RoboSprinT KIT NEO

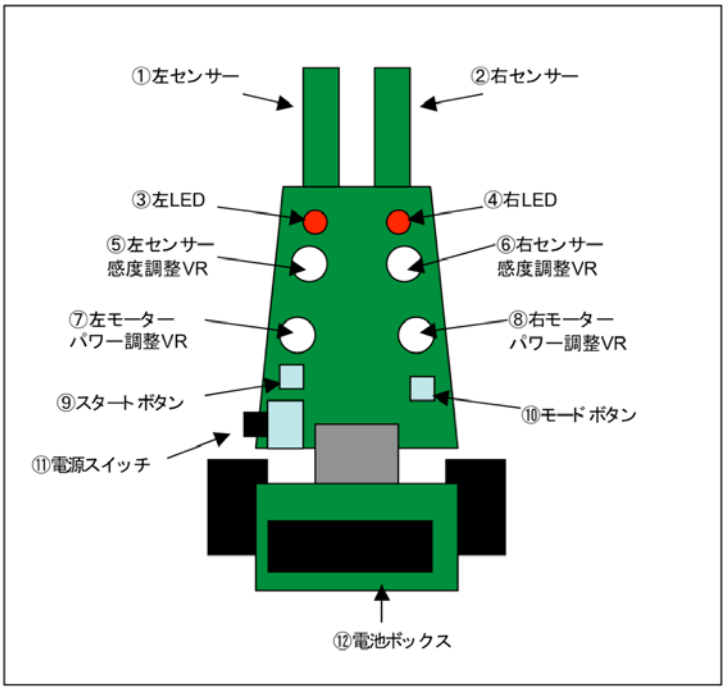
Ｃプログラム入門

REV 0.71

第１章　RoboSprint KIT NEO の使い方

皆さんの RoboSprinT KIT NEO（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。

⑨スタートボタン

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。

⑩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。

⑪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

⑫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	→ 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	→ 待機モードに戻ります。
モードボタン	→ 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。

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

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。

```
i = 3;
j = i + 7 * 2;
```

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。

```
int i,k;◀――――――――  変数 i と k を定義
k=0;
for(i=0;i<10;i++){◀――  変数 i の初期値が 0、i が 10 より小さい間、
                        i を 1 ずつ足しながら、{ と }の間を繰り返し実行
        k=k+i;  ◀――――  これを 10 回実行。(0 から 9 までの合計が k にはいる)
}
```

