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。

Application Note

RENESAS

HI7700/4 構築ガイド

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　2003.03

ご注意

1. 本製品(ソフトウエア製品及びその関連ソフトウエア製品を含む。以下、同じ。)の使用に際しては、「外国為替及び外国貿易法」等、技術輸出に関する日本及び関連諸国の関係法規の遵守が必要となります。
2. 弊社は、本製品の使用に際しては、弊社もしくは第三者の特許権、著作権、商標権、その他の知的所有権等の権利に関し、別途、個別の契約書等(マニュアルの記載を含む。以下、同じ。)にて弊社による明示的な許諾がある場合を除き、その保証または実施権の許諾を行うものではありません。また本製品を使用したことにより第三者の知的所有権等の権利に関わる問題が生じた場合、弊社はその責を負いませんので予めご了承ください。
3. 本製品およびその仕様、またはマニュアルに記載されている事柄については、将来、事前の予告なしに変更することがありますので、最終的な設計、ご購入、ご使用に際しましては、事前に最新の製品規格または仕様書(マニュアルを含む)をご確認ください。
4. 本製品の使用(マニュアル記載事項に基づくものも含む)により直接または間接に生ずるいかなる損害についても、弊社は一切の責任を負いません。また、本製品の配布に使用される搭載機器や媒体が原因の損害に対しましても、弊社は一切の責任を負いません。
5. 本製品を、宇宙、航空、原子力、燃焼制御、運輸、交通、各種安全装置、ライフサポート関連の医療機器等のように、特別な品質・信頼性が要求され、その故障や誤動作が直接人命を脅かしたり、人体に危害を及ぼす恐れのある用途向けには使用できません。お客様の用途がこれに該当するかどうか疑問のある場合には、事前に弊社営業担当迄ご相談をお願い致します。
6. 本製品を使用してお客様のシステム製品を設計される際には、通常予測される故障発生率、故障モードをご考慮の上、本製品の動作が原因での事故、その他の拡大損害を生じないようにフェールセーフ等の充分なシステム上の対策を講じて頂きますようお願い致します。
7. 本製品およびマニュアルの著作権は弊社が所有しております。お客様は、弊社から提供された本製品を、別途、個別の契約書等にて定める場合を除き、いかなる場合においても全体的または部分的に複写・解析・改変することはできないものとします。
8. お客様は、別途、個別の契約書等にて定める場合を除き、本製品のマニュアルの一部または全部を無断で使用、複製することはできません。
9. 弊社は、本製品を1台のコンピュータで使用する権利をお客様に対してのみ許諾します。よって、本製品を第三者へ譲渡、貸与、賃借することは許諾しないものとします。但し、別途、個別の契約書等にて定められる場合はその条件に従います。
10. 本製品をはじめ弊社半導体およびその関連製品についてのお問い合わせ、ご相談は弊社営業担当迄お願い致します。

製品に関する一般的注意事項

1. NC 端子の処理

【注意】NC端子には，何も接続しないようにしてください。

NC(Non-Connection)端子は、内部回路に接続しない場合の他、テスト用端子やノイズ軽減などの目的で使用します。このため、NC端子には、何も接続しないようにしてください。

2. 未使用入力端子の処理

【注意】未使用の入力端子は、ハイまたはローレベルに固定してください。

CMOS製品の入力端子は、一般にハイインピーダンス入力となっています。未使用端子を開放状態で動作させると、周辺ノイズの誘導により中間レベルが発生し、内部で貫通電流が流れて誤動作を起こす恐れがあります。

未使用の入力端子は、入力をプルアップかプルダウンによって、ハイまたはローレベルに固定してください。

3. 初期化前の処置

【注意】電源投入時は，製品の状態は不定です。

すべての電源に電圧が印加され、リセット端子にローレベルが入力されるまでの間、内部回路は不確定であり、レジスタの設定や各端子の出力状態は不定となります。この不定状態によってシステムが誤動作を起こさないようにシステム設計を行ってください。

リセット機能を持つ製品は、電源投入後は、まずリセット動作を実行してください。

4. 未定義・リザーブアドレスのアクセス禁止

【注意】未定義・リザーブアドレスのアクセスを禁止します。

未定義・リザーブアドレスは、将来の機能拡張用の他、テスト用レジスタなどが割り付けられています。

これらのレジスタをアクセスしたときの動作および継続する動作については、保証できませんので、アクセスしないようにしてください。

はじめに

このガイドは、HI7700/4 の構築方法について説明します。

HI7700/4 上でタスクとして登録したアプリケーションプログラムを実行するにあたり、デバッグ初期の段階においてターゲットボードとして日立超 LSI システムズ製 Solution Engine®を使用し、デバッガとして E10A エミュレータの HDI を使用するものとします。HI7700/4 に関する詳細は、HI7000/4 シリーズ（HI7000/4,HI7700/4,HI7750/4）ユーザーズマニュアル（以下、HI7000/4 シリーズユーザーズマニュアルと呼びます）を参照してください。また、アプリケーションプログラムの作成、HI7700/4 とのリンケージには、SuperH™ RISC engine C/C++コンパイラパッケージ（以下 SHC/C++コンパイラと呼びます）、および SuperH™ RISC engine C/C++コンパイラパッケージ付属の統合開発環境 HEW（Hitachi Embedded Workshop：以下、HEW と呼びます）を使用するものとします。

これらのターゲットボード、エミュレータおよびコンパイラの環境を使用することを前提に、マルチタスク環境下の先頭タスクが実行されるまでのプログラムの変更・追加の方法、構築手順について詳しく解説します。

【関連マニュアル】

- HI7000/4 シリーズ（HI7000/4,HI7700/4,HI7750/4）ユーザーズマニュアル
- SuperH™ RISC engine C/C++コンパイラユーザーズマニュアル
- SuperH™ RISC engine アセンブラユーザーズマニュアル
- H シリーズリンケージエディタ、ライブラリアン、オブジェクトコンバータユーザーズマニュアル
- Hitachi Embedded Workshop（HEW）ユーザーズマニュアル
- SH7729R Solution Engine®（MS7729RSE01）概説書
- 使用する SuperH™ マイコンのハードウェアマニュアル、プログラミングマニュアル

目次

1. 概要 1
1.1 概説 1
1.2 システム構成 1
1.3 準備するもの 2
2. アプリケーションプログラムの作成 5
2.1 CPU 初期化ルーチンの作成 6
2.2 タスクの作成 12
2.2.1 メインタスク 13
2.2.2 LED タスク 15
2.3 割り込みハンドラの作成 16
2.3.1 初期化モジュールの作成 17
2.3.2 割り込みハンドラの作成 20
3. コンフィギュレーション 21
3.1 コンフィギュレータの起動 21
3.2 割り込みマスクレベル 22
3.3 タスクの登録 22
3.4 割り込みハンドラの登録 25
3.5 初期化ルーチンの登録 28
3.6 イベントフラグ情報の登録 32
3.7 コンフィギュレーションファイルの生成 33
3.8 HEW によるビルド 33
3.8.1 HEW の起動 34
3.8.2 構築ファイルの定義 35
3.8.3 リンケージアドレスの変更 37
3.8.4 ビルド 41
3.9 パラメータチェック機能無しでの構築 42
4. E10A によるダウンロードと実行 43
4.1 Solution Engine®の初期化 43
4.2 アプリケーションプログラムのダウンロード 44
4.3 アプリケーションプログラムの実行 46

図目次

図 1-1　ハードウェア構成 1
図 1-2　HI7700/4 インストール直後のフォルダ構成 3
図 2-1　アプリケーションプログラム関連図 5
図 2-2　作成するプログラム 6
図 2-3　CPU 初期化ルーチン作成手順 6
図 2-4　_hi_cpuasm（7729_cpuasm.src）の変更個所 7
図 2-5　_hi_cpuasm（7729_cpuasm.src）の変更個所 8
図 2-6　_hi_cpuasm（7729_cpuasm.src）の変更個所 9
図 2-7　_hi_cpuasm（7729_cpuasm.src）の変更個所 10
図 2-8　hi_cpuini（7729_cpuini.c）の変更個所 11
図 2-9　タスクの作成および登録手順 12
図 2-10　MainTask の改造内容概略 13
図 2-11　MainTask の改造内容 14
図 2-12　task7 の改造内容 15
図 2-13　初期化モジュールおよび割り込みハンドラの作成および登録手順 16
図 2-14　初期化モジュール作成手順 18
図 2-15　TMU1_ini(tmu1.c)作成内容 19
図 2-16　割り込みハンドラ作成手順 20
図 2-17　TMU1_int(tmu1.c)作成内容 20
図 3-1　本ガイドで登録する項目 21
図 3-2　コンフィギュレータ起動画面 22
図 3-3　タスク情報の画面 23
図 3-4　タスク情報変更画面 24
図 3-5　割り込み,CPU 例外ハンドラ一覧の画面 25
図 3-6　割り込み,CPU 例外ハンドラの定義画面 26
図 3-7　割り込みハンドラ定義後の定義画面 27
図 3-8　割り込みハンドラ定義後の一覧画面 27
図 3-9　初期化ルーチン一覧の画面 29
図 3-10　初期化ルーチンの登録画面 30
図 3-11　初期化ルーチン登録後の登録画面 31
図 3-12　初期化ルーチン登録後の一覧画面 31
図 3-13　イベントフラグ情報の画面 32
図 3-14　イベントフラグの生成画面 33
図 3-15　HEW の起動画面 34
図 3-16　カレントプロジェクトの設定画面 35
図 3-17　ファイル追加の手順 36
図 3-18　ファイル追加の手順 36
図 3-19　OptLinker の選択 37
図 3-20　OptLinker options 画面 38
図 3-21　OptLinker options 画面(__kernel_pon_sp) 38
図 3-22　OptLinker options 画面(__kernel_man_sp) 39

図 3-23　セクション定義画面 39
図 3-24　Build の手順 41
図 3-25　パラメータチェック機能を取り外す方法 42
図 4-1　HDI の起動画面 43
図 4-2　Reset Go メニュー 44
図 4-3　ダウンロード方法 45
図 4-4　ダウンロード終了画面 46
図 4-5　レジスタ情報 46
図 4-6　PC 値変更画面 47
図 4-7　プログラム実行画面 47

表目次

表 1-1　ソフトウェア構成 2
表 1-2　必要なハードウェア／ソフトウェア 2
表 2-1　割り込みの条件 17
表 3-1　割り込みマスクレベルの関係 22
表 3-2　リンクの種類 33
表 3-3　プロジェクトファイルの種類 34
表 3-4　セクションアドレスの変更内容 40

1. 概要

1.1 概説

HI7700/4 を用いてアプリケーションプログラムを動作させるための手順を以下に示します。

（1）アプリケーションプログラムの作成
（2）コンフィギュレータによる HI7700/4 への登録（コンフィギュレーション）
（3）HEW によるビルド（ビルド）
（4）ターゲットボードへの組み込みおよび実行（ダウンロード）

上記の手順に従い、ターゲットボード上でアプリケーションプログラムを実際に動作させるまでの作業についてサンプルプログラムを例に詳しく解説します。

1.2 システム構成

本ガイドでは、タスクおよび割り込みハンドラのサンプルプログラムを作成し、ターゲットボードで動作させます。

図 1-1 に、本ガイドで想定するハードウェア構成を示します。

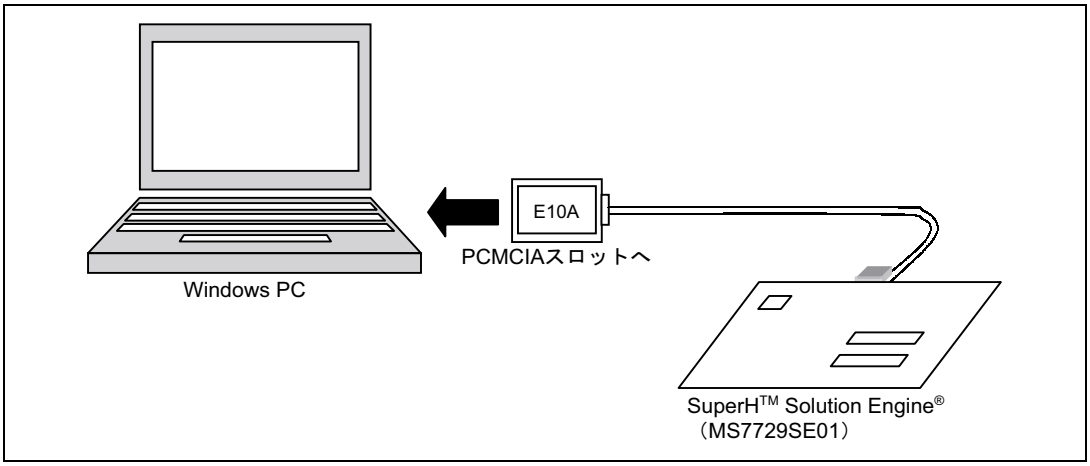


図1-1　ハードウェア構成

ソフトウェア構成を**表 1-1** に示します。

表1-1　ソフトウェア構成

項番	プログラム名	内容	種別	備考
1	CPU 初期化ルーチン	・バスコントローラの設定 ・ハードウェアの初期化	非タスク部	
2	メインタスク	・環境の初期化 ・初期化後、wai_flg により WAITING ・タイマ割り込みハンドラのイベントフラグセットにより WAITING が解除され、LED タスクを起動（sta_tsk）	タスク部	
3	LED タスク	・メインタスクにより起動され、LED を操作し、終了（LED が点灯していた場合は消灯、消灯していた場合は点灯）	タスク部	
4	タイマ割り込みハンドラ	・タイマ割り込みにより 1 秒毎に起動され、メインタスクイベントフラグにセット（set_flg）	非タスク部	

1.3　準備するもの

HI7700/4 を用いてアプリケーションプログラムを動作させるために、**表 1-2** に示すハードウェアおよびソフトウェアを準備してください。

表1-2　必要なハードウェア／ソフトウェア

項番	製　品　名	製品型名	メーカ	備考
1	Windows®パソコン	－	任意のメーカ	*1
2	SuperH™ Solution Engine®	MS7729RSE01	(株)日立超 LSI システムズ	
3	E10A エミュレータ	HS7729RKCM01H	(株)日立製作所	
4	SuperH™ RISC engine C/C++　コンパイラ	P0700CAS6-MWR	(株)日立製作所	*2
5	HI7700/4	HS0770ITI41SRE	(株)日立製作所	*3

【注】 *1　・ハードウェア環境：486DX2/66MHz 以上（Pentium®以上を推奨）を搭載した PC/AT 互換機
・対応 OS　：Windows®2000 、WindowsNT®4.0 、Windows®98 、Windows®95
・CD-ROM ドライブ
・PCMCIA カードスロット
・メモリ 32M バイト以上を推奨（Windows®2000 、WindowsNT®4.0 で使用する場合は 64M バイト以上を推奨）
・必要空きハードディスク容量：約 8 M バイト以上

*2　VER.6.0AR2 を使用します。(株)日立製作所以外に、(株)日立超ＬＳＩシステムズ製または日立ソフトウェアエンジニアリング(株)製のものをご使用いただいても構いません。

*3　評価用契約（オブジェクト）のものを使用することにしますが、既に量産用でご契約いただいている場合は、そちらをご使用いただいても構いません。

なお、Windows®パソコンに、E10A エミュレータの HDI、SuperH™ RISC engine C/C++コンパイラ、および HI7700/4（SHCV6 用）がインストール済みであることを前提に解説を進めます。また、本ガイドでは SH7729 を対象とします。

