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。

お客様各位

資料中の「日立製作所」、「日立XX」等名称の株式会社ルネサス テクノロジへの変更について

2003年4月1日を以って三菱電機株式会社及び株式会社日立製作所のマイコン、ロジック、アナログ、ディスクリート半導体、及びDRAMを除くメモリ（フラッシュメモリ・SRAM等）を含む半導体事業は株式会社ルネサス テクノロジに承継されました。従いまして、本資料中には「日立製作所」、「株式会社日立製作所」、「日立半導体」、「日立XX」といった表記が残っておりますが、これらの表記は全て「株式会社ルネサス テクノロジ」に変更されておりますのでご理解の程お願い致します。尚、会社商標・ロゴ・コーポレートステートメント以外の内容については一切変更しておりませんので資料としての内容更新ではありません。

ルネサステクノロジ ホームページ（http://www.renesas.com）

2003年4月1日
株式会社ルネサス テクノロジ
カスタマサポート部

RENESAS

ご注意

安全設計に関するお願い

1. 弊社は品質、信頼性の向上に努めておりますが、半導体製品は故障が発生したり、誤動作する場合があります。弊社の半導体製品の故障又は誤動作によって結果として、人身事故、火災事故、社会的損害などを生じさせないような安全性を考慮した冗長設計、延焼対策設計、誤動作防止設計などの安全設計に十分ご留意ください。

本資料ご利用に際しての留意事項

1. 本資料は、お客様が用途に応じた適切なルネサス テクノロジ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ルネサス テクノロジが所有する知的財産権その他の権利の実施、使用を許諾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の使用に起因する損害、第三者所有の権利に対する侵害に関し、ルネサス テクノロジは責任を負いません。
3. 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全ての情報は本資料発行時点のものであり、ルネサス テクノロジは、予告なしに、本資料に記載した製品または仕様を変更することがあります。ルネサス テクノロジ半導体製品のご購入に当たりましては、事前にルネサス テクノロジ、ルネサス販売または特約店へ最新の情報をご確認頂きますとともに、ルネサス テクノロジホームページ (http://www.renesas.com)などを通じて公開される情報に常にご注意ください。
4. 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誤りに起因する損害がお客様に生じた場合には、ルネサス テクノロジはその責任を負いません。
5. 本資料に記載の製品データ、図、表に示す技術的な内容、プログラム及びアルゴリズムを流用する場合は、技術内容、プログラム、アルゴリズム単位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ください。ルネサス テクノロジは、適用可否に対する責任は負いません。
6. 本資料に記載された製品は、人命にかかわるような状況の下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。本資料に記載の製品を運輸、移動体用、医療用、航空宇宙用、原子力制御用、海底中継用機器あるいはシステムなど、特殊用途へのご利用をご検討の際には、ルネサス テクノロジ、ルネサス販売または特約店へご照会ください。
7. 本資料の転載、複製については、文書によるルネサス テクノロジの事前の承諾が必要です。
8. 本資料に関し詳細についてのお問い合わせ、その他お気付きの点がございましたらルネサス テクノロジ、ルネサス販売または特約店までご照会ください。

User's Manual

HI-SH77 構築マニュアル

ユーザーズマニュアル

ルネサスマイクロコンピュータ開発環境システム
Industrial Realtime Operating System SH7000 Series

HS0770ITCN1S-2

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　1995.09

はじめに

本マニュアルでは、μITRON仕様に準拠した機器組込み用リアルタイム・マルチタスクOS(Operating System)であるＨＩ－ＳＨ７７(Hitachi Industrial Realtime Operating System SH3)のシステム構築方法について説明します。

本マニュアルは、UNIXまたはIBM PC、パーソナルコンピュータPC-9800シリーズ上で動作するMS-DOSを使用して、ＨＩ－ＳＨ７７システムの構築を行なう手順を説明します。

ＨＩ－ＳＨ７７の機能については、ＨＩ－ＳＨ７７ユーザーズマニュアルを参照してください。

HI-SH77をご使用になる前に本マニュアルをよく読んで理解してください。また、下記の関連マニュアルもお読みの上、理解してください。

＜マニュアルの構成＞

第1章では、システム構築手順の概説と必要なソフトウェアについて説明しています。
第2章では、提供ファイルのインストール方法を説明しています。
第3章では、システム構築手順の詳細を説明しています。
付録では、メモリ容量の算出表、システムの構築例、およびASCIIコード表を記載しています。

＜関連マニュアル＞

・HI-SH77　ユーザーズマニュアル
・使用するSH3マイコンのハードウェアマニュアル
・SHシリーズ　Cコンパイラ　ユーザーズマニュアル
・SHシリーズ　クロスアセンブラ　ユーザーズマニュアル
・Hシリーズ　リンケージエディタ　ユーザーズマニュアル

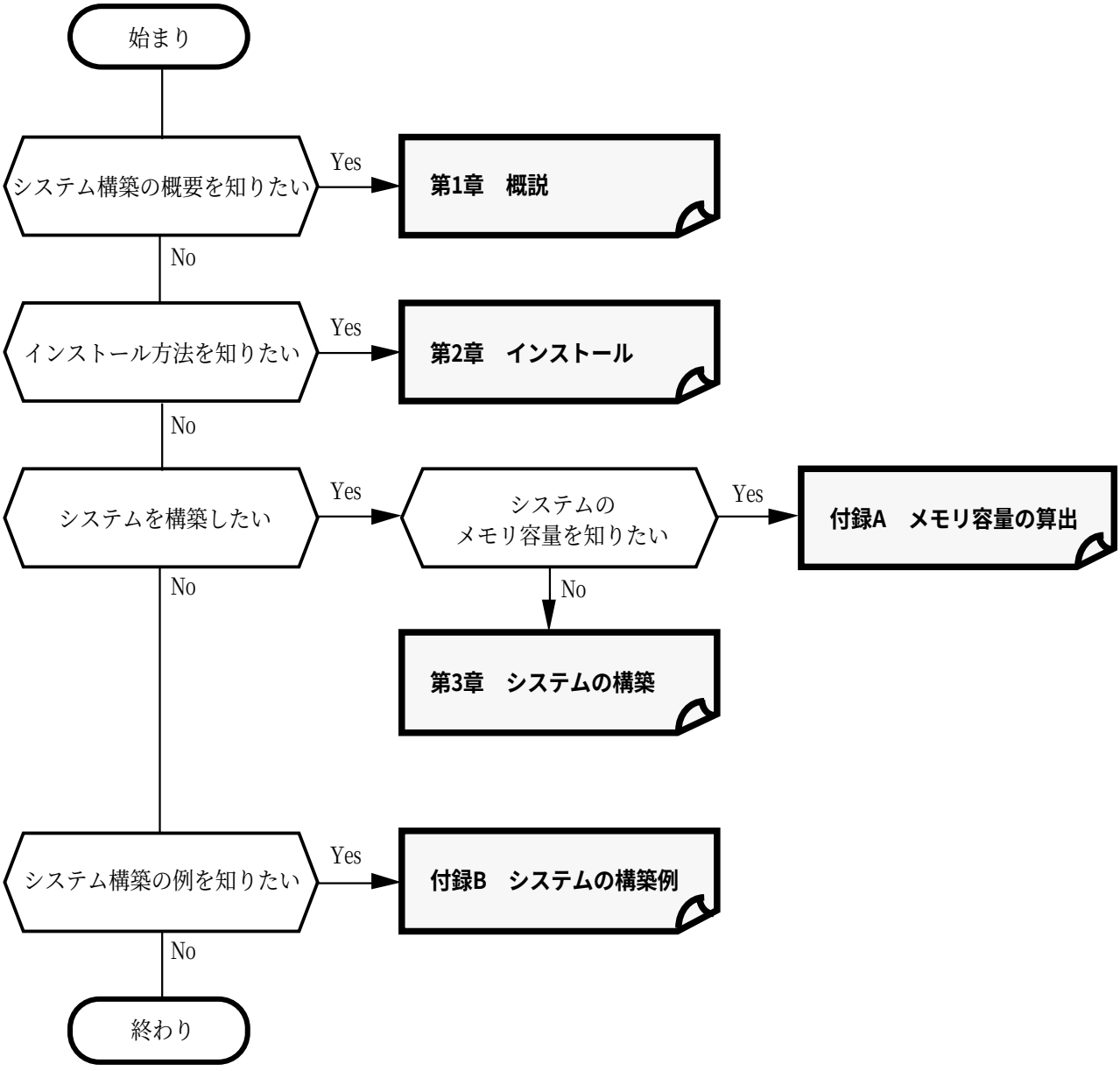
＜本マニュアルで使用する記号などの意味＞

< >	この記号で囲まれた内容を指定することを示します。
[ ]	省略してもよい項目を示します。
_	アンダーラインの部分はユーザがキー入力するコマンドラインです。
(RET)	リターンキーを示します。
△	1つ以上の空白またはタブを示します。
H'	整数定数の先頭にH'が付いているのは16進数です。それ以外は10進数です。

＜注意＞

μITRONは、"Micro Industrial TRON"の略称です。
TRONは、"The Real time Operating system Nucleus"の略称です。
UNIXは、X/Openカンパニーリミテッドがライセンスしている米国ならびに他の国における登録商標です。
IBM PCは、米国International Business Machines Corporationの登録商標です。
PC-9800は日本電気株式会社の商標です。
MS-DOSは米国マイクロソフト社の登録商標です。

マニュアルを読まれる前に、知りたい事項を下記フローからピックアップされることをおすすめします。



目　次

＜付録Ａ　メモリ容量の算出＞

＜付録Ｂ　システムの構築例＞

＜付録Ｃ　ＡＳＣＩＩコード表＞

＜索引＞

図目次

表目次

１．概　説

１．１　概　要

ＨＩ－ＳＨ７７は、日立SH3 CPU上で動作する、μITRON仕様に準拠した産業機器組込み用リアルタイム・マルチタスクOSです。

ＨＩ－ＳＨ７７の機器組み込みには、以下に示すシステム構築作業が必要です。

・ユーザプログラムの作成（ユーザシステムに必要なプログラムを作成します）
・セットアップテーブルの作成（カーネルの機能選択、環境定義を行ないます）
・メモリ領域定義プログラムの作成（ユーザシステムに必要なメモリ領域を定義します）
・実行形式プログラムの作成（実行可能なロードモジュールを作成します）

１．２　システムの構築手順

システムの構築手順を以下に示します。

(1)　ユーザプログラムの作成
　ユーザシステムに必要なプログラムをC言語またはアセンブリ言語で記述し、コンパイル、またはアセンブルを行なってオブジェクトモジュールを作成します。
(2)　セットアップテーブルの作成
　カーネルの機能選択、環境定義を行なうセットアップテーブルを提供しています。このセットアップテーブルに、ユーザシステムで必要なカーネルの機能、および環境をC言語で記述し、コンパイルを行なってオブジェクトモジュールを作成します。
(3)　メモリ領域定義プログラムの作成
　ユーザシステムに必要なメモリ領域を定義するメモリ領域定義プログラムを提供しています。このメモリ領域定義プログラムに、ユーザシステムで必要なメモリ領域をC言語で記述し、コンパイルを行なってオブジェクトモジュールを作成します。
(4)　実行形式プログラムの作成
　ユーザプログラム、セットアップテーブル、およびメモリ領域定義プログラムのオブジェクトモジュールと、カーネルのライブラリをリンケージエディタで結合し、 実行可能なロードモジュールを作成します。

１．３　プログラムのロード

作成した実行形式プログラムをユーザ実機にロードするためには、以下の方法があります。

(1)　SH3用インサーキットエミュレータを使用する方法
　SH3用インサーキットエミュレータを使用して、ユーザ実機に実行形式プログラムをロードします。
(2)　ROMを使用する方法
　実行形式プログラム（アブソリュートロードモジュール）をHシリーズオブジェクトコンバータでモトローラSタイプロードモジュールに変換し、ROMにプログラムを書き込み、ユーザ実機に搭載します。

図１－１にシステムの構築手順を示します。

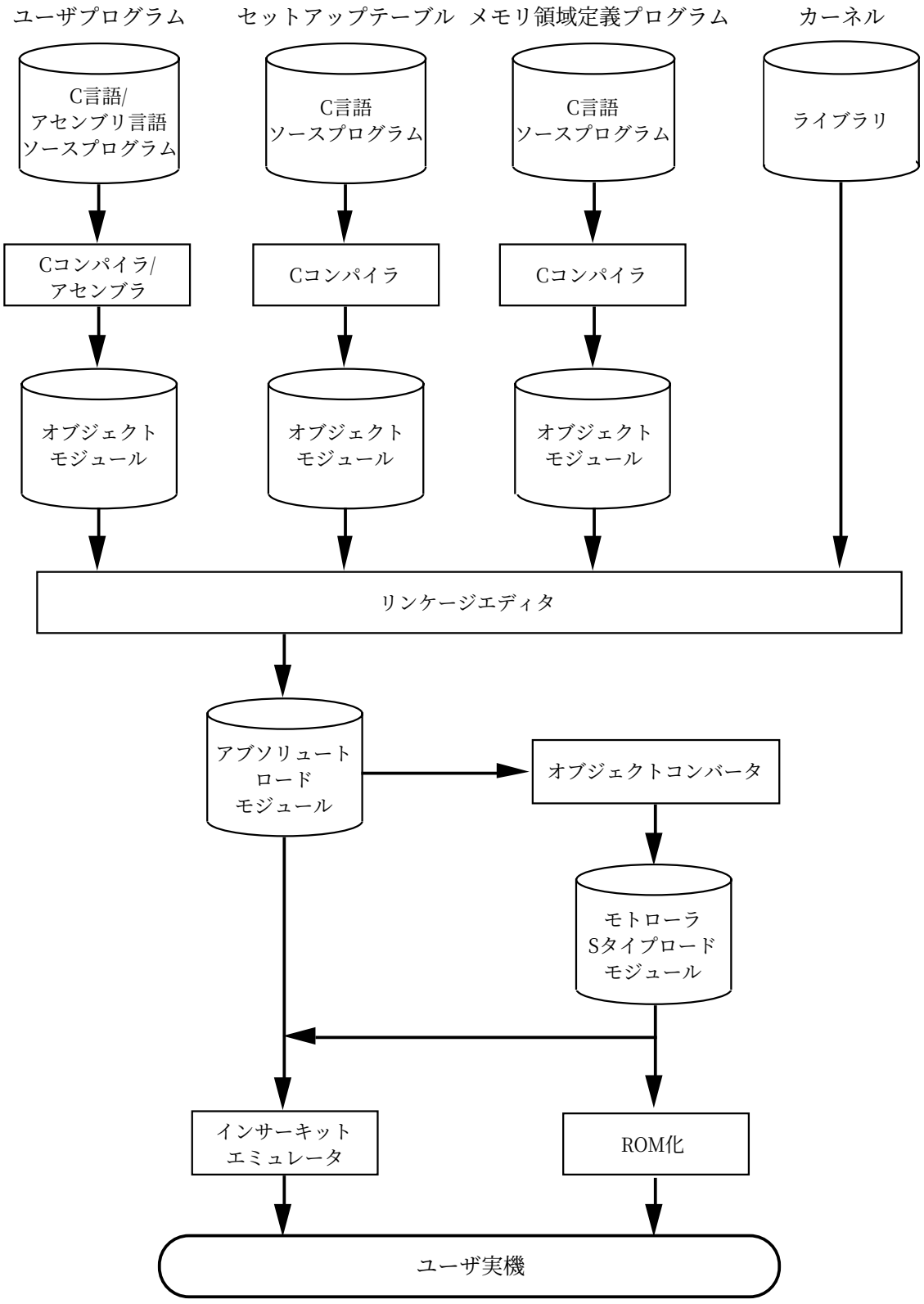


図１－１　システムの構築手順

１．４　システムの構築に必要なソフトウェア

表１－１にシステムの構築に必要なソフトウェアを示します。

表１－１　システムの構築に必要なソフトウェア

項番	ソフトウェア名称	用　途
1	SHシリーズ Cコンパイラ	C言語で記述されたプログラムのコンパイルに使用します
2	SHシリーズ クロスアセンブラ	アセンブリ言語で記述されたプログラムのアセンブルに使用します
3	Hシリーズ リンケージエディタ	システムの結合に使用します
4	Hシリーズ オブジェクトコンバータ	アブソリュートロードモジュールをモトローラSタイプロードモジュールへ変換したい場合に使用します
5	Hシリーズ ライブラリアン	ユーザプログラムのオブジェクトモジュールをライブラリ化したい場合に使用します

２．インストール

２．１　インストール手順

インストール手順についての詳細は『HI-SH77ソフトウェア引渡し添付資料』を参照してください。

２．２　提供ファイル

表２－１に提供ファイルの一覧を示します。

表２－１　提供ファイル一覧

分　類		ファイル名	内　　容
カーネル		hiknl.lib	カーネル ライブラリ（パラメータチェック機能なし）
		hiknlp.lib	カーネル ライブラリ（パラメータチェック機能あり）
		rst_srv.obj	リセットサービスルーチンオブジェクトモジュール
		exp_srv.obj	例外サービスルーチンオブジェクトモジュール
システムコール インタフェース	C言語	itron.h	C言語プログラム用ヘッダ
	アセンブリ言語	itron.def	アセンブリ言語プログラム用ヘッダ
システム構築用ファイル		himake.bat	システム構築用バッチファイル（PC-9800版,IBM PC版のみ提供）
		himake	システム構築用makeファイル（UNIX版,PC-9800版のみ提供）
		himake.sub	システム構築用リンケージサブコマンドファイル
セットアップ テーブル	ユーザ定義	hisuptbl.c	セットアップテーブルソースプログラム
		hisuptbl.obj	セットアップテーブルオブジェクトモジュール
	システム定義	hisuptbl.h	セットアップテーブルヘッダ
		hisuptbl.inc	セットアップテーブルインクルード
メモリ領域定義 プログラム	タスク用 スタック領域	hitskstk.h	タスク用スタック領域定義ヘッダ
		hitskstk.c	タスク用スタック領域定義ソースプログラム
		hitskstk.obj	タスク用スタック領域定義オブジェクトモジュール
	割込み ハンドラ用 スタック領域	hiintstk.h	割込みハンドラ用スタック領域定義ヘッダ
		hiintstk.c	割込みハンドラ用スタック領域定義ソースプログラム
		hiintstk.obj	割込みハンドラ用スタック領域定義オブジェクトモジュール
	メモリプール 領域	himempol.h	メモリプール領域定義ヘッダ
		himempol.c	メモリプール領域定義ソースプログラム
		himempol.obj	メモリプール領域定義オブジェクトモジュール
	トレース バッファ領域	hitrcbuf.h	トレースバッファ領域定義ヘッダ
		hitrcbuf.c	トレースバッファ領域定義ソースプログラム
		hitrcbuf.obj	トレースバッファ領域定義オブジェクトモジュール
ユーザ プログラム （サンプル）	ベクタ テーブル	hivcttbl.c	ベクタテーブルソースプログラム
		hivcttbl.obj	ベクタテーブルオブジェクトモジュール
	トラップベクタ テーブル	hitrptbl.c	トラップベクタテーブルソースプログラム
		hitrptbl.obj	トラップベクタテーブルオブジェクトモジュール
	初期化処理	himcuini.c	MCU初期化ルーチンソースプログラム
		himcuini.obj	MCU初期化ルーチンオブジェクトモジュール
		hisysini.c	システム初期化ハンドラソースプログラム
		hisysini.obj	システム初期化ハンドラオブジェクトモジュール
	異常処理	hisysdwn.c	システム異常終了処理ルーチンソースプログラム
		hisysdwn.obj	システム異常終了処理ルーチンオブジェクトモジュール
		hiintdwn.c	未定義割込み異常処理ルーチンソースプログラム
		hiintdwn.obj	未定義割込み異常処理ルーチンオブジェクトモジュール
		hitrpdwn.c	未定義トラップ異常処理ルーチンソースプログラム
		hitrpdwn.obj	未定義トラップ異常処理ルーチンオブジェクトモジュール
	コンソール ドライバ	hicnsdrv.h	コンソールドライバヘッダ
		hicnsdrv.c	コンソールドライバソースプログラム
		hicnsdrv.obj	コンソールドライバオブジェクトモジュール
	タイマ ドライバ	hitmrdrv.h	タイマドライバヘッダ
		hitmrdrv.c	タイマドライバソースプログラム
		hitmrdrv.obj	タイマドライバオブジェクトモジュール
HI-SH77システム （サンプル）		hisystem.abs	HI-SH77システムアブソリュートロードモジュール
		hisystem.mot	HI-SH77システムモトローラSタイプロードモジュール
		hisystem.map	HI-SH77システムリンケージマップリスト

３．システムの構築

３．１　概要

システムの構築には、次の作業が必要です。

(1)　ユーザプログラムの作成
ユーザシステムに必要となる、以下のユーザプログラムを作成します。

・ベクタテーブル
・トラップベクタテーブル
・MCU初期化ルーチン
・未定義割込み異常処理ルーチン
・未定義トラップ異常処理ルーチン
・システム異常終了処理ルーチン
・タスク
・システム初期化ハンドラ
・割込みハンドラ
・拡張SVCハンドラ
・例外処理ルーチン
・デバイスドライバ（タイマドライバ、コンソールドライバ等）

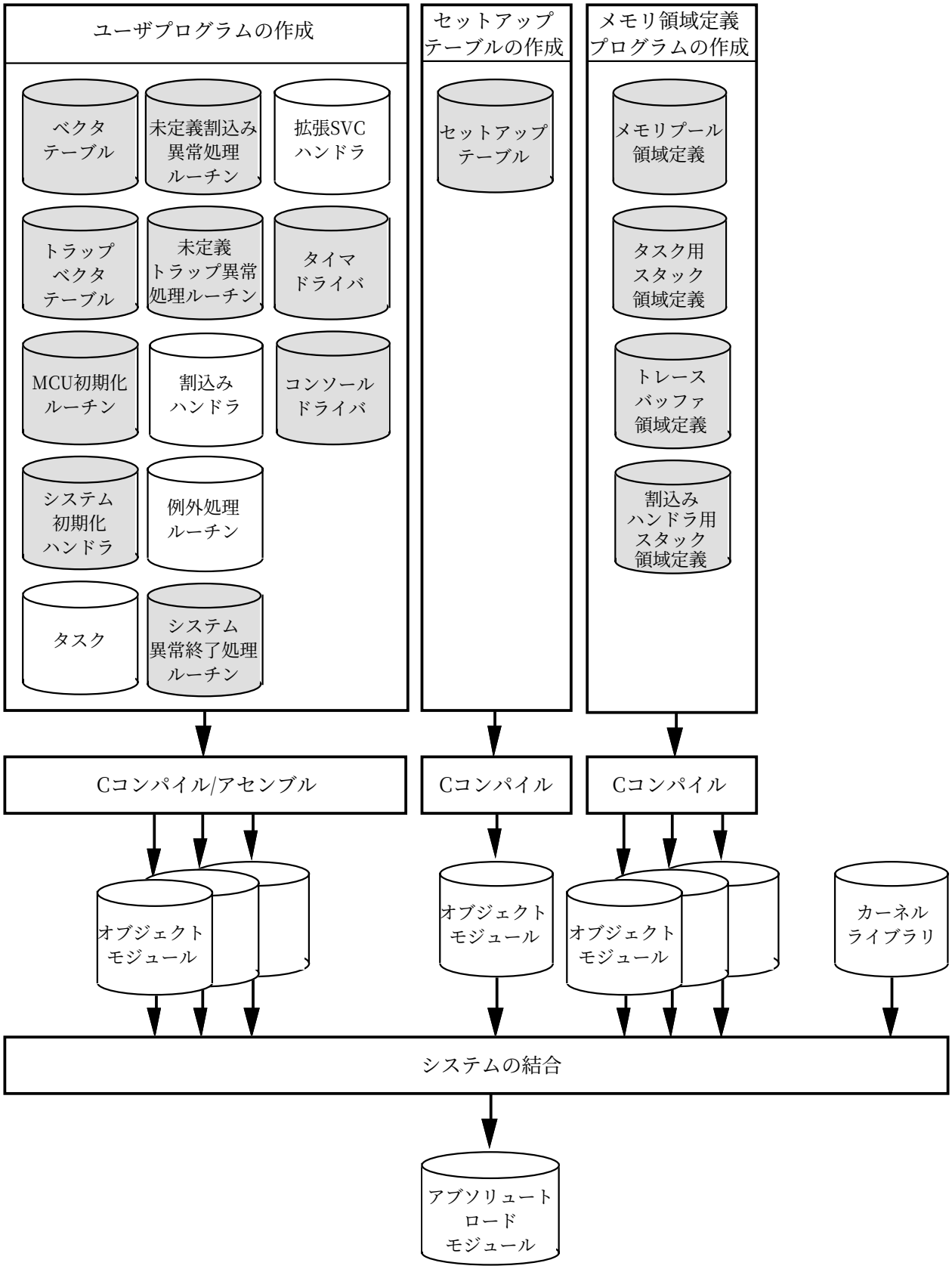
(2)　セットアップテーブルの作成
ユーザシステムに必要となるカーネル機能の選択、および環境の定義を行なうセットアップテーブルを作成します。

(3)　メモリ領域定義プログラムの作成
ユーザシステムに必要となる、以下のメモリ領域定義プログラムを作成します。

・タスク用スタック領域定義プログラム
・メモリプール領域定義プログラム
・トレースバッファ領域定義プログラム
・割込みハンドラ用スタック領域定義プログラム

(4)　実行形式プログラムの作成
(1)～(3)で作成したプログラムをコンパイル、またはアセンブルして、オブジェクトモジュールを作成します。次に、作成したオブジェクトモジュールとカーネルライブラリを結合して、実行可能なロードモジュールを作成します。

図３－１にシステム構築の概要を示します。



【注】網かけのファイルは、例題システムの環境を定義、記述したサンプルプログラムを提供しています

図３－１　システム構築の概要

３．２　ユーザプログラムの作成

ユーザシステムに必要となるプログラムをC言語、またはアセンブリ言語で作成します。

３．２．１　ベクタテーブルの作成

ベクタテーブルは、ベクタ番号（例外コードを右へ３ビットシフトすることで算出されます）順に割込みハンドラの先頭アドレスを登録したテーブルです。

ベクタテーブルの作成は、サンプルプログラム（hivcttbl.c）を参考にして作成されることをおすすめします。

ベクタテーブルは、必ず作成してください

表３－１にベクタテーブルの内容を示します。

ベクタテーブルに登録する割込みハンドラの先頭アドレスは、シンボル名によって登録することができます。登録したシンボル名は、システム結合時にリンケージエディタにより、アドレス値に変換されます。

表３－１　ベクタテーブルの内容

割込み要因		ベクタ番号	登録プログラム	シンボル名
パワーオンリセット	PC	H'00	MCU初期化ルーチンの先頭アドレス	ユーザ任意
マニュアルリセット	PC	H'01	MCU初期化ルーチンの先頭アドレス	ユーザ任意
ユーザ定義割込み		上記以外	定義する割込みハンドラの先頭アドレス	ユーザ任意
未定義割込み		上記以外	未定義割込み異常処理ルーチンの先頭アドレス	（注１参照）

