Asratec

V-Sido
CONNECT RC

「V-Sido CONNECT RC」
制御コマンド作成の手引き

第 0.9.6 版

2016/8/1
アスラテック株式会社

はじめに

●本マニュアルの目的

「V-Sido CONNECT」を使えば、対応しているロボット（サーボモータ）をさまざまなシリアルコマンドによって制御できるようになります。本マニュアルでは、V-Sido CONNECT に送るコマンドの作り方を説明します。

V-Sido CONNECT RC を搭載したロボットを動かす場合、主に以下のコマンドを送信して、求める動作を実現します。

- **目標角度設定**
- **IK 設定**
- **歩行設定**

本マニュアルでは、上記のコマンドを取り上げています。V-Sido CONNECT 対応ロボットを動作させるときに、具体的にどういった内容のコマンドを発行すればよいのかを、簡単な事例を元に解説します。

なお、コマンドの定義や通信仕様などの詳細は、別途『「V-Sido CONNECT」コマンドリファレンスマニュアル』※を参照してください。本マニュアルは、コマンドリファレンスマニュアルの副読本の位置づけのテキストとなります。

●本マニュアルの構成

本マニュアルでは最初の 1 章/2 章で、制御コマンドの作成において重要なサーボ ID と IK について説明します。

その後の 3～5 章で、ロボットにどういった動作を行わせたいかの事例を取り上げ、具体的なコマンドの作り方を紹介します。

そして最後の 6 章では、制御コマンドを利用するときに役立つテクニックや補足情報などを解説しています。

※ 『「V-Sido CONNECT」コマンドリファレンスマニュアル』は V-Sido 開発者支援サイト「V-Sido Developer」（https://v-sido-developer.com/）で公開されています

Asratec

目　次

1. 対応ロボットとサーボ ID

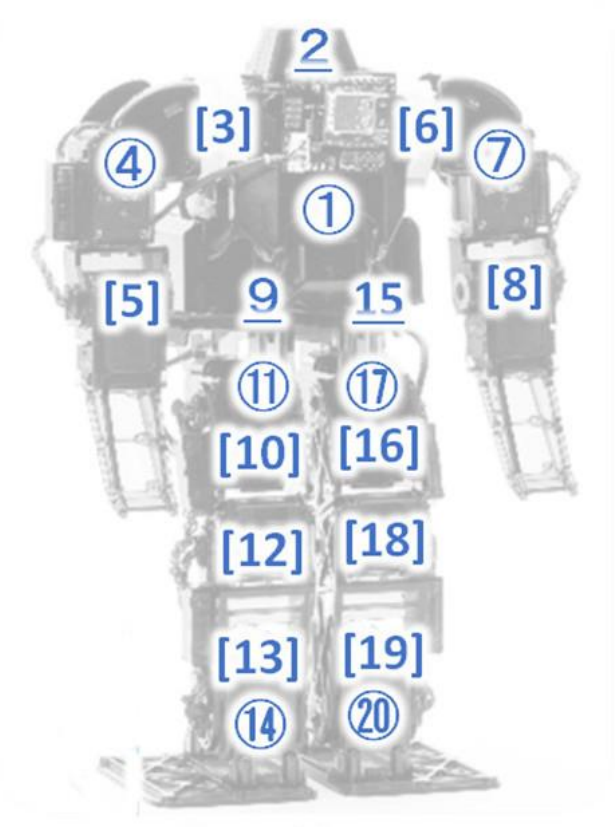
V-Sido CONNECT RC はロボットのサーボモータに命令を送ったり情報を受け取ったりする際、サーボ ID という識別子で対象となるサーボモータを判別しています。このサーボ ID と部位の対応は、各ロボットで異なっています。

そのためシリアルコマンドを作成する場合、動かしたいロボットの各関節がどのようなサーボ ID かを、あらかじめ把握しておくことが必要です。

1-1. GR-001 のサーボ ID

HPI Japan の「GR-001」のサーボ部位の名称とサーボ ID は下図の通りです。ロール/ピッチ/ヨーは、初期姿勢時の体幹からの向きで表しています。

サーボ ID	部位名
1	腰ピッチ
2	首ヨー
3	右上腕ピッチ
4	右上腕ロール
5	右前腕ピッチ
6	左上腕ピッチ
7	左上腕ロール
8	左前腕ピッチ
9	右大腿ヨー
10	右大腿ピッチ
11	右大腿ロール
12	右膝ピッチ
13	右足ピッチ
14	右足ロール
15	左大腿ヨー
16	左大腿ピッチ
17	左大腿ロール
18	左膝ピッチ
19	左足ピッチ
20	左足ロール

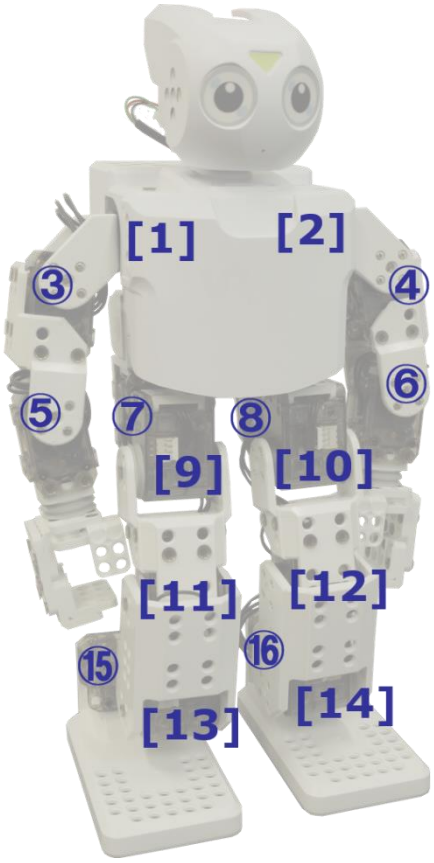


［n］…… ピッチ軸
ⓝ …… ロール軸
n …… ヨー軸

1-2. DARWIN-MINI のサーボ ID

ROBOTIS の「DARWIN-MINI」のサーボ部位の名称とサーボ ID は下図の通りです。ロール/ピッチ/ヨーは、初期姿勢時の体幹からの向きで表しています。

サーボ ID	部位名
1	右上腕ピッチ
2	左上腕ピッチ
3	右上腕ロール
4	左上腕ロール
5	右前腕ロール
6	左前腕ロール
7	右大腿ロール
8	左大腿ロール
9	右大腿ピッチ
10	左大腿ピッチ
11	右膝ピッチ
12	左膝ピッチ
13	右足ピッチ
14	左足ピッチ
15	右足ロール
16	左足ロール



［n］…… ピッチ軸
ⓝ …… ロール軸

2. ロボットの IK の考え方

V-Sido CONNECT RC では、IK を利用してロボットを動かすことができます。IK は Inverse Kinematics の略で、逆運動学を表します。IK により、ロボットの手先や足先などの位置などから、各関節の角度を導き出すことができます。

