RoboSprinT NEO

C プログラム入門

MPLAB X 対応版

REV 1.0

はじめに

本マニュアルは、RoboSprinT NEO を Microchip 社の MPLAB X IDE と C コンパイラ XC 8 を利用して開発する場合の入門マニュアルです。

本マニュアルは、旧キット(NEO)および2014年10月から販売されている新キット(NEO2)いずれ にも対応していますが、プログラムのプロジェクトフォルダ名は各々異なります。

- 旧キット RoboSprint_X.zip を展開して出来る RSOS フォルダ
- 新キット RoboSprint_X2. zip を展開して出来る RSOS2 フォルダ

新旧キットの見分け方は下記です。



メイン基板に白い四角があるのが新型(NEO 2)

第1章 RoboSprint NE0 2の使い方

皆さんの RoboSprinT NE0 2 (以下 RoboSprint と略します) は完成しましたか? この章では、RoboSprint の機能や、使い方、センサーやモーターの調整方法を説明します。



■1 各部の名称と働き

① 左センサー ② 右センサー

路面が黒いか白いかを赤外線の反射で検出。

③左 LED ④左 LED

プログラムで点滅可能。テストモードではセンサーの反応を表示。

⑤左センサー感度調整 VR ⑥右センサー感度調整 VR

対応するセンサーの感度を調整する可変抵抗(VR)。小型のドライバーで静かに回して

感度を調整。回転範囲を超えて無理に回そうとすると壊れるので注意してください。

⑦左モーターパワー調整 VR ⑨右モーターパワー調整 VR

対応するモーターのパワーを調整する可変抵抗(VR)。小型のドライバーで静かに回して

パワーを調整。回転範囲を超えて無理に回そうとすると壊れるので注意してください。

待機モードではユーザープログラムを実行。テストモードでは待機モードに戻る。

①モードボタン

待機モードではテストモードに入る。テストモードではモーターを ON/OFF。

①電源スイッチ

前方にスライドで電源 ON。待機モードになり LED がゆっくり交互に点滅 ①電池ボックス

アルカリ単三電池1本使用。プラスマイナスを間違えないように。

■2 動作モードの説明

RoboSprint には自分で作ったプログラムを実行するための「待機モード」と、センサーやモーターの設 定をするための「テストモード」があります。モードの切り替えは、モードボタンで行います。

●待機モード

電源を入れるとこのモードになります。皆さんの作ったプログラムを実行する事が出来ます。 左右の LED → ゆっくり交互に点滅。 スタートボタン → ユーザープログラム(皆さんが作ったプログラム)を開始します。 モードボタン → テストモードになります。

●テストモード

センサーの感度や、モーターのパワーを調整するモードです。待機モードの時、モードボタンを押すと、 このモードになります。

左右の LED	\rightarrow	センサーの状態を点滅のしかたで示します。
スタートボタン	\rightarrow	待機モードに戻ります。
モードボタン	\rightarrow	左右のモーターを回します。もう一度押すと停止します。

センサーの感度調整方法

テストモードでは、左右の LED のつき方で、左右のセンサーの反応を表示しています。

点滅 → 黒い路面に反応

点いたまま → 白い路面に反応

RoboSprint をコースに置き、センサーが黒い路面上にあれば LED が点滅し、白い路面上では LED が 点いたままになるように、対応する感度調整 VR で調整します。

モーターパワーの調整方法

テストモードでモードボタンを押すと、左右のモーターが回転します。この状態で対応するモーターパ ワー調整 VR を回して、モーターパワーを調整します。

パワーを調整したら、そのまま走らせてみて、まっすぐ走るかどうかを確認してください。

モーターを止めるには、もう一度モードボタンを押します。

第2章 プログラミングの基礎知識

■1 C言語を書くときのルール

C言語がまったく初めての方のために、基本的なC言語の「書き方」のルールを説明します。とりあえ ず、これから試してみる「チュートリアル」のプログラムを、入力するために必要な4項目を覚えましょ う。

ルール1:アルファベットの大文字と小文字は区別される。

普段はあまり気にしませんが、C言語では abc と Abc は違うものとして判断されます。間違えやすいの で気をつけましょう。

ルール2:プログラムの中に日本語は使えない。

プログラムの中に日本語は使えません。「合計=1+2+3;」のような書き方は出来ません。また、プログラ ムの中で空白をあけるのに全角スペース(日本語入力モードでスペースキーを押して入力した空白)を使 ってはいけません。エラーになります。