HI7700/4 をインストールした直後のフォルダ構成を図 1-2 に示します。

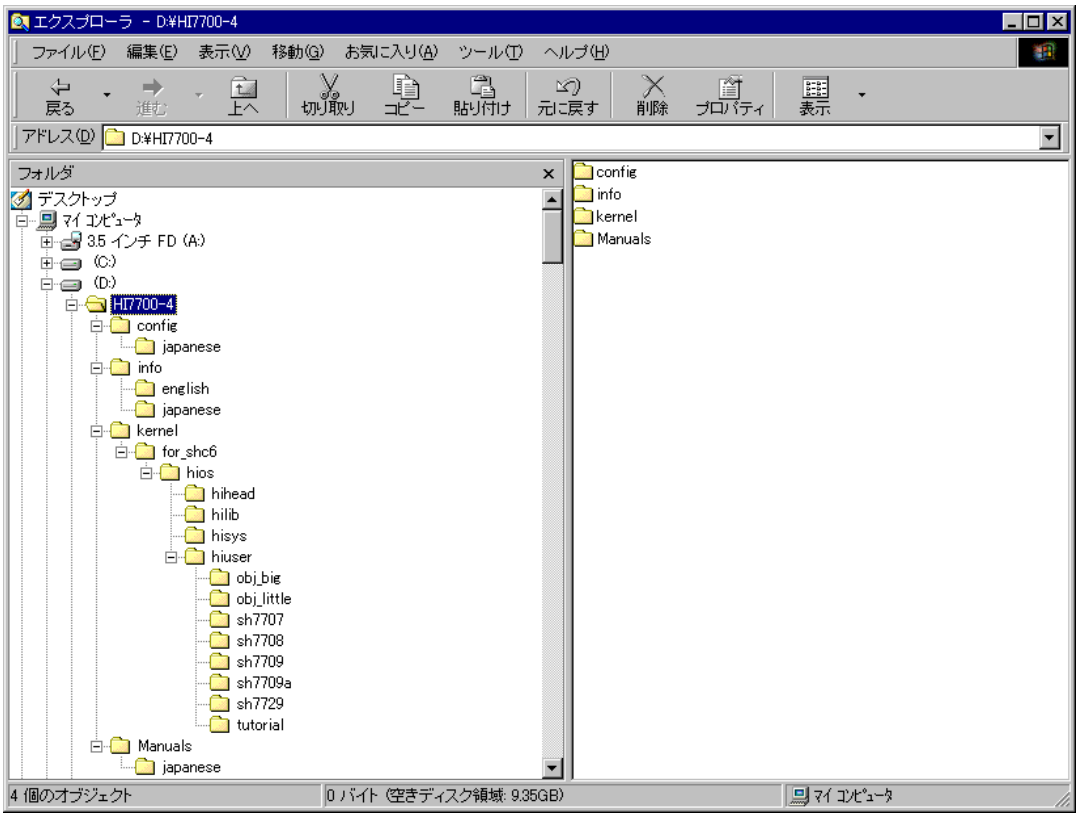


図1-2　HI7700/4 インストール直後のフォルダ構成

本ガイドでは、インストールドライブを D としていますが、ユーザ任意のドライブを使用してインストールしていただいて結構です。以後、解説の中で任意のフォルダを指す場合、「インストールフォルダ”任意のフォルダ名”」として表します。

2. アプリケーションプログラムの作成

HI7700/4 上で動作させるアプリケーションプログラムの作成方法について、以下に説明します。図 2-1 に、アプリケーションプログラムの関連を示します（太枠が、本ガイドで作成するプログラムです）。

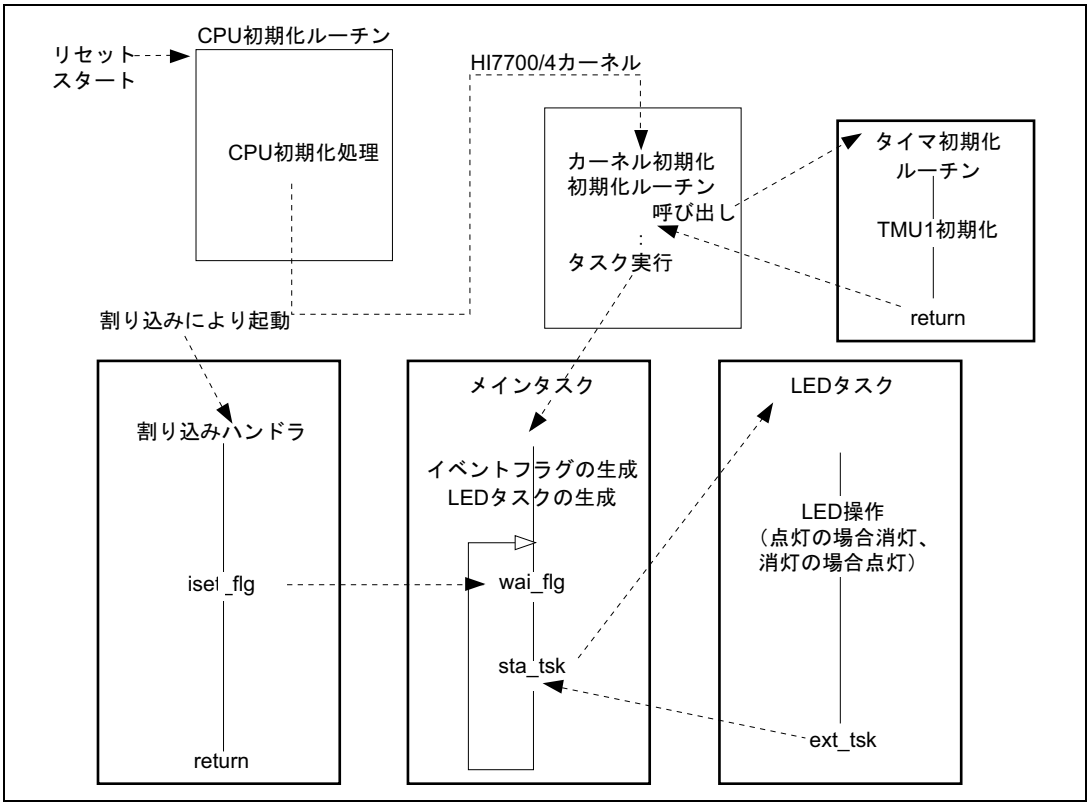


図2-1　アプリケーションプログラム関連図

図 2-2 に本ガイドで作成するプログラムを示します。

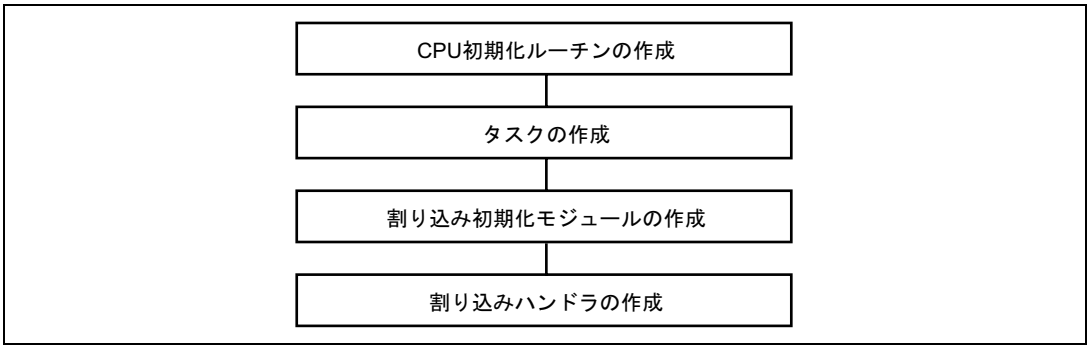


図2-2　作成するプログラム

2.1　CPU 初期化ルーチンの作成

CPU 初期化ルーチンは、CPU リセットによって初めに実行されるプログラムで、バスステートコントローラの設定やハードウェアの初期化などを行います。

本ガイドでは、Solution Engine®標準の ROM モニタが既にバスステートコントローラの設定やハードウェアの初期化が行われているため、新たにバスステートコントローラ等の設定は行いません。

図 2-3 に、CPU 初期化ルーチン作成手順を示します。

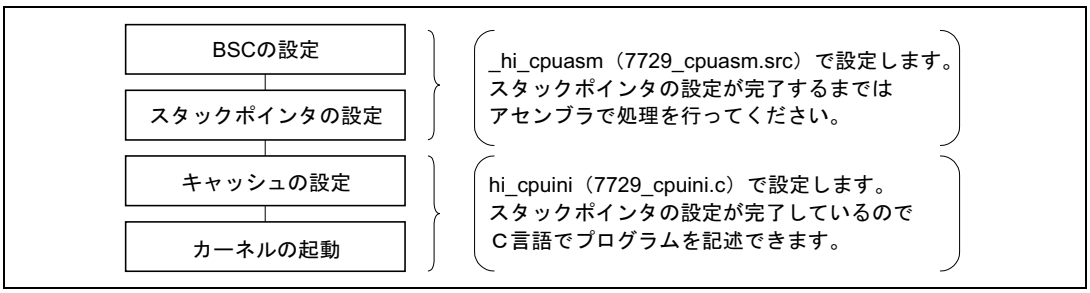


図2-3　CPU 初期化ルーチン作成手順

CPU 初期化ルーチンでは、スタックポインタの設定が完了するまではC言語で記述されたプログラムを実行しないよう注意してください。コンパイラが生成したプログラムが、スタックフレームや作業領域をスタックに確保する可能性があるためです。

図 2-4～図 2-7 に、_hi_cpuasm（7729_cpuasm.src）の変更個所を解説します。

```
;***************************************************************************;
;*              HI7700/4 CPU initialize routine                            ;*;
;*              Copyright (c)  Hitachi, Ltd. 2000.                         ;*;
;*              Licensed Material of Hitachi, Ltd.                         ;*;
;*              HI7700/4(HS0770ITI41SR) V1.0                               ;*;
;***************************************************************************;
;***************************************************************************;
;*  FILE      = 7729_cpuasm.src ;                                          ;*;
;*  CPU type  = SH7729                                                     ;*;
;***************************************************************************;
        .program        _hi_cpuasm
        .export         _hi_cpuasm
        .import         _hi_cpuini
        .import         __kernel_pon_sp
        .import         __kernel_man_sp
        .section        P_hicpuasm,code,align=4
;
;***************************************************************************;
;*              EXPEVT address, data                                       ;*;
;***************************************************************************;
CCN_BASE     .assign  h'ffffffd0           ; INTC(exception) base address
EXPEVT       .assign  h'ffffffd4-CCN_BASE  ; EXPEVT      address offset
;
PON_CODE     .assign  h'000                ; power-on reset exception code
;
```

図2-4　_hi_cpuasm（7729_cpuasm.src）の変更個所

```
;*************************************************************************;
;*              BSC address                                              ;*;
;*************************************************************************;
BSC_BASE     .assign  h'ffffff60           ; BSC         base address
BCR1         .assign  h'ffffff60-BSC_BASE  ; BCR1        address offset
BCR2         .assign  h'ffffff62-BSC_BASE  ; BCR2        address offset
WCR1         .assign  h'ffffff64-BSC_BASE  ; WCR1        address offset
WCR2         .assign  h'ffffff66-BSC_BASE  ; WCR2        address offset
MCR          .assign  h'ffffff68-BSC_BASE  ; MCR         address offset
DCR          .assign  h'ffffff6a-BSC_BASE  ; DCR         address offset
PCR          .assign  h'ffffff6c-BSC_BASE  ; PCR         address offset
RTCSR        .assign  h'ffffff6e-BSC_BASE  ; RTCSR       address offset
RTCNT        .assign  h'ffffff70-BSC_BASE  ; RTCNT       address offset
RTCOR        .assign  h'ffffff72-BSC_BASE  ; RTCOR       address offset
RFCR         .assign  h'ffffff74-BSC_BASE  ; RFCR        address offset
BCR3         .assign  h'ffffff7E-BSC_BASE  ; BCR3        address offset
;
MCSCR_REG_BASE .assign  h'ffffff50         ; MCSCR register base address
MCSCR0         .assign  h'ffffff50-MCSCR_REG_BASE ; MCSCR0 address offset
MCSCR1         .assign  h'ffffff52-MCSCR_REG_BASE ; MCSCR1 address offset
MCSCR2         .assign  h'ffffff54-MCSCR_REG_BASE ; MCSCR2 address offset
MCSCR3         .assign  h'ffffff56-MCSCR_REG_BASE ; MCSCR3 address offset
MCSCR4         .assign  h'ffffff58-MCSCR_REG_BASE ; MCSCR4 address offset
MCSCR5         .assign  h'ffffff5a-MCSCR_REG_BASE ; MCSCR5 address offset
MCSCR6         .assign  h'ffffff5c-MCSCR_REG_BASE ; MCSCR6 address offset
MCSCR7         .assign  h'ffffff5e-MCSCR_REG_BASE ; MCSCR7 address offset
;
SDMR_CS2     .assign  h'ffffd000           ; SDMR(CS2)   base address
SDMR_CS3     .assign  h'ffffe000           ; SDMR(CS3)   base address
;
CMF_BIT      .assign  h'0080               ; CMF bit in RTCSR
;
;*************************************************************************;
;*              BSC initial data                                         ;*;
;*************************************************************************;
;*  After reset, you must initialize BSC for memory(stack) access at first.;*;
;*  Please modify these definition in order to your hardware.            ;*;
;*************************************************************************;
BCR1_DATA    .assign  h'0000               ; BCR1        initial data
BCR2_DATA    .assign  h'3ffc               ; BCR2        initial data
WCR1_DATA    .assign  h'3fff               ; WCR1        initial data
WCR2_DATA    .assign  h'ffff               ; WCR2        initial data
MCR_DATA     .assign  h'0000               ; MCR         initial data
DCR_DATA     .assign  h'0000               ; DCR         initial data
PCR_DATA     .assign  h'0000               ; PCR         initial data
RTCSR_DATA   .assign  h'a500 + h'00        ; RTCSR       initial data
RTCNT_DATA   .assign  h'a500 + h'00        ; RTCNT       initial data
RTCOR_DATA   .assign  h'a500 + h'00        ; RTCOR       initial data
RFCR_DATA    .assign  h'a400 + h'000       ; RFCR        initial data
BCR3_DATA    .assign  h'0000               ; BCR3        initial data
;
STP_REFRESH  .assign  h'a500               ; RTCSR initial data(stop count-up)
;
SDMR2_DATA   .assign  h'0230               ; SDMR_CS2    initial data
SDMR3_DATA   .assign  h'0230               ; SDMR_CS3    initial data
;
MCSCR_DATA   .assign  h'0000               ; MCSCR       initial data
;
IDLE_TIME    .assign  h'566                ; loop counter for idle-time
REFRESH_CNT  .assign  h'8                  ; counter for dummy refresh
;
```

