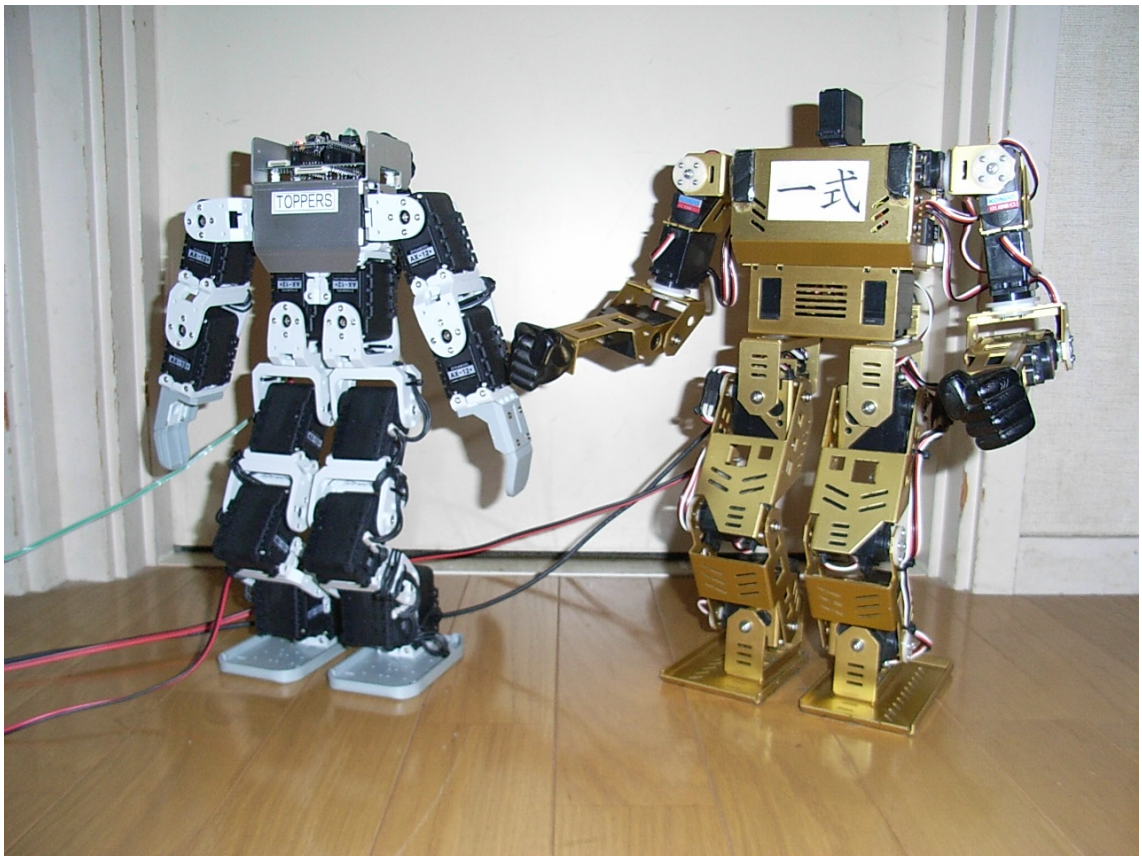
二足歩行ロボットトッパー君仕様書



(株)リコー　GJ事業部
竹内　良輔

００.変更履歴

履歴表

版	日付	作成者	変更内容
1.0	2007/02/10	竹内良輔	初版
1.1	2007/02/11	竹内良輔	ロボット制御コマンドの説明を追加
1.2	2007/02/24	竹内良輔	誤字等修正
1.3	2007/03/11	竹内良輔	タスクモニタの節を追加

０.目次

１．概要

今後、組込み教育の場で二足歩行ロボットの開発を行っていく方を対象として、二足歩行ロボット用ソフトウェア開発の参考となるように、二足歩行ロボット用ソフトウェアのフレームワーク化を行いました。対象としたロボットは ESP 企画「はじめロボット９号」、ソフィアシステム「eMasterGEAR」の２台の人型ロボットです。フレームワークはモーション機能と二足歩行機能に対応しています。二足歩行制御は足の部分の自由度が１０以上であるロボットを対象としています。また、制御用のコントローラとして低価格の３２ビットＣＰＵでもＲＴＯＳを用いて制御できるようにプログラムの高速化を行いました。

二台のロボットの制御ボードの仕様は以下の通りです。ロボット制御に関連した項目のみ列挙します。

	はじめロボット９号	eMasterGEAR
サーボモータ	KRS2346 ×21	BTX030 Dynamixel AX-12 ×16
制御ボード	CROBO-H3	BTE064 SAM7S I/O ボード
CPU	SH2(SH7047F) 40MHz	AT91SAMS128(ARM7) 外部 18.4 MHz
ROM	256KB(内部)	128KB(内部)
RAM	12KB(内部)	32KB(内部)
	128KB(外部)	
EEPROM	512KB	
サーボ制御	PWM 方式×24	１ワイヤ双方向 TTL レベル
周辺 I/F	RS232C×3 モニタ用：１ 無線用：１	RS232C×3 モニタ用：１ サーボ制御用：１ ZigBEE 用：１

表 1-1.ロボットコントローラの概要

ＲＴＯＳとして TOPPERS/JSP-1.4.2 を使用しました。TOPPERS/JSP の実装モードで ROM+RAM サイズで１２８ＫＢ以内に収まるようにプログラムを作成しています。

「はじめロボット９号」の場合、内蔵ＲＯＭにＲＯＭモニタを書込み、ロボット制御用のプログラムはＲＯＭモニタを使ってダウンロードして実行する形式にし、内蔵ＲＯＭの書込み回数問題を回避しました。ＲＯＭモニタは苫小牧高専で開発したＨ８用のＲＯＭモニタをＳＨ２用に改造したものを使用しました。「eMasterGEAR」の場合は ROM 実行の形になっています。

ロボットの通信に関しては、ＲＳ２３２Ｃのみサポートしています。「eMasterGEAR」のＯＳ版では ZigBEE の対応を行いました。種々の拡張は各自で改造をお願いします。

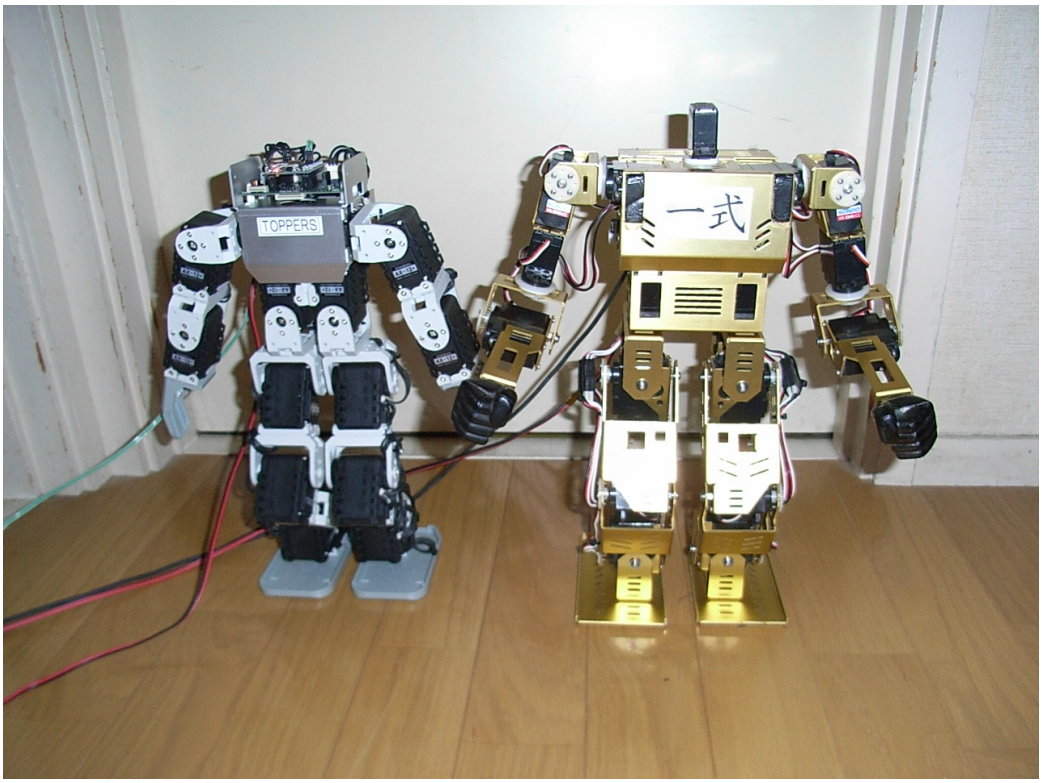


写真 1-1.右：TOPPERS 君１号（はじめロボット９号）
左：TOPPERS 君２号（eMasterGEAR）

1-1.開発環境

このソフトウェアは Windows-XP 上に Cygwin を載せ、ＳＨ２とＡＲＭ７用のＧＮＵツールと newlib をインストールして開発しました。また、LINUX でのビルド確認も行っています。その関係でソースファイルの漢字はＥＵＣとなっています。また、プログラムの環境はCygwinやLINUX上でTOPPERS/JSPを開発する環境とまったく同じです。開発環境のインストール手順については TOPPERS/JSP のドキュメント gnu_install.txt を参照してください。

