ロボットオペレータ PRO 版　 Ver2.3

2017/05/1

あべののじいさん

○ ソフトの概要

本プログラムは二足歩行ロボット「ラピロ」をパソコンで制御することを目的に作られたものです。

直接モーション命令（#M コマンド）をロボットに送り動作させることと
各サーボモータに対して回転角を指定してポーズをとらせることです。
上記の操作を元に作成した自動実行ファイルを用いて連続動作をさせることです。
さらに、**自動実行ファイルを 1 ステップずつ進めたり戻したりすることも可能です。**

１；ラピロへ動作命令を直接送り実行できます
２：モーション命令を具体的な表現（前進・後退など）でリストより選んで実行しそれを繰り返せます
３：上記２で選択した順に動作時間間隔を指定して連続動作をファイル（拡張子.rpo）保存してそれを読み込むことで自動実行できます
４：連続していろいろなポーズをとらせそれを繰り返せます
ポーズを取り終わるまでの時間（動作の速さを決める）とポーズから次のポーズまでの時間を指定して順番にファイル（拡張子.rpo）保存して、それを読み込むことで設定通りの動作を自動実行することができます
５：上記の自動実行を 1 ステップずつ進めたり戻したりできます
６：自動実行を途中で終了させるには、強制終了をクリックしてしばらく待つと入力モードに戻ります

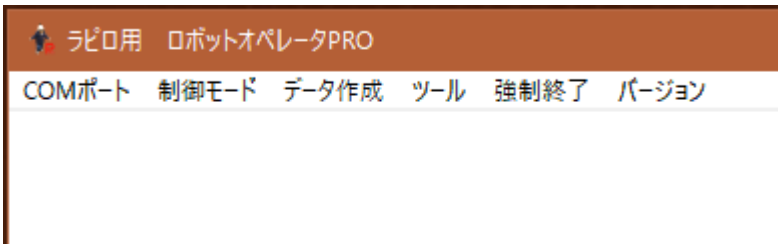
添付した ino スケッチで**ファームウエアーを書き換える**ことでモーション命令は２桁まで拡張されます。
PRO 版を用いることで、さらに以下のことができます。

７：増やしたモーション命令を付け加える為にモーション命令リスト（DirectCmd_List.txt）の編集ができます（ DirectCmd_List.txt は RobotOperatorPRO.exe と同じフォルダー内にあることが必要）
８：rpo, ino ファイルなどは**相互変換**ができます

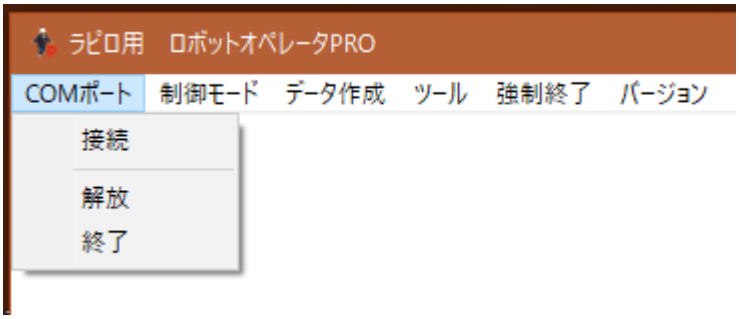
rpo, ino, DirectCmd_List.txt などは全てテキストエディタで内容を確認できますし、編集もできます。

ロボットオペレータ PRO Ver2　操作説明

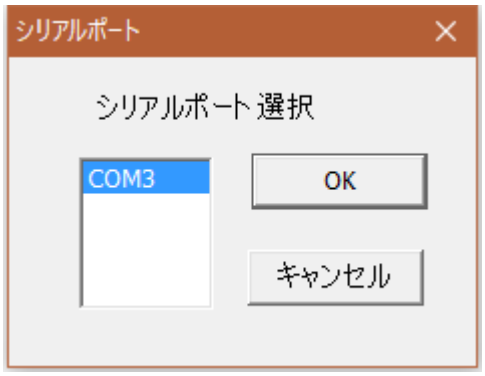
メニュー一覧



COM ポート



接続：PC にある COM ポートの一覧が表示され、一覧からロボットが接続されているシリアルポートをクリックして選びます。



OK ボタン：クリックすると、一旦消えたロボットの眼が再び青色に点灯して PC とロボットが接続されます。

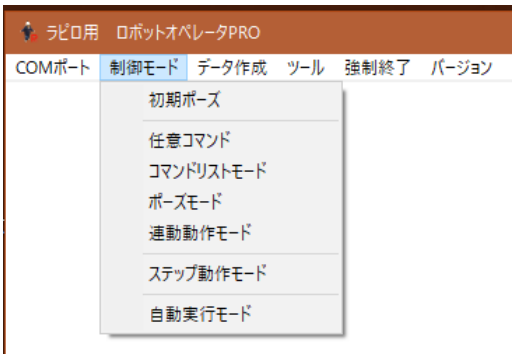
注：複数の COM ポートがある場合に、他の COM ポートと接続した場合にもロボットと接続したかのような動作をしますので注意してください。

