TriOrb BASE 球駆動式全方向移動機構

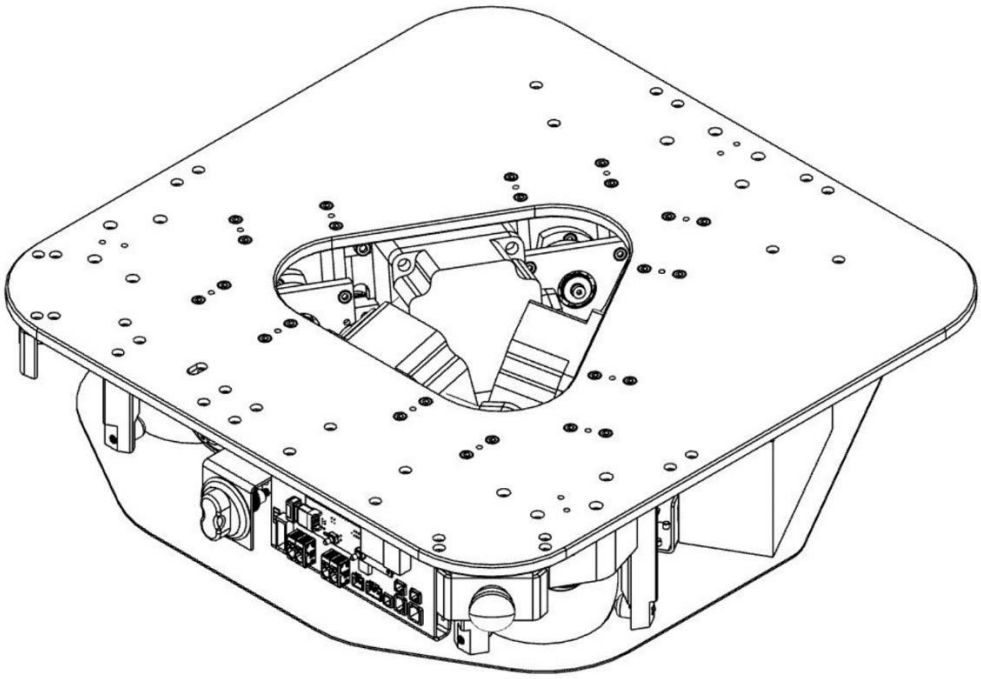
BASE500P100

取扱説明書

この度はトライオーブベースをお求めいただきまして、誠にありがとうございます。

よく読んで安全に正しくお使いください

- ご使用の前に、この取扱説明書をよくお読みの上、安全に正しくご使用ください。
- お読みになったあとは、大切に保管してください。



TriOrb

はじめに

本書は TriOrb BASE -BASE500P100-の操作方法、注意事項を説明したものです。
ご使用前に必ずお読みになり、使用方法を十分に理解された上でご使用ください。
また、ご使用時には周辺環境等、十分リスクを低減した上でご使用ください。
ご使用環境に必要な法令・規格への適合はお客様自身でご確認ください。

目次

1. 使用前のお願い

1.1. 安全上のご注意

ロボットの操作マニュアルをよく読んで理解してください。マニュアルに従って適切に操作することで、事故や怪我のリスクを最小限に抑えることができます。

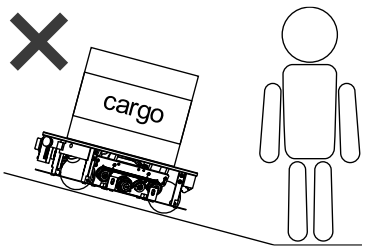
ロボットを使用する際には、安全な作業環境を確保してください。周囲に障害物や危険物がないことを確認し、ロボットの動作に支障をきたす可能性のある要素を除去してください。また、周囲の人々や作業員との十分な距離を確保してください。

ロボットの性能や適用範囲に関する制限を理解し、適正な使用方法を守るようにしてください。ロボットの改造や補器類に対して損害を与える可能性のある電子機器との接続は行わないでください。

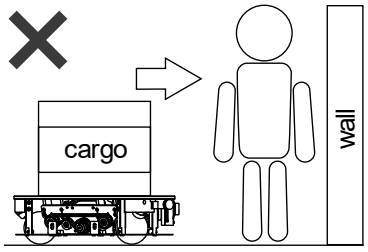
ロボットを使用する前に、緊急停止機能が正しく機能しているかを確認してください。万が一の異常動作時には、非常停止ボタンを押すことでロボットの動作を停止できます。

ロボットの異常動作、異音、異臭などの異変を感じた場合はすぐに作業を停止し、管理者や製造元に報告してください。問題を放置することで、より深刻な事故や損害が発生する可能性があります。

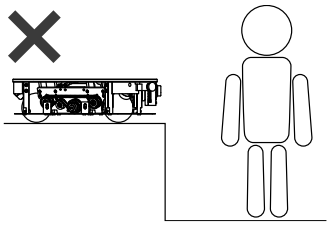
事故や怪我のリスクが高まるため以下のような状況でロボットを利用しないでください。



