International
IOR Rectifier

APPLICATION NOTE

アプリケーション・ノート：AN-1110

モーター制御IC「IRMCF/K300」シリーズに内蔵したプロセッサ（8051）のアプリケーション開発

Ali Husain　International Rectifier

目次

頁

はじめに.....2
- 目的.....2
- 条件.....2

作業開始のガイド.....3
- ソフトウエアの設定.....3
- ハードウエアの設定.....6
- ハードウエアとソフトウエアの始動.....6
- サンプル・コードの変更.....7

サンプル・コードの構造.....8
- クロック周波数.....8
- レジスタ.....8
- サンプル・コードのファイルと関数.....10

モーターの制御.....13
- ドライブの設定.....13
- MCE のヘッダー・ファイル.....13
- モーター関数.....13

高度な構成.....14
- 8051 コードの EEPROM へのダウンロード.....15
- 元の 8051 コードへの回復.....16
- トラブル・シューティング.....16

付録：Keil uVision2 のプロジェクト・オプション.....18

©インターナショナル・レクティファイアー・ジャパン
この文献の無断複製・転載を禁じます。

はじめに

モーター制御 IC「IRMCF/K300」シリーズに内蔵されているマイクロプロセッサ（8051）は、さまざまな制御機能と保護機能を実現するために使うことができます。IRMCF/K300 シリーズに内蔵した 8051 の命令セットと基本動作は、Intel 社の標準プロセッサである 8051 に準拠しています。ただし、動作をモーター制御向けに専用化するために、多くの周辺デバイスと特別な機能が追加されています。

IRMCF/K300 シリーズの IC は、8051 と Motion Control Engine （MCE）の 2 つのプロセッサを搭載しています。8051 と MCE は、両方のプロセッサからアクセスできる共用 RAM を介して通信します。MCE は、モーター制御ループの構築、帰還信号の処理、PWM（パルス幅変調）スイッチング信号の計算を行うために特別に設計されています。8051 は、外部制御信号（例えば、洗濯機のフロント・パネルからの信号）と MCE（最終的にモーターを動かす信号を発生）との間で仲裁役を果たします。

8051 のアプリケーション・ソフトウエアは、レジスタ・インタフェースの読み出し／書き込みを通じて MCE の動作を制御／監視します。MCEDesigner ツールと組み合わせて使う 8051 の開発用アプリケーションが、簡単なロック・ステップ（同時並行）法でこの作業を行います。すなわち、MCEDesigner が個別に読み出し／書き込みを行う各レジスタを指定し、要求に応じて 8051 のソフトウエアだけがこれらの動作を実行します。一方、8051 のユーザー・アプリケーションは一般に、動作のシーケンス全体を自動的に、あるいは“start”や“stop”などの簡単な入力コマンドに対応して、実行します。なお、ここでは、Windows 英語版を例に説明しています。

目的

このアプリケーション・ノートでは、IRMCF300 シリーズと IRMCK300 シリーズのモーター制御 IC に内蔵した 8051 プロセッサの制御方法を説明します。すなわち、8051 の制御に必要とされる初期化、設定、機能について説明します。いくつかの例とサンプル・コードも記載してあります。なお、このアプリケーション・ノートでは、ユーザーは、組み込みソフトウエアのプログラミングについて経験を持っているものと想定しています。

ここに記載する例やサンプル・コードは、アプリケーション開発が完了したときに、MCEDesigner を置き換えるための制御インタフェースを生成できるようにすることを目的としています。主要なタスクの 1 つは、8051 のコード内で MCEDesigner の機能を再現することです。コードを開発して、IRMCF 版の IC を使ってテストした後、8051 のコードは IRMCK300 シリーズの制御 IC の ROM に書き込まれます。

条件

8051 のアプリケーション・コードの開発には、以下のソフトウエアとハードウエアが必要です。

1. Keil uVision ドライバ（FS2 デバッグ・ポッド）付きの FS2 ISA-M8051EW デバッガ。
2. Keil PK51 Professional Developers Kit（Keil uVision2）。

Keil uVision2 を使うと、C 言語で制御プログラムを書いて、マシン・コードにコンパイルし、IRMCF300 シリーズの IC にダウンロードしてテストすることができます。uVision2 はシミュレーション・モードを備えているため、ハードウエアに依存しないプログラム部分を実

際の IC にダウンロードすることなくテストできます。ここに記載するソース・コードの例は、Keil uVision2 を使って開発しました。

8051 のソフトウエア開発に使った Keil コンパイラは、16 ビットや 32 ビットの値をメモリーに格納するときに、ビッグ・エンディアン・バイト・オーダーを使うコードを生成します。MCE は 16 ビットのプロセッサであり、データの格納にリトル・エンディアン・バイト・オーダーを使います。MCE のデータ格納最小単位は、16 ビットです（メモリー内で 1 バイトをアクセスすることはできません）。8051 プロセッサと MCE プロセッサとの間の情報交換のために使う共用 RAM は、8051 に対して 8 ビット・アドレス指定可能ですが、MCE に対しては 16 ビットのアドレス指定となります。共用 RAM に対する正しい読み出し／書き込みの関数は、サンプル・コードに含まれています。これらの関数は、必要に応じてバイトをスワップし、バスへのアクセスをロックして、一方のプロセッサの書き込み中に、他方のプロセッサがそのデータを読み出さないように防止しています。

IRMCF3xx のモーター制御 IC を実装した評価基板は、該当するコネクタ、ドライバ、FS2 のハードウエアとインタフェースするためのアイソレータを備えています。IR 社が用意した評価基板「IRMCS3041」には、この回路が内蔵されており、FS2 のハードウエアと接続するときの適切なアイソレーションも付いていますが、他のハードウエアを使う場合には、下記の警告に従ってください。



警告 ！

FS2 デバッグ・ポッドを回路基板に接続するときに、適切なアイソレーションを使用しないと、基板上の高電圧によって FS2 のハードウエアが損傷することがあります。この問題は、DC バスの負側（GND）の電位が接地（アース）の電位と一致しないために発生します。従って、適切なアイソレーションを使用しない場合には、FS2 のハードウエアを使うときに、絶縁された直流電源から基板に電源を供給してください。

作業開始のガイド

この節では、FS2 デバッグ・ポッド、Keil uVision ツール、IRMCF300 シリーズの IC を使った 8051 のアプリケーション・ソフトウエア開発の始め方を説明します。このガイドは一般的なハードウエア構成に適用できますが、特別な例として、評価基板 IRMCS3041 を参照しています。

ソフトウエアの設定

Keil uVision2:

1. Keil uVision2 を起動します。Project ⇒ Open Project を選択し、サンプル・コードに付属しているファイル IRSamples.Uv2 を開きます。“Project Workspace”ウインドウ画面で、最上部のフォルダ IRSamples をクリックします。
2. Project ⇒ Options for Target‘IRSamples’を選択します。Debug タブを選択します。

3. ラジオ・ボタン“Use”をクリックし、右側のフィールドを図 1 のように Fs2/Keil ISA-M8051EW Driver と設定します。このオプションがリストに表示されない場合は、FS2 のソフトウエアが正常にインストールされていません。
4. Setting をクリックして、Settings が TckRate: 62500 と Tvcc Threshold: 2500 になっていることを確認します。“Options for Target‘IRSamples’”の他のすべての設定は自動的に設定されます。このアプリケーション・ノートの付録に、設定が必要なすべてのオプションのリストが記載してあります。
5. Settings ウインドウで“OK”をクリックし、次に Options ウインドウで“OK”をクリックします

