	モーター初期位置を取得する	Args: なし Returns: int: 処理結果(0:成功, 0以外:エラー) Note: 戻り値はタプルで返され、先頭が処理結果、2つ目が結果文字列となり、成功した場合は下記の文字列が返ります。エラーの時、結果文字列は"NG"となります。 'INI,869.920,820,899.800,810,820' 左記のようなフォーマットで格納され INI,<モーター1の初期位置>,<モーター2の初期位置>,...,<モーター7の初期位置> となり、それぞれ1/10度単位の3桁の数値がbuffに格納されます	データ取得
qumcum.get_lib_ver()	ライブラリのバージョンを取得する	Args: なし Returns: str: バージョンを表す文字列(1.0.0.1など)	必須
qumcum.get_mic_value()	マイク計測値を取得する 直前のコマンド実行結果から保持した値です。	Args: なし Returns: int: 処理結果(0:成功, 0以外:エラー) Note: 戻り値はタプルで返され、先頭が処理結果、2つ目が結果文字列となり、成功した場合は下記の文字列が返ります。エラーの時、結果文字列は"NG"となります。 マイク計測値	必須
qumcum.get_motor_power()	モーター電源状態を取得する 直前のコマンド実行結果から保持した値です。	Args: なし Returns: int: 処理結果(0:成功, 0以外:エラー) Note: 戻り値はタプルで返され、先頭が処理結果、2つ目が結果文字列となり、成功した場合は下記の文字列が返ります。エラーの時、結果文字列は"NG"となります。 モーター電池残量の割合	データ取得
qumcum.get_position()	モーター座標値を取得する	Args: なし Returns: int: 処理結果(0:成功, 0以外:エラー) Note: 戻り値はタプルで返され、先頭が処理結果、2つ目が結果文字列となり、成功した場合は下記の文字列が返ります。エラーの時、結果文字列は"NG"となります。 7軸それぞれの座標が数値の配列としてbuffに格納されます	データ取得
qumcum.get_sensor_value()	超音波センサ計測値を取得する 直前のコマンド実行結果から保持した値です。	Args: なし Returns: int: 処理結果(0:成功, 0以外:エラー) Note: 戻り値はタプルで返され、先頭が処理結果、2つ目が結果文字列となり、成功した場合は下記の文字列が返ります。エラーの時、結果文字列は"NG"となります。 超音波センサ計測値	必須
qumcum.led_off(color)	RGB-LEDで指定した色をOFF(消灯)させる	Args: color (int): 1:赤, 2:青, 3:緑 Returns: int: 処理結果(0:成功, 0以外:エラー)	必須
qumcum.led_on(color)	RGB-LEDで指定した色をON(点灯)させる	Args: color (int): 1:赤, 2:青, 3:緑 Returns: int: 処理結果(0:成功, 0以外:エラー)	必須
qumcum.motor_adjust(no, val)	モーター調整値を設定する	Args: no (int): モーターの番号(1:右手, 2:右足, 3:右足首, 4:頭, 5:左足首, 6:左足, 7:左手) val: モーター調整値(1/10度単位) Returns: int: 処理結果(0:成功, 0以外:エラー)	調整用

関数名	機能	説明	分類
qumcum.motor_angle_multi_time(pos1, pos2, pos3, pos4, pos5, pos6, pos7, motor_time)	全モーター(7つ)の角度をセットする	Args: pos1 (int): モーター1の角度(0 ~ 180) pos2 (int): モーター2の角度(55 ~ 125) pos3 (int): モーター3の角度(55 ~ 125) pos4 (int): モーター4の角度(0 ~ 180) pos5 (int): モーター5の角度(55 ~ 125) pos6 (int): モーター6の角度(55 ~ 125) pos7 (int): モーター7の角度(0 ~ 180) motor_time (int): モーターの動作時間 Returns: int: 処理結果(0:成功, 0以外:エラー) Warning:この関数だけではモーターは動きません。 動かすには'motor_start'関数をコールしてください	必須
qumcum.motor_angle_time(no, pos, motor_time)	モーターの角度(絶対位置)、動作時間を指定する	Args: no (int): モーターの番号(1 ~ 7) pos (int): モーターの角度(0 ~ 180, no=2,3,5,6 は 55 ~ 125) motor_time (int): モーターの動作時間 Returns: int: 処理結果(0:成功, 0以外:エラー) Warning:この関数だけではモーターは動きません。 動かすには'motor_start'関数をコールしてください	必須
