お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

User's Manual

740ファミリ用
Cコンパイラパッケージ V.1.01

ユーザーズマニュアル

ルネサス エレクトロニクス
www.renesas.com

Rev.1.01　2007.06

本資料ご利用に際しての留意事項

１．本資料は、お客様に用途に応じた適切な弊社製品をご購入いただくための参考資料であり、本資料中に記載の技術情報について弊社または第三者の知的財産権その他の権利の実施、使用を許諾または保証するものではありません。
２．本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例など全ての情報の使用に起因する損害、第三者の知的財産権その他の権利に対する侵害に関し、弊社は責任を負いません。
３．本資料に記載の製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的、あるいはその他軍事用途の目的で使用しないでください。また、輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、それらの定めるところにより必要な手続を行ってください。
４．本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例などの全ての情報は本資料発行時点のものであり、弊社は本資料に記載した製品または仕様等を予告なしに変更することがあります。弊社の半導体製品のご購入およびご使用に当たりましては、事前に弊社営業窓口で最新の情報をご確認頂きますとともに、弊社ホームページ(http://www.renesas.com)などを通じて公開される情報に常にご注意下さい。
５．本資料に記載した情報は、正確を期すため慎重に制作したものですが、万一本資料の記述の誤りに起因する損害がお客様に生じた場合においても、弊社はその責任を負いません。
６．本資料に記載の製品データ、図、表などに示す技術的な内容、プログラム、アルゴリズムその他応用回路例などの情報を流用する場合は、流用する情報を単独で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断して下さい。弊社は、適用可否に対する責任は負いません。
７．本資料に記載された製品は、各種安全装置や運輸・交通用、医療用、燃焼制御用、航空宇宙用、原子力、海底中継用の機器・システムなど、その故障や誤動作が直接人命を脅かしあるいは人体に危害を及ぼすおそれのあるような機器・システムや特に高度な品質・信頼性が要求される機器・システムでの使用を意図して設計、製造されたものではありません（弊社が自動車用と指定する製品を自動車に使用する場合を除きます）。これらの用途に利用されることをご検討の際には、必ず事前に弊社営業窓口へご照会下さい。なお、上記用途に使用されたことにより発生した損害等について弊社はその責任を負いかねますのでご了承願います。
８．第７項にかかわらず、本資料に記載された製品は、下記の用途には使用しないで下さい。これらの用途に使用されたことにより発生した損害等につきましては、弊社は一切の責任を負いません。
　１）生命維持装置。
　２）人体に埋め込み使用するもの。
　３）治療行為（患部切り出し、薬剤投与等）を行なうもの。
　４）その他、直接人命に影響を与えるもの。
９．本資料に記載された製品のご使用につき、特に最大定格、動作電源電圧範囲、放熱特性、実装条件およびその他諸条件につきましては、弊社保証範囲内でご使用ください。弊社保証値を越えて製品をご使用された場合の故障および事故につきましては、弊社はその責任を負いません。
１０．弊社は製品の品質および信頼性の向上に努めておりますが、特に半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。弊社製品の故障または誤動作が生じた場合も人身事故、火災事故、社会的損害などを生じさせないよう、お客様の責任において冗長設計、延焼対策設計、誤動作防止設計などの安全設計（含むハードウエアおよびソフトウエア）およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特にマイコンソフトウエアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願い致します。
１１．本資料に記載の製品は、これを搭載した製品から剥がれた場合、幼児が口に入れて誤飲する等の事故の危険性があります。お客様の製品への実装後に容易に本製品が剥がれることがなきよう、お客様の責任において十分な安全設計をお願いします。お客様の製品から剥がれた場合の事故につきましては、弊社はその責任を負いません。
１２．本資料の全部または一部を弊社の文書による事前の承諾なしに転載または複製することを固くお断り致します。
１３．本資料に関する詳細についてのお問い合わせ、その他お気付きの点等がございましたら弊社営業窓口までご照会下さい。

製品の内容及び本書についてのお問い合わせ先
インストーラが生成する以下のテキストファイルに必要事項を記入の上、コンタクトセンタ csc@renesas.comまで送信ください。
¥SUPPORT¥製品名¥SUPPORT.TXT
株式会社ルネサス テクノロジ
コンタクトセンタ　csc@renesas.com
ユーザ登録窓口　regist_tool@renesas.com
ホームページ　http://japan.renesas.com/tools

目次

図目次

表目次

はじめに

このユーザーズマニュアルをお読みになる前に、製品に添付されているリリースノートを必ずお読みください。製品構成、製品の扱い、注意事項などの重要な内容が記述されています。

■対象読者

このユーザーズマニュアルでは、次の方を対象としています。

・C 言語による組み込み用プログラム開発およびデバッグ経験のある方
・High-performance Embedded Workshop を初めてご使用になる方

■参考マニュアル

High-performance Embedded Workshop に関する詳しい用語説明、機能説明、操作については、High-performance Embedded Workshop のユーザーズマニュアルを参照してください。

IAR Systems 社製 C コンパイラ ICC740 に関する詳しい用語説明、機能説明、操作については、ICC740 に付属の PDF マニュアルを参照してください。

1. 概要

740 ファミリ用 C コンパイラパッケージは IAR Systems 社製 C コンパイラ ICC740（以下 ICC740 と略します）と High-performance Embedded Workshop を組み合わせた開発環境システムです。C 言語およびアセンブリ言語によるプログラム開発を支援します。

以下のソフトウェアを同梱しています。

・SC74
ルネサス製 740 ファミリ用アセンブラパッケージで作成したソースファイルを ICC740 に付属のアセンブラ A740 のソースファイル形式に変換するソースファイルコンバータです。技術サポート対象外ソフトウェアです。SC74 の取り扱いは付属のファイル license.sj を参照してください。

本マニュアルでは、High-performance Embedded Workshop での ICC740 プロジェクトの開発方法について説明します。

2. クイックツアー

本章ではクイックツアーにより ICC740 を使用した一般的な開発手順を説明します。
各手順の詳細は 3 章以降を参照してください。

2.1. 新規のプロジェクトを作成する

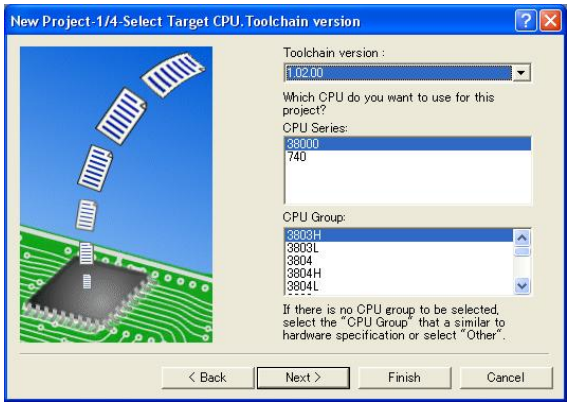
新規ワークスペースを作成します。
740 ファミリ用C コンパイラパッケージの[新規プロジェクトワークスペース]ダイアログボックス（図 1）で、以下の選択を行います。

CPU family： 740 Family
Tool chain： IAR ICC740 (740 Family)

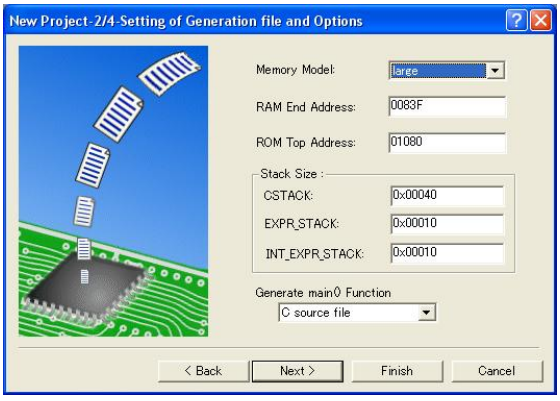
図 1：[New Project Workspace]ダイアログボックス

プロジェクトタイプで[Application (Enhance Version)]を選択後、ワークスペース名、プロジェクト名を設定して、OK ボタンをクリックします。

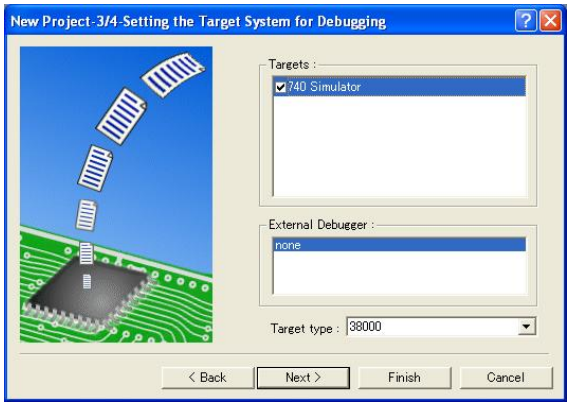
[New Project]ウィザード(図 2)によりプロジェクトを作成します。
[New Project]ウィザードの(3)では、[740 Simulator]にチェックマークを指定します。



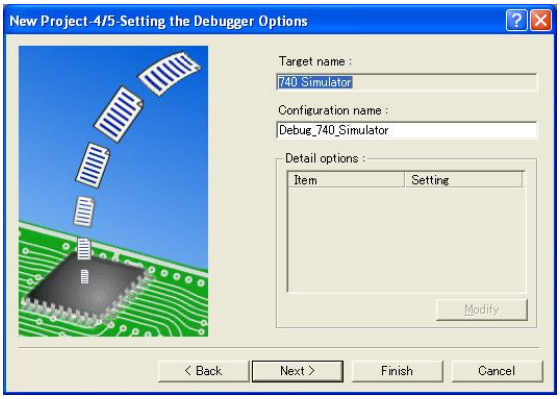
(1) ステップ 1



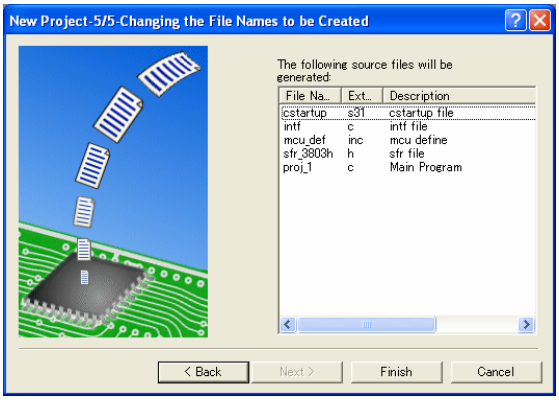
(2) ステップ 2



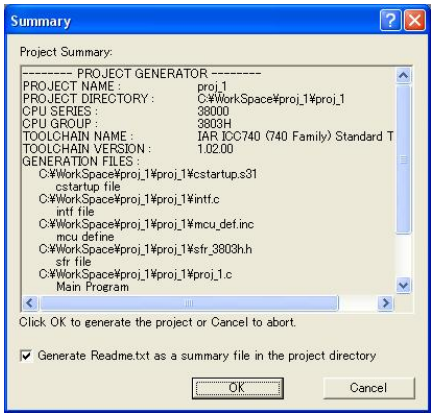
(3) ステップ 3



(4) ステップ 4



(5) ステップ 5



(6) Summary

図 2：[New Project]ウィザード

740 ファミリ用 C コンパイラパッケージでは、以下のコンフィグレーションとセッションを作成します。

表 1：コンフィグレーションとセッション

Configuration

Debug	デバッグ用コンフィグレーション
Release	リリース用コンフィグレーション

Session

DefaultSession	ターゲットを選択していないセッション
Session740_Simulator	シミュレータ用セッション

新規ワークスペース作成時はコンフィグレーションとしてDebug、セッションとしてDefaultSessionが選択されています（図 3)。

図 3：コンフィグレーションとセッションの表示

また、ワークスペースウィンドウのプロジェクトタブは以下の表示となります（図 4)。

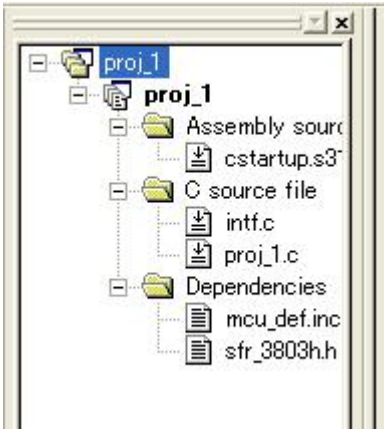


図 4：ワークスペースウィンドウのプロジェクトタブ

2.2.　ソースファイルを作成し、登録する

ソースプログラムを作成します。今回は、SmpTw74 フォルダにある tutor3.c を使用します。tutor3.c をプロジェクトフォルダにコピーしてください。
SmpTw74 フォルダは ICC740 をインストールしたフォルダ（通常は¥Program Files¥IAR Systems¥ew23¥740）に作成されています。
ファイルをプロジェクトフォルダに配置しただけでは、登録されませんので、[プロジェクト]メニューの[ファイルの追加...]により登録してください（図 5）。