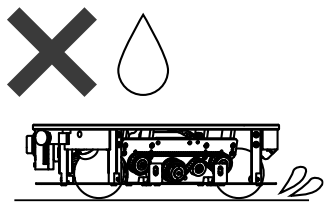
傾斜地



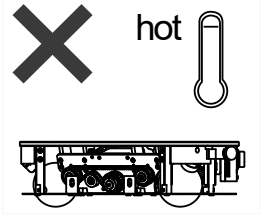
ロボットとの接触や挟まる恐れがある環境



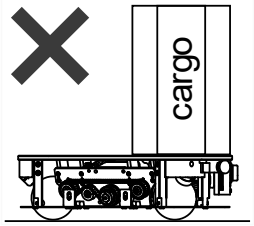
落下危険箇所



走行面に水や油のある環境



高温環境

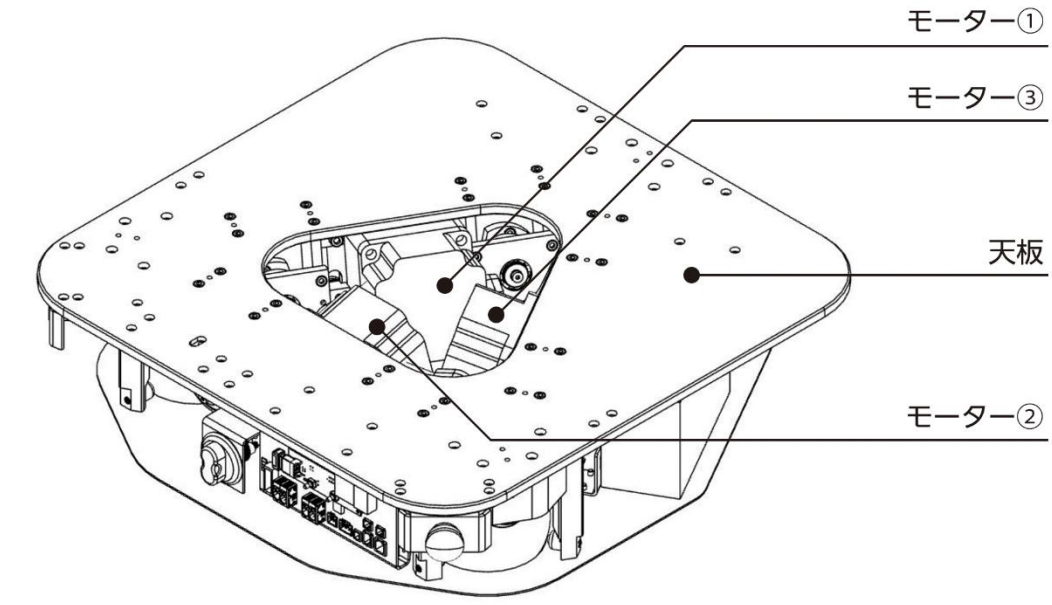


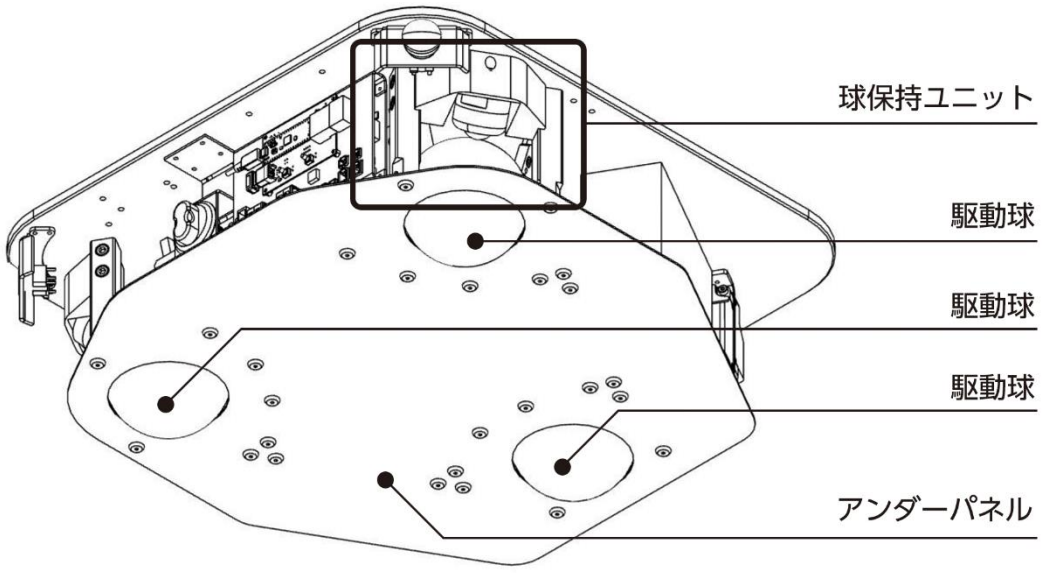
危険な積載方法

2. システム構成

2.1. 駆動部

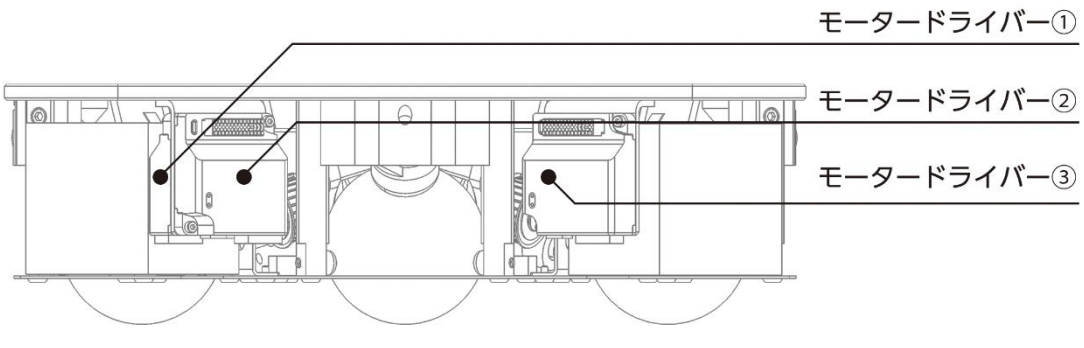
ロボット駆動部の主要な部品名称を下図に示します。各部品の形状は将来変更になる可能性があります。