!ポイント

本文中には日本語が使えませんが、「コメント」という、プログラムにつける注釈には、日本語が使えます。

ルール3:本来、インデント(段付け)はどうなっていてもかまわない。

Cのプログラムは、よくこんな風にかかれています。

2行目の「int i, j;」は一段下がった位置から書き出されていて、5行目の「j+=i;」はさらに一段下がっていますね。このように段付けすることを「インデント」といいます。

このようなプログラムを書き写すときに「どれくらい下げればいいんだろう?」ということが気になりま すね。

実はインデントはどうなっていてもかまわないのです。上のプログラムは

main{
int i, j;
j=0;
for(i=0;i<100;i++) {
j+=i;
}
}

こんな風に書いても問題なく実行できます。ですからインデントのつけ方で悩むことはありません。 インデントをつけるのは、プログラムを読みやすくするためです。プログラムをいろいろと書いていく

うちに、そのほうがわかりやすいということが自然と理解できると思います。

なお、インデントはスペースキーを何回も押すのではなく、「Tab」キー一発で入れると便利です。(ルール2の全角スペースが使えないことに注意してください)

ルール4:キーワードとキーワードは半角スペースで区切る

int や for など、Cプログラム中で用いられる、アルファベットの単語を「キーワード」といいます。 キーワード同士がつながる場合は、間に半角スペースを置きます。

■2 最も基本的なC言語のキーワード

この後の「チュートリアル」でのCプログラムに使われている、もっとも基本的なキーワードを説明しま す。読んだだけではよくわからないかも知れませんが、一通り目を通しておくと、チュートリアルがより わかりやすくなります。

変数

数値を記憶するためのものです。アルファベットで好きな名前がつけられます。iやjなどのアルファ ベット1文字のものから、goukei(合計)heikin(平均)など、自由につけることが出来ます。 変数は、使う前にプログラムの中で「定義」する必要があります。たとえばこんな感じです。

int kurikaeshi;

「int」は変数 kurikaeshi の「型」を表します。int 型は-32768 から 32767 までの整数を記憶することが 出来ます。チュートリアルで使う型は、すべて int 型にします。(注意してください。扱えるのは整数のみ です。1.56 などの小数点を含む数値は扱えません。)

このようにすると、コンピュータは kurikaeshi という名前の、-32768 から 32767 までの数値を入れる

ことのできる、「記憶装置」を作ります。この「記憶装置」に数値を入れるには、次のようにします。

kurikaeshi = 12;

● 計算式

変数を使うと、演算記号を使って計算することが出来ます。たとえば、iとjという int 型の変数がある場合、次のような式ではjに 10 が入ります。

i = 3; j = i + 7;

四則演算は計算式どおりに実行します。つまり、*(掛け算)と/(割り算)が+-より優先されます。また、 ()も使えます。次の式では、jに17が入ります。

これまでの例では、演算記号と数字や変数の間に、半角スペースを置いて、j = i + 7 のように書いて いますが、j=i+7;のようにくっつけて書いても問題ありません。

int型では、割り算では小数点以下を切り捨てます。3 / 2の結果は1になります。

!ポイント

C言語には、四則演算のほかに、特別な演算記号(演算子といいます)がいくつかあります。よく使うものは、「インク リメント演算子」「デクリメント演算子」で、それぞれ++ --と書きます。

この演算子は、変数にくっつけて i++ i-- のように使い、それぞれ、i に1を足す、i から1を引くという意味があ ります。カウンターのように数を数えたいときに使います。

● 繰り返し(for ループ)

C言語に限らず、コンピュータプログラムといえば、箇条書きに並んでいる命令を、上から順に実行していきます。

このようなプログラムの一部を、何回か繰り返したいというときがよくあります。そんなときに便利なのが「繰り返し」ループです。これには for というキーワードを使うので、「for ループ」と呼ばれます。

for ループは繰り返しの回数を数える変数とセットで使います。次の for ループは 10 回繰り返す設定に なっています。



● 条件判断 (if 文)

コンピュータプログラムでは、必ずといっていいほど、外部の条件でプログラムの内容を変えたい場合 が出てきます。

RoboSprint の場合なら、白いラインに沿って走るために、「もし右のセンサーが白いラインを検出した ならば、右のモーターを止めて、右へカーブする、そうでなければ、両方のモーターをオンにして前進」 といった処理が考えられるでしょう。ここで太字で書いた部分が、条件判断です。