IK を使えば、個々の関節の角度を一つずつ設定しなくても、指定した部位全体の動きをつくり出すことが可能になるのです。

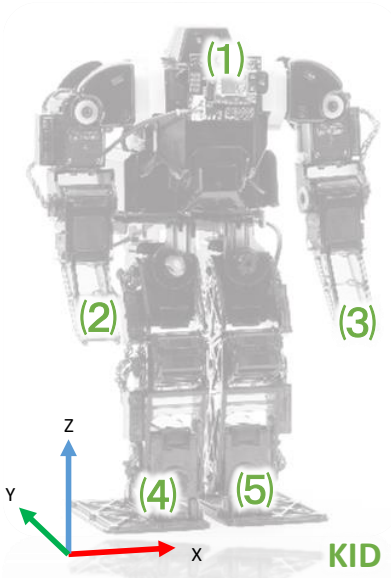
2-1. ロボットの KID

V-Sido CONNECT RC では、IK のコマンドをロボットとやりとりするとき、どの部位への命令なのかを KID で判別しています。KID とはロボットの IK 部位に当てられた識別子のことで、下の表の通り、頭部は 1、右手は 2、左手は 3、右足は 4、左足は 5 となっています。

KID	0	1	2	3	4	5	6	7
部位	体幹	頭部	右手	左手	右足	左足	予約	予約
KID	8	9	10	11	12	13	14	15
部位	予約	予約	予約	予約	予約	予約	予約	予約

この KID は、V-Sido CONNECT RC がサポートしている人型ロボットでは、すべて共通となります（サーボ ID はロボットの種類ごとに異なりますが、KID はどのロボットでも共通です）。

また、IK の値を設定する際、X/Y/Z の 3 軸を用いて指定しますが、その軸の方向は、下図の通りです。



IK を設定する手法として、「位置」「姿勢」「トルク」の 3 つが用意されています。ただし、現在の V-Sido CONNECT RC では、右手、左手、右足、左足の部位に関しては「位置」による IK 設定、頭部に関しては「姿勢」による IK 設定のみをサポートしています。

2-2. 位置による IK 設定

IK を「位置」で設定する場合、KID で指定した部位の先端の位置を、X 軸、Y 軸、Z 軸の値で指定します。このとき指定する値は、－100%～100%の範囲となります。この範囲は、XYZ の各軸で指定した部位の先端が到達できる範囲を示しています。

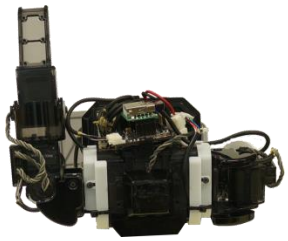
XYZ の指定値と位置の関係の例として、GR-001 の右手を IK で動かしたときの図を下記に示しますので、参考にしてください。

■IK 設定の位置での指定例（右手）

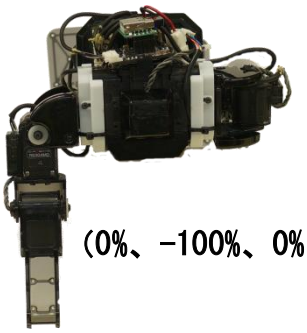
（0%、0%、-100%）

（0%、0%、100%）

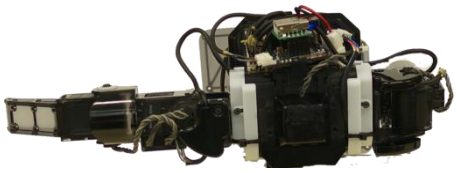
（-100%、0%、0%）



(0%、100%、0%)



(0%、-100%、0%)



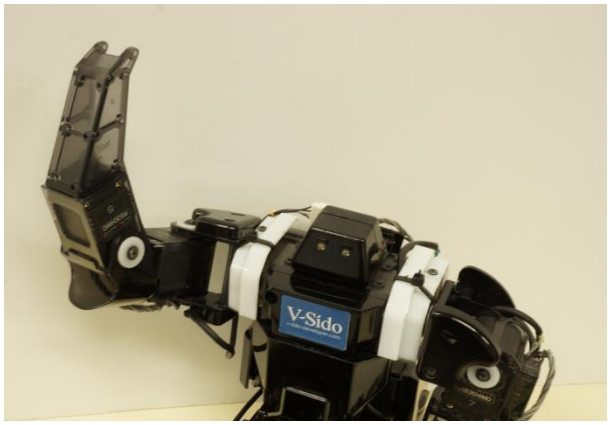
(-100%、0%、0%)

(0%、100%、0%)

(0%、-100%、0%)

(0%、0%、-100%)

(0%、0%、100%)



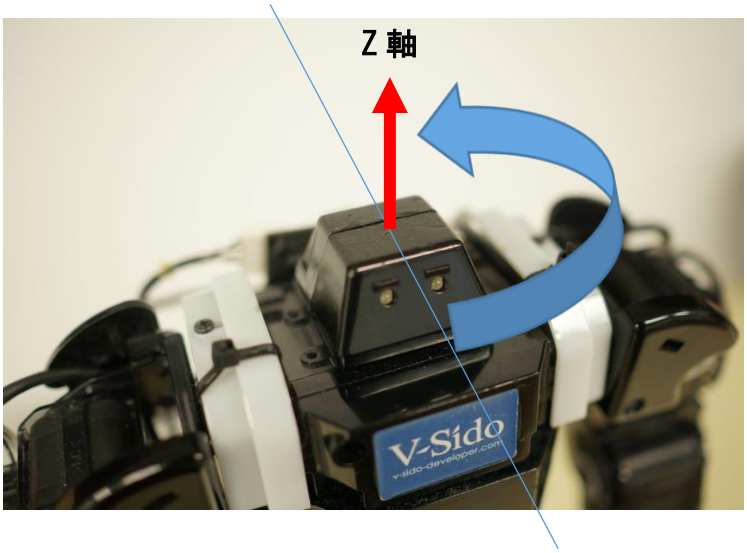
(-50%、-50%、50%)

2-3. 姿勢による IK 設定

IK を「姿勢」で設定する場合、X 軸、Y 軸、Z 軸を回転軸として、各軸をどのように回転させるかで指定します。時計回りを正として、Rx、Ry、Rz をそれぞれ－100%～100%の範囲の値で指定します。

現時点の V-Sido CONNECT RC では、姿勢による IK 設定は、頭部にのみ対応しています。また DARWIN-MINI は頭部が動かないため、GR-001 のみの対応となります。

GR-001 の頭部は Z 軸を回転軸とした動きのみ可能なので、Rx、Ry は 0%固定となり、Rz を－100～100%の範囲で指定することになります。具体的には、0%は正面を向いた状態で、左側を向くほうが正値で 100%は左側に回りきった状態、右側を向くほうが負値で－100%は右側に回りきった状態を示しています。