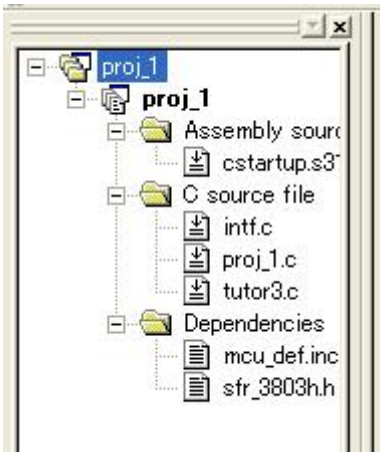


図 5：ファイルを登録したプロジェクトタブ

また、proj_1.c と intf.c は以下のように修正してください。エディタはファイルをダブルクリックすることで起動できます。

proj_1..c の修正

```
#include <intr740.h>

void tutor3( void );

void main(void)
{
        tutor3();
}
```

intf.c の修正

```
/*
void interrupt[0] I_BRK(void){

}
*/

void interrupt[2] I_AD_SIO3T(void){
```

ソースファイル tutor3.c について：
ソースファイル tutor3.c は、IAR Systems 社作成のサンプルプログラムを 740 シミュレータ用に変更したものです。このプログラムの処理およびソースは以下のとおりです。

＜処理＞
プログラムは tutor3()関数の while 文の中で無限ループとなります。
変数 my_char は関数により'a'～'z'の値をランダムに持ちます。
変数 my_char が'i'になると、関数により brk 割り込みが発生します。
関数 brk_interrupt は brk 割り込みにより実行される関数で変数 my_char を'.'に設定します。この関数の"interrupt [0x00]"記述は 3803 グループの brk 割り込みベクトルに関数 brk_interrupt のアドレスを設定します。
その後、tutor3()関数の無限ループの処理が繰り返されます。

ソースプログラム tutor3.c

```
/*-------------------------------------------------
 * File: tutor3.c
 *
 * Purpose: Handling real time interrupts
 *
 * Usage: ICC      -r -L -q tutor3.c
 *              XLINK              -r -f <link file> tutor3
 *
 * Description: Using BRK vector to print a character
 *
 * Copyright 1997 IAR Systems
 *
 * $Id: tutor3.c 1.3 1998/01/15 09:24:55Z Laban Exp $
 *-------------------------------------------------*/

#pragma language=extended       /* enable use of extended keywords */

#include <stdlib.h>
#include <stdio.h>
#include "intr740.h"                /* include intrinsics */

/*********************************
 *              Variables                        *
 *********************************/

char my_char = '*';
int call_count = 0;

/*********************************
 *              Start of code                    *
 *********************************/

void interrupt [0x00] brk_interrupt(void)
{
//      putchar('I');
        my_char='.';
}

void execute_brk(void)
{
        break_instruction();      /* Use intrinsic function */
}

void do_foreground_process(void)
{
        call_count++;
//      putchar(my_char);
        my_char = rand() % 26 + 'a';
}

void tutor3(void)
{
        while (1)
        {
                do_foreground_process();
                if (my_char=='i') execute_brk();
        }
}
```

2.3. ビルドする

ビルドは、［ビルド］、［すべてをビルド］、または［ファイルのコンパイル］ボタンにより行います。ここでは、［すべてをビルド］ボタンをクリックしてください（図 6）。

図 6：［ビルド］、［すべてをビルド］、［ファイルのコンパイル］ボタン

2.4. デバッガを起動する

ビルドによるエラーの発生が無くなれば、デバッグを行います。
デバッガを起動するためセッションを選択します（図 7）。
今回は、740 シミュレータを使用しますので、Session740_Simulator を選択します。

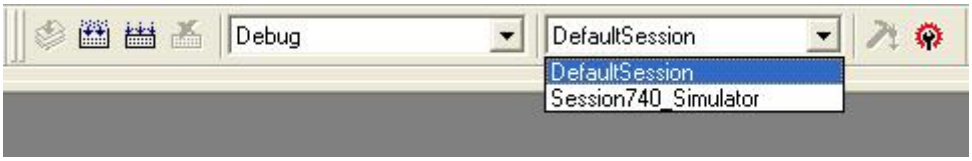


図 7：セッションの表示

Session740_Simulatorを選択すると以下のダイアログが起動されます（図 8）。

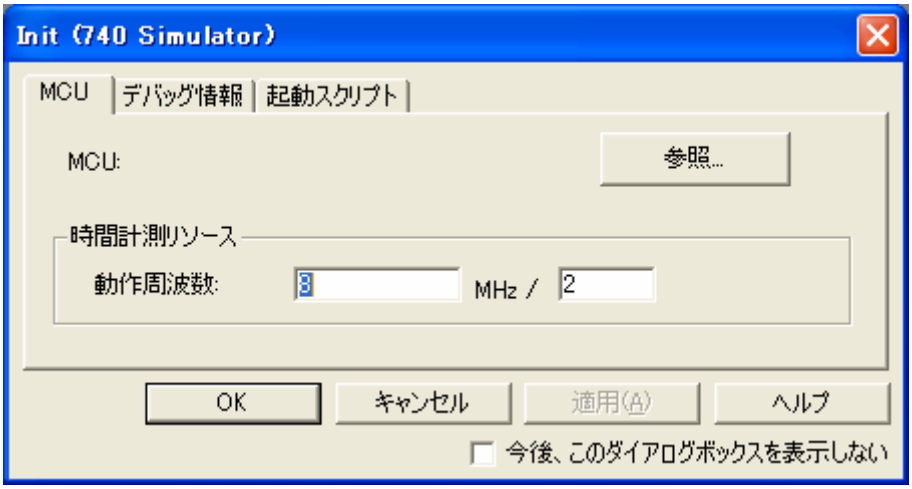


図 8：740 シミュレータの Init ダイアログ

このダイアログでは MCU を選択する必要があります。[Refer..]ボタンをクリックして M38000.mcu ファイルを選択してください。
[OK]ボタンをクリックすると、図 9の表示になります。

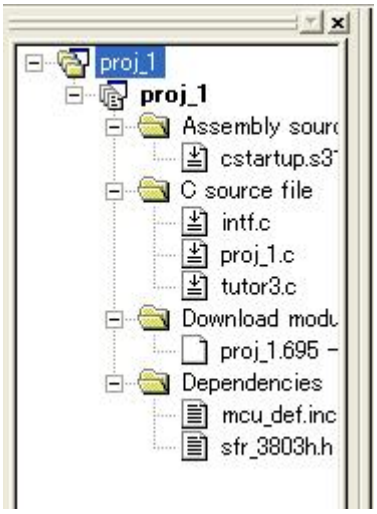


図 9：[ダウンロード]が追加されたプロジェクトタブ

2.4.1. プログラムをロードする

プログラムのダウンロードは、[デバッグ]メニューの[ダウンロード]で行います。
ダウンロードが完了するとプロジェクトタブ上の[ダウンロード]のアイコンが変わります（図 10）。