注１　未定義割込み異常処理ルーチンのシンボル名は、割込みの場合"hi_undefint"、例外の場合"hi_undefexp"を登録します。
　　未定義割込み異常処理ルーチンは、例外コードを受け取ることができます。
　　例外コードは、CPUが例外発生時に割込み事象レジスタ(INTEVT)または例外事象レジスタ(EXPEVT)に書き込む値です。

注２　アセンブリ言語でベクタテーブルを作成する場合は、シンボル名の先頭に、アンダーバー（_）を1つ付加して登録してください。例えば、"hi_undefint"の場合は"_hi_undefint"を登録します。

(1)　パワーオンリセットとマニュアルリセット

このベクタには、MCU初期化ルーチンの先頭アドレスを登録します。パワーオンリセットとマニュアルリセットの区別が必要ない場合は、同一のシンボル名（サンプルプログラムでは"hi_mcuini"）を登録します。

なお、MCU初期化ルーチンのスタックポインタ(R15)の初期値はシステム構築用リンケージサブコマンドファイル(himake.sub)に登録してください。詳細は、「３．６．２　システムの結合」を参照してください。

(2)　ユーザ定義割込み

ユーザが使用する割込みのベクタには、対応する割込みハンドラの先頭アドレスを登録します。カーネルの時間管理機能を使用する場合は、タイマ割込みハンドラの先頭アドレスを登録してください。

(3)　未定義割込み

ユーザが使用しない割込みのベクタには、カーネルの未定義割込み／例外コード解析ルーチンの先頭アドレスを登録してください。これによって、システム異常の原因究明や、システム障害を未然に防ぐことができます。

図３－２に未定義割込み異常処理ルーチンの構成を示します。

未定義割込み／例外コード解析ルーチンのシンボル名は、割込みの場合"hi_undefint"、例外の場合"hi_undefexp"です。これらのルーチンは、カーネルライブラリに含まれています。

未定義割込み異常処理ルーチンには、パラメータとして例外コードが渡されます。

例外コードは、例外発生時にCPUが割込み事象レジスタ(INTEVT)または例外事象レジスタ(EXPEVT)に設定する値です。

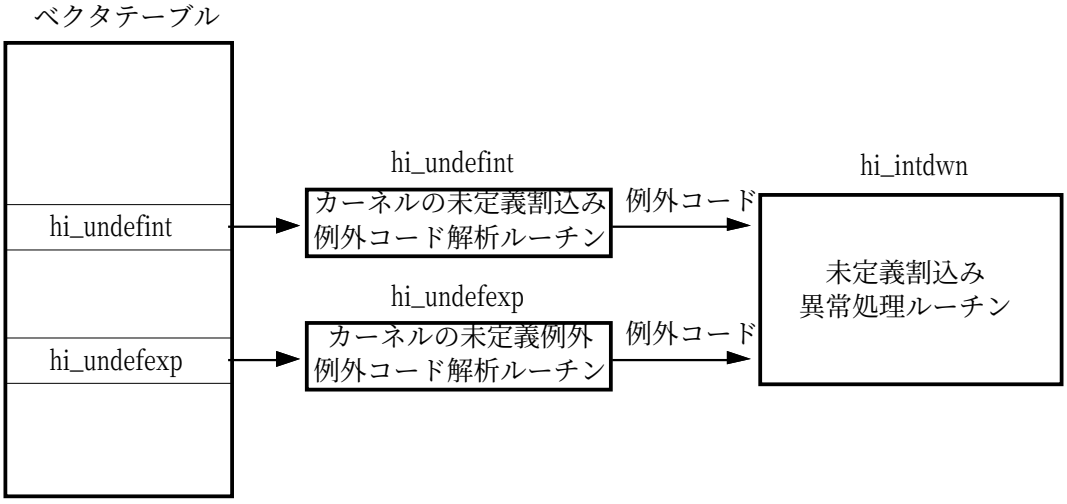


図３－２　未定義割込み異常処理ルーチンの構成

３．２．２　トラップベクタテーブルの作成

トラップベクタテーブルは、TRAPA命令の8ビットイミディエイトデータの番号順に、TRAPA命令による例外処理ルーチンの先頭アドレスを登録したテーブルです。

トラップベクタテーブルの作成は、サンプルプログラム（hitrptbl.c）を参考にして作成されることをおすすめします。

トラップベクタテーブルは、必ず作成してください

表３－２にトラップベクタテーブルの内容を示します。

トラップベクタテーブルに登録する例外処理ルーチンの先頭アドレスは、シンボル名によって登録することができます。登録したシンボル名は、システム結合時にリンケージエディタにより、アドレス値に変換されます。

表３－２　トラップベクタテーブルの内容

割込み要因		ベクタ番号	登録プログラム	シンボル名
トラップ命令	TRAPA #57	H'39	ret_excシステムコール処理の先頭アドレス（必須）	_ 0Hret_exc
	TRAPA #58	H'3A	システム予約	（注１参照）
	TRAPA #59	H'3B	システム予約	（注１参照）
	TRAPA #60	H'3C	sys_clk システムコール処理の先頭アドレス（時間管理機能使用時のみ）	_ 0Hsys_clk
	TRAPA #61	H'3D	ret_int システムコール処理の先頭アドレス（必須）	_ 0Hret_int
	TRAPA #62	H'3E	非タスク部用システムコール処理の先頭アドレス（必須）	_ 0Henn_svc
	TRAPA #63	H'3F	タスク部用システムコール処理の先頭アドレス（必須）	_ 0Hent_svc
ユーザ定義トラップ		上記以外	定義する例外処理ルーチンの先頭アドレス	ユーザ任意
未定義トラップ		上記以外	未定義トラップ異常処理ルーチンの先頭アドレス	（注２参照）

注１　未定義トラップ異常処理ルーチンのシンボル名を登録します。
注２　未定義トラップ異常処理ルーチンのシンボル名は"hi_undeftrp"を登録します。
　　未定義トラップ異常処理ルーチンは、TRAPA例外レジスタ(TRA)の値を受け取ることができます。TRAレジスタの値は、TRAPA例外発生時にCPUが書き込む値（TRAPA命令の８ビットイミディエイトデータ×４）です。
注３　アセンブリ言語でベクタテーブルを作成する場合は、シンボル名の先頭に、アンダーバー（_）を1つ付加して登録してください。例えば、"_0Hent_svc"の場合は"_ _0Hent_svc"を登録します。

(1)　トラップ命令

TRAPA #57からTRAPA #63までのベクタは、カーネルが使用します。表３－２に示すシンボル名を必ず登録してください。TRAPA #58、およびTRAPA #59は、将来用のシステム予約ベクタです。このベクタには、未定義トラップ異常処理ルーチンのシンボル名を登録してください。

ユーザシステムにおいて、sys_clkシステムコールを使用しない場合は、TRAPA #60のベクタに、未定義トラップ異常処理ルーチンのシンボル名を登録してください。この登録によって、sys_clkシステムコールの処理プログラムが組み込まれなくなり、システムのプログラムサイズが小さくなります。

なお、他のシステムコールの機能選択については、「３．３．４　システムコールの定義」を参照してください。

(2)　ユーザ定義トラップ

ユーザが使用するトラップのベクタには、対応する例外処理ルーチンの先頭アドレスを登録します。

(3)　未定義トラップ

ユーザが使用しないトラップのベクタには、未定義トラップTRA値解析ルーチンの先頭アドレスを登録してください。これによって、システム異常の原因究明や、システム障害を未然に防ぐことができます。

図３－３に未定義トラップ異常処理ルーチンの構成を示します。

未定義トラップTRA値解析ルーチンのシンボル名は、"hi_undeftrp"です。

未定義トラップ異常処理ルーチンには、TRAレジスタの値が渡されます。TRAレジスタの値は、TRAPA例外発生時にCPUが書き込む値(TRAPA命令の8ビットイミディエイトデータ × 4)です。

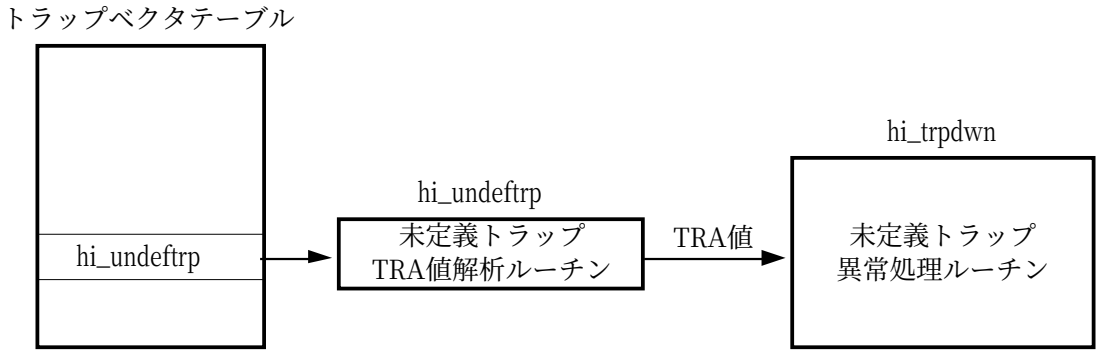


図３－３　未定義トラップ異常処理ルーチンの構成

３．２．３　MCU初期化ルーチンの作成

MCU初期化ルーチンは、パワーオンリセットまたはマニュアルリセットが発生したときに起動され、MCUの初期化とカーネルの起動を行なうプログラムです。

MCU初期化ルーチンの作成は、サンプルプログラム（himcuini.c）を参考にして作成されることをおすすめします。

MCU初期化ルーチンは、必ず作成してください

作成したMCU初期化ルーチンは、ベクタテーブルに登録します。

MCU初期化ルーチンの最終処理は、カーネル起動処理の呼び出し（呼び出し元へは戻りません）を行なってください。カーネル起動処理のシンボル名は、C言語の場合"_0Hrs__knl"、アセンブリ言語の場合"__0Hrs__knl"です。

なお、MCU初期化ルーチンについての詳細は、『HI-SH77ユーザーズマニュアル』の「２．１２　MCUの初期化」を参照してください。

３．２．４　未定義割込み異常処理ルーチンの作成

未定義割込み異常処理ルーチンは、未定義割込みが発生したときに実行されるプログラムです。
未定義割込み異常処理ルーチンの作成は、サンプルプログラム（hiintdwn.c）を参考にして作成されることをおすすめします。

未定義割込み異常処理ルーチンは、必ず作成してください

未定義割込み異常処理ルーチンのシンボル名は、C言語の場合"hi_intdwn"、アセンブリ言語の場合"_hi_intdwn"にしてください。
未定義割込み異常処理ルーチンには、未定義割込みが発生したときの例外コード(INTEVTまたはEXPEVTレジスタの内容)がパラメータとして渡されます。TRAレジスタの値は、TRAPA例外発生時にCPUが書き込む値(TRAPA命令の8ビットイミディエイトデータ × 4)です。

３．２．５　未定義トラップ異常処理ルーチンの作成

未定義トラップ異常処理ルーチンは、未定義トラップが発生したときに実行されるプログラムです。
未定義トラップ異常処理ルーチンの作成は、サンプルプログラム（hitrpdwn.c）を参考にして作成されることをおすすめします。

未定義トラップ異常処理ルーチンは、必ず作成してください

未定義トラップ異常処理ルーチンのシンボル名は、C言語の場合"hi_trpdwn"、アセンブリ言語の場合"_hi_trpdwn"にしてください。
未定義トラップ異常処理ルーチンには、未定義トラップが発生したときのTRAレジスタの値がパラメータとして渡されます。TRAレジスタの値は、TRAPA例外発生時にCPUが書き込む値(TRAPA命令の8ビットイミディエイトデータ × 4)です。

３．２．６　システム異常終了処理ルーチンの作成

システム異常終了処理ルーチンは、システム実行中に致命的なエラーが発生したときに実行されるプログラムです。
システム異常終了処理ルーチンの作成は、サンプルプログラム（hisysdwn.c）を参考にして作成されることをおすすめします。

システム異常終了処理ルーチンは、必ず作成してください

システム異常終了処理ルーチンのシンボル名は、C言語の場合"hi_sysdwn"、アセンブリ言語の場合"_hi_sysdwn"にしてください。
システム異常終了処理ルーチンには、エラー種別、エラーコード、およびシステムダウン情報がパラメータとして渡されます。
なお、システム異常終了処理ルーチンについての詳細は、『HI-SH77ユーザーズマニュアル』の「２．１４　システムの異常終了処理」を参照してください。

３．２．７　タスクの作成

タスクは、独立して並列に処理可能なプログラムの単位です。
ユーザシステムに応じて、タスクを作成します。
作成したタスクは、セットアップテーブルに初期登録タスクとして定義するか、またはcre_tskシステムコールをタスクから発行してください。
初期登録タスクの定義方法については、「３．３．６　タスク管理情報の定義」を参照してください。
なお、タスクについての詳細は、『HI-SH77ユーザーズマニュアル』の「２．３　タスク」を参照してください。

３．２．８　システム初期化ハンドラの作成

システム初期化ハンドラは、カーネルが起動処理を行なった後、タスクを起動する前に実行されるプログラムです。
ユーザシステムに応じて、システム初期化ハンドラを作成します。システム初期化ハンドラを作成することで、マルチタスク環境へ移行する前に、初期登録タスクの起動、資源の初期化、ハードウェアの初期化などの初期処理を行なうことができます。
システム初期化ハンドラの作成は、サンプルプログラム（hisysini.c）を参考にして作成されることをおすすめします。
作成したシステム初期化ハンドラは、セットアップテーブルに定義します。
システム初期化ハンドラの定義方法については、「３．３．１０　システム初期化ハンドラ情報の定義」を参照してください。
なお、システム初期化ハンドラについての詳細は、『HI-SH77ユーザーズマニュアル』の「２．１３　システムの起動処理」を参照してください。

３．２．９　割込みハンドラの作成

割込みハンドラは、割込み発生時に実行されるプログラムです。CPUが割込みを受け付けると、カーネルの例外サービスルーチンを介して起動されます。
ユーザシステムに応じて、各種割込みハンドラを作成します。カーネルの時間管理機能（sys_clk,set_tim,iset_tim,get_tim,iget_tim,wai_tskシステムコール）を使用するときは、タイマ割込みハンドラを作成する必要があります。
作成した割込みハンドラは、ベクタテーブルに登録します。
なお、割込みハンドラについての詳細は、『HI-SH77ユーザーズマニュアル』の「２．７　割込み」を参照してください。また、時間管理機能についての詳細は、『HI-SH77ユーザーズマニュアル』の「２．１０　時間」を参照してください。

３．２．１０　拡張SVCハンドラの作成

拡張SVCハンドラは、タスクからの拡張SVCの発行によって実行されるプログラムです。
ユーザシステムに応じて、拡張SVCハンドラを作成します。拡張SVCハンドラを作成することで、カーネルの機能拡張を行なうことができます。
作成した拡張SVCハンドラは、セットアップテーブルに定義します。
拡張SVCハンドラの定義方法については、「３．３．９　拡張SVC情報の定義」を参照してください。
なお、拡張SVCについての詳細は、『HI-SH77ユーザーズマニュアル』の「２．１１　拡張SVC」を参照してください。

３．２．１１　例外処理ルーチンの作成

例外処理ルーチンは、例外発生時に実行されるプログラムです。例外が発生すると、カーネルの例外サービスルーチンを介して実行されます。例外処理ルーチンは、例外発生前のタスクや割込みハンドラなどの一部として動作します。

ユーザシステムに応じて、例外処理ルーチンを作成します。

作成した例外処理ルーチンは、ベクタテーブルまたはトラップベクタテーブルに登録します。

なお、例外処理ルーチンについての詳細は、『HI-SH77ユーザーズマニュアル』の「２．８　例外」を参照してください。

３．３　セットアップテーブルの作成

３．３．１　セットアップテーブルの定義内容

セットアップテーブルは、システムに組み込むカーネル機能の選択と、カーネル実行に必要な動作環境を定義するテーブルです。

表３－３にセットアップテーブルの定義内容を示します。

表３－３　セットアップテーブルの定義内容

項番	定　義　項　目	定　義　内　容
1	セットアップテーブルチェック	・セットアップテーブルのチェック機能の使用／未使用
2	システムコール	・各システムコールの使用／未使用
3	カーネル情報	・カーネル割込みマスクレベル ・割込みネスト数（＞　カーネル割込みマスクレベル） ・割込みネスト数（≦　カーネル割込みマスクレベル）
4	タスク管理情報	・最大タスクID ・最大タスク優先度 ・初期登録タスク数 ・タスク用スタック領域最終アドレス ・初期登録タスク情報
5	同期／通信管理情報	・最大イベントフラグID ・最大セマフォID ・最大メイルボックスID
6	メモリプール管理情報	・最大メモリプールID ・メモリプール情報
7	拡張SVC情報	・拡張SVCの最大機能コード ・拡張SVCハンドラの開始アドレス ・拡張SVCハンドラの最大スタックサイズ
8	システム初期化ハンドラ情報	・システム初期化ハンドラの使用／未使用 ・システム初期化ハンドラの開始アドレス ・システム初期化ハンドラのスタックサイズ
9	トレース情報	・トレース機能の使用／未使用 ・トレースバッファ情報

３．３．２　セットアップテーブルの構成

セットアップテーブルは、次の3つのC言語記述したファイルから構成しています。

(a)　hisuptbl.c

ユーザは、このファイルを修正してセットアップテーブルを作成します。

このファイルには、すでに、例題システムの環境が定義されています。ユーザは、このファイルを修正して（再定義して）、ユーザシステムに応じた環境を定義してください。

なお、メモリ領域（タスク用スタック、メモリプール、トレースバッファ）の定義は、メモリ領域定義プログラムで行ないます。本ファイルは、メモリ領域定義プログラムで定義した内容を、セットアップテーブルに反映するため、以下のファイルを読み込みます。

・hitskstk.h（タスク用スタック領域定義ヘッダ）
・himempol.h（メモリプール領域定義ヘッダ）
・hitrcbuf.h（トレースバッファ領域定義ヘッダ）

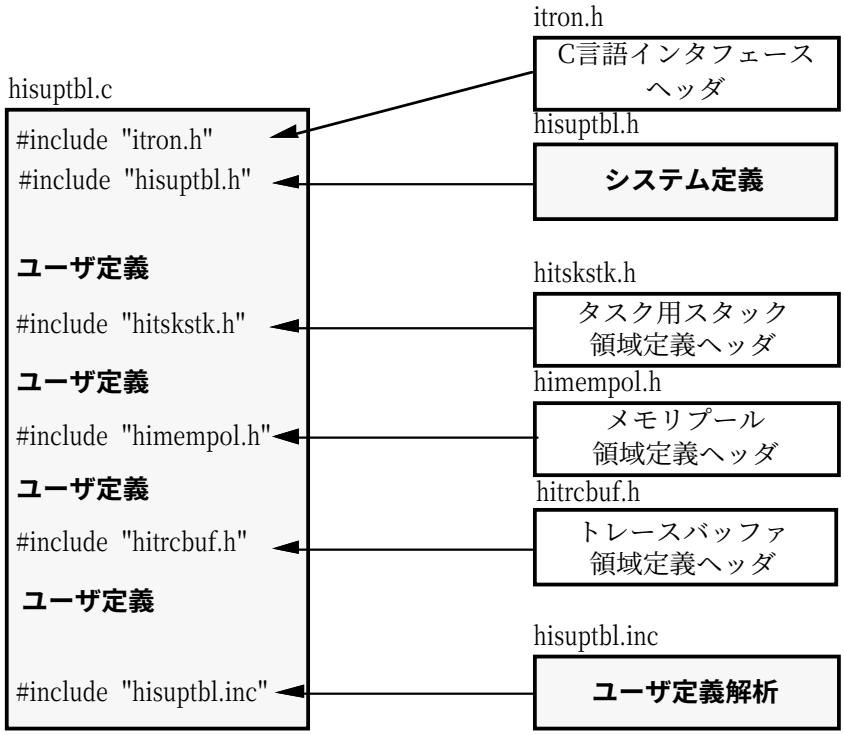
(b)　hisuptbl.h

このファイルでは、ユーザが各種情報を定義するために必要なシステム情報を定義しています。
このファイルは、絶対に変更しないでください。

(c)　hisuptbl.inc

このファイルは、ユーザが定義した各種情報を解析します。
このファイルは、絶対に変更しないでください。

図３－４にセットアップテーブルの構成を示します。



[注]網かけのファイルは、前述で説明したセットアップテーブルを構成する３つのファイルです。

図３－４　セットアップテーブルの構成

３．３．３　セットアップテーブルのチェック機能の定義

セットアップテーブルのチエック機能の使用／未使用を定義します。

セットアップテーブルのチェック機能は、カーネルの起動処理で、セットアップテーブルのユーザ定義内容をチェックし、誤りがあれば、システム異常終了処理ルーチンを呼び出し、起動処理を中断します。

システム異常終了処理ルーチンへは、エラーコードをパラメータに設定します。エラーコードの詳細は、『HI-SH77ユーザーズマニュアル』の「Ｅ．３　セットアップテーブルエラーコード」を参照してください。

セットアップテーブルのユーザ定義内容に誤りがなくなったら、セットアップテーブルのチエック機能を未使用に定義することをおすすめします。これによって、システムのプログラムサイズを小さくすることができます。

セットアップテーブルのチェック機能の使用／未使用は、hi_chksutに"USE"（使用）、または"NOTUSE"（未使用）を定義します。

図３－５にセットアップテーブルのチェック機能の定義例を示します。

```
/************************************************************************/
/*                                                                      */
/*          Definition setup table check function use in user system    */
/*                                                                      */
/*          Usage   :   #define hi_chksut { USE | NOTUSE }              */
/*                      USE     :   use the setup table check function  */
/*                      NOTUSE  :   not use the setup table check function */
/*                                                                      */
/************************************************************************/
#define hi_chksut       USE             /* setup table check function   */
```

図３－５　セットアップテーブルのチェック機能の定義例

３．３．４　システムコールの定義

各システムコールの使用／未使用を定義します。この定義によって、ユーザシステムで使用するシステムコールを選択することができます。

ユーザシステムで使用しないシステムコールは、未使用に定義することをおすすめします。これによって、システムのプログラムサイズを小さくすることができます。

なお、ret_int,ret_exc,sys_clkシステムコールの使用／未使用の選択は、トラップベクタテーブルで行ないます。詳細は、「３．２．２　トラップベクタテーブルの作成」を参照してください。

システムコールの使用／未使用は、hi_xyyy_zzz（xyyy_zzz はシステムコール名称）に"USE"（使用）、または"NOTUSE"（未使用）を定義します。ただし、一部のシステムコールはタスク部／非タスク部用共通に定義します。

表３－４にシステムコール使用／未使用定義用共通マクロ名を示します。

表３－４　システムコール使用／未使用定義用共通マクロ名

項番	システムコール	使用／未使用定義用共通マクロ名
1	get_tid、iget_tidシステムコール	hi_get_tid
2	tsk_sts、itsk_stsシステムコール	hi_tsk_sts
3	can_wup、ican_wupシステムコール	hi_can_wup
4	clr_flg、iclr_flgシステムコール	hi_clr_flg
5	pol_flg、ipol_flgシステムコール	hi_pol_flg
6	flg_sts、iflg_stsシステムコール	hi_flg_sts
7	preq_sem、ipreq_semシステムコール	hi_preq_sem
8	sem_sts、isem_stsシステムコール	hi_sem_sts
9	prcv_msg、iprcv_msgシステムコール	hi_prcv_msg
10	mbx_sts、imbx_stsシステムコール	hi_mbx_sts
11	chg_ims、ichg_imsシステムコール	hi_chg_ims
12	ims_sts、iims_stsシステムコール	hi_ims_sts
13	pget_blk、ipget_blkシステムコール	hi_pget_blk
14	rel_blk、irel_blkシステムコール	hi_rel_blk
15	mpl_sts、impl_stsシステムコール	hi_mpl_sts
16	set_tim、iset_timシステムコール	hi_set_tim
17	get_tim、iget_timシステムコール	hi_get_tim
18	get_ver、iget_verシステムコール	hi_get_ver

図３－６にシステムコールの定義例を示します。

```
/*****************************************************************************/
/*                                                                           */
/*          Definition SVC use in user system                                */
/*                                                                           */
/*          Usage   :   #define hi_xyyy_zzz { USE | NOTUSE }                 */
/*                      USE     :   use the SVC (xyyy_zzz)                   */
/*                      NOTUSE  :   not use the SVC (xyyy_zzz)               */
/*                                                                           */
/*****************************************************************************/
/*----------------------- task management function ------------------------*/
#define hi_cre_tsk      USE             /* cre_tsk  SVC                     */
#define hi_sta_tsk      USE             /* sta_tsk  SVC                     */
#define hi_ista_tsk     USE             /* ista_tsk SVC                     */
#define hi_del_tsk      USE             /* del_tsk  SVC                     */
#define hi_ext_tsk      USE             /* ext_tsk  SVC                     */
#define hi_exd_tsk      USE             /* exd_tsk  SVC                     */
#define hi_ter_tsk      USE             /* ter_tsk  SVC                     */
#define hi_chg_pri      USE             /* chg_pri  SVC                     */
#define hi_ichg_pri     USE             /* ichg_pri SVC                     */
#define hi_rot_rdq      USE             /* rot_rdq  SVC                     */
#define hi_irot_rdq     USE             /* irot_rdq SVC                     */
#define hi_rel_wai      USE             /* rel_wai  SVC                     */
#define hi_irel_wai     USE             /* irel_wai SVC                     */
#define hi_get_tid      NOTUSE          /* get_tid  SVC                     */
#define hi_tsk_sts      USE             /* tsk_sts  SVC                     */

/*--------------- task synchronization management function ----------------*/
#define hi_sus_tsk      USE             /* sus_tsk  SVC                     */
#define hi_isus_tsk     USE             /* isus_tsk SVC                     */
#define hi_rsm_tsk      USE             /* rsm_tsk  SVC                     */
#define hi_irsm_tsk     USE             /* irsm_tsk SVC                     */
#define hi_slp_tsk      USE             /* slp_tsk  SVC                     */
#define hi_wai_tsk      USE             /* wai_tsk  SVC                     */
#define hi_wup_tsk      USE             /* wup_tsk  SVC                     */
#define hi_iwup_tsk     USE             /* iwup_tsk SVC                     */
#define hi_can_wup      USE             /* can_wup  SVC                     */

/*-------------- synchronization and communication function --------------*/
#define hi_set_flg      USE             /* set_flg  SVC                     */
#define hi_iset_flg     USE             /* iset_flg SVC                     */
#define hi_clr_flg      USE             /* clr_flg  SVC                     */
#define hi_wai_flg      USE             /* wai_flg  SVC                     */
#define hi_pol_flg      USE             /* pol_flg  SVC                     */
#define hi_flg_sts      USE             /* flg_sts  SVC                     */
#define hi_sig_sem      USE             /* sig_sem  SVC                     */
#define hi_isig_sem     USE             /* isig_sem SVC                     */
#define hi_wai_sem      USE             /* wai_sem  SVC                     */
#define hi_preq_sem     USE             /* preq_sem SVC                     */
#define hi_sem_sts      USE             /* sem_sts  SVC                     */
#define hi_snd_msg      USE             /* snd_msg  SVC                     */
#define hi_isnd_msg     USE             /* isnd_msg SVC                     */
#define hi_rcv_msg      USE             /* rcv_msg  SVC                     */
#define hi_prcv_msg     USE             /* prcv_msg SVC                     */
#define hi_mbx_sts      USE             /* mbx_sts  SVC                     */

/*--------------------- interrupt management function ---------------------*/
#define hi_chg_ims      USE             /* chg_ims  SVC                     */
#define hi_ims_sts      USE             /* ims_sts  SVC                     */

/*------------------- memory pool management function --------------------*/
#define hi_get_blk      USE             /* get_blk  SVC                     */
#define hi_pget_blk     USE             /* pget_blk SVC                     */
#define hi_rel_blk      USE             /* rel_blk  SVC                     */
#define hi_mpl_sts      USE             /* mpl_sts  SVC                     */

/*----------------------- timer management function ----------------------*/
#define hi_set_tim      USE             /* set_tim  SVC                     */
#define hi_get_tim      USE             /* get_tim  SVC                     */

/*---------------------- system management function ----------------------*/
#define hi_get_ver      NOTUSE          /* get_ver  SVC                     */
```