3. 関節を動かすためのコマンド

3-1. 目標角度設定コマンド作成の手順

ロボットの関節だけを動かしたい場合は、下記の手順でロボットへのコマンドを作成します。

① 対応する関節のサーボ ID を確認

動かしたい関節のサーボ ID を確認します。

② 関節をどのように動かすかを指定する

どのように関節を動かしたいかに応じて、下記項目について任意の値を決定します。

- 目標角度
- 実行サイクル数（目標角度に到達するための時間）

③ サーボの目標角度設定コマンドを作成する

上記①、②の情報を基にコマンドを作成します。

④ コマンドを送信する

③で作成したコマンドを RS-232C シリアル経由で送信することで、ロボットの関節が動きます。

3-2. 目標角度設定コマンドの構成

ここで使用する「サーボの目標角度設定コマンド」に関しては、『「V-Sido CONNECT」コマンドリファレンスマニュアル』の 4-1 項（目標角度設定）で解説されています。

詳細はリファレンスマニュアルに譲りますが、目標角度設定のコマンドは下記のような構造になっています。

目標角度設定のコマンド

バイト数	1	2	3	4	5	6	7	…	n
項目	ST	‘o’	LN	CYC	SID	ANGLE			SUM

8 バイト目以降は、以下の通り対象 SID 分の繰り返しとなる。

8	9	10	…	…	…
SID	ANGLE		…	…	

ST:　パケットの開始(=*0xff*)

OP：　目標角度設定コマンド(=*0x6f*)

LN：　データ長

CYC：目標角度に移行するまでの時間

指定範囲　1～100　（単位は 10msec）

SID：サーボ ID

指定範囲　1～254

ANGLE：目標角度（符号付き 2 バイトデータ）

指定範囲　-1800～1800　（0.1deg 刻み）

SUM：　チェックサム

ユーザーが指定しなければならないのは、

- 4 バイト目の CYC（実行サイクル数）
- 5 バイト目の SID（サーボ ID）
- 6～7 バイト目の ANGLE（目標角度）

の 3 箇所です。これらが先述の手順の①と②で確認および決定した値に相当します。

3-3. 送信コマンド例 1――関節を動かす

たとえば、サーボ ID が 3、目標角度が 30 度（30deg）、実行サイクル数が 20msec の例で解説します。

この場合、4 バイト目の CYC は単位が 10msec なので“2”（＝0x02）、5 バイト目の SID は“3”（＝0x03）となります。また対象のサーボが 1 つだけなのでコマンドの長さは 8 バイトとなり、3 バイト目の LN も“8”（＝0x08）に決まります。

残る 6～7 バイト目の ANGLE ですが、こちらについては注意が必要です。ANGLE の指定範囲は－1800～1800 で 2 バイトデータですが、2 バイトデータに関しては、以下の手順でデータを変換する必要があります（詳細は『「V-Sido CONNECT」コマンドリファレンスマニュアル』を参照）。

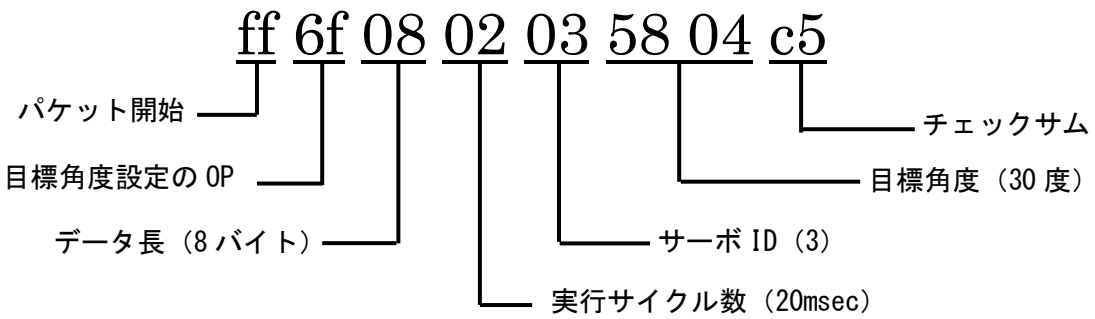
(1) 変換元の 2 バイトデータを 2 進数に変換する
(2) 2 バイトデータ全体を左へ 1bit シフトする
(3) 上位バイトを左へ 1bit シフトする
(4) 送信する場合、上位バイトと下位バイトをそれぞれ 16 進数に変換して、下位バイト→上位バイトの順にセットする

今回の例だと、目標角度が 30deg で 0.1deg 刻みのため、300 を元にして上記の 1～4 の作業を行っていきます。具体的には次のように加工していくことになり、6 バイト目に 0x58 が、7 バイト目に 0x04 をセットすることが分かります。

(1) 「300」→「0000 0001 0010 1100」（符号付き 2 バイトデータ）
(2) 「0000 0001 0010 1100」→「0000 0010 0101 1000」
(3) 「0000 0010 0101 1000」→「0000 0100 0101 1000」
(4) 「0000 0100 0101 1000」→「0x04 0x58」で 0x58、0x04 の順にセット

最後のチェックサム（SUM）は、データの SUM を除く全バイトを XOR したものになります（チェックサムの計算方法は、V-Sido CONNECT のシリアルコマンドすべてで共通）。この例では、0xff,0x6f,0x08,0x02,0x03,0x58,0x04 を XOR して、“0xc5”という値が算出されます。

以上をまとめると、作成すべきコマンドは下記のデータとなります。



このデータをシリアル通信で V-Sido CONNECT に送ることで、ロボットの関節が動きます（サーボ ID3 の関節が 30 度の位置にきます)。

3-4. 送信コマンド例 2――複数の関節を動かす

もちろん、複数のサーボへの命令を同時にまとめて送ることも可能です。たとえば、1 秒（1000msec）かけて、サーボ ID5 の関節を－60 度に、サーボ ID8 の関節を 120 度に動かす例で、コマンドを作成してみましょう。

基本的な考え方は前項とほぼ同じで、指定するサーボ ID ごとに値（ANGLE）を設定します。まずサーボ ID5 のほうですが、－60 度（deg）の 0.1deg 刻みなので、－600 が元の値となり、これを先ほどと同様に 2 バイトデータの変換を行います。今回はマイナスの値となっている点に注意してください。

ちなみに、符号付き 2 バイトデータは最上位ビットが符号を表しており、下の表のような対応となります（最上位ビットが 0 のとき正、1 のとき負）。

値	符号付き 2 バイトデータ
0	0000 0000 0000 0000
1	0000 0000 0000 0001
2	0000 0000 0000 0010
⋮	
128	0000 0000 1000 0000
⋮	
1799	0000 0111 0000 0111
1800	0000 0111 0000 1000
⋮	
32767	0111 1111 1111 1111
−32768	1000 0000 0000 0000
−32767	1000 0000 0000 0001
−32766	1000 0000 0000 0010
⋮	
−1800	1111 1000 1111 1000
−1799	1111 1000 1111 1001
⋮	
−128	1111 1111 1000 0000
⋮	
−3	1111 1111 1111 1101
−2	1111 1111 1111 1110
−1	1111 1111 1111 1111