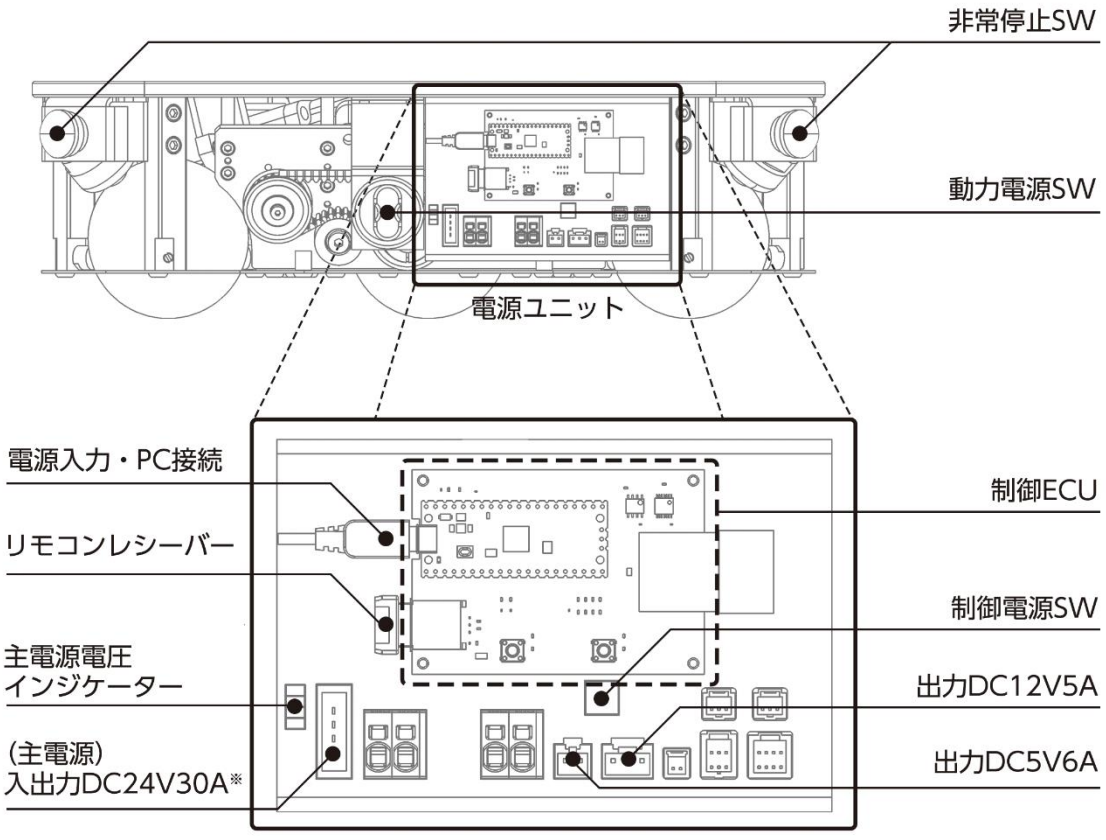


2.2. 補器類

ロボット補器類の部品名称を下図に示します。各部品の形状は将来変更になる可能性があります。



背面図



正面図（電源ユニット、制御 ECU、外部機器接続部）

3. 設置方法

3.1. セットアップ概要

- バッテリーの取り付け
  バッテリー（オプション）×2 個をロボット内部（アンダーパネル上の所定位置）に設置してください。設置の際に所定位置にある赤色と黒色のバッテリー接続線をバッテリーの対応する各端子に接続してください。接続線と端子の接続は、必ず「黒色線→赤色線」の順番で接続してください、（赤色線→赤色端子へ、黒色線→黒色端子へ）
  接続線と端子の接続先を間違えないよう十分に注意してください。
- リモコンレシーバーの取り付け
  リモコン（オプション）に付属しているレシーバーを制御 ECU の側面にある USB Type-A の差込口に差し込んでください。差し込む際は過度な力を加えず、機器やケースが破損しないように注意してください。

3.2. 開梱、再梱包

ロボットの開梱、再梱包にあたっては次の点に注意してください。

- 開梱後、ロボットが破損していないか必ず確認してください。故障したロボットに電源を入れると、予期できない動作の恐れがあります。
- ロボットを運搬する場合は運搬ルートを確保し、運搬先までは台車などを利用してください。
- 梱包箱からロボットを取り出す際に手で持ち上げる必要がある場合は、必ず持ち上げ可能な箇所（天板）のみを掴んで持ち上げてください。電気・電子機器やケーブルなど、誤った個所を掴んで持ち上げると破損する可能性があります。
- ロボットは重量物であるため、梱包箱からの出し入れは十分に配慮してください。
- 再梱包する際はロボットを必ず納入時のように梱包箱へ入れて固定してください。
- 再梱包する場合は必ずバッテリーを取り外してください。

3.3. 輸送及び保管

ロボットの輸送、保管にあたっては次の点に注意してください。

- 輸送、または保管する場合は必ずバッテリーを取り外してください。
- 運搬及び保管の際にロボットを手で持ち上げる必要がある場合は、必ず持ち上げ可能な箇所（天板）のみを掴んで持ち上げてください。電気・電子機器やケーブルなど、誤った個所を掴んで持ち上げると破損する可能性があります。
- 運搬する場合は運搬ルートを確保し、運搬先までは台車などを利用してください。
- 輸送後、ロボットが破損していないか必ず確認してください。故障したロボットに電源を入れると、予期できない動作の恐れがあります。

以下のような環境には、保管・設置しないでください。

- 直射日光が当たる場所
- 周囲温度、湿度が保管条件を超える場所
- 温度が急激に変化し、結露する場所
- 腐食性ガス、可燃性ガスの近くの場所
- ちり、埃、金属粉が多い場所
- 水、油、薬品などがかかる場所
- 振動や衝撃が本体に伝わる場所

4. 使用方法

4.1. 起動方法

(1) バッテリー接続、又は外部電源受給されていることを確認してください。
(2) 非常停止 SW が押されていないこと（ランプ点灯）を確認してください。
(3) 動力電源 SW を ON にしてください。
(4) 制御電源 SW を ON にしてください。(LED 点灯：緑)