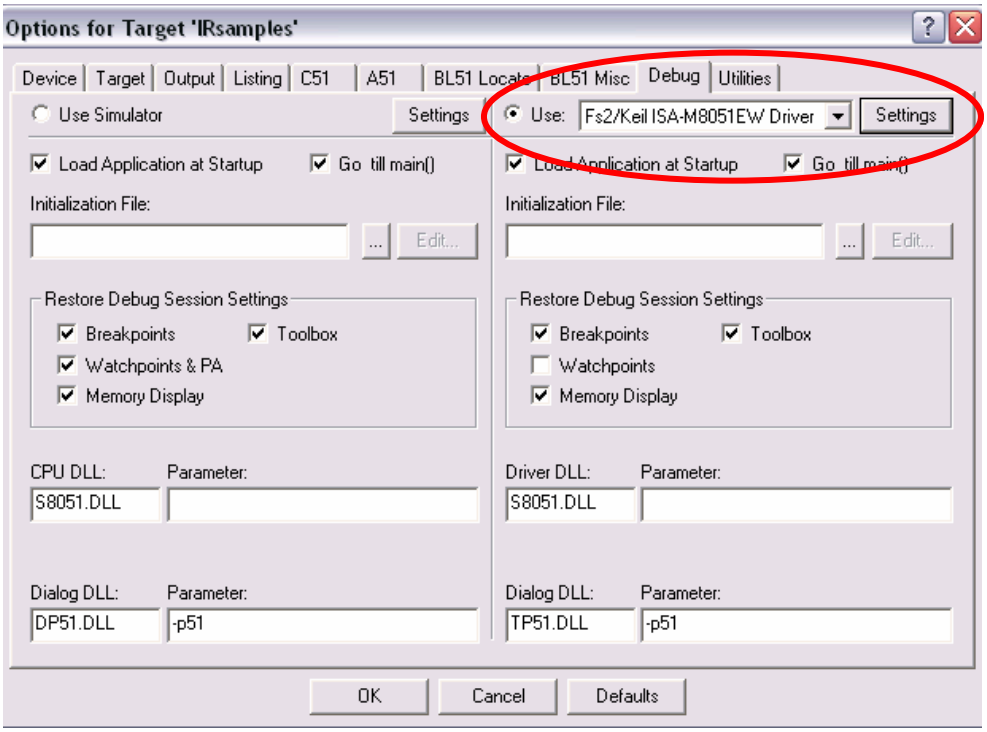


図 1　uVision2 project の Debug Options ウインドウ画面

HyperTerminal（ハイパーターミナル）：

1. HyperTerminal を開きます（Windows の Start（スタート） ⇒ Programs（プログラム、またはすべてのプログラム） ⇒ Accessories（アクセサリ） ⇒ Communications（通信） ⇒ HyperTerminal（ハイパーターミナル）を選択します）。
2. New Connection（新しい接続）ウインドウ画面が自動的に表示されない場合は、File（ファイル） ⇒ New Connection（新しい接続）を選択します。ユーザー接続の名前とアイコンを選択して、“OK”をクリックします。
3. “Connect Using（接続方法）"で、該当する COM ポートを選択します。これは、MCEDesigner が制御基板と交信する際に使うポートと同じかもしれません。“OK”をクリックすると、Properties（プロパティ）ウインドウが開きます。“Bits per Second（ビット／秒）”フィールドに 57600 を、“Flow control（フロー制御）”に None（なし）をそれぞれ設定して、残りの設定が図 2 のようになっていることを確認します。

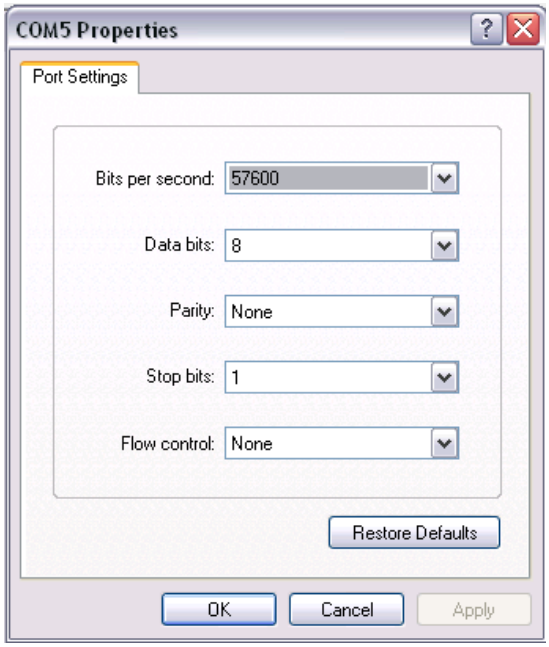


図 2　HyperTerminal（ハイパーターミナル）の接続プロパティのウインドウ画面

ハードウエアの設定

Keil uVision2 と MCEDesigner がインストールされているコンピュータに UART インタフェースを接続します。つまり、IRMCS3041 上で、制御基板からパソコンへ RS-232 （シリアル）ケーブルを接続します。さらに、FS2 デバッグ・ポッドを制御基板に接続します。IRMCS3041 上で、FS2 ポッドが図 3 に示すコネクタ J11 を使って制御 IC へインタフェースします。IRMCS3041 を使わない場合には、下記の警告に従ってください。

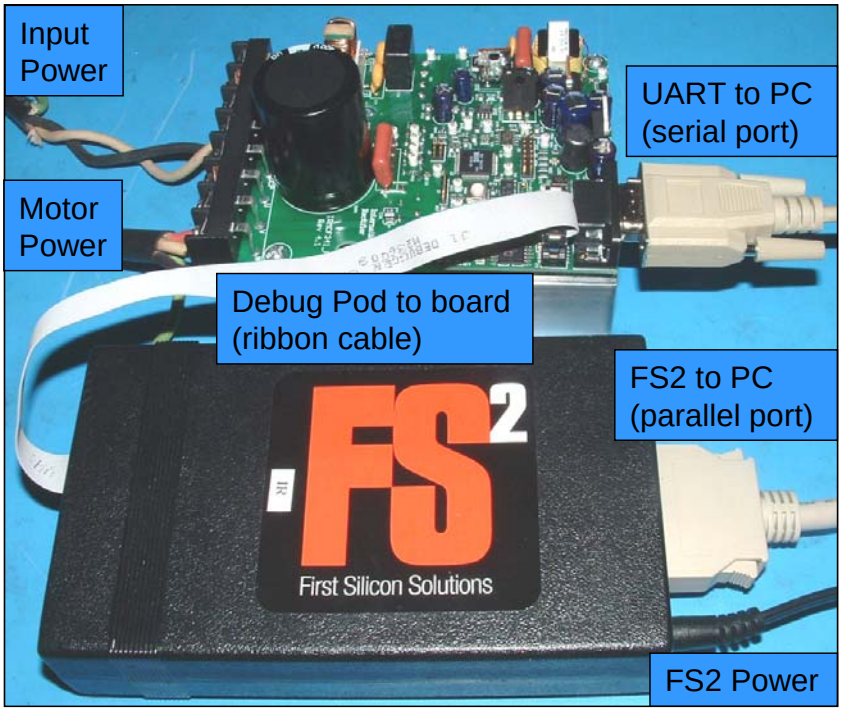


図 3　評価基板 IRMCS3041 と接続した FS2 デバッグ・ポッド



警告 ！

FS2 デバッグ・ポッドを回路基板に接続するとき、適切なアイソレーションを使用しないと、基板の高電圧により FS2 のハードウエアが損傷することがあります。この問題は、DC バスの負側（GND）の電位が接地（アース）の電位と一致しないために発生します。従って、適切なアイソレーションを使用しない場合には、FS2 のハードウエアを使うときに、絶縁された直流電源から基板に電源を供給してください。

ハードウエアとソフトウエアの起動

基板とソフトウエアの適切な起動は、以下の順序に従ってください。

1. 電源を制御基板に接続して、FS2 ポッドをオンします。
2. Keil uVision2 を起動します。Project ⇒ Open Project を選択し、IRSamples.Uv2 を開きます。