“－600”を符号付き 2 バイトデータで表すと“1111 1101 1010 1000”となり、下記のように変換していく形となります。

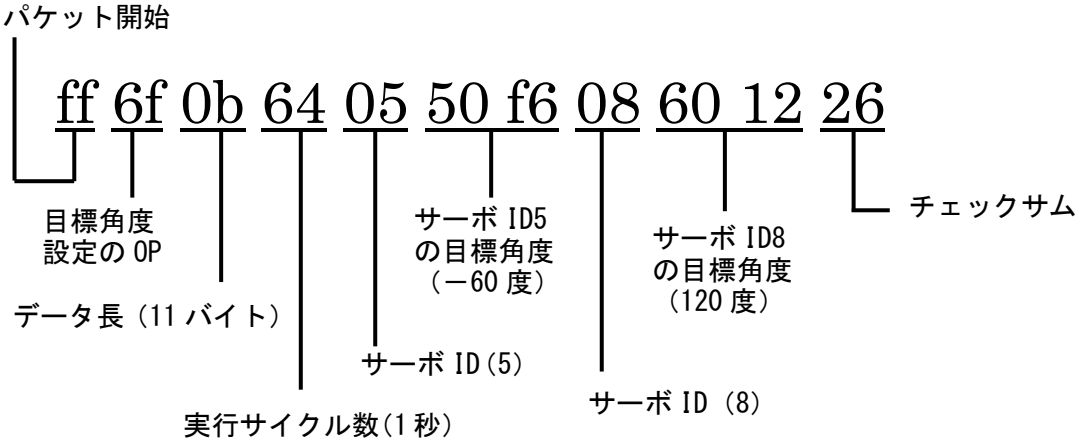
－600 → 1111 1101 1010 1000 → 1111 1011 0101 0000 → 1111 0110 0101 0000 → 0xf6 0x50 → 0x50、0xf6 の順にセット

これで、サーボ ID5 に関しては、SID＝0x05、ANGLE＝0x50 0xf6 と決まります。同様にサーボ ID8 の 120 度についても計算し、こちらは SID＝0x08、ANGLE＝0x60 0x12 となります。

実行サイクル数の CYC は設定できる最長値の 1 秒（1000msec）なので、10msec 単位の 100（＝0x64）を指定します。なお、複数の関節（サーボ）の角度を 1 つのシリアルコマンドで設定する場合、実行サイクル数は同一となります。関節ごとに異なる時間をかけて動かしたい場合は、それぞれ異なるコマンドを作成する必要があります。

最後のチェックサム(SUM)の箇所は、データの SUM を除く全バイトを XOR して、0x26 と算出できます。

以上をまとめると、作成すべきコマンドは以下のようになります。



このデータをシリアル通信で V-Sido CONNECT に送ることで、ロボットの 2 つの関節が同時に動きます（サーボ ID5 の関節が－60 度に、サーボ ID8 の関節が 120 度になります)。

4. 手先を動かすためのコマンド

4-1. IK コマンド作成の手順

ロボットの手先を動かしたい場合、手に関する IK コマンドを作成します。その手順は下記の通りです。

① 対応する KID を確認

動かしたいロボットの部位の KID を確認します。

② IK の指定方法を決める

位置、姿勢、トルクの 3 つのうち、どの方法で IK を指定するかを決めます。今回は手先を動かすので、位置で指定することになります。

③ 手先をどの位置に動かすかを指定する

手先を持っていく目標位置を、X 軸、Y 軸、Z 軸の値で指定します（それぞれ－100～100%で指定します)。

④ IK コマンドを作成する

上記①、②、③の情報を基にコマンドを作成します。

⑤ コマンドを送信する

④で作成したコマンドを RS-232C シリアル経由で送信することで、ロボットの手先が動きます。

4-2. IK コマンドの構成

ここで使用する「IK コマンド」に関しては、『「V-Sido CONNECT」コマンドリファレンスマニュアル』の 4-13 節（IK 設定）で解説されています。

詳細はリファレンスマニュアルに譲りますが、IK コマンドは下記のような構造になっています。

IK コマンド

バイト数	1	2	3	4	5	6	…	8/11/14		n
項目	ST	'k'	LN	IKF	KID	KDT			…	SUM

KDT は IKF の値に応じて 0/3/6/9 バイトのいずれかの長さとなる。
6/9/12/15 バイト目以降は、以下の通り対象 KID と KDT の繰り返しとなる（IKF が現在値要求のみの場合 KDT は含まれない）。

6/9/12/15	-/10/13/16	…	…
KID	KDT		…

ST： パケットの開始(=*0xff*)
OP： IK 設定コマンド(=*0x6b*)
LN： データ長
IKF： IK 設定フラグ
　IK 情報の設定もしくは取得の区別を行うためのフラグ。
　指定範囲　0～63
KID： IK 部位の ID
　指定範囲　0～15
KDT： IK 用の設定データ（オプションで 0～9 バイト長）
　IKF で目標値設定を行った場合のみ有効で、対応する情報の 3 軸（x/y/z）の値を各バイトに設定する。
　指定範囲　各バイト 0～200　（-100%が 0、100%が 200 に相当）
SUM： チェックサム

ユーザーが指定しなければならないのは、
- 4 バイト目の IKF（IK 設定フラグ）
- 5 バイト目の KID（IK 部位の ID）
- 6 バイト目以降の KDT（IK 用の設定データ）

の 3 箇所です。これらがそれぞれ先述の手順②、①、③で確認および決定した値に相当します。

4-3. 送信コマンド例 3――片手の手先を動かす

たとえば、左手の手先を真上に持っていきたいというケースで具体的なコマンドを考えてみます。

この場合は、左手を動かすので 5 バイト目の KID は左手を示す“3”（＝0x03）となります。4 バイト目の IKF（IK 設定フラグ）は、下記の表を基に、設定する値が決まります。

	MSB							LSB
bit	0	1	2	3	4	5	6	7
意味づけ	予約		現在値要求			目標値設定		
	0	0	トルク	姿勢	位置	トルク	姿勢	位置

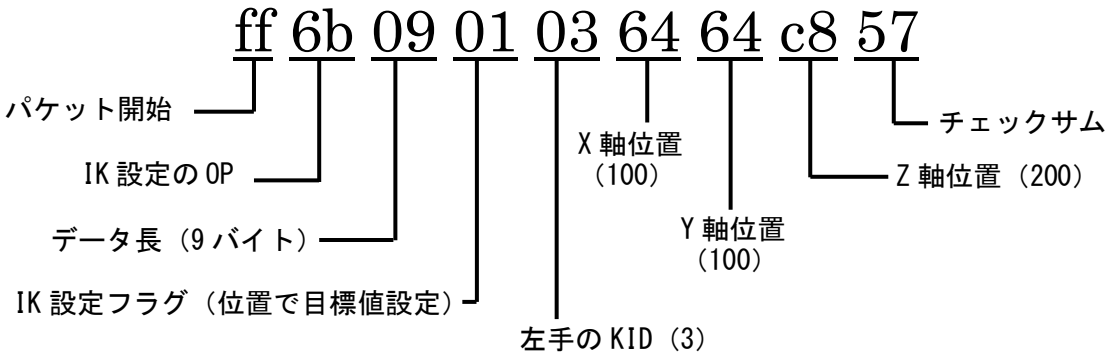
この表の各 bit に対して、無効の箇所は“0”、有効の箇所は“1”を入れたときの値が IKF に入る値となります。