4.2. 停止方法

(1) 制御電源 SW を OFF にしてください。(LED 消灯)
(2) 動力電源 SW を OFF にしてください。

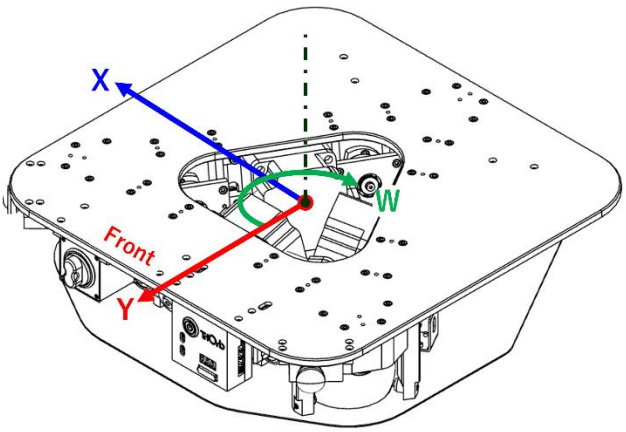
4.3. 充電

バッテリー残量が低下（18.0V 以下）した際は下記の方法でバッテリーを充電することができます。
充電の際には必ず用途に適した充電器を使用してください。互換性のない充電器の使用は、バッテリーの破損や火災を引き起こす恐れがあります。

- 充電器による充電
  ロボットからバッテリーを取り外し、充電器（オプション）を接続して充電することができます。
- 外部電源受給による充電
  ロボットにバッテリーを接続している状態で、電源ユニットに設置されている外部電源供給口へ電源供給を行うことでバッテリーを充電することができます。

4.4. ロボット座標系

ロボットの座標系は下図の通りです。電源ユニットが設置されている面が前進方向であることに注意してください。なお、座標系の原点は 3 つの駆動球を結び構成される正三角形の重心として定義しています。



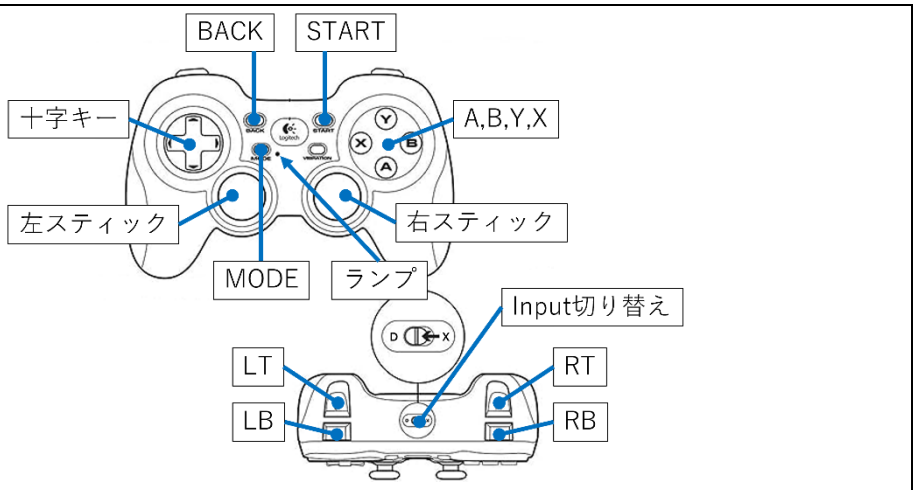
4.5.　リモコン制御

制御 ECU にリモコンレシーバーが接続されている場合、リモコンからロボットを制御することができます。ロボットを起動し、START や BACK などのボタンを押したときに MODE ボタンの隣のランプ（以下ランプ）が点灯または消灯していると操作可能です。点滅している場合、リモコンが認識されていないことを示します。数回ボタンを押しても認識しない場合は制御電源 SW を使い再起動してください。また、リモコンの上側面についている Input 切り替えスイッチが D になっていることを確認してください。

ロボットに移動指示を出すためには START ボタンを押してモーターを励磁（モーター巻き線に電流を流しトルクが発生する状態）する必要があります。励磁状態にある場合、モーターから小さな励磁音が聞こえます。BACK ボタンを押すと励磁を解除できます。

ロボットへの指示速度は速度設定値と速度指示を行うスティックの傾きによって変化します。速度設定値は 10 段階あり、ロボットの最高速度の 10~100%の値を設定できます。起動時の初期速度設定値は 5（最高速度の 50%）となっています。速度設定値を最大とし、スティックの傾きが小さい場合は小さな速度で動き、最大まで傾けると速度設定値の速度で動きます。

リモコンの各ボタンの機能は以下の通りです。



リモコン概観

Input 切り替え：D ポジションにすることでリモコンと制御 ECU が通信出来ます。

START：モーターを励磁し移動指示を受け付けます。

BACK：モーターの励磁を解除します。

MODE：並進指示ボタンを十字キーと左スティックとで切り替えます。ボタン隣のランプが点灯している場合、十字キーが並進指示ボタンとなっています。

十字キー：並進指示。上方向が前進に対応します。入力することで並進速度設定値どおりの値を指示することができます。

左スティック：並進指示。上方向が前進に対応します。スティックの傾きによって速度を調整することができ、最大まで傾けたときに並進速度設定値どおりの値を指示することができます。スティックを激しく操作すると適切に指示が送れないことがあります。

右スティック：旋回指示。左に傾けると半時計周りに、右に傾けると時計回りに旋回します。スティックの傾きによって旋回速度を調整することができ、右または左に最大まで傾けたときに旋回速度設定値どおりの値を指示することができます。スティックを激しく操作すると適切に指示が送れないことがあります。

X：リモコン制御とプログラム制御ごとに設定された速度・加速度の切り替えを行います。

LB：速度設定値を小さくします。

RB：速度設定値を大きくします。

LT：加減速時間設定値を小さくします。

RT：加減速時間設定値を大きくします。

4.6. プログラム制御

Python の USB シリアル通信によりロボットに指示を送ることができます。プログラム制御中でもリモコン操作は可能ですが、競合する指示を与えたり連続で指示を出したりすると予期せぬ動作を引き起こす場合があります。リモコン操作をする場合はプログラム制御が停止していることを確認してください。

4.6.1. USB ケーブルの接続

制御 ECU 図の「電源入力・PC 接続」のポートと PC を USB ケーブルで接続してください。制御 ECU 側は Micro USB Type-B、PC 側は任意の USB 規格を使用できます。

