

SH1、SH2、SH3対応リモートデバッガ

# Visual Monitor/SH

*Ver2.6*

*User's Manual*

14版 2003. 9. 1

**ALPHA PROJECT Co., LTD**

## Visual Monitor / SH User's Manual

この度は、弊社製品「Visual Monitor / SH Ver.2」をご購入いただき、誠にありがとうございます。

本製品は日立RISCマイコン SHシリーズ用に開発されたリモートデバッガです。

本ソフトウェアを有用なツールとしてお使いいただくために本マニュアルを十分お読みくださいますようお願いいたします。

★本製品の内容及び仕様は予告なしに変更されることがありますのでご了承ください。

★本ソフトウェア及び弊社製品についてのお問い合わせはFAXもしくはE-mailにてお願いいたします。

★本マニュアルに記載される会社名、商品名は各社の商標及び登録商標です。

### お使いになる前に

#### <梱包の確認>

次のものが揃っているかご確認ください。

万一足りないものがあつた場合には、販売店までご連絡ください。

\* インストールCD

- ・プログラム本体
- ・ユーザーズマニュアル
- ・ソフトウェア使用許諾書
- ・FAXサポート依頼書
- ・メンテナンスのご案内

\* ユーザ登録カード

### お問い合わせ先

株式会社 アルファプロジェクト  
〒431-3114 静岡県浜松市東区積志町 834

Fax (053) 401-0035 技術部 担当者宛

E-mail query@apnet.co.jp

## 目次

<b>第1章 概要</b>	<b>1</b>
1. 1 特徴	1
1. 2 使用環境	1
1. 3 構成	2
1. 4 機能概要	3
1. 5 対応チップ	4
<b>第2章 インストール</b>	<b>5</b>
2. 1 準備するもの	5
2. 2 インストール方法	6
2. 2. 1 インストール手順	6
2. 2. 2 ディレクトリ構成	7
<b>第3章 ターゲットモニタソフトの作成</b>	<b>10</b>
3. 1 カスタマイズの手順	10
3. 2 VMeditの使用法	11
3. 2. 1 ターゲットCPUの設定	11
3. 2. 2 アドレス設定	13
3. 2. 3 コンパイラパス設定	17
3. 2. 4 共通操作	18
3. 3 初期化処理のカスタマイズ	19
3. 3. 1 初期化処理の記述	19
3. 4 モニタの生成	20
3. 5 モニタROMの準備	21
3. 6 モニタROMの動作確認	21
3. 7 アルファボードシリーズでの使用について	21
3. 7. 1 カスタマイズ済みターゲットモニタ	21

**第4章 ユーザプログラムの作成****22**

4. 1	デバッグに必要なファイル	22
4. 2	ユーザプログラムの作成上の注意	22
4. 2. 1	ユーザプログラムの制限事項	22
4. 3	サンプルプログラム	23
4. 3. 1	サンプルプログラムの構成	23
4. 4	ベクタテーブル	24
4. 4. 1	ベクタテーブルの配置と構造	24
4. 4. 2	ベクタテーブルの登録方法 (SH1, SH2)	25
4. 4. 3	VisualMonitor が使用する割り込み (SH1, SH2)	26
4. 4. 4	SH3の割り込みについて	31
4. 4. 5	ベクタテーブルの登録方法 (SH3)	35
4. 4. 6	VisualMonitor が使用する割り込み (SH3)	37

**第5章 デバッグ時の使用ファイル****38**

5. 1	フォルダ構成	38
5. 2	デバッグ用ファイル	38
5. 3	コンフィグレーションファイル (*.CFG)	39
5. 3. 1	コンフィグレーションファイルウィンドウ	39
5. 3. 2	コンフィグレーションファイルの変更	40

**第6章 チュートリアル****41**

6. 1	基本操作手順	41
6. 2	ROM上のプログラムのデバッグ	46

**第7章 メニューとウィンドウの操作****47**

7. 1	ウィンドウ構成	47
7. 2	メニュー	48
7. 2. 1	ファイルメニュー	49
7. 2. 2	設定メニュー	49

7. 2. 3	デバッグメニュー	4 9
7. 2. 4	ローカルメニュー	4 9
7. 2. 5	表示メニュー	4 9
7. 2. 6	ウィンドウメニュー	5 0
7. 2. 7	システムメニュー	5 0
7. 3	ツールバー	5 1
7. 4	ステータスバー	5 1
7. 5	確認ダイアログ	5 1
7. 6	シンボル入力	5 2
7. 7	システム設定	5 3
7. 7. 1	色設定	5 3
7. 7. 2	ポート設定	5 4
7. 7. 3	コンパイラ設定	5 4
7. 7. 4	CPU設定	5 5
7. 8	プログラムのダウンロード	5 6
7. 9	フラッシュROMのクリア	5 7
7. 10	プログラムの実行と停止	5 8
7. 10. 1	実行	5 8
7. 10. 2	停止	5 8
7. 10. 3	ステップ実行	5 8
7. 10. 4	RESET	5 9
7. 10. 5	強制停止	5 9
7. 11	メモリの設定	6 0
7. 12	設定値の保存と読み込み	6 1
7. 12. 1	設定値の保存	6 1
7. 12. 2	設定値の読み込み	6 2
7. 13	ツリービューウィンドウ	6 3
7. 13. 1	ツリービューウィンドウの機能	6 3
7. 14	ソースウィンドウ	6 4
7. 14. 1	ソースウィンドウの機能	6 4
7. 14. 2	ブレークポイントの設定方法	6 5
7. 14. 3	ダイレクト変数表示機能	6 6

