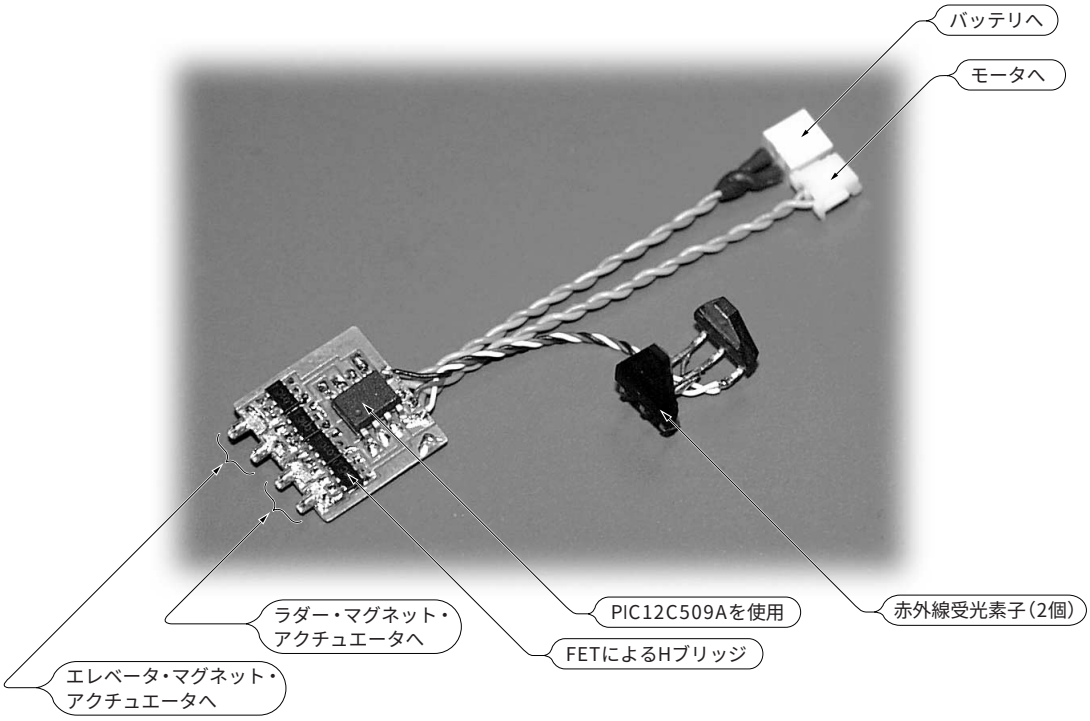
第7章

マグネット・アクチュエータ・ドライバを内蔵した赤外線受信機の製作

見本



マグネット・アクチュエータ・ドライバとスピード・コントローラを一つのPICに内蔵した3チャネル赤外線受信機を紹介します．マグネット・アクチュエータもスピード・コントローラと同様，PWM信号でコントロールします．

オリジナルのプログラムはイギリスのAndy Birkett氏によって，C言語で書かれたもので，PIC12C508A(509より少しメモリの容量が少ない品種)用です．

オリジナルのプログラムをそのままコンパイルしても日本メーカの送信機の信号タイミングにマッチしないことと，本書で紹介している受信機とPICの出力ピンの割り当てに違いがあるため，筆者が国内3社の送信機で使えるように，また本書で紹介しているほかの受信機と互換性が保てるようにプログラムの一部を変更しました．

7.1　C言語とCC5X Cコンパイラ

オリジナルのソースは，CC5X C Compilerを使ってコンパイルすることができます．CC5X C CompilerはノルウェのB Knudsen Data社のコンパイラで，CC5Xのフリー・エディションが提供されています．このバージョンを使って1Kワードまでのコードをコンパイルすることができるので，インドア・プレーンで使用する受信機のプログラムを組むのには十分です．

比較的頻繁にバージョン・アップされるので，最新版のCC5X C Compiler FreeをB Knudsen Data社のホーム・ページからダウンロードしてください．マニュアルもダウンロードすることができます．

```
http://www.bknd.com/cc5x/index.shtml
```

このコンパイラはMPLAB version 6.30 以降で，MPLAB統合環境の中で使用することができます．アセンブラのMPASMを使うのとまったく変わらない使い方で，Cコンパイラを使っていることを意識させません．

ダウンロードしたら解凍してください．解凍したファイルの中のINSTALL.TXTに使い方が書いてありますが難しいことはありません．これから説明するとおりに設定すれば使えるようになります．