以下にダウンロードプログラムのディレクトリ構成を示します。jsp-1.4.2 以下は TOPPERS/JSP1.4.2 をダウンロードして、そのディレクトリに二足歩行ロボットの jsp-1.4.2 の内容を上書きしてください。

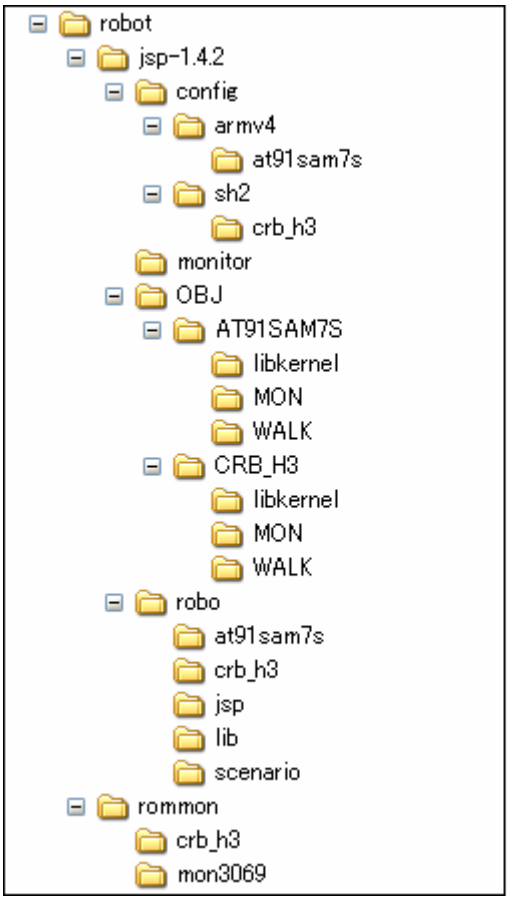


図 1-1.二足歩行ロボットダウンロードプログラムのディレクトリ構成図

筆者の Cygwin 上の開発環境のディレクトリ構成を図 1-2 に示します。筆者の場合は TOPPERS/JSP 1.4.2 をダウンロード後、ルートなるディレクトリ名を jsp から jsp-1.4.2 に書き換えて使用しています。筆者の開発環境では JSP1.4 や JSP1.4.1 の開発環境を並行して使用しているため名前を書き換えて区別をしています。TOPPERS/JSP 1.4.2 には H8 の教材で使用した TINET 1.3 がセットアップして、その上に上記の二足歩行ロボットを上書きしています。

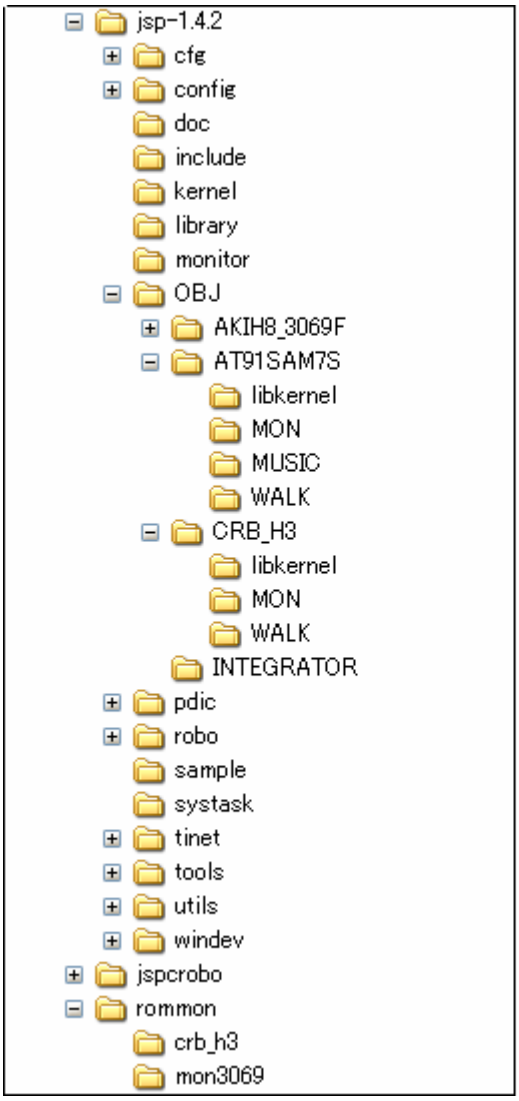


図 1-2.筆者の開発環境のディレクトリ構成図

「はじめロボット９号」用のＲＯＭモニタが rommon/crb_h3 以下にあります。「はじめロボット９号」用の実装を行う方は、まず、rommon/crb_h3/rommon.srec を「はじめロボット９号」に同梱された FW7047.exe を用いて CRB-H3 ボードに書き込んでください。書込み後、ＲＳ２３２Ｃと Tera Term を用いてモニタ操作可能となります。

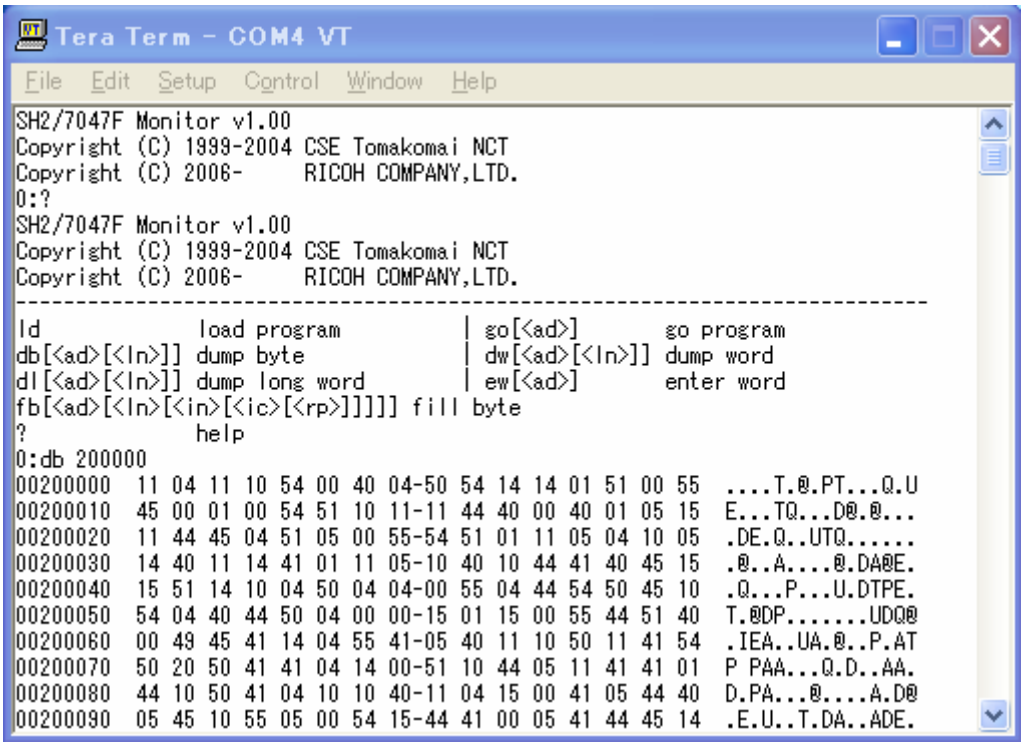


図 1-3.「はじめロボット９号」用のＲＯＭモニタの画面

この開発環境では、「はじめロボット９号」用のコントローラである CRB-H3 の TOPPERS/JSP の実装プログラム、CRB-H3 用の二足歩行プログラム、CRB-H3 用の TOPPERS/JSP 版の二足歩行プログラム、CRB-H3 用のＲＯＭモニタ、「eMasterGEAR」用のコントローラである AT91SAM7S の TOPPERS/JSP の実装プログラム、AT91SAM7S 用の二足歩行プログラム、AT91SAMS7S 用の TOPPERS/JSP 版の二足歩行プログラムが作成可能です。以下に、個々のプログラムに対応した開発ディレクトリの表を載せます。開発手順に関してはダウンロードにある readme.txt を参照してください。

jsp-1.4.2/robo/at91sam7s	RTOS なし at91sam7s プログラム開発ディレクトリ
jsp-1.4.2/robo/crb_h3	RTOS なし crb_h3 プログラム開発ディレクトリ
jsp-1.4.2/jsp	ロボットライブラリィ作成ディレクトリ
jsp-1.4.2/lib	STDIO 等のライブラリィ作成ディレクトリ
jsp-1.4.2/AT91SAM7S/MON	at91sam7s 用タスクモニタ作成ディレクトリ
jsp-1.4.2/AT91SAM7S/WALK	RTOS 版 at91sam7s プログラム作成ディレクトリ
jsp-1.4.2/CRB_H3/MON	crb_h3 用タスクモニタ作成ディレクトリ
jsp-1.4.2/CRB_H3/WALK	RTOS 版 crb_h3 プログラム作成ディレクトリ
rommon/crb_h3	crb_h3 用 ROM モニタ作成ディレクトリ
rommon/mon3069	H8-3069F 用 ROM モニタ作成ディレクトリ

表 1-2.ダウンロード物ディレクトリ構成

1-2.成果物について

このプログラムの作成にあたって、モーション実行機能のデータ互換性のために、データ構造を「はじめロボットのオリジナルプログラム」を参考にしました。また、各コントローラのハードウェア依存部のドライバプログラムは各々のロボットのオリジナルのプログラムを流用しています。