7. 15	レジスタウィンドウ	67
7. 15. 1	レジスタビューウィンドウの機能	67
7. 16	ブレークポイントウィンドウ	68
7. 16. 1	ブレークポイントウィンドウの機能	68
7. 16. 2	詳細なブレーク条件の設定	69
7. 16. 3	ブレークポイントの注意と制限	70
7. 17	ウォッチウィンドウ	71
7. 17. 1	ウォッチウィンドウの機能	71
7. 17. 2	ウォッチ変数の値変更	72
7. 18	メモリウィンドウ	73
7. 18. 1	メモリウィンドウの機能	73
7. 18. 2	メモリイメージの編集	74
7. 19	スタックウィンドウ	75
7. 19. 1	スタックウィンドウの機能	75
7. 20	ユーザログウィンドウ	76
7. 20. 1	ユーザログウィンドウの機能	76
7. 20. 2	ユーザログ出力関数	76
7. 20. 3	コーディング例	77
7. 21	ログウィンドウ	78
7. 21. 1	ログウィンドウの機能	78
7. 22	REAL i モニタウィンドウ	79
7. 22. 1	REAL i モニタウィンドウの機能	79

## 第8章. その他の機能

80

8. 1	自動ブート機能	80
------	---------	----

## 第9章. ハードウェアの注意事項と設計例

81

9. 1	ターゲットシステムの構成	81
9. 1. 1	CPUクロック	81
9. 1. 2	シリアルインターフェース	81
9. 1. 3	ウォッチドッグタイマ	81
9. 1. 4	フラッシュROM	82

<b>第10章. ROM化の方法</b>	<b>83</b>
----------------------	-----------

10.1 ユーザプログラムのROM化	83
10.1.1 ROM化用ファイル	83
10.1.2 ROM化プログラムの構造	84
10.1.3 ROM化の手順	85

<b>第11章. gccコンパイラ</b>	<b>92</b>
-----------------------	-----------

<b>第12章. Q&amp;A</b>	<b>93</b>
----------------------	-----------

<b>第13章. サポート</b>	<b>96</b>
-------------------	-----------

13.1 ユーザ登録	96
13.2 バージョンアップ	96
13.3 質問の受け付け	96

<b>第14章. 付録</b>	<b>97</b>
-----------------	-----------

14.1 アルファボードシリーズの仕様	97
---------------------	----

<b>第15章. 製品サポートと使用上の注意</b>	<b>107</b>
----------------------------	------------

15.1 製品サポートのご案内	107
15.1.1 弊社ホームページのご利用について	107
15.1.2 製品サポートの方法	107
15.1.3 製品サポートの範囲	107
15.2 使用上の注意	107

## 第1章. 概要

本製品は、日立SHシリーズ用に開発されたWindows 95/98/NT/2000対応リモートデバッガです。  
本製品は、従来のモニタプログラムにはない高機能により、お客様のアプリケーション開発を強力に支援します。

### 1. 1 特長

VisualMonitor Ver. 2には以下の特長があります。

- |                   |   |
|-------------------|---|
| ●高機能              | 従来のリモートモニタにはない高機能を実現。                         |
| ●カスタマイズが簡単        | 付属の専用ツールを使用すればカスタマイズが簡単です。                    |
| ●ROMプログラムのデバッグが可能 | ROM上のプログラムのデバッグが可能です。                         |
| ●フィールドデバッグが可能     | ターゲットに組み込むことでフィールドデバッグが可能です。                  |
| ●接続が容易            | 既存のハードウェアに手を加えることなくデバッグが可能です。                 |
| ●リアルタイムOS対応       | μITRON仕様準拠の弊社製リアルタイムOS「REALi」に対応します。          |
| ●F-ZTAT対応         | F-ZTATオンボード書き込みに対応します。(注1)                    |
| ●日立C・gcc両対応       | 日立C (SYSROF,ELF) とGCC (COFF) の両方に対応しています。(注2) |

注1) 弊社製FLASH WRITERもしくは日立純正の書き込みツールが別途必要です。

注2) GCCのELF形式には対応していません。

### 1. 2 使用環境

	使用機器等	環 境
ホ ス ト	パーソナルコンピュータ	PC/AT 互換機
	OS	Windows 95/98/NT4.0/2000
	メモリ	32Mバイト以上を推奨
	ハードディスク	15Mバイト以上の空き領域
	表示	800×600 以上
	CDドライブ	CD-ROM読み込み可能なドライブ
	その他	シリアルポート 1CH (COM1~COM4)
タ ー ゲ ッ ト	ターゲットボード	日立 SH1、SH2、SH3 CPU (後述を参照)
	CPUクロック	任意
	モニタコードサイズ	32Kバイト以下(SH1/SH2) 48Kバイト以下(SH3)
	モニタ使用RAMサイズ	約4Kバイト
	FROM	MBM29F800T/B, MBM29LV320B (富士通) 互換品, F-ZTAT
	シリアルポート	RS232Cレベル (ドライバ用)、TxD, RxD, GNDの3線接続
他	RS232Cケーブル	クロスケーブルを使用