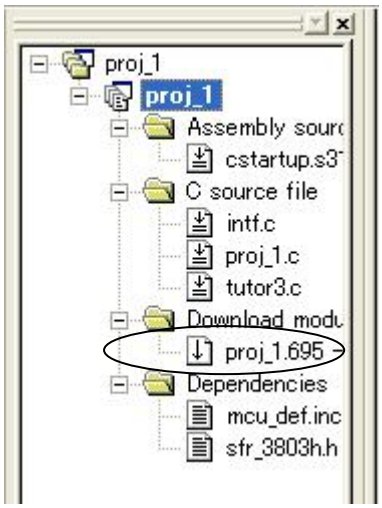


図 10：ダウンロード後のプロジェクトタブ

プログラムを実行するには、まず CPU のリセットを行ってください。

[デバッグ]メニューの[CPU のリセット]を選択します。

cstartup.s31 ファイルが 図 11のように表示されます。

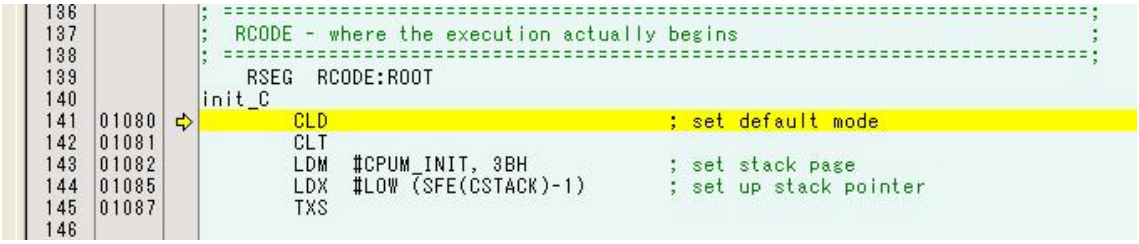


図 11：リセット後の cstartup.s31

2.4.2. main関数まで実行する

先ず、main()関数まで実行します。

main()関数にブレークポイントを設定します。

proj_1.c をダブルクリックしてファイルをオープンしてください。

そして、行番号 7 の表示場所にマウスを移動してダブルクリックください。

図 12のようにブレークポイントが設定されます。

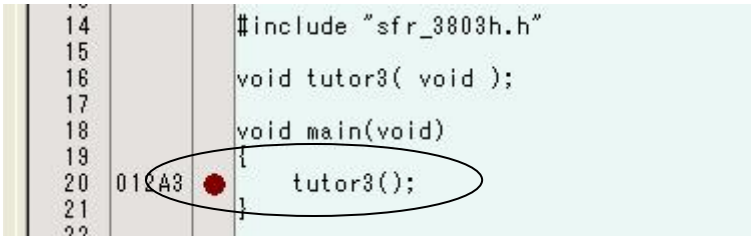


図 12：ブレークポイントの設定

この状態で[実行]ボタンをクリックしてください（図 13)。

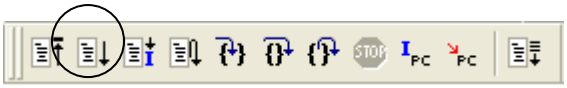


図 13：Debug Run toolbar

この位置までのプログラムを実行し、行番号 7 には黄色の矢印が付加されます（図 14)。

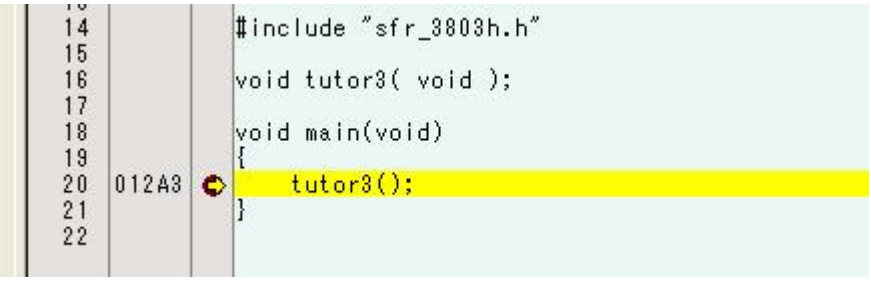


図 14：ブレークポイントでのプログラム停止

次に[ステップイン]ボタンをクリックしてください。tutor3.c が表示されます。

2.4.3. 割り込み発生の確認を行う

割り込み発生の確認を行います。
brk_interrupt()関数は、どの関数からも呼び出されません。BRK 命令を実行することで呼び出されます。
BRK 命令の実行は execute_brk()関数内の break_instruction()関数が行います。この break_instruction()関数は BRK 命令に置き換えられるインライン関数です。
execute_brk()関数は変数 my_char が'i'に設定されると呼び出されます。
変数 my_char の値をウォッチするには、[C ウォッチ]ウィンドウで行います。[表示]メニューから[シンボル]を選択し、[C ウォッチ]を選んでください。
[C ウォッチ]ウィンドウがオープンします（図 15)。

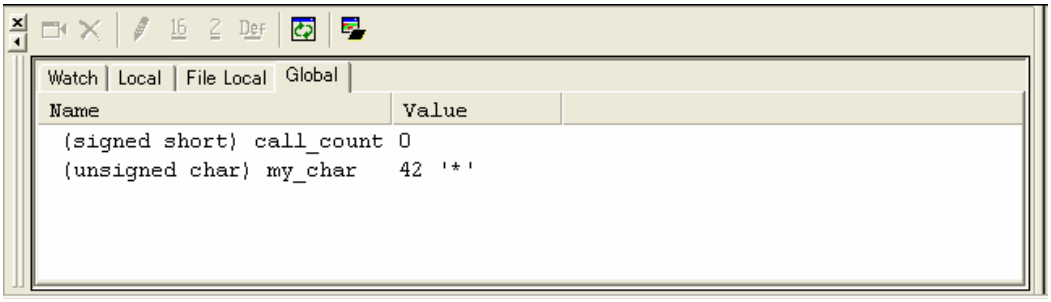


図 15：C Watch Window

この中から[Global]タブを選択してください。

tutor3.c の行番号 37 と 57 にブレークポイントを設定して、[実行]ボタンを何度かクリックしてください。
変数 my_char が'i'に設定された時に、次の[実行]ボタンのクリックで行番号 37 黄色の矢印となっていることで割り込み発生の確認を行います。

これにてクイックツアーを終了します。

3. プロジェクトの新規作成

740 ファミリ C コンパイラパッケージの[新規プロジェクトワークスペース]ダイアログボックス（図 16）で、以下の選択を行います。

CPU family： 740 Family
Tool chain： IAR ICC740 (740 Family)

図 16：[New Project Workspace]ダイアログボックス

3.1. 新規作成

ICC740 では、プロセッサ・グループ、メモリモデル、およびスタック領域を設定して、プログラムの開発を行う必要があります。740 ファミリ C コンパイラパッケージ V.1.01 Release 02 では、[New Project]ウィザードにより新規プロジェクトを作成します。

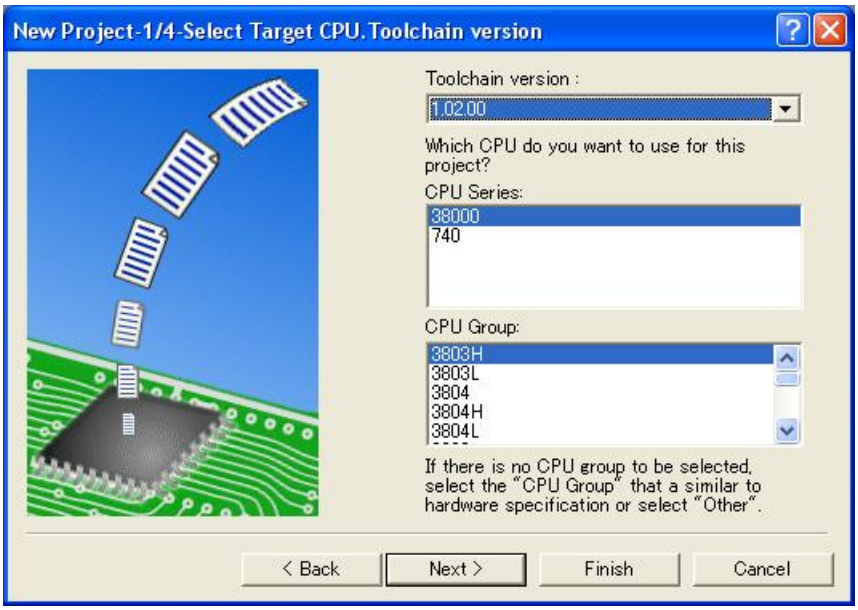


図 17：[New Project]ウィザード：ステップ 1

CPU のシリーズとグループを選択します。

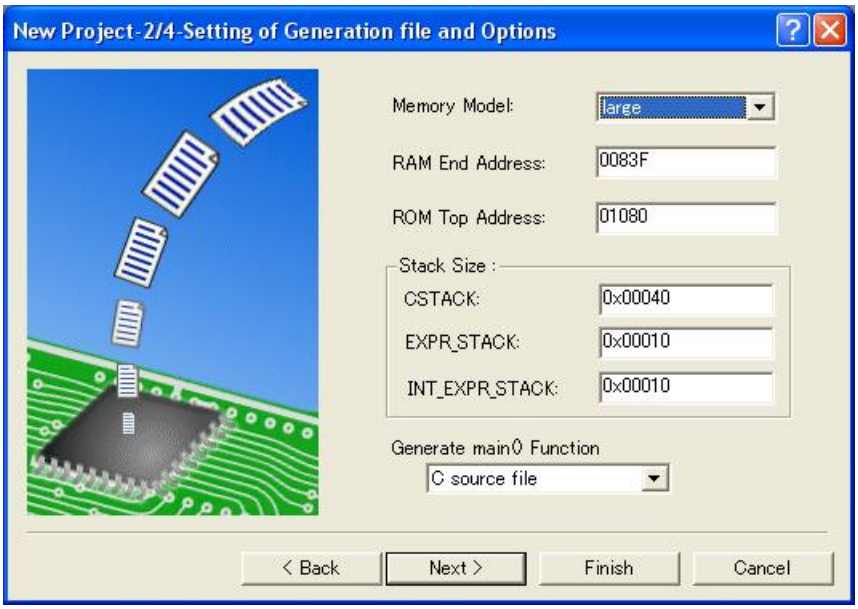


図 18：[New Project]ウィザード：ステップ 2

メモリモデル、RAM と ROM のアドレス、およびスタックサイズを設定します。
これらの値は各 CPU グループのメモリ容量が最小の場合の値を表示していめす。
ご使用の製品に合わせて変更してください。

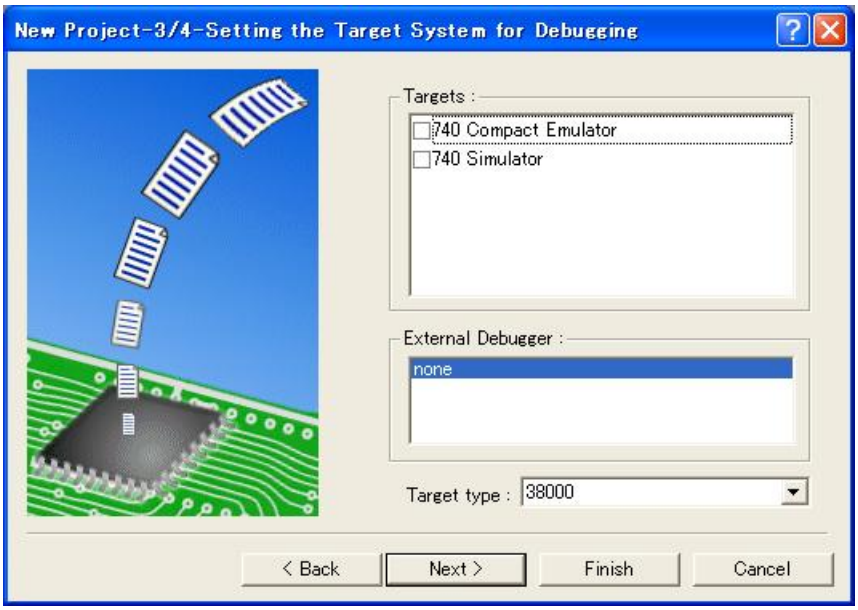


図 19：[New Project]ウィザード：ステップ 3

デバッグターゲットを選択します。図では既にコンパクトエミュレータがインストールされている場合となります。
デバッグターゲットを選択すると以下のダイアログボックスを表示します。