解放：シリアルポートを開放して、PC とロボットの接続を解除します。

終了：ロボットオペレータを終了します。

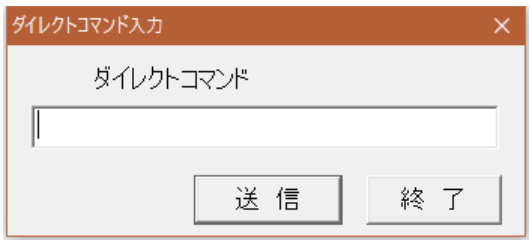
制御モード

シリアルポートで PC とロボットが接続されている必要があります



初期ポーズ：ロボットに初期状態（静止ポーズ）を取らせます

任意コマンド：キーボードからコマンドを入力し、コマンドを PC からロボットへ直接送ります。



送信ボタン：**クリック**するか Enter キーを押すことで入力したコマンドを PC からロボットへ直接送ります。

小文字での入力も可能です

終了ボタン：**任意コマンド**を終了します。

コマンドリストモード：リストからロボットにさせたい動作をマウスで選びクリックすると

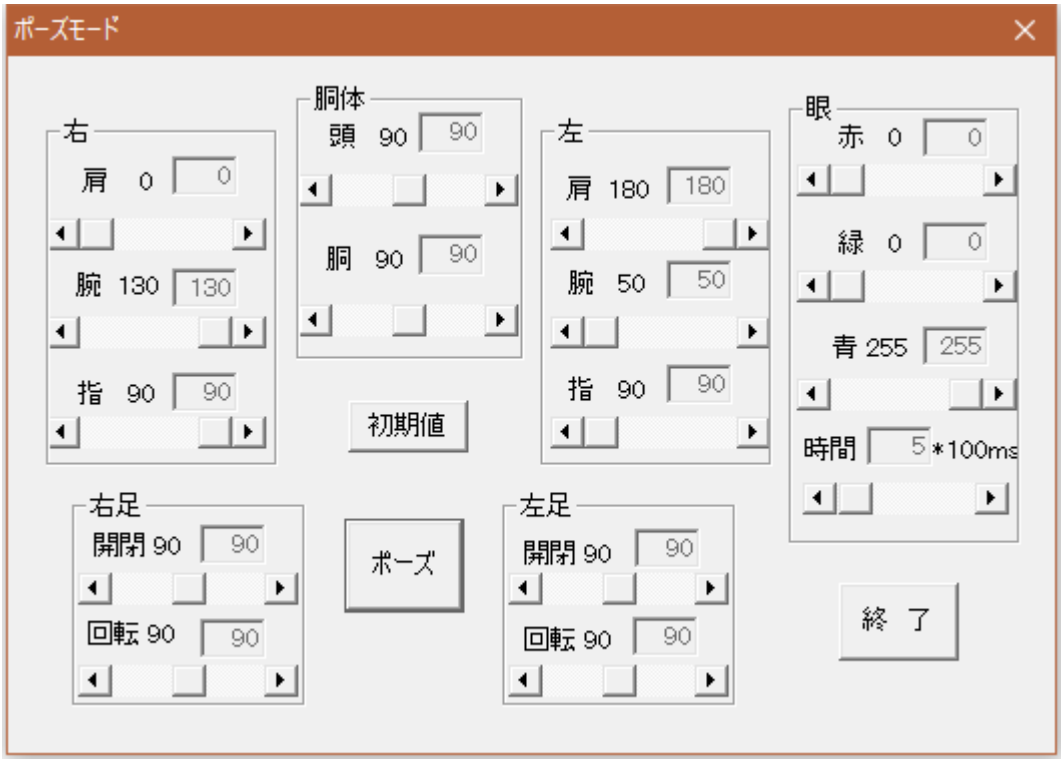
青色のバーが選択位置に移動します。

送信ボタン：カーソル位置のコマンド（**#M****）が PC からロボットへ送られます。

あるいは、左側リストの**項目をダブルクリック**しても PC からロボットへコマンド（**#M****）が送られます。

終了ボタン：**コマンドモード**を終了します。

ポーズモード：ロボットの各サーボモータに設定した回転角度と RGB による目の色を設定し、設定値に相当するポーズをとらせます。



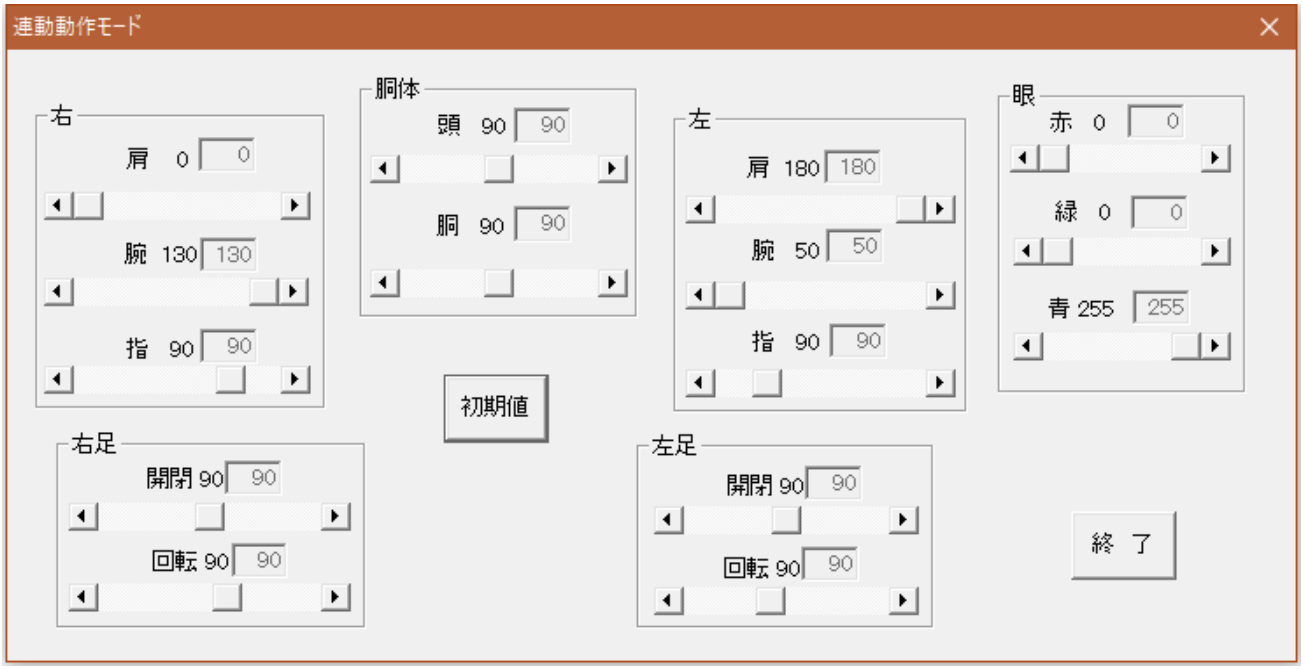