### 1. 3 構成

VisualMonitorは以下の2つのソフトウェアから構成されます。

- ターゲットモニタ

ターゲットのハードウェアに搭載するソフトウェアです。

各ターゲットの構成に合わせてカスタマイズし、ROM化してターゲットに搭載します。

- PCコントロールソフト

パソコン上で動作するコントロールソフトです。

このコントロールソフト上でダウンロードや実行、停止などの実際のデバッグ作業をおこないます。

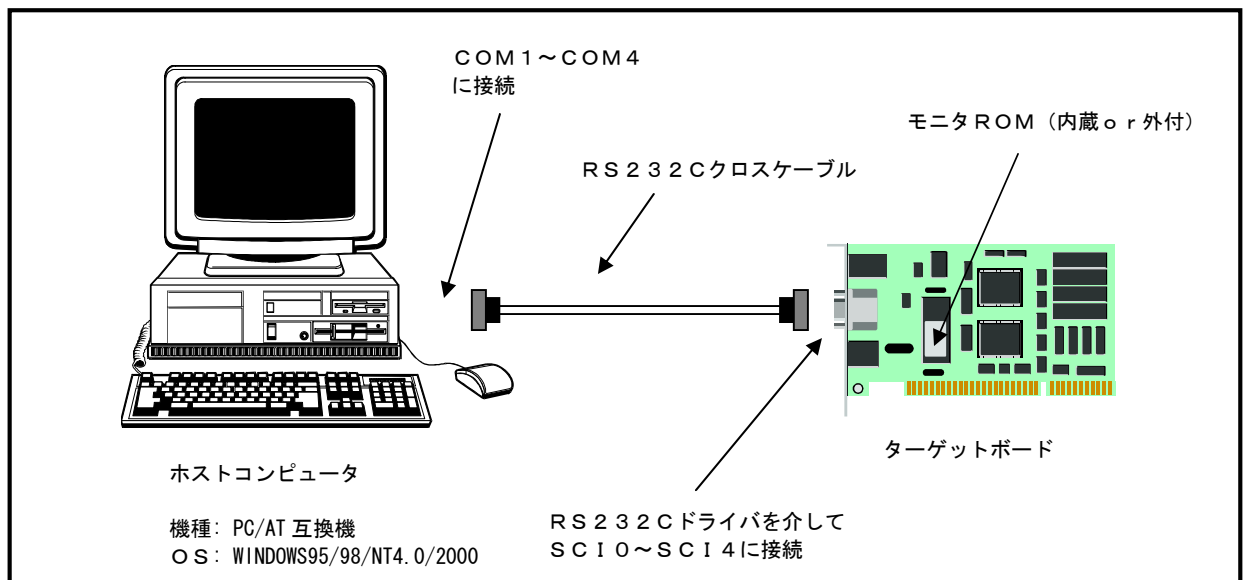


図 1. 3 VisualMonitor を使用したデバッグ環境の構成例

## 1. 4 機能概要

以下にVisualMonitorのデバッグ機能一覧を示します。

機能名称	機能内容	操作 WINDOW
プログラムダウンロード	ユーザプログラムをターゲットのRAMもしくはフラッシュROMにダウンロードします。(F-ZTAT対応)	メイン
フラッシュROMの消去	フラッシュROMの消去を行います。	メイン
ソースリスト表示	プログラムをC/アセンブラ/C+アセンブラ混在で表示します。	ソース
ツリービューの表示	ソースファイルや関数の情報をツリー表示します。	ツリービュー
設定値の読み込み/保存	ブレークポイント等の設定値を保存、読み込みします。	メイン
ユーザプログラムの実行/停止	ユーザプログラムの実行/停止を行います。	メイン
ステップ実行	ユーザプログラムのステートメントをステップ実行させます。	メイン
ソフトウェアリセット	プログラムカウンタをユーザプログラムの先頭にします。	メイン
ブレークポイント設定	ブレークポイント、ブレーク条件を指定して任意の点でプログラムを停止させます。ソース上とシンボルで指定可能です。	ソース ブレーク
レジスタ表示	レジスタの内容を表示します。	レジスタ
レジスタ書き込み	レジスタを任意の値に書き換えます。	レジスタ
メモリ編集	ターゲット上のメモリ内容を表示、編集します。サイズ、表示形式、シンボル指定が可能です。	メモリ
メモリ設定	ターゲット上のメモリ内容を任意の値に書き換えます。(FILL) アドレス指定とシンボル指定可能です。	メモリダイアログ
スタック表示	ターゲット上のスタック内容を表示します。	スタック
ウォッチ	指定された特定のメモリ領域をウォッチします。アドレス指定とシンボル指定可能です。	ウォッチ
REALiモニタ	REALi/SHの実行状態を表示します。	REALi モニタ
ユーザログ表示	ユーザプログラムからのログ出力を表示します。	ユーザログ
モニタログ表示	操作及びターゲット間の通信のログを表示します。	モニタログ
自動ブート機能	リセット時にデバッグホストが接続されていない場合は、一定時間経過後、自動的にユーザプログラムを実行します。	