BSCの設定値をハードウェアに合わせて変更します

図2-5　_hi_cpuasm（7729_cpuasm.src）の変更個所

```
;*************************************************************************;
;*  NAME      = _hi_cpuasm                                                 ;*;
;*  FUNCTION  = CPU initialize routine                                     ;*;
;*************************************************************************;
_hi_cpuasm:
;***** Initialize BSC
;       mov.l   #BSC_BASE,r0            ; set BCR base address to gbr
;       ldc     r0,gbr
;
;       mov.w   #BCR1_DATA,r0           ; Initialize BCR1
;       mov.w   r0,@(BCR1,gbr)
;
;       mov.w   #BCR2_DATA,r0           ; Initialize BCR2
;       mov.w   r0,@(BCR2,gbr)
;
;       mov.w   #WCR1_DATA,r0           ; Initialize WCR1
;       mov.w   r0,@(WCR1,gbr)
;
;       mov.w   #WCR2_DATA,r0           ; Initialize WCR2
;       mov.w   r0,@(WCR2,gbr)
;
;       mov.w   #MCR_DATA,r0            ; Initialize MCR
;       mov.w   r0,@(MCR,gbr)
;
;       mov.w   #DCR_DATA,r0            ; Initialize DCR
;       mov.w   r0,@(DCR,gbr)
;
;       mov.w   #PCR_DATA,r0            ; Initialize PCR
;       mov.w   r0,@(PCR,gbr)
;
;       mov.w   #STP_REFRESH,r0         ; stop refresh
;       mov.w   r0,@(RTCSR,gbr)
;
;       mov.w   #RTCNT_DATA,r0          ; Initialize RTCNT
;       mov.w   r0,@(RTCNT,gbr)
;
;       mov.w   #RTCOR_DATA,r0          ; Initialize RTCOR
;       mov.w   r0,@(RTCOR,gbr)
;
;       mov.w   #RFCR_DATA,r0           ; Initialize RFCR
;       mov.w   r0,@(RFCR,gbr)
;
;       mov.w   #BCR3_DATA,r0           ; Initialize BCR3
;       mov.w   r0,@(BCR3,gbr)
;
;       mov.l   #MCSCR_REG_BASE,r0      ; set MCSCR base address to gbr
;       ldc     r0,gbr
;
;       mov.w   #MCSCR_DATA,r0          ; set Initialize MCSCR_DATA
;
;       mov.w   r0,@(MCSCR0,gbr)        ; Initialize MCSCR0
;
;       mov.w   r0,@(MCSCR1,gbr)        ; Initialize MCSCR1
;
;       mov.w   r0,@(MCSCR2,gbr)        ; Initialize MCSCR2
;
;       mov.w   r0,@(MCSCR3,gbr)        ; Initialize MCSCR3
;
;       mov.w   r0,@(MCSCR4,gbr)        ; Initialize MCSCR4
;
;       mov.w   r0,@(MCSCR5,gbr)        ; Initialize MCSCR5
;
;       mov.w   r0,@(MCSCR6,gbr)        ; Initialize MCSCR6
;
;       mov.w   r0,@(MCSCR7,gbr)        ; Initialize MCSCR7
;
```

BSCの設定値を行う場合はコメントをはずします

図2-6　_hi_cpuasm（7729_cpuasm.src）の変更個所

```
;*** Initialize SDRAM
;
;       mov.l   #IDLE_TIME,r0           ; loop for idle-time
;hicpuasm010:
;       add     #-1,r0
;       cmp/eq  #0,r0
;       bf      hicpuasm010
;
;       mov.l   #SDMR_CS2,r0            ; Initialize SDMR(CS2)
;       mov.l   #SDMR2_DATA*4,r2
;       mov.b   r1,@(r0,r2)             ; write dummy data(r1)
;
;       mov.l   #SDMR_CS3,r0            ; Initialize SDMR(CS3)
;       mov.l   #SDMR3_DATA*4,r2
;       mov.b   r1,@(r0,r2)             ; write dummy data(r1)
;
;       mov.w   #RTCSR_DATA,r0          ; Initialize RTCSR
;       mov.w   r0,@(RTCSR,gbr)
;
;       mov.w   #REFRESH_CNT,r2
;hi_cpuasm020:
;       mov.w   @(RFCR,gbr),r0          ; read RFCR
;       cmp/ge  r2,r0                   ; if end dummy refresh
;       bf      hi_cpuasm020            ; else goto hi_cpuasm020
;
;hi_cpuasm030:
;
```

BSCの設定処理を行う場合はコメントをはずします

```
;***** Initialize sp and jump to hi_cpuini() written by C-language
        mov.l   #CCN_BASE,r2            ; get CCN base address
        mov.l   #PON_CODE,r3            ; get exception code to power-on
        mov.l   @(EXPEVT,r2),r0         ; get exception code
        cmp/eq  r3,r0                   ; if exception != power-on
        bf      hi_cpuasm050            ; then hi_cpuasm050
;
        mov.l   #__kernel_pon_sp,r2          ; get stack address
;
hi_cpuasm040:
        mov     r2,r15                  ; set SP
;
        mov.l   #_hi_cpuini,r0          ; get hi_cpuini address
        jmp     @r0                     ; jump to hi_cpuini()
        nop                             ; never return to this point
;
hi_cpuasm050:
        mov.l   #__kernel_man_sp,r2          ; get stack address
        bra     hi_cpuasm040
        nop
;
                .pool
;
        .end
```

スタックポインタを設定します

hi_cpuiniへジャンプします

スタックポインタを設定します

図2-7　_hi_cpuasm（7729_cpuasm.src）の変更個所

図 2-8 に、hi_cpuini（7729_cpuini.c）の変更個所を示します。

```
/************************************************************************/
/*              HI7700/4 CPU initialize routine                         */
/*              Copyright (c)  Hitachi, Ltd. 2001.                      */
/*              Licensed Material of Hitachi, Ltd.                      */
/*              HI7700/4(HS0770ITI41SR) V1.0A                           */
/************************************************************************/
/************************************************************************/
/*  FILE      = 7729_cpuini.c ;                                         */
/*  CPU type  = SH7729                                                  */
/************************************************************************/
#include    <machine.h>
#include    "itron.h"
#include    "kernel.h"

/************************************************************************/
/*              environment data                                        */
/************************************************************************/
#define IOBASE   0xfffffe80             /* I/O base address    = 0xfffffe80 */
#define CCR      (0xffffffec - IOBASE)  /* CCN CCR           address offset */

#define CACHE_ON  0x00000001            /* CACHE enable  data              */
#define CACHE_OFF 0x00000000            /* CACHE disable data              */

/* extern  void    _INITSCT(void); */   /* section-initialize routine      */

#pragma section _hicpuini
/************************************************************************/
/*  NAME      = hi_cpuini                                               */
/*  FUNCTION  = CPU initialize routine                                  */
/************************************************************************/
#pragma noregsave(hi_cpuini)

void    hi_cpuini(void)
{

/*** Initialize Hardware Environment ***/
    set_gbr((VP)IOBASE);                /* Set
    gbr_write_long(CCR, CACHE_OFF);     /* AC  gbr_write_long(CCR, CACHE_ON);

/*** Initialize Software Environment ***/

/*  _INITSCT();    */                  /* Call section-initialize routine  */

    vsta_knl();                         /* Start kernel                     */
}
```

キャッシュをONに変更します

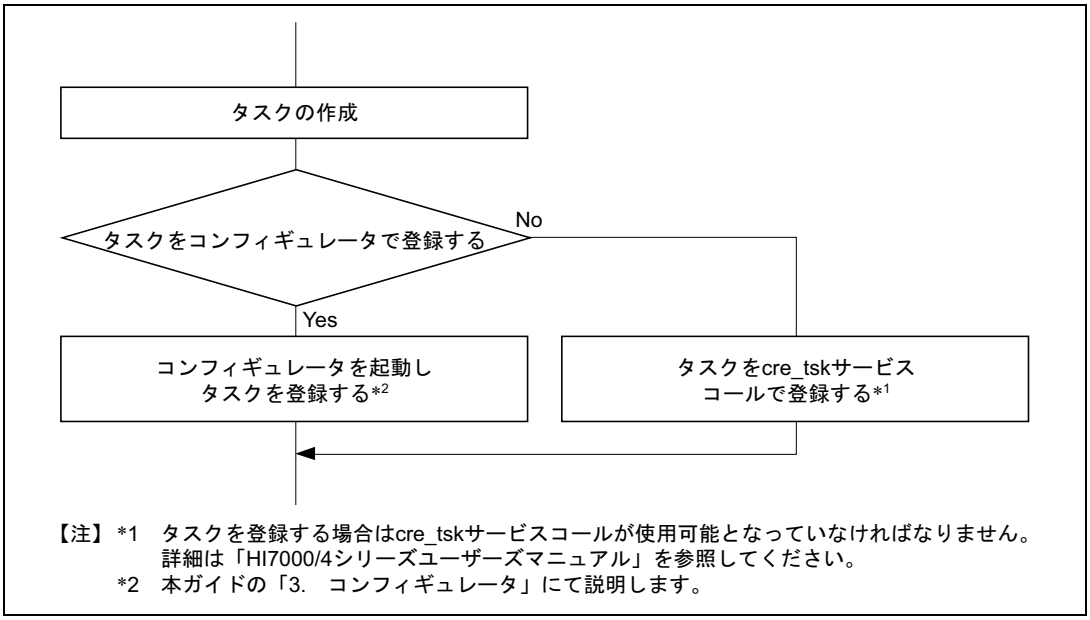
カーネルを起動します

図2-8　hi_cpuini（7729_cpuini.c）の変更個所

独自のハードウェアの場合は、そのハードウェアに合わせたバスステートコントローラの設定やハードウェア初期化ルーチンを作成してください。

2.2 タスクの作成

アプリケーションプログラムのメイン処理となるのがタスクです。
図 2-9 に、タスクの作成および登録手順を示します。



【注】 *1 タスクを登録する場合はcre_tskサービスコールが使用可能となっていなければなりません。
詳細は「HI7000/4シリーズユーザーズマニュアル」を参照してください。
*2 本ガイドの「3. コンフィギュレータ」にて説明します。

図2-9 タスクの作成および登録手順

タスクは、HI7700/4 に標準で付属しているサンプル（task.c）の内容を変更して作成するものとします。task.c は、インストールフォルダ”tutorial”の中にあります。

なお、本ガイドではメインタスク（MainTask）をコンフィギュレータで、LED タスク（task7）を cre_tsk サービスコールで登録します。

2.2.1 メインタスク

HI7700/4 に標準で付属されているサンプルプログラム（task.c）に含まれる MainTask を変更します。
図 2-10 に、MainTask の改造内容を示します。
task7 を周期的に起動させることで、LED の点滅処理を実現します。

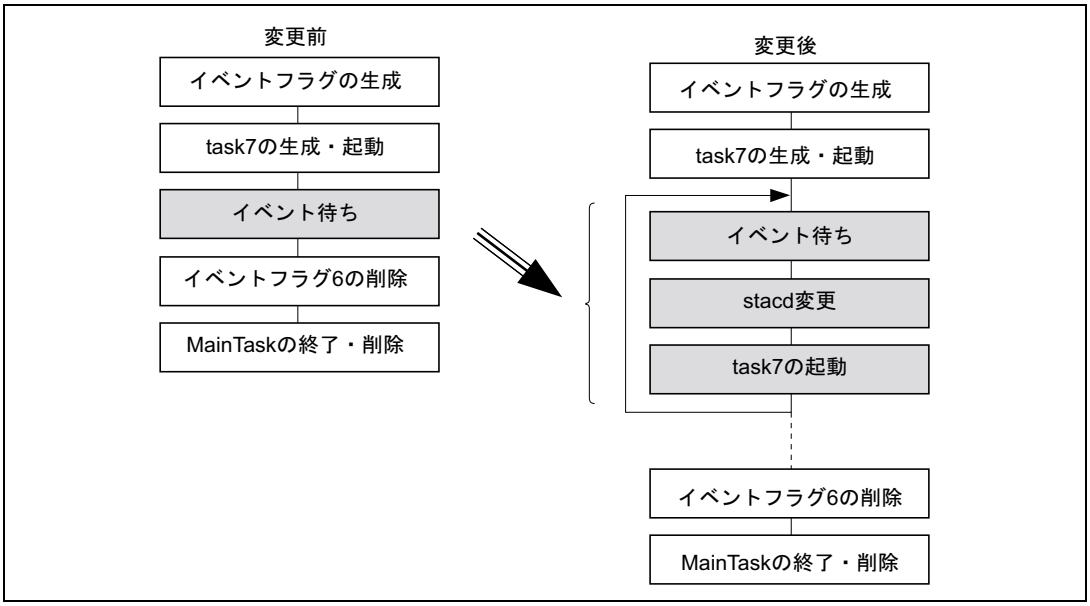


図2-10　MainTask の改造内容概略

図 2-11 に MainTask の改造内容を示します。

```
#include <machine.h>
```
Cコンパイラ付属のインクルードファイルを定義します

```
#include "itron.h"
#include "kernel.h"
#include "kernel_id.h"
```
HI7700/4のサービスコールを使用する場合は必須です

```
#define LED_ADR (UH *)0xb0c00000
```
LEDの出力ポートアドレスを定義します
詳細はSolution Engine概説書を参照してください

```
void MainTask(VP_INT exinf);
void task7(VP_INT exinf);

#pragma noregsave(MainTask, task7)
```
MainTaskとtask7は各タスクのメイン関数であるため
他の関数からコールされることはありません
#pragma noregsaveはスタック領域を抑止するために有効です

```
/*************************************************************************
 * MainTask()
 * This task is created and activated by Configurator.
 *  tskid : "ID_MainTask" (defined in kernel_id.h as this task's ID.)
 *  itskpri : 6
 *************************************************************************/
void MainTask(VP_INT exinf)
{
 union CrePacket{
     T_CTSK  t_ctsk;  /* Creation info. for task */
     T_CFLG  t_cflg;  /* Creation info. for eventflag */
 } packet;

 ER ercd;
 FLGPTN waiptn, flgptn;

 /*** Create eventflag-6 ***/
 packet.t_cflg.flgatr = TA_TFIFO|TA_WSGL|TA_CLR;
 packet.t_cflg.iflgptn = 0;

 ercd = cre_flg(6, (T_CFLG *)&packet);

 /*** Create task-7 ***/
 packet.t_ctsk.tskatr = TA_HLNG|TA_ACT;
 packet.t_ctsk.exinf = 0;
 packet.t_ctsk.task = (FP)task7;
 packet.t_ctsk.itskpri = 7;
 packet.t_ctsk.stksz = 0x200;
 packet.t_ctsk.stk = (VP)NULL;
```
タスク属性を定義します
もしtask7がDSPを使用する場合は" TA_COP0"をOR演算子で定義します（packet.t_ctsk.tskatr = TA_HLNG|TA_ACT| TA_COP0）
これにより、タスク切り替え時にDSPレジスタも退避（カーネルが保証）されるようになります

```
 ercd = cre_tsk(7, (T_CTSK *)&packet);
```
タスクを登録し起動します

```
 /*** Wait for eventflag-6 ***/
 waiptn = 0x11111111;
 ercd = wai_flg(6, waiptn, TWF_ANDW, &flgptn);

 /*** Delete eventflag-6 ***/
 ercd = del_flg(6);

 ext_tsk();
}
```

```
for(;;) {
/*** Wait for eventflag-6 ***/
waiptn = 0x11111111;
ercd = wai_flg(6, waiptn, TWF_ANDW, &flgptn);
if(exinf == 0x00000000L) {
     exinf = 0x0000ff00L;
} else {
     exinf = 0x00000000L;
}
ercd = sta_tsk(7,exinf);
}
```
割り込みハンドラから設定するイベントフラグを待ちます
exinfの値を変更しtask7を起動させます
このときexinfをスタートコードとしてtask7に渡します

図2-11　MainTask の改造内容

2.2.2 LED タスク

HI7700/4 に標準で付属されているサンプルプログラム（task.c）の task7 を変更します。図 2-12 に task7 の変更内容を示します。

```
/*************************************************************************
 * task7()
 * This task is created and activated by MainTask.
 *  tskid : 7
 *  itskpri : 7
 *************************************************************************/
void task7(VP_INT exinf)
{
     ER  ercd;

     ercd = set_flg(6, 0xffffffff);

     while(1);
}
```