3. Debug ⇒ Start/Stop Debug Session を選択します。FS2 装置が立ち上がった後に、左下のステータス・バーに 8051 コードの RAM へのローディングの進捗状況が表示されます（図 4）。

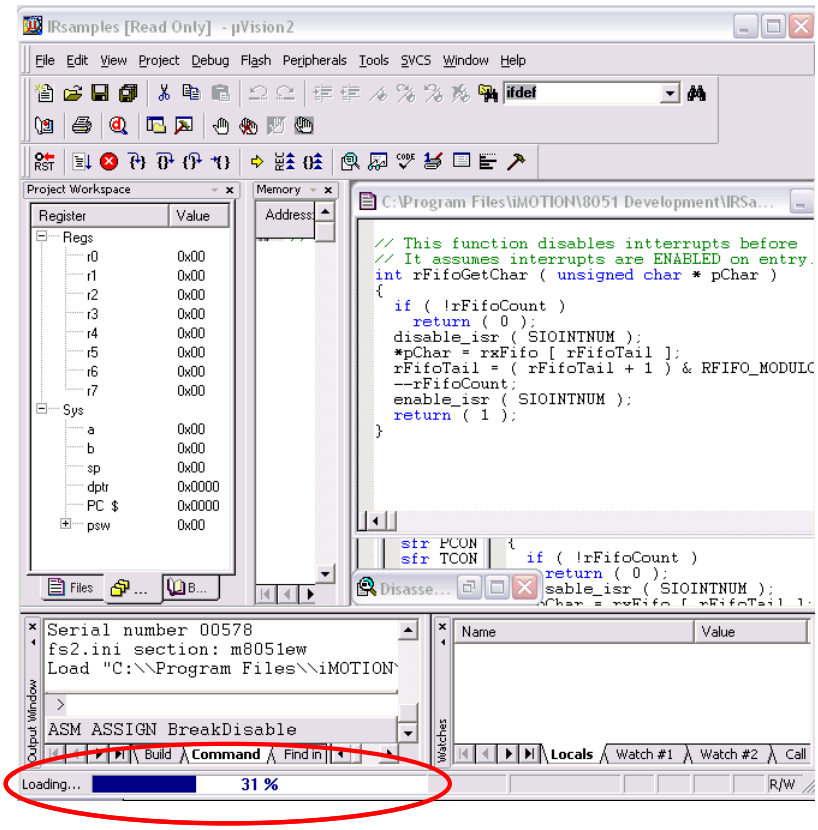


図 4　8051 のプログラムをローディングしているときの uVision ウインドウ画面

4. Debug ⇒ Go を選択します。
5. HyperTerminal（ハイパーターミナル）を起動して、上記の接続設定を開きます。モーターは、HyperTerminal（ハイパーターミナル）から送ったコマンドで制御できます（コマンドについては p.10 の「サンプル・コードのファイルと関数」の節を参照してください）。
6. テスト後に Keil を終了するときは、Debug ⇒ Stop を選択します。次に、Debug ⇒ Start/Stop Debug Session を選択します。プログラムはこの時点で停止します。

サンプル・コードの変更

1. サンプル・コードを変更するときは、サンプル・コードのすべてのファイルを新しいディレクトリにコピーします。
2. すべてのファイルを選択して、右クリックにより“Properties”を選択します。“Read-only”の選択を解除して、“OK”をクリックします。
3. 新しい project を開いて、必要に応じてファイルを変更します。Files ⇒ Save All を選択して、ファイルを保存します。
4. マシン・コードを再構築するときは、Project ⇒ Rebuild all target files を選択します。コンパイラは、新しい.hex ファイルを生成します。

5. 前項の「ハードウエアとソフトウエアの起動」の節の説明に従い、変更したプログラムをハードウエアでテストします。プログラムの前のレビジョンをテストした後、ハードウエアにまだ電源が入っている場合、ステップ 3 を開始できます。再ダウンロードする前に、ハードウエアの電源をいったん切って再投入する必要はありません。

サンプル・コードの構造

8051 に組み込まれたアプリケーションが実行する main 関数は、適切な駆動パラメータを使って MCE を設定し、モーターの起動、停止、速度調節を行います。これらは MCEDesigner が実行するのと同じ関数です。ホスト・アプリケーション（MCEDesigner）から 8051 アプリケーションへ“頭脳”が移動していることだけが違っています。タイミング遅延を含むすべての MCEDesigner 関数（既定の一連のレジスタ動作）が 8051 アプリケーション内で直接実行できるため、モーターを独立に制御できます。または簡単な外部コマンドで制御できます。

8051 は、インタフェース・レジスタの読み出し／書き込みを行うことによって、MCE を制御／監視します。レジスタについては次節で説明します。8051 の組み込みアプリケーションが実行しなければならない一般的なステップを次に示します。

1. ハードウエアの初期化：クロック周波数の設定、カウンタやタイマなどの初期化。
2. MCE の起動：有効な MCE プログラムがロードされていることの確認と、MCE プログラム・カウンタの初期化。
3. モーター駆動の設定：所望のモーター駆動パラメータの MCE レジスタへの書き込み。
4. 起動、停止、回転方向変更、速度変更：コマンドを正しく実行するために現在の状態を記録（例えば、モーターの回転中は向きを変更しないなど）。
5. 故障の監視：周期的割り込みにより、外部コマンドの処理、ウォッチドッグ・タイマのリセット、故障／誤りの検査、駆動回路のシャットダウン（必要に応じて）を行います。

クロック周波数

MCE コードは SFR（スペシャル・ファンクション・レジスタ）の PLLF0 と PLLF1 に書き込みを行うことで SYSCLK 周波数を発生させ、PLL（位相同期ループ）を設定しなければなりません。サンプル・コードには、クロック周波数を 32MHz、50MHz、64MHz、128 MHz の中から選択する命令が含まれています。timer.h の該当する#define 文のコメント化を外すだけで済みます。

クロック周波数に基づいて、コードが適切なボー・レート、タイマ初期値、リセット値を設定しなければなりません。クロック周波数が timer.h 内で選択されると、これらも正しく設定されます。ただし、駆動パラメータによっては、クロック周波数にも依存するものがあります。ユーザーは、新しいクロック周波数を使って“Parameter Configurator”から駆動パラメータを再生成する必要があります。p.13 の「ドライブの設定」の節を参照してください。

レジスタ

以下に示すようにさまざまな種類のレジスタがあります。

1. スペシャル・ファンクション・レジスタ（SFR）：SFR は 8051 からだけアクセスできます。このアプリケーション・ノートでは SFR のサブセットのみ説明します。SFR 全

体の一覧と説明は、「IRMCx3xx ユーザーズ・ガイド」に記載してあります。SFR は以下の目的に使うことができます。

- プロセッサ・レジスタの初期化と変更。
- I/O ポートの設定と読み出し。
- クロック周波数の設定。
- タイマの設定、初期化、リセット。
- UART の設定。
- オペアンプのようなアナログ機能のイネーブル。
- 低電力モードの開始と終了。
- 割り込みの設定とイネーブル。
- 故障とステータスの読み出し。
- I^2C/SPIシリアル・インタフェースの設定と使用。
- 固定の MCE レジスタの読み出しと書き込み。
- ユーザー定義の MCE レジスタの読み出しと書き込み。

SFR は 8051 からのみアクセス可能なので、これらのレジスタへの書き込みは比較的簡単です。これらは、irmcx3xx.h の中で、特別な“sfr”キーワードを使って定義されています。各 SFR には、メモリー・アドレスに対応する名前が与えられています。サンプル・コードでは、SFR 名はすべて大文字で表わす表記法が使われています。これらのレジスタは、他の変数と同様に名前を使って読み書きされます。SFR 割り当ての以下の例では、クロック周波数を 50MHz に設定しています。

```
PLLF0 = 0x62;   // set clock speed to 50 MHz
PLLF1 = 0xC0;
PLLF2 = 1;      // switch to PLL clock
PLLF3 = 0;
```

2. 固定 MCE レジスタ（FREG）：FREG には、MCE と 8051 の両方からアクセスできます。MCE は Motion Peripheral Blocks を経由して FREG にアクセスし、各 PWM 周期でこれらのサブセットを変更できます。8051 は、専用 SFR のセットを介して FREG にアクセスします。大部分の FREG は、モーターを動作させる前に 1 回だけ設定する必要があり、8051 のアプリケーションから実行できます。さらに、8051 のアプリケーションは FREG へ書き込みを行って、モーターの起動と停止、または動作条件の変動によるパラメータの調整を行うことができます。

   各 FREG は、“FREG_”で始まり、メモリー・オフセットに対応する別名を使って regIf.h 内に定義されます。サンプル・コード内の関数 DoRtlRd と DoRtlWr は、専用 SFR を介してこれらのレジスタに読み書きするために用意されています。これらの関数の特別なシーケンス動作は、正しい動作のために不可欠であるため、変更せずにそのまま使ってください。下の例では、数値“0”を固定の MCE レジスタ pwmctrl_1 へ書き込んでいます。