CC5Xコンパイラは専用のインストーラがなく，適当なフォルダを作って，その中にすべてのファイルをコピーして使うようになっています．アンインストールするときは，作ったフォルダを削除するだけです．通常はC:¥Program Filesの下にcc5xとでもして，cc5xフォルダに解凍したすべてのファイルをコピーします．作業用のフォルダをC:¥Program Files¥cc5xの下にprojectsとでもして作っておくとよいでしょう．これから先の説明は上記の設定で進めて行きます．MPLABのインストールについては，付録Dの「初めてのPICプログラミング」で詳しく説明しています．

準備として，付属CD-ROMのled-test.cというC言語のソース・ファイルを先ほど作った

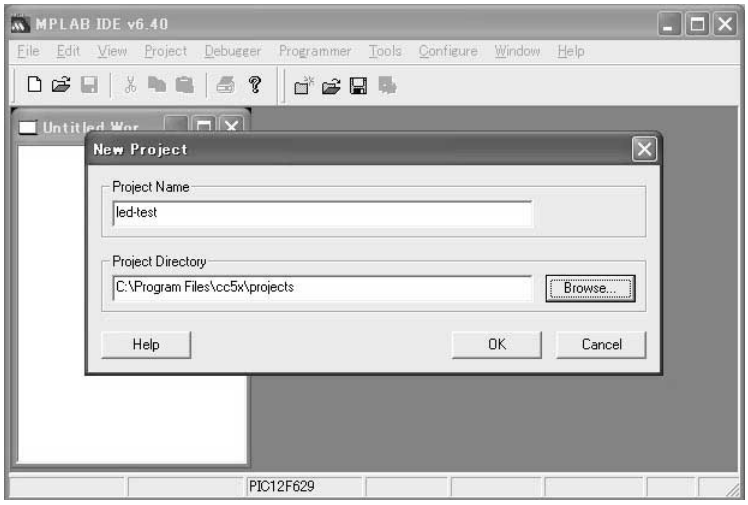


図7.1　プロジェクト名とプロジェクト・ディレクトリを指定

projectsフォルダにコピーしておいてください．このソース・ファイルは，付録Dの「初めてのPICプログラミング」で取り上げている，アセンブラで書かれた発光ダイオードの点滅プログラムをC言語で書きなおしたものです．興味のある人はソース・プログラムを比較してみてください．

● CC5X コンパイラをMPLABで使う

MPLABでCC5Xを使えるようにするための設定がいくつかあります．まずMPLABを立ち上げます．

図7.1は，led-testというプロジェクト名を新たに作って，C:¥Program Files¥cc5x¥projectsディレクトリで作業を行う設定をするところです．ディレクトリは[Browse]ボタンで設定します．設定が済んだら[OK]ボタンをクリックします．

次に，**図7.2**のように，[Configure]から[Select Device]をクリックして，ターゲット・デバイスを選択します(**図7.3**)．

デバイス選択がおわったら，[Configure]から[Configuration Bits]を選択してコンフィギュレーシ

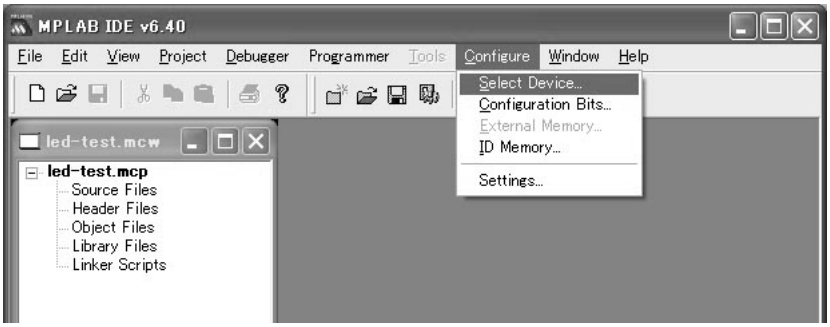


図7.2　デバイス選択へ

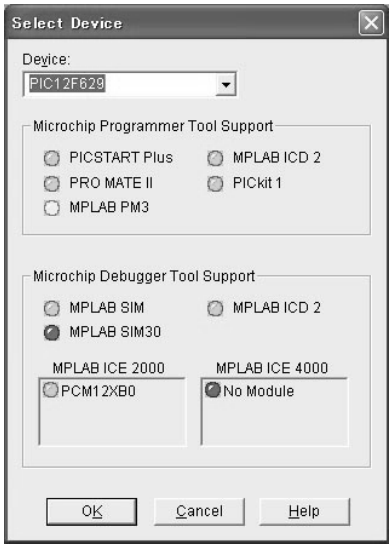


図7.3　ターゲット・デバイスの選択

ョン・ビットの設定を行いますが，今回コンパイルするled-test.cのソース・プログラムには，すでにコンフィギュレーション・ビットの設定を書き込んであるので，ここでの設定は省略できます．