２．プログラム構成

2-1.二足歩行プログラムの構成

二足歩行フレームワークは、以下の３つのパーツから成り立っています。各パーツはディレクトリごとに分られています。そのため、機種を増やす場合はデバイス依存部の追加、動作を増やすにはシナリオプログラムの追加、というように分割開発が可能な形に構成を行っています。

パーツ	機能	対応ディレクトリ
基本動作部	初期化を行う。 関節に設定された動作を時間に従ってサーボモータに設定を行う。 制御コマンドの解析を行う	jsp-1.4.2/robo
シナリオ	個別の動作（歩く、姿勢の設定等）ごとに詳細なシナリオを設定するプログラム郡	jsp-1.4.2/robo/scenario
デバイス依存部	サーボ制御、EEPROM などのデバイスを制御するプログラム	jsp-1.4.2/robo/at91sam7s jsp-1.4.2/robo/crb_h3

表 2-1.二足歩行プログラムの構成

サーボモータへの設定は、「はじめロボット９号」では１６ＭＳＥＣ周期、「eMasterGEAR」では２０ＭＳＥＣ周期に行われます。TOPPERS/JSP 上で実行する場合、この処理が最上位タスクで実行され、他のプログラム（タスク）はサーボモータへの設定タスクの空き時間に処理されます。もちろん、ロボットが停止状態ではアクチュエータの設定時間は短く、歩行などの座標計算を行う重い動作の場合はアクチュエータの設定時間が長くなり、他のタスクの実行を詐害する場合があります。このような構成で TOPPERS/JSP への実装を行うため、二足歩行フレームワークはライブラリィ化が可能となります。

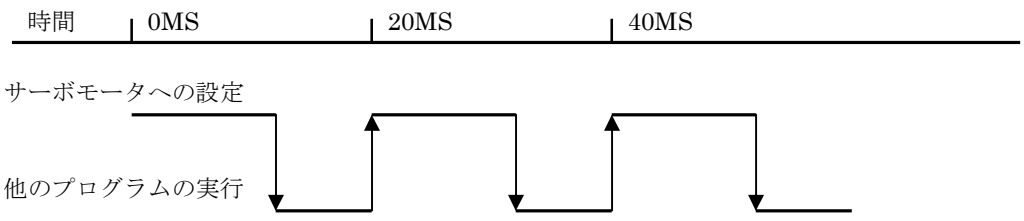
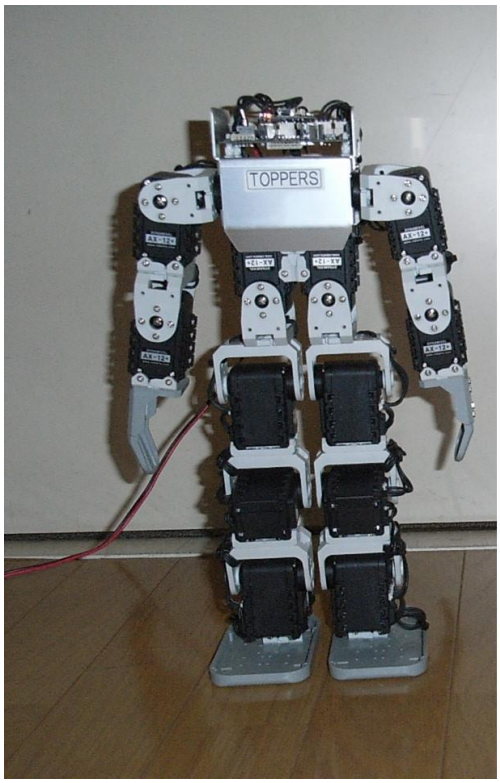
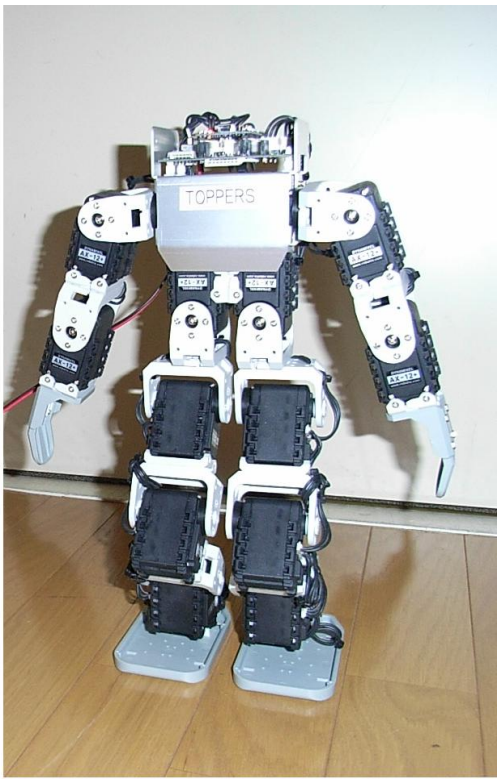


図 2-1.「eMasterGEAR」のプログラムの実行時間

次に、ロボットの姿勢の説明を行います。このプログラムでは２つの基本姿勢があります。一つは直立姿勢で、もう一つが歩行準備姿勢です。二足歩行フレームワークではモーションを用いてサーボモータの角度指定でロボットの制御を行う方式とＸＹＺ座標系を用いて足位置を指定し歩行させる方式を併用しています。どちらも、基本の姿勢からの変化量で動作を行なっていますので、基本の姿勢と管理している姿勢データに差異が発生すると正しい制御はできなくなります。モーション制御の場合、直立姿勢が基本位置で、座標を用いた歩行制御の場合、基本姿勢が歩行準備姿勢です。例えば、モーション制御から座標を用いた歩行制御に移る場合、直立姿勢から歩行準備姿勢を経由して歩行制御を行う必要があります。これは正確な直立姿勢は座標の設定値では設定が難しく、逆に、正確な歩行準備姿勢は角度で表すモーションデータでは設定が難しいからです。





直立姿勢　　歩行準備姿勢

写真 2-1.ロボットの基本姿勢

2-2.二足歩行フレームワークの機能

二足歩行フレームワークとして開発した本ソフトウェアでは以下の機能を用意しました。「ターゲットの位置設定機能」では座標またはサーボモータの角度と時間を指定し、指定時間後に変化テーブルに従った移動補正により、サーボモータの角度変化を行います。このために２つの関数を用意しました。この２つの関数をうまく組み合わせて一定時間単位に実行することにより、ロボットにいろいろな動作をさせることができます（このプログラムをシナリオと呼びます)。一定時間ごとの制御を取り仕切る機能として「シナリオ実行機能」を用意しました。「はじめロボット９号」や「eMasterGEAR」の既存のモーションデータを新規のターゲット位置の設定を用いて行うために、「モーション変換機能」を用意しました。

・シナリオ実行機能

「はじめロボット９号」や「eMasterGEAR」のコントローラはサーボモータに一定周期（１６ｍｓや２０ｍｓ）で動作設定を行います。しかし、ロボットの制御を行う場合、このような細かい単位で制御を行ったのではプログラム規模ばかり大きくなり複雑になり、歩く等の大きな単位での動作の制御が難しくなります。そこで、ターゲット位置の設定機能を用いて、アクチュエータの最終設定が変わるタイミング（０．３秒から１秒位のタイミング）で歩く動作を分割してロボットの制御を行っていきます。ここでは分割した制御単位をアクションと呼びます。アクションは同じ動作が周期的に実行されたり、逆向きのアクションを行うと、反対向きの動作を行ったり等、実行順序を変えたり、次のアクションに渡すパラメータをかえることにより、共通で効率良く実行ができるようになります。これらのアクションの順番を入れ替えたり、パラメータを変えたりする機能を「シナリオ実行機能」と呼びます。シナリオはパーツ：シナリオ部にあります。シナリオ実行機能を実現しているプログラムがパーツ：基本動作部の robocontrol.c 内のいくつかの関数です。

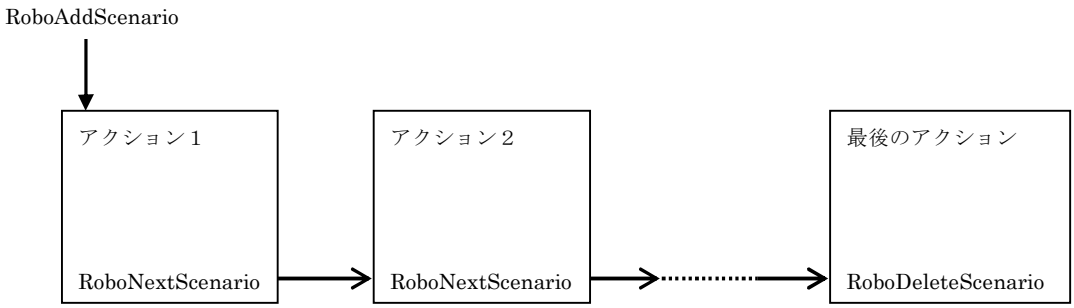


図 2-2.アクションの実行手順

以下にシナリオ設定で用いる４つの関数を一覧表にしました。RoboAddScenario と RoboCancelScenario 関数は、実行要求を行うプログラムから呼び出す関数であり、RoboNextScenario と RoboDeleteScenario 関数はアクション中で記述する関数です。RoboAddScenario 関数は最初に起動するアクションの関数を登録します。登録されたアクション関数に実行権が与えられると、アクション中に次に実行するアクション関数名を RoboNextScenario 関数で指定していきます。そして、次々にアクション関数をつないでいき、シナリオの最後のアクション関数にて RoboDeleteScenario 関数を呼び出すと、そのシナリオが終了し、次に登録されたシナリオを呼び出します。RoboCancelSecnario 関数は実行要求プログラムレベルで登録された全てのシナリオの実行をキャンセルする場合に呼び出す関数です。