スライダー：スライダーを左右に移動させると対応するロボットの各部のサーボモータが回転し、回転角度がボックス内に表示されます。スライダーの三角印をクリックすると５度づつ増減します。

初期値ボタン：各選択値が静止ポーズの値になり、ロボットの姿勢も初期状態に戻ります。

ポーズボタン：**時間ボックス**に指定した時間をかけて、各設定値に相当するポーズを取ります。

終了ボタン：**ポーズモード**を終了します。

連続動作モード：スライダーを操作すると、操作に連動して各サーボモータが回転したり、目の色が変化してポーズをとります。

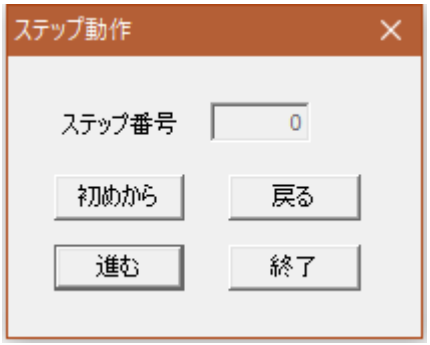


スライダー：スライダーを左右に移動させると対応するロボット各部のサーボモータが回転したり、目の色が変化します。この時、回転角度や LED の設定値がボックス内に表示されます。
　スライダーの三角印をクリックすると５度づつ増減します。

初期値ボタン：各選択値が静止ポーズの値になり、ロボットの姿勢も初期状態に戻りますます。

終了ボタン：**連動動作モード**を終了します。

ステップ動作モード：ステップ動作モードを選択するとファイル選択窓が開きます。

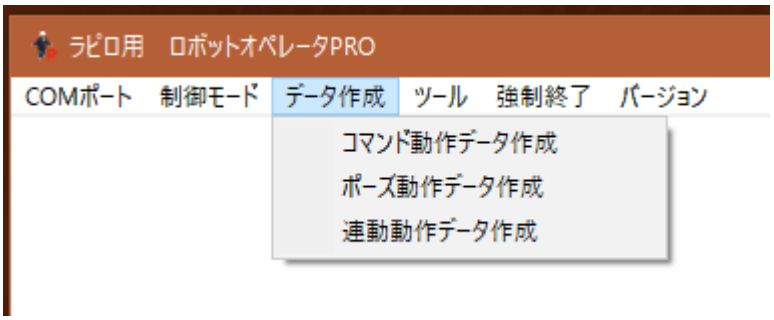


ステップ番号：自動実行ファイルの何フレーム目を実行しているかを表しています。
初めからボタン：フレーム ０に戻ります。
進むボタン：１フレーム動作を進めます。
戻るボタン：１フレーム動作を戻します
終了ボタン：ステップ動作モードを終了します。