今回は「位置」で「目標値設定」を行うので、最後の bit7 を有効にして、それ以外は無効にします。そうすると「00000001」となり、10 進数で“1”（＝0x01）が IKF に入る値となります。

IKF で位置による目標位置設定のみのフラグが立っているため、KDT のデータ長は 3 バイトとなります（姿勢とトルクの設定は入らない）。手の IK は、手先の位置を X 軸、Y 軸、Z 軸で指定し、今回は真上の位置なので、X 軸は 0%、Y 軸は 0%、Z 軸は 100%とします。KDT は 0～200 の範囲で指定するので、実際には X 軸の 6 バイト目は“100”（＝0x64）、Y 軸の 7 バイト目は“100”(=0x64)、Z 軸の 8 バイト目は“200”（＝0xc8）という値をセットすることになります。

そして 3 バイト目の LN にはコマンドの長さの“9”（＝0x09）が入ります。最後の 9 バイト目のチェックサムは、データの SUM を除く全バイトを XOR すると 0x57 が算出されるので、この値が入ります。

以上をまとめると、作成すべきコマンドは以下のようなデータとなります。



このデータをシリアル通信で V-Sido CONNECT に送ることで、ロボットの手先が動きます（左手が真上の位置にきます)。

4-4. 送信コマンド例 4――両手の手先を動かす

IK コマンドは同時に複数の部位に対して設定することも可能です。たとえば、右手をまっすぐ前方に伸ばし、同時に左手をまっすぐ横に伸ばした姿勢を例に、コマンドを作成してみましょう。

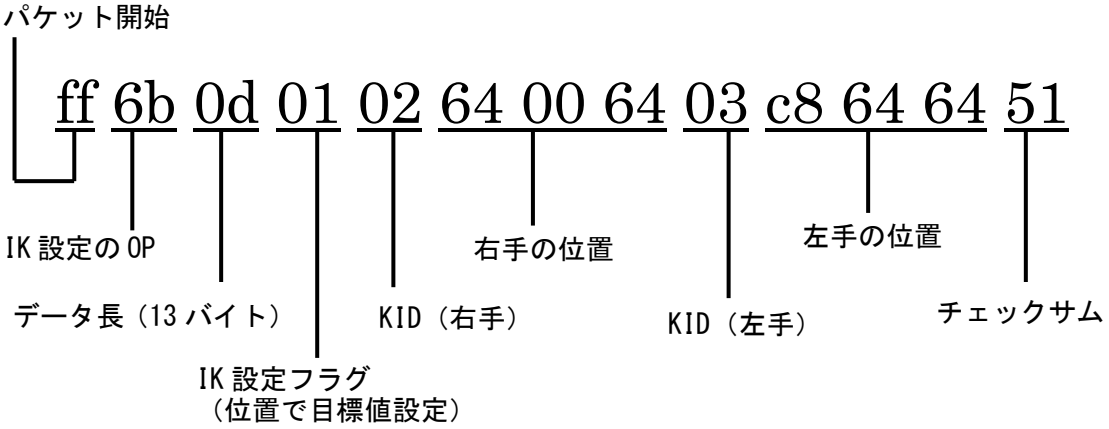
KID に関しては、右手は“2”（＝0x02）、左手は“3”（＝0x03）となります。

手の位置は、右手はまっすぐ前方なので、X 軸：0%、Y 軸：－100%、Z 軸：0%とします（それぞれ 0x64、0x00、0x64 をセット）。左手はまっすぐ真横に伸ばすので、X 軸：100%、Y 軸：0%、Z 軸：0%とします（それぞれ 0xc8、0x64、0x64 をセット）。

IKF は、右手、左手ともに「位置」で「目標値設定」を行うので、先ほどの例と同様のフラグとなり、“1”（＝0x01）となります。

データ長は、この例では対象部位が 2 つあり KID/KDT が 2 組入るために、“13”（＝0x0d）となります。チェックサムは、データの SUM を除く全バイトを XOR して算出される 0x51 が入ります。

以上をまとめると、作成すべきコマンドは以下のようなデータとなります。



このデータをシリアル通信で V-Sido CONNECT に送ることで、ロボットの手先が動きます（右手を前方に伸ばし、左手を真横に伸ばす）。

4-5. 送信コマンド例 5――手先と頭部を同時に動かす

ここまで手先のみを動かす事例でコマンドを見てきましたが、手先と同時に頭部を動かすケースを考えてみましょう。例として、左手を真横に伸ばしながら頭部を左側に向ける IK コマンドを作成します。

まず KID ですが、左手は“3”（＝0x03）、頭部は“1”（＝0x01）となります。

IKF については、頭部は手先と異なり、「位置」ではなく「姿勢」で IK 設定を行うので注意が必要です。今回は手先と頭部を両方動かすため、IKF のフラグの bit6 と bit7 を有効とします。つまり、「00000011」を 10 進数に変換した“3”（＝0x03）が IKF の値となります。