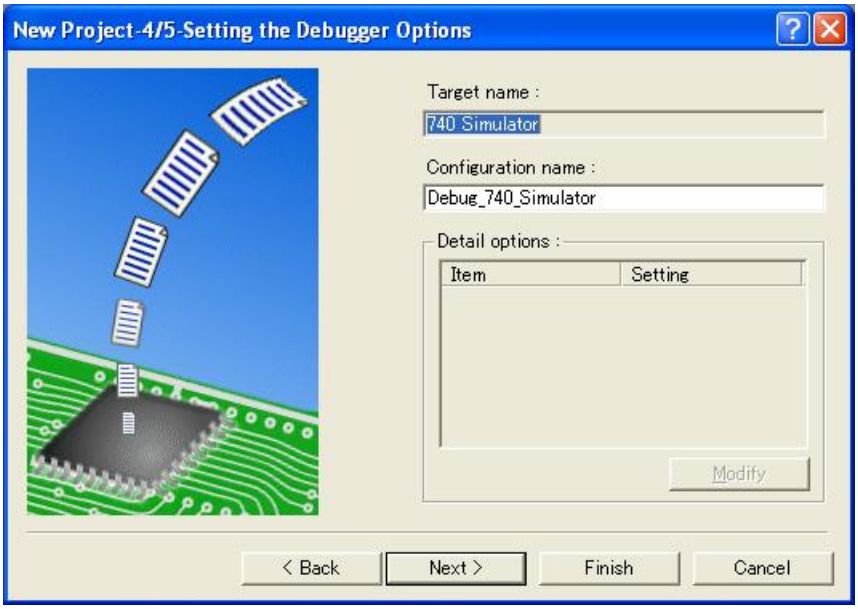


図 20：[New Project]ウィザード：ステップ 4

コンフィグレーション名を設定します。

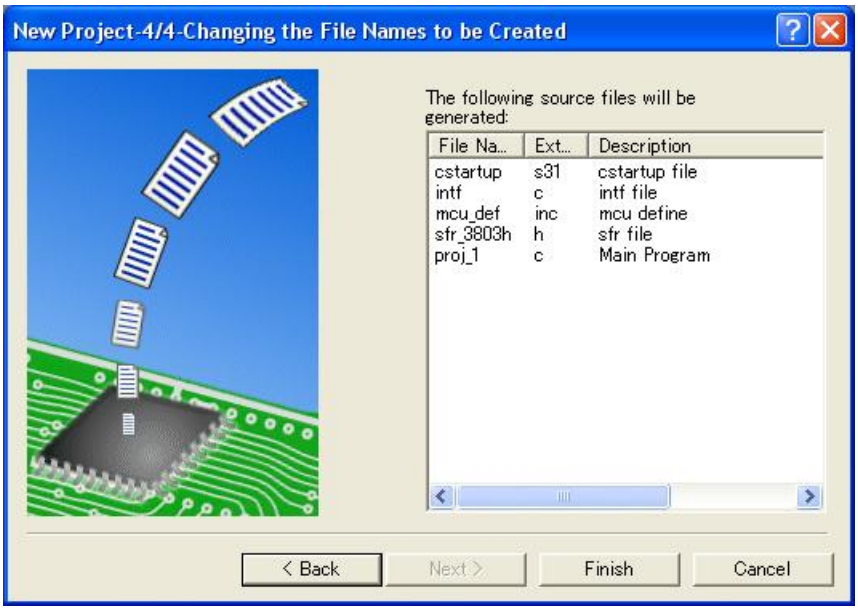


図 21：[New Project]ウィザード：ステップ 5

プロジェクトに登録されるファイルを表示します。

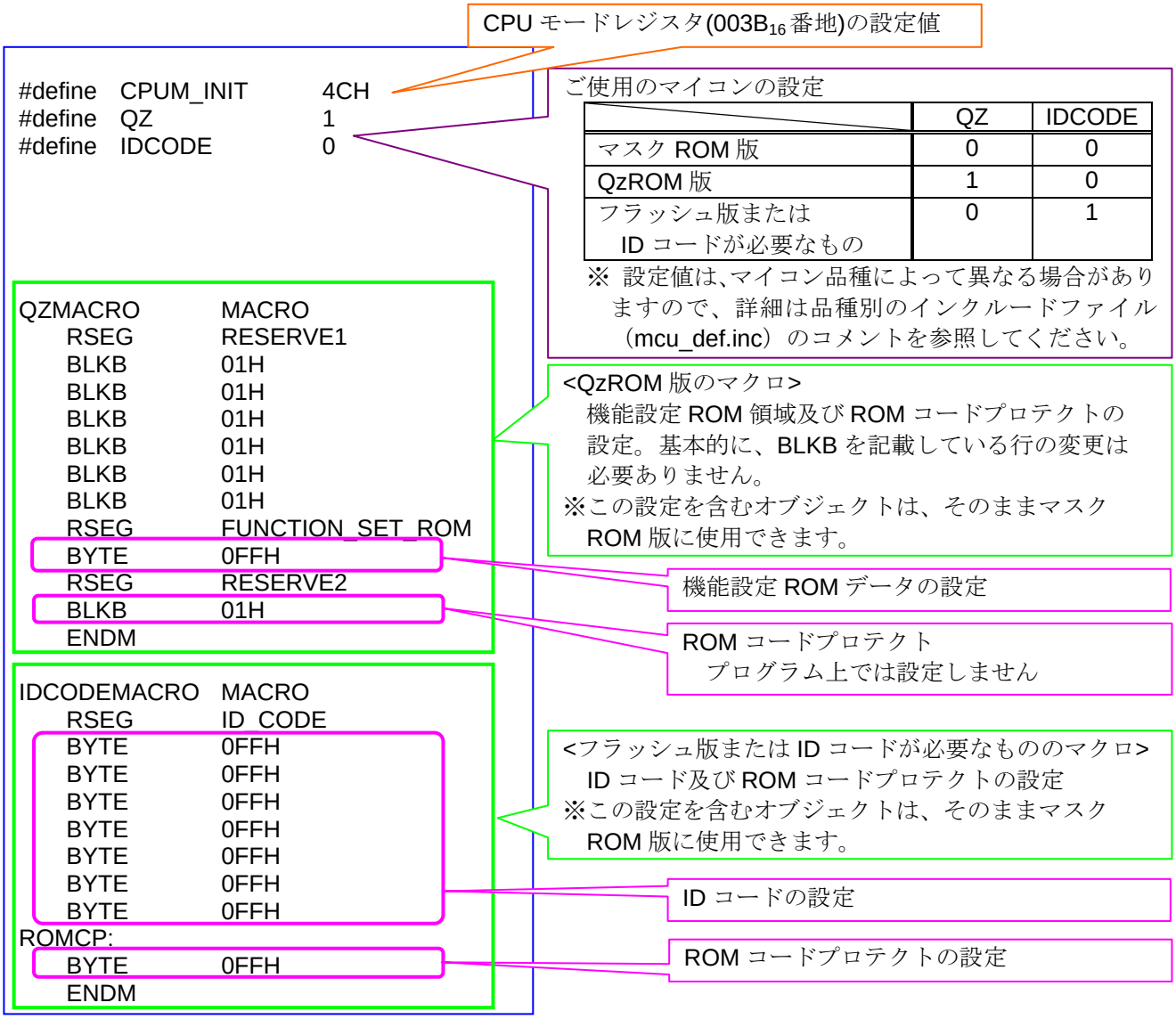
表 2：生成ファイル

ファイル	内容
cstartup.s31	スタートアップファイル。 [New Project]ウィザードのステップ 1 の内容が反映される
intf.c	割り込み関数のテンプレート
mcu_def.inc	MCU 情報ファイル。 [New Project]ウィザードのステップ 1 の内容が反映される
sfr_3803h.h	MCU に対応した SFR ヘッダファイル。 [New Project]ウィザードのステップ 1 の内容が反映される
proj_1.c	関数 main()を含んだ C ソースファイル

ご使用のマイコン、システムに応じて変更してください。ご使用のマイコンについては、マイコンデータシートを参照してください。

・mcu_def.inc の説明

インクルードファイル（mcu_def.inc）は、cstartup.s31 でインクルードしています。
下記内容をご使用のマイコンにあわせて設定してください。



＜注意事項＞

- “CPUM_INIT”はリセットスタート時に CPU モードレジスタへ設定する値です。CPU モードレジスタには 1 度しか書き込みできないビットを含むマイコンがあります。対象マイコンのデータシートでご確認ください。
- #define 行には、コメントを書かないでください。

・intf.c の説明

割り込み関数ファイル(intf.c)は、各マイコンで持っている割り込み関数を記述しています。ご使用になる割り込み関数のプログラムをこのファイル上に記述して使用できます。必要に応じてプロジェクトファイルに追加してください。

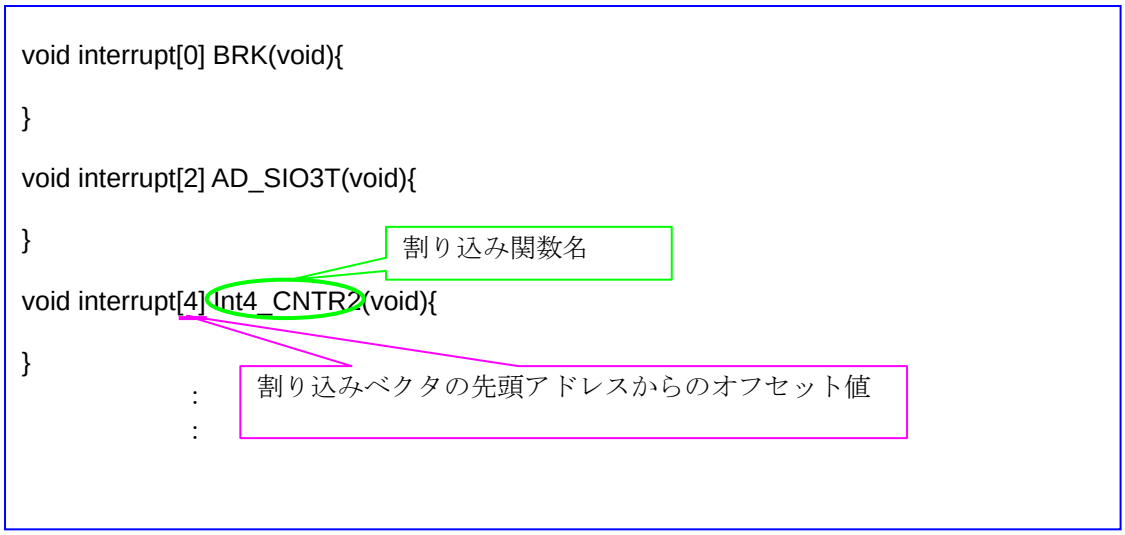


表 3のコンフィグレーションとセッションを作成します。

表 3：コンフィグレーションとセッション

Configuration

Debug	デバッグ用コンフィグレーション
Release	リリース用コンフィグレーション
Debug_740_Simulator	シミュレータ用コンフィグレーション [New Project]ウィザードのステップ 1

Session

DefaultSession	ターゲットを選択していないセッション
Session740_Simulator	シミュレータ用セッション デバッグターゲットに”740 Simulator”を選択した場合のセッション。
Session740_Compact_Emulator	コンパクトエミュレータ用セッション。 デバッグターゲットに”740 Compact Emulator”を選択した場合のセッション。

4. プロジェクトの編集

新規プロジェクト作成後、必要に応じてコンパイルオプション等を変更する必要があります。

4.1. オプションの編集

オプション編集は、[ビルド]メニューの[IAR ICC740 Toolchain...]により行います。この選択により 図 22の[Toolchain]ダイアログボックスが起動されます。

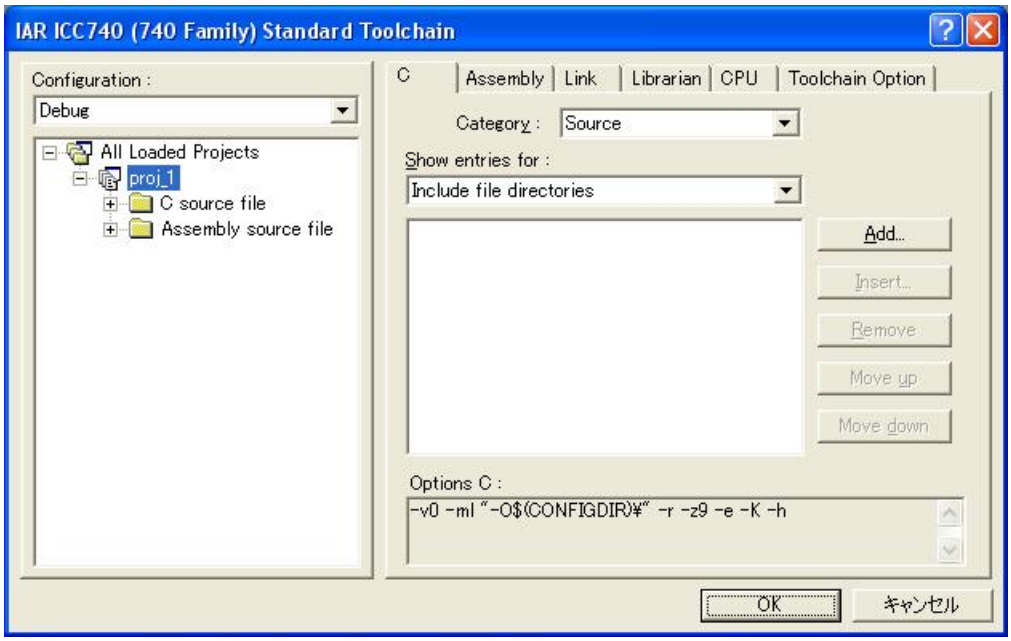


図 22：[Toolchain]ダイアログボックス

4.1.1. プロジェクトオプションの設定

[New Project]ウィザードにより、CPUタブに 表 4の内容を設定します。

表 4：CPU タブ

Item	内容	該当 option
Processor configuration	・740 with MUL/DIV 一般的なマイコン ・740 without MUL/DIV MUL/DIV 命令が無いマイコン ・740 with MUL/DIV and extended data memory access 拡張メモリアクセスを持つマイコン	-v0 -v1 -v2
Memory model	・Large ・Tiny ・0 page only	-ml -mt -mt