オートモード：オートモードを選択するとファイル選択窓が開きます。
　データ作成で作成されたファイル(.rpo)を選択するとファイルに書かれた命令を指示通りに順次実行します。

データ作成

ロボットとの接続がなくてもデータの作成はできますが、ポーズをとらせて動きを確認することはできません。



コマンド動作データ作成：操作の仕方は制御のコマンドリストモードとほぼ同じですが、オートモードのファイル（.rpo）を作成するのが目的です。　**登録できるデータ数は最大 255 コマンドです。**

動作間隔：前のコマンドが送られた後、選択したコマンドが送信されるまでの時間を表します。**0.1 秒**を単位として設定します。

この時間が長すぎると、前の動作が複数回繰り返されます。

また、この値が小さすぎると、ロボットの動作が追い付かず不自然な動作になります。

送信ボタン：カーソルで選択したコマンドをロボットへ送信し動作を確認します。

あるいは、左側リストの項目を**ダブルクリック**すると、送信ボタンをクリックしなくても PC からロボットへコマンド（**#M****）が送られます。

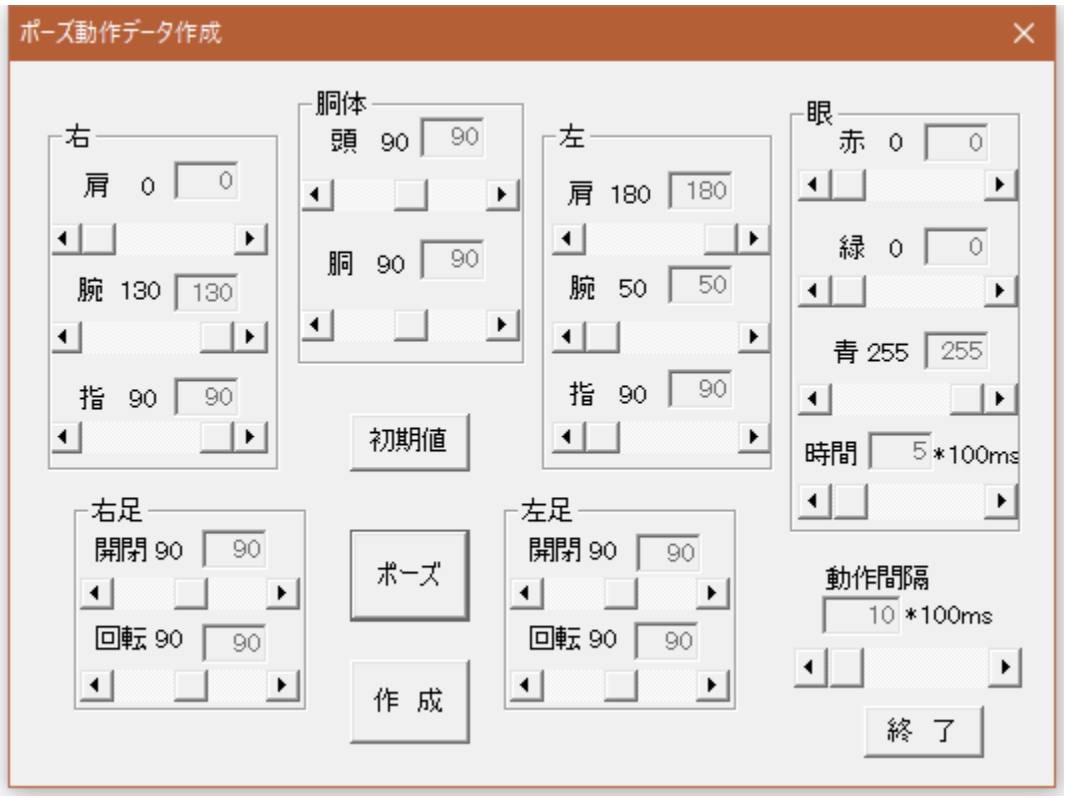
コマンドを送信しても**オートモードデータファイル**には付け足されません。

作成ボタン：選択したコマンドをオートモードデータファイルに作成します。**最大 255 コマンドまで。**

作成ボタンをクリックするたびに選択されているコマンドがファイルの後ろに付け足されます。

終了ボタン：データ作成を終わり、ファイルを保存するための窓が開きます。適当な名前を付けて保存してください。拡張子（.rpo）は自動的に付けられます。

ポーズ動作データ作成：操作の仕方は制御のポーズモードとほぼ同じです。ただ、オートモードのデータファイル（.rpo）を作成するのが目的です。　**データは最大 255 コマンドです。**



スライダー：スライダーを左右に移動させると対応するロボット各部のサーボモータが回転したり、目の色が変化します。この時、回転角度や LED の設定値がボックス内に表示されます。
　スライダーの三角印をクリックすると１度づつ増減します。