   ```
   DoRtlWr ( FREG_pwmctrl_1, 0 );
   ```

3. ユーザー定義の MCE レジスタ（RAM_REG）：これらのレジスタも、両プロセッサからアクセスできます。ただし、FREG とは対照的に、RAM_REG はメモリー内で固定されません。RAM_REG は MCE Simulink 設計内で定義され、MCE コンパイラによっ

て RAM アドレスが割り当てられます。コンパイラは、8051 コード内に組み込まれるヘッダー・ファイルを出力します（p.13 の「ドライブの設定」の節で説明）。レジスタに割り当てられる名前は、次の形式になっています。

<Simulink Sub-model>_<register name in Simulink>

RAM_REG は共用 RAM 内で 8051 から直接アドレス指定可能ですが、別の専用セットの SFR を使ってデータの破壊を防止する方法でアクセスされます（MCE は 1 回の 16 ビット動作としてレジスタをアクセスしますが、8051 は 8 ビット動作を 2 回行うため、このようなことが必要になります）。サンプル・コード内の関数 DoRamWr と DoRamRd は、専用 SFR を介して RAM_REG を読み書きするために用意されています。モーター1 の DC バスの過電圧レベル（Motor1_DcBusOvLevel ）を 172 に設定する例を下に示します。

```
DoRamWr ( Motor1_DcBusOvLevel, 172 );
```

注）MCEDesigner では、Motor1 ウインドウ画面の右側にある“Access”列を見ると、FREG と RAM_REG を区別できます。この列を表示するときは、Motor 1 ウインドウ画面の左側で“Register Structure Definitions”をクリックします。“Access”列に、Write または Read が表示され、その後ろに次に示すオプションの特別な文字が続きます。

* (asterisk)	Fixed MCE Register (FREG)
No character	User-defined MCE Register (RAM_REG)
X	Obsolete Register
+	Local 8051 register. These registers are currently unused.

サンプル・コードのファイルと関数

サンプル・コードは複数の C ソース・ファイルから構成されており、使い方に応じて機能がグループ分けされています。次の.c ファイルがあります。

1. main.c：ここから実行が始まります。main 関数が各サンプルをコールします。最後のサンプル関数 MotorCtrl はリターンしません。

2. regif.c：このファイルには、8051 の SFR を使うときに一貫性を維持して、共用 RAM 内の 16 ビット・レジスタの読み出し／書き込みを行う関数が含まれています。FREG インタフェースの低層レベルの構成については RtlRegs.SRC を、RAM_REG インタフェースの構成については Coherent.SRC を参照してください。レジスタ・インタフェース関数のコールの例は、MotorCtrl.c に記載してあります。

3. EepromI2C.c：このファイルには、I^2Cインタフェースを使ってEEPROMに読み出し／書き込みをするサンプル・コードが含まれています。

4. Timer.c：このファイルには、2 msec の間隔で割り込みを発生するように、タイマ 1 を初期化する関数が含まれています。各割り込みごとに、グローバル変数"systicks"がインクリメントされ、FREG_FaultFlags レジスタをチェックして故障状態を調べます。また、割り込みサービス・ルーチンもウォッチドッグ・タイマをリセットします。ウォ

ッチドッグ・タイマは周期的にリセットしてください。そうしないと、IC 全体がリセットされます。タイマの設定は、timer.h 内で設定されるクロック周波数に応じて変化します。

5. MceBoot.：このファイルには、MCEDesigner ツールにより EEPROM に書き込まれたコードを使って、MCE を初期化する関数が含まれています。自動ブート・プロセスが MCE コードを EEPROM から共用 RAM へ、さらに"MCE Info"構造を EEPROM から 8051 のプログラム RAM 内の固定ロケーションへ、それぞれコピーします。

   関数 StartMce は、まず"MCE Info"構造を 8051 のプログラム RAM からデータ RAM 内のロケーションへコピーし、構造の有効フィールドを確認します。有効フィールドが正しくない場合、構造全体を無効と見なし、MCE は初期化されません。その他の場合は、MCE Info 構造が RAM 内のロード開始アドレスと MCE 実行アドレスを提供します。StartMce 関数はこの情報を使って、MCE プログラムを起動する前に MCE データ領域をゼロにします。関数 doMceBoot をコールして、MCE スペシャル・レジスタを初期化し、MCE の実行を開始します。

6. asyncDriver.c：このファイルには、UART を設定して FIFO（first-in-first-out）バッファを使ってデータを読み書きする関数が含まれています。2 個の UART をサポートするデバイスの場合、USE_UART0 を定義する 20 行目をコメントにして UART1 用にコードをコンパイルすることができます。このファイルには、次の関数が含まれています。

   sioIsr：送受信の割り込み処理をする UART 割り込みサービス・ルーチンです。受信した文字を受信 FIFO に格納します。送信する文字を送信 FIFO から取り出します。

   sioInit：この関数は、送受信のデータ構造、UART を制御する SFR を初期化します。

   flushTx：送信 FIFO を初期化します。

   flushRx：受信 FIFO を初期化します。

   setBaudRate：デフォルトのクロック周波数 64 MHz に基づいて SFR のボー・レートを 57,600 bps に初期化します。

   putChar_：この関数は、文字を送信するために高層レベル（MotorCtrl 関数など）からコールされます。トランスミッタがビジーの場合、文字を送信 FIFO に追加します。転送中でない場合には、UART 送信バッファに文字を直接書き込みます。送信 FIFO がフルの場合（送信するために文字を受け取ることができない場合）、この関数は 0 を返します。正常終了の場合は 1 を返します。

   getChar_：この関数は、受信 FIFO から受信した文字を読み出すときに高層レベルからコールされます。受信 FIFO がエンプティ（文字がない）の場合、0 を返します。正常終了の場合は 1 を返します。

xFifoRoom：送信 FIFO の状態をチェックするときに putChar_からコールされます。送信 FIFO がフルのときは 0 を返します。その他のときは 1 を返します。

xFifoPutChar：文字を送信 FIFO へ追加するときに putChar_からコールされます。送信 FIFO がフルのときは 0 を返します。FIFO に文字を正常に追加できたときは 1 を返します。

xFifoGetChar：送信 FIFO から最も古い文字を取り出すときに sioIsr からコールされます。送信 FIFO がエンプティのときは 0 を返します。FIFO から文字を正常に取り出したときは 1 を返します。

rFifoRoom：受信 FIFO の状態をチェックするときに sioIsr からコールされます。FIFO がフルのときは 0 を返します。受信した文字を正常に FIFO に追加できたときは 1 を返します。

rFifoPutChar：受信 FIFO に文字を追加する際に sioIsr からコールされます。受信 FIFO がフルのときは 0 を返します。文字を正常に FIFO に追加できたときは 1 を返します。

rFifoGetChar：受信 FIFO から最も古い文字を取り出すときに getChar_からコールされます。受信 FIFO がエンプティのときは 0 を返します。文字を正常に FIFO から取り出したときは 1 を返します。

重要な注意：送信 FIFO と受信 FIFO は、割り込みレベルと"タスク"（非割り込み）レベルの両方から操作されます。このため、タスク・レベルで両 FIFO に文字を読み出し／書き込みをするときに、UART の割り込みをディセーブルにしておくことが非常に重要です。

7. MotorCtrl.c：このファイルには、モーター駆動の構築と制御の簡単な例が含まれています。UART ドライバから提供される関数を使ってシリアル・ポートから文字コマンドを読み出します。コマンドを送信し、応答を読み出すためには、HyperTerminal（ハイパーターミナル）または同等の接続を使うことができます。サポートされているコマンドの一覧とその説明は、次頁の「モーターの制御」の節に記載してあります。

   関数 MotorCtrl は、モーター制御動作を可能にする前に、regif.c 内に定義されていて、EEPROM からロードされた MCE バージョンが一致していることを確認します。詳細については、次頁の「MCE ヘッダー・ファイル」の節を参照してください。