## 1. 5 対応チップ

## SH 1

シリーズ名	動作確認済みチップ	弊社 CPU ボード	互換品	対応
SH7020シリーズ	-		SH7020	×
			SH7021	×
SH7030シリーズ	SH7032	AP-SH-0A	SH7032	○
			SH7034	○

## SH 2

シリーズ名	動作確認済みチップ	弊社 CPU ボード	互換品	対応
SH7604シリーズ	SH7604		SH7604	○ *1
SH7040シリーズ	SH7043	AP-SH2-0A	SH7043	○
			SH7040	○
			SH7041	○
			SH7042	○
	SH7044F SH7045F	AP-SH2F-2A、SF-7044F AP-SH2F-0A	SH7044F SH7045F	○ ○
SH7050シリーズ	SH7046F SH7047F	AP-SH2F-4A AP-SH2F-5A	SH7046F SH7047F	○ ○
	SH7050F SH7051F	AP-SH2F-1A	SH7050F	○
			SH7051F	○
	SH7055F		SH7052F	○
SH7053F			○	
SH7054F			○	
SH7065シリーズ	SH7065F	AP-SH2F-3A	SH7065F	○
SH7144シリーズ	SH7144F	AP-SH2F-7A、SF-7144F	SH7144F	○
	SH7145F	AP-SH2F-6A	SH7145F	○
SH7010シリーズ	-		SH7014	×
			SH7017F	×

\*1 ビッグエンディアンのみ対応      × 対応予定なし

## SH 3

シリーズ名	動作確認済みチップ	弊社 CPU ボード	互換品	対応
SH7700シリーズ	SH7709	AP-SH3-0A	SH7709	○
	SH7709A	AP-SH3-1A	SH7709A	○
	SH7709S	AP-SH3-2A	SH7709S	○
	SH7718R		SH7718R	○
			SH7708S	○
			SH7708R	○
	SH7729	AP-SH3D-0A	SH7729	○
SH7729R	AP-SH3D-1A	SH7729R	○	

## 第2章. インストール

### 2. 1 準備するもの

インストールを始める前に以下のものを準備して下さい。

① パーソナルコンピュータ

前項の使用環境を御覧下さい。

② ターゲット

前項の使用環境を御覧下さい。

③ インストールCD

VisualMonitor のインストールCD

④ ROM

モニタ用のPROMです。ターゲットにあわせたPROMを用意して下さい。

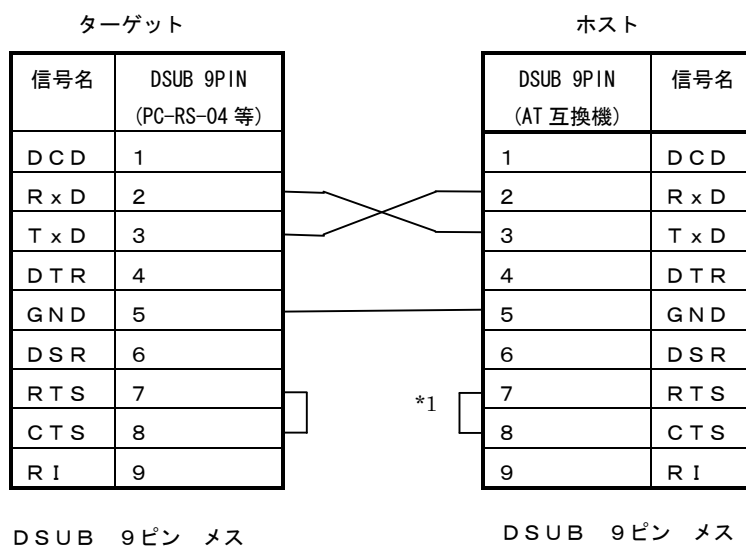
内蔵フラッシュROMを使用する場合には書き込みツールが別途必要です。（弊社製品 FlashWriter 等）

⑤ RS232Cケーブル

クロスケーブルを用意して下さい。

市販のクロスケーブルをご使用いただけますが、結線を確認して下さい。

#### ケーブル結線例（クロスケーブル）



\*1 ホスト側のRTSとCTSを短絡してご使用ください。

## 2. 2 インストール方法

### 2. 2. 1 インストール手順

- ① **【スタート】**メニューから、**【ファイル名を指定して実行】**を選び、<sup>\*1</sup>**「D:¥INSTALL¥SETUP.EXE」**を実行して下さい。

\*1 CD-ROM のドライブを指定してください。

#### 注意

StarterKit からアップグレードされたお客様はターゲットモニタを入れ替えていただく必要があります。  
後述の「3.7 アルファボードシリーズでの使用について」を参考にしてください。