図３－６　システムコールの定義例

３．３．５　カーネル情報の定義

次のカーネル情報を定義します。

(1)　カーネル割込みマスクレベル

カーネル割込みマスクレベルとは、カーネル内の処理中に、一時的に割込みをマスクするレベルのことです。したがって、カーネル割込みマスクレベル以下の割込みは、カーネル実行中は一時的に禁止されます。また、カーネル割込みマスクレベルより高い割込みハンドラからは、カーネルの機能を使用することはできません。

カーネル割込みマスクレベルは、 hi_knlmsklvlに1から15の値を定義します。

(2)　割込みネスト数

割込みネスト数とは、ユーザシステムで発生する多重割込みのネスト数（使用する割込みレベルの数）のことです。

定義する割込みネスト数には、カーネル割込みマスクレベルより高い割込みのネスト数と、カーネル割込みマスクレベル以下の割込みのネスト数があります。

カーネル割込みマスクレベルよりも高い割込みのネスト数は、hi_uppintnstに0以上の値を定義します。NMIを使用する場合は、NMIのネスト数も加えてください。（ユーザシステムによっては、NMIがネストする場合があります）

カーネル割込みマスクレベル以下の割込みのネスト数は、hi_lowintnstに0から15の値を定義します。

この定義によって、割込みネスト数に応じたカーネルの作業領域が確保されます。

図３－７にカーネル実行情報の定義例を示します。

```
/************************************************************************/
/*                                                                      */
/*          Definition kernel information                               */
/*                                                                      */
/************************************************************************/
#define hi_knlmsklvl    14              /* kernel mask level of interrupt   */
                                        /*      range : [1...15]            */
#define hi_uppintnst    0               /* number of interrupt levels       */
                                        /*           (>  kernel mask level) */
                                        /*      range : [0...]              */
#define hi_lowintnst    2               /* number of interrupt levels       */
                                        /*           (<= kernel mask level) */
                                        /*      range : [0...15]            */
```

図３－７　カーネル情報の定義例

３．３．６　タスク管理情報の定義

次のタスク管理情報を定義します。

(1) 最大タスクID
ユーザシステムで使用するタスクIDの最大値を定義します。
最大タスクIDは、hi_maxtskidに0から1023の値を定義します。
例えば、5を定義した場合は、1から5までのタスクIDを使用することができます。
0を定義した場合は、タスクは未登録となり、タスクを使用することはできません。
なお、定義する値が小さいほど、カーネル作業領域は小さくなります。

(2) 最大タスク優先度
ユーザシステムで使用するタスクの最大タスク優先度を定義します。
最大タスク優先度は、hi_maxtskpriに1から255の値を定義します。
例えば、5を定義した場合は、1から5までのタスク優先度を使用することができます。
なお、定義する値が小さいほど、カーネル作業領域は小さくなります。

(3) 初期登録タスク数
カーネルの起動処理で初期登録するタスク数を定義します。
初期登録タスク数は、hi_initsknumに0から1023の値を定義します。
例えば、2を定義した場合は、2つのタスクを初期登録します。
0を定義した場合は、タスクの初期登録は行ないません。

図３－８にタスク管理情報の定義例を示します。

```
/*----------------------------------------------------------------------*/
/*                                                                      */
/*          Definition task information                                 */
/*                                                                      */
/*----------------------------------------------------------------------*/
#define hi_maxtskid     5               /* max task id                  */
                                        /*      range : [0...1023]      */
#define hi_maxtskpri    5               /* max task priority            */
                                        /*      range : [1...255]       */
#define hi_initsknum    2               /* number of initial task       */
                                        /*      range : [0...1023]      */
```

図３－８　タスク管理情報の定義例

(4) タスク用スタック領域最終アドレス

タスクが使用するスタック領域の最終アドレスを定義します。

タスク用スタック領域の最終アドレスは、INITSP型に宣言された_0Hinitspテーブルに定義します。

INITSP型は、次のように宣言してあり、su_initsp（タスク用スタック領域最終アドレス）のメンバ変数で構成される構造体です。

```
typedef struct  initsp  {
        VW      *su_initsp;             /* task stack end address          */
} INITSP;
```

タスク用スタック領域の最終アドレスは、タスクIDの1から最大タスクIDまで順に、_0Hinitspテーブルに定義します。最大タスクIDが0の場合は、定義する必要はありません。

「共有スタック機能」を使用する場合は、共有するタスクIDと同じタスク用スタック領域の最終アドレスを定義します。

図３－９にタスク用スタック領域最終アドレスの定義例を示します。

```
/*--------------------- define task stack end address ----------------------*/
#if     hi_maxtskid                     /* case of hi_maxtskid > 0          */
#include        "hitskstk.h"            /* include "hitskstk.h"             */   ･･･(a)
extern  VW      hi_tskstk1[];           /* task stack of tskid = 1          */   ･･･(b)
extern  VW      hi_tskstk2[];           /* task stack of tskid = 2          */
extern  VW      hi_tskstk3[];           /* task stack of tskid = 3          */
extern  VW      hi_tskstk4[];           /* task stack of tskid = 4, 5       */

const   INITSP  _0Hinitsp[hi_maxtskid] = {
        /* define task stack end address    */
            (VW *)&hi_tskstk1[(hi_tskstksz1) / sizeof(VW)], /* tskid = 1    */   ･･･(c)
            (VW *)&hi_tskstk2[(hi_tskstksz2) / sizeof(VW)], /* tskid = 2    */
            (VW *)&hi_tskstk3[(hi_tskstksz3) / sizeof(VW)], /* tskid = 3    */
            (VW *)&hi_tskstk4[(hi_tskstksz4) / sizeof(VW)], /* tskid = 4    */
            (VW *)&hi_tskstk4[(hi_tskstksz4) / sizeof(VW)], /* tskid = 5    */
        };
```

図３－９　タスク用スタック領域最終アドレスの定義例

(a) タスク用スタック領域定義ヘッダを読み込んで、タスク用スタックサイズをシンボル参照します。

(b) タスク用スタック領域の先頭アドレスをシンボル参照します。

(c) タスク用スタック領域の最終アドレスを定義します。

タスク用スタックサイズは、タスク用スタック領域定義ヘッダ（hitskstk.h）で、"hi_tskstksz<n>"のシンボルに定義します。タスク用スタック領域は、タスク用スタック領域定義ソース（hitskstk.c）で、VW型の配列を"hi_tskstk<n>"のシンボルで定義します。したがって、タスク用スタック領域の最終アドレスは、"(VW *)&hi_tskstk<n>[(hi_tskstksz<n>) / sizeof(VW)]"を記述することで求めることができます。（<n>はスタック番号を表わします）

なお、タスク用スタック領域定義プログラム（hitskstk.h, hitskstk.c）についての詳細は、「３．４．３　タスク用スタック領域定義プログラムの作成」を参照してください。

(5) 初期登録タスク情報

初期登録するタスク情報を定義します。

初期登録タスク情報は、INITSK型に宣言された_0Hinitskテーブルに定義します。

INITSK型は、次のように宣言してあり、su_tskid（タスクID）、su_itskpri（初期タスク優先度）,su_stadr（タスク開始アドレス）のメンバ変数で構成される構造体です。

```
typedef struct  initsk  {
        ID      su_tskid;               /* task id                      */
        TPRI    su_itskpri;             /* initial task priority        */
        TASKP   su_stadr;               /* task start address           */
} INITSK;
```

初期登録タスク情報は、初期登録タスク数分、_0Hinitskテーブルに定義します。初期登録タスク数が0の場合は、定義する必要はありません。

初期登録タスク情報は、タスクID、初期タスク優先度、タスク開始アドレスの順に定義します。

図３－１０に初期登録タスク情報の定義例を示します。

```
/*-------------------- define initial task information ---------------------*/
#if     hi_initsknum                    /* case of hi_initsknum > 0         */
extern  TASK    task1();                /* task1                            */ ･･･ (a)
extern  TASK    task2();                /* task2                            */

const   INITSK  _0Hinitsk[hi_initsknum] = {
        /* define task1                 */
            (ID)1,                      /* task id              = 1         */ ･･･ (b)
            (TPRI)1,                    /* initial task priority = 1        */ ･･･ (c)
            (TASKP)task1,               /* task start address   = task1     */ ･･･ (d)

        /* define task2                 */
            (ID)2,                      /* task id              = 2         */
            (TPRI)3,                    /* initial task priority = 3        */
            (TASKP)task2,               /* task start address   = task2     */
        };
```

図３－１０　初期登録タスク情報の定義例

(a) タスク開始アドレスをシンボル参照します。

(b) タスクIDを定義します。

(c) 初期タスク優先度を定義します。

(d) タスク開始アドレスを定義します。

３．３．７　同期／通信管理情報の定義

次の同期／通信管理情報を定義します。

(1)　最大イベントフラグID
ユーザシステムで使用するイベントフラグIDの最大値を定義します。
最大イベントフラグIDは、hi_maxflgidに0から1023の値を定義します。
例えば、3を定義した場合は、1から3までのイベントフラグIDを使用することができます。
0を定義した場合は、イベントフラグは未登録となり、イベントフラグを使用することはできません。
なお、定義する値が小さいほど、カーネル作業領域は小さくなります。

(2)　最大セマフォID
ユーザシステムで使用するセマフォIDの最大値を定義します。
最大セマフォIDは、hi_maxsemidに0から1023の値を定義します。
例えば、3を定義した場合は、1から3までのセマフォIDを使用することができます。
0を定義した場合は、セマフォは未登録となり、セマフォを使用することはできません。
なお、定義する値が小さいほど、カーネル作業領域は小さくなります。

(3)　最大メイルボックスID
ユーザシステムで使用するメイルボックスIDの最大値を定義します。
最大メイルボックスIDは、hi_maxmbxidに0から1023の値を定義します。
例えば、3を定義した場合は、1から3までのメイルボックスIDを使用することができます。
0を定義した場合は、メイルボックスは未登録となり、メイルボックスを使用することはできません。
なお、定義する値が小さいほど、カーネル作業領域は小さくなります。

図３－１１に同期／通信管理情報の定義例を示します。

```
/************************************************************************/
/*                                                                      */
/*          Definition synchronization and communication object information */
/*                                                                      */
/************************************************************************/
#define hi_maxflgid     3               /* max event flag id            */
                                        /*      range : [0...1023]      */
#define hi_maxsemid     3               /* max semaphore id             */
                                        /*      range : [0...1023]      */
#define hi_maxmbxid     3               /* max mail box id              */
                                        /*      range : [0...1023]      */
```

図３－１１　同期／通信管理情報の定義例

３．３．８　メモリプール管理情報の定義

次のメモリプール管理情報を定義します。

(1)　最大メモリプールID

ユーザシステムで使用するメモリプールIDの最大値を定義します。
最大メモリプールIDは、hi_maxmplidに0から1023の値を定義します。
例えば、2を定義した場合は、1と2のメモリプールIDを使用することができます。
0を定義した場合は、メモリプールは未登録となり、メモリプールを使用することはできません。
なお、定義する値が小さいほど、カーネル作業領域は小さくなります。

(2)　メモリプール情報

メモリプール情報は、INIMPL型に宣言された_0Hinimplテーブルに定義します。
INIMPL型は、次のように宣言してあり、su_mpladr（メモリプール領域先頭アドレス）、su_blksz（メモリブロックサイズ）、su_blkcnt（メモリブロック数）のメンバ変数で構成される構造体です。

```
typedef struct  inimpl  {
        VW      *su_mpladr;             /* top address of memory pool       */
        UW      su_blksz;               /* block size                       */
        UW      su_blkcnt;              /* block count                      */
} INIMPL;
```

メモリプール情報は、メモリプールIDの1から最大メモリプールIDまで順に、_0Hinimplテーブルに定義します。最大メモリプールIDが0の場合は、定義する必要はありません。
メモリプール情報は、メモリプール領域の先頭アドレス、メモリブロックサイズ、メモリブロック数の順に定義します。

図３－１２にメモリプール管理情報の定義例を示します。

```
/*************************************************************************/
/*                                                                       */
/*          Definition memory pool information                           */
/*                                                                       */
/*************************************************************************/
#define hi_maxmplid     2               /* max memory pool id            */ ･･･ (a)
                                        /*      range : [0...1023]       */
/*************************************************************************/
/*                                                                       */
/*          Definition memory pool information to setup table (INIMPL)   */
/*                                                                       */
/*************************************************************************/
#if     hi_maxmplid                     /* case of hi_maxmplid > 0       */
#include        "himempol.h"            /* include "himempol.h"          */ ･･･ (b)
extern  VW      hi_mempol1[];           /* memory pool of mplid = 1      */ ･･･ (c)
extern  VW      hi_mempol2[];           /* memory pool of mplid = 2      */

const   INIMPL  _0Hinimpl[hi_maxmplid] = {
        /* define memory pool of mplid = 1  */
            (VW *)hi_mempol1,           /* top address = hi_mempol1      */ ･･･ (d)
            (UW)hi_memblksz1,           /* block size  = hi_memblksz1    */ ･･･ (e)
            (UW)hi_memblkcnt1,          /* block count = hi_memblkcnt1   */ ･･･ (f)

        /* define memory pool of mplid = 2  */
            (VW *)hi_mempol2,           /* top address = hi_mempol2      */
            (UW)hi_memblksz2,           /* block size  = hi_memblksz2    */
            (UW)hi_memblkcnt2,          /* block count = hi_memblkcnt2   */
        };
```

図３－１２　メモリプール管理情報の定義例

(a)　最大メモリプールIDを定義します。
(b)　メモリプール領域定義ヘッダを読み込んで、メモリブロックサイズ、メモリブロック数をシンボル参照します。
(c)　メモリプール領域の先頭アドレスをシンボル参照します。
(d)　メモリプール領域の先頭アドレスを定義します。
(e)　メモリブロックサイズを定義します。
(f)　メモリブロック数を定義します。

メモリブロックサイズ、メモリブロック数は、メモリプール領域定義ヘッダ（himempol.h）で、"hi_memblksz<n>"、"hi_memblkcnt<n>"のシンボルにそれぞれ定義します。メモリプール領域は、メモリプール領域定義ソース（himempol.c）で、VW型の配列を"hi_mempol<n>"のシンボルで定義します。（<n>はメモリプールID番号を表わします）

なお、メモリプール領域定義プログラム（himempol.h, himempol.c）についての詳細は、「３．４．４　メモリプール領域定義プログラムの作成」を参照してください。

３．３．９　拡張SVC情報の定義

次の拡張SVC情報を定義します。

(1)　拡張SVCの最大機能コード

ユーザシステムで使用する拡張SVCの機能コードの最大値を定義します。

最大機能コードは、hi_maxsvccdに0から255の値を定義します。

例えば、2を定義した場合は、1と2の機能コードを使用することができます。

0を定義した場合は、拡張SVCは未登録となり、拡張SVCを使用することはできません。

(2)　拡張SVCハンドラの開始アドレス

拡張SVCハンドラの開始アドレスは、INISVC型に宣言された_0Hinisvcテーブルに定義します。

INISVC型は、次のように宣言してあり、su_svchdr（拡張SVCハンドラ開始アドレス）のメンバ変数で構成される構造体です。

```
typedef struct  inisvc  {
        SVCHDRP su_svchdr;              /* start address of SVC handler     */
} INISVC;
```

拡張SVCハンドラの開始アドレスは、機能コードの1から最大機能コードまで順に、_0Hinisvcテーブルに定義します。最大機能コードが0の場合は、定義する必要はありません。

(3)　拡張SVCハンドラの最大スタックサイズ

定義する拡張SVCハンドラの中で、最もスタックを使用する拡張SVCハンドラのスタックサイズを定義します。

拡張SVCハンドラの最大スタックサイズは、hi_svcstkszに4の倍数の値を定義します。最大機能コードが0の場合は、定義する必要はありません。

拡張SVCハンドラのスタックサイズは、以下の計算式で算出します。

拡張SVCハンドラスタックサイズ = <ハンドラが独自に使用する最大スタックサイズ> + <例外>

<例外>は、例外が発生する場合に加算します。通常は、例外発生時にカーネルの例外サービスルーチンがSPC,SSRレジスタを退避するための8バイトと例外処理ルーチンが独自に使用するスタックサイズの和となります。なお、例外がネストする場合は、そのネスト分も考慮して加算してください。

これらの定義によって、拡張SVCハンドラが使用するスタック領域は、カーネルの作業領域に加算して確保されます。

図３－１３に拡張SVC情報の定義例を示します。

```
/*****************************************************************************/
/*                                                                           */
/*          Definition extended SVC information                              */
/*                                                                           */
/*****************************************************************************/
#define hi_maxsvccd     2               /* max function code of extended SVC*/   ･･･(a)
                                        /*      range : [0...255]           */
/*****************************************************************************/
/*                                                                           */
/*          Definition extended SVC information to setup table (INISVC)      */
/*                                                                           */
/*****************************************************************************/
#if     hi_maxsvccd                     /* case of hi_maxsvccd > 0          */
extern  SVCHDR  hi_svchdr1();           /* SVCHDR of fncd = 1               */   ･･･(b)
extern  SVCHDR  hi_svchdr2();           /* SVCHDR of fncd = 2               */

const   INISVC  _0Hinisvc[hi_maxsvccd] = {
            (SVCHDRP)hi_svchdr1,        /* define SVCHDR of fncd = 1        */   ･･･(c)
            (SVCHDRP)hi_svchdr2,        /* define SVCHDR of fncd = 2        */
        };
/*****************************************************************************/
/*                                                                           */
/*          Definition stack size of extended SVC handler (multiple of 4)    */
/*                                                                           */
/*          Usage   :   #define hi_svcstksz <maxsvcusesz> + <excusesz>       */
/*                                                                           */
/*              <maxsvcusesz>   :   max size of extended SVC handler use     */
/*              <excusesz>      :   size of exception use                    */
/*                                                                           */
/*****************************************************************************/
#define hi_svcstksz     (32 + 0)        /* define stack size of SVCHDR      */   ･･･(d)
```

図３－１３　拡張SVC情報の定義例

(a) 拡張SVCの最大機能コードを定義します。
(b) 拡張SVCハンドラの開始アドレスをシンボル参照します。
(c) 拡張SVCハンドラの開始アドレスを定義します。
(d) 拡張SVCハンドラの最大スタックサイズを定義します。

３．３．１０　システム初期化ハンドラ情報の定義

次のシステム初期化ハンドラ情報を定義します。

(1)　システム初期化ハンドラの使用／未使用

システム初期化ハンドラの使用／未使用を定義します。

システム初期化ハンドラの使用／未使用は、hi_inihdrに"USE"（使用）、または"NOTUSE"（未使用）を定義します。

(2)　システム初期化ハンドラの開始アドレス

システム初期化ハンドラの開始アドレスを定義します。

システム初期化ハンドラの開始アドレスは、INIIHDR型に宣言された_0Hiniihdrテーブルに定義します。

INIIIHDR型は、次のように宣言してあり、su_inihdr（システム初期化ハンドラ開始アドレス）のメンバ変数で構成される構造体です。

```
typedef struct  iniihdr  {
        INIHDRP su_inihdr;              /* start address of initial handler */
} INIIHDR;
```

システム初期化ハンドラを未使用にした場合は、定義する必要はありません。

(3)　システム初期化ハンドラのスタックサイズ

システム初期化ハンドラが使用するスタックのスタックサイズを定義します。

システム初期化ハンドラのスタックサイズは、hi_inistkszに4の倍数の値を定義します。システム初期化ハンドラを未使用にした場合は、定義する必要はありません。

システム初期化ハンドラのスタックサイズは、以下の計算式で算出します。

システム初期化ハンドラスタックサイズ = <ハンドラが独自に使用するスタックサイズ> + <例外>

<例外>は、例外が発生する場合に加算します。通常は、例外発生時にカーネルの例外サービスルーチンがSPC,SSRレジスタを退避するための8バイトと例外処理ルーチンが独自に使用するスタックサイズの和となります。なお、例外がネストする場合は、そのネスト分も考慮して加算してください。

これらの定義によって、システム初期化ハンドラが使用するスタック領域は、カーネルの作業領域に加算して確保されます。

図３－１４にシステム初期化ハンドラ情報の定義例を示します。

```
/************************************************************************/
/*                                                                      */
/*          Definition system initial handler use in user system        */
/*                                                                      */
/*          Usage   :   #define hi_inihdr { USE | NOTUSE }              */
/*                      USE     :   use the system initial handler      */
/*                      NOTUSE  :   not use the system initial handler  */
/*                                                                      */
/************************************************************************/
#define hi_inihdr       USE             /* system initial handler       */   ･･･(a)
/************************************************************************/
/*                                                                      */
/*          Definition system initial handler information to setup table*/
/*                                                              (INIIHDR)*/
/************************************************************************/
#if     hi_inihdr                       /* case of hi_inihdr = USE      */
extern  INIHDR  hi_sysini();            /* system initial handler       */   ･･･(b)

const   INIIHDR _OHiniihdr[1] = {
            (INIHDRP)hi_sysini          /* define system initial handler*/   ･･･(c)
        };
/************************************************************************/
/*                                                                      */
/*          Definition stack size of system initial handler (multiple of 4) */
/*                                                                      */
/*          Usage   :   #define hi_inistksz <iniusesz> + <excusesz>      */
/*                                                                      */
/*              <iniusesz>      :   size of system initial handler use  */
/*              <excusesz>      :   size of exception use               */
/*                                                                      */
/************************************************************************/
#define hi_inistksz     (4 + 0)         /* define stack size of INIHDR  */   ･･･(d)
```

図３－１４　システム初期化ハンドラ情報の定義例

(a) システム初期化ハンドラの使用／未使用を定義します。
(b) システム初期化ハンドラの開始アドレスをシンボル参照します。
(c) システム初期化ハンドラの開始アドレスを定義します。
(d) システム初期化ハンドラのスタックサイズを定義します。

３．３．１１　トレース情報の定義

トレース情報の定義では、トレース機能に必要な情報を定義します。トレース機能についての詳細は、『HI-SH77ユーザーズマニュアル』の「２．１６　トレース機能」を参照してください。

次のトレース情報を定義します。

(1)　トレース機能の使用／未使用

トレース機能の使用／未使用を定義します。

トレース機能の使用／未使用は、hi_traceに"USE"（使用）、または"NOTUSE"（未使用）を定義します。

(2)　トレースバッファ情報

トレースバッファ情報は、INITRC型に宣言された_0Hinitrcテーブルに定義します。

INITRC型は、次のように宣言してあり、su_trbtop（トレースバッファ領域先頭アドレス）、su_trbcnt（トレース情報取得数）のメンバ変数で構成される構造体です。

```
typedef struct  initrc  {
        VW      *su_trbtop;             /* top address of trace buffer      */
        UW      su_trbcnt;              /* number of trace entry            */
} INITRC;
```

トレースバッファ情報は、_0Hinitrcテーブルに定義します。トレース機能を未使用にした場合は、定義する必要はありません。

トレースバッファ情報は、トレースバッファ領域先頭アドレス、トレース情報取得数の順に定義します。

図３－１５にトレース情報の定義例を示します。

```
/*****************************************************************************/
/*                                                                           */
/*          Definition trace function use in user system                     */
/*                                                                           */
/*          Usage   :   #define hi_trace { USE | NOTUSE }                    */
/*                      USE     :   use the trace function                   */
/*                      NOTUSE  :   not use the trace function               */
/*                                                                           */
/*****************************************************************************/
#define hi_trace        USE             /* trace function                */  ･･･ (a)

/*****************************************************************************/
/*                                                                           */
/*          Definition trace information to setup table (INITRC)             */
/*                                                                           */
/*****************************************************************************/
#if     hi_trace                        /* case of hi_trace = USE        */
#include        "hitrcbuf.h"            /* include "hitrcbuf.h"          */  ･･･ (b)
extern  VW      hi_trcbuf[];            /* trace buffer                  */  ･･･ (c)

const   INITRC  _0Hinitrc[1] = {
        /* define trace information     */
            (VW *)hi_trcbuf,            /* top address     = hi_trcbuf   */  ･･･ (d)
            (UW)hi_trcentnum            /* number of entry = hi_trcentnum*/  ･･･ (e)
        };
```

図３－１５　トレース情報の定義例

(a) トレース機能の使用／未使用を定義します。
(b) トレースバッファ領域定義ヘッダを読み込んで、トレース情報取得数をシンボル参照します。
(c) トレースバッファ領域の先頭アドレスをシンボル参照します。
(d) トレースバッファ領域の先頭アドレスを定義します。
(e) トレース情報取得数（トレースエントリ数）を定義します。

トレース情報取得数は、トレースバッファ領域定義ヘッダ（hitrcbuf.h）で、"hi_trcentnum"、のシンボルに定義します。トレースバッファ領域は、トレースバッファ領域定義ソース（hitrcbuf.c）で、VW型の配列を"hi_trcbuf"のシンボルで定義します。

なお、トレースバッファ領域定義プログラム（hitrcbuf.h, hitrcbuf.c）についての詳細は、
「３．４．５　トレースバッファ領域定義プログラムの作成」を参照してください。

３．４　メモリ領域定義プログラムの作成

３．４．１　メモリ領域定義プログラムの定義内容

メモリ領域定義プログラムは、各種メモリ領域（RAM）のメモリサイズとメモリ領域の定義を行なうプログラムです。

表３－５にメモリ領域定義プログラムの定義内容を示します。

表３－５　メモリ領域定義プログラムの定義内容

項番	メモリ領域定義プログラム	定　義　内　容	ファイル名	セクション名
1	タスク用スタック領域 定義プログラム	タスク用スタックサイズ	hitskstk.h	hitskstk
		タスク用スタック領域	hitskstk.c	
2	メモリプール領域 定義プログラム	メモリブロックサイズ メモリブロック数 メモリプールサイズ	himempol.h	himempol
		メモリプール領域	himempol.c	
3	トレースバッファ領域 定義プログラム	トレース情報取得数 トレースバッファサイズ	hitrcbuf.h	hitrcbuf
		トレースバッファ領域	hitrcbuf.c	
4	割込みハンドラ用 スタック領域定義プログラム	割込みハンドラ用スタックサイズ	hiintstk.h	hiintstk
		割込みハンドラ用スタック領域	hiintstk.c	