これらのアクション関数の設定制御はパーツ：基本動作部の robodispatch.c に記述しています。また、個々の動作のシナリオはパーツ：シナリオ中に動作シナリオごとに記述しています。

型	関数名	引数型	引数名	引数の意味	関数の機能
int	RoboAddScenario	int	(*func)()	最初のアクション関数	シナリオの登録
		int	param1	実行パラメータ１	
		int	param2	実行パラメータ２	
		int	param3	実行パラメータ３	
		eScenarioType	type	シナリオタイプ	
int	RoboNextScenario	int	(*func)()	次のアクション関数	次のアクション関数の指定
		int	param1	次に渡すパラメータ	
		long	time	次のアクションまでの実行時間	
int	RoboDeleteScenario	なし	なし		シナリオの削除
void	RoboCancelScenario	なし	なし		シナリオの実行キャンセル

表 2-2.シナリオ実行機能用の関数

・ターゲット位置設定機能

アクション関数中でロボットに動作をさせるためには、どうすれば良いのでしょうか？動作をさせるためにモーション制御と歩行制御の２つの設定があります。モーション制御では各アクチュエータのターゲット角度と移動時間を指定することにより動作の制御を行います。歩行制御は左右の足に対する専用の制御で骨格の右と左の足の移動先ＸＹＺ座標と移動時間を用いて制御を行います。そのため、左右の足に関しては、どちらの制御を行うかの選択を行う必要があります。ターゲット位置設定機能として以下の５つの関数を

用意しました。RoboChangeMoveMode はモーション制御と歩行制御の切り替えを行います。RoboJointAction がモーション制御による角度設定関数です。ACTUATORID にて対象のアクチュエータを指定して最終角度と移動時間を指定することによりサーボモータを直接制御します。残りの３つの関数が歩行制御を行う関数です。歩行座標系は左右の足ごとにＸＹＺ位置の指定ができます。これに加えて C_PITCH、C_ROLL、C_FOOT の補正値の設定ができます。C_PITCH と C_ROLL は体の PITCH と ROLL の補正値で、C_FOOT は上げた足に対しての ROLL の補正値です。RoboWalkAction 関数が９つの位置と補正値に対する移動指定です。RoboWalkStop 関数は移動の停止を要求する関数で移動モードの変更が可能です。

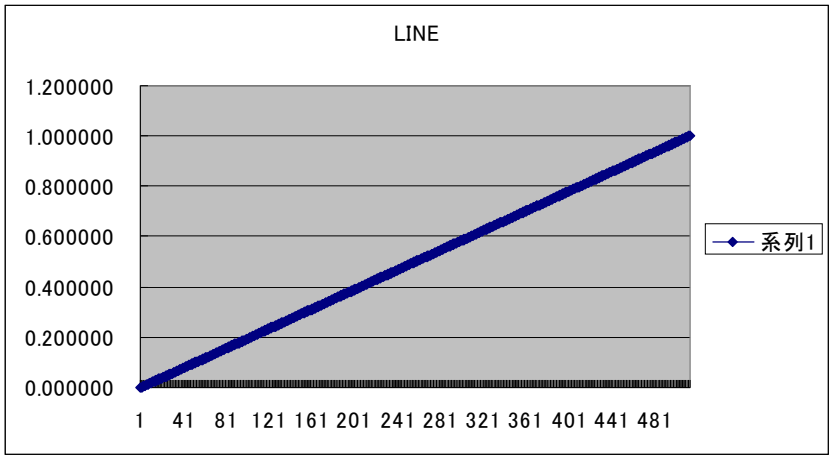
型	関数名	引数型	引数名	引数の意味	関数の機能
void	RoboChangeMoveMode	ACTMODE	mode	制御モード	モード切替
int	RoboJointAction	ACTUATORID	id	アクチュエータＩＤ	モーション制御：角度、移動時間指定
		long	time	移動時間(ms)	
		long	pos	角度(度数*1000)	
int	RoboWalkAction	WALKID	id	歩行座標ＩＤ	歩行制御：座標、移動時間、変化テーブル指定
		long	time	移動時間(ms)	
		long	pos	座標(mm*1000)	
		int	mode	移動モード	
int	RoboWalkStop	WALKID	id	歩行座標ＩＤ	現在位置で停止
		int	mode	設定移動モード	
int	RoboWalkReset	WALKID	id	歩行座標ＩＤ	指定位置にリセット
		long	pos	リセット位置(mm*1000)	

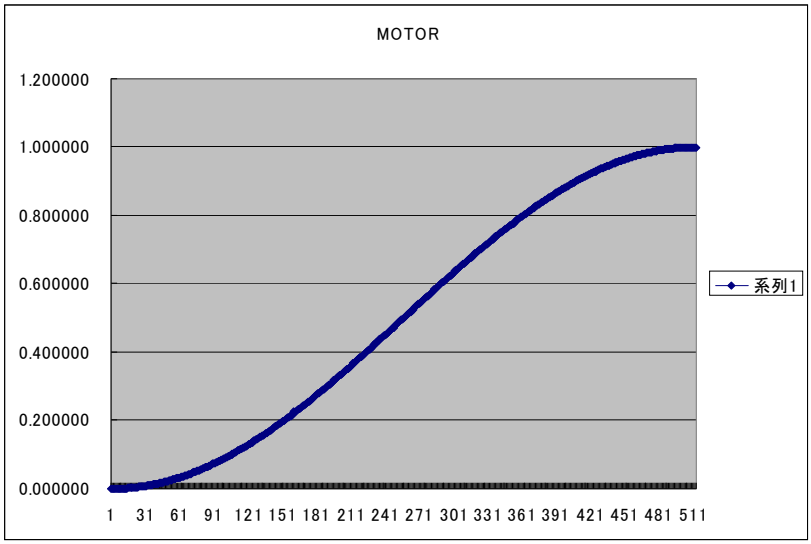
表 2-3.ターゲット位置設定機能の関数

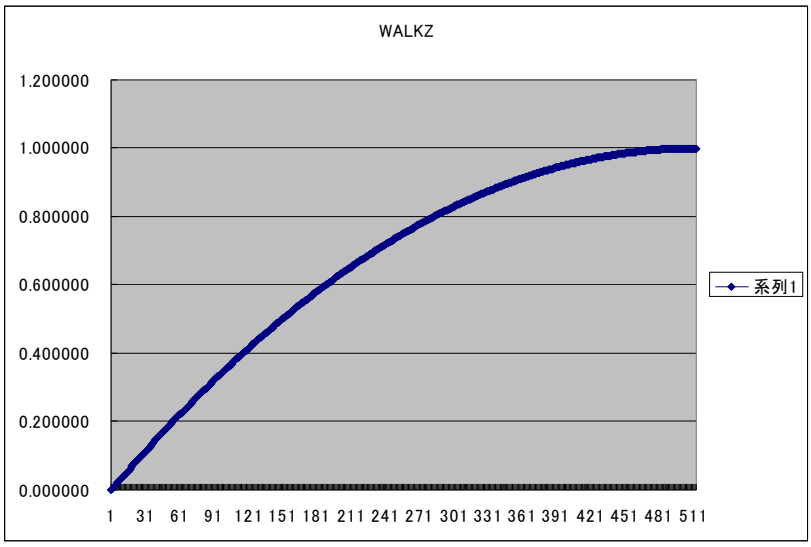
一般的に、サーボモータを動作させる場合、動き始めはゆっくりと、段々速くして停止位置に近づいたところで速度を落とすような制御を行います。上記の制御関数でも、速度制御が可能です。但し、モーション制御の RoboJointAction 関数では固定となります。歩行制御の RoboWalkAction 関数では３つ目の引数の mode により、７つの変化テーブルから選択ができます。７つの変化テーブルを以下に示します。７つの変化テーブルは以下の４つの基本曲線から速度を算出します。RoboWalkAction で以前の変化テーブルをそのまま使用したい場合は－１を指定してください。

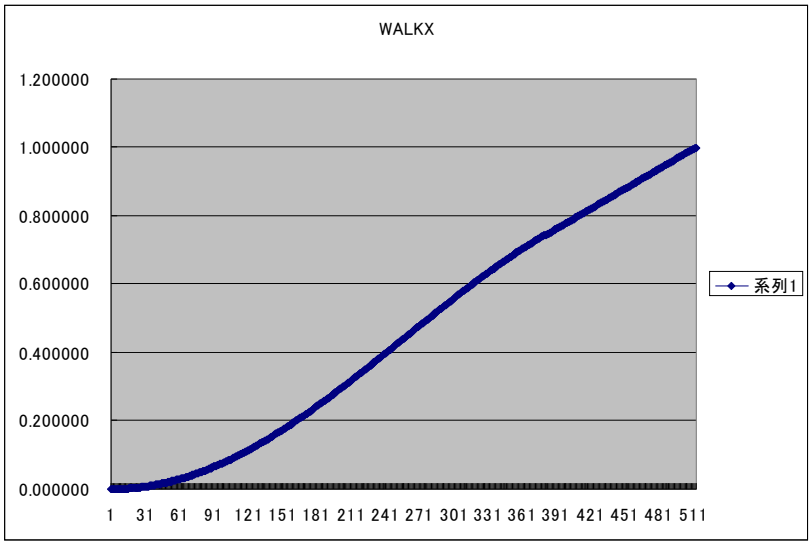
変化テーブルのラベル	適応	参考曲線	備考
MV_TBL_LAMP	比例移動	LINE 曲線を使用	
MV_TBL_SIN	角度移動	MOTOR 曲線を使用	モーション制御用
MV_TBL_Y	Ｙ座標移動	MOTOR 曲線を使用	
MV_TBL_Z_UP	Ｚ座標ＵＰ移動	WALKZ 曲線を使用	
MV_TBL_Z_DOWN	Ｚ座標ＤＯＷＮ移動	WALKZ 曲線を左右反転して使用	
MV_TBL_X_START	Ｘ座標動き始め	WALKX 曲線を使用	
MV_TBL_X_END	Ｘ座標停止前	WALKX 曲線を左右反転して使用	