出荷時は電源ユニット図の「DC5V2A」 からの電力供給ケーブルが接続されているため、プログラム制御時は電力供給ケーブルを取外して PC と接続ください。

4.6.2. 制御機能一覧

機能	コード	概要
エラー情報	0x0007	現在モーターで発生中のエラー情報を取得する
運転状態	0x0009	ロボットの運転状態を取得する
電源電圧	0x0109	主電源電圧を取得する
駆動電力	0x010B	現在各モーターで消費中の電力量を取得する
エラーリセット	0x0201	現在モーターで発生中のエラーを消去する
起動・休止	0x0301	モーターの励磁/解除を設定・参照する
標準加減速時間	0x0305	ロボットの標準加減速時間を設定・参照する
標準並進速度	0x0309	ロボットの標準並進速度を設定・参照する
標準旋回速度	0x030B	ロボットの標準旋回速度を設定・参照する
速度制御（相対）	0x030F	ロボットの目標速度を設定・参照する（即時運転）
姿勢制御（相対）	0x0313	ロボットの目標姿勢（x, y, θ）を設定・参照する（即時運転）
加速時間	0x0315	ロボットの加速時間を設定・参照する
減速時間	0x0317	ロボットの減速時間を設定・参照する
運転トルク制限値	0x0319	全モーターのトルク制限減速時間を設定・参照する

4.6.3. UART 制御

通信仕様

インターフェース	UART
通信速度	115200bps
フロー制御	無効
データ長	8bit
ストップビット	1bit

パリティチェック	無し
バイト順序	little endian

通信データ構造

Start	コード 1		値 1				コード 2		値 2				End	
00													0d	0a

Start の 1 バイトから End の 2 バイトまでを 1 つの通信データとする。

ロボット（制御コンピュータ）への送信データ、およびユーザーのシステム（PC など）への返送データ、共に本項記載のデータ構造です。データ長は最大 64 バイトで可変であり任意の種類※1 の通信コードを含めることができます※2。値のデータ長は通信コード一覧表の型に従い変更してください。上図は 2 つのコード（各 4 バイト）を通信する例です。

通信コード一覧において R/W 列が R は読込み専用、W は書込み専用、RW は読書き可能であることを示します。 R コードを送信する際にも適切な長さのバイト列を付加する必要があります（値は影響しない)。ただし、R*となっているものは各モーター番号を指定し参照するため、バイト列の末尾に対象の番号を付加する必要があります。 W コードがロボットへ送信された場合、送信時の値をエコーバックします。 RW コードがロボットへ送信された場合、値書込み後の現在値を返送します。RW コードで無効値がロボットへ送信された場合、値書込みは行わずに現在値を返送します。

ロボットはデータの受信と返送をセットで行います。そのため、R・W コードに関わらずデータの送信ごとに通信バッファを読み込む必要があります。本項記載のデータ構造とは異なるバイト列を送信した場合、意図しない動作となる可能性があります。

各コードの値範囲は通信仕様上の範囲であり製品が運転可能な範囲ではありません。

※1：同じコードを繰り返す、一覧にある通信コードを可能な限り含む等の冗長なデータが送信されることは想定しておりません。返送に時間がかかる、正常に動作しない等の問題が発生する場合があります。
※2：競合するような指示が送られたとき、逐次処理の結果後のものが優先されます。例：起動・休止コードを同時に送信。

通信コード一覧

コード	内容	型	単位	R/W
0x0007	エラー情報	TriOrbBaseError：2 バイト	（型定義参照）	R*
0x0009	運転状態	TriOrbBaseState：2 バイト	（型定義参照）	R*

0x0109	電源電圧	float32	V	R*
0x010B	駆動電力	float32	W	R*
0x0201	エラーリセット	uint8	0x01: リセット	W
0x0301	起動・休止	uint8	0x00: エコーバック 0x01: 休止 0x02: 起動 0x03 – 0xFF：無効	RW
0x0305	標準加減速時間	uint32	0: auto 1 - : ms 最大：1,000,000,000	RW
0x0309	標準並進速度	float32	m/s 範囲：-3,600 to +3,600	RW
0x030B	標準旋回速度	float32	rad/s 範囲：-3,600 to +3,600	RW
0x030F	速度制御（相対）	TriOrbDrive3Pose	m/s, deg/s 範囲：-3,600 to +3,600	RW
0x0313	姿勢制御（相対）	TriOrbDrive3Pose	m, deg 範囲：-3,600,000 to +3,600,000	RW
0x0315	加速時間	TriOrbDrive3Vector	ms 範囲：0.001 to 3,600	R/W
0x0317	減速時間	TriOrbDrive3Vector	ms 範囲：0.001 to 3,600	R/W
0x0319	運転トルク制限値	uint16	0.1% 範囲：0 - 2000	R/W

型定義

TriOrbBaseError：2 バイト

Position	Type: Name	内容
1	uint8: alarm	モーターアラームコード
2	uint8: motor_id	対象のモーターID

TriOrbBaseState：2 バイト

Position		Type: Name	内容
1	1	-	-
	2	-	-
	3	-	-

	4	bool: trq	上限トルクに到達
	5	bool: move	移動中
	6	bool: in_poss	位置制御が終了
	7	bool: s_on	励磁中
	8	bool: success	モーターのステータス取得の成否
2		uint8: motor_id	対象のモーターID

TriOrbDrive3Pose：12 バイト

Position	Type: Name	内容
1-4	float: x	ロボット X 方向の値
5-8	float: y	ロボット Y 方向の値
9-12	float: w	ロボット W 方向の値

TriOrbDrive3Vector：12 バイト

Position	Type: Name	内容
1-4	float: v1	モーター①の値
5-8	float: v2	モーター②の値
9-12	float: v3	モーター③の値

4.6.4. Python 制御

最新 Release のインストール

Python による制御ライブラリを以下のコマンドでインストールします。なお、Python のバージョン 3.8 で機能することを確認しています。

```
pip install git+https://github.com/TriOrb-Inc/triorb-core.git
```