8. RtlRegs.SRC：SFR インタフェースを介して RTL レジスタを読み書きするアセンブリ言語関数です。

9. Coherent.SRC：データ転送の一貫性を維持するため、SFR レジスタを使って共用 RAM レジスタを読み書きするアセンブリ言語関数です。

10. utils.c：ファイルの先頭で定義された割り込み番号によって指定される特定の割り込みをイネーブル/ディセーブルするユーティリティ関数です。

モーターの制御

ドライブの設定

電源投入後、正しいパラメータ設定値によって MCE が設定されるまで、モーターは正常に動作しません。これらの駆動パラメータ値は、“Parameter Configurator”（Excel のスプレッドシート）を使って生成します。スプレッドシートの 2 番目のタブには、MCE レジスタ用に正確にスケールされた値が含まれています。これらの値で、MotorCtrl.h 内で定義されたサンプル値を置き換えてください。この処理は、このシートをテキスト形式でエキスポートし、各行の先頭に“#define”を追加することにより、ある程度自動化できます。

MCE ヘッダー・ファイル

Simulink モデル・ファイルをコンパイルすると、MCE コードが生成されます。コンパイラはヘッダー・ファイル（.h）も生成します。このファイルには、1）ユーザー定義の MCE レジスタの定義、2）ユーザー定義の MCE レジスタをアドレス指定するときのレジスタ・マップ構造、3）製品やバージョン情報が含まれています。このコードで、regIf.c と regIf.h の先頭にあるサンプルを置き換えてください。regIf.h では、“COMPILER GENERATED DEFINITIONS”のセクションを、MCE コンパイラ・ヘッダー・ファイルの対応するセクションに置き換えます。同様に regIf.c では、“COMPILER GENERATED INITIALIZATIONS”セクションをヘッダー・ファイル内の対応するコードに置き換えます。

正しい動作のためには、ヘッダー・ファイルの MCE の設計 ID とバージョン番号が、EEPROM からロードされた MCE コードと一致しなければならないことに注意してください。Simulink モデルが変更された場合は、コンパイル時に新しいヘッダー・ファイルが生成されます。前節で説明したように、それをコードに追加してください。

サンプル・コードは、評価基板 IRMCS3041 の RAM_REG を正しく書き込むように設定されており、Golden Age GK6040-6AC31 モーターを動作させるための適切な駆動パラメータを備えています。

モーター関数

サンプル・コードでは、モーターを DRIVE_IDLE、DRIVE_RUN、DRIVE_FAULT の 3 つの状態を備えるステート・マシンとして扱います。関数 MotorCtrl は、シリアル・ポートから入力コマンドを受け取り、有効なコマンドを MotorSeq へ渡します。現在のモーターの状態に基づいて、MotorSeq は該当する関数をコールして、コマンドを実行するか、エラーを返してコマンドが無効であることを表示します。無効なコマンドが入力されたとき、X が HyperTerminal（ハイパーターミナル）ディスプレイに返されます。関数 MotorCtrl がサポートしているコマンドの一覧と動作説明を次に示します。

C または c

モーター駆動を設定し、Fault をクリアします。設定が完了すると、シリアル・ポートを使ってコマンド文字がホスト・コンピュータにエコーバックされます。モーターが動作中の場合は、コマンドが無視され、コマンド文字の代わりに'X'が送信されます。上の「ドライブの設定」の節を参照してください。

＋
順方向回転を設定します。動作が完了すると、コマンド文字がエコーバックされます。モーターが動作中または故障状態のときは、コマンドは無視され、代わりに'X'が送信されます。

－
逆方向回転を設定します。動作が完了すると、コマンド文字がエコーバックされます。モーターが動作中または故障状態のときは、コマンドが無視され、代わりに'X'が送信されます。

F または f
Fault をクリアします。動作が完了すると、コマンド文字がエコーバックされます。ドライブが故障状態ではない場合は、コマンドが無視され、代わりに'X'が送信されます。

G または g
モーターを動作させます。モーターが動作状態にされ、設定された向きに低速で回転します。動作が完了すると、コマンド文字がエコーバックされます。モーターがすでに動作中または故障状態のときは、コマンドが無視され、代わりに'X'が送信されます。

S または s
モーターを停止させます。モーターが停止されて、動作が完了すると、コマンド文字がエコーバックされます。モーターがすでに停止中または故障状態のときは、コマンドが無視され、代わりに'X'が送信されます。

R または r
モーター速度を設定します。これは複数文字のコマンドです。コマンド文字の後ろには、ローターの RPM で表わした目標速度を指定する正確な 4 桁の 10 進数（0～9）が続かなければなりません。モーターが動作していない場合、または要求速度がモーターの範囲外である場合（定義 Mtr_Max_Speed の値に従う）、コマンドは無視され、'X'がエコーバックされます。その他の場合は、4 桁すべてを受信した後に動作が実行され、この時点で、コマンド文字だけがエコーバックされます。数値桁以外の文字が受信されると、'X'がエコーバックされ、コマンドは廃棄されます。

?
モーター速度を取り出します。現在の速度を読み出して、1 桁の 10 進数（0～9）に変換します。ここで、各値は全速度範囲の 10％を表わします（ローターRPM 0～Mtr_Max_Speed）。コマンド文字ではなく、数値がエコーバックされます。速度取得コマンドは、現在の状態に無関係に常に有効です。

高度な構成

8051 のコードは、上で説明した例よりも複雑なモーター制御関数の作成にも使えます。例えば、洗濯機の「洗濯サイクル」では、ある方向へのモーターの急激な加速と停止、逆方向への急激な加速が必要です。これは、回転数が汚れ度合設定の低、中、高に依存するという方法で実現できます。実現できるその他の動作としては、キャッチ・スピンやオート・リバランスなどがあります。

UART 以外のコマンド・インタフェースが使用できることに注意してください。例えば、デジタル I/O ピンからコマンドを受け取るときは、該当するピンに対応する SFR の状態を監視して、必要に応じて MotorSeq をコールするように MotorCtrl を変更します。

クロック速度などのある変数を設定したら、変数を EEPROM の設定領域に格納できます。

8051 コードの EEPROM へのダウンロード

FS2 ポッドと Keil uVision を使って 8051 のアプリケーション・コードを完全にテストしたら、スタンドアロン・テストのためにコードを EEPROM にダウンロードできます。これは、次のステップに従って行います。

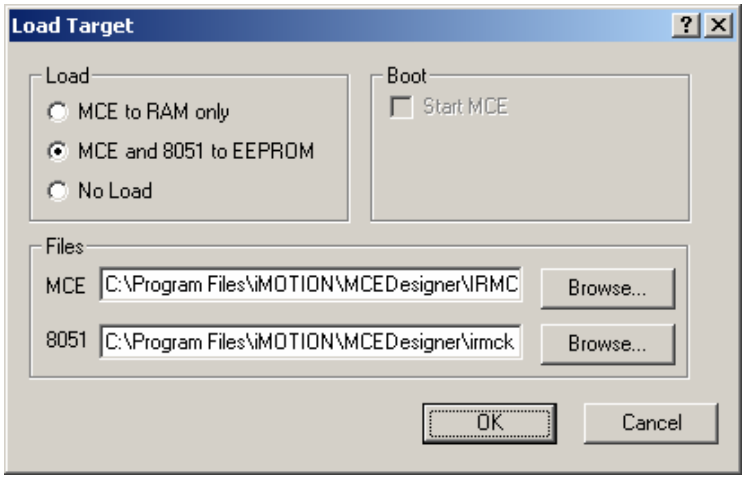


図 5　MCEDesigner の Load Target ウインドウ画面

1. 制御基板の電源を落とし、次に FS2 デバッグ・ポッドの電源を落とします。制御基板から FS2 ポッドを取り外します。
2. 基板の電源を入れて、MCEDesigner を起動します。コントローラ基板用の.irc ファイルを開きます。
3. System ウインドウをクリックし、次に Tools ⇒ Load Target を選択します。上の図 5 に、Load Target ウインドウ画面を示します。
4. “MCE and 8051 to EEPROM” を選択します。該当する MCE プログラムを選択し、uVision2 で生成した 8051 プログラムの.hex ファイルを選択します。
5. ダウンロードが完了したら（約 2 分）、制御基板の電源を落とし、COM が Down するのを待ちます。
6. 基板の電源を入れて、COM は Down したままにします。MCEDesigner を閉じて、COM ポートを開きます。
7. HyperTerminal（ハイパーターミナル）または他の UART 通信プログラムを起動して、モーター制御関数が正常に動作していることを確認します。