表 2-4.実行曲線









実際の設定に従ったサーボモータの制御は、先に記述を行った一定周期（２０ＭＳまたは１６ＭＳ）のサーボ制御単位に移動角度を計算してサーボモータに対して設定を行います。

・モーションデータ変換機能

外部のモーション作成ツールのデータを用いてロボットのモーション制御を行うためにモーションデータ変換機能を用意します。しかし、現状対象となるモーションデータが「はじめロボット９号」のものしかなかったため、このデータを解析してモーション制御を行う仕組みを用意しました。この仕組みを用いて「eMasterGEAR」のサンプルプログラムにあるデータにて「eMasterGEAR」のモーション制御したところ、サーボモータの設定に「はじめロボット９号」には設定できない角度設定があり、補正値を入れることで

この対応を行いました。

簡単に「はじめロボット９号」用のモーションデータについて説明を行います。「はじめロボット９号」の場合、４００バイトのテンポラリィなモーション保存バッファをもつ１つのモーションデータは２５バイトで構成されるため、この保存バッファでは１６モーションの実行が可能となります。このバッファはページ番号を付けてＥＥＰＲＯＭに保存ができ、保存されたモーションデータは'M'コマンドにて再生が可能です。再生は指定したページ番号から順番に行われ、モーションデータ中に停止情報が入ったモーションまで連続して再生が可能となります。以下に２５バイトのモーション保存バッファの構成を示します。

オフセット	データ型	対応部位
0	signed char	移動時間、100ms 時間単位の時間-128。特定の値の場合コマンドとして扱う
1	signed char	LEFT_FOOT_ROLL:左足首のロール角度＋補正値
2	signed char	LEFT_FOOT_PITCH:左足首のピッチ角度＋補正値
3	signed char	LEFT_NEE_PITCH:左膝のピッチ角度＋補正値
4	signed char	LEFT_LEG_PITCH:左足のピッチ角度＋補正値
5	signed char	LEFT_LEG_ROLL:左足のロール角度＋補正値
6	signed char	LEFT_LEG_YAW:左足のヨー角度＋補正値
7	signed char	RIGHT_FOOT_ROLL:右足首ロール角度＋補正値
8	signed char	RIGHT_FOOT_PITCH:右足首のピッチ角度＋補正値
9	signed char	RIGHT_NEE_PITCH:右膝のピッチ角度＋補正値
10	signed char	RIGHT_LEG_PITCH:右足のピッチ角度＋補正値
11	signed char	RIGHT_LEG_ROLL:右足のロール角度＋補正値
12	signed char	RIGHT_LEG_YAW:右足のヨー角度＋補正値
13	signed char	WAIST_YAW:腰のヨー角度＋補正値
14	signed char	LEFT_ARM_PITCH:左腕ピッチ角度＋補正値
15	signed char	RIGHT_ARM_PITCH:右腕ピッチ角度＋補正値
16	signed char	LEFT_ARM_ROLL:左腕ロール角度＋補正値
17	signed char	LEFT_ELBOW_YAW:左肘ヨー角度＋補正値
18	signed char	LEFT_ELBOW_PITCH:左肘ピッチ角度＋補正値
19	signed char	RIGHT_ARM_ROLL:右腕ロール角度＋補正値
20	signed char	RIGHT_ELBOW_YAW:右肘ヨー角度＋補正値
21	signed char	RIGHT_ELBOW_PITCH:右肘ピッチ角度＋補正値
22	signed char	HEAD_YAW:頭ヨー角度＋補正値
23	signed char	未定義
24	signed char	未定義

表 2-5.モーションデータの構成

移動時間に関しては、１２８を引いた値を保存し、参照する場合１２８を足すことにより８ビットの正の値として操作しています。移動時間が２５１以上（２５秒を超える）の場合、コマンドして扱う。２５１の場合は動作の停止指定、２５４以上の場合は次のページ番号のモーションを継続して実行するコマンドになります。各サーボモータに対する設定角度は補正値を引いて保存されます。signed char では１２７から－１２８までの２５６度の角度しか設定できませんので、補正値にて設定角度をずらし適切な角度設定を行おうというものです。しかし、「eMasterGEAR」では「はじめロボット９号」の補正値では設定できない設定角度が発生したため補正テーブルを切り替えて使用しています。

番号	はじめロボット９号の補正値	eMasterGEAR の補正値
1	0	0
2	0	0
3	-127	-64
4	64	0
5	0	0
6	0	0
7	0	0
8	0	0
9	-127	-64
10	64	0
11	0	0
12	0	0
13	0	0
14	0	16
15	0	16
16	-64	-64
17	0	0
18	128	128
19	64	64
20	0	0
21	128	128
22	0	0
23	×	×
24	×	×

表 2-6.ロボットごとの補正値

テンポラリィなモーション保存バッファに登録するコマンド中のモーションデータは上記のような補正値はありません。しかし、「はじめロボット９号」にあらかじめ登録されているデータをそのまま再生するために、この機能を追加しました。

2-3.二足歩行プログラムの高速化

二足歩行プログラムは時間、長さ、角度等のデータの管理、演算を行います。これらの計算を浮動小数点で行うとフローティング演算プロセッサを持たないＣＰＵでは整数演算を使用したライブラリィにて演算を行いますので、演算時間負荷が高くなりサーボモータへの設定時間が長くなりデッドラインミスが発生する割合が高くなります。本プログラムでは演算時間の短縮のために時間、長さ、角度を以下の単位に変更しできる限り整数演算処理で計算できるように設計しました。

データ	尺度
時間	MSEC 秒を基本とする
長さ	1mm×1000 を基本単位とする
角度	1 度×1000 を基本単位とする

表 2-7.プログラムで扱う単位

2-4. TOPPERS/JSP の対応

TOPPERS/JSP 上の実装に当たっては、フレームワーム自体をライブラリィ化しタスクとして取り込む方式をとりました。そのため、非ＲＴＯＳ版では main 関数となっているプログラムを walk_task というタスクに置き換えることでＲＴＯＳ化を行っています。従って、TOPPERS/JSP に限らず、どのようなＯＳでも比較的簡単にポーティングが可能です。walk_task は、「はじめロボット９号」では１６ＭＳ周期に、「eMasterGEAR」では２０ＭＳ周期に起動する周期タスクです。このタスクは最上位の優先度で実行します。従って、その他のタスクはこのタスクの実行していない状態で実行可能です。walk_task にて実行するロボット制御用のプログラムは、ロボットが動作中でない場合、アクションプログラムはスキップしサーボモータの設定のみ行うようにプログラミングしています。そのため、歩行のような重いアクション計算を行う場合以外は、かなりの空き時間できる設計としています。

３．データ構造

二足歩行フレームワークでは、データ構造に骨格、関節の定義を行い上位モジュールがサーボモータを意識せずに動作制御を行えるように配慮しました。便宜上、論理的な関節制御部を**アクチュエータ**と呼び、それに対応した物理的なサーボモータを**サーボモータ**と呼ぶことにします。具体的にはコントローラの初期化時に、骨格、関節データとサーボモータの対応データを作成し、以後はアクチュエータに対して制御を行うことにより、これに対応したサーボモータを間接的に操作するような設計としました。以下にTOPPERS/JSP 版「eMasterGEAR」の電源投入時のモニタ画面を示します。frame[0]から frame[4]までが骨格に当たる表示です。joints が骨格に対応した関節の数、actuator が骨格に対応したアクチュエータの数です。