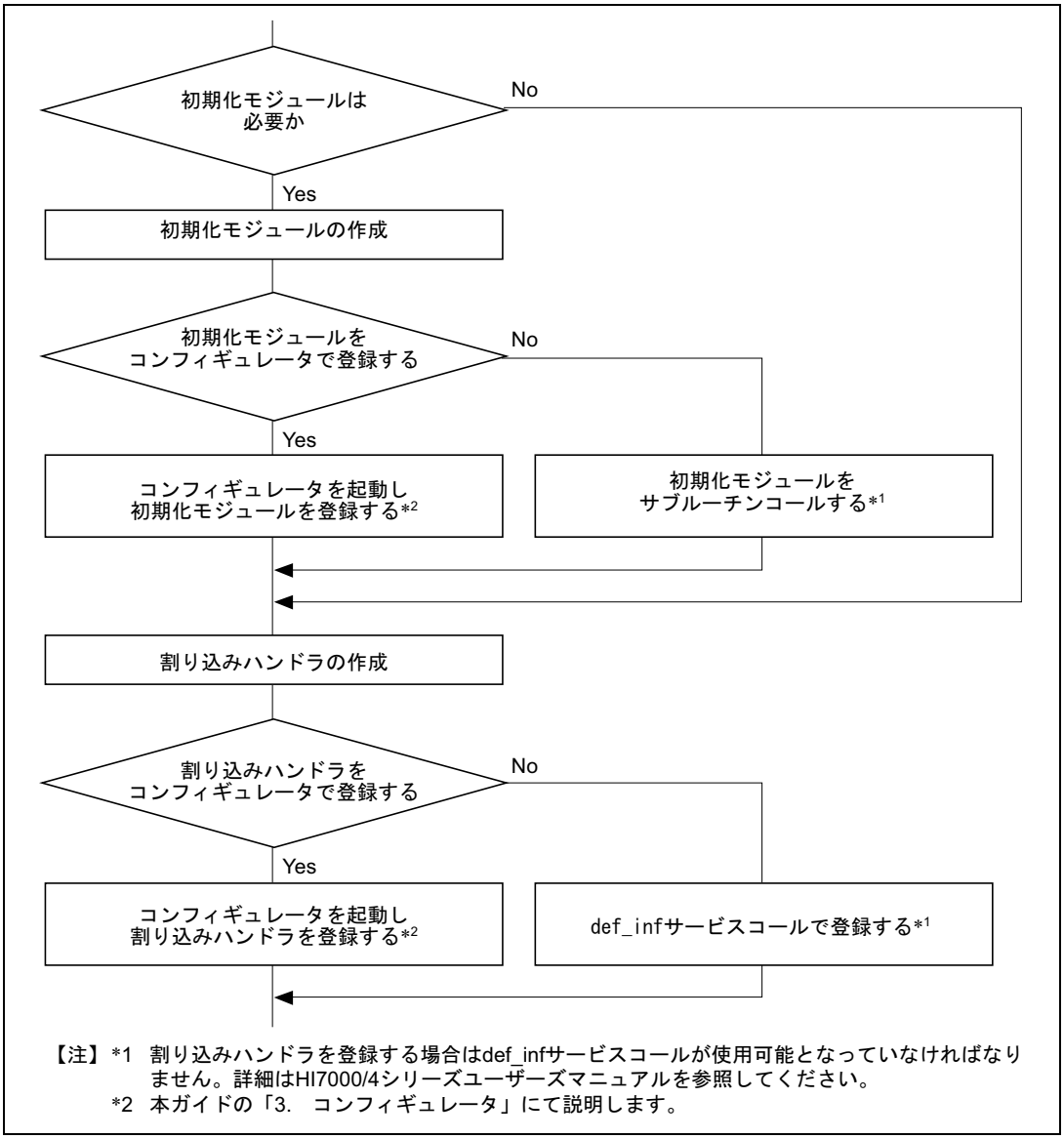
```
*LED_ADR = (UH)exinf;
ext_tsk();
```

exinfの値によりLEDの点灯、消灯を切り替えます。
最後に自身（task7）を終了します

図2-12　task7 の改造内容

2.3 割り込みハンドラの作成

外部からの割り込みにより、他の処理を一時中断し起動されるプログラムが割り込みハンドラです。
図 2-13 に、初期化モジュールおよび割り込みハンドラの作成および登録手順を示します。



【注】*1 割り込みハンドラを登録する場合はdef_infサービスコールが使用可能となっていなければなりません。詳細はHI7000/4シリーズユーザーズマニュアルを参照してください。
*2 本ガイドの「3. コンフィギュレータ」にて説明します。

図2-13 初期化モジュールおよび割り込みハンドラの作成および登録手順

本ガイドでは、SH7729 内蔵 TPU1 を使用した割り込みハンドラの作成およびコンフィギュレータを使用した登録方法について説明します。

初期化モジュールおよび割り込みハンドラは tmu1.c というファイル名で作成し、インストールフォルダ”tutorial”に格納するものとします。なお、本ガイドで使用する割り込みは、カーネルの時間

管理機能で使用するタイマとは無関係です。

また、この割り込みの条件を**表 2-1** に示します。

表2-1　割り込みの条件

	項目	説明	関数名	ファイル名
1	初期化モジュール	必要。コンフィギュレータを使用し登録	TMU1_ini	tmu1.c
2	割り込みハンドラ	コンフィギュレータを使用し登録	TMU1_int	tmu1.c
3	割り込み発生条件	1 秒周期に発生する	ー	ー
4	割り込みレベル	1	ー	ー

2.3.1　初期化モジュールの作成

ここでは SH7729R 内蔵 TPU1 の初期化モジュールを作成ます。TPU1 の初期設定を行い、割り込み発生条件、割り込みレベルを設定します。**図 2-14** に初期化モジュール作成手順を示します。

図2-14　初期化モジュール作成手順

図 2-15 に TMU1_ini(tmu1.c)を示します。

```
#include <machine.h>
#include "itron.h"
#include "kernel.h"

#define BL_BIT 0x10000000               /* BL bit pattern                    */

/* 周辺クロック(CPG設定値=H'0102(CPU:Bus:P=133:33:33MHz)) */
#define PCLK            33333400


/* TSTR 設定値 */
#define TCNT1_STA       0x02            /* TMU1のTCNTを動作開始              */
#define TCNT1_STP       0xfd            /* TMU1のTCNTを動作停止              */


/* TCR 設定値 */
/* TCNTの最高動作周波数は2MHz */
#define DIV             64              /* 分周比=64                          */
#define DIV64           0x0002          /* 分周比を1/64とする                */
#define UNIE            0x0020          /* TCR1のアンダフロー発生で割り込み発生*/


/* TCNT 設定値 */
#define INTERVAL        1000000         /* 1秒:1000ms:1000000us              */
#define TCNT1_DAT       (UW)(((double)INTERVAL/(((double)1/(double)PCLK)*(double)DIV))-(double)1)
                                        /*(1秒/((1秒/33.3334MHz)*64))-1      */

/* IPRA 設定値 */
#define IPRA_CLR_TPU1   0xf0ff          /* IPR bit8-11 clear data            */
#define TMU1_LVL        1               /* TMU1割り込みレベル = 1            */

/* TMU,IPRA I/O address */
#define IOBASE   0xfffffe80             /* I/O base address = 0xfffffe80    */
#define TSTR    (0xfffffe92 - IOBASE)   /* TMU TSTR          address offset */
#define TCOR1   (0xfffffea0 - IOBASE)   /* TMU TCOR(ch1)     address offset */
#define TCNT1   (0xfffffea4 - IOBASE)   /* TMU TCNT(ch1)     address offset */
#define TCR1    (0xfffffea8 - IOBASE)   /* TMU TCR (ch1)     address offset */
#define IPRA    (0xfffffee2 - IOBASE)   /* IPRA(TMU0:TMU1:TMU2:RTC) address */

/*************************************************************************/
/*  NAME      = TMU1_ini                                                 */
/*  FUNCTION  = TMU1を初期化する                                         */
/*************************************************************************/
void TMU1_ini(void)
{
    VP      gbrsave;                    /* GBRセーブ領域                     */
    UH      ipra;                       /* IPRA保持領域                      */

    gbrsave = get_gbr();                /* GBRをセーブする                   */
    set_gbr((VP)IOBASE);                /* GBRにI/O base addressを設定する   */

    gbr_and_byte(TSTR,TCNT1_STP);       /* TMU1のTCNTを停止させる           */
    ipra = gbr_read_word(IPRA);         /* IPRAをリードする                  */
    ipra &= IPRA_CLR_TPU1;              /* IPRA-TMU1のレベルをクリアする    */
    ipra |= TMU1_LVL << 8;              /* IPRA-TMU1のレベルを1とする       */

    set_cr(BL_BIT | get_cr());          /* BL bitを設定し割り込みをマスクする */
    gbr_write_word(IPRA,ipra);          /* IPRAを設定する                    */
    gbr_read_word(IPRA);                /* IPRAをダミーリードする           */

    gbr_write_word(TCR1,UNIE|DIV64);    /* TCRを設定する                     */
    gbr_read_word(TCR1);                /* TCRをダミーリードする            */

    gbr_write_long(TCOR1,TCNT1_DAT);    /* TCORを設定する                    */
    gbr_write_long(TCNT1,TCNT1_DAT);    /* TCNTを設定する                    */

    gbr_or_byte(TSTR, TCNT1_STA);       /* TMU1のTCNT起動させる             */

    set_cr(~BL_BIT & get_cr());         /* BL bitをクリアし割り込みを許可する */
    set_gbr(gbrsave);                   /* GBRを元に戻す                     */
}
```

図2-15　TMU1_ini(tmu1.c)作成内容

2.3.2 割り込みハンドラの作成

ここでは SH7729R 内蔵 TPU1 の割り込みハンドラを作成します。TPU1 の割り込み要因をクリアし、task7 にイベントフラグを発行します。図 2-16 に割り込みハンドラ作成手順を示します。

図2-16 割り込みハンドラ作成手順

図 2-17 に TMU1_int(tmu1.c)を示します。

```
/*************************************************************************/
/*  NAME       = TMU1_int                                                */
/*  FUNCTION   = TMU1 割り込みハンドラ                                    */
/*************************************************************************/
void    TMU1_int(void)
{
    VP      gbrsave;                    /* GBRセーブ領域                 */

    gbrsave = get_gbr();                /* GBRをセーブする               */
    set_gbr((VP)IOBASE);                /* GBRにI/O base addressを設定する */
    set_cr(BL_BIT | get_cr());          /* BL bitを設定し割り込みをマスクする */

    gbr_write_word(TCR1,TCNT1_DAT);     /* UNFをクリアする               */
    gbr_read_word(TCR1);                /* TCRをダミーリードする         */

    iset_flg(6, 0xffffffff);            /* task7に対しイベントフラグを設定 */

    set_cr(~BL_BIT & get_cr());         /* BL bitをクリアし割り込みを許可する */
    set_gbr(gbrsave);                   /* GBRを元に戻す                 */
}                                       /* ret_int                       */
```

図2-17 TMU1_int(tmu1.c)作成内容

3. コンフィギュレーション

「2. **アプリケーションプログラムの作成**」で作成したそれぞれのプログラムを、HI7700/4 に登録する作業がコンフィギュレーションです。HI7700/4 では、GUI ベースで構築作業を円滑に行うためのツール、コンフィギュレータを標準で付属しています。

コンフィギュレータを用いたアプリケーションプログラムの登録方法について説明します。

図 3-1 に本ガイドで登録する項目を示します。

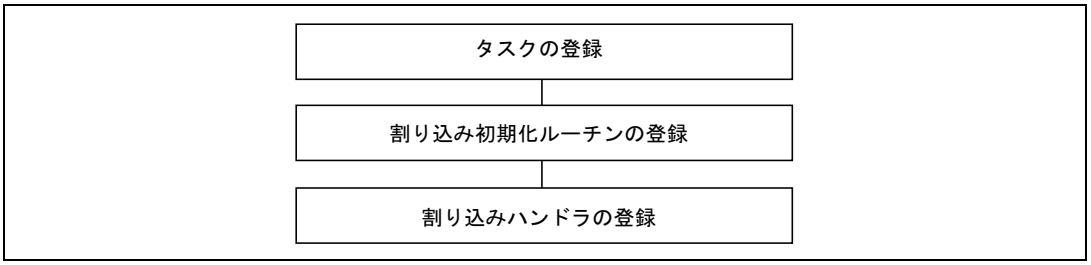


図3-1　本ガイドで登録する項目

これ以外の項目については、標準の設定のまま使用することとします。

なお、コンフィギュレータで設定する各項目の詳細説明についてはコンフィギュレータのヘルプを参照してください。

3.1　コンフィギュレータの起動

コンフィギュレータを起動するには、コンフィギュレータ設定ファイル 7729.hcf をダブルクリックします。7729.hcf は、インストールフォルダ“sh7729”の中にあります。

図 3-2 に、コンフィギュレータ起動画面を示します。

図3-2　コンフィギュレータ起動画面

3.2　割り込みマスクレベル

表 3-1 に、本ガイドで実現するアプリケーションプログラムにおける割り込みマスクレベルの関係を示します。

表3-1　割り込みマスクレベルの関係

項番	アプリケーションの種類	割り込みマスクレベル	備　　考
1	タスク	0	
2	割り込みハンドラ	1	
3	カーネル	14	デフォルト値

コンフィギュレータ起動画面の、HI7700/4 コンフィギュレーション情報の領域で「カーネル動作条件」をクリックすると、図 3-2 に示すコンフィギュレータ起動画面と同じ画面が表示されます。ここで、カーネル割り込みマスクレベルの設定値を変更できます。なお、本ガイドでは、カーネル割り込みマスクレベルをデフォルト値の 14 のままとするため変更は行いません。

3.3　タスクの登録

コンフィギュレータ起動画面の、HI7700/4 コンフィギュレーション情報の領域で、「タスク」をクリックすると、図 3-3 に示すタスク情報の画面が表示されます。

図3-3　タスク情報の画面

図 3-3 の「タスク情報」部分の「変更」ボタンをクリックすると、**図 3-4** に示すタスク情報の変更画面が表示されます。

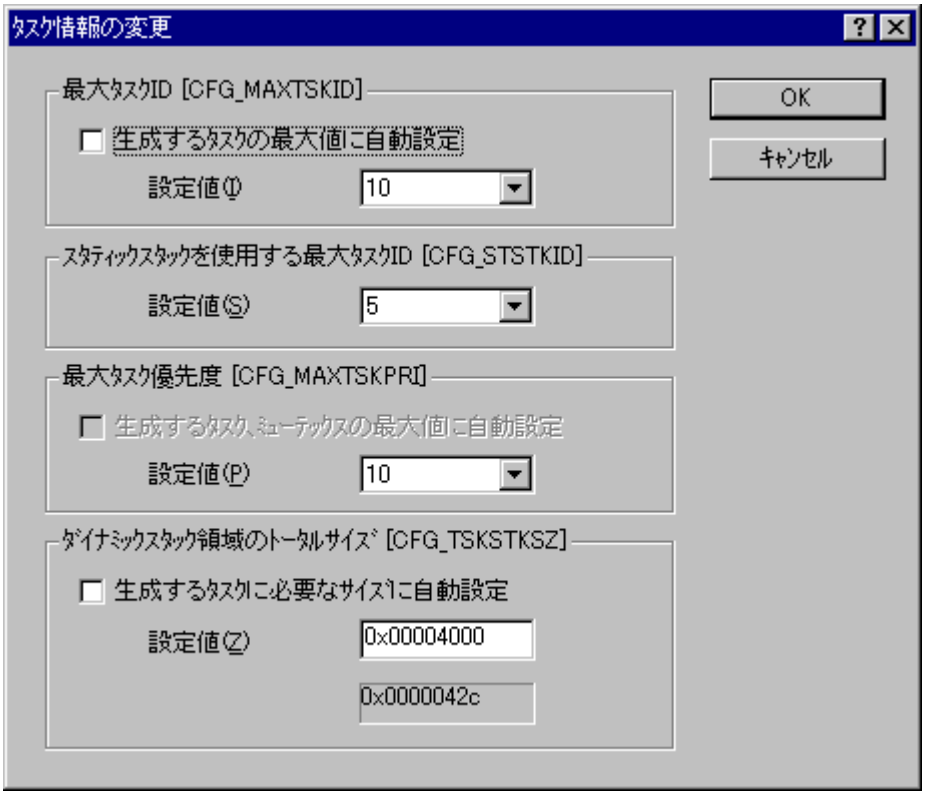


図3-4　タスク情報変更画面

図 3-4 に示す画面で、最大タスク ID、スタティックスタックを使用する最大タスク ID、最大タスク優先度およびダイナミックスタック領域のトータルサイズをそれぞれ変更することができます。スタティックスタックおよびダイナミックスタックの相違点については、HI7000/4 シリーズユーザーズマニュアルの**「2.6.6　タスクのスタック」**をご参照ください。