KDT についても、位置と姿勢の 2 つのフラグが有効となっているため、下記のような 6 バイト長のデータとなります。

バイト	0	1	2	3	4	5
内容	位置（-100～100%）			姿勢（-100～100%）		
	x	y	z	Rx	Ry	Rz

KDT は、左手に関しては位置で、頭部に関しては姿勢で指定します。

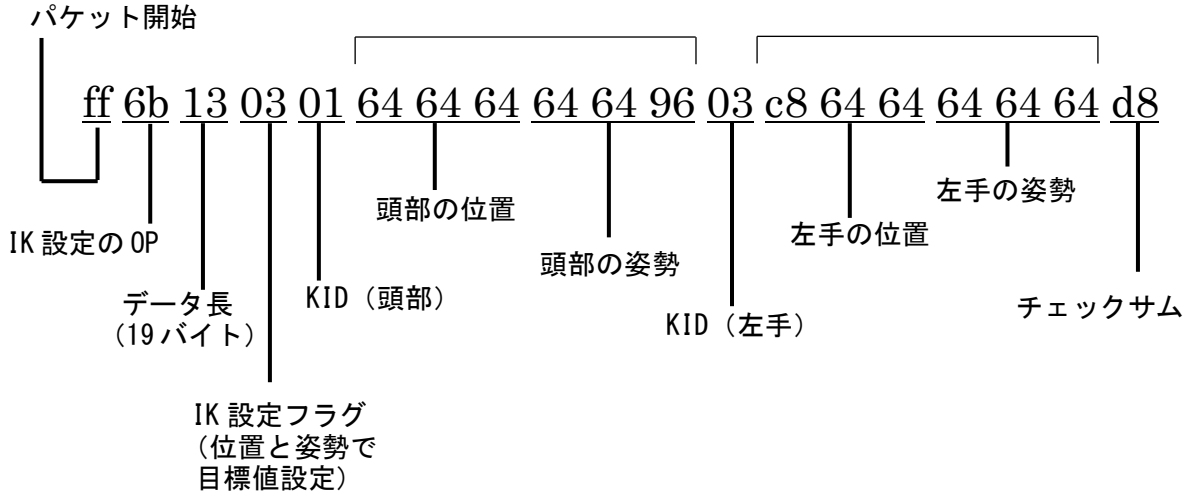
今回の例では、左手は真横に伸ばすので、位置を x：100%、y：0%、z：0%で指定します（それぞれ 0xc8、0x64、0x64 を各バイトにセット）。左手の姿勢（Rx/Ry/Rz）は利用しないので、これらの値はすべて 0%（=0x64）をセットします。

また頭部は左側に向けるので、姿勢を Rx：0%、Ry：0%、Rz：50%とします（それぞれ 0x64、0x64、0x96 を各バイトにセット）。頭部の位置（x/y/z）は利用しないので、これらの値はすべて 0%（=0x64）をセットします。

以上で、左手と頭部のそれぞれの KDT の値か決まりました。

パケット全体のデータ長は、この例では“19”（＝0x13）となります。チェックサムは、データの SUM を除く全バイトを XOR して算出される 0xd8 が入ります。

以上をまとめると、作成すべきコマンドは以下のようなデータとなります。



このデータをシリアル通信で V-Sido CONNECT に送ることで、ロボットの手先と頭部が同時に動きます（左手を真横に伸ばし、頭部を左側に向ける）。

5. ロボットを歩かせるためのコマンド

5-1. 歩行コマンド作成の手順

ロボットを歩行させたい場合、歩行コマンドを作成します。その手順は下記の通りです。

① ロボットをどのように歩かせるかを指定する

どのように歩行するか、下記項目について任意の値を決定します。このパラメータはロボットの最高速度・最高旋回速度を 100%として、その割合で表しています。後退する場合は、前進速度をマイナスの値で指定します。

- 前進速度（－100～100%）
- 旋回速度（－100～100%）

※旋回は右回りを正とします

② 歩行コマンドを作成する

上記①の情報を基にコマンドを作成します。

③ コマンドを送信する

②で作成したコマンドを RS-232C シリアル経由で送信することで、ロボットが歩行します。

5-2. 歩行コマンドの構成

ここで使用する「歩行コマンド」に関しては、『「V-Sido CONNECT」コマンドリファレンスマニュアル』の 4-14 項（移動情報指定）で解説されています。

詳細はリファレンスマニュアルに譲りますが、歩行コマンドは下記のような構造になっています。

歩行コマンド

バイト数	1	2	3	4	5	6		n
項目	ST	't'	LN	WAD	WLN	WDT1	…	SUM

WLN で指定したバイト数が 2 バイト以上の場合、7 バイト目以降に WLN で指定した分だけ WDT を入れる。

7	8	…	…
WDT2	…	…	…

ST： パケットの開始(=*0xff*)

OP： 移動情報指定コマンド(=*0x74*)

LN： データ長

WAD： 歩行情報データの先頭アドレス

指定範囲 0～1 （RC 版の仕様で、今後拡張予定）

WLN： 歩行情報データのデータ長

指定範囲 1～2 （RC 版の仕様で、今後拡張予定）

WDT： 歩行情報設定データ

WAD で指定した先頭アドレス、WLN で指定したデータ長に従い、値を設定する。RC 版では、下記の歩行情報を設定できる。

アドレス	0	1
指定範囲	前後の速度	旋回の速度
	0～200	0～200

最大速度を 100%として、設定値 0 が－100%、設定値 100 が 0%、設定値 200 が 100%に対応する。旋回は時計回りが正、反時計回りが負となる。

SUM： チェックサム

V-Sido CONNECT RC では、前後の移動と旋回のみをサポートしています（RC 版の仕様で、今後拡張予定です）。ユーザーが指定しなければならないのは、WAD、WLN、WDT ですが、これらの値は歩行の形態によって変わってき

ます。RC 版では下記の 3 パターンに分けることができます。

パターン A：前後の移動のみ

WAD は“0”、WLN は“1”となり、WDT は 1 バイトで前後の速度を指定します

パターン B：旋回のみ

WAD は“1”、WLN は“1”となり、WDT は 1 バイトで旋回速度を指定します

パターン C：前後に移動しながら同時に旋回も行う

WAD は“0”、WLN は“2”となり、WDT は 2 バイトで、WDT1 で前後の速度、WDT2 で旋回速度を指定します

なお、パターン C の書式で「前後の移動のみ」もしくは「旋回のみ」を実現することも可能です。その場合は、前後移動のみにしたいときは WDT2 の値を 0%（“100”＝0x64）に、旋回のみにしたいときは WDT1 の値を 0%（“100”＝0x64）に設定します。

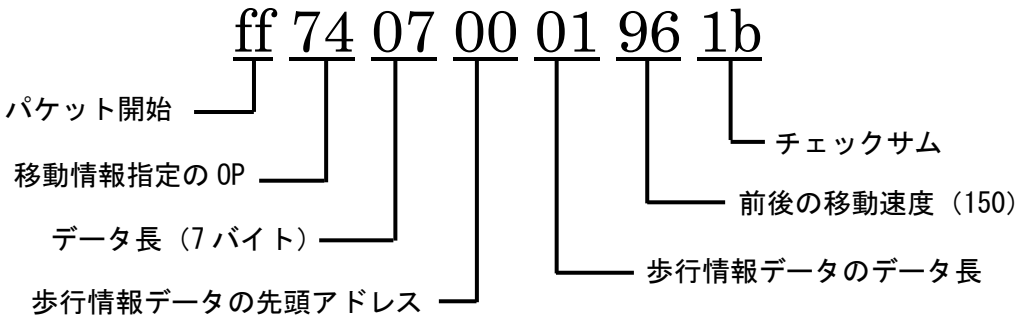
5-3. 送信コマンド例 6――まっすぐ前進させる

たとえば、最大速度の 50%の速さでロボットを前進させたいケースを例に、具体的なコマンドを作成してみましょう。

前後のみの動きなので、WAD は“0”、WLN は“1”となります。速度は前進 50%なので、WDT は 1 バイトのみで“150”（＝0x96）が入ります（－100%が“0”、0%が“100”、100%が“200”に相当します）。

コマンドの長さは 7 バイトなので LN には“7”（＝0x07）が入ります。チェックサムは SUM を除いた全バイトの XOR を計算した結果の 0x1b です。