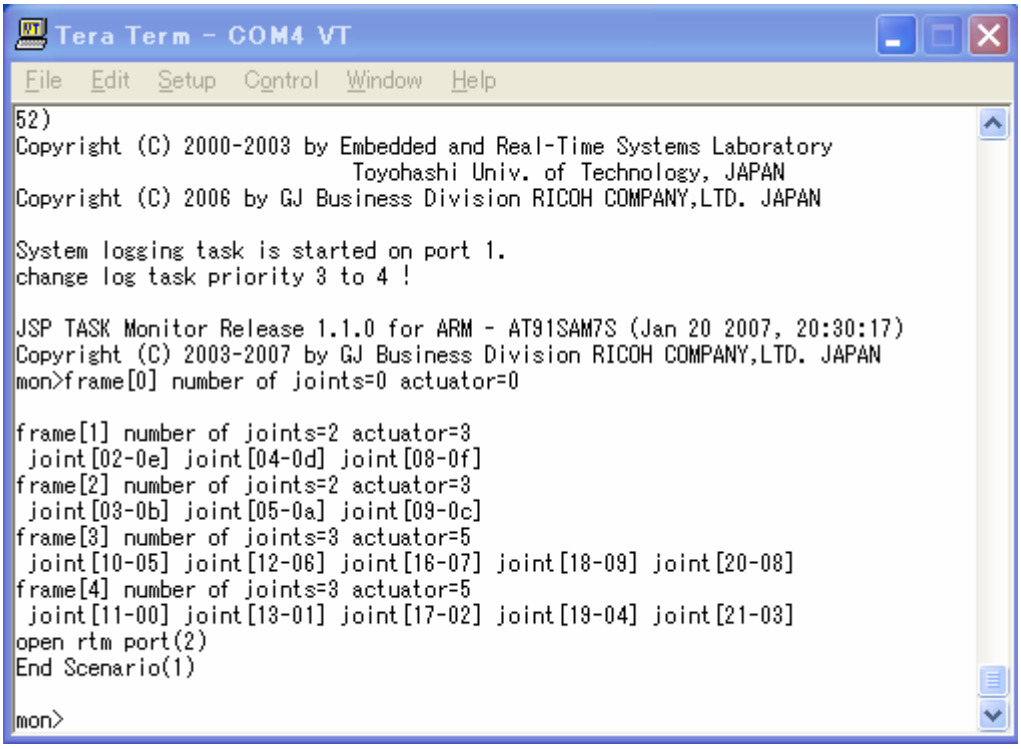


図 3-1.eMaterGEAR の実行ログ（TOPPERS/JSP 版）

骨格は以下の表になります。５つの骨格でロボット全体の構成を示します。

部位	ラベル	内容	はじめロボット９号 関節（アクチュエータ）	eMasterGEAR 関節（アクチュエータ）
frame[0]	BACKBONE	背骨	2(2)	0(0)
frame[1]	RIGHT_ARM	右手	2(4)	2(3)
frame[2]	LEFT_ARM	左手	2(4)	2(3)
frame[3]	RIGHT_LEG	右足	3(6)	3(5)
frame[4]	LEFT_LEG	左足	3(6)	3(5)

表 3-1.骨格の構成

骨格は tROBOFRAME 構造体のグローバル配列 RoboFrames によって構成されます。配列の項目が各部位となります。各部位にはそれに属するアクチュエータの構造体（tROBOACTUATOR）がポインタのチェーンとしてつながれ、部位毎の操作を行いたい場合は、このチェーンを用いてアクチュエータを選択的に操作が可能です。

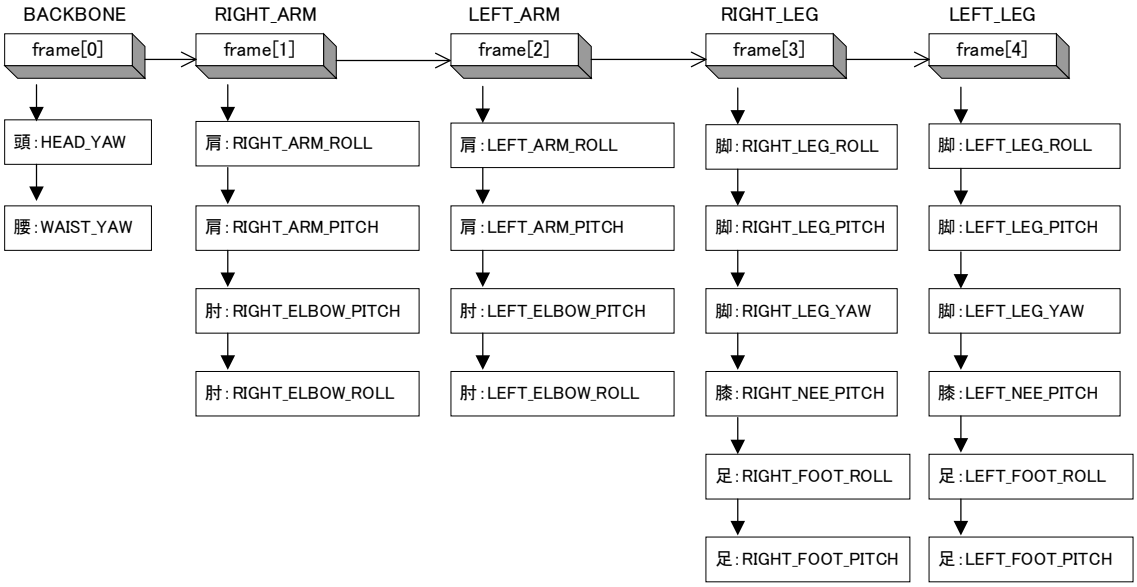


図 3-2.骨格とアクチュエータの接続図（はじめロボット）

歩行や関節の操作を行う基本がアクチュエータの構造体（tROBOACTUATOR）です。この構造体はアクチュエータの定義とサーボモータの状態を保存する構造体（tROBOELEMENT）から構成されます。サーボモータの状態はモーションにより関節角度で制御する場合は時間と角度で設定され、歩行座標により制御を行う場合は時間と長さで設定されます。それぞれ値は整数化された数値で演算します。モーションにおいても、歩行においても、サーボの設定時間（１６ｍｓや２０ｍｓ）毎に、細かく移動位置を設定してわけにはいかないので、時間（例えば５００ｍｓ後）を指定してその時間にサーボの角度や座標が現在の値からどのように変化するかは変化パターンを与えて、後は自動制御という形態とします。そのための変数は全て tROBOELEMENT 構造体に収められています。以下に tROBOELEMENT 構造体の構成を示します。この表の位置や差は角度（度数×1000）の場合と座標（ｍｍ×1000）の場合があります

オフセット	ラベル	型	内容	備考
0	select	long(整数)	変化テーブル番号	下１ビットは方向
4	time	long(時間:ms)	現在時間	
8	max	long(時間:ms)	終了時間	

12	dt	long(時間:ms)	変化時間	
16	start	long(位置)	最初の位置	度数×1000、mm×1000
20	amp	long(差)	位置の差	
24	current	long(位置)	現在位置	
28	target	long(位置)	目標位置	

表 3-2.tROBOELEMENT 構造体

アクチュエータ構造体（tROBOACTUATOR）に含まれるエレメントは度数の設定のみで、歩行用のエレメントは別の構造体(tROBOWALK)で定義します。しかし、歩行計算を行った後、計算後のサーボの角度はアクチュエータ構造体の deg に設定され、まとめてサーボモータへの設定を行います。mflag は実行モードを指定します。ビットの対応で以下の状態を示します。

・ビット７：アクチュエータ位置変化中
・ビット１：オンで歩行とモーション併用のアクチュエータ
・ビット０：オンで座標系による歩行モード
　　オフならばモーション制御

オフセット	ラベル	型	内容	備考
0	*next	*tROBOACTUATOR	次の構造体	骨格のチェーン用
4	id	unsigned char	アクチュエータＩＤ	
5	frame	unsigned char	所属のフレームＩＤ	
6	type	unsigned char	関節の型	
7	servo	signed char	対応のサーボ番号	負の値で未実装
8	mflag	unsigned char	実行フラグ	
9	dummy	unsigned char	未使用	
10	bias[3]	short	モーションのバイアス	３パターン可能
16	deg	long	サーボ角度×1000	
20	element	tROBOELEMENT	変化要素	

表 3-3.tROBOACTUATOR 構造体

3-2.ロボット制御のコマンド

ロボットの制御コマンドはＲＴＯＳ版と非ＲＴＯＳ版では、設定方法が異なります。非ＲＴＯＳ版ではできるだけ「はじめロボット９号」のオリジナルコマンドをベースに機能拡張、または、不要なコマンドの削除を行いました。ＲＴＯＳ版ではユーザインターフェイスをタスクモニタベースに実装しましたので、タスクモニタの拡張コマンドであるＰＩＰＥ（Ｐ）コマンドからコマンド発行する必要があります。しかし、非ＲＴＯＳ版コマンドの前に'P',‘を付加してコマンドを送れば、ＲＴＯＳ版でも非ＲＴＯＳ版コマンドと同等のコマンド設定が可能です。

*	サイズ	コマンド	データ	終了子
アスタリスク	２バイト 16 進数	２バイト 16 進数	バイトデータ	¥0,'#','¥n'

表 3-4.コマンドの基本フォーマット

説明：コマンドの最初はアスタリスクキャラクタから始まり、終了は終了子で終わる。終了子は'¥0'、'#'、'¥n'のいずれでもかまわない。コマンドから終了子までのバイトサイズをサイズで指定する。サイズは２バイトの１６進数の文字。コマンドも２バイトの１６進数文字、本仕様では、０ｘ３０から０ｘ５Ｆまでの範囲のコマンドしか扱いません。

データ中のキャラクタは以下の約束に従う数値として扱います。

'0'：ゼロ

'1','2'～'9'： 1 から 9 までの数値

'A','B'～'Z'：1 から 26 までの数値

'a','b'～'z'：-1 から-26 までの数値

例として

"FF00"は 6*4096 + 6*256 + 0*16 + 0

"ff00"は -6*4096 + -6*256 + 0*16 + 0

・ロボット制御コマンド

通常のロボット制御コマンドは０ｘ３０となります。

ロボット制御コマンドは以下のフォーマットとなります。ＲＴＯＳ版のＰＩＰＥコマンドからの設定はデータのみ設定すれば、前の"*0730"と最後の終了子はＰＩＰＥコマンドで付加してロボット制御部に転送を行います。