● **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

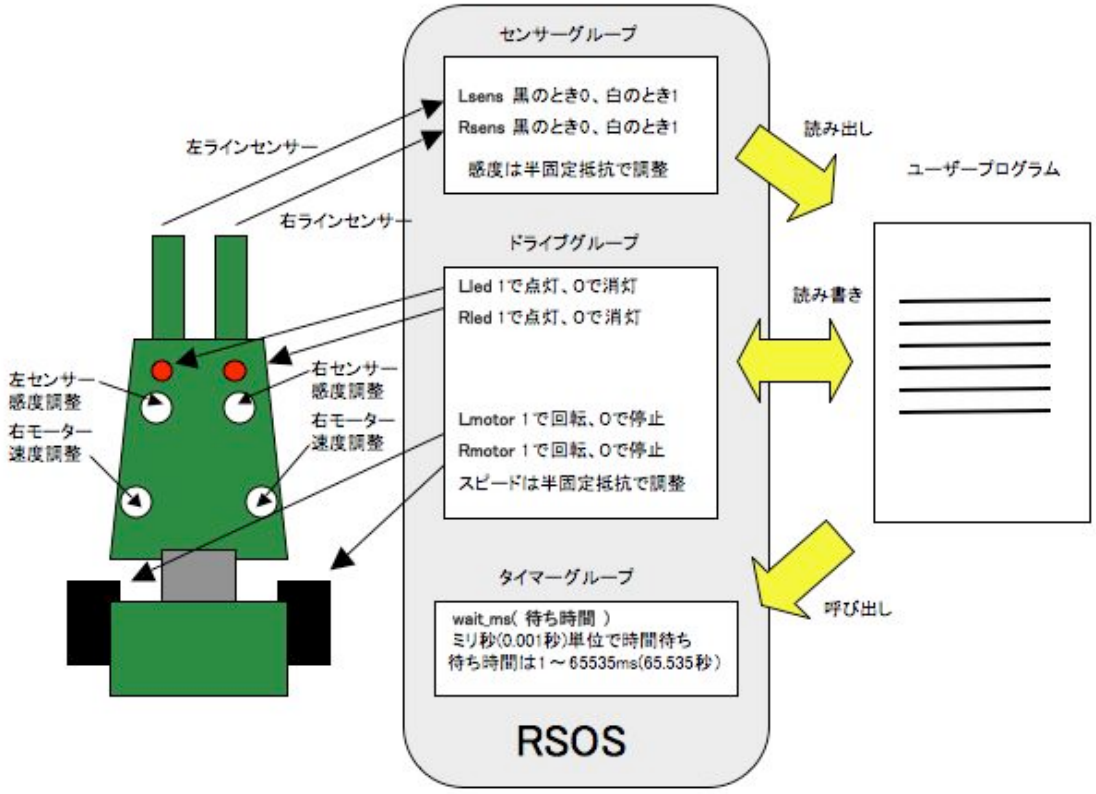
■３　センサーとモーターを使う

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→０、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→0（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

・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。

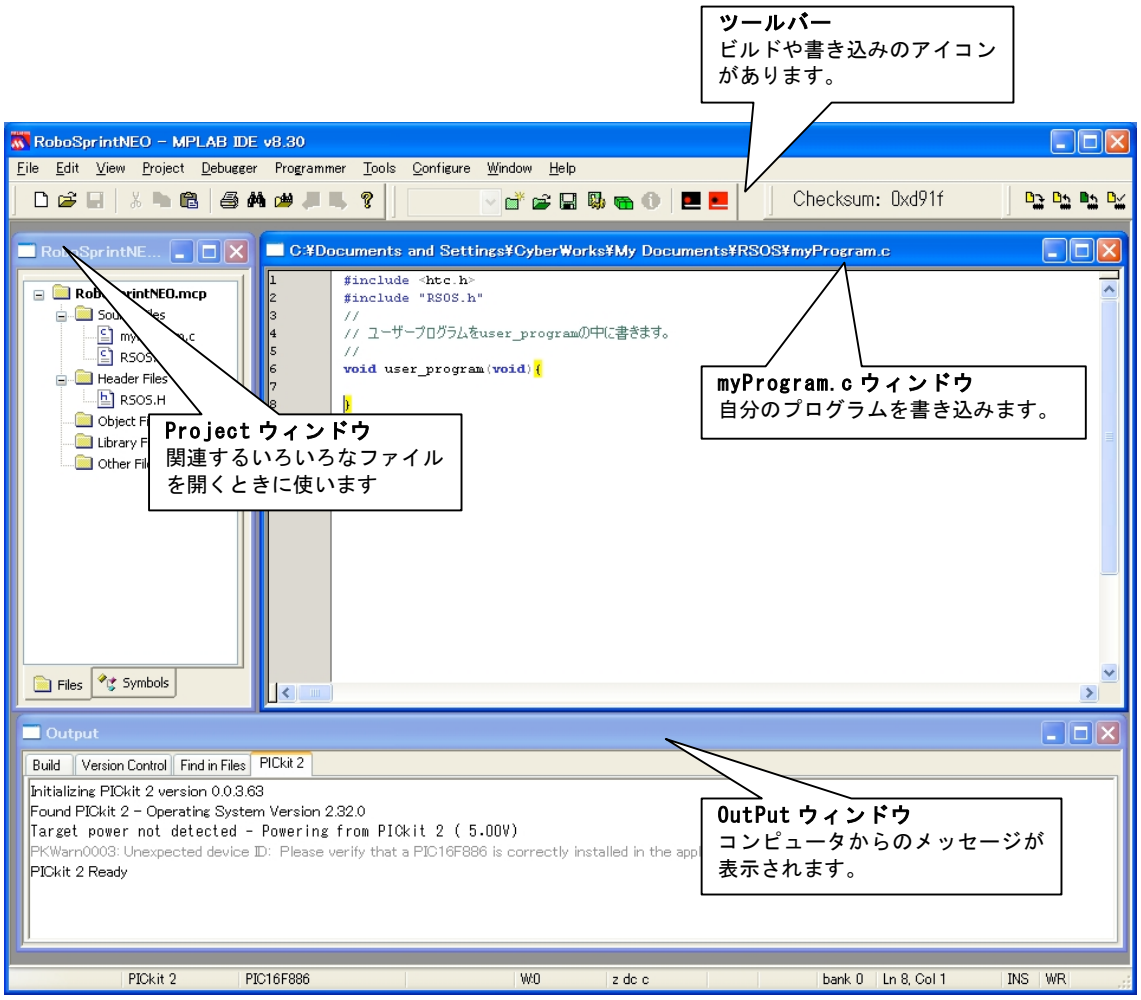
チュートリアル１	Ｃプログラムを書いて、RoboSprint に転送する方法を体験
チュートリアル２	繰り返し処理を使ったプログラムの作り方を体験
チュートリアル３	条件判断を使ったプログラムの作り方を体験