ここで，ひとまずソース・ファイルの内容を説明しておきましょう．**リスト7.1**は，PIC12F629のためのプログラム・ソースled-test.cの全リストです．

#pragma config |= 0x2184がコンフィギュレーション・ビットの設定部分です．ここではパワー・アップ・タイマだけをONにしています．この部分はあとで説明します．

では次に，[Project]から[Select Language Toolsuite]を選択します(**図7.4**)．

図7.5に示すように，[Active Toolsuite]に[B Knudsen Data CC5X]を選択し，[Toolsuite

リスト7.1　LEDが0.5秒点灯、0.5秒消灯を繰り返すプログラム

```
#include "hexcodes.h"                  // インライン・アセンブラをMPASMと違和感
                                       // なく使えるようにするための
                                       // ヘッダ・ファイル
//#include "12f629.h"                  // MPLAB でコンパイルする場合は記述不要
#pragma config |= 0x2184               // コンフィギュレーション・ビットの設定

void delay(char millisec)              // ウェイト・ルーチン
{                                      // delay timeの調整はMPLAB SIMで行う
        uns8    i;                     // 8bit unsigned(MIN 0, MAX 255)
        do {
                TMR0 = 0;
                while (TMR0 < 127)
                        ;
        } while (-- millisec > 0);
        for (i = 0; i < 178; i++)
                ;
}
void main(void)                        // メイン
{
                                       // 初期設定
        #pragma update_RP 1            // Bank1に切り替え
        #asm                           // インライン・アセンブラを使って
                DW __CALL(0x3FF)       // メモリの最終番地をコール
        #endasm                        // インライン・アセンブラの終了
        OSCCAL = W;                    // OSCCALレジスタに内部発振周波数
                                       // 校正値をセット
        TRISIO = 0b.0000.0000;         // すべてのポートを出力に設定
        OPTION_REG = 0b.0000.0011;     // TMR0のプリスケーラを 1:16に設定
        CMCON = 0b.0000.0111;          // コンパレータ未使用
        #pragma update_RP 0            // Bank0に戻る
        GPIO = 0b.0000.0000;           // ゼロで初期化

        for(;;) {                      // メイン・ループ
                GPIO4 = 1;             // GP4をHighにしてLEDを点灯
                delay(500);            // 499.992ms
                nop2();                // cc5xではgoto $+1を生成する
                nop2();                // nop();を二つ記述するのと同じ
                nop2();                // ここまでで0.5秒経過
                GPIO4 = 0;             // GP0をLowにしてLEDを消灯
                delay(500);            // 499.992ms
                nop2();                // 時間調整
                nop2();                // 時間調整
        }
}
```