## 2. 2. 2 ディレクトリ構成

以下にインストールしたフォルダの構成を示します。

なお、以下の図はフルインストール時の構成であり、インストール方法によってはフォルダが存在しない場合もあります。

### 【インストールフォルダの構成】

インストールフォルダ (名称変更可能)	説明
VM2.exe	VisualMonitor実行ファイル
apsys.txt	環境設定ファイル
apcol.txt	環境設定ファイル
header	userlog用ヘッダファイル
Lib	ターゲットモニタ用、uselog用ライブラリ
sample	ユーザプログラムサンプル (詳細は次頁を参照)
gcc	gcc用ユーザプログラムサンプル
hitachi	日立C用ユーザプログラムサンプル
vmedit	ターゲットモニタ用フォルダ
703x	SH7032/34用ターゲットモニタファイル
704x	SH7040/41/42/43/44/45用ターゲットモニタファイル
7046	SH7046/47用ターゲットモニタファイル
705x	SH7050/51用ターゲットモニタファイル
7052	SH7052/53/54用ターゲットモニタファイル
7055	SH7055用ターゲットモニタファイル
7065	SH7065用ターゲットモニタファイル
7144	SH7144/45用ターゲットモニタファイル
7604	SH7604用ターゲットモニタファイル
7709	SH7709用ターゲットモニタファイル
7709A	SH7709A/S用ターゲットモニタファイル
7708	SH7708S/R用ターゲットモニタファイル
7718	SH7718R用ターゲットモニタファイル
7729	SH7729/R用ターゲットモニタファイル
ap_board	アルファボードシリーズ用ターゲットモニタ
flash	フラッシュROMダウンロード用
ram	RAMダウンロード用
1chip	内蔵メモリのみシステム用
src	ソースファイルフォルダ (変更しないでください)
template	テンプレートファイル (変更しないでください)
tools	gccコンパイラ等

ユーザプログラムサンプルフォルダ内の構成は次ページを参照してください。

ユーザプログラムサンプルは、CPUの種類やダウンロードするメモリによって以下の構成で分類されています。  
該当するサンプルプログラムをご利用ください。

SH1、SH2の場合

¥Vmonitor¥sample¥<コンパイラ名>¥<ボード名、またはCPU名>¥<モニタのタイプ>

SH3の場合

¥Vmonitor¥sample¥<コンパイラ名>¥<ボード名、またはCPU名>¥<エンディアン>¥<モニタのタイプ>

<コンパイラ名>

ディレクトリ名	コンパイラの種類
gcc	gcc
hitachi	日立C

<ボード名、またはCPU名> (flash、ram、lchip は、モニタのタイプの有無を示しています。)

ディレクトリ名	ボード、CPUの種類	flash	ram	lchip
ap_sh_0a	AP-SH-0A (SH7032)		○	
ap_sh2_0a	AP-SH2-0A (SH7043)	○	○	
ap_sh2f_0a	AP-SH2F-0A (SH7045F)	○	○	
ap_sh2f_1a	AP-SH2F-1A (SH7051F)	○	○	○
ap_sh2f_2a	AP-SH2F-2A (SH7044F)	○	○	
ap_sh2f_3a	AP-SH2F-3A (SH7065F)	○	○	
ap_sh2f_4a	AP-SH2F-4A (SH7046F)	○	○	○
ap_sh2f_5a	AP-SH2F-5A (SH7047F)	○	○	○
ap_sh2f_6a_32	AP-SH2F-6A CS1=32BIT BUS (SH7145F)	○	○	○
ap_sh2f_6a_16	AP-SH2F-6A CS1=16BIT BUS (SH7145F)	○	○	○
ap_sh2f_7a_16	AP-SH2F-7A CS1=16BIT BUS (SH7144F)	○	○	○
ap_sh2f_7a_8	AP-SH2F-7A CS1=8BIT BUS (SH7144F)	○	○	○
sf_7044f	SF-7044F (SH7044F)	○	○	
sf_7144f	SF-7144F (SH7144F)	○	○	○
ap_sh3_0a	AP-SH3-0A (SH7709)	○	○	
ap_sh3_1a	AP-SH3-1A (SH7709A)	○	○	
ap_sh3_2a	AP-SH3-2A (SH7709S)	○	○	
ap_sh3d_0a	AP-SH3D-0A (SH7729)	○	○	
ap_sh3d_1a	AP-SH3D-1A (SH7729R)	○	○	
sh7052	SH7052			○
sh7053	SH7053			○
sh7054	SH7054			○
sh7055	SH7055			○

※AP-SH2F-6A、AP-SH2F-7Aは、ジャンプ設定によって、CS1のバス幅を変更することができるためそれぞれのバス幅に合わせてターゲットモニタを用意してあります。  
ボードの設定に合わせ、選択してご使用下さい。

## &lt;エンディアン&gt;

ディレクトリ名	エンディアン
big	ビッグエンディアン
little	リトルエンディアン

## &lt;モニタのタイプ&gt;

ディレクトリ名	モニタのタイプ
flash	ユーザプログラムをフラッシュ ROM ダウンロードするモニタ。
ram	ユーザプログラムを RAM ダウンロードするモニタ。
lchip	内蔵 RAM と内蔵フラッシュ ROM のみを使用するモニタ。 ユーザプログラムは内蔵フラッシュ ROM へダウンロードされます。