Cではキーワード if と else がこれに相当します。次のプログラムで見てみましょう。

if(Rsens == 1) {	もし Rsens の内容が 1 だったら{から}の間を実行
Rmotor=0;	
Lmotor=1;	
}	
else{	そうでなければ、こちらの{から}の間を実行
Rmotor=0;	
Lmotor=1;	
}	

このプログラムの Rsens == 1 の部分を「条件式」といいます。この式が成り立つかどうかを判断して、 プログラムの流れを変えています。条件式には次のようなものがあります。(普通の数式とはちょっと異な るので注意しましょう)

- A == B A と B が等しい
- A != B A と B が等しくない
- A > B A が B より大きい
- A < B A が B より小さい
- A >= B A が B 以上
- A <= B A が B 以下

■3 センサーとモーターを使う

RoboSprint には専用の OS(基本ソフト)である RSOS が搭載されています。これは Windows などと同じように、RoboSprint のハードウェアとユーザープログラムの橋渡しをする特別のソフトウェアです。これを 使うことで、初めての方でもセンサーやモーターを使ったプログラムを簡単に作ることが出来ます。

RoboSprint を動かすには、皆さんの作るプログラム(これをユーザープログラムと呼びます)から、 RoboSprintの目であるセンサーで路面が黒いか白いかを調べたり、左右のモーターを回したり止めたりし なくてはなりません。

RSOS ではこれらの受け渡しに「変数」を使います。ユーザープログラムではセンサーやモーターに割り当てられた変数を読み書きすることで、センサーの反応を読み込んだり、モーターを回したりすることが出来ます。

次の図を見ながら RSOS の基本的な使い方を学びましょう。実際の使い方は、この後の「チュートリアル 編」でやってみます。



センサーを調べる

左右のラインセンサーの反応は Rsens、Lsens という変数に入ります。この変数が1ならばセンサーの下の路面が白、0ならばセンサーの下の路面は黒です。センサーの感度はそれぞれの感度調整 VR(可変抵抗)で調整します。

!ポイント センサーの一瞬の変化も見逃さないよう、RSOS は1秒間に 2000 回も変数の内容を最新の状態に書き換えています。

● モーターと LED を使う

モーターを回転させるには、Rmotor と Lmotor という変数を使います。これに 1 を書けばモーターは回転し、0 を書けばモーターは停止します。

それぞれのモーターのパワーは対応する調整 VR をまわして調整します。(「テストモード」で行います) 左右の LED(発光ダイオード)も同じように Rled、Lled に1か0を書くことでつけたり消したりすること が出来ます。

● 待ち時間を作る

プログラムを作っていると、何秒間か時間待ちをしたいときがあります。

たとえば、走行中、右へ少し方向を変えて走らせたいときなどは、右のモーターを止め、左のモーター だけを回して、そのまま1秒待ってからまた両方のモーターを回せばよいでしょう。このようなときに使 うのが、wait_ms(ミリ秒)という RSOS の関数呼び出しです。

関数呼び出しは、今まで出てきた変数とは違って、括弧の中に時間の長さをミリ秒(1/1000 秒)単位でいれると、その時間だけプログラムの実行を中断します。

! ポイント

関数呼び出しは変数と比べると理解するのがちょっと難しいかもしれません。でも、使い方は簡単ですから、使いな がらなれていきましょう。

■ 4 高度な RSOS の機能(API)

RoboSprint の OS、RSOS にはより高度なハードとのインターフェイスがあります。ここではプログラミングの知識のある方向けに、API のリストを示します。

なお、ソースファイルの RSOS.h がこれらのヘッダファイルになっています。

・センサーAPI (変数読み出し)

Rsens, Lsens	左右センサのデジタル読み出し 黒→O、白→1(VR にて閾値を調整)
Rsens_read, Lsens_read	左右センサにて路面の色(赤外線の反射の程度)をアナログ読み出し
	0 - 255の値(1 バイト)が読み出される

モーターAPI (変数書き込み)

Rmotor, Lmotor	左右モーターの回転停止 ON→1、OFF→O(VRにてパワー調整)
Rmotor_pwr,Lmotor_pwr	左右モーターのパワーをパーセントで指定(0→100%)
	この API を使う時は Rmotor, Lmotor には2を入れておく
	* 参考:使用例
	Rmotor = 2; //2 をセットしてパワー制御を有効に
	Rmotor_pwr = 50; //右モーターをパワー50%で回転

wait_ms(3000); //3 秒待って

Rmotor_pwr = 0; //モーター停止

・LED API (変数書き込み)