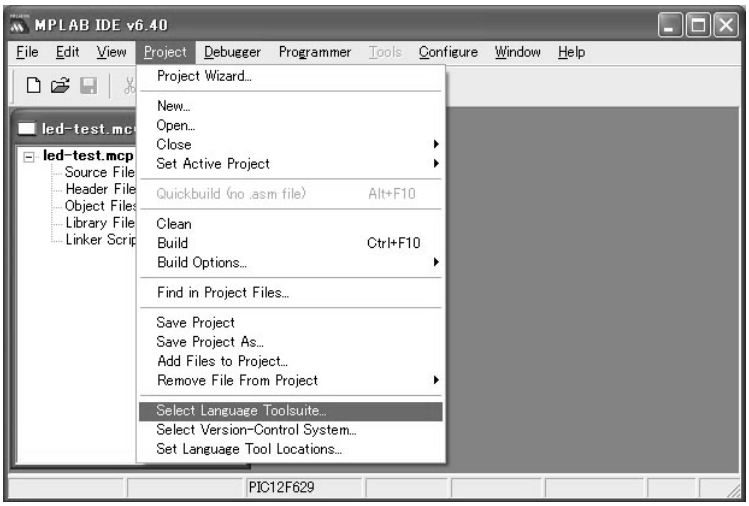


図7.4　[Select Language Toolsuite]選択画面へ

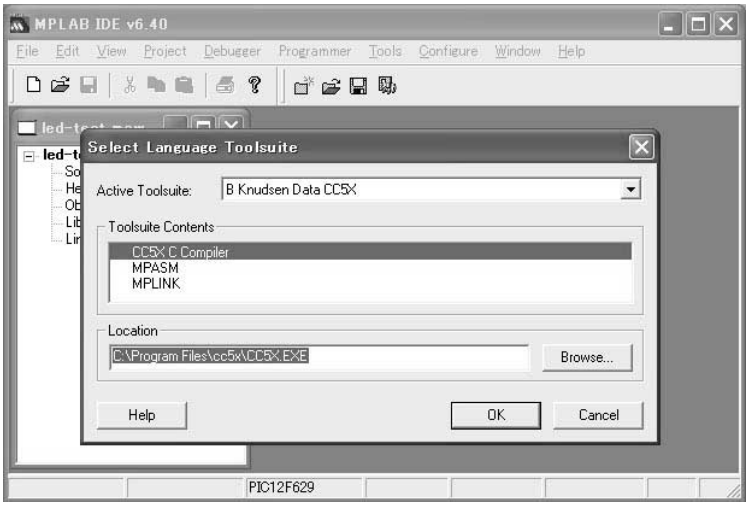


図7.5　CC5X C Compilerを設定

Contents]ではCC5X C Compilerを選択します．[Location]に`C:¥Program Files¥cc5x.EXE`と表示されるので，[OK]ボタンをクリックします．

次に，インクルード・ファイルのパスを設定します．[Project]から[Build Options]で[Project]を選択します．

図7.6のように，[Include Path, $(INCDIR)]の右にある[Browse]ボタンでインクルード・ファイルが置かれているCC5Xフォルダを選択します．ところがこのパスにはスペースが含まれていて，このままの設定ではパスが通りません．**図7.7**のように，スペースの前後をダブル・クォートでくくってあげる必要があります．

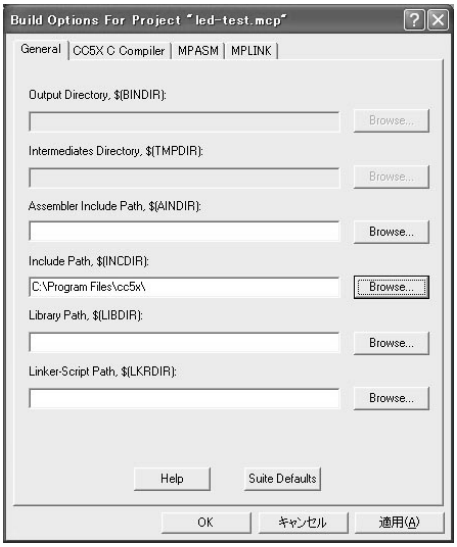


図7.6　ビルド・オプション設定画面

図7.7　パスのスペースをダブル・クォートでくくる

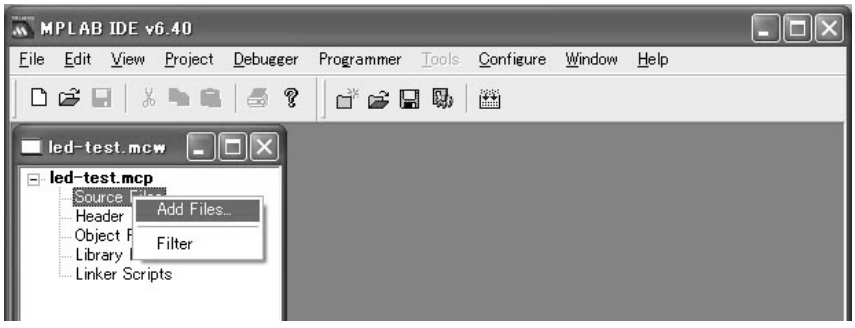


図7.8　プロジェクト・ファイルにソース・ファイルを追加

[適用]ボタンをクリックしてから[OK]ボタンをクリックして，ウィンドウを閉じます．

これでいつでもCC5XコンパイラをMPLABで使うことができます．早速`led-test.c`のソース・プログラムを読み込んで，コンパイルしてみましょう．

ここからの手順は，アセンブラのソース・プログラムを読み込むときと同じです(**図7.8**)．

C言語で書かれたソース・プログラムを表示したものを**図7.9**に示します．[Project]から[Build]を選