初期値ボタン：各選択値が静止ポーズの値になり、ロボットの姿勢も初期状態に戻りますます。

ポーズボタン：**時間ボックス**に指定した時間をかけて、各設定値に相当するポーズを取ります。

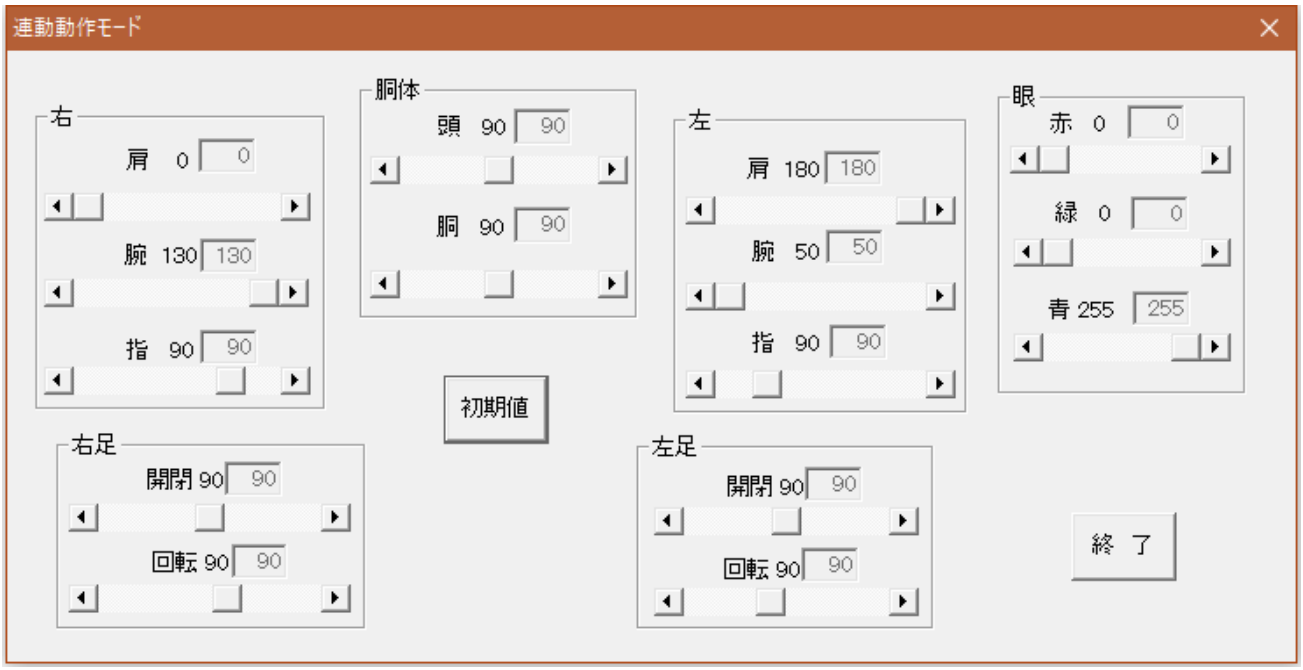
作成ボタン：スライダーで設定した値をオートモードデータファイルに作成します。**最大 255 フレームまで。作成**ボタンをクリックするたびに選択されているコマンドがファイルの後ろに付け足されます。

時間：前のポーズから設定したポーズに変化するのにかかる時間長さで、値が大きいほどゆっくりとした動きになります。1～255 で 0.1 秒を単位として設定します。**0 を設定すると何もしません。**

動作間隔：設定したポーズに変化した後、次のコマンドが送信されるまでの時間を表します。最大値 999 で **0.1 秒**を単位として設定します。

終了ボタン：データ作成を終わり、ファイルを保存するための窓が開きます。適当な名前を付けて保存してください。拡張子（.rpo）は自動的に付けられます。

連動動作データ作成:操作の仕方は連動動作モードとほぼ同じで、オートモードデータファイル(.rpo)を作成するのが目的です。PC とロボットが接続されている必要があります



スライダー：スライダーを左右に移動させると対応するロボット各部のサーボモータが回転したり、目の色が変化します。この時、回転角度や LED の設定値がボックス内に表示されます。