■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 と書き込みツールとして PicKit2（または PicKit3）を使います。C 言語は、同じく Microchip 社の HP からダウンロードできる、HI-TECH C 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、RSOS 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」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「プロジェクトフォルダ」を作る必要があります。元になる「RSOS フォルダ」が用意されているので、これをコピーして簡単に作ることが出来ます。

<u>最初に RoboSprint.zip をダウンロードなどで入手し、「マイ　ドキュメント」に展開してください。</u>

手順：

１・「マイ　ドキュメント」の「RoboSprint」フォルダを開く

２・「RSOS」フォルダを右クリックし、表示されたメニューから「コピー」を選ぶ

３・同じウィンドウ内の何もないところを右クリックし、メニューから「貼り付け」を選ぶ

４・「コピー～RSOS」というフォルダが出来るので、フォルダ名を右クリックして半角アルファベットで好みの名前をつける（このチュートリアルでは sample1 とします）

５・プロジェクトフォルダの完成

!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 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 IDE でプログラムを打ち込んでみます。MPLAB 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 IDE を起動します。
- File メニューから Open Workspace…を選択。ファイル選択画面で「マイ　ドキュメント」→「RoboSprint」→「sample1」→「RoboSprintNEO」を開く

!ポイント

ここで開いたファイルは「ワークスペース」という、ホームページの「お気に入り」のようなものです。前に作業した状態を再現して、すぐ続きが出来るよう、いろいろなファイルを設定してくれます。IDE ではこのようにワークスペースを開いて作業を始めるのが普通です。

- 一番手前になったウィンドウのタイトルバー（ウィンドウ上部）に myProgram.c という文字があることを確認

!ポイント

myProgram.c が開いていないときや誤って閉じてしまったときなどは、左上のウィンドウ（Project ウィンドウ）の myProgram.c のアイコンをダブルクリックすると開きます。

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。

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

 ←これがビルドアイコン。画面右上のツールバーにあります。

２・Output ウィンドウに文字が流れて、最後に下記のように表示されることを確認

********** Build successful! **********

！ポイント

********** Build failed! **********

このメッセージが出た場合はプログラムのどこかに書き方の誤りがあります。ビルド失敗です。

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。

・ Output ウィンドウの青色で表示されているエラーメッセージをダブルクリック（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、次の手順で接続します。

・ Pickit2 を USB ケーブルでパソコンに接続
・ IDE の Programmer メニューから Select Programmer を選び、Pickit2 を選択（チェックマークがついている状態にする）
・ 同じく Programmer メニューから Connect を選ぶ
・ Output ウィンドウの最後に「PICkit 2 Ready」と表示されることを確認

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。似たものがあるので注意！ 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 PicKit2 Ready と表示が出れば OK です。Pikkit2 を外してかまいません。

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、１０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、「もしーならば」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){
    for( ; ; ){  //無限ループ
        //センサーをチェックする
        if(Rsens == 1){ //センサー反応あり（床の上）
            //モーター回転にセット
            Rmotor=1;
            Lmotor=1;
        }
        else{ //センサー反応なし（持ち上げられている）
            //モーター停止にセット
            Rmotor=0;
            Lmotor=0;
        }
    }
}
```

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、「ーである」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。