Rled,Lled 左右 LED の ON/OFF ON→ 1、OFF→0

・タイマー(変数読み書き)

timer10ms0, timer10ms1 各々10ms ごとにカウントアップする 16bit 符号なしのカウンタ timer100ms0, timer100ms1 各々100ms ごとにカウントアップする 16bit 符号なしのカウンタ

タイマーは 10ms ごとにカウントアップするものが 2本、100ms ごとのものが 2本、計 4本あります。 いずれも符号なし 16bit なので、それぞれ 655,35 秒、6553,5 秒まで計測可能です。

ユーザープログラムでリセット、プリセットして使います。OS では単位時間ごとにインクリメントするだけです。

たとえば、スタートしてから一定時間経過したことを知りたかったら、スタート時に0 リセットし、その値を定期的に監視していれば良い訳です。

第3章 チュートリアル

ここでは、プログラミングの実際を体験するために、3つの「チュートリアル」を用意しました。それ ぞれの目的は、次のようになっています。

チュートリアル 1	Cプログラムを書いて	こ、RoboSprint に転送する方法を体験
-----------	------------	-------------------------

チュートリアル2 繰り返し処理を使ったプログラムの作り方を体験

チュートリアル3	条件判断を使ったプログラムの作り方を体験
----------	----------------------

■開発環境について

RoboSprintの開発には Microchip 社の純正開発環境(IDE)、MPLAB X と書き込みツールとして PicKit2 (または PicKit3)を使います。C 言語のコンパイラは、同じく Microchip 社の HP からダウンロードで きる、XC 8 無償板を使用します。

インターネット上にいろいろと情報がありますから、調べておくとよいでしょう。

IDE で本キット用のプロジェクトファイル、RSOS2 を開くとこのような画面が現れます。



チュートリアル1:ダッシュ走行

■このチュートリアルについて

ミッション:

スタートボタンを押すとダッシュし、少し走ったら自動的に止まるプログラムを作れ。

解説:

ここでは一般的なプログラムの作り方を体験するため、最も簡単なプログラムを作ります。左右のモー ターを一定時間だけ回して、その後停止するプログラムです。

ステップ:

このチュートリアルでは下記の項目をステップ・バイ・ステップで体験します。 ステップ1・プログラムの元になるフォルダ、「プロジェクトフォルダ」を作る ステップ2・「MPLAB IDE X」を起動してC言語のプログラムを書く ステップ3・プログラムを「ビルド」してロボスプリントに転送できるようにする ステップ4・プログラムの間違いを直す ステップ5・「PicKit2」をロボスプリントに接続し、プログラムを転送する ステップ6・思ったとおりに動くまで3,4,5を繰り返す

■やってみよう

ステップ1・プロジェクトフォルダを作る

解説:

自分のプログラムを作るには、プログラムに必要ないろいろなファイルをひとまとめにした「プロジェ クト」を作る必要があります。元になる「RSOS2 プロジェクト」が用意されているので、これを元にして 簡単に作ることが出来ます。

<u>最初に RoboSprint_X2.zip をダウンロードなどで入手し、展開して作成される RSOS2 フォルダを</u> MPLABProjects フォルダに保存してください。

(注意!旧キットの場合は RoboSprint_X.zip をダウンロードして RSOS フォルダを使います:p1 参照)

!ポイント

MPLABProjects フォルダは、MPLAB X IDE をインストールしたときに、WorksSpace(作業用のフォルダ)として作成されるものです。デフォルトだとこの名前のフォルダが下記に作られます。

Windows C:¥Users¥<あなたのログイン名>¥MPLABXProjects

Mac /Users/<あなたのログイン名>/MPLABXProjects

RSOS2 フォルダは他の場所に置いてもいいですが、パス名に日本語が入るとダメです。c:直下などがお勧めです。

手順:

- 1 · MPLAB X IDE を起動する
- 2 · File メニューから Open Project...を選び、プロジェクトファイル「RSOS2」を開く
- 3・IDEのProject ウィンドウに表示される RSOS2 を右クリックしてメニューを表示

4 ・そのメニューから Copy...を選択し、開いたウィンドウの一番上に好みの名前を半角アルファベット で入力し、画面の Copy ボタンをクリック(このチュートリアルでは samplel とします)



- 5 · IDE の Project ウィンドウに作成したプロジェクト(sample1)が表示されるのを確認
- 6 ・再び RSOS2 を右クリックしてメニューを開き、Close を選択
- 7 · RSOS2 が Project ウィンドウから消え samplel だけが表示されることを確認
- 8・プロジェクト samplel の開発準備完了です!