・Large モデルと Tiny モデルについて

Large モデルと Tiny モデルでは、変数の配置場所が異なります。

Large モデルでは 0x100 番地以降に配置され、Tiny モデルでは 0x00 番地から 0xFF 番地に配置されます。

Large モデルで変数を 0x00 番地から 0xFF 番地に配置するには拡張キーワード zpage を使用します。アセンブルソースプログラムではオペランド zp:を用いてアクセスすることでコードサイズを削減可能です。

Tiny モデルで変数を 0x100 番地以降に配置する場合は、拡張キーワード npage を使用します。アセンブルソースプログラムではオペランド np:を用いてアクセスします。

オペランド np:を使用しない場合、リンク時にエラーが発生します。

表 5：Large モデルと Tiny モデル

項目	Large モデル	Tiny モデル
変数の配置場所	0x100 番地以降	0xFF 番地以下
C 言語での 0x100 番地以降への配置		npage を使用して定義。 npage int v1; extern npage int v2;
C 言語での 0xFF 番地への配置	zpage を使用して定義。 zpage int v3; extern zpage int v4;	
アセンブルソースプログラムでの 0x100 番地以降のアクセス方法		オペランド np:を使用 lda np:v1
アセンブルソースプログラムでの 0x100 番地以下のアクセス方法	オペランド zp:を使用することでコードサイズ削減 lda zp:v3	

拡張キーワード zpage, npage は外部変数、自動変数、および関数引数に指定できます。

4.2. ICC740 オプションの設定

オプションの設定はCタブで行います。Cタブの内容は 表 6の通りです。

表 6：C タブ

Category	Show entries for	Item	該当 option
Source	Include file directories	登録 box	-I
	Defines	登録 box	-D
	Undefines	チェック box	-U
Object	Output	Global strict type check	-g
		Output directory	-O
	Debug	Generate debug information	-r
List		Generate list file	-L, -q, -i, -T, -t
Optimize		Optimization	-z, -s
Other		Miscellaneous options	-e, -c, -K , -C, -w -h
		User defined options	その他

オプションの詳細は、ICC コンパイラ・プログラミング・ガイド（icc740_jp.pdf）の 27～52 ページを参照してください。ICC コンパイラ・プログラミング・ガイドの GUI 画面は IAR Systems 社の Embedded Workbench のものです。

ICC740 のデフォルトのオプション設定は 表 7のとおりです。

表 7：ICC740 のデフォルトオプション

オプション	説明
-z9	コードサイズ優先の最適化で最高レベルを指定します。
-e	拡張仕様(zpage, npage 等)を有効にします。
-K	“//”コメントを有効にします。
-r	デバッグ情報を出力します。
-ml	メモリモデルを Large モデルに設定します。
-mt	メモリモデルを Tiny モデルまたは 0 page only モデルに設定します。
-v0	MUL/DIV を持つ MCU を選択します。（一般的な MCU です。）
-O	オブジェクトファイルフォルダを設定します。 $(CONFIGDIR)はDebugフォルダまたはReleaseフォルダを指定します。
-h	多重割り込みをサポートします。

4.2.1. ヘッダファイルの登録

740 ファミリ用 C コンパイラパッケージ V.1.01 Release 02 では、ソースファイルと同じフォルダにあるヘッダファイルは自動的に登録されます。
他のフォルダにある場合は”Include file directories”を使用して、該当するフォルダを登録してください。

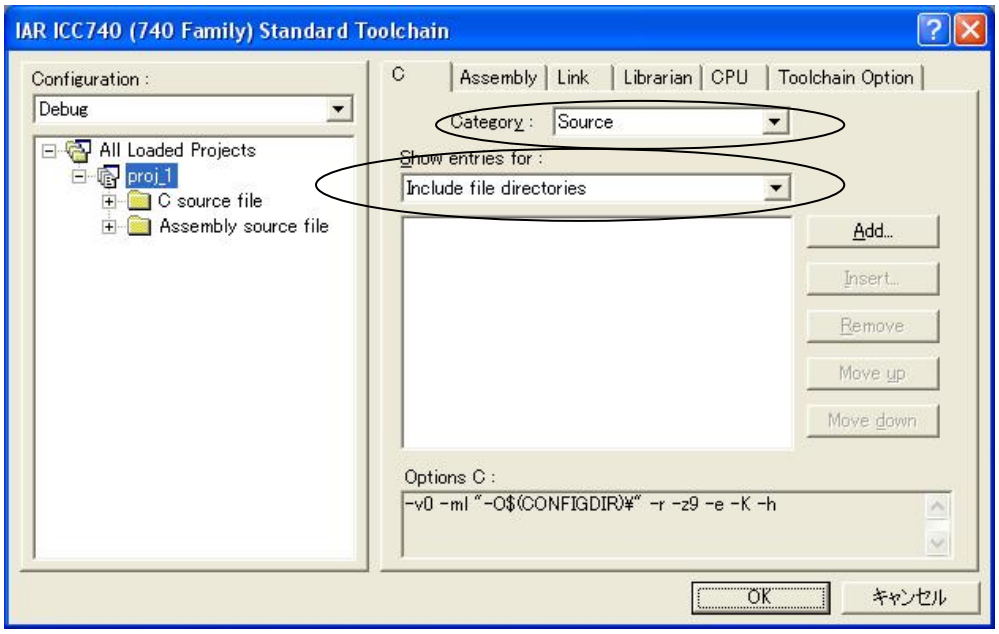


図 23：[Toolchain]ダイアログボックス：C タブの Category:Source

4.2.2. リストファイルの作成

デフォルトではリストファイルを作成しませんので、必要な場合は 図 24の設定を行ってください。

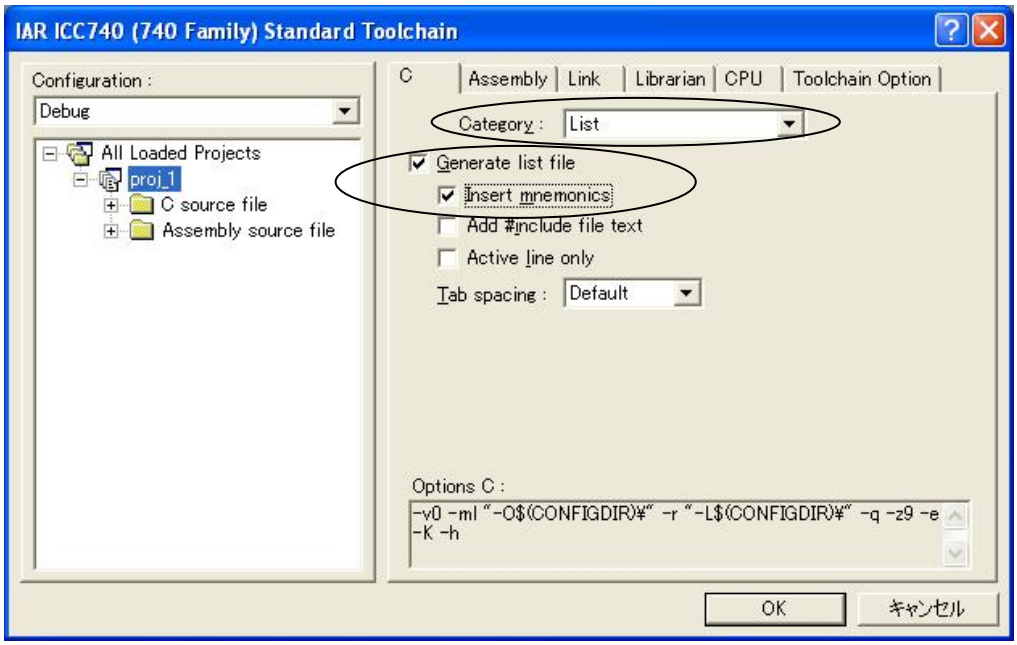


図 24 : C タブの Category : List

4.3. A740 オプションの設定

オプションの設定はAssemblyタブで行います。Assemblyタブの内容は 表 8の通りです。

表 8 : Assembly タブ

Category	Show entries for	Item	該当 option
Source	Include file directories	登録 box	-I
	Defines	登録 box	-D
	Undefines		-U
	Output	Output directory	-O
	Debug	Generate debug information	-r
List		Generate list file	-L, -i, -t
Other		Miscellaneous options	-s, -w
		User defined options	

オプションの詳細は、IMA アセンブラ・プログラミング・ガイド（A740_jp.pdf）の 21～34 ページを参照してください。740 Assembler, Linker, and Librarian Programming Guide の GUI 画面は IAR Systems 社の Embedded Workbench のものです。

A740 のデフォルトのオプション設定は 表 9のとおりです。

表 9：A740 のデフォルトオプション

オプション	説明
-uN	16 ビットアドレッシングを設定します。 Large モデル選択時に登録します
-v0	MUL/DIV を持つ MCU を選択します。（一般的な MCU です。）
-r	デバッグ情報を出力します。
-O$(CONFIGDIR)¥	オブジェクトファイル名を設定します。

4.3.1. リストファイルの作成

デフォルトではリストファイルを作成しませんので、必要な場合は 図 25の設定を行ってください。

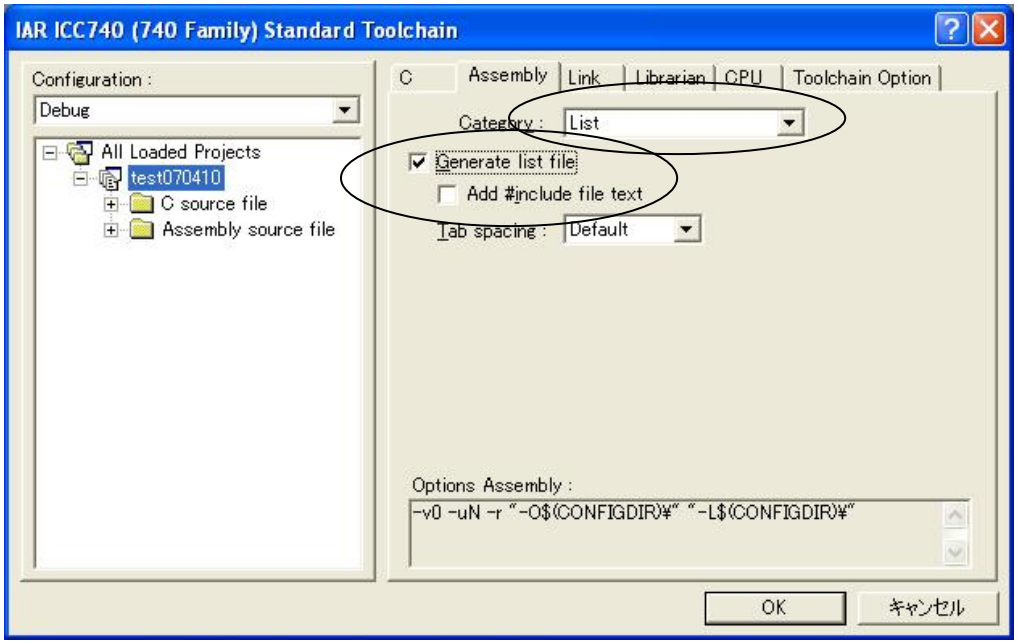


図 25：Assembly タブの Category：List

4.4. XLINKオプションの設定

オプションの設定はlinkタブで行います。linkタブの内容は 表 10の通りです。

表 10：link タブ

Category	Show entries for	Items	該当 option
Memory			-Z
		Use segment definition subcommand file	-f
Input	Library files	登録 box	無し
	Object files	登録 box	無し
	Defines	登録 box	-D
Output		Type for output file	-F
		Fill unused code memory	-H
		Output file path	-O
List		Generate list file	-L, -x
Other		Miscellaneous options	-z, -w
		User defined options	その他
Subcommand file		Use external subcommand file	-f