また、タスクスタックサイズの算出方法については、HI7000/4 シリーズユーザーズマニュアルの**「付録 C 作業領域サイズの算出」**をご参照ください。

本ガイドでは、デフォルト設定のままで使用しますので、タスク情報の変更は必要ありません。

3.4 割り込みハンドラの登録

コンフィギュレータ起動画面の、HI7700/4 コンフィギュレーション情報の領域で、
「割り込み,CPU 例外ハンドラ」をクリックすると、**図 3-5** に示す、割り込み,CPU 例外ハンドラ一覧の画面が表示されます。

図3-5 割り込み,CPU 例外ハンドラー覧の画面

本ガイドでは、SH7729 内蔵 TPU1 を割り込みソースとして使用します。割り込み,CPU 例外ハンドラー覧の領域の右側スクロールバーをマウスで操作し、例外コード 0x0420 の部分を表示させます。例外コード 0x0420 の個所をダブルクリックすると、**図 3-6** に示す、割り込み,CPU 例外ハンドラの定義画面が表示されます。

例外コード 0x0420 は今回作成した TMU1 の割り込み例外コードです。詳細は**「SH7729R ハードウェアマニュアル」**を参照してください。

例外コード 0x0420 の個所をダブルクリックすると**図 3-6** の画面が表示されます。

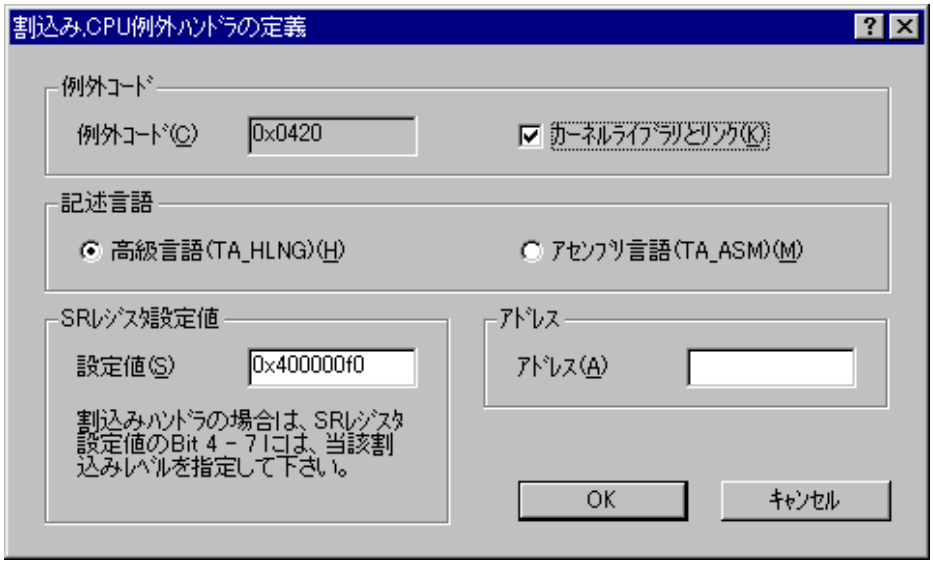


図3-6　割り込み,CPU 例外ハンドラの定義画面

アドレス欄に TMU1_int を設定します。本ガイドで実現する割り込みハンドラのマスクレベルは 1 なので、SR レジスタ設定値は 0x40000010（SR の下位 bit4～7 が割り込みマスク値）に変更します。

この SR レジスタ設定値は、TMU1_int 割り込みハンドラに制御が渡されたときの SR レジスタの値となります。必ずハードウェアの割り込みレベル以上を設定してください。ただし、割り込みハンドラからサービスコールを発行する場合は、カーネル割り込みレベル以下に設定する必要があります。

図 3-7、**図 3-8** に、割り込みハンドラ定義後の画面を示します。

図3-7　割り込みハンドラ定義後の定義画面

図3-8　割り込みハンドラ定義後の一覧画面

割り込みハンドラスタックサイズの算出方法にてついては、HI7000/4 ユーザーズマニュアルの**「付録 C　作業領域サイズの算出」**をご参照ください。

3.5　初期化ルーチンの登録

コンフィギュレータ起動画面の、HI7700/4 コンフィギュレーション情報の領域で、「初期化ルーチン」をクリックすると、**図 3-9** に示す、初期化ルーチン一覧の画面が表示されます。

ここで登録した初期化ルーチンは、カーネルの起動（セットアップ）完了直後に呼び出され、
カーネルマスクレベル（コンフィギュレーション情報のカーネル動作条件で設定した値）で実行されます。
したがって、リセット直後に実行される「CPU 初期化ルーチン」とは異なります。

初期化ルーチンではカーネルのサービスコールを発行することができます。
発行可能なサービスコールは、HI7000/4 シリーズユーザーズマニュアルの**「第 3 章　サービスコール」**で呼び出し可能なシステム状態が"N"：非タスクコンテキストが可能なサービスコールです。

初期化ルーチンは、一般的に以下のような目的で活用します。

(1)　割り込み初期化の処理
(2)　タスクをセットアップするための初期化ルーチン
(3)　イベントフラグ、メールボックス、メモリープールなど、タスクや割り込みハンドラに制御が渡る前に初期設定を完了させておきたい処理に利用できます。

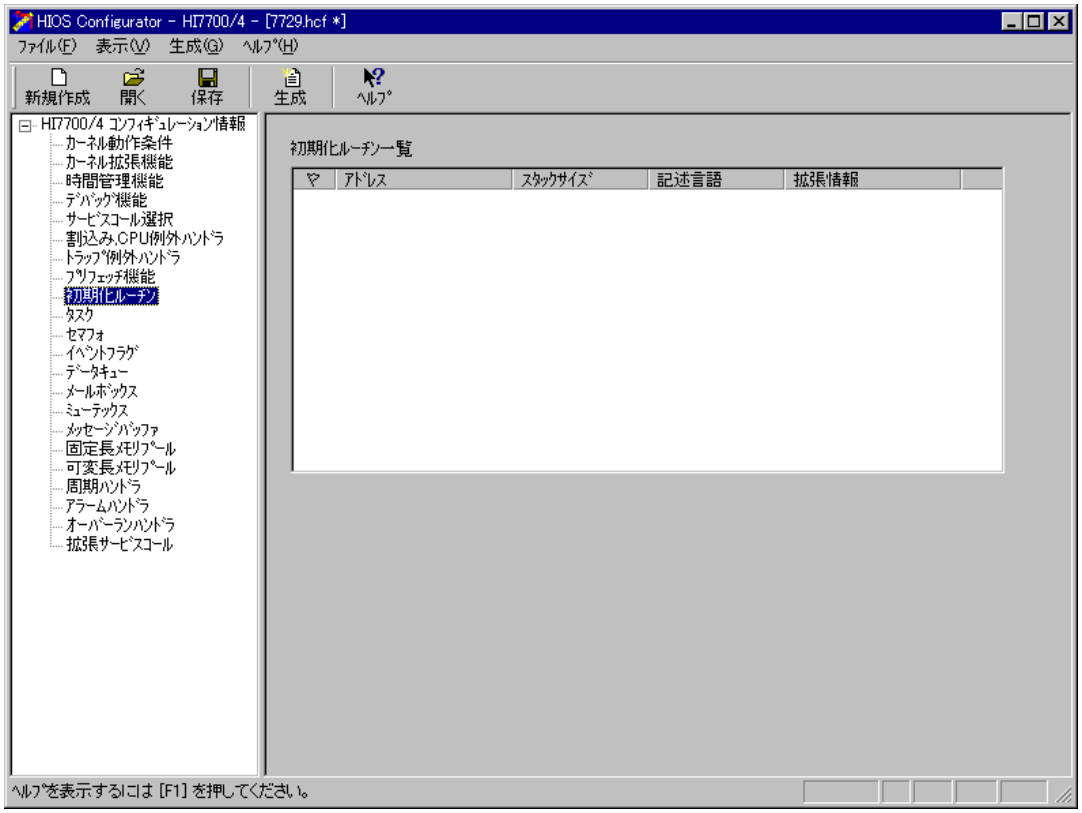


図3-9　初期化ルーチン一覧の画面

初期化ルーチン一覧の空白画面の部分で、右クリックするとメニューが表示されます。そのメニューから登録を選択すると、**図 3-10** に示す初期化ルーチンの登録画面が表示されます。

本ガイドで実現する初期化ルーチンの登録について以下に説明します。

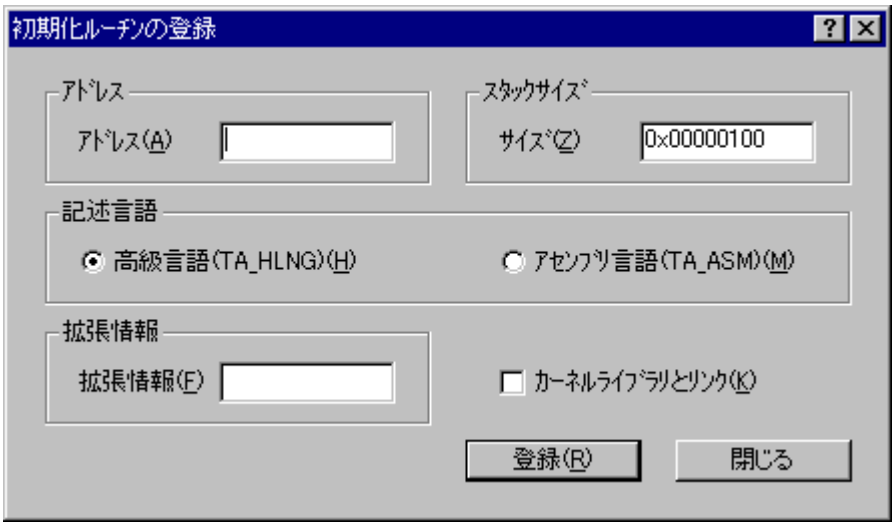


図3-10　初期化ルーチンの登録画面

アドレス欄に TMU1_ini を設定し、登録ボタンをクリックし閉じるボタンをクリックします。スタックサイズは以下の計算式より求めます。

- TMU1_ini のスタックフレームサイズ　：8 バイト
- 初期化ルーチン必須分　　：184＋24 バイト

合計：216 バイト

スタックサイズの計算方法の詳細については、HI7000/4 シリーズユーザーズマニュアルの「付録 C 作業領域サイズの算出」をご参照ください。ここで、計算したスタックサイズはデフォルト値以下なので、スタックサイズの値はデフォルト値のままとします。

図 3-11 に初期化ルーチン登録後の登録画面を、**図 3-12** に初期化ルーチン登録後の一覧画面を示します。

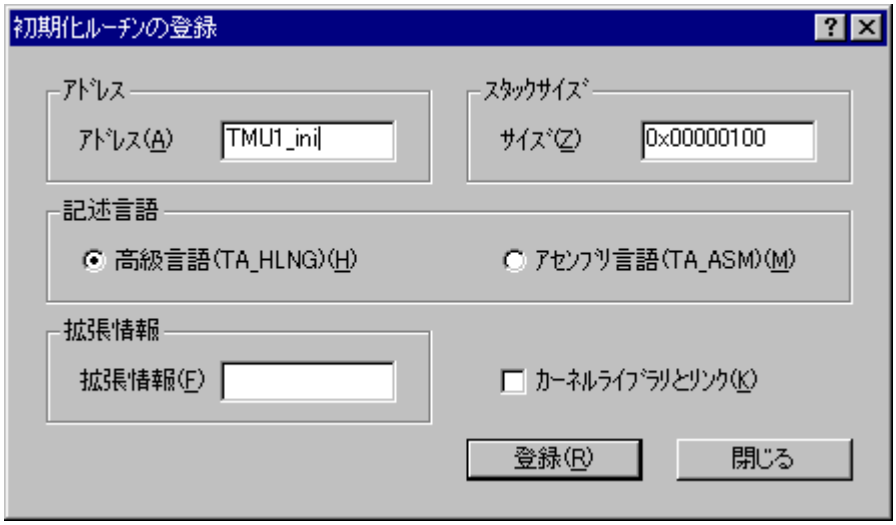


図3-11　初期化ルーチン登録後の登録画面

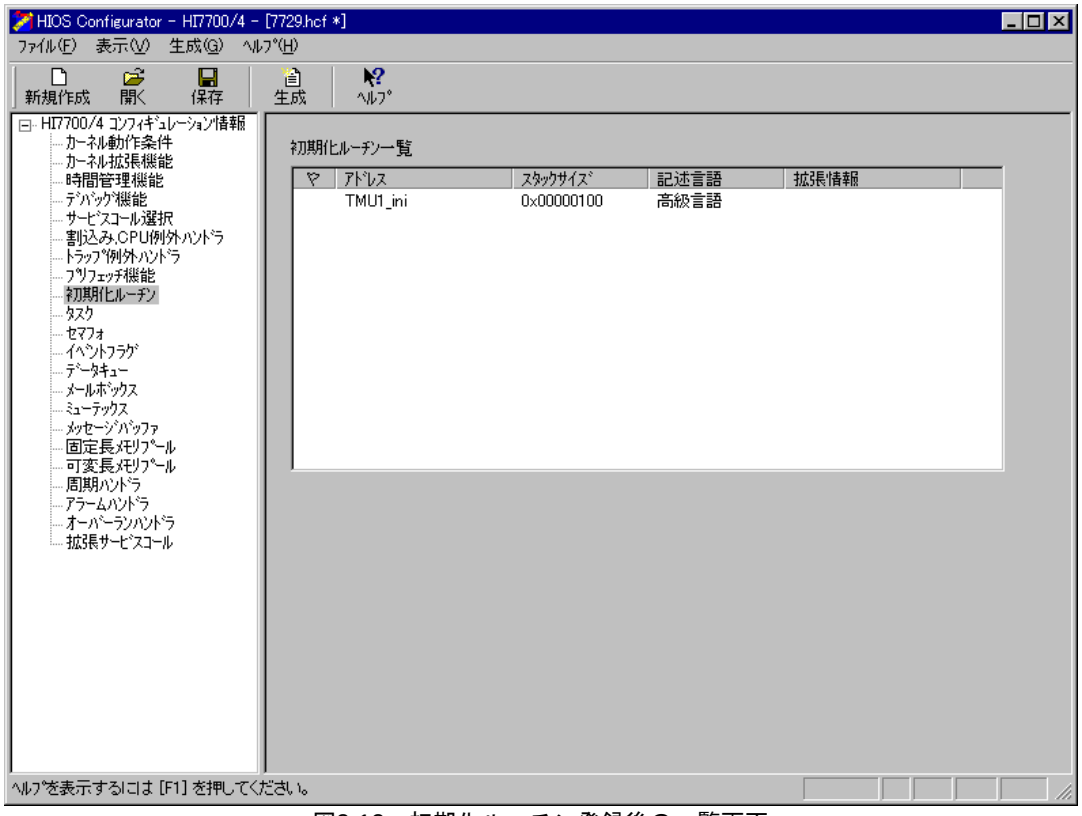


図3-12　初期化ルーチン登録後の一覧画面

3.6 イベントフラグ情報の登録

コンフィギュレータ起動画面の、HI7700/4 コンフィギュレーション情報の領域で「イベントフラグ」をクリックすると、図 3-13 に示すイベントフラグ情報の画面が表示されます。