ロボットの接続と起動および休止

```
from triorb_core import robot

vehicle = robot("COM1")
vehicle.wakeup() # 起動
vehicle.sleep() # 休止
```

robot インスタンスの第 1 引数にはロボット（制御 ECU）と接続されたシリアルポートを指定します。Windows システムでは”COM1, COM2,,,”などが一般的で、Linux システムでは”/dev/ttyUSB0, ttyUSB1,,,”などが一般的です。Robot インスタンスの関数を使用することにより 4.6.3 章で示したデータ列の生成と送受信を簡易化できます。このとき、各関数の戻り値は返送されたバイナリデータを通信コードに対応する型に変換したものです。

wakeup 関数を呼び出すことでモーターが励磁状態に移行し、位置制御や速度制御を受け付けるようになります。sleep 関数を呼び出すことで無励磁状態に移行し、モーターが停止します。

ロボットの状態取得

```
from triorb_core import robot

vehicle = robot("COM1")

print(vehicle.get_motor_status(params=['error','state','voltage','power
'], _id=[1,2,3]))    # 指定した ID のモータードライバの各種ステータスを取得
```

上記の関数でモーターの状態を取得できます。

get_motor_status 関数は引数に取得したい情報と ID を指定することができます。引数を与えない場合は指定可能な全てのステータス情報と走行に使用されるモーターの ID が設定されます。戻り値は len(params)*len(_id)数の要素を持つ配列です。また、モータードライバにエラーがある状態で実行すると、エラーリセットが実行されすべてのモーターの励磁状態が解除されます。また、モータードライバのステータスの取得に失敗することがあります。失敗したモータードライバの戻り値には例外的な値が与えられます。ロボット状態一

覧は下表の通りです。

Key	内容	取得失敗時
error	モータードライバのアラーム情報を取得する。	alarm 変数：-1
state	TriOrbBaseState 参照	motor_id を除き 0
voltage	電源電圧（単位：V）	0
power	駆動電力（単位：W）	0

状態コード	値	内容
I0001	Float	動作完了予想時間（単位：秒）
I0002	None	休止中
I0003	Float	電源電圧（単位：V）
E0001	None	[ERROR] モータードライバとモーターが通信できない。駆動電源喪失時・非常停止時などに発生。
E0002	None	[ERROR] 過電流異常停止

ロボットの姿勢制御

```
from triorb_core import robot

vehicle = robot("COM1")
vehicle.wakeup() # 起動

vehicle.set_pos_relative(x=1.0, y=-1.0, w=90.0) # ロボットの姿勢をローカル座標系で設定。並進位置 x=1.0[m], y=-1.0[m]および回転位置 w=90.0[deg] の姿勢を取ります。
vehicle.join() # 動作完了待ち

vehicle.set_pos_relative(x=1.0, y=0, w=0, acc=3000, dec=2000) # 加速時間を 3 秒、減速時間を 2 秒に設定しロボットをローカル座標系の位置 x=1.0[m] の姿勢を取ります。
vehicle.join()

print(vehicle.set_pos_relative(x=360001, y=0, w=0)) # 無効値を入力した場合、前回の指示値を取得できます。
```

ロボットの姿勢（並進位置、回転角）を変化させるプログラム例を示します。ロボットは標準速度まで加速し、目標姿勢で停止するように減速します。加速・減速時間は姿勢制御関数で引数として与えられる他、後述する write_config 関数で設定できます。

姿勢制御関数はプログラムを中断しません。join 関数を使用することで動作完了までプログラムを待機させることができます。join 関数を使用せずに他の処理を行うこともできますが、停止前に進行方向を大きく変える指示を出すと予期せぬ動作を引き起こす場合があります。なお、本指令ではキネマティクスに基づく理論値であるため、路面状況等によって実際の姿勢と異なる恐れがあります。

set_pos_relative：

- 指示を出した時点でのロボットの姿勢を原点としたときの姿勢を目標とする
- 同じ指示を与えたときの移動量が一定になる

並進位置と回転角の指示を同時に与えると、期待される姿勢にならないことに注意してください。例えば、vehicle.set_pos_relative(x=1.0, y=0.0, w=180.0) を実行しても、ロボットの姿勢はほとんどの場合(1,0,180)となりません。ロボットは旋回しながら前進するので弧を描くような動作となります。最終的な姿勢は加減速時間と標準速度によって変化します。

x,y,w のいずれかに上限・下限値の範囲外の値を与えると読込みモードとなり、最後に与えた x,y,w の値を返します。これは各関数で値が保持されます。

ロボットの速度制御

```
import time
from triorb_core import robot

vehicle = robot("COM1")
vehicle.wakeup() # 起動

vehicle.set_vel_relative(x=0.1, y=0.1, w=0.1) # ロボットの速度をローカル座標系で設定。並進 x=0.1[m/s], y=0.1[m/s]および旋回 w=0.1[rad/s] の速度で移動。
time.sleep(5.0) # 5 秒待つ
vehicle.brake() # 減速停止する
vehicle.join()  # 移動完了待ち

vehicle.set_vel_relative(x=0.1, y=0, w=0, acc=3000, dec=2000) # 加速時間 3 秒、減速時間 2 秒に設定し、ロボットの速度をローカル座標系の並進 x=0.1[m/s]に設定。
time.sleep(5.0) # 5 秒待つ
vehicle.brake() # 減速停止する
vehicle.join()  # 移動完了待ち

print(vehicle.set_vel_relative(x=360001, y=0, w=0)) # 現在速度を取得
```

ロボットの速度を変化させるプログラム例を示します。ロボットは指示速度まで加速した後、新たな速度制御や姿勢制御を行うまで等速で動き続けます。停止させる場合は brake 関数を使用することもできます。加速時間・減速時間は速度制御関数で引数として与えられる他、後述する write_config 関数で設定できます。

