Protch オブジェクト

プロッチ

- プロッチを制御するためのオブジェクトです。
- プロッチは、対話的な動作と自律的な動作があります。(すみません。ここ修正します。)
- 対話的な動作のプログラムでは、先頭に次の1 行を記述してください。

システム□ "protchrm"□使う。

- 自律的な動作のプログラムでは、先頭に次の1 行を記述してください。

システム□ "protch"□使う。

- Protch オブジェクトは「プロ□□protch□という名前で使用できます。
- これらのオブジェクトはあらかじめ用意されており、「作る」を実行する必要がありません。
- 以下に、Protch オブジェクトに共通の命令を示します。
- 待つ: プログラムの実行を止める命令です。引数に秒数を指定してください。
    - （例）実行を1 秒待ちます。

プロ！ 1　待つ。

- 転送 : ドリトルのプログラムをコンパイルし、Protch に転送する命令です。プログラムの最後の行に記述してください。
    - （例）プログラムを転送します□ <code> □プロ！ 転送□ </code> ## ライト
- ライト(LED) のオブジェクトです。
- 左ライトと右ライトのオブジェクトを準備しています。
- 制御したい方のオブジェクトを記述してください。
- 点灯: LED の点灯命令です。
- （例）左ライトを点灯します□ <code> □左ライト！ 点灯□ </code>
- 消灯: LED の消灯命令です。
- （例）右ライトを消灯します□ <code> □右ライト！ 消灯□ </code>

サーボモータ

- サーボモータの回転角度を制御します。
- 0〜180 までの角度を制御値としてパラメータに設定します。
- 左サーボモータと右サーボモータのオブジェクトを準備しています。
- 制御したい方のオブジェクトを記述してください。
- 角度設定: サーボモーターの制御を行う動作命令です。
    - （例）左サーボモーターを180 度に制御をします。

左サーボモータ！ 180　角度設定。

- （例）右サーボモーターを0 度に制御をします。

右サーボモータ！ 0　角度設定。

モータ

＊モーターの回転を制御します。　＊前進、後進、停止、左折、右折、右回り、左回り命令の引数に秒数を指定することで指定した時間の間動作し、その後0.5秒停止します。　＊引数を与えない場合は、実行後に停止しません。

- **前進**: 両輪を同じ速度で正回転します。
- （例）1秒間、前進します。

```
モータ！ 1　前進。
```

- 後進: 両輪を同じ速度で逆回転します。

（例）1秒間、後進します。

```
モータ！ 1　後進。
```

- 停止: 両輪の回転を停止します。

（例）1秒間、停止します。

```
モータ！ 1　停止。
```

- 左折: 右側のモータを正回転します。

（例）3秒間、左折します。

```
モータ！ 3　左折。
```

- 右折: 左側のモータを正回転します。

（例）3秒間、右折します。

```
モータ！ 3　右折。
```

- 左回り: 右側のモータが標準の速度で正回転し、左側のDC モータが標準の速度/5の速度で前転します。

（例）2秒間、左回りします。

```
モータ！ 2　左回り。
```

- 右回り: 左側のモータが標準の速度で正回転し、右側のDC モータが標準の速度/5の速度で前転します。

（例）2秒間、右回りします。

```
モータ！ 2　右回り。
```

- 速度設定: DC モータの速度を0 から255 までの値で設定します。標準では100 になっています。

（例）両輪の速度を255 に設定し、2秒間、前進します。

```
モータ！ 255　255　速度設定。
```

モータ！ 2　前進。

ブザー

* 書き込み版のみのオブジェクトです。

- 演奏: ブザーの出力命令です。
- （例）ドの音階を鳴らします。

ブザー! "ド" 演奏。

メロディ

* 譜面に相当するオブジェクトです。　* 生成したメロディを演奏の引数に与えることができます。

- 作る: メロディオブジェクトの生成命令です。
- （例）音楽という名前のメロディオブジェクトを作成し、ドの音階を設定します。

音楽＝メロディ！ "ド"作る。

追加: 生成したメロディオブジェクトに音階を追加する命令です。。

- （例）メロディオブジェクトの「音楽」にレの音階を追加します。

音楽！ "レ"追加。

- （例）メロディオブジェクトを演奏の引数に与えます。

ブザー! (音楽) 演奏。

スイッチ

* スイッチの計測値を取得します。　* 左スイッチと右スイッチのオブジェクトを準備しています。
* 取得したい方のオブジェクトを記述してください。

- 読む: スイッチが押されている時は0 を、押されていない時は1 を返します

（例）左スイッチの入力値が0 なら『…』を実行します。

「(左スイッチ！ 読む)==0」！ なら「…」実行。

- 接触?: スイッチが押されている時は0 を、押されていない時は1 を返します

（例）左スイッチの接触があれば『…』を実行します。

「左スイッチ！ 接触?」！ なら「…」実行。

光センサ

＊左光センサと右光センサのオブジェクトを準備しています。　＊取得したい方のオブジェクトを記述してください。

- 読む: 光センサの値を0～255 の値で返します。センサの周辺が暗いほど値が大きくなります。

（例）左光センサの入力値が100 より大きかったら『…』を実行します。

```
「(左光センサ！ 読む) > 100」！ なら「…」実行。
```

- 明るさ?: 光センサの値を0～255 の値で返します。センサの周辺が暗いほど値が大きくなります。
- （例）左光センサの入力値が100 より大きかったら『…』を実行します。

```
「(左光センサ！ 明るさ?)> 100」！ なら「…」実行。
```

ラインセンサ

＊ラインセンサ（赤外線フォトリフレクタ）の値を取得します。　＊左ラインセンサと右ラインセンサのオブジェクトを準備しています。　＊取得したい方のオブジェクトを記述してください。

- 読む: ラインセンサの値を0～255 の値で返します。反射が明るいほど値が大きくなります。
- （例）右ラインセンサの入力値が130 より大きかったら『…』を実行します。

```
「(右ラインセンサ！ 読む) > 130」！ なら「…」実行。
```

- 明るさ?: ラインセンサの値を0～255 の値で返します。反射が明るいほど値が大きくなります。
- （例）右ラインセンサの入力値が130 より大きかったら『…』を実行します。

```
「(右ラインセンサ！ 明るさ?) > 130」！ なら「…」実行。
```

バッテリ

- 読む: バッテリの残量を1～5 で返します。
- （例）バッテリの残量が3 以上なら『…』を実行します。

```
「(バッテリ！ 読む) >= 3」！ なら「…」実行。
```

- 残量?: バッテリの残量を1～5 で返します。
- （例）バッテリの残量が3 以上なら『…』を実行します。

```
「(バッテリ！ 残量?) >= 3」！ なら「…」実行。
```

距離センサ

＊距離センサの値を取得します。

- 読む: 距離センサの値をmm(ミリメートル) で返します。
- （例）距離が130mm より大きかったら『…』を実行します。

「(距離センサ！ 読む) > 130」！ なら「…」実行。

- 距離?: 距離センサの値をmm(ミリメートル) で返します。
- （例）距離が130mm より大きかったら『…』を実行します。

「(距離センサ！ 距離?) > 130」！ なら「…」実行。

シリアル

＊ リモート版のみ利用可能なオブジェクトです。

- 読む: プロッチが受信した文字列を取得します。
- （例）プロッチが受信した文字列をラベルを使って表示します。

ラベル!(シリアル!読む) 作る。

- 出力: 引数に指定した文字列を出力する命令です。
- （例）「こんにちわ」の文字列を出力します。

シリアル!"こんにちわ"出力。