３．４．２　メモリ領域定義プログラムの構成

メモリ領域定義プログラムは、次の4つのC言語記述したプログラムで構成しています。

(a)　タスク用スタック領域定義プログラム（hitskstk.h, hitskstk.c）
(b)　メモリプール領域定義プログラム（himempol.h, himempol.c）
(c)　トレースバッファ領域定義プログラム（hitrcbuf.h, hitrcbuf.c）
(d)　割込みハンドラ用スタック領域定義プログラム（hiintstk.h, hiintstk.c）

図３－１６にメモリ領域定義プログラムの構成を示します。

メモリ領域定義プログラムで定義した内容は、セットアップテーブルやユーザプログラムから参照されます。

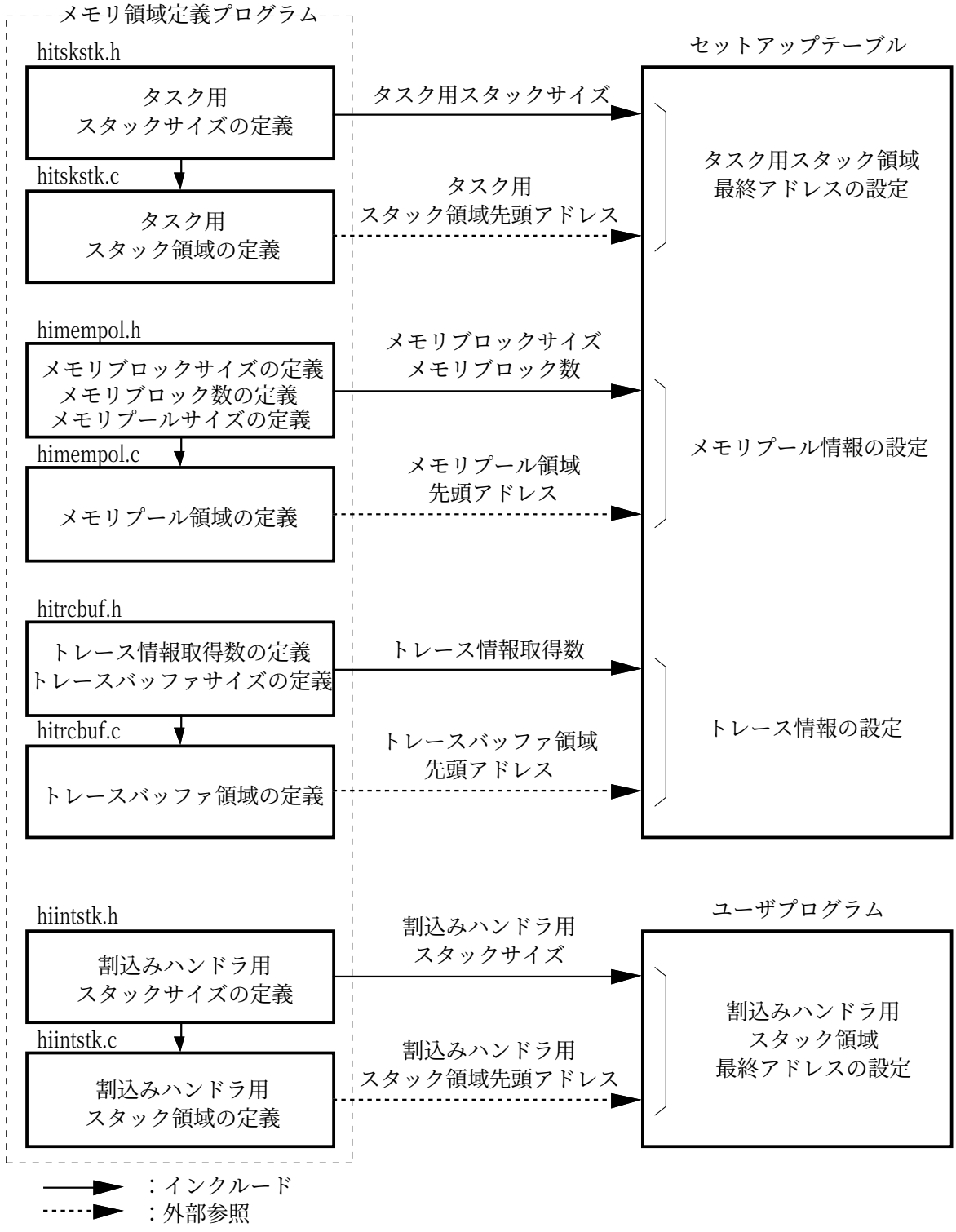


図３－１６　メモリ領域定義プログラムの構成

３．４．３　タスク用スタック領域定義プログラムの作成

タスク用スタック領域定義プログラムは、タスクが使用するスタック領域を定義するプログラムです。

タスク用スタック領域定義プログラムは、サンプルプログラム（hitskstk.h, hitskstk.c）を参考にして作成してください。セットアップテーブルで最大タスクIDに0（タスクを未登録）を定義した場合は、作成する必要はありません。

(1)　タスク用スタックサイズの定義

タスク用スタック領域定義ヘッダ（hitskstk.h）に、タスク用スタックサイズを定義します。

図３－１７にタスク用スタックサイズの定義例を示します。

```
/*****************************************************************************/
/*                                                                           */
/*          Definition constant size of task stack                           */
/*                                                                           */
/*****************************************************************************/
#define KNLTUSESZ       124                 /* size of kernel use           */   ･･･(a)
/*****************************************************************************/
/*                                                                           */
/*          Definition task stack size of interrupt use                      */
/*                                                                           */
/*  Usage   :                                                                */
/*                                                                           */
/*  #define inttusesz       12 * (<hi_uppintnst> + <hi_lowintnst>)           */
/*                                                                           */
/*      <hi_uppintnst>  :   number of interrupt levels                       */
/*                          (>  kernel mask level)                           */
/*      <hi_lowintnst>  :   number of interrupt levels                       */
/*                          (<= kernel mask level)                           */
/*                                                                           */
/*****************************************************************************/
#define inttusesz       12 * (0 + 2)        /* size of interrupt use        */   ･･･(b)
/*****************************************************************************/
/*                                                                           */
/*          Definition task stack size (multiple of 4)                       */
/*                                                                           */
/*  Usage   :                                                                */
/*                                                                           */
/*  #define hi_tskstk<n>    KNLTUSESZ + inttusesz + <tskusesz> + <excusesz>  */
/*                                                                           */
/*      <n>             :   task stack number                                */
/*      <tskusesz>      :   size of task use                                 */
/*          (note : when tasks are executed as non-tasks, 28 must be added)  */
/*      <excusesz>      :   size of exception use                            */
/*                                                                           */
/*****************************************************************************/
#define hi_tskstksz1    (KNLTUSESZ + inttusesz +  52 + 0)  /* stack no. = 1 */   ･･･(c)
#define hi_tskstksz2    (KNLTUSESZ + inttusesz + 108 + 0)  /* stack no. = 2 */
#define hi_tskstksz3    (KNLTUSESZ + inttusesz + 108 + 0)  /* stack no. = 3 */
#define hi_tskstksz4    (KNLTUSESZ + inttusesz + 108 + 0)  /* stack no. = 4 */
```

図３－１７　タスク用スタックサイズの定義例

(a) カーネルが使用するサイズ（124バイト）をKNLTUSESZに定義しています。この定義は変更しないでください。

(b) 割込みハンドラが使用するサイズをinttuseszに定義します。この定義が必要である理由は、割込みハンドラが、割込み発生直後、タスク用スタック領域を12バイト消費するためです。
（スタックポインタを割込みハンドラ用スタック領域に切り換えるまでの処理で消費されます）
inttuseszは、以下の計算式で算出します。

```
inttusesz = 12 × ( <hi_uppintnst> +  <hi_lowintnst> )
```

<hi_uppintnst>は、カーネル割込みマスクレベルより高い割込みのネスト数です。<hi_lowintnst>は、カーネル割込みマスクレベル以下の割込みネスト数です。このパラメータには、セットアップテーブルで定義した数値と同じ値を定義してください。（「３．３．５　カーネル情報の定義」参照）

(c) タスク用スタックサイズをhi_tskstksz<n>に定義します。<n>は、スタック番号を表わします。
「共有スタック機能」を使用しない場合は、タスク数分（1～最大タスクID）定義してください。
「共有スタック機能」を使用する場合は、共有するタスク数分だけ定義する数が少なくなります。
hi_tskstksz<n>は、以下の計算式で算出します。

```
hi_tskstksz<n> = KNLTUSESZ + inttusesz + <tskusesz> + <excusesz>
```

<tskusesz>は、タスクが独自に使用するスタックサイズ（4の倍数）を定義します。なお、「共有スタック機能」を使用する場合は、共有するタスクの中で、最もスタック使用量の多いタスクのスタックサイズを定義してください。タスクを非タスク（割込みマスクレベル≠0）で実行するケースがある場合、28を加算してください。
<excusesz>は、例外が使用するスタックサイズ（4の倍数）を定義します。
例外が発生しない場合は、0を定義します。
例外が発生する場合は、例外発生時にカーネルの例外サービスルーチンがSPC,SSRレジスタを退避するための8バイトと例外処理ルーチンが独自に使用するスタックサイズの和となります。なお、例外がネストする場合は、そのネスト分も考慮して加算してください。

タスク用スタックサイズの算出方法についての詳細は、「付録Ａ　メモリ容量の算出」を参照してください。
なお、定義したタスク用スタックサイズは、セットアップテーブルから参照されます。
（「３．３．６　タスク管理情報の定義」参照）

(2) タスク用スタック領域の定義
タスク用スタック領域定義ソース（hitskstk.c）に、タスク用スタック領域を定義します。
図３－１８にタスク用スタック領域の定義例を示します。

```
/************************************************************************/
/*                                                                      */
/*          Definition task stack area                                  */
/*                                                                      */
/************************************************************************/
VW  hi_tskstk1[(hi_tskstksz1) / sizeof(VW)];    /* stack no. = 1        */
VW  hi_tskstk2[(hi_tskstksz2) / sizeof(VW)];    /* stack no. = 2        */
VW  hi_tskstk3[(hi_tskstksz3) / sizeof(VW)];    /* stack no. = 3        */
VW  hi_tskstk4[(hi_tskstksz4) / sizeof(VW)];    /* stack no. = 4        */
```

図３－１８　タスク用スタック領域の定義例

タスク用スタック領域をVW型の配列で定義します。
記述方法は、"VW　hi_tskstk<n>[(hi_tskstksz<n>) / sizeof(VW)];"です。 <n>は、スタック番号を表わします。
なお、定義したタスク用スタック領域の先頭アドレスは、セットアップテーブルから参照されます。
（「３．３．６　タスク管理情報の定義」参照）

３．４．４　メモリプール領域定義プログラムの作成

メモリプール領域定義プログラムは、メモリプール領域を定義するプログラムです。

メモリプール領域定義プログラムは、サンプルプログラム（himempol.h, himempol.c）を参考にして作成してください。セットアップテーブルで最大メモリプールIDに0（メモリプールを未登録）を定義した場合は、作成する必要はありません。

(1)　メモリプールサイズの定義

メモリプール領域定義ヘッダ（himempol.h）に、メモリプールサイズを定義します。

図３－１９にメモリプールサイズの定義例を示します。

```
/************************************************************************/
/*                                                                      */
/*          Definition constant size of memory pool                     */
/*                                                                      */
/************************************************************************/
#define MBKMNGSZ        4           /* size of memory block management area */   ･･･(a)

/************************************************************************/
/*                                                                      */
/*          Definition memory pool size                                 */
/*                                                                      */
/************************************************************************/
/*-------------------------------- mplid = 1 --------------------------------*/
#define hi_memblksz1    0x00000020      /* memory block size (multiple of 4)*/   ･･･(b)
#define hi_memblkcnt1   0x00000004      /* memory block count               */   ･･･(c)
                                        /* memory pool size                 */
#define hi_mempolsz1    ((MBKMNGSZ + hi_memblksz1) * hi_memblkcnt1)              ･･･(d)

/*-------------------------------- mplid = 2 --------------------------------*/
#define hi_memblksz2    0x00000010      /* memory block size (multiple of 4)*/
#define hi_memblkcnt2   0x00000008      /* memory block count               */
                                        /* memory pool size                 */
#define hi_mempolsz2    ((MBKMNGSZ + hi_memblksz2) * hi_memblkcnt2)
```

図３－１９　メモリプールサイズの定義例

(a)　メモリブロック管理領域のサイズ（4バイト）をMBKMNGSZに定義しています。この定義は変更しないでください。

(b)　メモリブロックサイズ（4の倍数）をhi_memblksz<n>に定義します。<n>は、メモリプールID番号を表わします。

(c)　メモリブロック数（1以上）をhi_memblkcnt<n>に定義します。

(d)　メモリブロックサイズ、メモリブロック数から必要なメモリプールサイズをhi_mempolsz<n>に定義します。

hi_mempolsz<n>は、以下の計算式で算出します。

hi_mempolsz<n> = (MBKMNGSZ + hi_memblksz<n>) × hi_memblkcnt<n>

メモリプールサイズの算出方法についての詳細は、「付録Ａ　メモリ容量の算出」を参照してください。

なお、定義したメモリブロックサイズ、メモリブロック数は、セットアップテーブルから参照されます。（「３．３．８　メモリプール管理情報の定義」参照）

(2) メモリプール領域の定義

メモリプール領域定義ソース（himempol.c）に、メモリプール領域を定義します。

図３－２０にメモリプール領域の定義例を示します。

```
/*=========================================================================*/
/*                                                                         */
/*          Definition memory pool area                                    */
/*                                                                         */
/*=========================================================================*/
VW  hi_mempol1[(hi_mempolsz1) / sizeof(VW)];    /* mplid = 1               */
VW  hi_mempol2[(hi_mempolsz2) / sizeof(VW)];    /* mplid = 2               */
```

図３－２０　メモリプール領域の定義例

メモリプール領域をVW型の配列で定義します。

記述方法は、"VW hi_mempol<n>[(hi_mempolsz<n>) / sizeof(VW)];"です。 <n>は、メモリプールID番号を表わします。

なお、定義したメモリプール領域の先頭アドレスは、セットアップテーブルから参照されます。

（「３．３．８　メモリプール管理情報の定義」参照）

３．４．５　トレースバッファ領域定義プログラムの作成

トレースバッファ領域定義プログラムは、トレース機能で使用するトレースバッファ領域を定義するプログラムです。

トレースバッファ領域定義プログラムは、サンプルプログラム（hitrcbuf.h, hitrcbuf.c）を参考にして作成してください。セットアップテーブルでトレース機能を未使用に定義した場合は、作成する必要はありません。

(1)　トレースバッファサイズの定義

トレースバッファ領域定義ヘッダ（hitrcbuf.h）に、トレースバッファサイズを定義します。

図３－２１にトレースバッファサイズの定義例を示します。

```
/*****************************************************************************/
/*                                                                           */
/*          Definition constant size of trace buffer                         */
/*                                                                           */
/*****************************************************************************/
#define TRCMNGSZ        0x10            /* size of trace management area    */   ･･･(a)
#define TRCENTSZ        0x24            /* size of trace entry area         */   ･･･(b)

/*****************************************************************************/
/*                                                                           */
/*          Definition trace information                                     */
/*                                                                           */
/*****************************************************************************/
#define hi_trcentnum    32              /* number of trace entry            */   ･･･(c)
                                        /*      range : [1...]              */
                                        /* trace buffer size                */
#define hi_trcbufsz     (TRCMNGSZ + TRCENTSZ * hi_trcentnum)                     ･･･(d)
```

図３－２１　トレースバッファサイズの定義例

(a)　トレースバッファ管理領域のサイズ（16バイト）をTRCMNGSZに定義しています。この定義は変更しないでください。

(b)　トレース情報1個あたりのバッファサイズ（36バイト）をTRCENTSZに定義しています。この定義は変更しないでください。

(c)　トレース情報取得数（1以上）をhi_trcentnumに定義します。

(d)　トレース情報取得数から必要なトレースバッファサイズをhi_trcbufszに定義します。

hi_trcbufszは、以下の計算式で算出します。

hi_trcbufsz = TRCMNGSZ + TRCENTSZ × hi_trcentnum

トレースバッファサイズの算出方法についての詳細は、「付録Ａ　メモリ容量の算出」を参照してください。

なお、定義したトレース情報取得数は、セットアップテーブルから参照されます。

（「３．３．１１　トレース情報の定義」参照）

(2) トレースバッファ領域の定義

トレースバッファ領域定義ソース（hitrcbuf.c）に、トレースバッファ領域を定義します。

図３－２２にトレースバッファ領域の定義例を示します。

```
/*****************************************************************************/
/*                                                                           */
/*          Definition trace buffer area                                     */
/*                                                                           */
/*****************************************************************************/
VW      hi_trcbuf[(hi_trcbufsz) / sizeof(VW)];  /* trace buffer area         */
```

図３－２２　トレースバッファ領域の定義例

トレースバッファ領域をVW型の配列で定義します。

記述方法は、"VW　hi_trcbuf[(hi_trcbufsz) / sizeof(VW)];"です。

なお、定義したトレースバッファ領域の先頭アドレスは、セットアップテーブルから参照されます。

（「３．３．１１　トレース情報の定義」参照）

３．４．６　割込みハンドラ用スタック領域定義プログラムの作成

割込みハンドラ用スタック領域定義プログラムは、割込みハンドラが使用するスタック領域を定義するプログラムです。

割込みハンドラ用スタック領域定義プログラムは、サンプルプログラム（hiintstk.h, hiintstk.c）を参考にして作成してください。

(1)　割込みハンドラ用スタックサイズの定義

割込みハンドラ用スタック領域定義ヘッダ（hiintstk.h）に、割込みハンドラ用スタックサイズを定義します。

図３－２３に割込みハンドラ用スタックサイズの定義例を示します。

```
/************************************************************************/
/*                                                                      */
/*          Definition constant size of interrupt handler stack         */
/*                                                                      */
/************************************************************************/
#define KNLUUSESZ       16          /* size of kernel use                */ ・・・ (a)
                                    /*(interrupt level >  kernel mask level)*/
#define KNLLUSESZ       148         /* size of kernel use                */
                                    /*(interrupt level <= kernel mask level)*/
/************************************************************************/
/*                                                                      */
/*          Definition stack size of interrupt handler (multiple of 4)  */
/*                                                                      */
/*  Usage   :                                                           */
/*  case of interrupt level =  not use                                  */
/*    #define hi_intstk<n>  0                                           */
/*                                                                      */
/*  case of interrupt level >  kernel mask level                        */
/*    #define hi_intstk<n>  KNLUUSESZ + <othusesz> + <curusesz> + <excusesz>*/
/*                                                                      */
/*  case of interrupt level <= kernel mask level                        */
/*    #define hi_intstk<n>  KNLLUSESZ + <othusesz> + <curusesz> + <excusesz>*/
/*                                                                      */
/*      <n>             :   interrupt level number                      */
/*      <othusesz>      :   size of other interrupt use                 */
/*                      =   12 * (<uppintnst> + <lowintnst>)            */
/*          <uppintnst> :   number of interrupt levels                  */
/*                          (>  kernel mask level, > current interrupt level)*/
/*          <lowintnst> :   number of interrupt levels                  */
/*                          (<= kernel mask level, > current interrupt level)*/
/*      <curusesz>      :   size of current interrupt handler use       */
/*      <excusesz>      :   size of exception use                       */
/*                                                                      */
/************************************************************************/
#define hi_intstksz1    0                                   /* level =  1 */
#define hi_intstksz2    0                                   /* level =  2 */
#define hi_intstksz3    0                                   /* level =  3 */
#define hi_intstksz4    0                                   /* level =  4 */
#define hi_intstksz5    0                                   /* level =  5 */
#define hi_intstksz6    0                                   /* level =  6 */
#define hi_intstksz7    0                                   /* level =  7 */
#define hi_intstksz8    0                                   /* level =  8 */
#define hi_intstksz9    0                                   /* level =  9 */
#define hi_intstksz10   0                                   /* level = 10 */
#define hi_intstksz11   (KNLLUSESZ + 12 * (0 + 1) + 44 + 0) /* level = 11 */  ・・・ (b)
#define hi_intstksz12   0                                   /* level = 12 */
#define hi_intstksz13   (KNLLUSESZ + 12 * (0 + 0) + 28 + 0) /* level = 13 */
#define hi_intstksz14   0                                   /* level = 14 */
#define hi_intstksz15   0                                   /* level = 15 */
```

図３－２３　割込みハンドラ用スタックサイズの定義例

(a) カーネル割込みマスクレベルより高い割込みの場合のカーネルが使用するサイズ（16バイト）をKNLUUSESZに定義しています。また、カーネル割込みマスクレベル以下の割込みレベルの場合のカーネルが使用するサイズ（148バイト）をKNLLUSESZに定義しています。これらの定義は変更しないでください。

(b) 割込みハンドラ用スタックサイズをhi_intstksz<n>に定義します。<n>は、割込みレベル番号を表わします。

hi_intstksz<n>は、以下の計算式で算出します。

・使用しない割込みレベルの場合

```
hi_intstksz<n> = 0
```

・カーネル割込みマスクレベルより高い割込みレベルの場合

```
hi_intstksz<n> = KNLUUSESZ + <othusesz> + <curusesz> + <excusesz>
```

・カーネル割込みマスクレベル以下の割込みレベルの場合

```
hi_intstksz<n> = KNLLUSESZ + <othusesz> + <curusesz> + <excusesz>
```

<othusesz>は、他の割込みレベルのハンドラが使用するサイズを定義します。この定義が必要である理由は、より高いレベルの割込みハンドラが、割込み発生直後に、それまで実行していた割込みハンドラのスタック領域を12バイト消費するためです。（スタックポインタを新しい割込みハンドラ用スタック領域に切り換えるまでの処理で消費されます）

<othusesz>は、以下の計算式で算出します。

```
<othusesz> = 12 × ( <uppintnst> + <lowintnst> )
```

<uppintnst>は、カーネル割込みマスクレベルより高く、かつ自割込みレベルより高い割込みのネスト数です。<lowintnst>は、カーネル割込みマスクレベル以下、かつ自割込みレベルより高い割込みのネスト数です。

<curusesz>は、自割込みレベルのハンドラが独自に使用するスタックサイズ（4の倍数）を定義します。なお、複数の割込みハンドラがある場合は、その中で最もスタック使用量の多い割込みハンドラのスタックサイズを定義してください。（同レベルの割込みハンドラは、スタック領域を共有することができます）

<excusesz>は、例外が使用するスタックサイズ（4の倍数）を定義します。

例外が発生しない場合は、0を定義します。

例外が発生する場合は、例外発生時にカーネルの例外サービスルーチンがSPC,SSRレジスタを退避するための8バイトと例外処理ルーチンが独自に使用するスタックサイズの和となります。なお、例外がネストする場合は、そのネスト分も考慮して加算してください。

割込みハンドラ用スタックサイズの算出方法についての詳細は、「付録Ａ　メモリ容量の算出」を参照してください。

なお、定義した割込みハンドラ用スタックサイズは、割込みハンドラプログラムから参照します。詳細は、『HI-SH77ユーザーズマニュアル』「４．４　C言語による割込みハンドラの記述方法」を参照してください。

(2) 割込みハンドラ用スタック領域の定義

割込みハンドラ用スタック領域定義ソース（hiintstk.c）に、割込みハンドラが使用するスタック領域を定義します。

図３－２４に割込みハンドラ用スタック領域の定義例を示します。

```
/************************************************************************/
/*                                                                      */
/*          Definition stack area of interrupt handler                  */
/*                                                                      */
/************************************************************************/
#if hi_intstksz1                                /* case of hi_intstksz1 > 0 */
VW  hi_intstk1[(hi_intstksz1) / sizeof(VW)];    /* interrupt level =  1     */
#endif
#if hi_intstksz2                                /* case of hi_intstksz2 > 0 */
VW  hi_intstk2[(hi_intstksz2) / sizeof(VW)];    /* interrupt level =  2     */
#endif
#if hi_intstksz3                                /* case of hi_intstksz3 > 0 */
VW  hi_intstk3[(hi_intstksz3) / sizeof(VW)];    /* interrupt level =  3     */
#endif
#if hi_intstksz4                                /* case of hi_intstksz4 > 0 */
VW  hi_intstk4[(hi_intstksz4) / sizeof(VW)];    /* interrupt level =  4     */
#endif
#if hi_intstksz5                                /* case of hi_intstksz5 > 0 */
VW  hi_intstk5[(hi_intstksz5) / sizeof(VW)];    /* interrupt level =  5     */
#endif
#if hi_intstksz6                                /* case of hi_intstksz6 > 0 */
VW  hi_intstk6[(hi_intstksz6) / sizeof(VW)];    /* interrupt level =  6     */
#endif
#if hi_intstksz7                                /* case of hi_intstksz7 > 0 */
VW  hi_intstk7[(hi_intstksz7) / sizeof(VW)];    /* interrupt level =  7     */
#endif
#if hi_intstksz8                                /* case of hi_intstksz8 > 0 */
VW  hi_intstk8[(hi_intstksz8) / sizeof(VW)];    /* interrupt level =  8     */
#endif
#if hi_intstksz9                                /* case of hi_intstksz9 > 0 */
VW  hi_intstk9[(hi_intstksz9) / sizeof(VW)];    /* interrupt level =  9     */
#endif
#if hi_intstksz10                               /* case of hi_intstksz10 > 0*/
VW  hi_intstk10[(hi_intstksz10) / sizeof(VW)];  /* interrupt level = 10     */
#endif
#if hi_intstksz11                               /* case of hi_intstksz11 > 0*/
VW  hi_intstk11[(hi_intstksz11) / sizeof(VW)];  /* interrupt level = 11     */
#endif
#if hi_intstksz12                               /* case of hi_intstksz12 > 0*/
VW  hi_intstk12[(hi_intstksz12) / sizeof(VW)];  /* interrupt level = 12     */
#endif
#if hi_intstksz13                               /* case of hi_intstksz13 > 0*/
VW  hi_intstk13[(hi_intstksz13) / sizeof(VW)];  /* interrupt level = 13     */
#endif
#if hi_intstksz14                               /* case of hi_intstksz14 > 0*/
VW  hi_intstk14[(hi_intstksz14) / sizeof(VW)];  /* interrupt level = 14     */
#endif
#if hi_intstksz15                               /* case of hi_intstksz15 > 0*/
VW  hi_intstk15[(hi_intstksz15) / sizeof(VW)];  /* interrupt level = 15     */
#endif
```

図３－２４　割込みハンドラ用スタック領域の定義例

割込みハンドラ用スタック領域をVW型の配列で定義します。この定義例は、割込みハンドラ用スタック領域定義ヘッダ（hiintstk.h）で、スタックサイズに0以外の値を定義した割込みレベルのみ、そのスタック領域を確保します。