*	07	30	データ（4 バイト）	終了子

表 3-5.ロボット制御コマンドのフォーマット

データの設定を以下に示します。

データ１	データ２	データ３	データ４	機能	備考
F	歩数	左右	速度	前進歩行	歩数０で停止なし 左右はバランス角度２度から３度が最適 速度は１が20ms単位で正負の設定あり
B	歩数	左右	速度	後退歩行	同上
S	歩数	左右	０	ステップ	左右は移動のmmであり、1=2mm で最大 20mm
D	歩数	左右	０	横歩き	左右は正と負の値で設定
M	頁番号 100の位	頁番号 10の位	頁番号 １の位	モーション実行	
N	頁番号 100の位	頁番号 10の位	頁番号 １の位	固定値モーション実行	１が仰向けからの起き上がり、２がうつ伏せからの起き上がり
C	機能	０	０	キャンセル	機能：０がシナリオキャンセル、１が歩行停止、２が歩行バランスの修正
T	時間	０	０	直立	時間は秒単位
G	回数	時間	オプション	お辞儀	オプションはeMasterGEAR のみ音楽演奏が可能

表 3-6.ロボット制御コマンドの詳細

・モーション設定コマンド

モーション設定コマンドは、データの転送、確認実行、ＥＥＰＲＯＭへの書き込みなどの機能があります。ＥＥＰＲＯＭのない「eMasterGEAR」では書き込みは行えません。この機能はコマンドごとに機能が対応しています。ＲＴＯＳ版ではＰＩＰＥコマンドの後の手入力となります。手入力の場合、最初のアスタリスクとサイズはＰＩＰＥコマンドで発行し自動的に追加しますので省略できます。

１）データ転送

モーションデータをテンポラリィなモーション保存バッファに転送します。

*	63	40	データ(96 バイト)	終了子

データ	設定フォーマット	内容	単位
データ 1～4	４バイト１６進数	モーション番号（０～１５）	
データ 5～8	４バイト１６進数	移動時間	100MS
データ 9～12	４バイト１６進数	LEFT_FOOT_ROLL:左足首のロール角度	度数
データ 13～16	４バイト１６進数	LEFT_FOOT_PITCH:左足首のピッチ角度	度数
データ 17～20	４バイト１６進数	LEFT_NEE_PITCH:左膝のピッチ角度	度数
データ 21～24	４バイト１６進数	LEFT_LEG_PITCH:左足のピッチ角度	度数
データ 25～28	４バイト１６進数	LEFT_LEG_ROLL:左足のロール角度	度数
データ 29～32	４バイト１６進数	LEFT_LEG_YAW:左足のヨー角度	度数
データ 33～36	４バイト１６進数	RIGHT_FOOT_ROLL:右足首ロール角度	度数
データ 37～40	４バイト１６進数	RIGHT_FOOT_PITCH:右足首のピッチ角度	度数
データ 41～44	４バイト１６進数	RIGHT_NEE_PITCH:右膝のピッチ角度	度数
データ 45～48	４バイト１６進数	RIGHT_LEG_PITCH:右足のピッチ角度	度数
データ 49～52	４バイト１６進数	RIGHT_LEG_ROLL:右足のロール角度	度数
データ 53～56	４バイト１６進数	RIGHT_LEG_YAW:右足のヨー角度	度数
データ 57～60	４バイト１６進数	WAIST_YAW:腰のヨー角度	度数
データ 61～64	４バイト１６進数	LEFT_ARM_PITCH:左腕ピッチ角度	度数
データ 65～68	４バイト１６進数	RIGHT_ARM_PITCH:右腕ピッチ角度	度数
データ 69～72	４バイト１６進数	LEFT_ARM_ROLL:左腕ロール角度	度数
データ 73～76	４バイト１６進数	LEFT_ELBOW_YAW:左肘ヨー角度	度数
データ 77～80	４バイト１６進数	LEFT_ELBOW_PITCH:左肘ピッチ角度	度数
データ 81～84	４バイト１６進数	RIGHT_ARM_ROLL:右腕ロール角度	度数
データ 85～88	４バイト１６進数	RIGHT_ELBOW_YAW:右肘ヨー角度	度数
データ 89～92	４バイト１６進数	RIGHT_ELBOW_PITCH:右肘ピッチ角度	度数
データ 93～96	４バイト１６進数	HEAD_YAW:頭ヨー角度	度数

表 3-7.データ転送コマンド

２）テンポラリィモーションの実行

テンポラリィなモーション保存バッファ内の指定モーションを実行し、実行後停止状態とする。

*	07	41	モーション番号(4 バイト)	終了子

表 3-8.テンポラリィモーション実行コマンド

３）モーション実行

テンポラリィなモーション保存バッファ内の指定モーション以下のモーションを順次実行する。

*	07	42	モーション番号(4 バイト)	終了子

表 3-9.モーション実行コマンド

４）テンポラリィモーションの保存

テンポラリィなモーション保存バッファの内容を、ＥＥＰＲＯM上のページに保存する。（「はじめロボット９号」のみ有効）

*	07	43	ページ番号（１以上）	終了子

表 3-10.テンポラリィモーションの保存のコマンド

・メンテナンス用コマンド

ロボットのホームポジションが正しく設定されていないと歩行中に転倒したり、直線の前進でも、右や左に曲がって歩行します。サーボモータは初期設定位置にばらつきがあり、各々のロボットで微小にホームポジションにばらつきがあります。ロボットに直立を行わせ左右の足や手にずれがないかを確認してください。メンテナンスコマンドはホームポジションの調整を行うためのコマンドです。ポジション情報は ROBOSERV 構造体上に用意しています。以下は ROBOSERV 構造体の項目です。ホームポジションの設定値はゼロとなるように補正が行っています。deg_offset が補正値ですが。この項目を修正することにより、ホームポジションの補正が可能です。

オフセット	ラベル	型	内容	備考
0	deg_sign	short(整数)	プラスマイナス補正	
2	deg_in	short(番号)	対応アクチュエータ番号	ACTATORID
4	deg_offset	long(角度*1000)	角度へのオフセット値	
8	deg_lim	long(角度*1000)	最大角度	
12	deg2pls	long(補正値)	サーボに与える角度の整数化値	はじめロボットの設定方式をそのまま使用

表 3-11.ROBOSERV 構造体

ROBOSERV 構造体は RoboServ の変数名でサーボモータの数だけ用意されています。

この変数の設定はロボットの起動時に motor_get_data というロボット固有の関数により読み出された設定値により設定されます。「はじめロボット９号」の場合、この関数はＥＥＰＲＯＭから初期設定値を読み出しますが。「eMasterGEAR」はＥＥＰＲＯＭがないため storage.c 中のコンスタントテーブルから設定を行っています。TOPPERS プロジェクトには３台の「eMasterGEAR」がありますが、ROBONO という定数をコンパイルスイッチに入れ機種ごとの設定をコンパイル時に指定するようにしています。

RoboServ の初期値（特に deg_offset）を補正するためのツールがメンテナンスコマンドです。ＲＴＯＳ版ではＰＩＰＥコマンドの後の手入力となります。手入力の場合、最初のアスタリスクとサイズはＰＩＰＥコマンドで発行し自動的に追加しますので省略できます。

１）現状のポジションデータをＥＥＰＲＯＭに書き込む

RoboServ の内容を保存用の形式に変換して、ＥＥＰＲＯＭに保存する。(「はじめロボット９号」のみ有効)

*	03	50	終了子

表 3-12.ポジションデータの書き込みコマンド

２）現在のポジションデータの表示

個々の RoboServ 変数の内容を表示します。

*	03	51	終了子

表 3-13.現在のポジションデータの表示コマンド

３）RoboServ 変数の内容変更

RoboServ 変数の内容を変更します。サーボ設定は定期的に行われていますので、設定データによりロボットの設定位置は微小変化しますので、位置確認が行えます。

*	07	52	補正データ(4 バイト)	終了子

データ 1,2	データ３	データ４	機能	備考
サーボ番号	0	0	deg_offset の変更：+1000	
		1	deg_offset の変更：-1000	
	1	0	deg_sign の変更：-1 に	
		1	deg_sign の変更：+1 に	

表 3-14.ポジションデータ変更コマンド

４）ＥＥＰＲＯＭデータの直接表示

ＥＥＰＲＯＭの内容を 256 バイト（ＥＥＰＲＯＭの単位の１ページ）単位でＤＵＭＰす

る。

*	07	54	EEPROM のページ番号	終了子

表 3-15.ＥＥＰＲＯＭデータのＤＵＭＰコマンド

3-3.タスクモニタによる操作

フレームワークを直接操作する場合は、前節の制御コマンドにより操作を行います。しかし、ＲＴＯＳ版では、教育WGで使ってきたタスクモニタをフロントエンドとして、ロボットの制御を行いますので、より簡単なコマンドで操作が行えます。

TOPPERS/JSP に搭載したタスクモニタの内容については、添付ソースの jsp-1.4.2/monitor 中の README.txt を参照ください。また、タスクモニタの操作についてはモニタプロンプトが出た状態で以下のコマンドを入力すれば、ヘルプメニューが表示されます。

mon>help(return)