イベントフラグ情報の部分で「変更」をクリックすると最大イベントフラグ ID を変更することができます。また、イベントフラグ一覧の空白部分で右クリックし、「生成」を選択すると図 3-14 に示すイベントフラグの生成画面が表示されます。イベントフラグを初期生成したい場合は、その情報をここで設定します。

本ガイドで実現するアプリケーションは、イベントフラグを 1 つ、タスクで動的に生成しますので、イベントフラグ情報については、デフォルト設定のまま変更しないものとします。

図3-13 イベントフラグ情報の画面

図3-14　イベントフラグの生成画面

3.7　コンフィギュレーションファイルの生成

コンフィギュレータ起動画面の生成ボタンをクリックすると、HI7700/4 の構築に必要なコンフィギュレーションファイル群が生成されます。コンフィギュレーションファイルの詳細については、HI7000/4 シリーズユーザーズマニュアル「5.1.2　コンフィギュレータの出力ファイル」をご参照ください。

これでコンフィギュレータによる定義、登録作業は完了です。なお、7729.hcf を終了する際、ファイルメニューから”上書き保存”または”名前をつけて保存”を選択し、以上の情報を保存して終了されることをお奨めします。

3.8　HEW によるビルド

コンフィギュレータを用いて作成したファイル群を、コンパイル、リンクし、ダウンロードできる実行形式ファイルを作成します。これには、SHC/C++コンパイラに付属している HEW を用います。HEW によるビルドの方法について説明します。

HI7700/4 は 2 種類の構築方法があります。表 3-2 に、リンクの種類について説明します。

表3-2　リンクの種類

リンクの種類	説明
一括リンク	一括リンクはカーネルとすべてのコンフィギュレーションファイルを 1 つのロードモジュール（これを「一括ロードモジュール」と呼びます）にする方式です。
分割リンク	分割リンクは、カーネルのコード部分（これを「カーネルロードモジュール」と呼びます）とデータ部分（これを「カーネル環境ロードモジュール」と呼びます）を別々のロードモジュールにする方式です。 アプリケーションプログラムはカーネルロードモジュールもしくはカーネル環境ロードモジュールに含めることも、別のアプリケーションロードモジュールにすることもできます。

詳細は、HI7000/4 シリーズユーザーズマニュアルの「5 章　コンフィギュレーション」をご参照ください。

本ガイドでは、一括リンクを使用したビッグエンドィアン用プログラムの構築方法を説明します。

3.8.1 HEW の起動

インストールフォルダ”hios”の中の hios.hws をダブルクリックすると、HI7700/4 をビルドするための HEW が起動されます。**図 3-15** に、HEW の起動画面を表示します。

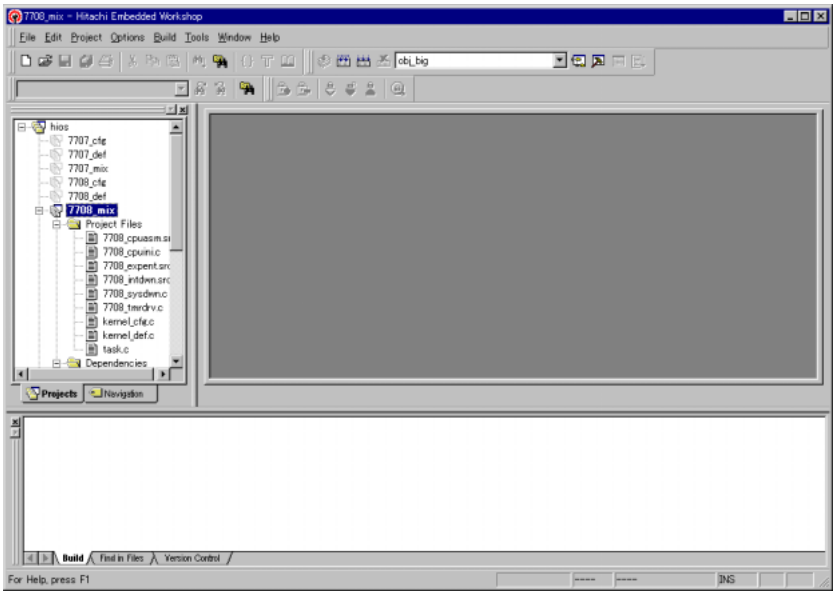


図3-15　HEW の起動画面

標準提供のプロジェクトファイル「hios.hws」には、ターゲット CPU 用のプログラムを構築するために 3 つのサブプロジェクトがあります。**表 3-3** に、プロジェクトファイルの種類を示します。

表3-3　プロジェクトファイルの種類

7729_mix	一括リンクの一括ロードモジュール生成用プロジェクトファイル
7729_cfg	分割リンクのカーネルロードモジュール生成用プロジェクトファイル
7729_def	分割リンクのカーネル環境ロードモジュール生成用プロジェクトファイル

一括ロードモジュール生成用プロジェクトファイル「7729_mix」を選択します。
図 3-16 にカレントプロジェクトの設定画面を示します。

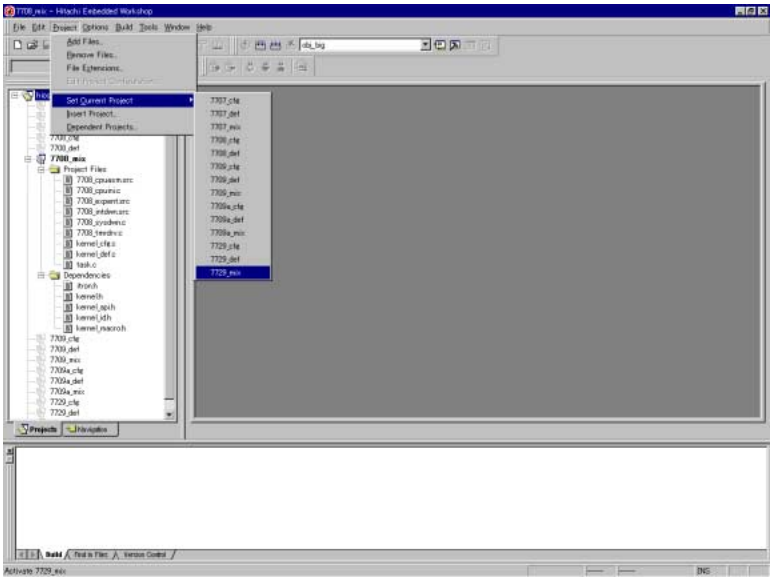


図3-16　カレントプロジェクトの設定画面

3.8.2　構築ファイルの定義

「2. アプリケーションプログラムの作成」で作成した各アプリケーションプログラムを、プロジェクトファイルに定義します。なお、標準提供時のプロジェクトファイルの構成をそのまま利用し、実際にはタイマドライバのみを定義すれば、本ガイドのサンプルプログラムの動作を実現できます。

カレントプロジェクト設定後の画面で、メニューバーの Project から Add Files...を選択し、tmu1.c をプロジェクトファイルに追加します。図 3-17、図 3-18 にファイル追加の手順を示します。

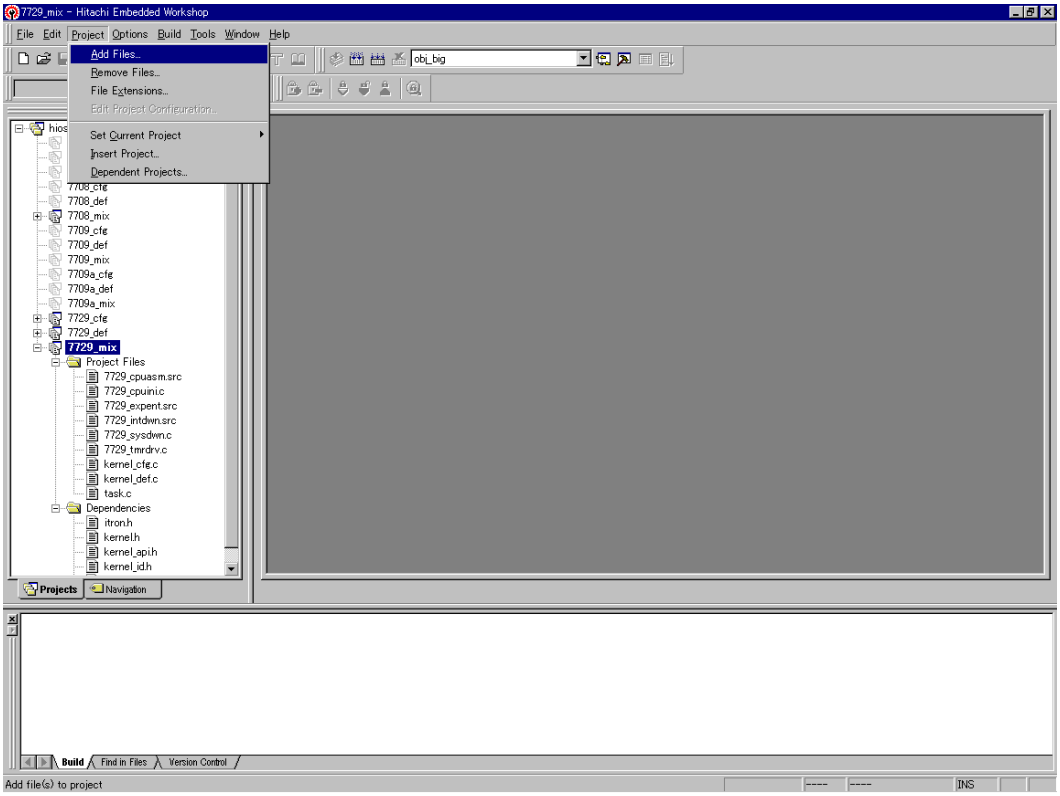


図3-17　ファイル追加の手順

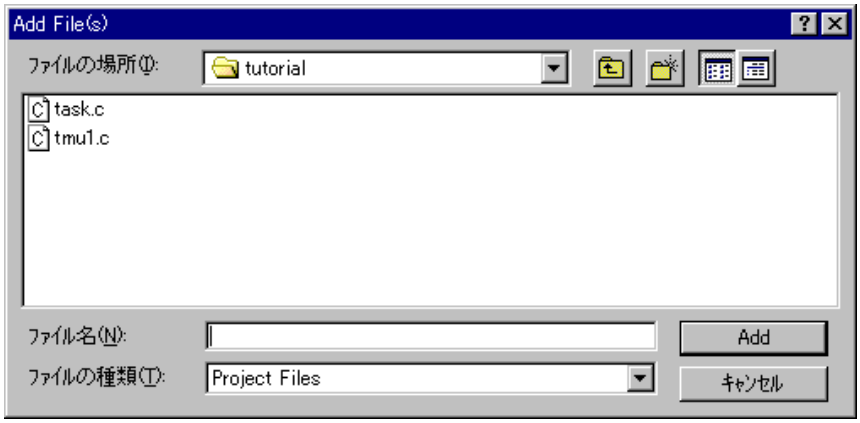


図3-18　ファイル追加の手順

以上で構築ファイルの定義は完了です。

3.8.3 リンケージアドレスの変更

Solution Engine®のアドレスマップで動作させるため、リンケージアドレスの変更を行います。
Solution Engine®は 0x0C00000 から 0x0FFFFFFF の 64M バイト実装されています。
本ガイドでは、0x0C00000 から 0x0CFFFFFF の 16M バイト使用することとします。

始めに、メニューバーの Options から OptLinker を選択し、OptLinker Options の画面を表示させます（図 3-19）。

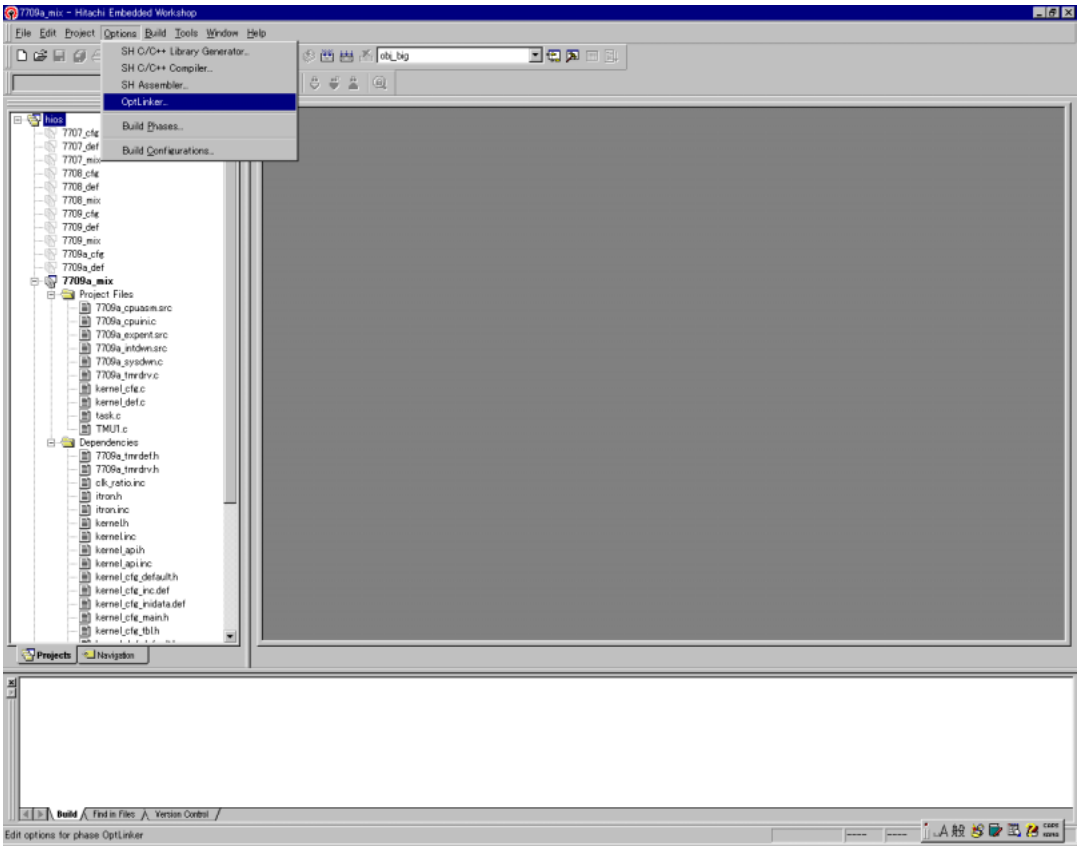


図3-19　OptLinker の選択

(1) カーネルスタックポインタの変更

Input タブ画面の Defines の__kernel_pon_sp および__kernel_man_sp をそれぞれダブルクリックし、それぞれハードウェアの RAM が実装エリアを末尾アドレス＋1 番地(0xAD000000 : P2 空間)をポイントさせます（図 3-20～図 3-22）。