なお、定義した割込みハンドラ用スタック領域の先頭アドレスは、割込みハンドラプログラムから参照します。詳細は、『HI-SH77ユーザーズマニュアル』「４．４　C言語による割込みハンドラの記述方法」を参照してください。

３．５　タイマドライバの登録

ＨＩ－ＳＨ７７は、ハードウェアからの一定周期の割込み（タイマ割込み）を利用して、時間管理を行ないます。したがって、時間管理機能（wai_tsk, set_tim,iset_tim, get_tim,iget_tim, sys_clkシステムコール）を使用する場合は、タイマドライバを作成、登録する必要があります。

提供しているタイマドライバのサンプルプログラム（hitmrdrv.h, hitmrdrv.c）は、タイマ初期化ルーチンとタイマ割込みハンドラで構成しています。
以下にサンプルのタイマドライバの登録方法を説明します。

(1)　タイマ初期化ルーチンの登録
タイマ初期化ルーチン（hi_tmrini）を、システム初期化ハンドラ（hi_sysini）から呼び出します。

(2)　タイマ割込みハンドラの登録
タイマ割込みハンドラ（hi_tmrhdr）を、ベクタテーブルに登録します。
なお、タイマ割込みハンドラは、sys_clkシステムコールを発行しています。これによって、タイムアウト時間、システムクロック（年月日時刻データ）を更新することができます。

(3)　sys_clkシステムコールの登録
トラップベクタテーブルにsys_clkシステムコール処理の先頭アドレス（シンボル名 : _0Hsys_clk）を登録します。詳細は「３．２．２　トラップベクタテーブルの作成」を参照してください。

なお、タイマドライバのサンプルプログラムの詳細は、『HI-SH77ユーザーズマニュアル』の「付録Ｂ　タイマドライバの例題」を参照してください。

３．６　実行形式プログラムの作成

３．６．１　プログラムのコンパイルとアセンブル

ユーザプログラム、セットアップテーブル、およびメモリ領域定義プログラムをコンパイル、またはアセンブルしてオブジェクトモジュールを作成します。

(1) C言語プログラムのコンパイル
C言語プログラムのコンパイルは、以下のコマンドを入力します。なお、

shc△-cpu=sh3[△-endian=<bigまたはlittle>][△-debug]△<C言語ソースプログラムのファイル名>
[△-section=<セクション名>](RET)

-cpu オプション	：	SH3のオブジェクトを生成します
-endian オプション	：	エンディアンを指定します。省略した場合はビッグエンディアンになります。 big ：ビッグエンディアン little：リトルエンディアン
-debug オプション	：	デバッグ情報を出力します
-section オプション	：	セクション名を次の形式で指定します -section=p=P,c=C,d=D,b=B P：プログラム領域セクション名 C：定数領域セクション名 D：初期化定数領域セクション名 B：未初期化定数領域セクション名
上記以外のオプション	：	必要に応じて指定してください オプションの詳細は『SHシリーズ Cコンパイラ ユーザーズマニュアル』を参照してください

(2) アセンブリ言語プログラムのアセンブル
アセンブリ言語プログラムのアセンブルは、以下のコマンドを入力します。

asmsh△<アセンブリ言語ソースプログラムのファイル名>△-cpu=sh3[△-endian=<bigまたはlittle>]
[△-debug](RET)

-cpu オプション	：	SH3のオブジェクトを生成します
-endian オプション	：	エンディアンを指定します。省略した場合はビッグエンディアンになります。 big ：ビッグエンディアン little：リトルエンディアン
-debug オプション	：	デバッグ情報を出力します
上記以外のオプション	：	必要に応じて指定してください オプションの詳細は『SHシリーズ クロスアセンブラ ユーザーズマニュアル』を参照してください

(3) 提供ファイルのコンパイル

提供ファイル（セットアップテーブル，メモリ領域定義プログラム、ユーザプログラム（サンプル））は、表３－６に示すセクション名を指定して、コンパイルしてください。

表３－６　提供ファイルのセクション名

項番	ファイル名	内　　容	セクション名（-sectionへの指定内容）
1	hisuptbl.c	セットアップテーブル	-section=c=hisuptbl,b=hiknlwrk
2	hivcttbl.c	ベクタテーブル	-section=c=hivcttbl
3	hitrptbl.c	トラップベクタテーブル	-section=c=hitrptbl
4	himcuini.c	MCU初期化ルーチン	-section=p=himcuini
5	hisysini.c	システム初期化ハンドラ	-section=p=hisysini
6	hisysdwn.c	システム異常終了処理 ルーチン	-section=p=hisysdwn
7	hiintdwn.c	未定義割込み異常処理ルーチン	-section=p=hiintdwn
8	hitrpdwn.c	未定義トラップ異常処理ルーチン	-section=p=hitrpdwn
9	hitskstk.c	タスク用スタック領域定義	-section=b=hitskstk
10	hiintstk.c	割込みハンドラ用スタック領域定義	-section=b=hiintstk
11	himempol.c	メモリプール領域定義	-section=b=himempol
12	hitrcbuf.c	トレースバッファ領域定義	-section=b=hitrcbuf
13	hicnsdrv.c	コンソールドライバ	-section=p=hicnsdrv,c=hicnscst,b=hicnswrk
14	hitmrdrv.c	タイマドライバ	-section=p=hitmrdrv,c=hitmrcst

３．６．２　システムの結合

ユーザシステムに必要となるオブジェクトモジュールとカーネルライブラリをリンケージエディタで結合し、実行形式プログラム（アブソリュートロードモジュール）を作成します。

(1) リンケージエディタの実行

リンケージエディタの実行は、以下のコマンドを入力します。

lnk△-subcommand=<リンケージサブコマンドファイル名>(RET)

(2) リンケージサブコマンドファイルの作成

リンケージサブコマンドファイルは、結合したいオブジェクトモジュールやライブラリ、セクションのメモリ配置などを指定するファイルです。

リンケージサブコマンドファイルは、提供しているシステム構築用リンケージサブコマンドファイル（himake.sub）を参考にして作成してください。

図３－２５にシステム構築用リンケージサブコマンドファイル（himake.sub）を示します。

```
;*************************************************************************;
;*                              object file                              *;
;*************************************************************************;
input   hivcttbl.obj,&                  ;* vector table                   *;   ・・・(a)
        hitrptbl.obj,&                  ;* trap vector table              *;
        himcuini.obj,&                  ;* MCU initialize routine         *;
        hiintdwn.obj,&                  ;* interrupt down routine         *;
        hitrpdwn.obj,&                  ;* trap down routine              *;
        hisysdwn.obj,&                  ;* system down routine            *;
        hisuptbl.obj,&                  ;* setup table                    *;
        hisysini.obj,&                  ;* system initial handler         *;
    <ユーザプログラムのオブジェクトモジュールのファイル名>,&                    ・・・(b)
        hitmrdrv.obj,&                  ;* timer driver                  *;
        hicnsdrv.obj,&                  ;* console driver                *;
        himempol.obj,&                  ;* memory pool area              *;
        hitskstk.obj,&                  ;* task stack area               *;
        hiintstk.obj,&                  ;* interrupt stack area          *;
        hitrcbuf.obj,&                  ;* trace buffer area             *;
        rst_srv.obj,&                   ;* reset service routine         *;
        exp_srv.obj                     ;* exception service routine      *;

;*************************************************************************;
;*                              library file                             *;
;*************************************************************************;
library hiknlp.lib                      ;* kernel (parameter check)      *;   ・・・(c)
library <ユーザが用意したライブラリのファイル名>                                ・・・(d)

;*************************************************************************;
;*                              section                                  *;
;*************************************************************************;
;*------------------- area 0 (top address : 0xa0000000) -------------------*;
start   hirstsrv(0a0000000),&           ;* reset service routine         *;   ・・・(e)
        hiexpsrv,&                      ;* exception  service routine    *;
        hivcttbl,&                      ;* vector table                  *;
        hitrptbl,&                      ;* trap vector table             *;
        himcuini,&                      ;* MCU initialize routine        *;
        hiintdwn,&                      ;* interrupt down routine        *;
        hitrpdwn,&                      ;* trap down routine             *;
        hisysdwn,&                      ;* system down routine           *;
        hisuptbl,&                      ;* setup table                   *;
        hiknl,&                         ;* kernel                        *;
        hisysini,&                      ;* system initial handler        *;
    <ユーザプログラムのセクション名（ROM領域）>,&                               ・・・(f)
        hitmrdrv,&                      ;* timer driver                  *;
        hitmrcst,&                      ;* constant area of timer driver *;
        hicnsdrv,&                      ;* console driver                *;
        hicnscst(0a0000100)             ;* constant area of console driver *;

;*------------------- area 3 (top address : 0xac000000) -------------------*;
start   hiknlwrk,&                      ;* work area of kernel           *;   ・・・(g)
        hicnswrk,&                      ;* work area of console driver   *;
        himempol,&                      ;* memory pool area              *;
        hitskstk,&                      ;* task stack area               *;
        hiintstk,&                      ;* interrupt stack area          *;
    <ユーザプログラムのセクション名（RAM領域）>,&                               ・・・(h)
        hitrcbuf(0ac000000)              ;* trace buffer area            *;
```

図３－２５　システム構築用リンケージサブコマンドファイル（himake.sub）（１／２）

```
;*************************************************************************
;*                              other                                    *
;*************************************************************************
define  __0Hpon_sp(0ac080000),&          ;* SP value at power-on reset       *;  ･･･(i)
        __0Hman_sp(0ac080000)           ;* SP value at manual reset         *;
entry   __0Hrst_srv                     ;* entry address     : __0Hrst_srv  *;  ･･･(j)
output  hisystem.abs                    ;* output file       : hisystem.abs *;  ･･･(k)
print   hisystem.map                    ;* linkage map file  : hisystem.map *;  ･･･(l)
form    a                               ;* form              : absolute     *;  ･･･(m)
debug                                   ;* debug information : output       *;  ･･･(n)
exit                                    ;* exit                            *;
```

図３－２５　システム構築用リンケージサブコマンドファイル（himake.sub）（２／２）

(a) 結合するオブジェクトモジュールを指定しています。
以下のオブジェクトモジュールは、必ず指定してください。

・hivcttbl.obj（ベクタテーブル）
・hitrptbl.obj（トラップベクタテーブル）
・himcuini.obj（MCU初期化ルーチン）
・hiintdwn.obj（未定義割込み異常処理ルーチン）
・hitrpdwn.obj（未定義トラップ異常処理ルーチン）
・hisysdwn.obj（システム異常終了処理ルーチン）
・hisuptbl.obj（セットアップテーブル）

以下のオブジェクトモジュールは、必要なものだけ指定してください。

・hisysini.obj（システム初期化ハンドラ）
・hitmrdrv.obj（タイマドライバ）
・hicnsdrv.obj（コンソールドライバ）
・himempol.obj（メモリプール領域）
・hitskstk.obj（タスク用スタック領域）
・hiintstk.obj（割込みハンドラ用スタック領域）
・hitrcbuf.obj（トレースバッファ領域）

(b) 結合するオブジェクトモジュールを追加する場合に指定します。
(c) 結合するライブラリにカーネルライブラリを指定しています。
カーネルライブラリには、「パラメータチェック機能あり」のライブラリと、「パラメータチェック機能なし」のライブラリがあります。
パラメータチェック機能とは、ユーザプログラムからシステムコールを発行したときに、カーネルがシステムコールのパラメータをチェックする機能です。
ユーザプログラムをデバッグするときは、「パラメータチェック機能あり」のライブラリを指定してください。
ユーザプログラムのデバッグが完了したときは、「パラメータチェック機能なし」のライブラリを指定することをおすすめします。これによって、システムのプログラムサイズを小さくすることができ、また、カーネルの処理速度を速くすることができます。
表３－７にカーネルライブラリのファイル名を示します。

表３－７　カーネルライブラリのファイル名

項番	ファイル名	内　　容
1	hiknl.lib	カーネルライブラリ（パラメータチェック機能なし）
2	hiknlp.lib	カーネルライブラリ（パラメータチェック機能あり）

(d) 結合するライブラリを追加する場合に指定します。（例えば、Cコンパイラが提供しているライブラリを指定します）

(e) ROM領域（先頭アドレス：H'A0000000）にメモリ配置するセクションを指定しています。
以下のセクションは、必ず指定してください。

・hirstsrv（カーネルのリセットサービスルーチン）・・必ずH'A0000000に配置してください。
・hiexpsrv（カーネルの例外サービスルーチン）　・・・P1またはP2領域の先頭からH'100番地以降（H'80000100番地以降またはH'A0000100番地以降）に配置してください。
・hivcttbl（ベクタテーブル）・・・・・・・・・・・・・エリア0に配置してください。
・hitrptbl（トラップベクタテーブル）
・himcuini（MCU初期化ルーチン）・・・・・・・・・・P2領域に配置してください。
・hiintdwn（未定義割込み異常処理ルーチン）
・hitrpdwn（未定義トラップ異常処理ルーチン）
・hisysdwn（システム異常終了処理ルーチン）
・hisuptbl（セットアップテーブル）
・hiknl（カーネル本体）

以下のセクションは、必要なものだけ指定してください。

・hisysini（システム初期化ハンドラ）
・hitmrdrv（タイマドライバ）
・hitmrcst（タイマドライバの定数領域）
・hicnsdrv（コンソールドライバ）
・hicnscst（コンソールドライバの定数領域）

(f) ROM領域にメモリ配置するセクションを追加する場合に指定します。
(g) RAM領域（先頭アドレス：H'AC000000）にメモリ配置するセクションを指定しています。
以下のセクションは、必ず指定してください。

・hiknlwrk（カーネル作業領域）

以下のセクションは、必要なものだけ指定してください。

・hicnswrk（コンソールドライバ作業領域）
・himempol（メモリプール領域）
・hitskstk（タスク用スタック領域）
・hiintstk（割込みハンドラ用スタック領域）
・hitrcbuf（トレースバッファ領域）

(h) RAM領域にメモリ配置するセクションを追加する場合に指定します。
(i) 外部参照シンボルの強制定義を指定します。
以下の外部参照シンボルは、必ず指定してください。定義する値は、リセット直後に使用可能なRAM領域の最終アドレス＋１にしてください。

・__0Hpon_sp（パワーオンリセット時のスタックポインタの初期値）
・__0Hman_sp（マニュアルリセット時のスタックポインタの初期値）

(j) 実行開始アドレス（__0Hrst_srv：リセットサービスルーチン）を指定しています。
(k) ロードモジュールの出力ファイル名（hisystem.abs）を指定しています。
(l) リンケージマップリストの出力ファイル名（hisystem.map）を指定しています。
(m) ロードモジュールのファイル形式（アブソリュート形式）を指定しています。
(n) デバッグ情報を出力することを指定しています。

なお、リンケージエディタの詳しい説明は、『Hシリーズ　リンケージエディタ　ユーザーズマニュアル』を参照してください。

３．６．３　makeファイルの実行

１．UNIX版HI-SH77

UNIX版HI-SH77では、システムを自動的に構築するためのmakeファイル（himake）を提供しています。

UNIXシステムのmake機能を利用すれば、プログラムの更新状況（ファイル日付）から、必要なソースプログラムのコンパイル（またはアセンブル）とシステムの結合を自動的に行なうことができ、プログラムの変更管理が容易になります。

出荷時のhimakeは、提供する全ソースプログラムをコンパイルし、サブコマンドファイルhimake.subを用いてシステムを結合します。必要に応じてhimakeを修正することで、ユーザシステムの構築を自動化することができます。

なお、make機能に関する詳細は、ホストコンピュータ上のUNIXのマニュアル等を参照してください。

makeファイルの実行は、以下のコマンドを入力します。

```
make△-f△himake (RET)
```

２．MS-DOS版HI-SH77

MS-DOS版HI-SH77では、システムを自動的に構築するためのバッチファイル（himake.bat）を提供しています。

出荷時のhimake.batは、提供する全ソースプログラムをコンパイルし、サブコマンドファイルhimake.subを用いてシステムを結合します。必要に応じてhimake.batを修正することで、ユーザシステムの構築を自動化することができます。

himake.batの実行は、以下のコマンドを入力します。

```
himake (RET)
```

himake.batの処理内容は、PC-9800シリーズ版とIBM PC版では異なります。

(1)PC-9800シリーズ用himake.bat

PC-9800シリーズ用himake.batは、プログラムの更新状況（ファイル日付）から、必要なソースプログラムのコンパイル（またはアセンブル）とシステムの構築を自動的に行なうmakeファイルを起動します。

なお、make機能はPC9800シリーズ日本語MS-DOS拡張機能セットのものを対象としています。make機能に関する詳細は、日本語MS-DOS拡張機能セットのマニュアル等を参照してください。

(2)IBM PC用himake.bat

IBM PCのMS-DOSには、プログラムの更新状況（ファイル日付）から必要なソースプログラムのコンパイルを行なうmake機能がありません。

IBM PC用himake.batは全てのコンパイルコマンドを実行するため時間がかかります。そのため、ユーザプログラムのコンパイル（またはアセンブル）とシステムの結合はコマンドライン上で行なうことを勧めます。

以下に、ユーザプログラムのコンパイル（またはアセンブル）とシステムの結合方法を示します。

コマンドライン上

```
<コンパイルコマンド>（RET） ……(a)
lnk -subcommand=himake.sub（RET）……(b)
cnvs hisystem.abs hisystem.mot（RET）……(c)
```

【説明】

(a) ユーザプログラムをコンパイル（またはアセンブル）します。

(b) HI-SH77システムとユーザプログラムのオブジェクトをリンクします。

(c) オブジェクトコンバータを実行し、Ｓタイプロードモジュールファイルを作成します。

３．７　システムの起動

システムを起動するには、ロードモジュールをインサーキットエミュレータなどの実機デバッグ装置にダウンロードする方法と、ROMに書き込んで実機に搭載する方法があります。

３．７．１　E7000PCを使用したロードモジュールのダウンロード

E7000PC（インサーキットエミュレータ）にロードモジュールをダウンロードする方法を説明します。

⑴ダウンロード
　⒜**E7000PC**システムを起動します。
　⒝**PC**インタフェースソフトウェアを起動します。

```
IPI(RET)
        （PCインタフェースソフトウェア起動メッセージ）
        （E7000PC起動メッセージ）
:       （E7000プロンプト）
```

　⒞**E7000**の**MAP**コマンドでエミュレーションメモリを割り当てます。以下の例では**H'0**番地から**H'7FFFF**番地および**H'C000000**番地から**H'C07FFFF**番地にエミュレーションメモリを割り当てています。なお、**H'0**番地から**H'7FFFF**番地はエミュレーションメモリを書き込み禁止として割り当てています。

```
: MAP△0△7FFFF;SW(RET)
: MAP△C000000△C07FFFF;S(RET)
```

　⒟　**E7000**の**LOAD**コマンドを使用し、**IBM PC**からロードモジュールをダウンロードします。

```
: LOAD△;R:<アブソリュートロードモジュールのファイル名>(RET)
```

⑵　システムの起動
　ロードモジュールの開始アドレスから実行すると、システムが起動します。
　開始アドレスには、カーネルのリセットサービスルーチンの先頭アドレス（__0Hrst_srv）を指定します。なお、E7000のRESETコマンドを入力すると、開始アドレスは自動的に設定されます。

```
: reset(RET)
: go(RET)
```

３．７．２　ROMによる実機搭載

ROMを実機に搭載する方法を説明します。

⑴　アブソリュートロードモジュールをモトローラSタイプロードモジュールに変換します。

```
cnvs△<アブソリュートロードモジュールのファイル名> (RET)
```

⑵　モトローラSタイプロードモジュールをROMライタへ転送し、ROMに書き込みます。

⑶　ROMを実機に搭載します。電源投入（パワーオンリセット）により、システムが起動します。

付録A．メモリ容量の算出

A．1　システムのメモリ領域

表A－1にシステムのメモリ領域（RAM）を示します。

システム全体のメモリ容量は、表A－1に示す各領域とユーザが独自に使用する領域の総和となります。

表A－1　システムのメモリ領域

項番	メモリ領域	メモリサイズ定義ファイル	メモリ領域定義ファイル	セクション名
1	カーネル作業領域		hisuptbl.inc	hiknlwrk
2	タスク用スタック領域	hitskstk.h	hitskstk.c	hitskstk
3	割込みハンドラ用スタック領域	hiintstk.h	hiintstk.c	hiintstk
4	メモリプール領域	himempol.h	himempol.c	himempol
5	トレースバッファ領域	hitrcbuf.h	hitrcbuf.c	hitrcbuf

A．2　カーネル作業領域のメモリ容量

表A－2にカーネル作業領域のメモリ容量算出表を示します。

カーネル作業領域は、セットアップテーブル（hisuptbl.c）のユーザ定義内容に基づいて、セットアップテーブルインクルード（hisuptbl.inc）で自動的に確保されます。表A－2は、ユーザシステム作成の参考として利用してください。

表A－2　カーネル作業領域のメモリ容量算出表

項番	内　訳	計　算　式	容量(バイト)	備考
1	カーネル スタック領域	{56 または 148}*1 ＋ 12 ×（hi_uppintnst(割込みﾈｽﾄ数:＞ｶｰﾈﾙﾏｽｸﾚﾍﾞﾙ) ＋ hi_lowintnst(割込みﾈｽﾄ数:≦ｶｰﾈﾙﾏｽｸﾚﾍﾞﾙ)）＋ hi_svcstksz(拡張SVCﾊﾝﾄﾞﾗの最大ｽﾀｯｸｻｲｽﾞ) ＋ hi_inistksz(ｼｽﾃﾑ初期化ﾊﾝﾄﾞﾗのｽﾀｯｸｻｲｽﾞ)		hi_svcstkszは、 hi_maxsvccd > 0 の場合のみ必要 hi_inistkszは、 hi_inihdr = USE の場合のみ必要
2	システム管理領域	12		
3	タスク管理領域	32 ＋ 20 × hi_maxtskid(最大タスクID) ＋ 4 × hi_maxtskpri(最大タスク優先度)		
4	イベントフラグ 管理領域	6 × hi_maxflgid(最大イベントフラグID)		
5	セマフォ管理領域	4 × hi_maxsemid(最大セマフォID)		
6	メイルボックス 管理領域	8 × hi_maxmbxid(最大メイルボックスID)		
7	メモリプール 管理領域	16 × hi_maxmplid(最大メモリプールID)		
8	時間管理領域	12		*2
9	トレースバッファ 管理領域	4		hi_trace = USE の場合のみ必要
合　計				

【注】*1　hi_maxsvccd=0 かつ hi_inihdr=NOTUSE の場合は56、その他の場合は148となります。

*2　hi_wai_tsk = USE または hi_get_tim = USE または hi_set_tim = USE の場合のみ必要です。

Ａ．３　タスク用スタック領域のメモリ容量

表Ａ－３にタスク用スタック領域のメモリ容量算出表を示します。本算出表を使用して、タスクID毎のタスク用スタックサイズを決定してください。タスク用スタック領域全体のメモリ容量は、各タスクID毎に求めたメモリ容量の総和となります。

なお、「共有スタック機能」を使用する場合は、同じスタック領域を使用するタスクの中で、最も大きなスタックサイズを指定してください。

表Ａ－３　タスク用スタック領域のメモリ容量算出表

項番	内　訳	計　算　式	容量(バイト)	備考
1	タスクが独自に使用するスタック領域	C言語でプログラムを作成する場合、 SHシリーズCコンパイラユーザーズマニュアルの「2.メモリ領域の割り付け」を参照してください		*1
2	カーネル使用領域	124		
3	多重割込み用スタック領域	12 ×（hi_uppintnst(割込みﾈｽﾄ数:＞ｶｰﾈﾙﾏｽｸﾚﾍﾞﾙ) ＋ hi_lowintnst(割込みﾈｽﾄ数:≦ｶｰﾈﾙﾏｽｸﾚﾍﾞﾙ)）		
4	例外処理ルーチン用スタック領域	8　＋ 例外処理ルーチンで使用するスタック容量		例外が発生する場合に加算します
合　　計				

【注】*1　タスクを非タスク（割込みマスクレベル≠0）で実行するケースがある場合、28を加算してください。

Ａ．４　割込みハンドラ用スタック領域のメモリ容量

表Ａ－４に割込みハンドラ用スタック領域のメモリ容量算出表を示します。本算出表を使用して、割込みレベル毎の割込みハンドラ用スタックサイズを決定してください。割込みハンドラ用スタック領域全体のメモリ容量は、各割込みレベル毎に求めたメモリ容量の総和となります。

なお、割込みハンドラのスタック領域は、割込みレベル毎に共有できます。同じ割込みレベルの割込みハンドラの中で、最も大きなスタックサイズを指定してください。

また、表Ａ－４項番１の容量が０で、かつシステムコールを発行しない場合、その割込みハンドラではスタックを確保する必要はありません。したがって、起動時にスタックを切り換える必要もありません。

表Ａ－４　割込みハンドラ用スタック領域のメモリ容量算出表

項番	内　訳	計　算　式	容量(バイト)	備考
1	割込みハンドラが独自に使用するスタック領域	C言語でプログラムを作成する場合、 SHシリーズCコンパイラユーザーズマニュアルの「2.メモリ領域の割り付け」を参照してください		
2	カーネル使用領域	16または148		*1
3	多重割込み用スタック領域	12 ×（割込みネスト数(＞ｶｰﾈﾙﾏｽｸﾚﾍﾞﾙ,＞自ﾚﾍﾞﾙ)*2＋ 割込みネスト数(≦ｶｰﾈﾙﾏｽｸﾚﾍﾞﾙ,＞自ﾚﾍﾞﾙ)*3）		
4	例外処理ルーチン用スタック領域	8　＋ 例外処理ルーチンで使用するスタック容量		例外が発生する場合に加算します
合　　計				

【注】*1　カーネル割込みマスクレベルより高い割込みハンドラの場合は16、カーネル割込みマスクレベル以下の割込みハンドラの場合は148となります。

*2　カーネル割込みマスクレベルより高く、かつ自割込みレベルより高い割込みのネスト数です。

*3　カーネル割込みマスクレベル以下、かつ自割込みレベルより高い割込みのネスト数です。

Ａ．５　メモリプール領域のメモリ容量

表Ａ－５にメモリプール領域のメモリ容量算出表を示します。本算出表を使用して、メモリプールID毎のメモリプール領域のメモリ容量を求めることができます。メモリプール領域全体のメモリ容量は、各メモリプールID毎に求めたメモリ容量の総和となります。

なお、メモリプール領域のメモリ容量は、メモリプール領域定義ヘッダ（himempol.h）にメモリブロックサイズとメモリブロック数を定義することで、自動的に算出されます。

表Ａ－５　メモリプール領域のメモリ容量算出表

項番	内　訳	計　算　式	容量(バイト)	備考
1	メモリプール領域	（メモリブロックサイズ　＋　4）　× メモリブロック数		ﾒﾓﾘﾌﾞﾛｯｸ毎に、 管理領域(4バイト) が必要です
合　　計				