\* REAL i 用のサンプルプログラムは、インストールされませんので CD-ROM から必要なサンプルをコピーしてご使用ください。

フォルダ名    ¥REAL i ¥<REAL i バージョン番号> ¥<コンパイラ名> ¥<ボード名> ¥<ダウンロードメモリ領域名>

注意) REAL i は SH2E、SH2DSP、SH3、SH3E、SH3DSP、SH4 には対応していません。

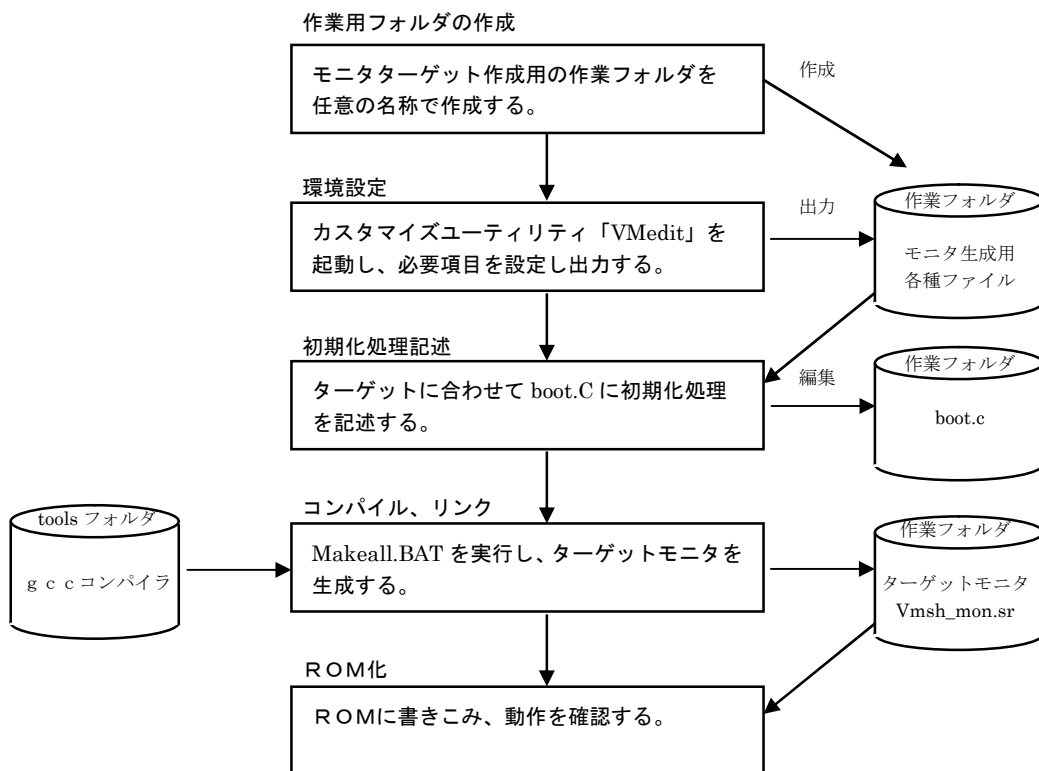


## 第3章. ターゲットモニタソフトの作成

VisualMonitor のモニタソフトは、ユーザターゲットにあわせてカスタマイズする必要があります。  
 弊社製のCPUボードをご使用になる場合には、カスタマイズ済みのターゲットモニタを用意しております。  
 後述の「アルファボードシリーズでの使用について」をご覧ください。

### 3. 1 カスタマイズの手順

カスタマイズは以下の手順に沿っておこないます。



全てのターゲットモニタはVisualMonitorと同時にインストールされるgccコンパイラで生成されます。  
 これはデバッグ対象となるユーザプログラムが日立Cの場合でも同様です。

## 3. 2 VMeditの使用方式

VMeditを使用して、ターゲットになるハードウェアの条件を設定します。

### 3. 2. 1 ターゲットCPUの設定

¥インストールフォルダ¥VMedit¥VMedit.exe を実行すると以下の設定画面が表示されます。  
ターゲットのハードウェアに合わせて、各項目を設定します。



設定項目	設定内容	設定値
CPU	該当するCPUを選択します。 SH7046/47、SH7144/45 の設定はCPUの動作モードによって2種類ありますのでご注意ください。(φ=Pφ、φ=Pφ×2)	該当CPUを選択
エンディアン	バスのエンディアンを選択します。	big または little を選択
動作周波数	周辺モジュールの動作周波数を設定します。	4 ~ 100MHz 小数点以下8桁
SCIチャンネル	VisualMonitor で使用するSCIのチャンネルを設定します。	該当するチャンネルを選択
通信レート	ターゲットとPC間の通信スピードを設定します。 当然、通信レートが早いほどレスポンスは早くなりますが、ターゲットの動作周波数が遅い場合には追従できませんので注意する必要があります。 目安値を記載しますので参考にしてください。	9600/14400/19200/38400 57600/115200 から選択
ユーザプログラムブート待ち時間	自動ブート機能でユーザプログラムが起動するまでの待ち時間を設定します。	100 ~ 5000 msec
起動メッセージ	起動時にログウィンドウに表示されるメッセージを設定します。	80文字 ASCIIコード