オプションの詳細は、IMA アセンブラ・プログラミング・ガイド（A740_jp.pdf）の 141～169 ページを参照してください。IMA アセンブラ・プログラミング・ガイドの GUI 画面は IAR Systems 社の Embedded Workbench のものです。

XLINKのデフォルトのオプション設定は 表 11のとおりです。

表 11：XLINK のデフォルトオプション

オプション	説明
-C cl7400l.r31	ライブラリを指定します。
-c740	740 ファミリを指定します。
-Fmotolora	モトローラ形式ファイルを指定します。 通常は IEEE695 形式ファイルも作成します。
-o$(CONFIGDIR)¥$(PROJECTNAME).mot	オブジェクトファイル名を設定します。
-l$(CONFIGDIR)¥$(PROJECTNAME).map	マップファイルを出力します。
-xmnos	マップファイルに相互参照リストを出力します。
-Z...	セグメントの配置を指定します

V.1.01 Release 02 では、XLINKのオプションのGUI化を行い、lnk740.xclを使用しない形式にしました(図 26)。

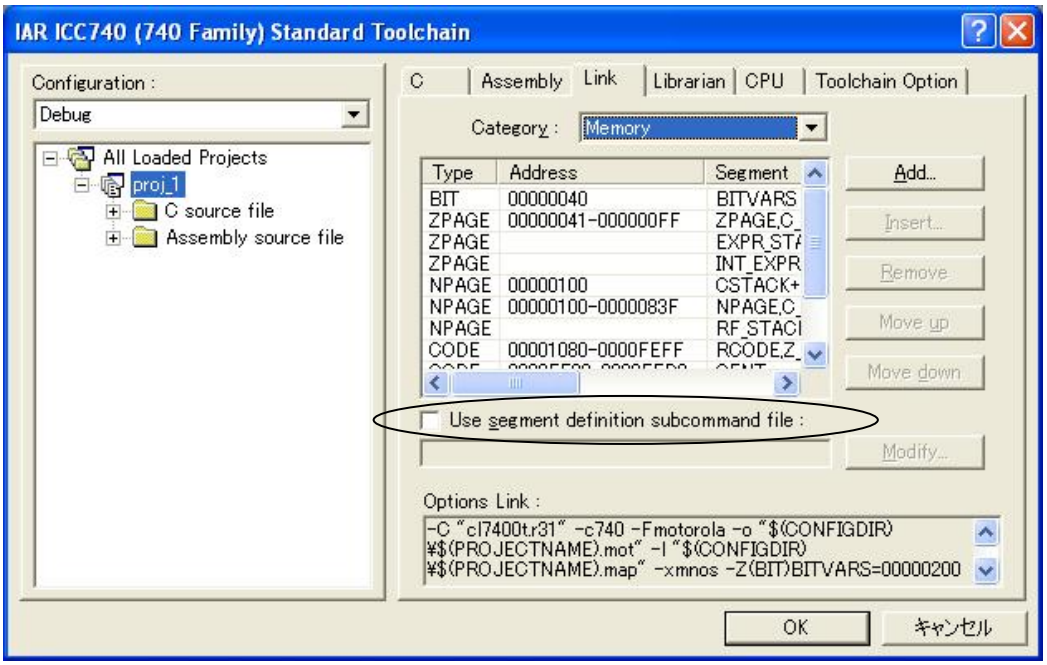


図 26 : Link タブ

lnk740.xcl を使用する場合は"Use segment definition subcommand file:"にチェックを入れてください。

5. プロジェクトの開発

5.1. ソースファイルの作成および登録

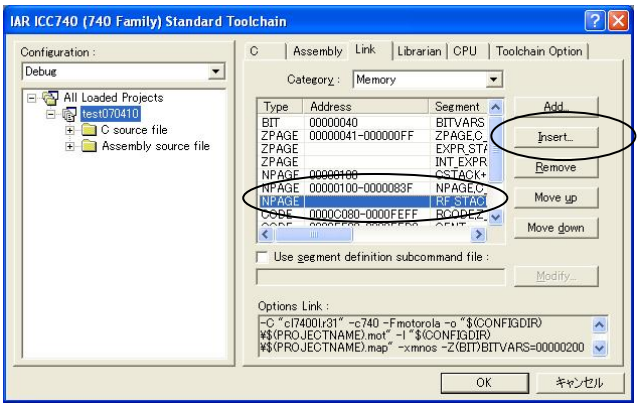
ソースファイルは[ファイル]メニューの[新規作成]によりエディタが起動されます。
作成したソースファイルはプロジェクトに登録する必要があります。プロジェクトディレクトリに置いただけでは登録されません。[プロジェクト]メニューの[ファイルの追加...]で登録してください。

5.2. セグメントの追加

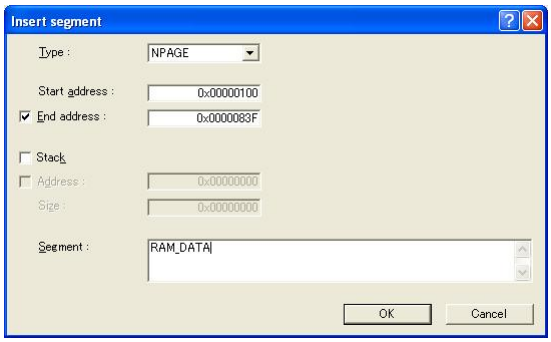
アセンブリ言語ソースファイル等で新しいセグメントを作成している場合は、そのセグメントの配置を Link タブの Category:Memory に設定する必要があります。
以下に例を示します(図 27)。

```
< sample.s31 >
            ...
RSEG RAM_DATA
BLKB 10
            ...
```

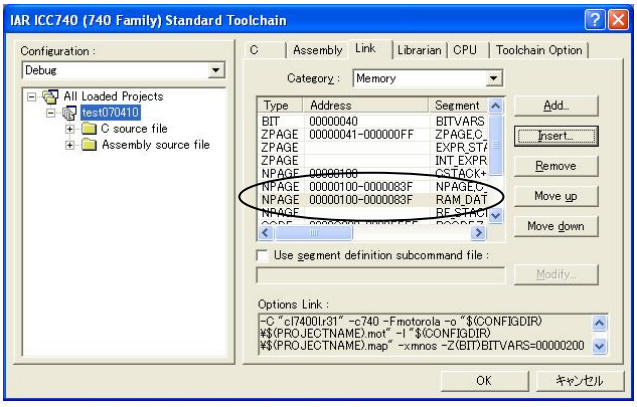
アセンブリ言語ソースファイル



Category:Memory<Before>



[Insert Segment]ダイアログボックス



Category:Memory<After>

図 27：Category:Memory の設定

5.3. セグメントの変更

セグメントのアドレスまたはサイズを変更する場合は、Link タブの Category:Memory で該当セグメントをダブルクリックして変更してください。
[Modify Segment]ダイアログボックス(図 28)にて該当項目を変更してください。

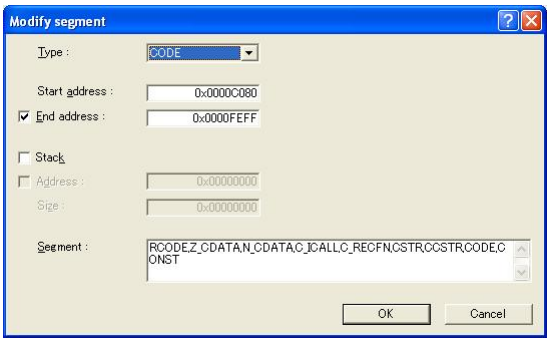


図 28：[Modify Segment]ダイアログボックス

6. プロジェクトのビルド

ソースファイルの作成および登録が完了しましたら、ビルドによりアブソリュートモジュールを作成します。 [ビルド]、[すべてをビルド]、または[ファイルのコンパイル]ボタンをクリックしてください。

図 29 : [ビルド]、[すべてをビルド]、[ファイルのコンパイル]ボタン

エラーが発生した場合は、メッセージに従ってプロジェクトの修正をしてください。

6.1. CコンパイラICC740 およびアセンブラA740 のエラー

C コンパイラ ICC740 およびアセンブラ A740 の実行時にエラーが発生した場合は、該当ファイルを修正してください。
行番号を表示しているエラーでは、その行をダブルクリックすることで、エラー箇所を表示します。
これにより、エラー発生箇所の特定が行いやすくなります。

6.2. リンカXLINKのエラー

リンカ XLINK の実行時にエラーが発生した場合は、メッセージに従ってプロジェクトの修正をしてください。
ここでは、よく発生するエラーについて対処方法を説明します。

- Error[e16]
  セグメントにデータが入りきらない場合、XLINK が以下のエラーを出力します。
  Error[e16]: Segment *セグメント* is too long for segment definition.
  この場合は、セグメントのサイズの変更またはセグメントの移動を行ってください。
  なお、以下のセグメントはゼロページから移動できません。
  ZPAGE, Z_UDATA, Z_IDATA, C_ARGZ, EXPR_STACK, INT_EXPR_STACK
  Z_UDATA, Z_IDATA, C_ARGZ セグメントは変数が配置されているセグメントです。
  変数に npage を付加することにより、該当変数を N_UDATA, N_IDATA, C_ARGN セグメントに移動することが可能です。但し、参照する側の外部宣言およびプロトタイプ宣言にも npage を忘れないようにしてください。

- Error[e18]

Nページに配置したデータを、ゼロページアドレッシングモードでアクセスした場合、以下のエラーが発生します。

Error[e18]: Range error in (*モジュール*), segment *セグメント* at address *アドレス*. Value *値*, in tag *タグ*, is out of bounds (0x0-0xff)

この場合は N ページのデータをゼロページに配置するか、または参照方法を変更してください。

ゼロページに配置：

C ソースでは、定義時に zpage を付加してください。

アセンブラソースでは、該当シンボルがゼロページに配置されるようにしてください。

N ページに配置した場合の参照方法：

C ソースでは、外部宣言時に npage を付加してください。関数引数に npage を指定している場合は、プロトタイプ宣言にも npage が必要です。

```
extern npage int n_i1;
void func( char a, npage int n_b );
```