Ａ．６　トレースバッファ領域のメモリ容量

表Ａ－６にトレースバッファ領域のメモリ容量算出表を示します。本算出表を使用して、トレースバッファ領域のメモリ容量を求めることができます。

なお、トレースバッファ領域のメモリ容量は、トレースバッファ領域定義ヘッダ（hitrcbuf.h）にトレース情報取得数を定義することで、自動的に算出されます。

表Ａ－６　トレースバッファ領域のメモリ容量算出表

項番	内　訳	計　算　式	容量(バイト)	備考
1	トレース バッファ管理領域	16		
2	トレース エントリ情報領域	36　×　トレース情報取得数		
合　　計				

Ａ．７　メモリ容量算出の注意

ソースプログラムをコンパイルした場合は、必ずスタック使用サイズを算出してスタックを確保してください。算出方法は、SHシリーズCコンパイラユーザーズマニュアルのシステム組み込み編「２．２　動的領域の割り付け」を参照してください。また、このサイズを「ユーザ（タスク、割り込みハンドラ）独自」に設定してください。

標準提供のサンプルプログラムをそのまま使用する場合も、使用するコンパイラのバージョン、オプションによってはスタック使用サイズが変わる場合があるので、必ずスタック使用サイズを確認の上、スタック定義ファイルを設定してください。

付録Ｂ．システムの構築例

Ｂ．１　概要

提供しているタイマドライバとコンソールドライバのサンプルプログラムを使用して、ＨＩ－ＳＨ７７システムを作成する例を示します。ＨＩ－ＳＨ７７システム作成は、IBM PC上で行ないます。

なお、タイマドライバとコンソールドライバのサンプルプログラムについての詳細は、『HI-SH77ユーザーズマニュアル』を参照してください。

Ｂ．２　システム構築の環境

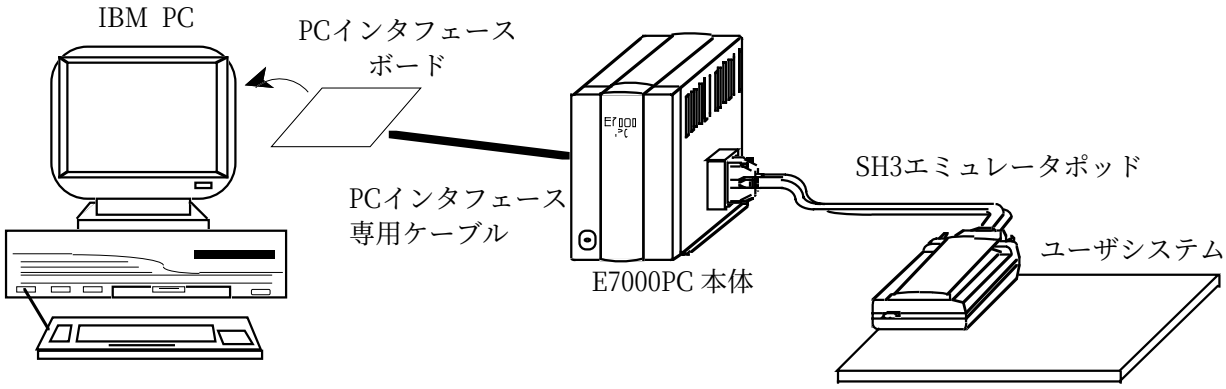
Ｂ．２．１　ハードウェア構成

図Ｂ－１にシステムを構築するために必要なハードウェア構成を示します。

ホストコンピュータにIBM PC、実機デバッグ装置にSH3用E7000PCエミュレータ（以下、E7000と略します）を使用します。

（１）ハードウェア
- IBM PC
  ハードディスク装置１台
  フロッピーディスクドライバ１台

（２）システムソフトウェア
- MS-DOS Ver3.3



図Ｂ－１　システム構築のハードウェア構成

Ｂ．２．２　ソフトウェア構成

(1) ユーティリティソフトウェア

例題システムを構築するために必要なユーティリティソフトウェアを次に示します。

- SHシリーズ　Cコンパイラ
- Hシリーズ　リンケージエディタ
- Hシリーズ　オブジェクトコンバータ
- SH3E7000PC用グラフィカルユーザインタフェースソフトウェアまたはPCインタフェースソフトウェア

(2) ＨＩ－ＳＨ７７のソフトウェア

「２．　インストール」の表２－１に示した提供ファイルを使用して、例題システムを構築します。

B．３　例題システムの概要

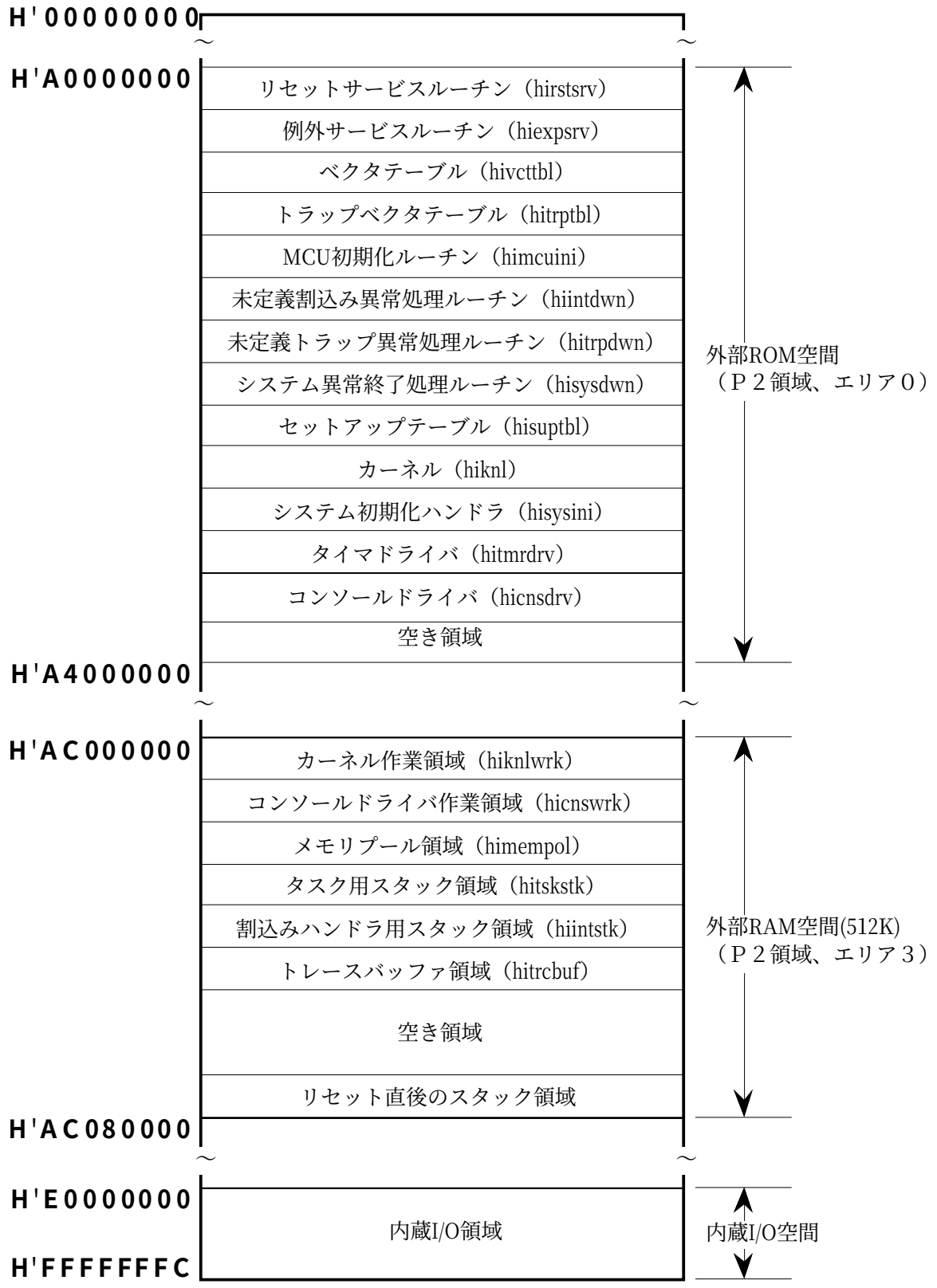
表Ｂ－１に例題システムの概要を示します。

表Ｂ－１　例題システムの概要

項番	項　目	構　築　内　容		
1	MCU	SH7708（60MHz）		
2	メモリ構成	ROM	外部ROM	
		RAM	外部RAM（512KB）	
3	カーネル	セットアップテーブルのチェック機能		使用
		システムコール		すべて使用
		カーネル割込みマスクレベル		14
		割込みネスト数（＞カーネル割込みマスクレベル）		0
		割込みネスト数（≦カーネル割込みマスクレベル）		2
		パラメータチェック機能		あり
4	割込み	割込みレベル11	シリアル送受信割込み（コンソールドライバ）	
		割込みレベル13	タイマ割込み（タイマドライバ）	
5	タスク	タスクID	1～5	
		ID=1	コンソールドライバタスク	
		ID=2～5	未使用	
		タスク優先度	1～5	
		初期登録タスク	コンソールドライバタスク	
6	イベントフラグ	イベントフラグID	1～3	
7	セマフォ	セマフォID	1～3	
8	メイルボックス	メイルボックスID	1～3	
9	メモリプール	メモリプールID	1～2	
		ID=1	メモリブロックサイズ	32
			メモリブロック数	4
		ID=2	メモリブロックサイズ	16
			メモリブロック数	8
10	拡張SVC	未登録		
11	MCU初期化ルーチン	キャッシュ無効化、MMU無効化、バスステートコントローラ設定、RAMクリア後、カーネル起動処理へジャンプ		
12	システム初期化ハンドラ	タイマの初期化とコンソールドライバタスクの起動		
13	トレース機能	トレース情報取得数	32	
14	未定義割込み異常処理ルーチン	例外をマスク解除、割込みをマスク（レベル15）して永久ループ		
15	未定義トラップ異常処理ルーチン	例外をマスク解除、割込みをマスク（レベル15）して永久ループ		
16	システム異常終了処理ルーチン	例外をマスク解除、割込みをマスク（レベル15）して永久ループ		

B．４　例題システムのメモリマップ

図Ｂ－２に例題システムのメモリマップを示します。



【注】（　）内は、セクション名です。

図Ｂ－２　例題システムのメモリマップ

B．５　例題システムのメモリ容量

表Ｂ－２に例題システムのメモリ（RAM）容量算出表を示します。
各メモリ領域の算出方法は、「付録A　メモリ容量の算出」を参照してください。

表Ｂ－２　例題システムのメモリ容量算出表

項番	内　訳	計　算　式	容量（バイト）
1	カーネル作業領域	(カーネルスタック領域 = 148 ＋ 12×(0 ＋ 2)＋ 0 ＋ 4) ＋ (システム管理領域 = 12) ＋ (タスク管理領域 = 32 ＋ 20×5 ＋ 4×5) ＋ (イベントフラグ管理領域 = 6×3) ＋ (セマフォ管理領域 = 4×3) ＋ (メイルボックス管理領域 = 8×3) ＋ (メモリプール管理領域 = 16×2) ＋ (時間管理領域 = 12) ＋ (トレースバッファ管理領域 = 4)	442 バイト
2	タスク用 スタック領域	((タスクID = 1) = 124 + (12× (0 ＋ 2)) + 52 + 0) ＋ ((タスクID = 2) = 124 + (12× (0 ＋ 2)) + 108 + 0) ＋ ((タスクID = 3) = 124 + (12× (0 ＋ 2)) + 108 + 0) ＋ ((タスクID = 4) = 124 + (12× (0 ＋ 2)) + 108 + 0) ＋ ((タスクID = 5) = 0) ･･･ (タスクID = 4)と共有スタックのため	968 バイト
3	割込みハンドラ用 スタック領域	((割込みレベル=11) = 148 + (12×(0 ＋ 1) + 44 + 0) ＋ ((割込みレベル=13) = 148 + (12×(0 ＋ 0) + 28 + 0)	380 バイト
4	メモリプール領域	((メモリプールID = 1) = (32 ＋ 4) × 4) ＋ ((メモリプールID = 2) = (16 ＋ 4) × 8)	304 バイト
5	トレース バッファ領域	(トレースバッファ管理領域 = 16) ＋ (トレースエントリ情報領域 = 36 × 32)	1168 バイト
6	その他	(コンソールドライバ作業領域 = 2) ＋ (境界補正 = 4)	6 バイト
合　　計			3268 バイト

B．6　例題システムのセットアップテーブル

図Ｂ－３に例題システムのセットアップテーブル（hisuptbl.c）を示します。

```
/************************************************************************/
/*                                                                      */
/*                                                                      */
/*              HI-SH77 setup table ( Ver. 1.0 )                        */
/*                                                                      */
/*                                                                      */
/*              Copyright (c)  Hitachi, Ltd. 1995.                      */
/*              Licensed Material of Hitachi, Ltd.                      */
/*                                                                      */
/*              HI-SH77(HS0770ITCN1SM) V1.0                             */
/*                                                                      */
/*                                                                      */
/************************************************************************/
/************************************************************************/
/*SPECIFICATIONS ;                                                      */
/*  FILE      = hisuptbl.c ;                                            */
/*  DATE      = 95/02/20 ;                                              */
/*  AUTHOR    = Hitachi, Ltd. ;                                         */
/*  FUNCTION  = HI-SH77 setup table ;                                   */
/*  ATTRIBUTE = PUBLIC ;                                                */
/*  HISTORY   = V1.0 ;                                                  */
/*END OF SPECIFICATIONS ;                                               */
/************************************************************************/
#include    "itron.h"
#include    "hisuptbl.h"

/************************************************************************/
/*                                                                      */
/*          Definition setup table check function use in user system    */
/*                                                                      */
/*          Usage   :   #define hi_chksut { USE | NOTUSE }              */
/*                      USE     :   use the setup table check function  */
/*                      NOTUSE  :   not use the setup table check function  */
/*                                                                      */
/************************************************************************/
#define hi_chksut       USE             /* setup table check function       */←セットアップテーブルチェック機能 = 使用

/************************************************************************/
/*                                                                      */
/*          Definition SVC use in user system                           */
/*                                                                      */
/*          Usage   :   #define hi_xyyy_zzz { USE | NOTUSE }            */
/*                      USE     :   use the SVC (xyyy_zzz)              */
/*                      NOTUSE  :   not use the SVC (xyyy_zzz)          */
/*                                                                      */
/************************************************************************/
/*---------------------- task management function -------------------------*/←全システムコール = 使用
#define hi_cre_tsk      USE             /* cre_tsk  SVC                     */
#define hi_sta_tsk      USE             /* sta_tsk  SVC                     */
#define hi_ista_tsk     USE             /* ista_tsk SVC                     */
#define hi_del_tsk      USE             /* del_tsk  SVC                     */
#define hi_ext_tsk      USE             /* ext_tsk  SVC                     */
#define hi_exd_tsk      USE             /* exd_tsk  SVC                     */
#define hi_ter_tsk      USE             /* ter_tsk  SVC                     */
#define hi_chg_pri      USE             /* chg_pri  SVC                     */
#define hi_ichg_pri     USE             /* ichg_pri SVC                     */
#define hi_rot_rdq      USE             /* rot_rdq  SVC                     */
#define hi_irot_rdq     USE             /* irot_rdq SVC                     */
#define hi_rel_wai      USE             /* rel_wai  SVC                     */
#define hi_irel_wai     USE             /* irel_wai SVC                     */
#define hi_get_tid      USE             /* get_tid  SVC                     */
#define hi_tsk_sts      USE             /* tsk_sts  SVC                     */
```

図Ｂ－３　例題システムのセットアップテーブル（hisuptbl.c）（１／６）

```
/*-------------- task synchronization management function -----------------*/
#define hi_sus_tsk      USE             /* sus_tsk  SVC                     */
#define hi_isus_tsk     USE             /* isus_tsk SVC                     */
#define hi_rsm_tsk      USE             /* rsm_tsk  SVC                     */
#define hi_irsm_tsk     USE             /* irsm_tsk SVC                     */
#define hi_slp_tsk      USE             /* slp_tsk  SVC                     */
#define hi_wai_tsk      USE             /* wai_tsk  SVC                     */
#define hi_wup_tsk      USE             /* wup_tsk  SVC                     */
#define hi_iwup_tsk     USE             /* iwup_tsk SVC                     */
#define hi_can_wup      USE             /* can_wup  SVC                     */

/*-------------- synchronization and communication function ----------------*/
#define hi_set_flg      USE             /* set_flg  SVC                     */
#define hi_iset_flg     USE             /* iset_flg SVC                     */
#define hi_clr_flg      USE             /* clr_flg  SVC                     */
#define hi_wai_flg      USE             /* wai_flg  SVC                     */
#define hi_pol_flg      USE             /* pol_flg  SVC                     */
#define hi_flg_sts      USE             /* flg_sts  SVC                     */
#define hi_sig_sem      USE             /* sig_sem  SVC                     */
#define hi_isig_sem     USE             /* isig_sem SVC                     */
#define hi_wai_sem      USE             /* wai_sem  SVC                     */
#define hi_preq_sem     USE             /* preq_sem SVC                     */
#define hi_sem_sts      USE             /* sem_sts  SVC                     */
#define hi_snd_msg      USE             /* snd_msg  SVC                     */
#define hi_isnd_msg     USE             /* isnd_msg SVC                     */
#define hi_rcv_msg      USE             /* rcv_msg  SVC                     */
#define hi_prcv_msg     USE             /* prcv_msg SVC                     */
#define hi_mbx_sts      USE             /* mbx_sts  SVC                     */

/*--------------------- interrupt management function ----------------------*/
#define hi_chg_ims      USE             /* chg_ims  SVC                     */
#define hi_ims_sts      USE             /* ims_sts  SVC                     */

/*------------------- memory pool management function ---------------------*/
#define hi_get_blk      USE             /* get_blk  SVC                     */
#define hi_pget_blk     USE             /* pget_blk SVC                     */
#define hi_rel_blk      USE             /* rel_blk  SVC                     */
#define hi_mpl_sts      USE             /* mpl_sts  SVC                     */

/*----------------------- timer management function ------------------------*/
#define hi_set_tim      USE             /* set_tim  SVC                     */
#define hi_get_tim      USE             /* get_tim  SVC                     */

/*--------------------- system management function -----------------------*/
#define hi_get_ver      USE             /* get_ver  SVC                     */
```

図Ｂ－３　例題システムのセットアップテーブル（hisuptbl.c）（２／６）

```
/************************************************************************/
/*                                                                      */
/*          Definition kernel information                               */
/*                                                                      */
/************************************************************************/
#define hi_knlmsklvl    14              /* kernel mask level of interrupt   */←カーネル割込みマスクレベル = 14
                                        /*      range : [1...15]            */
#define hi_uppintnst    0               /* number of interrupt levels       */←割込みネスト数 = 0
                                        /*           (>  kernel mask level) */（>カーネル割込みマスクレベル）
                                        /*      range : [0...]              */
#define hi_lowintnst    2               /* number of interrupt levels       */←割込みネスト数 = 2
                                        /*           (<= kernel mask level) */（≦カーネル割込みマスクレベル）
                                        /*      range : [0...15]            */

/************************************************************************/
/*                                                                      */
/*          Definition task information                                 */
/*                                                                      */
/************************************************************************/
#define hi_maxtskid     5               /* max task id                      */←最大タスクID = 5
                                        /*      range : [0...1023]          */
#define hi_maxtskpri    5               /* max task priority                */←最大タスク優先度 = 5
                                        /*      range : [1...255]           */
#define hi_initsknum    1               /* number of initial task           */←初期登録タスク数 = 1
                                        /*      range : [0...1023]          */

/************************************************************************/
/*                                                                      */
/*          Definition task information to setup table (INITSP, INITSK) */
/*                                                                      */
/************************************************************************/
/*--------------------- define task stack end address ----------------------*/
#if     hi_maxtskid                     /* case of hi_maxtskid > 0          */
#include        "hitskstk.h"            /* include "hitskstk.h"             */←タスク用スタックサイズの参照
extern  VW      hi_tskstk1[];           /* task stack of tskid = 1          */←タスク用スタック先頭アドレスの参照
extern  VW      hi_tskstk2[];           /* task stack of tskid = 2          */
extern  VW      hi_tskstk3[];           /* task stack of tskid = 3          */
extern  VW      hi_tskstk4[];           /* task stack of tskid = 4, 5       */

const   INITSP  _OHinitsp[hi_maxtskid] = {                                  ←タスク用スタック最終アドレスの定義
        /* define task stack end address    */
            (VW *)&hi_tskstk1[(hi_tskstksz1) / sizeof(VW)], /* tskid = 1    */
            (VW *)&hi_tskstk2[(hi_tskstksz2) / sizeof(VW)], /* tskid = 2    */
            (VW *)&hi_tskstk3[(hi_tskstksz3) / sizeof(VW)], /* tskid = 3    */
            (VW *)&hi_tskstk4[(hi_tskstksz4) / sizeof(VW)], /* tskid = 4    */←タスクID=4とタスクID=5は共有スタック
            (VW *)&hi_tskstk4[(hi_tskstksz4) / sizeof(VW)], /* tskid = 5    */
        };
#else                                   /* case of hi_maxtskid = 0          */
#define _OHinitsp       NADR            /* not define task stack            */
#endif
```

図Ｂ－３　例題システムのセットアップテーブル（hisuptbl.c）（３／６）

```
/*-------------------- define initial task information ---------------------*/←初期登録タスク情報の定義
#if     hi_initsknum                    /* case of hi_initsknum > 0          */
extern  TASK    hi_cnsdrv();            /* console driver task               */←タスク開始アドレスの参照

const   INITSK  _OHinitsk[hi_initsknum] = {
        /* define console driver task   */
            (ID)1,                      /* task id              = 1          */←タスクIDの定義
            (TPRI)1,                    /* initial task priority = 1         */←初期タスク優先度の定義
            (TASKP)hi_cnsdrv,           /* task start address    = hi_cnsdrv*/←タスク開始アドレスの定義
        };
#else                                   /* case of hi_initsknum = 0          */
#define _OHinitsk       NADR            /* not define initial task           */
#endif

/*************************************************************************/
/*                                                                       */
/*          Definition synchronization and communication object information */
/*                                                                       */
/*************************************************************************/
#define hi_maxflgid     3               /* max event flag id                 */←最大イベントフラグID = 3
                                        /*      range : [0...1023]           */
#define hi_maxsemid     3               /* max semaphore id                  */←最大セマフォID = 3
                                        /*      range : [0...1023]           */
#define hi_maxmbxid     3               /* max mail box id                   */←最大メイルボックスID = 3
                                        /*      range : [0...1023]           */

/*************************************************************************/
/*                                                                       */
/*          Definition memory pool information                           */
/*                                                                       */
/*************************************************************************/
#define hi_maxmplid     2               /* max memory pool id                */←最大メモリプールID = 2
                                        /*      range : [0...1023]           */

/*************************************************************************/
/*                                                                       */
/*          Definition memory pool information to setup table (INIMPL)   */
/*                                                                       */
/*************************************************************************/
#if     hi_maxmplid                     /* case of hi_maxmplid > 0           */
#include        "himempol.h"            /* include "himempol.h"              */←メモリブロックサイズ､メモリブロック数の参照
extern  VW      hi_mempol1[];           /* memory pool of mplid = 1          */←メモリプール先頭アドレスの参照
extern  VW      hi_mempol2[];           /* memory pool of mplid = 2          */

const   INIMPL  _OHinimpl[hi_maxmplid] = {                                    ←メモリプール情報の定義
        /* define memory pool of mplid = 1  */
            (VW *)hi_mempol1,           /* top address = hi_mempol1          */←メモリプール先頭アドレスの定義
            (UW)hi_memblksz1,           /* block size  = hi_memblksz1        */←メモリブロックサイズの定義
            (UW)hi_memblkcnt1,          /* block count = hi_memblkcnt1       */←メモリブロック数の定義

        /* define memory pool of mplid = 2  */
            (VW *)hi_mempol2,           /* top address = hi_mempol2          */
            (UW)hi_memblksz2,           /* block size  = hi_memblksz2        */
            (UW)hi_memblkcnt2,          /* block count = hi_memblkcnt2       */
        };
#else                                   /* case of hi_maxmplid = 0           */
#define _OHinimpl       NADR            /* not define memory pool            */
#endif
```

図Ｂ－３　例題システムのセットアップテーブル（hisuptbl.c）（４／６）

```
/************************************************************************/
/*                                                                      */
/*          Definition extended SVC information                         */
/*                                                                      */
/************************************************************************/
#define hi_maxsvccd     0               /* max function code of extended SVC*/←拡張SVC＝未登録
                                        /*      range : [0...255]           */

/************************************************************************/
/*                                                                      */
/*          Definition extended SVC information to setup table (INISVC) */
/*                                                                      */
/************************************************************************/
#if     hi_maxsvccd                     /* case of hi_maxsvccd > 0      */
                                                ←拡張SVCを登録する場合、拡張SVCハンドラ開始アドレスを参照してください
extern  SVCHDR  hi_svchdr1();           /* SVCHDR of fncd = 1           */
extern  SVCHDR  hi_svchdr2();           /* SVCHDR of fncd = 2           */

                                                ←拡張SVCを登録する場合、拡張SVCハンドラ開始アドレスを定義してください
const   INISVC  _OHinisvc[hi_maxsvccd] = {
            (SVCHDRP)hi_svchdr1,        /* define SVCHDR of fncd = 1    */
            (SVCHDRP)hi_svchdr2,        /* define SVCHDR of fncd = 2    */
        };

/************************************************************************/
/*                                                                      */
/*          Definition stack size of extended SVC handler (multiple of 4)*/
/*                                                                      */
/*          Usage   :   #define hi_svcstksz <maxsvcusesz> + <excusesz>   */
/*                                                                      */
/*              <maxsvcusesz>   :   max size of extended SVC handler use*/
/*              <excusesz>      :   size of exception use               */
/*                                                                      */
/************************************************************************/
                                                ←拡張SVCを登録する場合、拡張SVCハンドラのスタックサイズを定義してください
#define hi_svcstksz     (32 + 0)        /* define stack size of SVCHDR  */
#else                                   /* case of hi_maxsvccd = 0      */
#define _OHinisvc       NADR            /* not define SVCHDR            */
#define hi_svcstksz     0               /* not define stack size of SVCHDR */
#endif
```