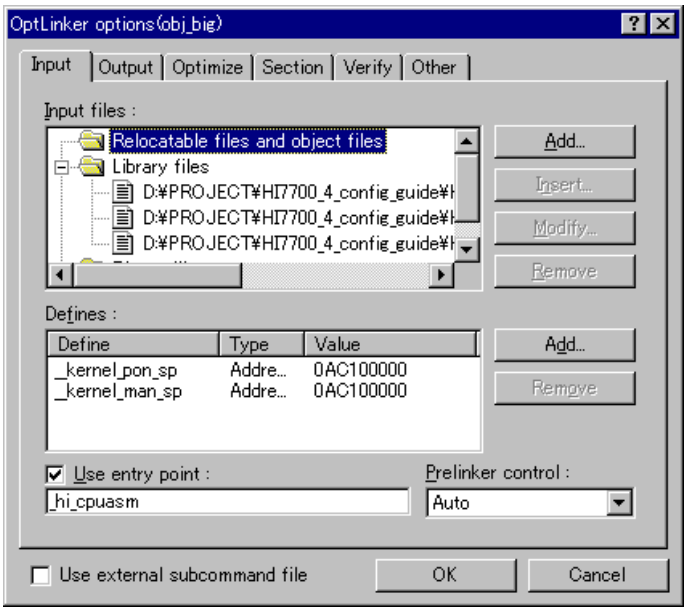


図3-20　OptLinker options 画面

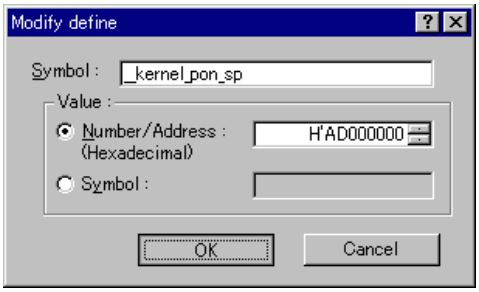


図3-21　OptLinker options 画面(__kernel_pon_sp)

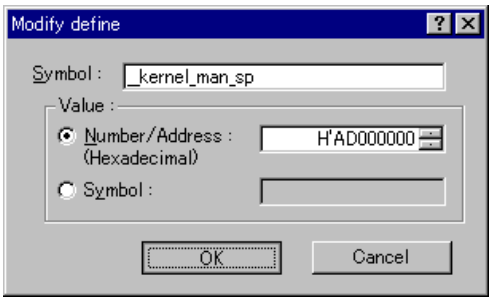


図3-22　OptLinker options 画面(__kernel_man_sp)

(2)　セクションアドレスの変更

次に Section タブをクリックし、セクション定義画面を表示します（図 3-23）。

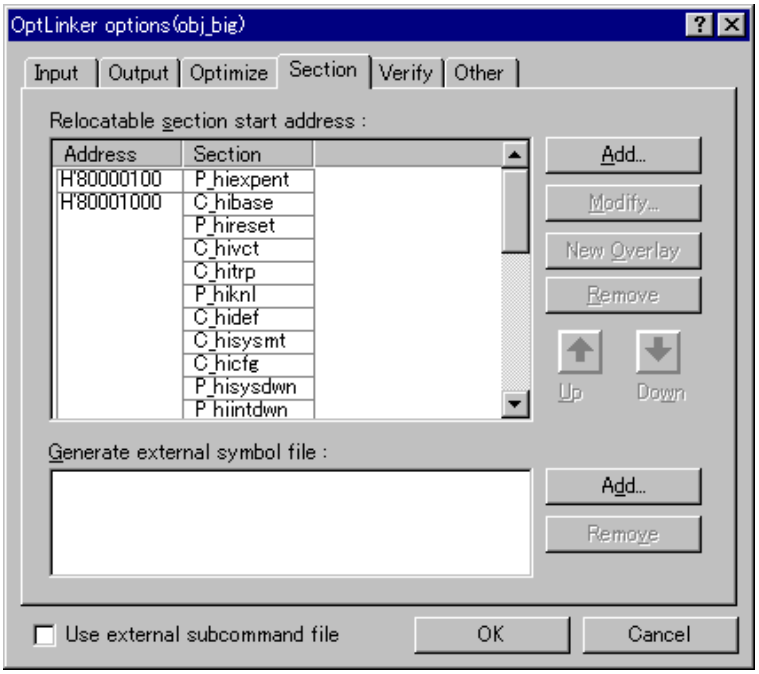


図3-23　セクション定義画面

それぞれのセクションの Address をクリックすると、Modify...ボタンが有効になります。セクションのアドレスを表 3-4 のように変更してください。

表3-4　セクションアドレスの変更内容

セクション名	変更前	変更後	セクション名	変更前	変更後
P_hiexpent	80000100	8C000100	B_hiwrk	8C000000	8C010000
C_hibase	80001000	8C001000	B_himpl		
P_hireset			B_hidystk		
C_hivct			B_histstk		
C_hitrp			B_hiirqstk		
P_hiknl			B_hitrcbuf		
C_hidef			B_hitrceml		
C_hisysmt			B		
C_hicfg			R		
P_hisysdwn			P_hicpuasm	A0000000	AC000000
P_hiintdwn			P_hicpuini		
P_hitmrdrv					
P					
C					
D					

3.8.4 ビルド

HEW のビルドを実行し、E10A エミュレータで Solution Engine®にダウンロードできる実行形式ファイルを作成します。メニューバーの Build から Build を選択します。**図 3-24** にビルドの手順を示します。

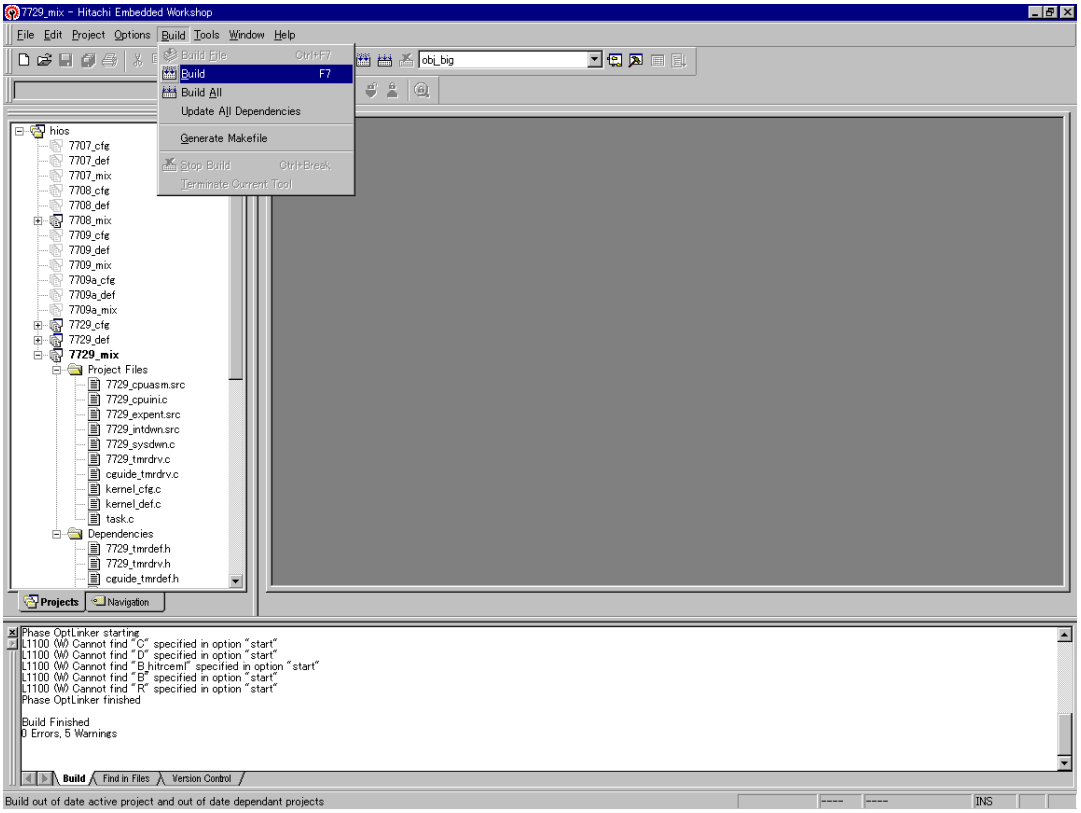


図3-24　Build の手順

以上の操作で実行形式ファイルが作成されます。なお、コンパイル、リンクの結果が最下部のウィンドウに表示されますので、コンパイルエラー等が発生した場合は、該当するソースを修正した後、再度ビルドを行ってください。実行形式ファイル（拡張子.abs）は、インストールフォルダ”obj_big”に生成されます。

以上で、E10A エミュレータで Solution Engine®にダウンロードし、実行する環境が整いました。この後のダウンロード、実行の方法については**「4. E10A によるダウンロードと実行」**をご参照ください。

3.9 パラメータチェック機能無しでの構築

アプリケーションプログラムのデバッグが完了し、いよいよ実際の製品に組込めるレベルに仕上がった場合、サービスコールの先頭で行うパラメータチェック処理は無駄なルーチンとなるため、HI シリーズ OS では、このパラメータチェック機能を取り外すことができるようになっています。

パラメータチェック無しの指定は、コンフィギュレータで簡単に行うことができます。図 3-25 に、パラメータチェック機能を取り外す方法について説明します。

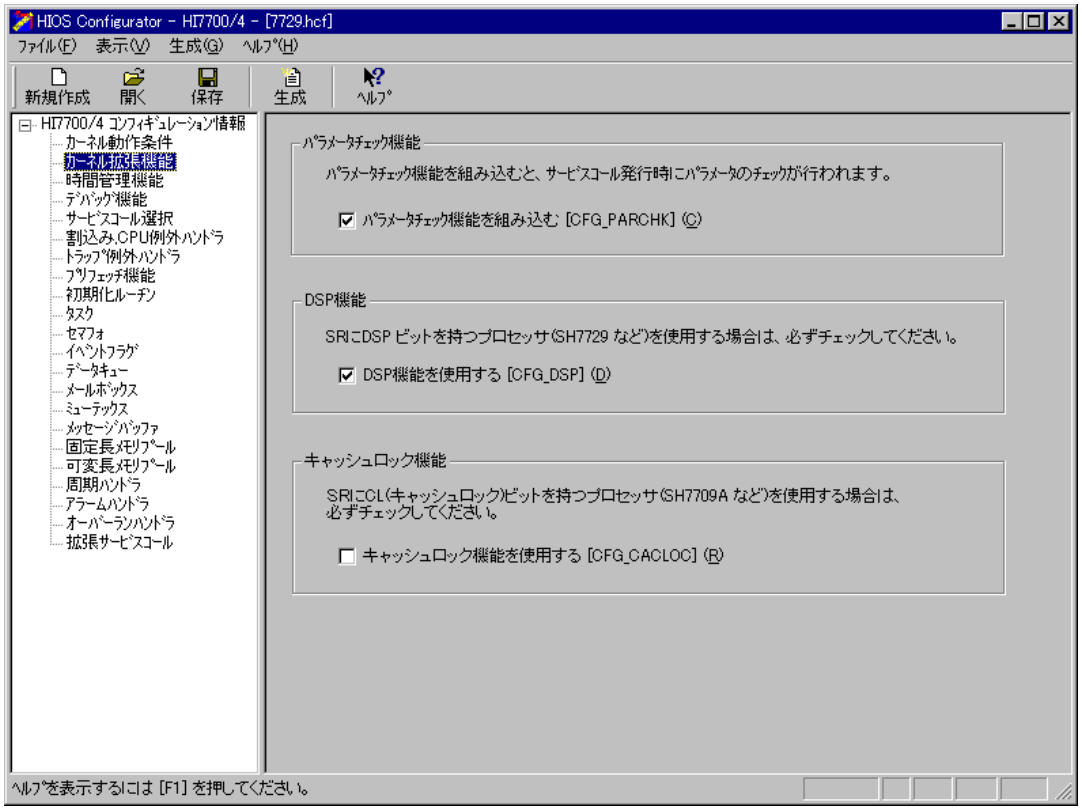


図3-25 パラメータチェック機能を取り外す方法

コンフィギュレータの起動画面で、カーネル拡張機能をクリックすると、図 3-25 の画面が表示されます。ここで、「パラメータチェック機能を組み込む」のチェックボックスをクリックし、チェック印を外してコンフィギュレーションファイルを生成し、ビルドすれば、パラメータチェック機能が取り外された実行形式ファイルが作成されます。

4. E10A によるダウンロードと実行

「3. コンフィギュレーション」で作成した実行形式ファイルを、Solution Engine®にて動作させるために、本ガイドでは E10A を用いてダウンロードし、実行する方法を説明します。

4.1 Solution Engine®の初期化

Solution Engine®には標準で ROM モニタが搭載されており、CPU の初期化を行います。本ガイドでは、CPU 初期化までは Solution Engine®の ROM モニタを利用することとします。（他のボードを使用する場合、CPU 初期化処理はユーザが独自に作成しなければなりません。CPU 初期化処理については、「2.1 CPU 初期化ルーチンの作成」をご参照ください。）

「1. 概要」の図 1-1 で示したハードウェア構成の通り接続します。ホストコンピュータを起動し、Solution Engine®の電源を入れ、Windows®のスタートメニューから「HDI for E10A SH7729」を選択し、HDI を起動します。図 4-1 に、HDI の起動画面を示します。