また、ロボットの操作はタスクモニタ用コマンドの中の PIPE コマンドを用いて操作が行えます。PIPE コマンドも以下のように HELP メニュー表示機能があり、プロンプト表示状態で、mon>pipe help(return) と入力すてば、ロボット操作用コマンドメニューを表示します。コマンドメニューに従って、PIPE に続けてコマンドを設定すればロボットの操作が行いえます。

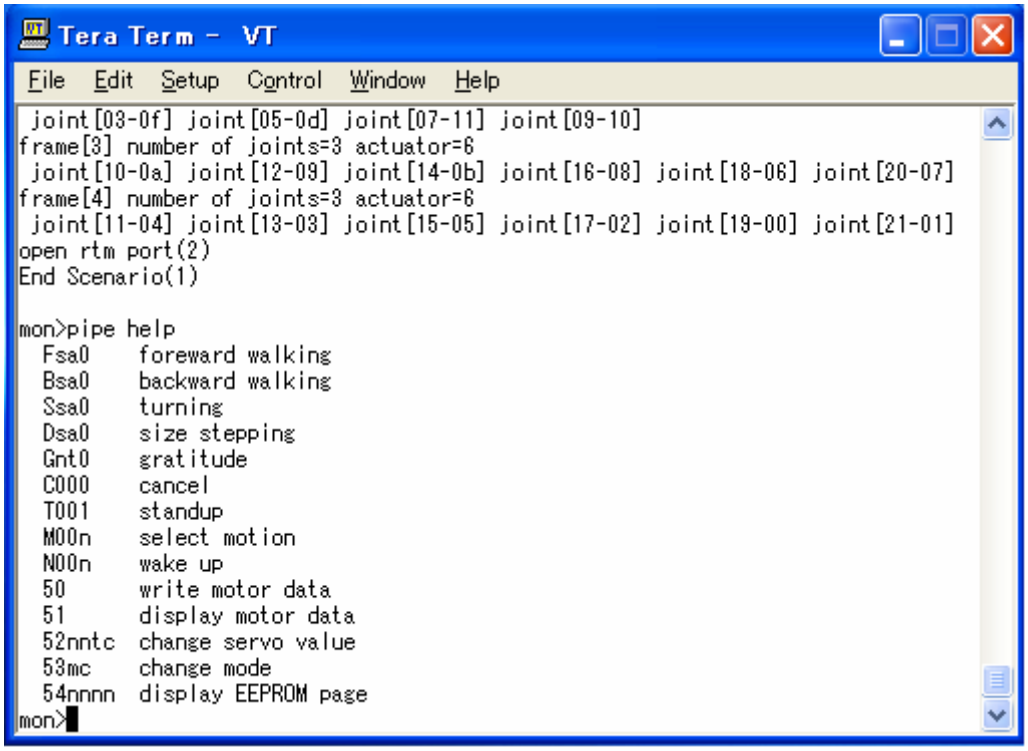


図 3-3.「はじめロボット９号」の PIPE コマンド

PIPE コマンドは、前節で説明したコマンドのデータから付随のフォーマットを自動的

に追加してフレームワークに設定を行う機能を持ち、簡単なコマンドで操作が可能です。例えば、*0730 で始まるコマンドについては、最初のアスタリスク、サイズ、コマンド（３０）、終了子を自動的に追加するため、データ部の４文字のみの設定でロボットの設定を行えます。その他のコマンドについては、最初のアスタリスク、サイズと終了子を自動的に追加します。

基本歩行動作の６足前進については、基本フォーマットでは

*0730FF00#

となりますが、PIPE コマンドでは

mon>pipe FF00(return)

で操作が可能です。

「eMasterGEAR」では、タスクモニタはデバッグシリアルに接続しており、ZigBEE がシリアルポート２に接続されています。TOPPERS/JSP の実装では、デバッグシリアルはＰＯＲＴ１、シリアルポート２はＰＯＲＴ２に割り当てを行っています。デバッグシリアルはタスクモニタのコマンドで操作を行いますが。ZigBEE は初期設定ではタスクモニタに設定していませんので、前節のロボットの制御コマンドを用いて操作を行わなければなりません。

４.簡単なシナリオ制作

4-1. お辞儀シナリオの作成

シナリオ実行機能をもちいて、お辞儀を行うプログラムの説明を行います。お辞儀プログラムは robo/jap-1.4.2/robo/scenario/robogratitude.c に記載されています。このプログラムには以下の３つのアクションとなる関数が記載されています。

①ScenarioGratitudeActionStart

初期設定

②ScenarioGratitudeAction1

お辞儀の状態をモーションで指定、回数分行ったあとは RoboDeleteScenario で終了

③ScenarioGratitudeAction2

直立状態を指定、お辞儀回数を減算

アクションの中で、次の動作状態を指定した後、動作が終了した時間を指定して次のアクションが起動するように RoboNextScenario 関数で設定します。

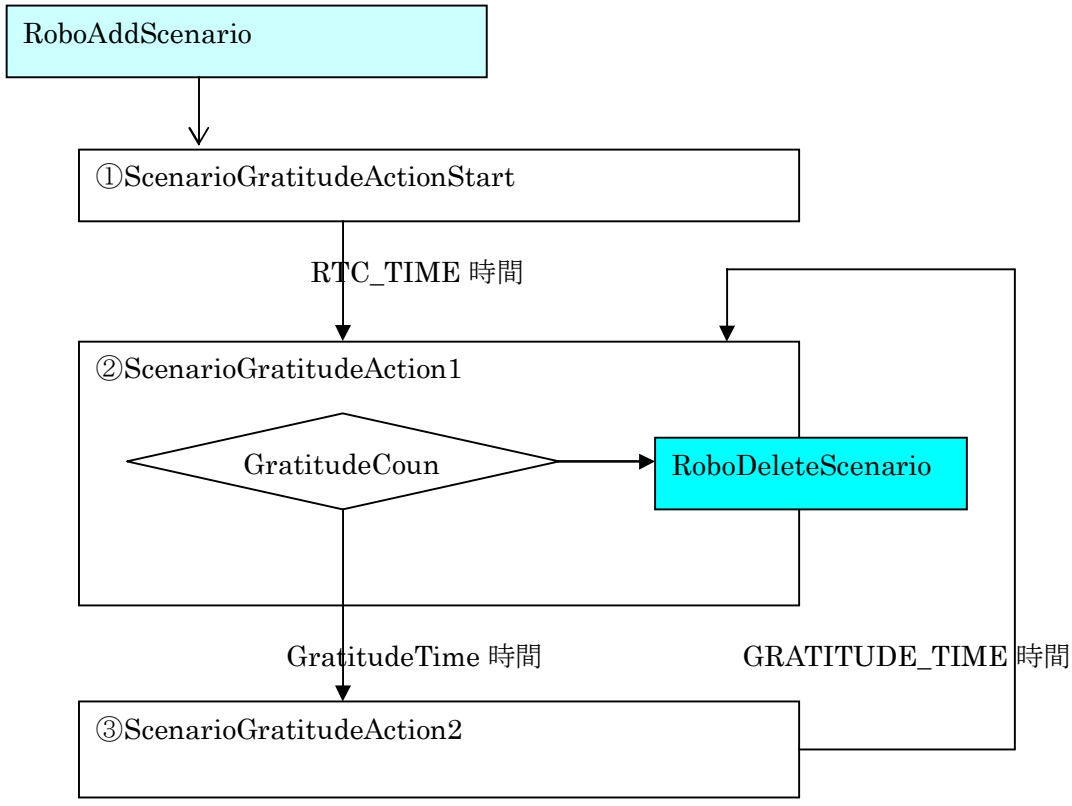


図 3-3.お辞儀シナリオのアクション実行

５.最後に

二足歩行ロボットプログラムを作成するために、去年の２月から１１月までの休みの日に開発を行いました。かなり大変な作業であり、すっかり、心のバランスを失ったのか、正月はＰＳ２のゲームペルソナ３にはまってしまい、このドキュメントもなかなか手がつきませんでした。二足歩行のプログラムは、シナリオ機能やターゲット位置設定機能を用いていろいろな動作を設定していく作業となります。このとき、いろいろなロボットのいろいろな動作を良く見ることが大きな参考となりました。

開発時間が短く、このコンテンツでやり残したことが多々あります。ロボットの左右の方向転換が思ったような完成度となっていません。公開までに時間があるので、もう少し、改善して見るつもりです。また、せっかく２台のロボットに共通なプログラムを搭載したのに、できれば共通のモーションエデッタを開発したかったのですが、今後の課題とします。センサーを搭載した自動歩行もやりたかったのですが、もっとロボットの値段が下がったら３台目としてプログラムを搭載してチャレンジしたいと思います。

最後に、今回の制作に協力した頂いた方々に助力を頂きました。この場を借りてお礼を申し上げます。

（株）イーエスピー企画の江崎寛康さん

はじめロボット製作者の坂本元さん（メールでのみ、ご助力いただきました）

（株）ベストテクノロジの上光隆義さん

開発にあたり以下の書物を参考としました。

1) ヒューマノイドロボット

梶田秀司編著

発行 株式会社オーム社

2) ROBO-ONE のための二足歩行ロボット製作ガイド

ROBO-ONE 委員会編

発行 株式会社オーム社

3) 図解雑学 ロボット

新井健生監修

発行 ナツメ出版企画株式会社

4) わかりやすいロボットシステム入門-メカニズムから制御まで-

松日楽信人、大明準治著

発行 株式会社オーム社