qumcum.motor_pos_all(pos1, pos2, pos3, pos4, pos5, pos6, pos7)	全モーター(7つ)の角度をセットする	Args: pos1 (int): モーター1の角度(0 ~ 180) pos2 (int): モーター2の角度(55 ~ 125) pos3 (int): モーター3の角度(55 ~ 125) pos4 (int): モーター4の角度(0 ~ 180) pos5 (int): モーター5の角度(55 ~ 125) pos6 (int): モーター6の角度(55 ~ 125) pos7 (int): モーター7の角度(0 ~ 180) Returns: int: 処理結果(0:成功, 0以外:エラー) Warning:この関数だけではモーターは動きません。 動かすには'motor_start'関数をコールしてください	必須
qumcum.motor_power_off()	モーター電源をOFFにする	Args: なし Returns: int: 処理結果(0:成功, 0以外:エラー)	必須
qumcum.motor_power_on(motor_time_ms)	モーター電源をONにする	Args: motor_time_ms (int): 初期モーター動作時間(単位:msec) Returns: int: 処理結果(0:成功, 0以外:エラー)	必須
qumcum.motor_set_pos(no, pos)	モーターを指定した角度(絶対位置)に移動する	Args: no (int): モーターの番号(1:右手, 2:右足, 3:右足首, 4:頭, 5:左足首, 6:左足, 7:左手) pos (int): モーターの角度(絶対位置) Returns: int: 処理結果(0:成功, 0以外:エラー) Warning:この関数だけではモーターは動きません。 動かすには'motor_start'関数をコールしてください	必須
qumcum.motor_start()	モーターを指定した角度(絶対位置)に移動させる	Args: なし Returns: int: 処理結果(0:成功, 0以外:エラー)	必須
qumcum.set_motorpower(buff)	モーター電源の供給を再開する	Args: buff: 応答格納用配列 Returns: int: 処理結果(0:成功, 0以外:エラー)	調整用
qumcum.sound(freq, time_ms)	指定した時間(msec)だけ指定した周波数の音を発生させる	Args: freq (int): 出力させる周波数 time_ms (float): 出力する時間(単位:msec) Returns: int: 処理結果(0:成功, 0以外:エラー)	必須
qumcum.voice(words)	発話する	Args: words (str): 発話させる言葉(ローマ字読み) Returns: int: 処理結果(0:成功, 0以外:エラー)	必須
qumcum.voice_word(words)	発話する	Args: words (str): 発話させる言葉(ローマ字読み) Returns: int: 処理結果(0:成功, 0以外:エラー)	必須
qumcum.voice_number(number)	数値を数字としてそのまま発話する	Args: number (int): 発話させる数値 Returns: int: 処理結果(0:成功, 0以外:エラー)	応用
qumcum.voice_numalpha(num_str)	数値を数値文字列としてそのまま発話する	Args: number (str): 発話させる数値を表現した文字列 Returns: int: 処理結果(0:成功, 0以外:エラー)	応用
qumcum.voice_numkana(number)	数値を桁を含めて発話する	Args: number (int): 発話させる数値 Returns: int: 処理結果(0:成功, 0以外:エラー)	応用
qumcum.voice_numhun(number)	数値を「〇〇分」として発話する	Args: number (int): 発話させる数値 Returns: int: 処理結果(0:成功, 0以外:エラー)	応用

関数名	機能	説明	分類
qumcum.voice_numhon(number)	数値を「〇〇本」として発話する	Args: number (int): 発話させる数値 Returns: int: 処理結果(0:成功、0以外:エラー)	応用
qumcum.voice_speed(speed)	発話する速度を指定する	Args: speed (int): 発話速度(30-300) Returns: int: 処理結果(0:成功、0以外:エラー)	必須
qumcum.wait(time_sec)	一定時間待ちます(1の場合1秒)、time.sleepを呼ぶだけです	Args: time_sec: 待ち時間(単位:秒) Returns: int: 処理結果(0:成功、0以外:エラー)	必須