元の 8051 コードへの回復

スタンドアロン・テストの後、元の 8051 コードを EEPROM に再び戻せば、MCEDesigner を使ってアプリケーション開発を続けられます。次のステップに従って行います。

1. すべての装置の電源を落とします。コンピュータと FS2 ポッドを制御基板に接続します。
2. FS2 ポッドの電源を入れ、次いで制御基板の電源を入れます。
3. uVision2 内で、MCEDesigner の 8051 コードに対応する Project を開きます（例えば、IRMCx341Lib.Uv2）。Debug ⇒ Start/Stop Debug Session を選択します。プログラムがロードされるのを待ちます。Debug ⇒ Go を選択します。
4. MCEDesigner を開くと、COM が動作します（緑色）。
5. System ウインドウをクリックし、次に Tools ⇒ Load Target を選択します。
6. "MCE and 8051 to EEPROM"を選択します。該当する MCE プログラムを選択し、元のコードに対応した 8051 プログラムの.hex ファイル（例えば、IRMCx341Lib.hex）を選択します。
7. ダウンロードが完了したら（約 2 分）、制御基板の電源を落とし、COM が停止するのを待ちます。デバッガを停止させて、デバッグ・セッションを終わります。FS2 ポッドをオフして切り離します。
8. 基板の電源を入れると、COM は再び動作します。システムは、MCEDesigner からのコマンドを受け付けられる状態になります。

トラブル・シューティング

デバッグ・セッションが開始されて、Output ウインドウにメッセージ“***error122: AGDI: memory read failed”が表示されたとき。

- FS2 デバッグ・ポッドがパラレル・ポートに接続されていて、オンしていることを確認します。
- Under Project ⇒ Options for Target 'IRSamples'で Debug タブをクリックし、次に Settings ボタンをクリックして、Comm Port Setting が Lpt1（または使用パソコンに応じて適切なポート）になっていることを確認します。

HyperTerminal（ハイパーターミナル）に文字がエコーバックされないとき。

- FS2 デバッグ・ポッドが正しくハードウエアに接続されていて、オンしていることを確認します。
- Under Project ⇒ Options for Target 'IRSamples'で、Debug タブをクリックし、“Use:”が Fs2/Keil ISA-M8051EW Driver に設定されていることを確認します。Settings が TckRate: 62500 と Tvcc Threshold : 2500 になっていることを確認します。
- HyperTerminal（ハイパーターミナル）が動作しているコンピュータが、ハードウエアに接続（シリアル・ケーブル使用）されていることを確認します。このドキュメント（ソフトウエア・セットアップ）で説明したように、HyperTerminal（ハイパーターミナル）が正しい通信オプションの正しいポートを使っていることを確認します。COM ポートは、MCEDesigner が使っているポートと同じでなければなりません。

ドライブの設定後（HyperTerminal（ハイパーターミナル）では c または C）、LED が赤から緑の点滅へ変わらないとき（IRMCS3041 の場合）。

- DC バス電圧が範囲内であり、故障の原因がないことを確認します。

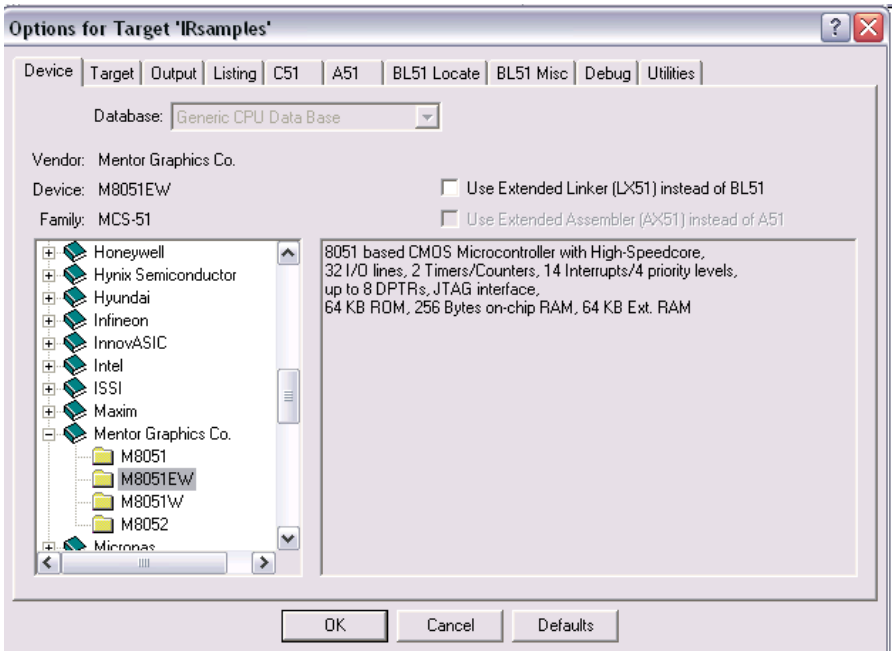
ドライブの設定後、LED が赤から緑の点滅へ変わるが（IRMCS3041 の場合）、コマンドを与えてもモーターが回転しないとき。

— 正しい駆動パラメータが MotorControl.h に入力されていることを確認します。
— MCE が正常に起動されたことを確認します。
— MCEBoot.c にブレーク・ポイントを設定して doMceBoot をコールしたかどうかを調べます。コールしていない場合は、MceInfo 構造が正しくないか、メモリー・アドレス RomMceInfo が正しくない可能性があります。

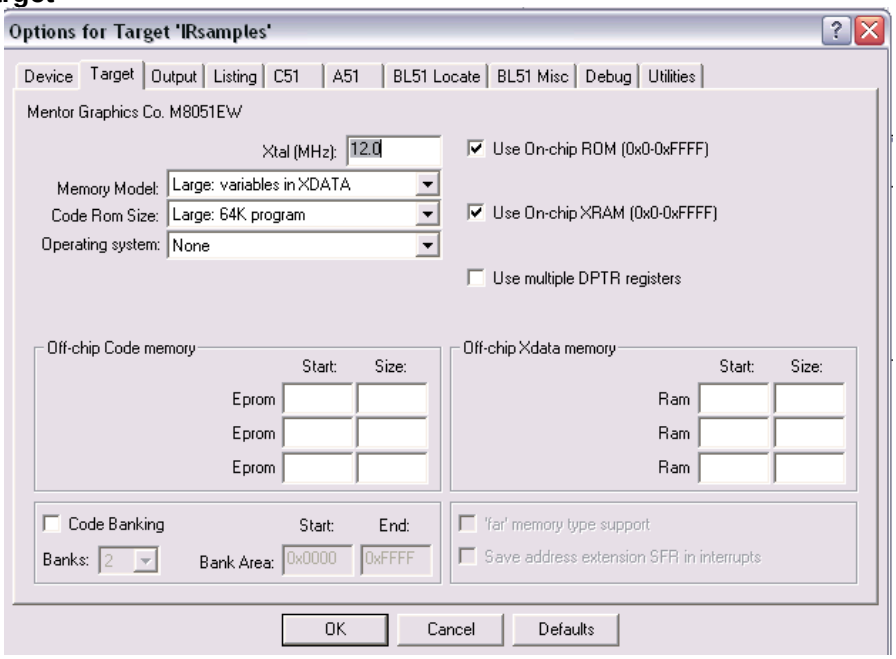
付録：Keil uVision プロジェクト・オプション

以下は、Project ⇒ Options for Target 'IRSamples'で設定するオプションです。uVision2 プロジェクト・ファイル（IRsamples.Uv2）は、次に示すすべてのオプション・セットと一緒に出荷されています。