アセンブラソースでは、参照時に”np:”を付加してください。

```
lda   np:n_data
```

- Error[e27]

同名のシンボル（含 C 言語の変数および関数）が、複数のファイルで定義されている場合、以下のエラーが発生します。

Error[e27]: Entry " *シンボル* " in module *モジュール 1* (*ファイル 1*) redefined in module *モジュール 2* (*ファイル 2*)

この場合は、該当ファイルを調べて、シンボル名が重複しないように変更してください。

- Error[e46]

未定義シンボルがある場合、以下のエラーが発生します。

Error[e46]: Undefined external " *シンボル* " referred in *モジュール* (*ファイル*)

この場合は、シンボル名の確認を行ってください。

6.3. リンカXLINKの注意事項

- 割り込み処理に関する注意事項
  割り込み処理実行中に呼び出す関数が、割り込み処理関数以外の関数からも呼び出される場合、リンク時に出力されるべき以下のウォーニングが出力されません。

  Warning[w16]: Function "name" in module (file) is called from
  two function trees (with roots name1 and name2)

  ICC740 では、関数のローカル変数(関数引数および auto 変数)を静的に配置しています。そのため、ローカル変数を持つ関数を割り込み処理関数と、割り込み処理以外の関数の両方で使用するとローカル変数を破壊することがあります。
  割り込み処理関数と割り込み処理以外の関数で同じ関数を使用しないでください。

  発生例：

```
------------------------------------------------------------------
void func2( int );

interrupt[2] void   intr_1(void)
{
    func2( 2 );       /* func1()で、func2()を実行中に割り込みが発生 */
                      /* すると、割り込みから復帰した func2()の      */
                      /* ローカル変数が破壊される                    */
}

void func1( void )
{
    func2( 1 );
}

void main( void )
{
    func1();
}
------------------------------------------------------------------
```

- 未定義関数に関する注意事項

  未定義関数を呼び出した場合、リンク時に出力されるべき以下のエラーが出力されません。

  Error[e46:] Undefined external "external" referred in module(file)

  未定義関数を呼び出していないかどうか確認してください。

  呼び出している場合はその関数を定義してください。

  発生例：

```
---------------------------------------------------------------------
void func3( int );

void main( void )
{
    func3();            /* 未定義関数 */
}
---------------------------------------------------------------------
```

7. プロジェクトのデバッグ

ビルドが完了したプロジェクトでは、Debug フォルダにアブソリュートファイルが生成されます。
デバッグを行う場合、セッションから該当するデバッガを選択してください（図 30）。

図 30：セッション

該当するデバッガがインストールされている場合、デバッガの初期化ダイアログがオープンしますので、各種値を設定して[OK]ボタンをクリックしてください。
740 シミュレータの場合は、図 31のダイアログとなります。

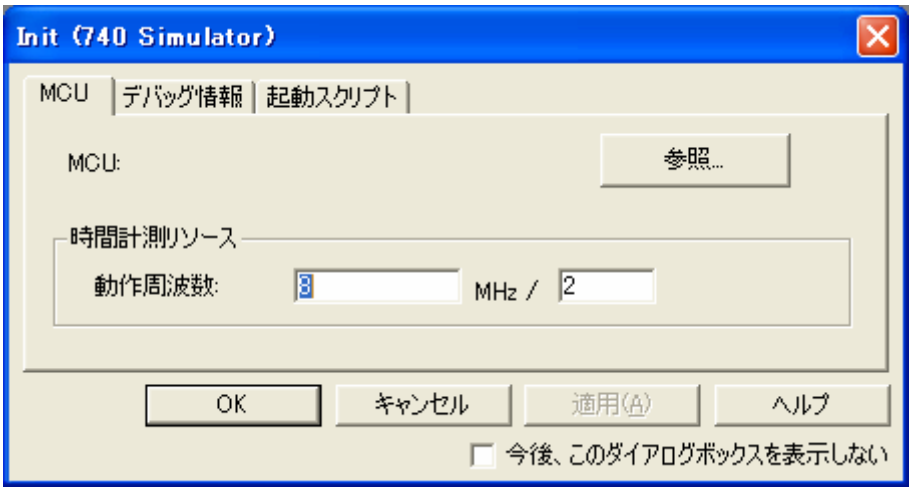


図 31：740 シミュレータの Init ダイアログ

MUC ファイルは[参照]ボタンにより選択してください。ターゲットマイコンに該当する MCU ファイルが見つからない場合は、「740 シミュレータデバッガ V.1.2 ユーザズマニュアル」の「4.3 MCU ファイルの作成」を参照してください。

セッションの変更が完了しましたら、プログラムのダウンロードを行います。
[デバッグ]メニューの[ダウンロード]により行ってください。

8. ヘキサファイルの作成

V.1.01 Release 02 では、通常、アブソリュートロードモジュールとして IEEE695 形式ファイルとモトローラ形式の２つのファイルを作成します。
アブソリュートロードモジュールの変更は[Toolchain]ダイアログボックスのLinkタブのOutputカテゴリ(図 32)で選択できます。

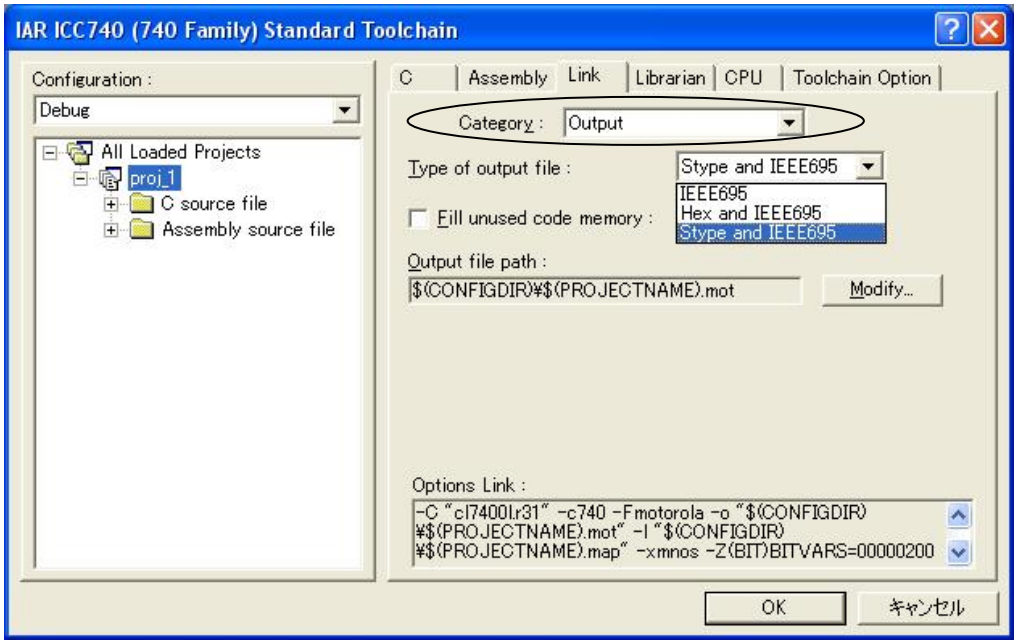


図 32：[Toolchain]ダイアログボックスの Link タブの Category:Output

表 12：Type of output file の内容

Type of output file	ファイル
IEEE695	IEEE695 形式ファイルを作成
Hex and IEEE695	IEEE695 形式ファイルとインテル Hex 形式ファイルを作成
Stype and IEEE695	IEEE695 形式ファイルとモトローラ形式ファイルを作成

9. リビジョンアップ時の注意

9.1. V.1.01 Release 01 からのリビジョンアップ

リビジョンアップアップ時は、V.1.01 Release 01 をアンインストールしてください。
V.1.01 Release 01 のプロジェクトタイプは使用できません。

Application
Application(Tiny)
Application for E8
Application for E8 (Tiny)

V.1.01 Release 01 のプロジェクトタイプ

9.2. V.1.01 Release 01 で作成したプロジェクトのコンバート

V.1.01 Release 01 のプロジェクトをV.1.01 Release 02 で使用する場合は以下のダイアログボックスを表示します(図 33)。メッセージに従って変更してください。

図 33：ツールチェインの変更メッセージ

V.1.01 Release 02 では、マイコン指定(-c740)、ライブラリ指定(-C cl7400l.r31 等)はデフォルトで設定しますので、lnk740.cxl ファイルから削除してください。