表 3. 2. 1 通信レートを目安

動作周波数 (MHz)	4	4.9152	6	6.144	7.37288	8	9.8304
通信レート	9600	19200	19200	19200	19200	19200	38400

動作周波数 (MHz)	10	12	12.288	14.7456	16	19.6608	20
通信レート	38400	38400	38400	38400	38400	38400	38400

動作周波数 (MHz)	24	24.576	28.7	30	33		
通信レート	38400	38400	57600	115200	115200		

注意)

通信レートは動作周波数によって誤差を生じるため、動作周波数と通信レートが比例するとは限りません。動作周波数によって、通信レートの誤差が大きい場合には通信が不安定になる場合がありますので注意してください。(誤差はできるだけ±3%前後に収めてください)

通信レートと誤差の計算方法は各 CPU のハードウェアマニュアルを参照してください。

例) SH7045 の場合

通信レートの算出式

$$N = \frac{\phi}{64 \times 2^{2n-1} \times B} \times 10^6 - 1$$

誤差の算出式

$$\text{誤差 (\%)} = \left\{ \frac{\phi \times 10^6}{(N+1) \times B \times 64 \times 2^{2n-1}} \right\} \times 100$$

B : ビットレート (bit/S)

$\phi$  : 動作周波数 (MHz)

N : ポーレートジェネレータの BRR の設定値 ( $0 \leq N \leq 255$ )

n : ポーレートジェネレータ入力クロック ( $n=0, 1, 2, 3$ )

VMe d i t で設定した場合、N と n の値は自動的に算出されます。

算出された値は VMe d i t の出力で指定したフォルダに作成される set.h を参照すれば確認できます。

誤差が大きい場合には直接編集して、最適値に設定してください。

<set.h>

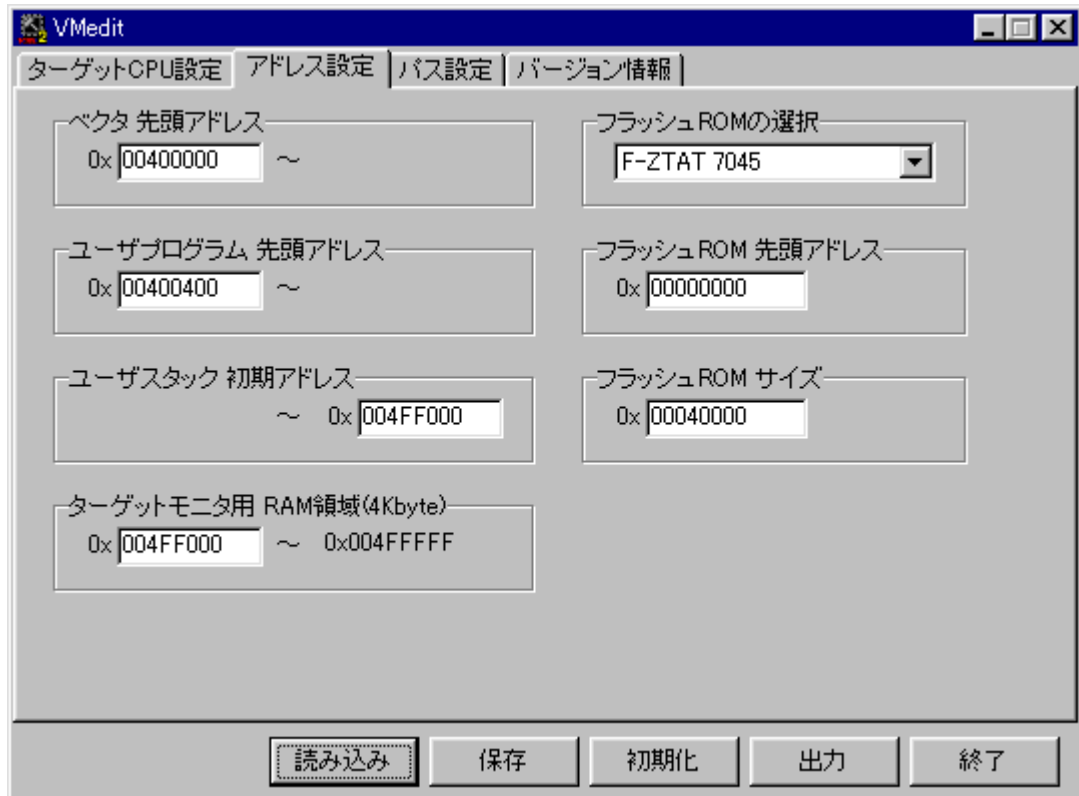
```
#define SIO_CLK 0x0000 /* ポーレートジェネレータ入力クロック */
#define SIO_BRR 22 /* ポーレートジェネレータ (BRR) の設定値 */
```