以上をまとめると、作成すべきコマンドは以下のようなデータとなります。



このデータをシリアル通信で V-Sido CONNECT に送ることで、ロボットが歩行します（最大速度の 50%の速さで前進）。

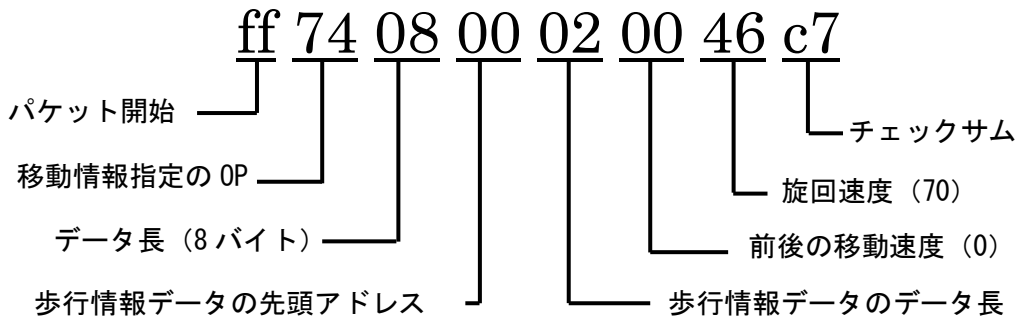
5-4. 送信コマンド例 7――前後移動と旋回を同時に実行する

もう 1 つ、前後と旋回を同時に行うケースも見てみましょう。最大速度で後退しながら最大旋回速度の 30%の速さで左に旋回する場合のコマンドを作成します。

この場合、WAD は“0”、WLN は“2”、WDT は 2 バイトで WDT1 には前後の速度である－100%を示す“0”（＝0x00）が、WDT2 には旋回速度である－30%を示す“70”（＝0x46）が入ります。

コマンドの長さは 8 バイトなので LN には“8”（＝0x08）が入り、チェックサムの SUM は SUM を除いた全バイトの XOR を計算して 0xc7 が入ります。

以上をまとめると、作成すべきコマンドは以下のようなデータとなります。



このデータをシリアル通信で V-Sido CONNECT に送ることで、ロボットが左に旋回しながら後退します。

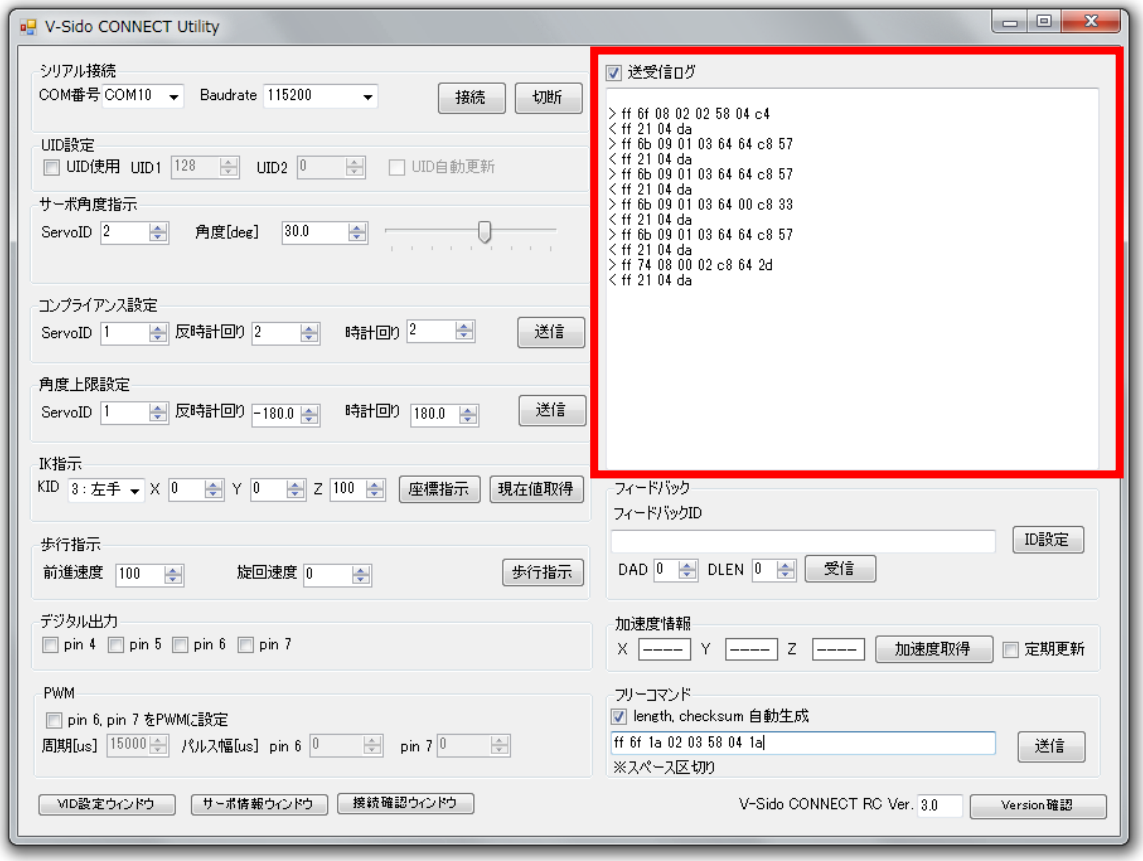
6. シリアルコマンド作成に関するヒント

6-1. 「V-Sido CONNECT Utility」を用いたコマンドの確認方法

シリアルコマンドによるロボットの動作を確認するときなどは、V-Sido CONNECT のユーティリティツール「V-Sido CONNECT Utility」を使うのが便利です。

V-Sido CONNECT Utility では GUI 画面でパラメータを指定して、V-Sido CONNECT やロボットの動作確認を行えますが、「送受信ログ」のチェックボックスを ON にすれば、シリアル通信でやり取りされているデータの中身を見ることができます。

ログ表示画面で、「＞」で示されるのが送信データ（To V-Sido CONNECT）で、「＜」で示されるのが返信データ（From V-Sido CONNECT）です。



また、「フリーコマンド」の箇所にコマンドを直接記入して、実際のロボットの挙動などを確認することも可能です。フリーコマンドでは、「length、checksum 自動生成」のチェックを ON にすることで、LN や SUM を自動的に