!ポイント

プロジェクトフォルダの名前には半角アルファベットと数字、_(アンダーバー)だけが使えます。これ以外の記号や、 日本語を使うと、うまく「ビルド」出来ない場合があるので気をつけましょう。

ステップ2・MPLAB X IDE でプログラムを入力

解説:

RoboSprint に搭載されているマイクロコンピュータをプログラムするためのソフト、MPLAB X IDE でプ ログラムを打ち込んでみます。MPLAB X IDE はワープロソフトのようなもので、ワープロで作る文章が C 言語のプログラム、印刷作業が RoboSprint への転送と考えるとイメージがつかめるでしょう。

手順:

- 1 · IDE を起動しプロジェクトを開く
- プログラムを入力する準備をします。
- ・ デスクトップのショートカットアイコンかスタートメニューから MPLAB X IDE を起動
- 「sample1」が Project ウィンドウに表示されてなければ、File メニューから Open Project…を選択 しファイル選択画面で「sample1」を開く
- ・ 「sample1」フォルダリストが表示されていなければ、ダブルクリックして構成フォルダリストを開く
- ・ 構成フォルダの Source Files を同様の方法で開き、表示される myProgram.c をダブルクリックで開く

・ 中央のウィンドウに myProgram. c の内容が表示される

```
2 · 下記のプログラムを myProgram. c に入力
```

プログラム本体の入力です。詳しい説明はあとでするので、今は入力することに集中しましょう。

 すでにプログラムが書いてあれば消し、下のような状態にする void user_program(void) {

}

「両方のモーターを1秒だけ回して止める」プログラムを{と}の間に下記のように入力
 void user_program(void) {

```
Rmotor=1;
Lmotor=1;
wait_ms(1000);
Rmotor=0;
Lmotor=0;
```

}

!ポイント

「C 言語を書くときのルール」を思い出して書き込みましょう。また、行の最後のセミコロン(;) は忘れやすいので 注意しましょう。 入力方法は普通のワープロソフトと同じです。コピーや貼り付けも同じように出来ます。Edit メニューから Copy(コ ピー) Paste(貼り付け)などを選ぶか、マウスの右クリックで編集メニューを出して行います。 プログラムを書き込んでいると、文字の色が変わったり、文字に波線がついたりすることがあります。これは、IDE がプログラムを認識し、わかりやすく表示しているのであまり気にすることはありません。ただし、行番号部分に

ステップ3 · 入力したCプログラムを「ビルド」してロボットに転送できるようにする 解説:

この作業は人間にわかりやすく書かれたC言語のプログラムを、RoboSprintのマイコンが理解できるよう変換する作業です。難しそうですが、アイコンをひとつクリックするだけで後はコンピュータがやってくれます。ご心配なく。

手順:

1・IDEのツールパレットから「ビルド」アイコンをクリック



2・Output ウィンドウに文字が流れて、最後の方に下記のように表示されることを確認(カッコ内は作業 に掛かった時間で、状況によって変化するので気にしなくてよい)

BUILD SUCCESSFUL (total time: 1s)

!ポイント

BUILD FAILED (exit value 2, total time: 371ms)

このメッセージが出た場合はプログラムのどこかに誤りがあります。ビルド失敗です。(カッコ内は変わります)

ステップ4・プログラムの書き方の誤りを直す

ステップ3でビルドに失敗した場合、プログラムの書き方に誤りがあります。ビルドに成功した人も、 どれかの;を消してみてエラーの出かたを確認してください。

- スクロールして Output ウィンドウの<u>青色</u>で表示されているエラーメッセージを探し、ダブルクリック (いくつかある場合はとりあえず最初のエラーを選ぶ)
- myProgram のエラーのありそうな行にマークがつき、カーソルが移動するので、その行とそれ以前の 行に間違いがないかよく調べる
- ・ 間違いを直したらビルドしてエラーがなくなったかどうか確認

!ポイント

エラーはマークのついた行から(その行も含む)前のほうにあります。大体はその行か、ひとつ前の行にありますが、 数行前にある場合もあるので要注意です。

!ポイント

ひとつのエラーが原因で、エラーメッセージがいくつも出ることがあります。こうなるとなぜエラーなのかわからな くなってしまいます。そのため、ひとつのエラーを直したら、必ずビルドしてエラーの状態を確認してください。それ を直したために他のエラーメッセージも出なくなる場合が多いからです。