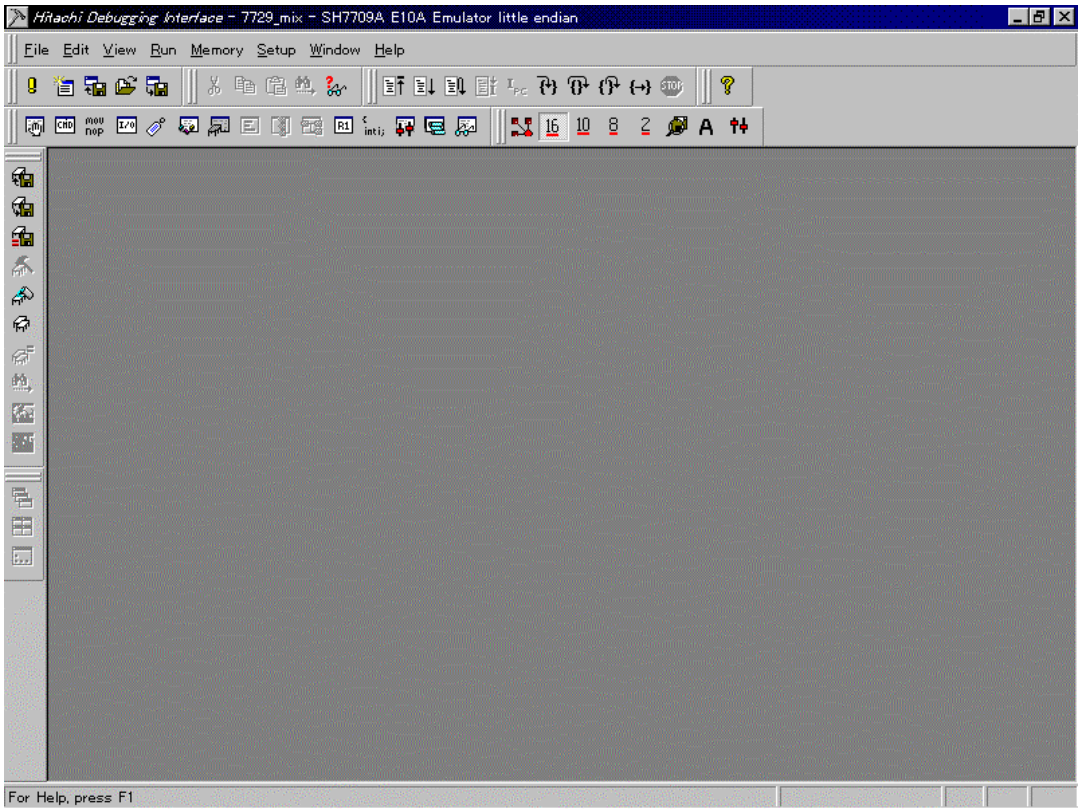


図4-1 HDI の起動画面

次に、メニューバーの Run から Reset Go を選択します（図 4-2）。

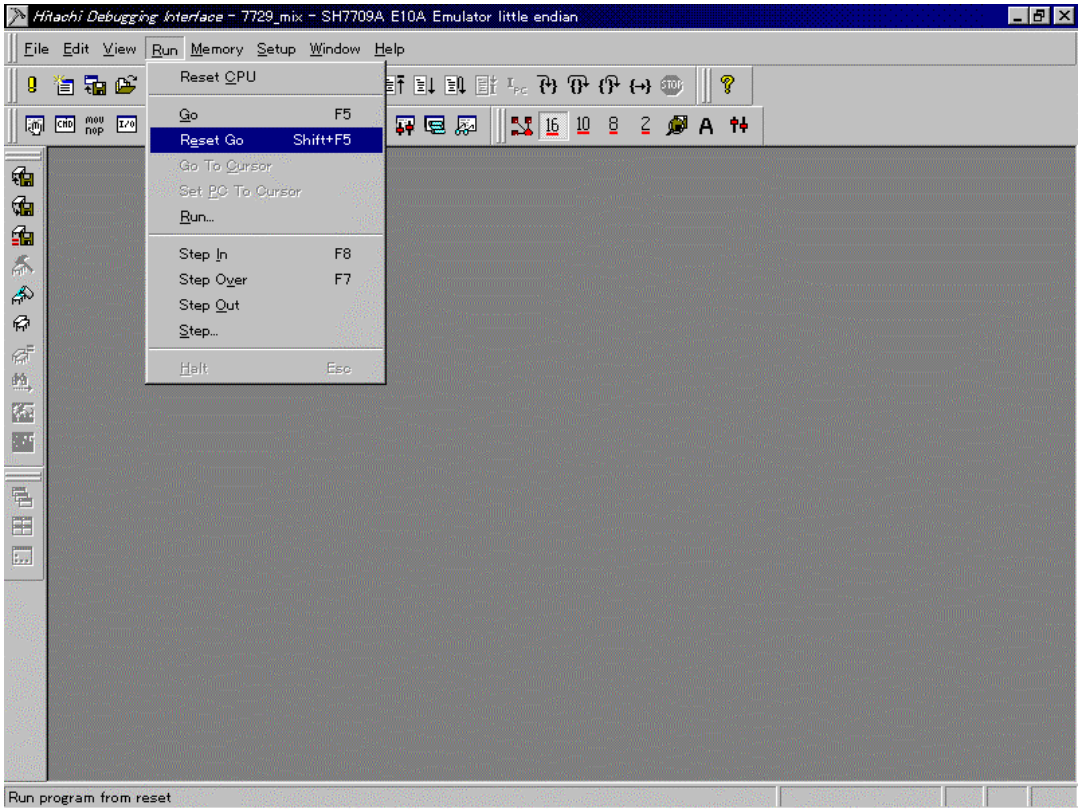


図4-2　Reset Go メニュー

1～2 秒経過後、メニューバーの STOP ボタン（赤色）をクリックします。これで Solution Engine® の初期化が終了し、Solution Engine®に搭載されている SDRAM へのリード／ライト等が可能になります。

4.2　アプリケーションプログラムのダウンロード

「3. コンフィギュレーション」で作成した実行形式ファイルを、実際に E10A にダウンロードします。図 4-3 に、ダウンロード方法を示します。

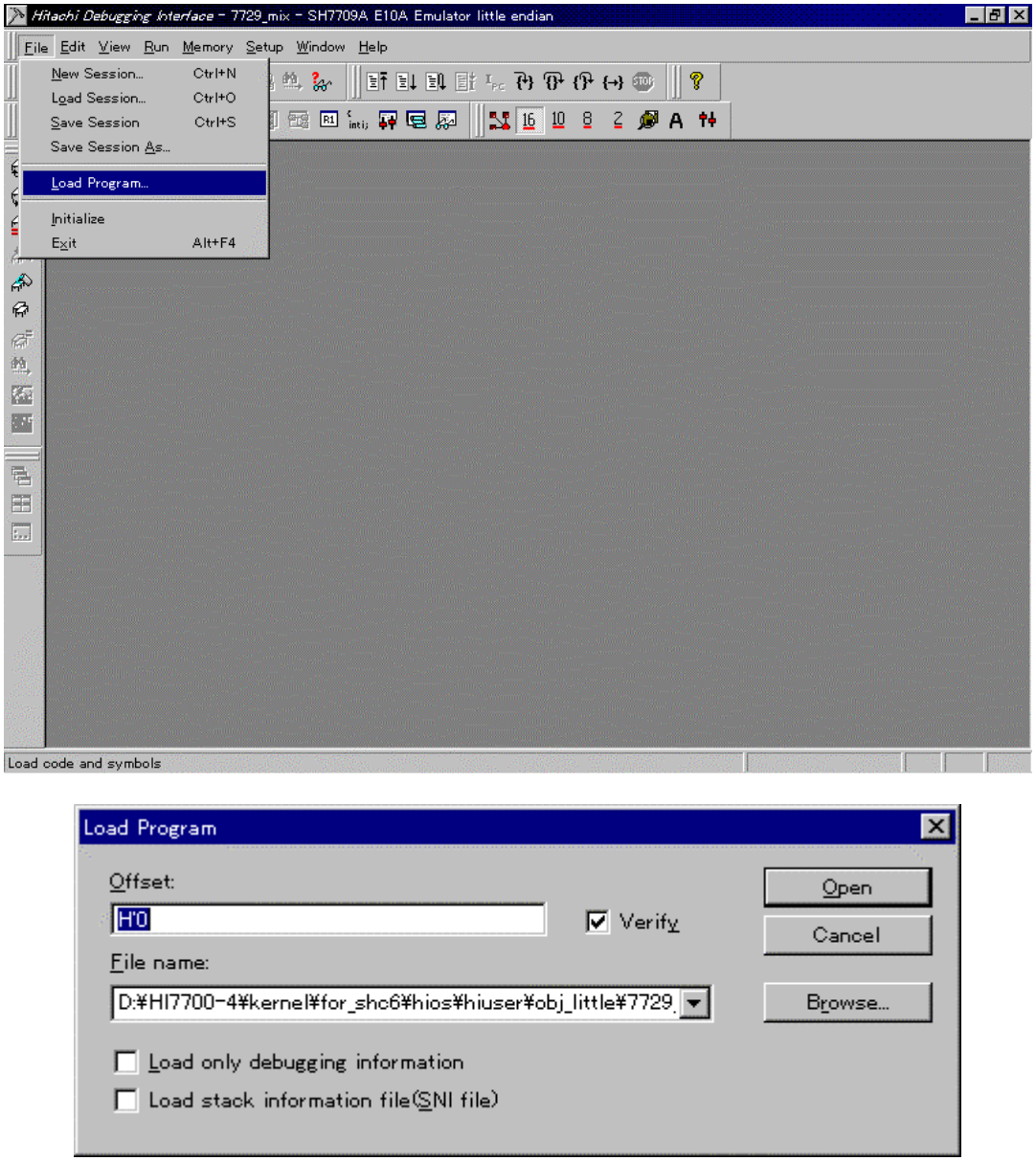


図4-3　ダウンロード方法

HDI 起動画面のメニューバーから File を選択し、Load Program...を選択します。図 4-3 に示す Load Program の画面で、ダウンロードする実行形式ファイルのファイル名を File Name 欄に入力し、Open ボタンをクリックするとダウンロードが始まります。実行形式ファイルは、インストールフォルダ”obj_big”の中の 7729_mix.abs となります。

ダウンロードが成功すると、図 4-4 に示すダウンロード終了画面が表示されます。

図4-4　ダウンロード終了画面

ダウンロード終了画面で OK ボタンをクリックしてください。

4.3　アプリケーションプログラムの実行

ダウンロードが終了したら、いよいよプログラムを実行します。まず、実行する準備として、HDI 起動画面のメニューバーの View から Registers を選択し、レジスタ情報を表示させます（図 4-5）。

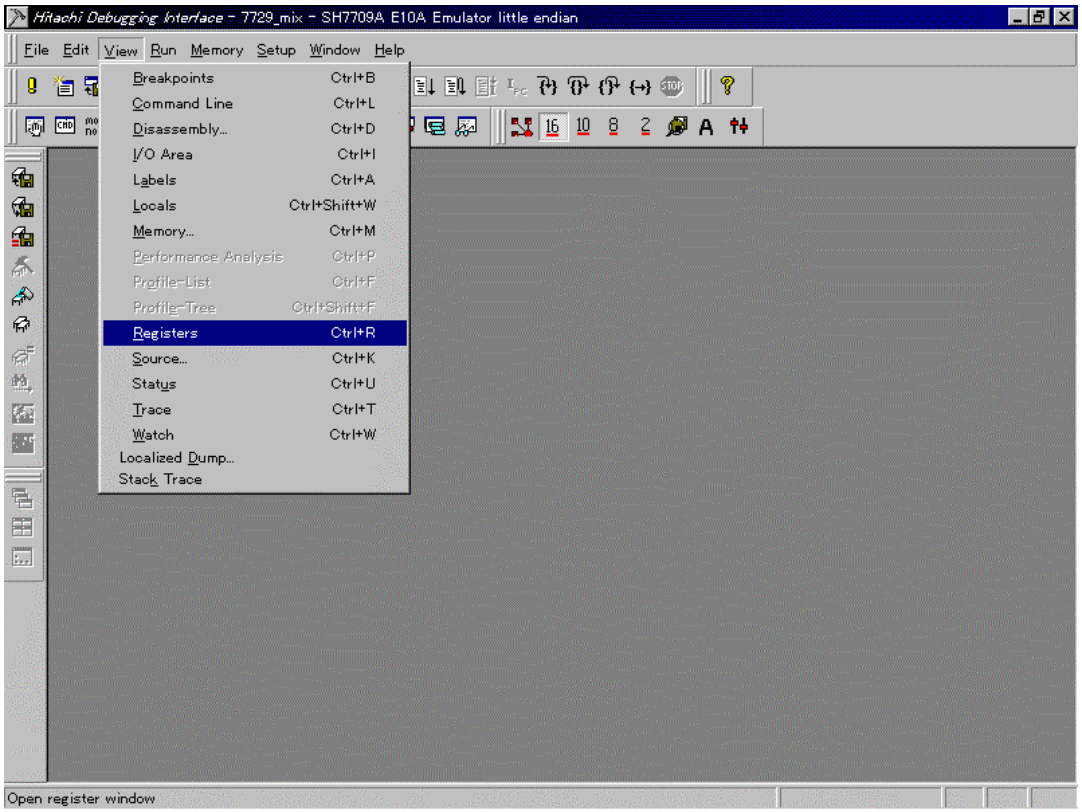


図4-5　レジスタ情報

次に PC の値を変更します。レジスタ情報画面の PC の値をダブルクリックすると、値を変更できる画面が表示されます（図 4-6）。

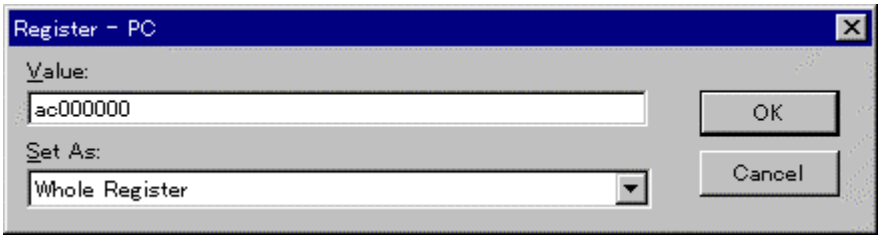


図4-6　PC 値変更画面

ここで、図 4-6 のように PC の値を AC000000 に変更し OK ボタンをクリックします。この値は、CPU 初期化ルーチンの先頭アドレスです。

以上で実行の準備が整いました。メニューバーの Run から Go を選択するとプログラムが実行されます（図 4-7）。

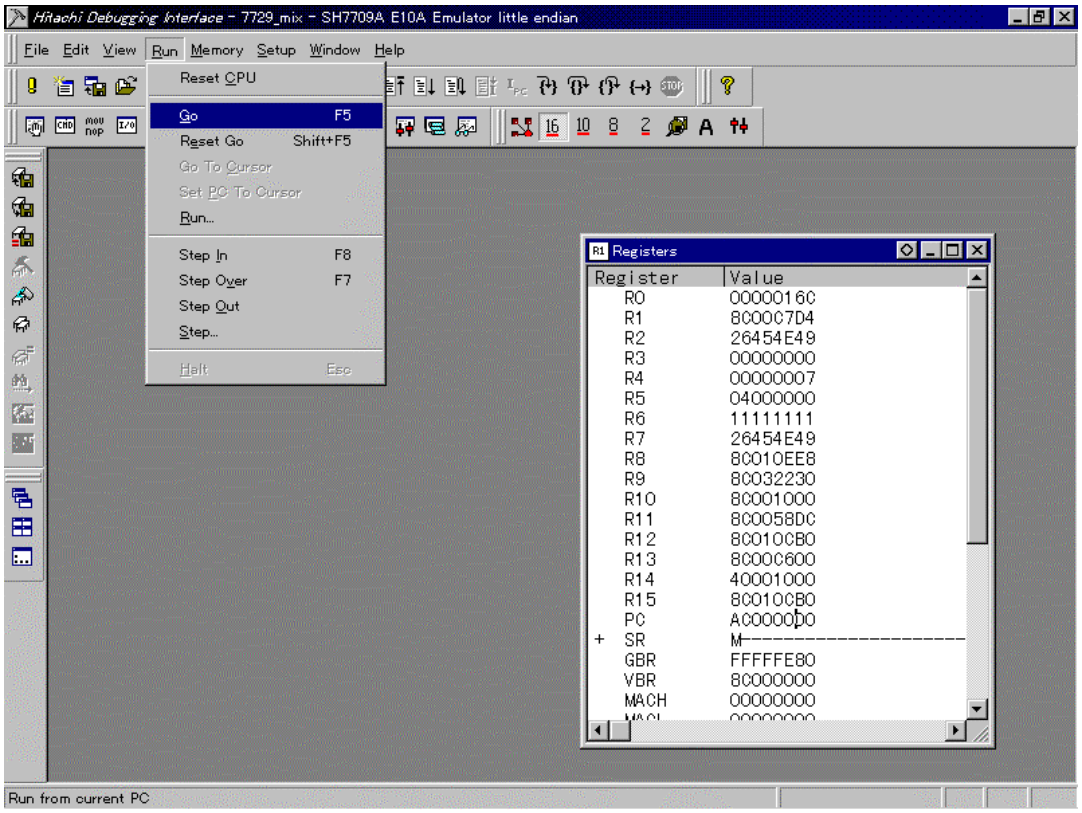


図4-7　プログラム実行画面

HI7700/4　構築ガイド

発行年月　2003 年 3 月　第 1 版
発　　行　株式会社　日立製作所
　　半導体グループビジネスオペレーション本部
編　　集　株式会社　日立小平セミコン
　　技術ドキュメントグループ

HI7700/4 構築ガイド

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

ADJ-502-177