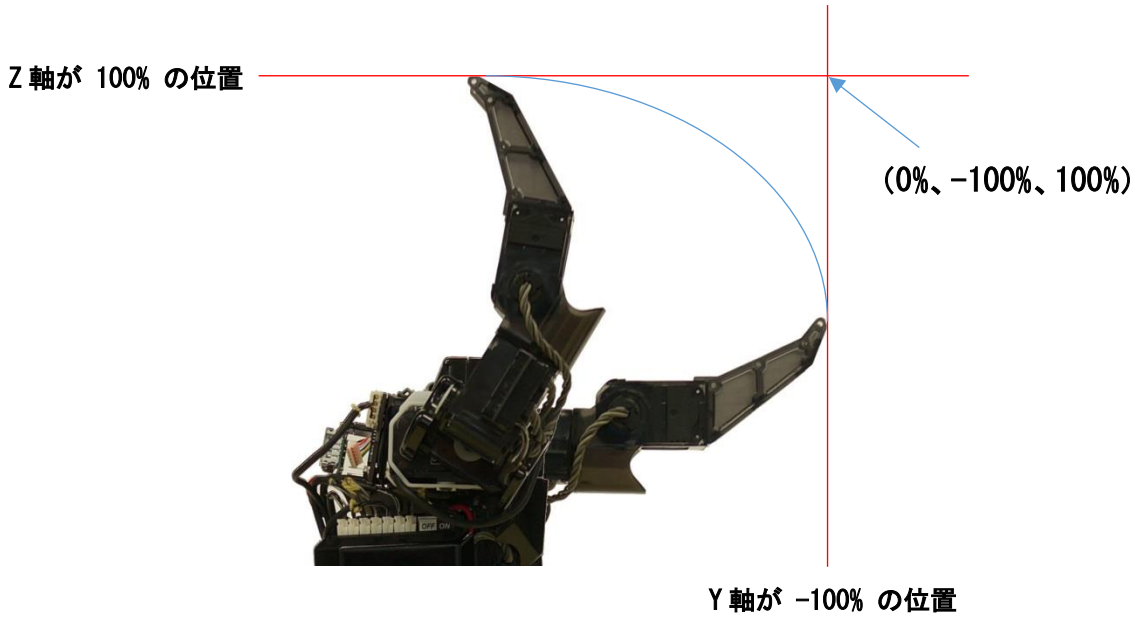
補完する機能も備えています。LN と SUM の場所に適当なダミー値を入れてコマンドを記入して「送信」ボタンを押せば、正しい LN と SUM に変換されたコマンドが発行されます。

V-Sido CONNECT Utility は、V-Sido 開発者支援サイト「V-Sido Developer」（https://v-sido-developer.com/）より無償で入手できるので、ぜひご活用ください（Windows 版のみ）。

6-2. IK コマンドにおける目標位置と可動範囲

IK コマンドを使って手先などの目標位置で IK 設定を行った場合、指定した目標位置と、実際の手先の位置が異なることもあるので注意してください。

たとえば IK コマンドで、左の手先の目標位置について、X 軸を 0%、Y 軸を－100%、Z 軸を 100%で指定したケースを例に考えてみましょう。Y 軸の－100%は手を前方にいっぱいまで伸ばした位置を示し、Z 軸の 100%は手を上方にいっぱいまで伸ばした位置を示します。そのため、「X 軸：0%、Y 軸：－100%、Z 軸：100%」の位置に手先を持っていくことは物理的に不可能です。



こうしたコマンドを受信した場合、V-Sido CONNECT はロボットの手先を「X 軸：0%、Y 軸：－100%、Z 軸：100%」の位置になるべく近づけるように、手先を動かします。そのため、コマンドで指定した位置の値と、実際の位置の値が異なってくるのです。

たとえば、V-Sido CONNECT Utility を使って、IK 指示で左手を「X 軸：0%、

Y 軸：－100%、Z 軸：100%」にもっていくコマンドを発行してみてください（この例ではロボットは GR-001 を使用）。

```
> ff 6b 09 01 03 64 00 c8 33
< ff 21 04 da
```

上記のコマンドが発行され、ロボットの左手が動きます。そしてその直後に、IK 指示で「現在値取得」ボタンを押すと、次のようなコマンドのやり取りとなり、実際の左手の手先は「X 軸：0%、Y 軸：－70%、Z 軸：70%」にあることがわかります（返信データの 6～8 バイト目が位置を意味しており、0x64 は 100 で 0%、0x1e は 30 で－70%、0xaa は 170 で 70%を示す）。

```
> ff 6b 06 08 03 99
< ff 6b 09 08 03 64 1e aa 46
```

なお、ここで示した例のほかにも、目標位置と実際の位置が異なるケースはあります。IK による目標値の指定は、あくまで“目標”とする値だと考えてプログラミングするとよいでしょう。

6-3. IK コマンドを使ったモーション作成

ロボットのモーションを作成するときは、IK コマンドが便利です。モーションにおけるキーフレームを IK コマンドで作成し、それを連続して再生することでロボットの流れるような動作を実現することが可能です。

たとえば「ロボットがパンチをする動作を実現したい」ときは、以下のような 3 つの IK コマンドを作成します。

① 通常時の姿勢をとる IK コマンド
② パンチをしている途中の姿勢をとる IK コマンド
③ パンチの動作を完了したときの姿勢をとる IK コマンド

これらのコマンドをシリアル通信によって、①→②→③の順に V-Sido CONNECT に送ることで、ロボットがパンチをする動作をとります。

関節の目標角度設定と違って、IK コマンドでは目標値に達するまでの時間を指定できません。IK コマンドで作成したロボットのモーションの動きに速さのメリハリをつけたい場合は、キーフレームへの移行タイミングなどで調整してみてください。

一般にモーションの作成は、キーフレームを増やしたりコマンド送信のタイミングを調整したりするなど試行錯誤して、イメージに合ったものを作り込むことになります。

6-4. 連続歩行と歩行の停止

1回だけ歩行コマンドを送ると、ロボットは一定時間の歩行動作を続けたあとに停止します（より正確に言えば、V-Sido CONNECT が最後に歩行コマンドを受信してから一定時間後に歩行を止めます)。この時間は利用するロボットやパラメータなどの条件によって異なりますが、おおよそ3秒程度です。

そのため、ロボットを連続して歩行させたい場合は、歩かせる間は歩行コマンドを繰り返し送り続けるようにしてください。

また、すぐに歩行を停止させたい場合は、前進速度0%、旋回速度0%に設定した歩行コマンドを送ります。

6-5. 歩行の「速度」とパラメータ

歩行コマンドでは前後移動や旋回を“速度”の割合で指定していますが、これはサーボモータを動かす速度を変えるわけではなく、実質的には“歩幅”で速度を調整しています。

ちなみに、V-Sido OS を使ったロボット操作アプリケーション「V-Sido Lite」では、歩幅だけでなく歩調や足上高さなど、さまざまな歩行設定が可能となっていますが、ファームウェアバージョン 3.0 時点の V-Sido CONNECT RC では未実装です。『「V-Sido CONNECT」コマンドリファレンスマニュアル』にも記載されている通り、V-Sido CONNECT RC の歩行コマンドは今後拡張される予定なので、バージョンアップでの対応をお待ちください。

・ 各ロボットの詳細については、ロボットに付属する取扱説明書を参照してください。
・ 記載された社名、製品名は一般に各社の商標または登録商標です。

「V-Sido CONNECT RC」制御コマンド作成の手引き

アスラテック株式会社
〒101-0042　東京都千代田区東松下町 45