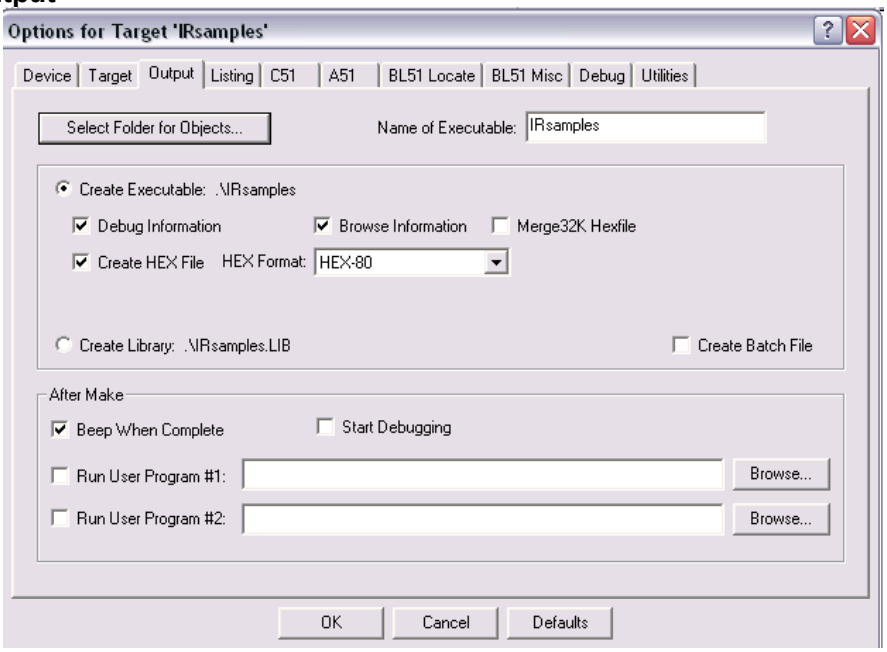
Device



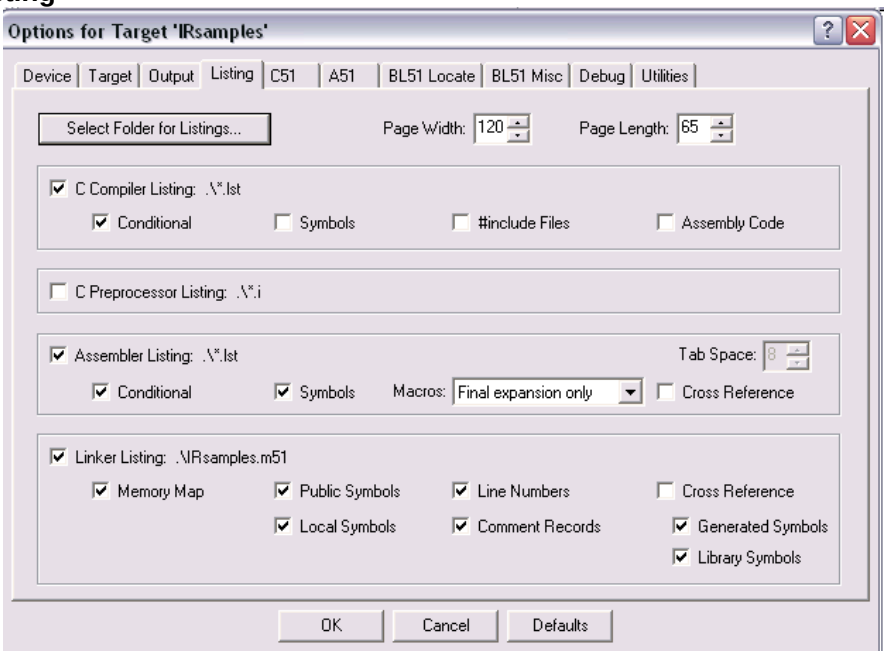
Target



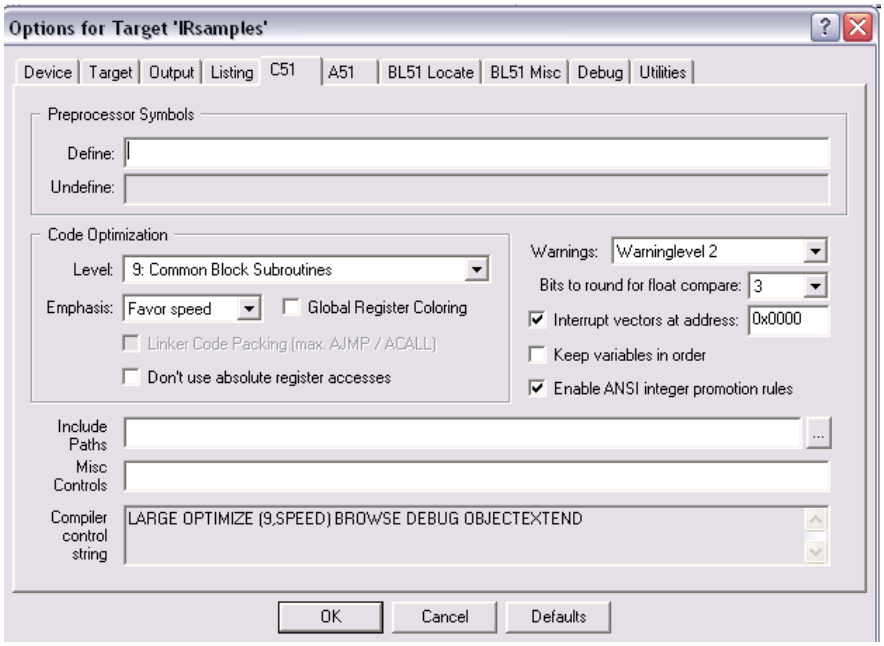
Output



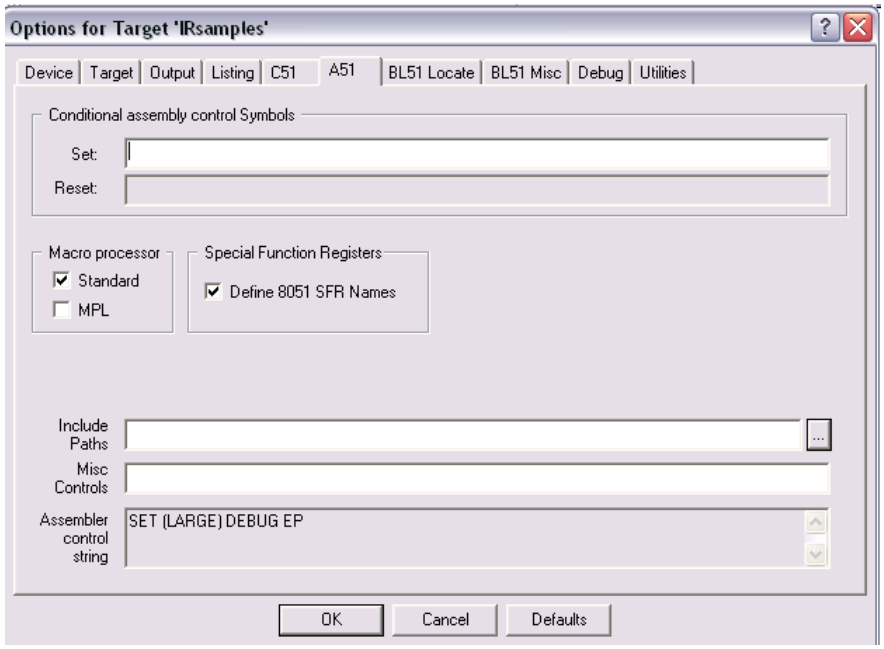
Listing



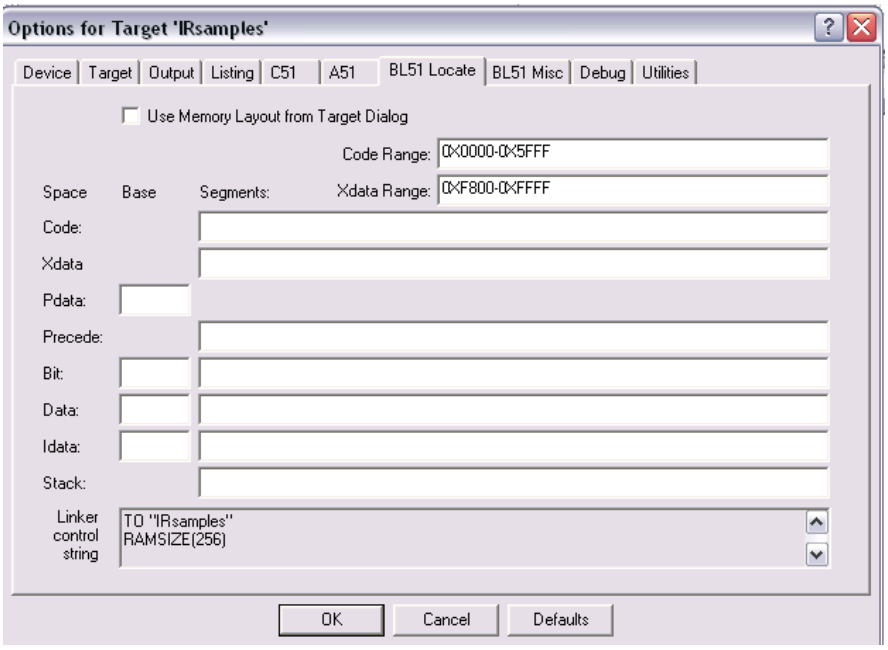
C51


Options for Target 'IRsamples'
Device
Target
Output
Listing
C51
A51
BL51 Locate
BL51 Misc
Debug
Utilities
Preprocessor Symbols
Define:
Undefine:
Code Optimization
Level:
9: Common Block Subroutines
Emphasis:
Favor speed
Global Register Coloring
Linker Code Packing (max. AJMP / ACALL)
Don't use absolute register accesses
Warnings:
Warninglevel 2
Bits to round for float compare:
3
Interrupt vectors at address:
0x0000
Keep variables in order