スライダーの三角印をクリックすると１度づつ増減します。

初期値ボタン：初期値ボタンをクリックすると各ボックス内の値が静止ポーズの値になります。

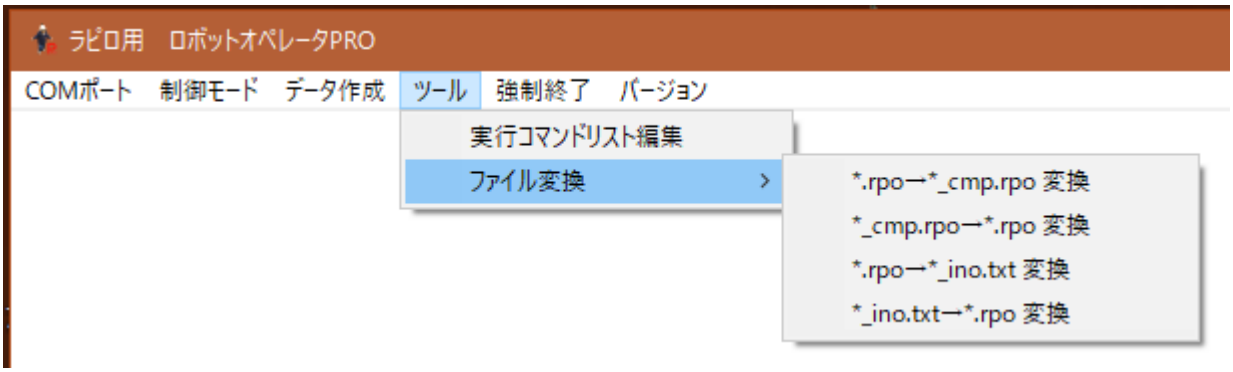
作成ボタン：スライダーで設定した値をオートモードデータファイルに作成します。**最大 255 フレームまで。**作成ボタンをクリックするたびに選択されているコマンドがファイルの後ろに付け足されます。

時間：**前のポーズから設定したポーズに変化するのにかかる時間長さで、値が大きいほどゆっくりとした動きになります。1～255 で 0.1 秒**を単位として設定します。**0 を設定すると何もしません。**

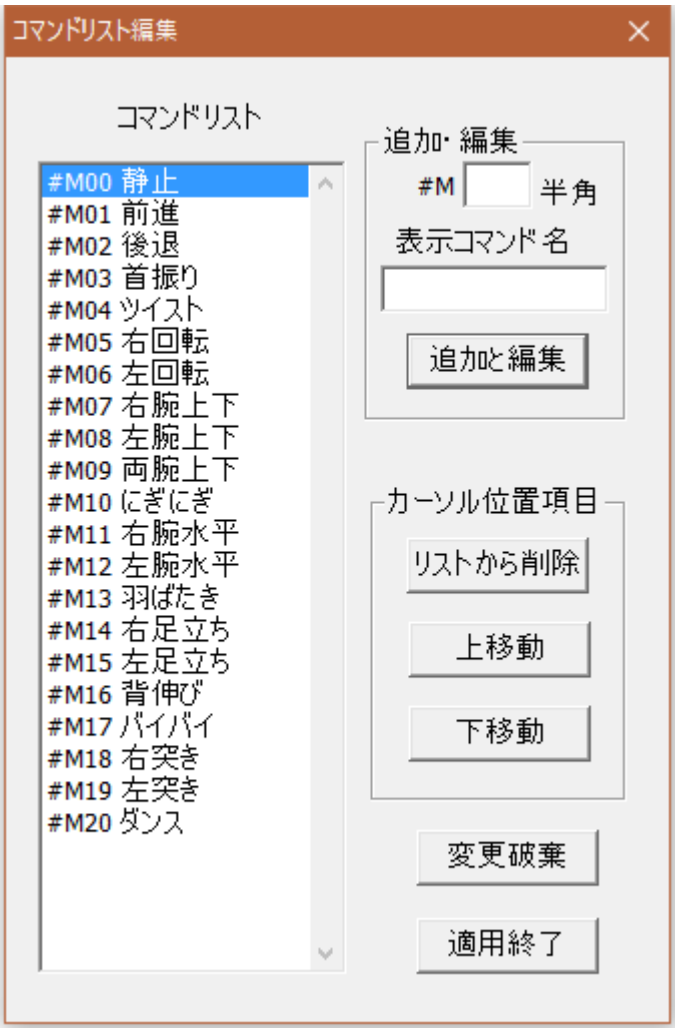
動作間隔：**設定したポーズに変化し**た後、次のコマンドが送信されるまでの時間を表します。最大値 999 で **0.1 秒**を単位として設定します。

終了ボタン：データ作成を終わり、ファイルを保存するための窓が開きます。適当な名前を付けて保存してください。拡張子（.rpo）は自動的に付けられます。

ツール



リスト編集：ダイレクトコマンドを選択する　DirectCmd_List.txt リストを編集するツールです。



＃M エディットボックス:2 桁の整数値を半角で書き込み、#M** の新しい動作モードを登録する。

表示コマンド名エディットボックス：#M** に分かりやすい名前を付けて登録する。

追加と編集ボタン：編集したいコマンド番号を入力し、新たらしいコマンド名を入力して、追加と編集ボタンをクリックすると、#M** に表示されたコマンド名と書き換わります。

この時＃M**が存在しなければ最下行に付け加わります。

表示コマンド名が空欄の場合何もしません。

コマンド番号が空欄の場合、自動でコマンド番号が割り振られ、入力したコマンド名で最下行に書き加えられます。

カーソル位置には関係しません。

リストから削除ボタン：左側のリスト一覧の項目をクリックすると青色カーソルがクリックした項目に移動します。
項目を選択した上で**リストから削除ボタン**をクリックすると選択した項目がリストから削除されます。