速度制御関数はプログラムを中断しません。join 関数を使用する場合、必ず直前に brake 関数を実行してください。

set_vel_relative：

- 指示時のロボットの姿勢を原点とする
- 前進指示を出すと指示時のロボットの向いている方向に進む

x,y,w のいずれかに上限・下限値の範囲外の値を与えると読込みモードとなり、現在のロボットの x,y,w の値を返します。これはモータードライバが検出した旋回速度から計算されます。計算に失敗した場合、各値が 0 として返されます。

ロボットの設定変更

```
from triorb_core import robot

vehicle = robot("COM1")
print(vehicle.read_config()) # 現在設定を読み込み
vehicle.write_config({ # 設定を書込み
    "acc" : 1500, # 標準加減速時間を 1.5[s]に設定
    "std-vel" : 0.25, # 標準並進速度を 0.25[m/s]に設定
    "std-rot" : 0.5, # 標準旋回速度を 0.5[rad/s]に設定
    "torque" : 500,  # トルク制限値を 50[%]に設定
})
print(vehicle.read_config(["acc","std-vel","torque"])) # 現在設定を読み込み
```

ロボットのパラメータを読み書きするプログラム例を示します。書き込み用の関数は write_config、読み込む用の関数は read_config です。書き込んだ値はロボットの制御 ECU が再起動されるまで保持されます。

read_config および write_config 関数で指定できる値は標準加減速時間・標準並進速度・標準旋回速度・トルク制限値です。これらの値は主に速度制御および姿勢制御で用いられ、標準速度は姿勢制御にのみ適用されます。read_config は読み込む値を配列型で指定し、write_config は書き込む対象とその値を辞書型として下表のとおり設定できます。

Key	内容	出荷時設定
acc	標準加速時間（単位：ms）。0 以下を設定すると自動制御となる。	500

std-vel	標準並進速度（単位：メートル毎秒）	0.2
std-rot	標準旋回速度（単位：ラジアン毎秒）	0.2
torque	トルク制限値（単位：0.1%）	1000

通信コードと値を指定して通信データを送受信する例

```
from triorb_core import *
import time
import numpy as np
vehicle = robot("COM1")

query = []
query.append([RobotCodes(0x0301), b'¥x02'])        # 励磁
query.append([RobotCodes(0x0313), TriOrbDrive3Pose(1,-1,0)])  # 相対姿勢制御
query.append([b'¥x05¥x03', np.uint32(1000)])    # 標準加減速時間設定

vehicle.tx(code_array=query) # 送信
print(vehicle.rx()) # 受信

query = []
query.append([RobotCodes(0x0009), b'¥x00¥x01'])  # モータードライバのステータス取得(ID1)
val = TriOrbBaseState(motor_id=2)
query.append([b'¥x09¥x00', val])  # モータードライバのステータス取得(ID2)

vehicle.tx(code_array=query) # 送信
print(vehicle.rx()) # 受信

time.sleep(5.0)
vehicle.tx(code_array=[[RobotCodes(0x0301), b'¥x01']]) # 励磁解除
vehicle.rx()

# 誤ったデータの送信
vehicle.tx(code_array=[[RobotCodes(0x0301), b'¥x02'],  # 励磁(正)
                       [RobotCodes(0x0305), b'¥x03']]) # 加速度設定(誤)
# 返送(START と END のみ)
```

```
print(vehicle.rx()) # -> b'¥x00¥r¥n'
```

通信コードと値を指定して通信するプログラム例を示します。書込みは tx 関数、読込みは rx 関数を利用できます。tx 関数実行後は rx 関数でデータを受信してください。tx 関数を続けて実行した場合、以降の rx 関数のほとんどの場合正常に処理されず、戻り値は要素数 0 の配列になります。また、tx 関数の実行前に rx 関数を実行すると受信データを待機し続けるためプログラムが停止します。これは連続で rx 関数を実行した場合でも同様です。

tx 関数の引数は、通信コードとその値の配列を要素に持つ 2 重配列です。通信コードは RobotCodes 型に変換し、値は対応する型に変換してください。一部の型は numpy を利用しています。どちらも bytes 型でも渡すことができますが、Little-Endian であることに注意してください。通信データ構造と異なるデータを送信した場合、ロボットは Start+End の返送のみ行います。プログラム例の「誤ったデータの送信」で示すような正常な値と誤った値が含まれるデータを送信した場合も同様です。

rx 関数は返送データの値を適切な型に変換し、それらを要素として持つ配列を返します。バイト型のまま受信したい場合は rx_bytes 関数が利用できます。

5. メンテナンス

5.1. メンテナンス時の注意事項

作業を開始する前に必ずロボットの電源を切断し、バッテリーを取り外してください。不適切なメンテナンス作業は重大な障害や物的損害が発生する可能性があります。

- 作業を開始する前に十分な作業スペースを確保してください。
- 作業の際は必ず作業手袋などの安全装備を着用してください。
- ロボットの周囲に障害物がないか確認してください。
- ロボットの機体に破損がないか確認してください。

5.2. メンテナンス方法

駆動球、ケーブル類、タイミングベルト、ボールキャスタについては使用前後など定期的なメンテナンスを行ってください。各部品はロボット底部のアンダーパネルを取り外し、メンテナンスを行うことができます。ロボットを掴んで持ち上げる必要がある場合は、必ず持ち上げ可能な箇所（天板）のみを掴んで持ち上げてください。電気・電子機器やケーブルなど、誤った個所を掴んで持ち上げると破損する可能性があります。

- ロボットの各部品を清掃し、汚れやごみを除去してください。
- ロボットの各部品を点検し、異常がないか確認してください。
- ロボットの各部品を調整し、正常に動作していることを確認してください。
- 水や油が付着した場合は、きれいな布等で拭き取ってください。

6. サービス

6.1. 警告

- 自重が不足している場合は駆動球に滑りが発生する可能性がありますので、滑りが発生する場合はバラストなどを搭載し駆動摩擦を確保する検討をしてください。
- 電源電圧が 18.0V 以下の場合ロボットが正常に動作しない可能性がありますので、速やかに運転を停止しバッテリーを充電してください。