図Ｂ－３　例題システムのセットアップテーブル（hisuptbl.c）（５／６）

```
/*****************************************************************************/
/*                                                                           */
/*         Definition system initial handler use in user system              */
/*                                                                           */
/*         Usage   :   #define hi_inihdr { USE | NOTUSE }                     */
/*                     USE     :   use the system initial handler            */
/*                     NOTUSE  :   not use the system initial handler        */
/*                                                                           */
/*****************************************************************************/
#define hi_inihdr       USE             /* system initial handler           */←システム初期化ハンドラ = 使用

/*****************************************************************************/
/*                                                                           */
/*         Definition system initial handler information to setup table      */
/*                                                              (INIIHDR)    */
/*****************************************************************************/
#if     hi_inihdr                       /* case of hi_inihdr = USE          */
extern  INIHDR  hi_sysini();            /* system initial handler           */←システム初期化ハンドラ開始アドレスの参照

const   INIIHDR _OHiniihdr[1] = {
            (INIHDRP)hi_sysini          /* define system initial handler    */←システム初期化ハンドラ開始アドレスの定義
        };

/*****************************************************************************/
/*                                                                           */
/*         Definition stack size of system initial handler (multiple of 4)   */
/*                                                                           */
/*         Usage   :   #define hi_inistksz <iniusesz> + <excusesz>            */
/*                                                                           */
/*             <iniusesz>      :   size of system initial handler use        */
/*             <excusesz>      :   size of exception use                     */
/*                                                                           */
/*****************************************************************************/
#define hi_inistksz     (4 + 0)         /* define stack size of INIHDR      */←システム初期化ハンドラスタックサイズの定義
#else                                   /* case of hi_inihdr = NOTUSE       */
#define _OHiniihdr      NADR            /* not define system initial handler*/
#define hi_inistksz     0               /* not define stack size of INIHDR  */
#endif

/*****************************************************************************/
/*                                                                           */
/*         Definition trace function use in user system                      */
/*                                                                           */
/*         Usage   :   #define hi_trace { USE | NOTUSE }                     */
/*                     USE     :   use the trace function                    */
/*                     NOTUSE  :   not use the trace function                */
/*                                                                           */
/*****************************************************************************/
#define hi_trace        USE             /* trace function                   */←トレース機能 = 使用

/*****************************************************************************/
/*                                                                           */
/*         Definition trace information to setup table (INITRC)              */
/*                                                                           */
/*****************************************************************************/
#if     hi_trace                        /* case of hi_trace = USE           */
#include        "hitrcbuf.h"            /* include "hitrcbuf.h"             */←トレース情報取得数の参照
extern  VW      hi_trcbuf[];            /* trace buffer                     */←トレースバッファ先頭アドレスの参照

const   INITRC  _OHinitrc[1] = {
        /* define trace information     */
            (VW *)hi_trcbuf,            /* top address     = hi_trcbuf      */←トレースバッファ先頭アドレスの定義
            (UW)hi_trcentnum            /* number of entry = hi_trcentnum   */←トレース情報取得数の定義
        };
#else                                   /* case of hi_trace = NOTUSE        */
#define _OHinitrc       NADR            /* not define trace information     */
#endif
#include    "hisuptbl.inc"
```

図Ｂ－３　例題システムのセットアップテーブル（hisuptbl.c）（６／６）

Ｂ．７　例題システムのタスク用スタック領域定義プログラム

(1)　タスク用スタック領域定義ヘッダ

図Ｂ－４に例題システムのタスク用スタック領域定義ヘッダ（hitskstk.h）を示します。

```
/************************************************************************/
/*                                                                      */
/*                                                                      */
/*            HI-SH77 header file for task stack ( Ver. 1.0 )           */
/*                                                                      */
/*                                                                      */
/*            Copyright (c)  Hitachi, Ltd. 1995.                        */
/*            Licensed Material of Hitachi, Ltd.                        */
/*                                                                      */
/*            HI-SH77(HS0770ITCN1SM) V1.0                               */
/*                                                                      */
/*                                                                      */
/************************************************************************/
/************************************************************************/
/*SPECIFICATIONS ;                                                      */
/*  FILE      = hitskstk.h ;                                            */
/*  DATE      = 95/02/20 ;                                              */
/*  AUTHOR    = Hitachi, Ltd. ;                                         */
/*  FUNCTION  = HI-SH77 header file for task stack ;                    */
/*  ATTRIBUTE = PUBLIC ;                                                */
/*  HISTORY   = V1.0 ;                                                  */
/*END OF SPECIFICATIONS ;                                               */
/************************************************************************/
/************************************************************************/
/*                                                                      */
/*         Definition constant size of task stack                       */
/*                                                                      */
/************************************************************************/
#define KNLTUSESZ       124                 /* size of kernel use       */←変更しないでください
/************************************************************************/
/*                                                                      */
/*         Definition task stack size of interrupt use                  */
/*                                                                      */
/*  Usage   :                                                           */
/*                                                                      */
/*  #define inttusesz       12 * ( <hi_uppintnst> + <hi_lowintnst> )    */
/*                                                                      */
/*      <hi_uppintnst>  :   number of interrupt levels                  */
/*                          (>  kernel mask level)                      */
/*      <hi_lowintnst>  :   number of interrupt levels                  */
/*                          (<= kernel mask level)                      */
/*                                                                      */
/************************************************************************/
#define inttusesz       12 * ( 0 + 2 )      /* size of interrupt use    */←割込みハンドラの使用サイズ定義
/************************************************************************/
/*                                                                      */
/*         Definition task stack size (multiple of 4)                   */
/*                                                                      */
/*  Usage   :                                                           */
/*                                                                      */
/*  #define hi_tskstk<n>    KNLTUSESZ + inttusesz + <tskusesz> + <excusesz> */
/*                                                                      */
/*      <n>             :   task stack number                           */
/*      <tskusesz>      :   size of task use                            */
/*          (note : when tasks are executed as non-tasks, 28 must be added) */
/*      <excusesz>      :   size of exception use                       */
/*                                                                      */
/************************************************************************/
#define hi_tskstksz1    (KNLTUSESZ + inttusesz +  52 + 0)  /* stack no. = 1 */←タスク用スタックサイズの定義
#define hi_tskstksz2    (KNLTUSESZ + inttusesz + 108 + 0)  /* stack no. = 2 */
#define hi_tskstksz3    (KNLTUSESZ + inttusesz + 108 + 0)  /* stack no. = 3 */
#define hi_tskstksz4    (KNLTUSESZ + inttusesz + 108 + 0)  /* stack no. = 4 */
```

図Ｂ－４　例題システムのタスク用スタック領域定義ヘッダ（hitskstk.h）

(2)　タスク用スタック領域定義ソース

図Ｂ－５に例題システムのタスク用スタック領域定義ソース（hitskstk.c）を示します。

```
/************************************************************************/
/*                                                                      */
/*                                                                      */
/*              HI-SH77 define task stack area ( Ver. 1.0 )             */
/*                                                                      */
/*                                                                      */
/*              Copyright (c)  Hitachi, Ltd. 1995.                      */
/*              Licensed Material of Hitachi, Ltd.                      */
/*                                                                      */
/*              HI-SH77(HS0770ITCN1SM) V1.0                             */
/*                                                                      */
/*                                                                      */
/************************************************************************/
/************************************************************************/
/*SPECIFICATIONS ;                                                      */
/*  FILE      = hitskstk.c ;                                            */
/*  DATE      = 95/02/20 ;                                              */
/*  AUTHOR    = Hitachi, Ltd. ;                                         */
/*  FUNCTION  = HI-SH77 define task stack area ;                        */
/*  ATTRIBUTE = PUBLIC ;                                                */
/*  HISTORY   = V1.0 ;                                                  */
/*END OF SPECIFICATIONS ;                                               */
/************************************************************************/
#include    "itron.h"
#include    "hitskstk.h"

/************************************************************************/
/*                                                                      */
/*          Definition task stack area                                  */
/*                                                                      */
/************************************************************************/
VW  hi_tskstk1[(hi_tskstksz1) / sizeof(VW)];    /* stack no. = 1        */←タスク用スタック領域の定義
VW  hi_tskstk2[(hi_tskstksz2) / sizeof(VW)];    /* stack no. = 2        */
VW  hi_tskstk3[(hi_tskstksz3) / sizeof(VW)];    /* stack no. = 3        */
VW  hi_tskstk4[(hi_tskstksz4) / sizeof(VW)];    /* stack no. = 4        */
```

図Ｂ－５　例題システムのタスク用スタック領域定義ソース（hitskstk.c）

Ｂ．８　例題システムの割込みハンドラ用スタック領域定義プログラム

(1)　割込みハンドラ用スタック領域定義ヘッダ

図Ｂ－６に例題システムの割込みハンドラ用スタック領域定義ヘッダ（hiintstk.h）を示します。

```
/************************************************************************/
/*                                                                      */
/*                                                                      */
/*            HI-SH77 header file for interrupt handler stack ( Ver. 1.0 )*/
/*                                                                      */
/*                                                                      */
/*            Copyright (c) Hitachi, Ltd. 1995.                         */
/*            Licensed Material of Hitachi, Ltd.                        */
/*                                                                      */
/*            HI-SH77(HS0770ITCN1SM) V1.0                               */
/*                                                                      */
/*                                                                      */
/************************************************************************/
/************************************************************************/
/*SPECIFICATIONS :                                                      */
/*  FILE      = hiintstk.h ;                                            */
/*  DATE      = 95/02/20 ;                                              */
/*  AUTHOR    = Hitachi, Ltd. ;                                         */
/*  FUNCTION  = HI-SH77 header file for interrupt handler stack ;       */
/*  ATTRIBUTE = PUBLIC ;                                                */
/*  HISTORY   = V1.0 ;                                                  */
/*END OF SPECIFICATIONS :                                               */
/************************************************************************/
/************************************************************************/
/*                                                                      */
/*         Definition constant size of interrupt handler stack          */
/*                                                                      */
/************************************************************************/
#define KNLUUSESZ       16          /* size of kernel use (upper interrupt) */   ←変更しないでください
#define KNLLUSESZ       148         /* size of kernel use (lower interrupt) */←変更しないでください
/************************************************************************/
/*                                                                      */
/*         Definition stack size of interrupt handler (multiple of 4)   */
/*                                                                      */
/*  Usage   :                                                           */
/*  case of interrupt level =  not use                                  */
/*    #define hi_intstk<n>  0                                           */
/*                                                                      */
/*  case of interrupt level >  kernel mask level                        */
/*    #define hi_intstk<n>  KNLUUSESZ + <othusesz> + <curusesz> + <excusesz>*/
/*                                                                      */
/*  case of interrupt level <= kernel mask level                        */
/*    #define hi_intstk<n>  KNLLUSESZ + <othusesz> + <curusesz> + <excusesz>*/
/*                                                                      */
/*      <n>             :   interrupt level number                      */
/*      <othusesz>      :   size of other interrupt use                 */
/*                      =   12 * ( <uppintnst> + <lowintnst> )          */
/*          <uppintnst> :   number of interrupt levels                  */
/*                          (>  kernel mask level, > current interrupt level)*/
/*          <lowintnst> :   number of interrupt levels                  */
/*                          (<= kernel mask level, > current interrupt level)*/
/*      <curusesz>      :   size of current interrupt handler use       */
/*      <excusesz>      :   size of exception use                       */
/*                                                                      */
/************************************************************************/
#define hi_intstksz1    0                                       /* level =  1 */
#define hi_intstksz2    0                                       /* level =  2 */
#define hi_intstksz3    0                                       /* level =  3 */
#define hi_intstksz4    0                                       /* level =  4 */
#define hi_intstksz5    0                                       /* level =  5 */
#define hi_intstksz6    0                                       /* level =  6 */
#define hi_intstksz7    0                                       /* level =  7 */
#define hi_intstksz8    0                                       /* level =  8 */
#define hi_intstksz9    0                                       /* level =  9 */
#define hi_intstksz10   0                                       /* level = 10 */
#define hi_intstksz11   (KNLLUSESZ + 12 * ( 0 + 1 ) + 44 + 0)  /* level = 11 */←割込みレベル＝11のスタックサイズ定義
#define hi_intstksz12   0                                       /* level = 12 */
#define hi_intstksz13   (KNLLUSESZ + 12 * ( 0 + 0 ) + 28 + 0)  /* level = 13 */←割込みレベル＝13のスタックサイズ定義
#define hi_intstksz14   0                                       /* level = 14 */
#define hi_intstksz15   0                                       /* level = 15 */
```

図Ｂ－６　例題システムの割込みハンドラ用スタック領域定義ヘッダ（hiintstk.h）

(2)　割込みハンドラ用スタック領域定義ソース

図Ｂ－７に例題システムの割込みハンドラ用スタック領域定義ソース（hiintstk.c）を示します。

```
/************************************************************************/
/*                                                                      */
/*                                                                      */
/*          HI-SH77 define stack area of interrupt handler ( Ver. 1.0 ) */
/*                                                                      */
/*                                                                      */
/*          Copyright (c) Hitachi, Ltd. 1995.                           */
/*          Licensed Material of Hitachi, Ltd.                          */
/*                                                                      */
/*          HI-SH77(HS0770ITCN1SM) V1.0                                 */
/*                                                                      */
/*                                                                      */
/************************************************************************/
/************************************************************************/
/*SPECIFICATIONS :                                                      */
/*  FILE      = hiintstk.c ;                                            */
/*  DATE      = 95/02/20 ;                                              */
/*  AUTHOR    = Hitachi, Ltd. ;                                         */
/*  FUNCTION  = HI-SH77 define stack area of interrupt handler ;        */
/*  ATTRIBUTE = PUBLIC ;                                                */
/*  HISTORY   = V1.0 ;                                                  */
/*END OF SPECIFICATIONS :                                               */
/************************************************************************/
#include    "itron.h"
#include    "hiintstk.h"

/************************************************************************/
/*                                                                      */
/*          Definition stack area of interrupt handler                  */
/*                                                                      */
/************************************************************************/
#if hi_intstksz1                            /* case of hi_intstksz1 > 0 */←割込みハンドラ用スタック領域の定義
VW  hi_intstk1[(hi_intstksz1) / sizeof(VW)];    /* interrupt level =  1     */
#endif
#if hi_intstksz2                            /* case of hi_intstksz2 > 0 */
VW  hi_intstk2[(hi_intstksz2) / sizeof(VW)];    /* interrupt level =  2     */
#endif
#if hi_intstksz3                            /* case of hi_intstksz3 > 0 */
VW  hi_intstk3[(hi_intstksz3) / sizeof(VW)];    /* interrupt level =  3     */
#endif
#if hi_intstksz4                            /* case of hi_intstksz4 > 0 */
VW  hi_intstk4[(hi_intstksz4) / sizeof(VW)];    /* interrupt level =  4     */
#endif
#if hi_intstksz5                            /* case of hi_intstksz5 > 0 */
VW  hi_intstk5[(hi_intstksz5) / sizeof(VW)];    /* interrupt level =  5     */
#endif
#if hi_intstksz6                            /* case of hi_intstksz6 > 0 */
VW  hi_intstk6[(hi_intstksz6) / sizeof(VW)];    /* interrupt level =  6     */
#endif
#if hi_intstksz7                            /* case of hi_intstksz7 > 0 */
VW  hi_intstk7[(hi_intstksz7) / sizeof(VW)];    /* interrupt level =  7     */
#endif
#if hi_intstksz8                            /* case of hi_intstksz8 > 0 */
VW  hi_intstk8[(hi_intstksz8) / sizeof(VW)];    /* interrupt level =  8     */
#endif
#if hi_intstksz9                            /* case of hi_intstksz9 > 0 */
VW  hi_intstk9[(hi_intstksz9) / sizeof(VW)];    /* interrupt level =  9     */
#endif
#if hi_intstksz10                           /* case of hi_intstksz10 > 0*/
VW  hi_intstk10[(hi_intstksz10) / sizeof(VW)];  /* interrupt level = 10     */
#endif
#if hi_intstksz11                           /* case of hi_intstksz11 > 0*/
VW  hi_intstk11[(hi_intstksz11) / sizeof(VW)];  /* interrupt level = 11     */
#endif
#if hi_intstksz12                           /* case of hi_intstksz12 > 0*/
VW  hi_intstk12[(hi_intstksz12) / sizeof(VW)];  /* interrupt level = 12     */
#endif
#if hi_intstksz13                           /* case of hi_intstksz13 > 0*/
VW  hi_intstk13[(hi_intstksz13) / sizeof(VW)];  /* interrupt level = 13     */
#endif
#if hi_intstksz14                           /* case of hi_intstksz14 > 0*/
VW  hi_intstk14[(hi_intstksz14) / sizeof(VW)];  /* interrupt level = 14     */
#endif
#if hi_intstksz15                           /* case of hi_intstksz15 > 0*/
VW  hi_intstk15[(hi_intstksz15) / sizeof(VW)];  /* interrupt level = 15     */
#endif
```

図Ｂ－７　例題システムの割込みハンドラ用スタック領域定義ソース（hiintstk.c）

Ｂ．９　例題システムのメモリプール領域定義プログラム

(1)　メモリプール領域定義ヘッダ

図Ｂ－８に例題システムのメモリプール領域定義ヘッダ（himempol.h）を示します。

```
/************************************************************************/
/*                                                                      */
/*                                                                      */
/*            HI-SH77 header file for memory pool ( Ver. 1.0 )          */
/*                                                                      */
/*                                                                      */
/*            Copyright (c)  Hitachi, Ltd. 1995.                        */
/*            Licensed Material of Hitachi, Ltd.                        */
/*                                                                      */
/*            HI-SH77(HS0770ITCN1SM) V1.0                               */
/*                                                                      */
/*                                                                      */
/************************************************************************/
/************************************************************************/
/*SPECIFICATIONS ;                                                      */
/*  FILE      = himempol.h ;                                            */
/*  DATE      = 95/02/20 ;                                              */
/*  AUTHOR    = Hitachi, Ltd. ;                                         */
/*  FUNCTION  = HI-SH77 header file for memory pool ;                   */
/*  ATTRIBUTE = PUBLIC ;                                                */
/*  HISTORY   = V1.0 ;                                                  */
/*END OF SPECIFICATIONS ;                                               */
/************************************************************************/
/************************************************************************/
/*                                                                      */
/*         Definition constant size of memory pool                      */
/*                                                                      */
/************************************************************************/
#define MBKMNGSZ        4           /* size of memory block management area */←変更しないでください

/************************************************************************/
/*                                                                      */
/*         Definition memory pool size                                  */
/*                                                                      */
/************************************************************************/
/*------------------------------ mplid = 1 -------------------------------*/←メモリプールID＝1の定義
#define hi_memblksz1    0x00000020      /* memory block size (multiple of 4)*/←メモリブロックサイズの定義
#define hi_memblkcnt1   0x00000004      /* memory block count               */←メモリブロック数の定義
                                        /* memory pool size                 */
#define hi_mempolsz1    ((MBKMNGSZ + hi_memblksz1) * hi_memblkcnt1)          ←メモリプールサイズの算出

/*------------------------------ mplid = 2 -------------------------------*/←メモリプールID＝2の定義
#define hi_memblksz2    0x00000010      /* memory block size (multiple of 4)*/
#define hi_memblkcnt2   0x00000008      /* memory block count               */
                                        /* memory pool size                 */
#define hi_mempolsz2    ((MBKMNGSZ + hi_memblksz2) * hi_memblkcnt2)
```

図Ｂ－８　例題システムのメモリプール領域定義ヘッダ（himempol.h）

(2)　メモリプール領域定義ソース

図Ｂ－９に例題システムのメモリプール領域定義ソース（himempol.c）を示します。

```
/************************************************************************/
/*                                                                      */
/*                                                                      */
/*            HI-SH77 define memory pool area ( Ver. 1.0 )              */
/*                                                                      */
/*                                                                      */
/*            Copyright (c)  Hitachi, Ltd. 1995.                        */
/*            Licensed Material of Hitachi, Ltd.                        */
/*                                                                      */
/*            HI-SH77(HS0770ITCN1SM) V1.0                               */
/*                                                                      */
/*                                                                      */
/************************************************************************/
/************************************************************************/
/*SPECIFICATIONS ;                                                      */
/*  FILE      = himempol.c ;                                            */
/*  DATE      = 95/02/20 ;                                              */
/*  AUTHOR    = Hitachi, Ltd. ;                                         */
/*  FUNCTION  = HI-SH77 define memory pool area ;                       */
/*  ATTRIBUTE = PUBLIC ;                                                */
/*  HISTORY   = V1.0 ;                                                  */
/*END OF SPECIFICATIONS ;                                               */
/************************************************************************/
#include    "itron.h"
#include    "himempol.h"

/************************************************************************/
/*                                                                      */
/*          Definition memory pool area                                 */
/*                                                                      */
/************************************************************************/
VW  hi_mempol1[(hi_mempolsz1) / sizeof(VW)];    /* mplid = 1            */←メモリプール領域の定義
VW  hi_mempol2[(hi_mempolsz2) / sizeof(VW)];    /* mplid = 2            */
```

図Ｂ－９　例題システムのメモリプール領域定義ソース（himempol.c）

Ｂ．１０　例題システムのトレースバッファ領域定義プログラム

(1)　トレースバッファ領域定義ヘッダ

図Ｂ－１０に例題システムのトレースバッファ領域定義ヘッダ（hitrcbuf.h）を示します。

```
/************************************************************************/
/*                                                                      */
/*                                                                      */
/*            HI-SH77 header file for trace buffer ( Ver. 1.0 )         */
/*                                                                      */
/*                                                                      */
/*            Copyright (c)  Hitachi, Ltd. 1995.                        */
/*            Licensed Material of Hitachi, Ltd.                        */
/*                                                                      */
/*            HI-SH77(HS0770ITCN1SM) V1.0                               */
/*                                                                      */
/*                                                                      */
/************************************************************************/
/************************************************************************/
/*SPECIFICATIONS ;                                                      */
/*  FILE      = hitrcbuf.h ;                                            */
/*  DATE      = 95/02/20 ;                                              */
/*  AUTHOR    = Hitachi, Ltd. ;                                         */
/*  FUNCTION  = HI-SH77 header file for trace buffer ;                  */
/*  ATTRIBUTE = PUBLIC ;                                                */
/*  HISTORY   = V1.0 ;                                                  */
/*END OF SPECIFICATIONS ;                                               */
/************************************************************************/
/************************************************************************/
/*                                                                      */
/*         Definition constant size of trace buffer                     */
/*                                                                      */
/************************************************************************/
#define TRCMNGSZ        0x10            /* size of trace management area    */←変更しないでください
#define TRCENTSZ        0x24            /* size of trace entry area         */←変更しないでください

/************************************************************************/
/*                                                                      */
/*         Definition trace information                                 */
/*                                                                      */
/************************************************************************/
#define hi_trcentnum    32              /* number of trace entry            */←トレース情報取得数の定義
                                        /*      range : [1...]              */
                                        /* trace buffer size                */←トレースバッファサイズの算出
#define hi_trcbufsz     (TRCMNGSZ + TRCENTSZ * hi_trcentnum)
```

図Ｂ－１０　例題システムのトレースバッファ領域定義ヘッダ（hitrcbuf.h）

(2)　トレースバッファ領域定義ソース

図Ｂ－１１に例題システムのトレースバッファ領域定義ソース（hitrcbuf.c）を示します。

```
/************************************************************************/
/*                                                                      */
/*                                                                      */
/*            HI-SH77 define trace buffer area ( Ver. 1.0 )             */
/*                                                                      */
/*                                                                      */
/*            Copyright (c)  Hitachi, Ltd. 1995.                        */
/*            Licensed Material of Hitachi, Ltd.                        */
/*                                                                      */
/*            HI-SH77(HS0770ITCN1SM) V1.0                               */
/*                                                                      */
/*                                                                      */
/************************************************************************/
/************************************************************************/
/*SPECIFICATIONS ;                                                      */
/*  FILE      = hitrcbuf.c ;                                            */
/*  DATE      = 95/02/20 ;                                              */
/*  AUTHOR    = Hitachi, Ltd. ;                                         */
/*  FUNCTION  = HI-SH77 define trace buffer area ;                      */
/*  ATTRIBUTE = PUBLIC ;                                                */
/*  HISTORY   = V1.0 ;                                                  */

/*END OF SPECIFICATIONS ;                                               */
/************************************************************************/
#include    "itron.h"
#include    "hitrcbuf.h"

/************************************************************************/
/*                                                                      */
/*          Definition trace buffer area                                */
/*                                                                      */
/************************************************************************/
VW      hi_trcbuf[(hi_trcbufsz) / sizeof(VW)];  /* trace buffer area       */←トレースバッファ領域の定義
```

図Ｂ－１１　例題システムのトレースバッファ領域定義ソース（hitrcbuf.c）

Ｂ．１１　例題システムのベクタテーブル

図Ｂ－１２に例題システムのベクタテーブル（hivcttbl.c）を示します。

```
/************************************************************************/
/*                                                                      */
/*                                                                      */
/*            HI-SH77 vector table ( Ver. 1.0 )                         */
/*                                                                      */
/*                                                                      */
/*            Copyright (c)  Hitachi, Ltd. 1995.                        */
/*            Licensed Material of Hitachi, Ltd.                        */
/*                                                                      */
/*            HI-SH77(HS0770ITCN1SM) V1.0                               */
/*                                                                      */
/*                                                                      */
/************************************************************************/
/************************************************************************/
/*SPECIFICATIONS ;                                                      */
/*  FILE      = hivcttbl.c ;                                            */
/*  NAME      = hi_vcttbl ;                                             */
/*  DATE      = 95/02/20 ;                                              */
/*  AUTHOR    = Hitachi, Ltd. ;                                         */
/*  FUNCTION  = HI-SH77 vector table ;                                  */
/*  ATTRIBUTE = PUBLIC ;                                                */
/*  HISTORY   = V1.0 ;                                                  */
/*END OF SPECIFICATIONS ;                                               */
/************************************************************************/
#include    "itron.h"

/*-------------- handler of MCU initialize routine for HI-SH77 -------------*/
extern  hi_mcuini();                    /* MCU initialize routine           */←MCU初期化ルーチン開始アドレスの参照

/*------------------- handler of timer driver for HI-SH77 ------------------*/←タイマ割込みハンドラ開始アドレスの参照
extern  hi_tmrhdr();                    /* hardware timer handler           */   （タイマドライバ）

/*------------------ handler of console driver for HI-SH77 -----------------*/←シリアル送受信割込みハンドラ開始アドレスの参照
extern  hi_cnshdrer();                  /* receive error handler            */   （コンソールドライバ）
extern  hi_cnshdrrx();                  /* receive handler                  */
extern  hi_cnshdrtx();                  /* trans handler                    */

/*---------------------- handler of undefine interrupt ---------------------*/←未定義割込み例外コード解析ルーチン
extern  hi_undefint();                                                                  開始アドレスの参照
extern  hi_undefexp();
```