上移動・下移動ボタン：移動させたい項目に青色カーソルを移動させてボタンを押すと選択した項目が 上 あるいは 下 へ移動します。
リストに並ぶ順番は任意ですが、左側に表示される＃M**がコマンドになります。

変更破棄ボタン：変更結果が反映されずに終了します。

適用終了ボタン：リストに加えた変更が **DirectCmd_List.txt** 適用されて終了します。

DirectCmd_List0.txt（Windows(C:)>Program File(x86)>RobotOperatoe フォルダ内にあります）**オリジナル版**
DirectCmd_List.txt（Windows(C:)>Program File(x86)>RobotOperatoePRO フォルダ内にあります）**PRO 版**ともにテキストエディタで編集できますが、フォーマットを守る必要があります。行数は５０行までです。

注：リスト編集でコマンド数を増やしても ino スケッチを書き換えなければコマンド数は増えません。

ファイル変換

実行ファイルの形式を相互に変換します。

***.rpo→*_cmp.rpo 変換：ポーズ動作データ作成**で作成されたオートモードデータファイルはポーズをとるために必要なデータのすべてが含まれているのですが、連続動作をするときに必要なデータはひとつ前のポーズデータと比較して変化したデータ（差分データ）があれば十分との考えでデータ量を減らしたものです。

オートモードデータファイル(***.rpo)をファイル選択窓から選択して OK をクリックすれば自動的に圧縮されたファイル(***_cmp.rpo)が作られます。圧縮元ファイルは保存されます。

注：コマンド動作データ作成で作成されたオートモード**データファイルは圧縮できません**

***_cmp.rpo→*.rpo 変換**：圧縮ファイルの逆変換で、一旦圧縮したデータファイルから元データファイルを復元するものです。

***.rpo→*_ino.txt 変換：自動実行ファイルの各フレームデータを ino スケッチデーターの形式に変換します。**

このデータファイル（*_ino.txt）は８行一組が一命令になり、ino スケッチに追加してモーション命令数を増やすことができます。

作成データ数が８より少ない場合は、不足数分のデータが自動的に付け加えられて８行に成るように作成されますが、多い場合は全フレームを作成しますので行数を減らすようにエディタで編集してください。

例：上から７行が作成データで最後の１行が付け加えられたデータ。(****_ino.txt)

```
{{ 90,  90,   0,130,  90,180,  50,  90,  90,  90,  90,  90,121,   0,255,   5},
 { 90,  90,   0,130,  90,180,  50,  90,  90,  90,  90,  90,121,117,255,   5},
 { 90,  90,   0,130,  90,180,  50,  90,  90,  90,  90,  90,121,117,117,   5},
 { 90,  90,   0,130,  90,180,  50,  90,  90,  90,  90,  90,121,117,  0,   5},
 { 90,  90,   0,130,  90,180,  50,  90,  90,  90,  90,  90,121,251,  0,   5},
 { 90,  90,   0,130,  90,180,  50,  90,  90,  90,  90,  90,  0,251,  0,   5},
 { 90,  90,   0,130,  90,180,  50,  90,  90,  90,  90,  90,  0,130,  0,   5},
 { 90,  90,   0,130,  90,180,  50,  90,  90,  90,  90,  90,  0,  0,255,  0}},
```

モーション命令数を変更した場合には ino スケッチの 13 行目

#define MAXMN 21　　// Max Number of Motions

の MAXMN を＃M の 命令数（＃M00 を含む） に変更が必要です。

***_ino.txt→*_ino.rpo 変換：ino スケッチデータから自動実行ファイルを作成するものです。**

変換可能な ino スケッチデータは上記の例のように先頭に {{ 最後尾に }}, が必要ですが、行数は 255 以下であれば任意です。

***_ino.txt ファイルの動作時間を表すデータが 0 の場合 *_ino.rpo ファイルからは省かれます。**

上記の例の場合は８フレーム目は省かれて７行の .rpo 自動実行ファイルになります。

添付スケッチファイル（RobotOperatorPRO_ver2_1.ino）の説明

ロボットを組み立て電源を入れると、ジージーとサーボモータの音がして足元が落ち着きません。

早速 ino スケッチ RAPIRO_ver0_0.ino の trim[]の値を設定して、コンパイル・書き込みをしなければなりません。

```
// Fine angle adjustments (degrees)
int trim[MAXSN] = { 0,   // Head yaw
                    0,   // Waist yaw
                    0,   // R Sholder roll
                    0,   // R Sholder pitch
                    0,   // R Hand grip
                    0,   // L Sholder roll
                    0,   // L Sholder pitch
                    0,   // L Hand grip
                    0,   // R Foot yaw
                    0,   // R Foot pitch
                    0,   // L Foot yaw
                    0 }; // L Foot pitch
```