!ポイント

ここでわかるエラーは「文法」のエラーです。日本語にたとえると「ここには句点が必要です」とか「こういう言葉 は日本語にはありません」といったことをチェックしています。だから Rmotor=1;を誤って Rmotor=0;と書いてもこの場 合エラーにはなりません。1 も 0 も Rmotor に与えることが出来る数値だからです。

ステップ5・プログラムを Pickit2 で転送する

ビルドしたプログラムを RoboSprint に送って実際に動かしてみます。これには MicroChip 社の Pickit2 というプログラム書き込み機を使います。事前にインストールしておく必要があります。

手順:

1 · Pickit2 がパソコンに接続されていなければ接続

普通は Pickit2 をパソコンに接続したままで問題ありません。はずしてある場合は、IDE をいったん終 了してから接続します。

2 · Pickit2 を RoboSprint に接続

RoboSprintの電源スイッチを切って、方向に注意して写真のように接続します。このとき PicKit2 から 電源が供給され、LED が点滅したりする場合がありますが、問題ありません。そのまま作業を進めてくだ さい。



! ポイント

接続した状態で力をかけると RoboSprintの接続コネクターが曲がってしまいます。気をつけましょう。

3・IDE のプログラムアイコンをクリック

◆これがプログラムアイコン。矢印の向きに注意! 画面右上のツールバーにあります。 数秒後、Output ウィンドウに下記のような表示が出れば OK です。Pickit2 を外してかまいません。 Programming completed Running target...



ステップ6・プログラムをテスト

プログラムを書き込んだら、早速テストしてみましょう。電源スイッチをいれ、スタートボタンを押す と、約1秒走行します。

思ったように動かないときには、ステップ2に戻って、プログラムが正しく入力できているかを確認の うえ、ステップ3、4、5をもう一度行ってください。

!ポイント

左右のモーターのスピードはそれぞれ VR4 と VR3 をまわすことで調整できます。「テストモード」であらかじめ調整し ておきましょう。

■プログラムの説明

●プログラムの実行

自分で作ったプログラムは

void user_program(void) {

から、最後の

}

の間に書き込みます。

RoboSprint の「スタート」ボタンを押すと、プログラムの最初の行から順番に実行します。

●それぞれの命令の説明

モーターの回し方などは第2章の「■3 センサーやモーターを使う」に説明があります。

Rmotor=1;

Lmotor=1;

1をモーターの変数に入れるとモーターが VR で設定したスピードで回ります。右モーターのほうが先に 回りだすように見えますが、それぞれの命令は1マイクロ秒(百万分の1秒)ほどで実行してしまうので、 実際は同時にモーターに回転指令が行くと考えてかまいません。

wait_ms(1000);

この命令は、この行で1000ミリ秒(1秒)プログラムの進行を止める働きをします。wait_msのカッコ内に書いた時間だけ止まり、時間が過ぎると次の命令に進みます。

ここでは、モーターが回った状態で、1秒間プログラムを止めておくということです。

Rmotor=0;

Lmotor=0;

前の wait_ms 命令で1秒待った後、モーターを止めるのがこの部分です。これで RoboSprint は停止しま す。0 を変数に入れるとモーターは停止します。これがプログラムの最後の命令なので、RSOS はこれを実 行したらすぐに「待機モード」に戻ります。

!ポイント

実際は、待機モードに戻るとモーターは自動的に止まるようになっています。ですから、モーター停止の処理をしな くともよいのです。しかし、プログラムの作法として、ある処理が終了したら、きちんと後始末をして、他の処理へ戻 るという原則があります。「開けたら閉める」ということですね。もっと大規模なプログラムだと、これをおろそかにす ることで、正体不明のトラブルに見舞われることもあり、ここではキチンと後始末をしています。

チュートリアル2:連続走行

■このチュートリアルについて

ミッション:

前進→カーブを繰り返して連続走行し、しばらくしたら停止するプログラムを作れ。

解説:

前進→カーブを繰り返して走り続けます。少し広い場所があればダイナミックに走らせて遊ぶことが出 来ます。あまり広い場所がなければ、前進やカーブの時間を調節して壁や家具にぶつからないようにしま しょう。

ここでは一連の処理を決められた回数繰り返す、「繰り返し処理」を試して見ます。

ステップ:

ステップ1・どんなプログラムにするか決める ステップ2・新しいプロジェクトフォルダを作る ステップ3・プログラムを書く ステップ4・RoboSprintに書き込んでテストする ステップ5・3、4を繰り返してうまく動くようにする

■やってみよう

ステップ1:どんなプログラムにするか決める

解説:

いきなり IDE でプログラムを書き始めるのではなく、まず、どんなプログラムにするのかの計画を立て ます。

チュートリアル1では、作るプログラムがとても簡単なので、この手順は省きました。しかし、少し複 雑なプログラムを書くときには、これをしっかりやっておかないと、きちんと働かないプログラムに悩ま されることになります。

手順:

1・プログラムしたいことを箇条書きに書き出す

スタートボタンを押したあと、RoboSprint にやらせたいことを簡単に箇条書きにしてみます。紙に書き 出す場合は、後から書き足せるように、コピーの裏紙やノートなどの少し大きめの紙がよいでしょう。ワ ープロソフトなどでもかまいません。

今回のミッションならこんな風になるでしょう。

- 前に少し走る
- ・ カーブする
- 以上をしばらくの間繰り返す
- 2 ・ 箇条書きの項目に数値などを加えて詳しくする

1 で書き出した項目を、具体的にしていきます。簡単に言うと「なにを」や「どれだけ」や「どうする」 などをはっきりさせていくことです。これをするとプログラムが作りやすくなります。

たとえば最初の項目はこんな風になるでしょう。

前に少し走る

→ 左右のモーターを回転させて1秒待つ

1秒かどうか、迷うところでしょうが、ここでは必ず数字にしていくことがポイントです。今は「仮に 1秒」でもいいのです。項目に書き足していきましょう。

次の項目は、同様にこんな風でしょう。この設定だと RoboSprint は左にカーブします。

・ カーブする

→ 右のモーターを回転、左のモーターを停止して1秒待つ

では、最後の項目です。

「前に走る」と「カーブする」が「なにを」にあたります。「どうする」は順番に繰り返し実行するということですね。

では「しばらくの間」はどうしましょう? 普通に考えると「10秒間」などと時間を指定するのがよさ そうです。でも、今は時間をどう指定するかはわかりません。(モチロン、時間を指定するための機能はあ ります。後ほど勉強しましょう)

ここでは、何回この操作を繰り返したかで、時間を計ることにします。現在、直進とカーブの合計時間 が2秒なので、これを10回繰り返すことで、20秒間走行させることにしましょう。

つまり最後の項目を具体的にするとこうなります。

・ 以上をしばらくの間繰り返す

→ 「前に走る」「カーブする」を10回繰り返したら、左右のモーターを停止する

!ポイント

このようにプログラムが実現しようとする機能を、文章や図でわかりやすく説明したものを「プログラムの仕様書」 と呼びます。たとえばゲームのプログラムも、開発前には詳しい仕様書を作り、これを何回も修正しながら、面白さや、 遊びやすさなどを検討しています。これはプログラムを作り始めてしまうと、大きな変更がやりにくくなるためです。

ステップ2・新しいプロジェクトフォルダを作る

解説:

チュートリアル1でやったように新しいプロジェクトフォルダを作ります。今回のプロジェクト名は sample2にしましょう。

ステップ3・プログラムを書く

解説:

ステップ2で作った仕様書に基づいて、プログラムを作ります。仕様書の項目を、「コメント」と言う形 でプログラムに書き込んでおくと、わかり易いプログラムになります。

今回入力するプログラムは下記です。

```
void user_program(void) {
    int i;
```

}

for(i=0; i<10; i++){ //10回繰り返す

```
//1秒間前進する
Rmotor=1;
Lmotor=1;
wait_ms(1000); //前進時間を決める
//1秒間左カーブする
```

```
Rmotor=1;
Lmotor=0;
wait_ms(1000); //カーブの角度を決める
```

```
//モーターを止めて終了
Rmotor=0;
Lmotor=0;
```

このプログラムで新しく出てきたのは、「for ループ」と「コメント」です。

for ループの仕組み:

}

変数を使って、同じ処理を繰り返し行います。書き方は次のようになります。

for(変数の初期値;繰り返しを続ける条件;変数の増分) {

この部分のプログラムを繰り返す

}

この例では、

変数の初期値	i=0	繰り返し回数のカウントは、変数 i で行う。初期値は0
繰り返しを続ける条件	i<10	i が 10 を越えない間は、処理を繰り返す
変数の増分	i++	1回処理を行ったごとに、i に1を足す

と、なっています。つまり、i は 0 から 9 までカウントするので、1 0 回繰り返し処理を行うことになります。

コメント:

プログラムリストの中で、//から始まっているのがコメントです。//から行末まで、好きな言葉を書くこ とが出来ます。システムはプログラムをビルドする際、コメントは無視するようになっています。