lnk740.xclの指定はLinkタブのOtherカテゴリに設定してあります(図 34)。

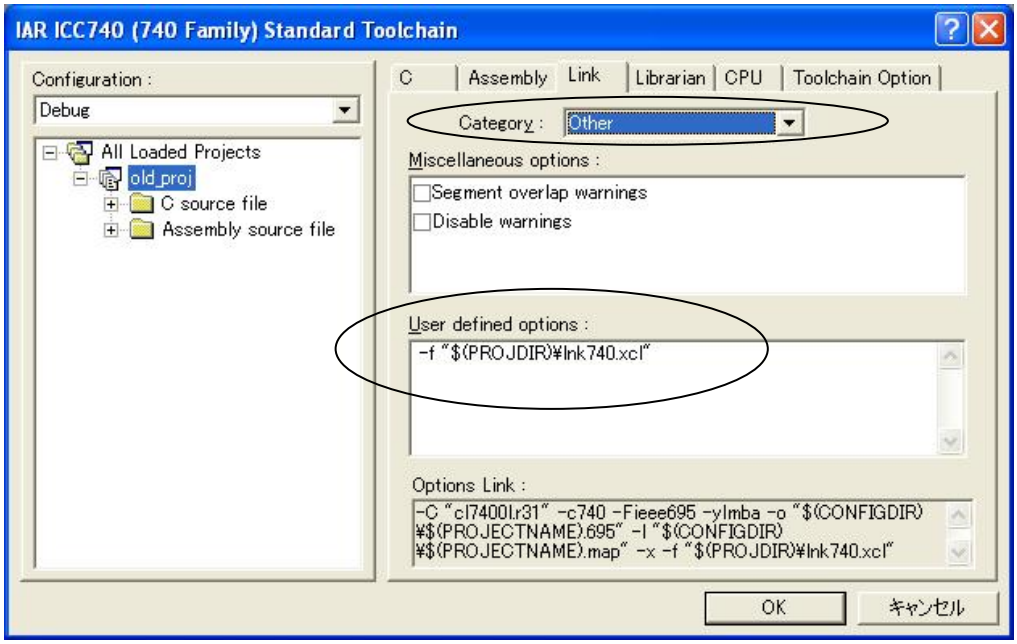


図 34：ツールチェイン変更後の lnk740.xcl ファイルの設定箇所

なお、ツールチェイン変更後のLinkタブのMemoryカテゴリは以下の通りとなります(図35)。

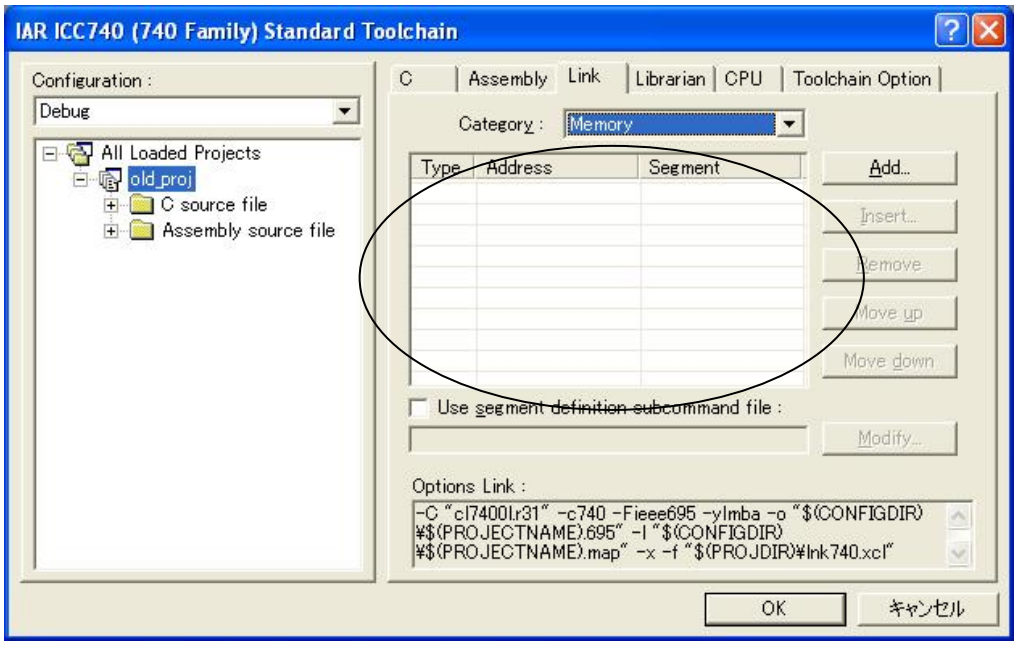


図 35：ツールチェイン変更後のメモリカテゴリ

セグメント配置は lnk740.xcl ファイルに設定があるため、この状態で使用してください。Memoryカテゴリでセグメント配置を行う場合は 図 35のlnk740.xclファイルの設定を削除してください。

10. cstartup.s31 とlnk740.xclファイルの編集

新規プロジェクト作成で[Empty Application]を選択した場合等、ICC740 に付属の cstartup.s31 と lnk740.xcl 使用する場合は、これらを編集する必要があります。

10.1. cstartup.s31 の編集

cstartup.s31 では、表 13の項目を必要に応じて編集します。

表 13：cstartup.s31 の編集項目

編集項目	要因	該当行
スタックページ	スタック領域	137 行
割り込みベクトル領域	ターゲットマイコン	375 行

10.1.1. スタックページの変更

137 行で、スタックページの設定を行っています。ここでは 3803 グループの CPU モードレジスタに対してスタックページを 1 ページに設定しています。

LDM #0CH, 3BH ; set stack page：3803 Group

ターゲットマイコンに合わせてスタックページを設定してください。
スタックページをゼロページに設定した場合は、lnk740.xcl ファイルの CSTACK セグメントの設定を変更する必要があります。
なお、CPU モードレジスタの他のビットはターゲットマイコンに合わせて設定してください。

10.1.2. 割り込みベクトル領域の変更

375 行で、割り込みベクトル領域のサイズ設定を行っています。ここでは 3803 グループの割り込みベクトル領域のサイズを設定しています。

BLKB 0FFFEH - 0FFDCH - 2 ; 3803 Group

この BLKB で設定する領域ではリセットベクトル分を引いてください（上記の-2 が該当します)。
なお、割り込みベクトル領域の先頭アドレスの設定は lnk740.xcl ファイルで行います。
割り込みベクトル領域に直接、各割り込みのベクトルを記述する方法もあります。
この場合は全てのベクトルを登録してください。使用しない割り込みはリセットベクトルと同じアドレスにする等の設定を行い、不定アドレスへジャンプしないよう対策を行って

ください。

```
?CSTARTUP_INTVEC:
       WORD init_C    ; +0x00 : BRK
       WORD init_C    ; +0x02 : AD_SIO3T
       WORD init_C    ; +0x04 : INT4_CNTR2
       WORD init_C    ; +0x06 : INT3
       WORD Int2      ; +0x08 : INT2
       WORD init_C    ; +0x0a : SIO2_TimerZ
       WORD init_C    ; +0x0c : CNTR1_SIO3R
       WORD init_C    ; +0x0e : CNTR0
       WORD init_C    ; +0x10 : Timer2
       WORD Timer1    ; +0x12 : Timer1
       WORD init_C    ; +0x14 : TimerY
       WORD init_C    ; +0x16 : TimerX
       WORD init_C    ; +0x18 : SIO1T
       WORD init_C    ; +0x1a : SIO1R
       WORD init_C    ; +0x1c : INT1
       WORD Int0      ; +0x1e : INT0_TimerZ
?CSTARTUP_RESETVEC:
       WORD init_C    ; +0x20 : reset
ENDMOD init C
```

10.2. lnk740.xclファイルの編集

ターゲットマイコン、メモリモデル等の設定に応じて、lnk740.xcl ファイルの編集が必要となります。

表 14：Lnk74.xcl の編集項目

編集項目	要因	lnk740.xcl の該当行
スタック領域	メモリモデル、ターゲットマイコン	60
ゼロページ先頭アドレス	ターゲットマイコン	38
N ページ終了アドレス	ターゲットマイコン	65
ROM 領域アドレス	ターゲットマイコン	77
割り込みベクトル	ターゲットマイコン	91
ライブラリ	プロセッサ・グループ	110

10.2.1. スタック領域の変更

740 ファミリでは、スタック領域はゼロページ（00h～FFh）または 1 ページ（100h～1FFh）を選択して使用します。
ICC740 では、このスタック領域を CSTACK セグメントとして使用します。
740 ファミリ C コンパイラパッケージの標準では、1 ページの 100h～13Fh までを使用する設定となっています。

-Z(NAPGE)CSTACK+40=100

変更例を示します。

例1） スタックの使用領域を 1 ページの 120h～14Fh までを使用する場合は、以下のように変更してください。

-Z(NAPGE)CSTACK+30=120

例 2）スタック領域をゼロページに変更する場合は、以下のように変更してください。

-Z(ZPAGE)CSTACK+40

ゼロページには、SFR 領域およびゼロページに配置しなくてはならないセグメントがあります。上記の記述は、54 行目に記述している INT_EXPR_STACK セグメント以降の 40 h バイトを使用する設定となります。

なお、スタック領域をゼロページに変更する場合は、cstartup.s31 ファイルの変更も必要となります。

10.2.2. ゼロページ先頭アドレスの変更

ICC740 では、RAM 領域をゼロページと N ページ（100h 以降）に分けて設定します。ゼロページの設定には”-Z(ZPAGE)”を指定します。

-Z(ZPAGE)ZPAGE,C_ARGZ,Z_UDATA,Z_IDATA=41-FF

上記の設定は、SFR 領域（0h～40h）後の 41h 番地から FF h 番地までの RAM 空間に、ZPAGE、C_ARGZ、Z_UDATA、および Z_IDATA セグメントを配置する設定です。
これらのセグメントは ICC740 が使用するセグメントですので、削除しないでください。
変更例を示します。

例）7220 グループ等 SFR 領域が 0h 番地から設定されていないマイコンでは以下のように変更してください。

-Z(ZPAGE)ZPAGE,C_ARGZ,Z_UDATA,Z_IDATA=0-BF

7200 グループでは、SFR 領域が C0h 番地から開始されますので、BFh 番地までとします。

10.2.3. Nページ終了アドレスの変更

N ページの設定には"-Z(NPAGE)"を指定します。
N ページの終了アドレスは､ターゲットマイコンの RAM の終了アドレスを設定してください。

-Z(NPAGE)NPAGE,C_ARGN,N_UDATA,N_IDATA,ECSTR=100-43F

上記の設定は、100h 番地から 43F h 番地までの RAM 空間に、NPAGE、C_ARGN、N_UDATA、N_IDATA、および ECSTR セグメントを配置する設定です。
これらのセグメントも ICC740 が使用するセグメントですので、削除しないでください。
１ページに CSTACK セグメントを配置している場合は、NPAGE セグメントは CSTACK セグメント後のアドレス（標準の設定では 140h 番地）から配置されます。
終了アドレスはターゲットマイコンの RAM 領域の最終アドレスを設定してください｡この設定により、RAM 領域のオーバーフローが検査できます。
38C2 グループ等、SFR 領域が N ページにも存在するマイコンでは、その領域を外してください。

10.2.4. ROM領域アドレスの変更

ターゲットマイコンに合わせて ROM 領域の設定を行ってください。一般プログラム領域、スペシャルページの設定を行います。
一般プログラム領域の設定を行ってください。

-Z(CODE)RCODE,Z_CDATA,N_CDATA,C_ICALL,C_RECFN,CSTR,
CCSTR,CODE,CONST=C080-FEFF

上記の設定は、ROM 領域の先頭アドレス C080 h からスペシャルページ前 FEFF h までの ROM 空間に、ROM 領域用セグメントを配置する設定です。
なお、予約 ROM 領域を持つマイコンでは、先頭アドレスをその領域以降にしてください。
終了アドレスはスペシャルページの手前まで、または、割り込みベクトル領域の手前までを指定してください。
スペシャルページの終了アドレスは割り込みベクトル領域の手前までを指定してください。
C_FNT セグメントがスペシャルページ領域を示します。
この設定により、割り込みベクトル領域へのオーバーラップが検査できます。

-Z(CODE)C_FNT=FF00-FFDB

10.2.5. 割り込みベクトル領域の変更

割り込みベクトル領域の変更を行います。
ICC740 では INTVEC セグメントが割り込みベクトル領域を示します。

-Z(CODE)INTVEC=FFDC-FFFD

上記は、3803 グループの割り込みベクトル領域の設定です。ターゲットマイコンに合わせて設定してください。

10.2.6. ライブラリの削除

V.1.01 Release 02 では、CPU の選択に応じてライブラリを指定します。
lnk740.xcl のライブラリ指定箇所は削除してください。

```
-! -C cl7400l -!
```

10.2.7. セグメントの追加

アセンブラソースファイル等で新しいセグメントを作成している場合は、そのセグメントの配置を lnk740.xcl ファイルに設定する必要があります。
以下に例を示します。

```
< sample.s31 >
        …
RSEG RAM_DATA
BLKB 10
        …
RSEG ROM_DATA
BYTE ‘Please enter your name’
        …
```

```
< lnk740.xcl >
        …
-Z(NPAGE)NPAGE,C_ARGN,N_UDATA,N_IDATA,ECSTR=100-43F

-Z(NPAGE) RAM_DATA=100-43F
        …
-Z(CODE)RCODE,Z_CDATA,N_CDATA,C_ICALL,C_RECFN,CSTR,CCSTR,CODE,
CONST=C080-FEFF

-Z(CODE) ROM_DATA= C080-FEFF
        …
```

上記では、RAM_DATA セグメントが、ECSTR セグメントの後に、また、ROM_DATA セグメントが CONST セグメントの後に配置されます。

MEMO

740ファミリ用
Cコンパイラパッケージ V.1.01
ユーザーズマニュアル

発行年月日　2007年6月16日　　Rev.1.01

発行　　株式会社 ルネサス テクノロジ　営業企画統括部
〒100-0004　東京都千代田区大手町2-6-2

編集　　株式会社 ルネサス ソリューションズ　ツール開発部

740 ファミリ用　C コンパイラパッケージ V.1.01
ユーザーズマニュアル

ルネサス エレクトロニクス株式会社
神奈川県川崎市中原区下沼部1753 〒211-8668

RJJ10J1409-0101