6.2. エラー

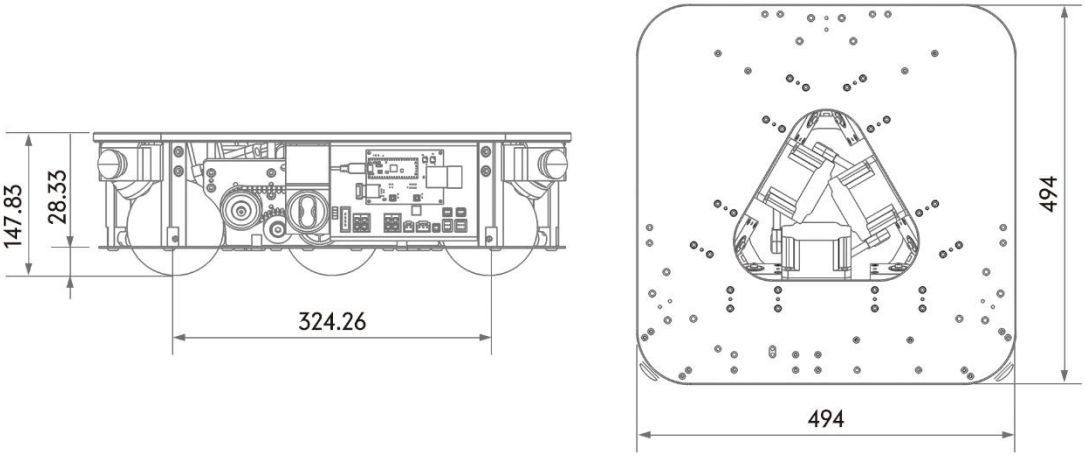
- リモコン制御で正常にロボットが動かない場合は START ボタンを押した際にリモコンのランプが点滅しないかを確認してください。点滅する場合は制御 ECU やリモコン通信に不具合が発生している可能性がありますので、制御電源 SW を一旦 OFF にし再度 ON にするなどにより制御 ECU の再起動を実施してください。改善しない場合はメーカーに修理を依頼してください。
- 非常停止 SW のランプが正しく点灯しない場合は非常停止 SW 接続ケーブルの不具合か、非常停止 SW が故障している可能性があります。コネクタの接続に問題が無いか確認し、改善しない場合はメーカーに修理を依頼してください。

- 制御電源 SW のランプが正しく点灯しない場合は主電源が正しく供給されていないか、電源ユニットが故障している可能性があります。主電源およびケーブルが正常か確認し、改善しない場合はバッテリーを取り外してメーカーに連絡してください。
- 全てのモーターが起動しない（励磁音もしない）場合は制御 ECU から出ているモーター制御信号線に不具合が発生している可能性があります。コネクタの接続に問題が無いか確認し、改善しない場合はメーカーに修理を依頼してください。
- 特定のモーターが起動しない（励磁音もしない）場合は「通信用電源」「RUN 信号」「モーター制御信号」などに不具合が発生している可能性があります。コネクタの接続に問題が無いか確認し、改善しない場合はメーカーに相談してください。
- プログラム制御時にはモーターの警告情報が取得できます。「オリエンタルモーター（株） BLV シリーズ R タイプ 機能編(HP-5141-4)」アラーム一覧から内容を参照してください。

7. 仕様

7.1. 外形寸法および重量

BASE500P100（28kg）



7.2. 基本仕様

本データは本体限界値ではなく、弊社検証環境での実験値になります。外乱に対する走破性は速度、進入角度、路面状況等によって変動がありますので、安全を十分に配慮してご使用ください

		BASE500P100
環境	使用場所	屋内
	周辺温度	0～40℃
	湿度	確認中、結露無きこと
	その他	水、油濡れ厳禁
運動仕様	運動軸	前後・左右・旋回
	駆動球	ϕ100mm
	モーター	100W(×3 軸)
	ギア比	20
	並進速度(V_{max})[*1]	0.67m/s
	旋回速度(VR_{max})	6.0rad/s
	登坂(積載 0kg)	1/6
	登坂(積載 300kg)	1/6
	溝通過(積載 0kg)	60mm
	溝通過(積載 300kg)	45mm
	段差(積載 0kg)	15mm

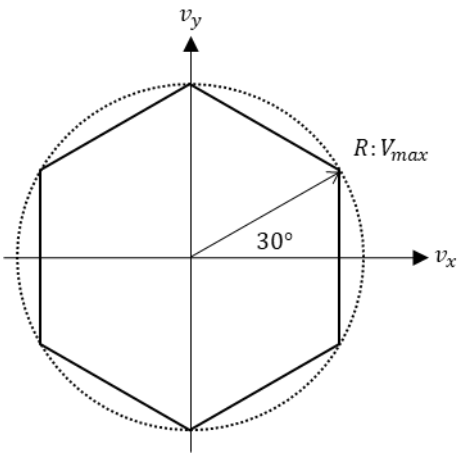
	段差(積載 300kg)	7mm
稼働時間	バッテリー	WP1236W（×2 個）
	容量	9Ah
	連続稼働時間*2	3.5h
	バッテリー重量	5.4kg
	充電方法	・取外し後 12V 充電(100Vac、120W) ・コネクタ 24V 充電（100Vac、450W）

*1 進行方向により最大並進速度が異なります

*2 検証：平滑路面において速度 0.3m/s で連続稼働を行ったデータにより算出

並進速度の出力特性

TriOrb BASE は制御上、速度出力に以下の特性があります。下図は TriOrb BASE の駆動球と座標系および最大並進速度の上面模式図です。駆動球 2 つの辺を前進y方向とし、その最大速度が別表V_{max}となります。



7.3.　付属品（オプション）

- バッテリー（LONG シールドバッテリー　WP1236W）　・・・2 個
- リモコン（Logicool　ゲームパッドコントローラ F750r）　・・・1 個
- 充電器（大橋産業 12V　ACE CHARGER 10A）　・・・2 個

TriOrb

株式会社TriOrb

福岡県北九州市小倉北区浅野3-8-1 AIMビル6F
TEL：093-513-1023
https://www.triorb.co.jp