参考：じいさんの場合の初期値

```
int trim[MAXSN] = { 0,
                   -1,
                    2,
                   -3,
                    0,
                    0,
                    0,
                    0,
                   -5,
                   -4,
                   -5,
                   -6 };
```

私の場合、コンパイラ（ARDUINO ver1.8.0）が新しいためか関数が見つかりませんエラーが出て書き込みができませんでした。

エラーを解決しても メモリー不足による動作不安定 の警告が出て不安が残ります。

ポーズモードで各サーボモータに指示通りの値が送られたかが PC 側では分からない。

RobotOperator の作成にあたり、PC とロボット間でデータのやり取りをするのですが、ロボットから送られてくるデータには終了コードがないために PC 側ではデータの終了判定ができず、シリアルポートのタイムアウトで判断する必要があります。（タイムアウトを待つのは面白くありませんね）

モーション動作も＃M0～＃M9 までの 10 通りしか登録できません。（1 桁が問題）

ロボットに書き込まれたスケッチのバージョン情報が読み取れない。

サーボモータの回転角が読めないし電源が切れない。

上記の問題を解決するために RobotOperatorPRO_ver2_1.ino スケッチと PC 用ソフト RobotOperatorPRO.exe（ver2.3）を作りました。

RobotOperatorPRO_ver2_1.ino スケッチの変更点（動作に影響する変更点のみ）

1：関数が見つかりませんエラー解消

下記の行を書き加えることで関数宣言をした

```
23  void nextFrame( void );
24  void nextPose( void );
25  int getPose( void );
26  void printThreeDigit( int );
27  void printTwoDigit( int );    //  新しく作成した関数
28  int readThreeDigit( int );
29  int readTwoDigit( int );
30  int readOneDigit( void );
```

2：モーション動作の 2 桁拡張（＃M00～）と動作不安定警告解消

モーション動作の種類数に合わせるために変更（静止を含め 21 種類登録）

```
13   #define MAXMN 21            // Max Number of Motions
```

RAM 領域を多く確保するために、値が変化しない変数を cnst 宣言をして RAM 領域から ROM 領域へと移動させた

（興味のある方はネットで“ino スケッチ　Flash 領域へのデーター格納”で検索してください）

```
3   #include <avr/pgmspace.h>
38   PROGMEM const int Trim[MAXSN] = {
73   PROGMEM const unsigned char motion[MAXMN][MAXFN][TIME+1] = {
307   ***  (int)pgm_read_byte( &motion[0][0][i] ) << SHIFT;
309   ***  (int)pgm_read_byte( &motion[0][0][i] ) + pgm_read_word( &Trim[i] ));
397   ***  - pgm_read_word( &Trim[i] );
424   ***  + pgm_read_word( &Trim[i] ));
456   ***   = pgm_read_byte( &motion[motionNumber][frameNumber][i] );
458   ***   = pgm_read_byte( &motion[motionNumber][frameNumber][MAXSN+i] );
439   ***   = pgm_read_byte( &motion[motionNumber][frameNumber][TIME] );
```

3：ロボットの動作に負担になりますが、ロボットの動作状態を知るため、ポーズモードでは各サーボモータに送られたデータを PC に送り返すようにしました。（ロボットからのデータは PC 画面上青色表示）

```
345, 367, 395, 526, 620     void printTwoDigit( int buf );
371, 377, 398, 533, 547, 559, 571, 580, 606     void printThreeDigit( int buf );     等々
```

4：ロボットから送られてくるデータの最後に LF を付加した

```
416  Serial.print("¥n");          終了コード LF (0x0a)
```

5: ＃M コマンドが実行されたとき frameNumber 変数が初期化されるようにして、フレーム 0 のデータから読み取るようにした。(342, 460 行)

6: ＃R でサーボモータの回転位置角度を読み出せるようにした。(391～400 行)

7: ＃S でサーボモータのスイッチが切れるようにした。(402～405 行)

8: ＃V で ino スケッチのバージョンを調べられるようにした。(5 行, 407～410 行)

RobotOperatorPRO_ver2_1.ino スケッチに手を加える場合のお願い

1：ロボットへの動作コマンドを送るタイミング確認のために ＃C コマンドを送信して、ロボットから ＃C0 が帰ってくるのを確認してから動作コマンドを送っています。
＃C コマンドの変更や削除はしないでください。

2：ロボットから送られてくるデータの最後尾には LF が必要です。 上記 416 行目の
Serial.print("¥n"); は削除や変更をしないでください。
Serial.println( ); では CR/LF が送られ正しく動作しません。

3：＃M00 は静止ポーズのままで、削除や変更はしないでください。

4：＃S コマンドは削除しないでください。モータを止めて節電しています。

5：Operator のシリアルポートはボーレート 57,600 bps 動作に設定されています。

6：5 行目 #define VERSION 2.1 // ver 2.1 にバージョン番号を定義しています