この例のように、仕様書で決めたことを書き込んでおくと、仕様書のどの部分がプログラムのどの部分 にあたるのかがわかり易いので、デバッグやプログラムの手直しのときに有効です。

ステップ4 · RoboSprint に書き込んでテストする

解説:

チュートリアル1でやったようにプログラムを書き込みテストしましょう。

チュートリアル3:センサーを使う

■このチュートリアルについて

ミッション:

RoboSprint を床に置くと走り出し、持ち上げると止まるプログラムを作れ。

解説:

センサーを使って床面を検出し、モーターを ON/OFF します。

センサーは赤外線を発光し、床からの反射を検出しています。このため、明るい色の床では反応して1 になりますが、黒い床の上や、RoboSprintを持ち上げて床から離した状態では、赤外線が反射しないので、 0になります。

ここでは、センサーの状態で、プログラムの流れを切り替える方法を試してみます。

ステップ :

ステップ1・どんなプログラムにするか決める ステップ2・新しいプロジェクトフォルダを作る ステップ3・プログラムを書く ステップ4・RoboSprintに書き込んでテストする ステップ5・3、4を繰り返してうまく動くようにする

■やってみよう

ステップ1:どんなプログラムにするか決める

解説:

ここでは、右側のセンサーで床の有無を調べることにして、チュートリアル2でやったように仕様を作 ってみましょう。

・ センサー反応ありならばモーター回転

→ 右のセンサーが1だったら、左右のモーターを回転にセット

- ・ センサー反応なしならばモーター停止
 - → 右のセンサーが0だったら、左右のモーターを停止にセット
- ・ 以上を繰り返す
 - → この処理を無限に繰り返す。

繰り返し回数を「無限大」としたので、このプログラムを止めるには、RoboSprintの電源を切ります。 今回の新しい処理は、「もし~ならば」という条件判断です。第2章の「2・最も基本的なC言語のキーワ ード」で説明しているので復習しておきましょう。

ステップ2・新しいプロジェクトフォルダを作る

解説:

チュートリアル1でやったように新しいプロジェクトフォルダを作ります。今回のプロジェクト名は sample3 にしましょう。

ステップ3・プログラムを書く

解説:

ステップ1で作った仕様書に基づいて、プログラムを作ります。 今回入力するプログラムは下記です。

```
void user_program(void) {
```

このプログラムで新しく出てきたのは、「無限ループ」と「if(条件判断)」です。

if (条件判断):

}

キーワード if については第2章の「2・最も基本的な C 言語のキーワード」で説明しているので、確認 しておきましょう。

else 以降(条件が成立しない時の処理)が必要ない時は、省略して if だけをこのように書くことも出来

ます。

if(条件式) {

条件式が成立した時の処理

}

!ポイント

条件式の書き方は普通の数式と少し異なるので注意が必要です。特に間違い易いのは、「~である」と言う意味の「==」 です。イコールは二つです。第2章の「2・最も基本的なC言語のキーワード」で確認しておきましょう。

無限ループ:

for ループでは指定回数だけ処理を繰り返すことが出来ましたが、このようにパラメータを省略すると、いつまでも処理を繰り返すループ処理、「無限ループ」になります。

for(;;) {

この部分のプログラムをいつまでも繰り返す

}

無限ループを実行していると、ここから抜け出すことが出来ません。しかし break というキーワードを 使うと、条件判断との組み合わせで、無限ループから抜け出すプログラムが作れます。

```
for(;;) {
    if(Rsens == 1) {
        break; //右センサーが反応したら無限ループを終わる
    }
}
//break を実行したら、ここから処理を続ける
```

このプログラムは、無限ループで右センサーの反応をチェックし続け、反応があったら、無限ループから抜け出して、次の処理に移ります。

ステップ4 · RoboSprint に書き込んでテストする

解説:

チュートリアル1でやったようにプログラムを書き込みテストしましょう。

プログラムを始める前に、テストモードでセンサーが床の上で反応するように(LED が点きっぱなしに

なるように)調整しておきます。プログラムを止めるときには、電源スイッチをいったん切って、1秒ほど待ってから入れ直してください。

著作権などについて 本マニュアルは非売品です。複製・無償配布は自由に行ってかま いませんが、販売は禁止します。 本マニュアルの著作権はバンダイロボット研究所にあります。 本マニュアルで使用されている画像には、Microchip 社他に著作権 のあるものが含まれる場合があります。また、本文中には Microchip 社他の登録商標が含まれる場合があります。