Enable ANSI integer promotion rules
Include Paths
Misc Controls
Compiler control string
LARGE OPTIMIZE (9,SPEED) BROWSE DEBUG OBJECTEXTEND
OK
Cancel
Defaults

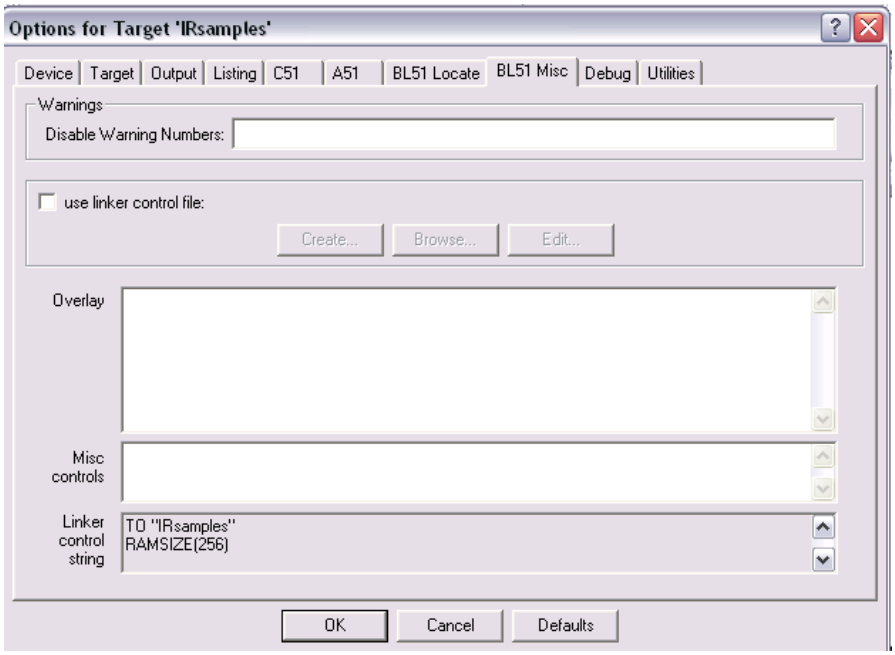
A51


Options for Target 'IRsamples'
Device
Target
Output
Listing
C51
A51
BL51 Locate
BL51 Misc
Debug
Utilities
Conditional assembly control Symbols
Set:
Reset:
Macro processor
Standard
MPL
Special Function Registers
Define 8051 SFR Names
Include Paths
Misc Controls
Assembler control string
SET (LARGE) DEBUG EP
OK
Cancel
Defaults

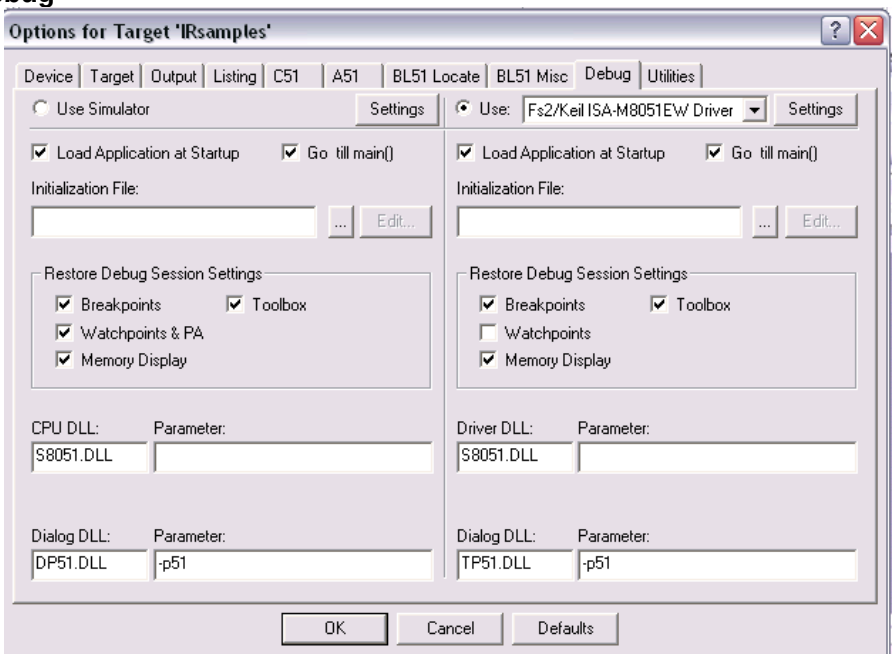
BL51 Locate



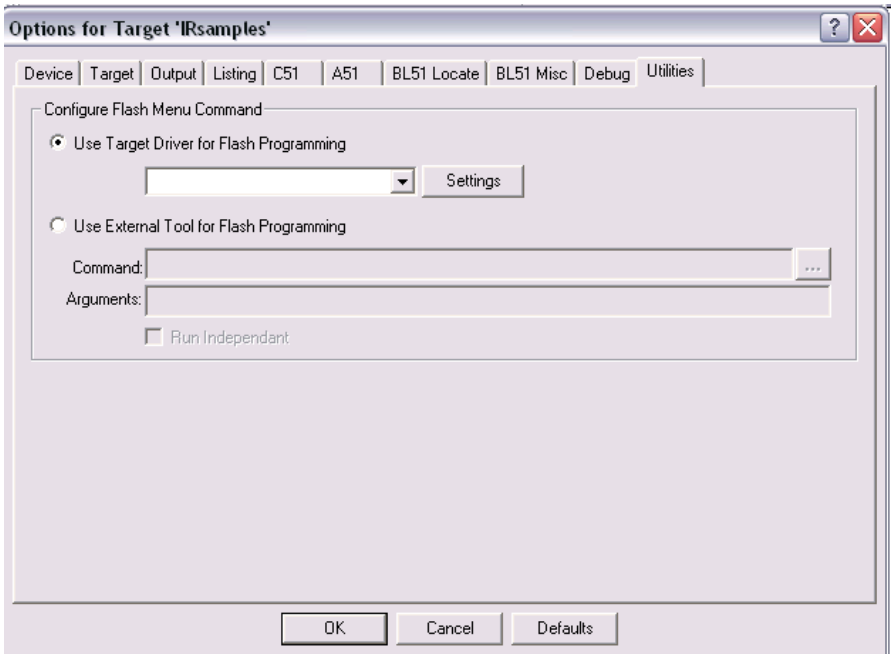
BL51 Misc



Debug



Utilities



©インターナショナル・レクティファイアー・ジャパン
この文献の無断複製・転載を禁じます。