図Ｂ－１２　例題システムのベクタテーブル（hivcttbl.c）（１／２）

```
/************************************************************************/
/*                                                                      */
/*          Definition vector table                                     */
/*                                                                      */
/************************************************************************/
const   VP  hi_vcttbl[] = {                                              ←ベクタテーブルの定義
        (VP)hi_mcuini,  /* power on reset                (vctno = 00) */←MCU初期化ルーチンの登録（パワーオンリセット）
        (VP)hi_mcuini,  /* manual reset                 (vctno = 01) */←MCU初期化ルーチンの登録（マニュアルリセット）
        (VP)hi_undefexp,/* TLB miss (load)              (vctno = 02) */
        (VP)hi_undefexp,/* TLB miss (store)             (vctno = 03) */
        (VP)hi_undefexp,/* illegal page write           (vctno = 04) */
        (VP)hi_undefexp,/* TLB protection violation(load)  (vctno = 05) */
        (VP)hi_undefexp,/* TLB protection violation(store) (vctno = 06) */
        (VP)hi_undefexp,/* address error                (vctno = 07) */
        (VP)hi_undefexp,/* address error                (vctno = 08) */
        (VP)hi_undefexp,/* reserve                      (vctno = 09) */
        (VP)hi_undefexp,/* reserve                      (vctno = 0A) */
        (VP)hi_undefexp,/* unconditional trap (TRAPA)   (vctno = 0B) */
        (VP)hi_undefexp,/* reserved instruction         (vctno = 0C) */
        (VP)hi_undefexp,/* illegal slot instruction     (vctno = 0D) */
        (VP)hi_undefint,/* NMI                          (vctno = 0E) */
        (VP)hi_undefexp,/* user breakpoint trap         (vctno = 0F) */
        (VP)hi_undefint,/* IRL15                        (vctno = 10) */
        (VP)hi_undefint,/* IRL14                        (vctno = 11) */
        (VP)hi_undefint,/* IRL13                        (vctno = 12) */
        (VP)hi_undefint,/* IRL12                        (vctno = 13) */
        (VP)hi_undefint,/* IRL11                        (vctno = 14) */
        (VP)hi_undefint,/* IRL10                        (vctno = 15) */
        (VP)hi_undefint,/* IRL9                         (vctno = 16) */
        (VP)hi_undefint,/* IRL8                         (vctno = 17) */
        (VP)hi_undefint,/* IRL7                         (vctno = 18) */
        (VP)hi_undefint,/* IRL6                         (vctno = 19) */
        (VP)hi_undefint,/* IRL5                         (vctno = 1A) */
        (VP)hi_undefint,/* IRL4                         (vctno = 1B) */
        (VP)hi_undefint,/* IRL3                         (vctno = 1C) */
        (VP)hi_undefint,/* IRL2                         (vctno = 1D) */
        (VP)hi_undefint,/* IRL1                         (vctno = 1E) */
        (VP)hi_undefint,/* IRL0                         (vctno = 1F) */
        (VP)hi_tmrhdr,  /* TMU0 TUNI0 (use timer driver) (vctno = 20) */←タイマ割込みハンドラの登録
        (VP)hi_undefint,/* TMU1 TUNI1                   (vctno = 21) */
        (VP)hi_undefint,/* TMU2 TUNI2                   (vctno = 22) */
        (VP)hi_undefint,/* TMU2 TICPI2                  (vctno = 23) */
        (VP)hi_undefint,/* RTC AT1                      (vctno = 24) */
        (VP)hi_undefint,/* RTC PRI                      (vctno = 25) */
        (VP)hi_undefint,/* RTC CUI                      (vctno = 26) */
        (VP)hi_cnshdrer,/* SCI ERI (use console driver) (vctno = 27) */←シリアル送受信割込みハンドラの登録
        (VP)hi_cnshdrrx,/* SCI RXI (use console driver) (vctno = 28) */
        (VP)hi_cnshdrtx,/* SCI TXI (use console driver) (vctno = 29) */
        (VP)hi_undefint,/* SCI TEI                      (vctno = 2A) */
        (VP)hi_undefint,/* WDT ITI                      (vctno = 2B) */
        (VP)hi_undefint,/* REF RCMI                     (vctno = 2C) */
        (VP)hi_undefint,/* REF ROVI                     (vctno = 2D) */
        (VP)hi_undefint,/* reserve                      (vctno = 2E) */
        (VP)hi_undefint,/* reserve                      (vctno = 2F) */
        (VP)hi_undefint,/* reserve                      (vctno = 30) */
        (VP)hi_undefint,/* reserve                      (vctno = 31) */
        (VP)hi_undefint,/* reserve                      (vctno = 32) */
        (VP)hi_undefint,/* reserve                      (vctno = 33) */
        (VP)hi_undefint,/* reserve                      (vctno = 34) */
        (VP)hi_undefint,/* reserve                      (vctno = 35) */
        (VP)hi_undefint,/* reserve                      (vctno = 36) */

                                    中略

        (VP)hi_undefint,/* reserve                      (vctno = 7F) */
    };
```

図Ｂ－１２　例題システムのベクタテーブル（hivcttbl.c）（２／２）

Ｂ．１２　例題システムのトラップベクタテーブル

図Ｂ－１３に例題システムのトラップベクタテーブル（hitrptbl.c）を示します。

```
/************************************************************************/
/*                                                                      */
/*                                                                      */
/*            HI-SH77 trap vector table ( Ver. 1.0 )                    */
/*                                                                      */
/*                                                                      */
/*            Copyright (c)  Hitachi, Ltd. 1995.                        */
/*            Licensed Material of Hitachi, Ltd.                        */
/*                                                                      */
/*            HI-SH77(HS0770ITCN1SM) V1.0                               */
/*                                                                      */
/*                                                                      */
/************************************************************************/
/************************************************************************/
/*SPECIFICATIONS ;                                                      */
/*  FILE      = hitrptbl.c ;                                            */
/*  NAME      = hi_trptbl ;                                             */
/*  DATE      = 95/02/20 ;                                              */
/*  AUTHOR    = Hitachi, Ltd. ;                                         */
/*  FUNCTION  = HI-SH77 trap vector table ;                             */
/*  ATTRIBUTE = PUBLIC ;                                                */
/*  HISTORY   = V1.0 ;                                                  */
/*END OF SPECIFICATIONS ;                                               */
/************************************************************************/
#include    "itron.h"

/*---------------------- handler of kernel for HI-SH77 ----------------------*/←カーネル用SVCハンドラ開始アドレスの参照
extern  _OHret_exc();                   /* ret_exc    SVC handler          */←ret_excシステムコール開始アドレスの参照
extern  _OHsys_clk();                   /* sys_clk    SVC handler          */←sys_clkシステムコール開始アドレスの参照
extern  _OHret_int();                   /* ret_int    SVC handler          */←ret_intシステムコール開始アドレスの参照
extern  _OHenn_svc();                   /* interrupt SVC handler           */←変更しないでください
extern  _OHent_svc();                   /* task       SVC handler          */←変更しないでください

/*---------------------- handler of undefine trap --------------------------*/
extern  hi_undeftrp();
```

図Ｂ－１３　例題システムのトラップベクタテーブル（hitrptbl.c）（１／２）

```
/************************************************************************/
/*                                                                      */
/*          Definition trap vector table                                */
/*                                                                      */
/************************************************************************/
const   VP  hi_trptbl[] = {
            (VP)hi_undeftrp,/* trapa #00                    (vctno = 00) */
            (VP)hi_undeftrp,/* trapa #01                    (vctno = 01) */
            (VP)hi_undeftrp,/* trapa #02                    (vctno = 02) */
            (VP)hi_undeftrp,/* trapa #03                    (vctno = 03) */
            (VP)hi_undeftrp,/* trapa #04                    (vctno = 04) */
            (VP)hi_undeftrp,/* trapa #05                    (vctno = 05) */
            (VP)hi_undeftrp,/* trapa #06                    (vctno = 06) */
            (VP)hi_undeftrp,/* trapa #07                    (vctno = 07) */
            (VP)hi_undeftrp,/* trapa #08                    (vctno = 08) */
            (VP)hi_undeftrp,/* trapa #09                    (vctno = 09) */
            (VP)hi_undeftrp,/* trapa #10                    (vctno = 0A) */
            (VP)hi_undeftrp,/* trapa #11                    (vctno = 0B) */
            (VP)hi_undeftrp,/* trapa #12                    (vctno = 0C) */
            (VP)hi_undeftrp,/* trapa #13                    (vctno = 0D) */
            (VP)hi_undeftrp,/* trapa #14                    (vctno = 0E) */
            (VP)hi_undeftrp,/* trapa #15                    (vctno = 0F) */

                                    中略

            (VP)hi_undeftrp,/* trapa #50                    (vctno = 32) */
            (VP)hi_undeftrp,/* trapa #51                    (vctno = 33) */
            (VP)hi_undeftrp,/* trapa #52                    (vctno = 34) */
            (VP)hi_undeftrp,/* trapa #53                    (vctno = 35) */
            (VP)hi_undeftrp,/* trapa #54                    (vctno = 36) */
            (VP)hi_undeftrp,/* trapa #55                    (vctno = 37) */
            (VP)hi_undeftrp,/* trapa #56                    (vctno = 38) */
            (VP)_OHret_exc, /* trapa #57 (use HI-SH77)      (vctno = 39) */←ret_excシステムコールの登録
            (VP)hi_undeftrp,/* trapa #58 (reserve HI-SH77)  (vctno = 3A) */←このベクタは使用しないでください
            (VP)hi_undeftrp,/* trapa #59 (reserve HI-SH77)  (vctno = 3B) */←このベクタは使用しないでください
            (VP)_OHsys_clk, /* trapa #60 (use HI-SH77)      (vctno = 3C) */←sys_clkシステムコールの登録
            (VP)_OHret_int, /* trapa #61 (use HI-SH77)      (vctno = 3D) */←ret_intシステムコールの登録
            (VP)_OHenn_svc, /* trapa #62 (use HI-SH77)      (vctno = 3E) */←変更しないでください
            (VP)_OHent_svc, /* trapa #63 (use HI-SH77)      (vctno = 3F) */←変更しないでください
            (VP)hi_undeftrp,/* trapa #64                    (vctno = 40) */
            (VP)hi_undeftrp,/* trapa #65                    (vctno = 41) */
            (VP)hi_undeftrp,/* trapa #66                    (vctno = 42) */
            (VP)hi_undeftrp,/* trapa #67                    (vctno = 43) */
            (VP)hi_undeftrp,/* trapa #68                    (vctno = 44) */
            (VP)hi_undeftrp,/* trapa #69                    (vctno = 45) */

                                    中略

            (VP)hi_undeftrp,/* trapa #240                   (vctno = F0) */
            (VP)hi_undeftrp,/* trapa #241                   (vctno = F1) */
            (VP)hi_undeftrp,/* trapa #242                   (vctno = F2) */
            (VP)hi_undeftrp,/* trapa #243                   (vctno = F3) */
            (VP)hi_undeftrp,/* trapa #244                   (vctno = F4) */
            (VP)hi_undeftrp,/* trapa #245                   (vctno = F5) */
            (VP)hi_undeftrp,/* trapa #246                   (vctno = F6) */
            (VP)hi_undeftrp,/* trapa #247                   (vctno = F7) */
            (VP)hi_undeftrp,/* trapa #248                   (vctno = F8) */
            (VP)hi_undeftrp,/* trapa #249                   (vctno = F9) */
            (VP)hi_undeftrp,/* trapa #250                   (vctno = FA) */
            (VP)hi_undeftrp,/* trapa #251                   (vctno = FB) */
            (VP)hi_undeftrp,/* trapa #252                   (vctno = FC) */
            (VP)hi_undeftrp,/* trapa #253                   (vctno = FD) */
            (VP)hi_undeftrp,/* trapa #254                   (vctno = FE) */
            (VP)hi_undeftrp,/* trapa #255                   (vctno = FF) */
        };
```

図Ｂ－１３　例題システムのトラップベクタテーブル（hitrptbl.c）（２／２）

Ｂ．１３　例題システムのMCU初期化ルーチン

図Ｂ－１４に例題システムのMCU初期化ルーチン（himcuini.c）を示します。

```
/************************************************************************/
/*                                                                      */
/*                                                                      */
/*            HI-SH77 MCU initialize routine ( Ver. 1.0 )               */
/*                                                                      */
/*                                                                      */
/*            Copyright (c)  Hitachi, Ltd. 1995.                        */
/*            Licensed Material of Hitachi, Ltd.                        */
/*                                                                      */
/*            HI-SH77(HS0770ITCN1SM) V1.0                               */
/*                                                                      */
/*                                                                      */
/************************************************************************/
/************************************************************************/
/*SPECIFICATIONS ;                                                      */
/*  FILE      = himcuini.c ;                                            */
/*  NAME      = hi_mcuini ;                                             */
/*  DATE      = 95/02/20 ;                                              */
/*  AUTHOR    = Hitachi, Ltd. ;                                         */
/*  FUNCTION  = HI-SH77 MCU initialize routine ;                        */
/*  ATTRIBUTE = PUBLIC ;                                                */
/*  HISTORY   = V1.0 ;                                                  */
/*END OF SPECIFICATIONS ;                                               */
/************************************************************************/
#include    <umachine.h>
#include    <smachine.h>
#include    "itron.h"

#define IOBASE   0xfffffe80             /* I/O base address    = 0xfffffe80 */←GBRに設定するI/Oベースアドレス
#define BCR1     (0xffffff60 - IOBASE)  /* BCN BCR1         address offset */←BSCレスタのI/Oベースアドレス
#define BCR2     (0xffffff62 - IOBASE)  /* BCN BCR2         address offset */                からのオフセット
#define WCR1     (0xffffff64 - IOBASE)  /* BCN WCR1         address offset */
#define WCR2     (0xffffff66 - IOBASE)  /* BCN WCR2         address offset */
#define MCR      (0xffffff68 - IOBASE)  /* BCN MCR          address offset */
#define RTCSR    (0xffffff6e - IOBASE)  /* BCN RTCSR        address offset */
#define RTCNT    (0xffffff70 - IOBASE)  /* BCN RTCNT        address offset */
#define RTCOR    (0xffffff72 - IOBASE)  /* BCN RTCOR        address offset */
#define RFCR     (0xffffff74 - IOBASE)  /* BCN RFCR         address offset */
#define MMUCR    (0xfffffffe0 - IOBASE) /* CCN MMUCR        address offset */←MMUCRのI/Oベースアドレスからのオフセット
#define CCR      (0xfffffffec - IOBASE) /* CCN CCR          address offset */←CCRのI/Oベースアドレスからのオフセット

#define BCR1DAT  (UH)0x0000             /* bus control data                */←BSCレスタの設定データ
#define BCR2DAT  (UH)0x2aa8             /* bus control data                */
#define WCR1DAT  (UH)0x3fff             /* wait control data               */
#define WCR2DAT  (UH)0xffff             /* wait control data               */
#define MCRDAT   (UH)0x0000             /* memory control data             */
#define RTCSRDAT (UH)0xa500             /* refresh timer control/status data*/
#define RTCNTDAT (UH)0xa500             /* refresh timer counter data      */
#define RTCORDAT (UH)0xa500             /* refresh time constant data      *
#define RFCRDAT  (UH)0xa400             /* refresh count data              */

#define MMU_OFF   (UW)0x00000000        /* MMU   disable data              */←MMU無効化設定値
#define CACHE_ON  (UW)0x00000001        /* CACHE enable  data              */←キャッシュ有効化設定値
#define CACHE_OFF (UW)0x00000000        /* CACHE disable data              */←キャッシュ無効化設定値

#define RAMSTA   (VW *)0xac000000       /* RAM start address   = 0xac000000 */←RAMクリア領域の設定
#define RAMEND   (VW *)0xac07fffc       /* RAM end address     = 0xac07fffc */
```

図Ｂ－１４　例題システムのMCU初期化ルーチン（himcuini.c）（１／２）

```
extern  void    _0Hrs__knl(void);       /* kernel reset routine             */
/* extern void    _INITSCT(void);         section initialization           */
                                        /*                   routine for C */


void    hi_mcuini(void)
{
    register    VW      *p;             /* pointer to memory                */

    set_cr(MD_SET | (SR_IMS15 << 4));   /* clear BL, set imask              */←例外マスク解除、割込みマスク（レベル＝15）

    set_gbr((VP)IOBASE);                /* set I/O base address to GBR      */
    gbr_write_long(CCR, CACHE_OFF);     /* CACHE disable                    */←キャッシュ無効化
    gbr_write_long(MMUCR, MMU_OFF);     /* MMU disable                      */←MMU無効化
    gbr_write_word(BCR1, BCR1DAT);      /* set bus control data to BCR1     */←バスステートコントローラの設定
    gbr_write_word(BCR2, BCR2DAT);      /* set bus control data to BCR2     */
    gbr_write_word(WCR1, WCR1DAT);      /* set wait control data to WCR1    */
    gbr_write_word(WCR2, WCR2DAT);      /* set wait control data to WCR2    */
    gbr_write_word(MCR, MCRDAT);        /* set memory control data to MCR   */
    gbr_write_word(RTCSR, RTCSRDAT);    /* set refresh timer control/status */
                                        /*                   data to RTCSR */
    gbr_write_word(RTCNT, RTCNTDAT);    /* set refresh timer counter        */
                                        /*                     data to MCR */
    gbr_write_word(RTCOR, RTCORDAT);    /* set refresh time constant        */
                                        /*                   data to RTCOR */
    gbr_write_word(RFCR, RFCRDAT);      /* set refresh count data to RFCR   */

    for(p = RAMSTA; p <= RAMEND; p++)   /* RAM clear                        */←RAMクリア
        *p = 0x00000000;

/*  _INITSCT();                         call section initialization      */
                                        /*                   routine for C */

    _0Hrs__knl();                       /* go to kernel reset routine       */←カーネル起動処理へジャンプ
}
```

図Ｂ－１４　例題システムのMCU初期化ルーチン（himcuini.c）（２／２）

Ｂ．１４　例題システムのシステム初期化ハンドラ

図Ｂ－１５に例題システムのシステム初期化ハンドラ（hisysini.c）を示します。

```
/************************************************************************/
/*                                                                      */
/*                                                                      */
/*            HI-SH77 system initial handler ( Ver. 1.0 )               */
/*                                                                      */
/*                                                                      */
/*            Copyright (c)  Hitachi, Ltd. 1995.                        */
/*            Licensed Material of Hitachi, Ltd.                        */
/*                                                                      */
/*            HI-SH77(HS0770ITCN1SM) V1.0                               */
/*                                                                      */
/*                                                                      */
/************************************************************************/
/************************************************************************/
/*SPECIFICATIONS ;                                                      */
/*  FILE      = hisysini.c ;                                            */
/*  NAME      = hi_sysini ;                                             */
/*  DATE      = 95/02/20 ;                                              */
/*  AUTHOR    = Hitachi, Ltd. ;                                         */
/*  FUNCTION  = HI-SH77 system initial handler ;                        */
/*  ATTRIBUTE = PUBLIC ;                                                */
/*  HISTORY   = V1.0 ;                                                  */
/*END OF SPECIFICATIONS ;                                               */
/************************************************************************/
#include    <umachine.h>
#include    "itron.h"

extern  void    hi_tmrini(void);        /* timer initialize routine     */
extern  void    hi_sysdwn(W, ER, UW);   /* system down routine          */

INIHDR  hi_sysini(void)
{
    ER      ercd;                       /* error code                   */

    hi_tmrini();                        /* call timer initialize routine*/←タイマの初期化

    ercd = ista_tsk((ID)1);             /* start task (tskid = 1)       */←コンソールドライバタスクの起動
    if(ercd < E_OK)                     /* if ista_tsk fail             */
        hi_sysdwn((W)1, ercd, (UW)0);   /* go to system down routine    */
}
```

図Ｂ－１５　例題システムのシステム初期化ハンドラ（hisysini.c）

Ｂ．１５　例題システムの未定義割込み異常処理ルーチン

図Ｂ－１６に例題システムの未定義割込み異常処理ルーチン（hiintdwn.c）を示します。

```
/************************************************************************/
/*                                                                      */
/*                                                                      */
/*             HI-SH77 interrupt down routine ( Ver. 1.0 )              */
/*                                                                      */
/*                                                                      */
/*             Copyright (c)  Hitachi, Ltd. 1995.                       */
/*             Licensed Material of Hitachi, Ltd.                       */
/*                                                                      */
/*             HI-SH77(HS0770ITCN1SM) V1.0                              */
/*                                                                      */
/*                                                                      */
/************************************************************************/
/************************************************************************/
/*SPECIFICATIONS ;                                                      */
/*  FILE      = hiintdwn.c ;                                            */
/*  NAME      = hi_intdwn ;                                             */
/*  DATE      = 95/02/20 ;                                              */
/*  AUTHOR    = Hitachi, Ltd. ;                                         */
/*  FUNCTION  = HI-SH77 interrupt down routine (undefine interrupt handler);*/
/*  ATTRIBUTE = PUBLIC ;                                                */
/*  HISTORY   = V1.0 ;                                                  */
/*END OF SPECIFICATIONS ;                                               */
/************************************************************************/
#include    <smachine.h>
#include    "itron.h"

void    hi_intdwn(code)
UW      code;                           /* exception code(INTEVT or EXPEVT) */←パラメータ（例外コード）
{
    set_cr(MD_SET | (SR_IMS15 << 4));   /* clear BL, set imask              */←例外マスク解除、割込みマスク（レベル＝15）

    while(TRUE);                        /* endless loop                     */←無限ループ
}
```

図Ｂ－１６　例題システムの未定義割込み異常処理ルーチン（hiintdwn.c）

Ｂ．１６　例題システムの未定義トラップ異常処理ルーチン

図Ｂ－１７に例題システムの未定義割込みトラップ異常処理ルーチン（hitrpdwn.c）を示します。

```
/************************************************************************/
/*                                                                      */
/*                                                                      */
/*             HI-SH77 trap down routine ( Ver. 1.0 )                   */
/*                                                                      */
/*                                                                      */
/*             Copyright (c)  Hitachi, Ltd. 1995.                       */
/*             Licensed Material of Hitachi, Ltd.                       */
/*                                                                      */
/*             HI-SH77(HS0770ITCN1SM) V1.0                              */
/*                                                                      */
/*                                                                      */
/************************************************************************/
/************************************************************************/
/*SPECIFICATIONS ;                                                      */
/*  FILE      = hitrpdwn.c ;                                            */
/*  NAME      = hi_trpdwn ;                                             */
/*  DATE      = 95/02/20 ;                                              */
/*  AUTHOR    = Hitachi, Ltd. ;                                         */
/*  FUNCTION  = HI-SH77 trap down routine (undefine trap handler);      */
/*  ATTRIBUTE = PUBLIC ;                                                */
/*  HISTORY   = V1.0 ;                                                  */
/*END OF SPECIFICATIONS ;                                               */
/************************************************************************/
#include    <smachine.h>
#include    "itron.h"

void    hi_trpdwn(code)
UW      code;                           /* TRAPA 8-bit immediate data * 4   */←パラメータ（TRAPA命令の8ビット
                                                                                イミディエイトデータ × 4）
{
    set_cr(MD_SET | (SR_IMS15 << 4));   /* clear BL, set imask              */←例外マスク解除、割込みマスク（レベル = 15）

    while(TRUE);                        /* endless loop                     */←無限ループ
}
```

図Ｂ－１７　例題システムの未定義トラップ異常処理ルーチン（hitrpdwn.c）

Ｂ．１７　例題システムのシステム異常終了処理ルーチン

図Ｂ－１８に例題システムのシステム異常終了処理ルーチン（hisysdwn.c）を示します。

```
/************************************************************************/
/*                                                                      */
/*                                                                      */
/*            HI-SH77 system down routine ( Ver. 1.0 )                  */
/*                                                                      */
/*                                                                      */
/*            Copyright (c)  Hitachi, Ltd. 1995.                        */
/*            Licensed Material of Hitachi, Ltd.                        */
/*                                                                      */
/*            HI-SH77(HS0770ITCN1SM) V1.0                               */
/*                                                                      */
/*                                                                      */
/************************************************************************/
/************************************************************************/
/*SPECIFICATIONS ;                                                      */
/*  FILE      = hisysdwn.c ;                                            */
/*  NAME      = hi_sysdwn ;                                             */
/*  DATE      = 95/02/20 ;                                              */
/*  AUTHOR    = Hitachi, Ltd. ;                                         */
/*  FUNCTION  = HI-SH77 system down routine ;                           */
/*  ATTRIBUTE = PUBLIC ;                                                */
/*  HISTORY   = V1.0 ;                                                  */
/*END OF SPECIFICATIONS ;                                               */
/************************************************************************/
#include    <smachine.h>
#include    "itron.h"

void        hi_sysdwn(type, ercd, inf)
W           type;               /* type of system down                  */←パラメータ（システムダウン種別）
                                /*      type >=  1 : system down of user program */
                                /*      type ==  0 : setup table error  */
                                /*      type == -1 : context error of ext_tsk */
                                /*      type == -2 : context error of exd_tsk */
                                /*      type == -3 : context error of ret_int */
                                /*      type == -4 : context error of sys_clk */
                                /*      type <= -5 : system reserve     */
ER          ercd;               /* error code of system down            */←パラメータ（エラーコード）
                                /*      type ==  0 : error code of setup table */
                                /*      type == -1 : error code of ext_tsk */
                                /*      type == -2 : error code of exd_tsk */
                                /*      type == -3 : error code of ret_int */
                                /*      type == -4 : error code of sys_clk */
UW          inf;                /* information of system down           */←パラメータ（システムダウン情報）
                                /*      type == -1 : address of ext_tsk call */
                                /*      type == -2 : address of exd_tsk call */
                                /*      type == -3 : address of ret_int call */
                                /*      type == -4 : address of sys_clk call */
{
      set_cr(MD_SET | (SR_IMS15 << 4));       /* clear BL, set imask    */←例外マスク解除、割込みマスク（レベル
= 15）

      while(TRUE);              /* endless loop                         */←無限ループ
}
```

図Ｂ－１８　例題システムのシステム異常終了処理ルーチン（hisysdwn.c）

B．１８　例題システムの構築

例題システムの構築は、提供しているシステム構築用バッチファイル（**himake.bat**）を実行します。
バッチファイルの実行は、以下のコマンドを入力します。

```
himake (RET)
```

なお、例題システムのメモリマップは、提供しているシステム構築用リンケージサブコマンドファイル（**himake.sub**）に記述しています。

B．１９　例題システムの起動

E7000PC（インサーキットエミュレータ）にロードモジュールをダウンロードする方法を説明します。

(1)　ダウンロード
　(a)**E7000PC**システムを起動します。
　(b)**PC**インタフェースソフトウェアを起動します。

```
IPI(RET)
        （PCインタフェースソフトウェア起動メッセージ）
        （E7000PC起動メッセージ）
:       （E7000プロンプト）
```

　(c)**E7000**の**MAP**コマンドでエミュレーションメモリを割り当てます。以下の例では**H'0**番地から**H'7FFFF**番地および**H'C000000**番地から**H'C07FFFF**番地にエミュレーションメモリを割り当てています。なお、**H'0**番地から**H'7FFFF**番地はエミュレーションメモリを書き込み禁止として割り当てています。

```
: MAP△0△7FFFF;SW(RET)
: MAP△C000000△C07FFFF;S(RET)
```

　(d)　**E7000**の**LOAD**コマンドを使用し、**IBM PC**からロードモジュールをダウンロードします。

```
: LOAD△;R:<アブソリュートロードモジュールのファイル名>(RET)
```

(2)　システムの起動
　ロードモジュールの開始アドレスから実行すると、システムが起動します。
　開始アドレスには、カーネルのリセットサービスルーチンの先頭アドレス（**__0Hrst_srv**）を指定します。なお、**E7000**の**RESET**コマンドを入力すると、開始アドレスは自動的に設定されます。

```
: reset(RET)
: go(RET)
```

付録C．ＡＳＣＩＩコード表

C．１　ＡＳＣＩＩコード

上位4ビット / 下位4ビット	0	1	2	3	4	5	6	7
0	NULL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	＼	l	¦
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

D．索引

D．1　五十音順・索引

五十音順・索引

タ 行

ハ 行

マ 行

ヤ 行

ラ 行

ワ 行

アルファベット順・索引

H

I

K

M

R

S

T

HI-SH77　構築マニュアル
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

ADJ-702-200