SIO\_CLK が n、SIO\_BRR が N となります。

SH7065 の場合は、SIO\_CLK の下位 1byte が n になり、上位 1byte は SC2SSR の N1, N0 の値になります。

### 3. 2. 2 アドレス設定

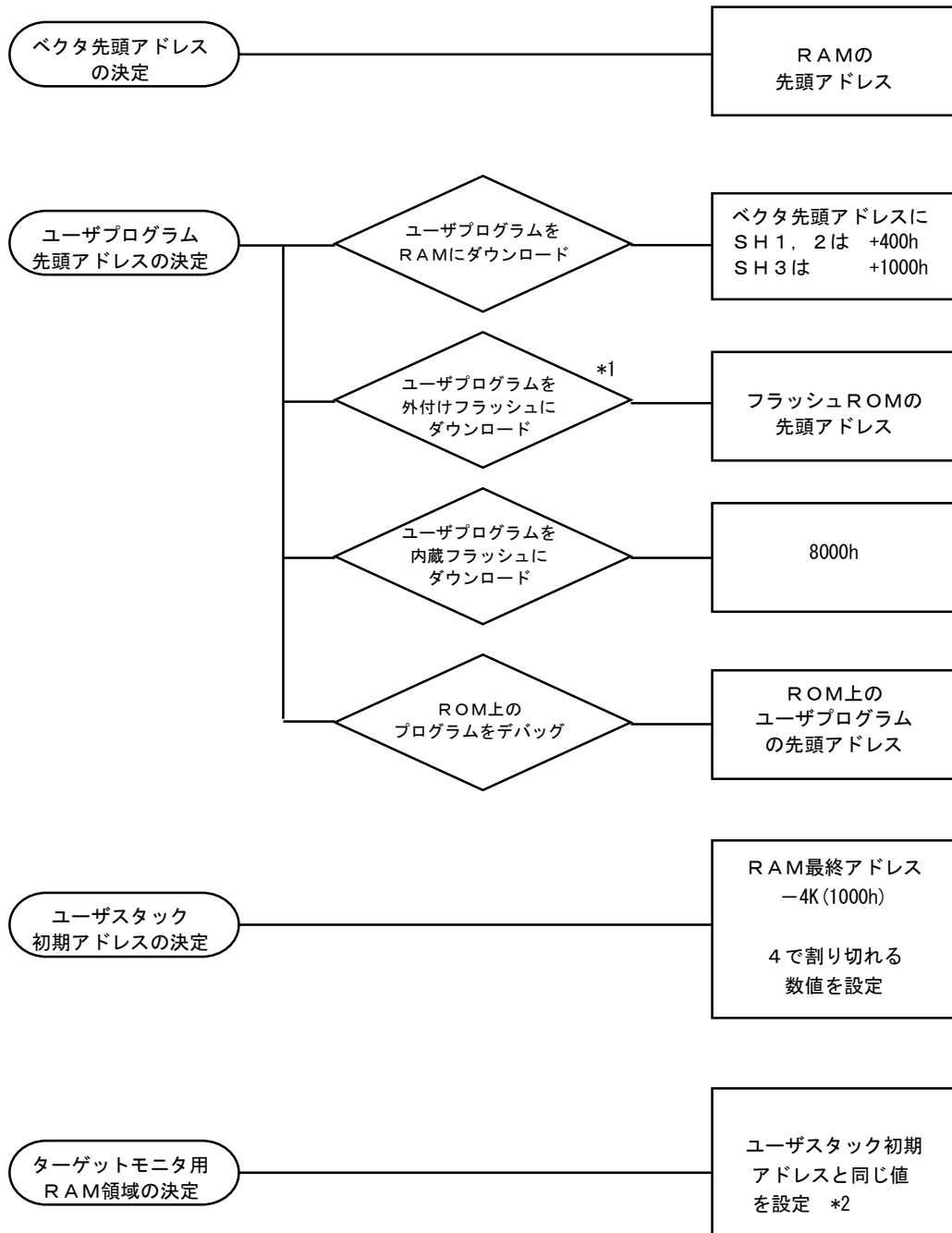
ターゲットのメモリ構成に合わせて、各アドレスを設定します。



設定項目	設定内容	設定値
ベクタ 先頭アドレス	ベクタの先頭アドレスを設定します。	16進数で指定
ユーザプログラム 先頭アドレス	ユーザプログラムのスタートアドレスを設定します。	16進数で指定
ユーザスタック 初期アドレス	ユーザプログラムが使用するスタックの初期アドレスを設定します。	16進数で指定
ターゲットモニタ用 RAM領域	ターゲットモニタが使用するワーク領域の先頭アドレスを指定します。	16進数で指定 4Kbyte使用します。
フラッシュROMの 選択	ユーザプログラムのダウンロード領域がフラッシュROMの場合、フラッシュROMの種類を指定します。コードの一部がフラッシュROMにある場合にも指定します。	29F800B、29F800T、29LV320Bのいずれかを選択 F-ZTATの場合は該当CPUを選択
フラッシュROM 先頭アドレス	使用するフラッシュROMの先頭アドレスを指定します。	16進数で指定
フラッシュROM サイズ	使用するフラッシュROMのサイズが表示されず。	

### <各アドレスの決定方法>

各アドレスは自由に配置可能ですが、一般的には以下のフローにしたがって配置を決定します。

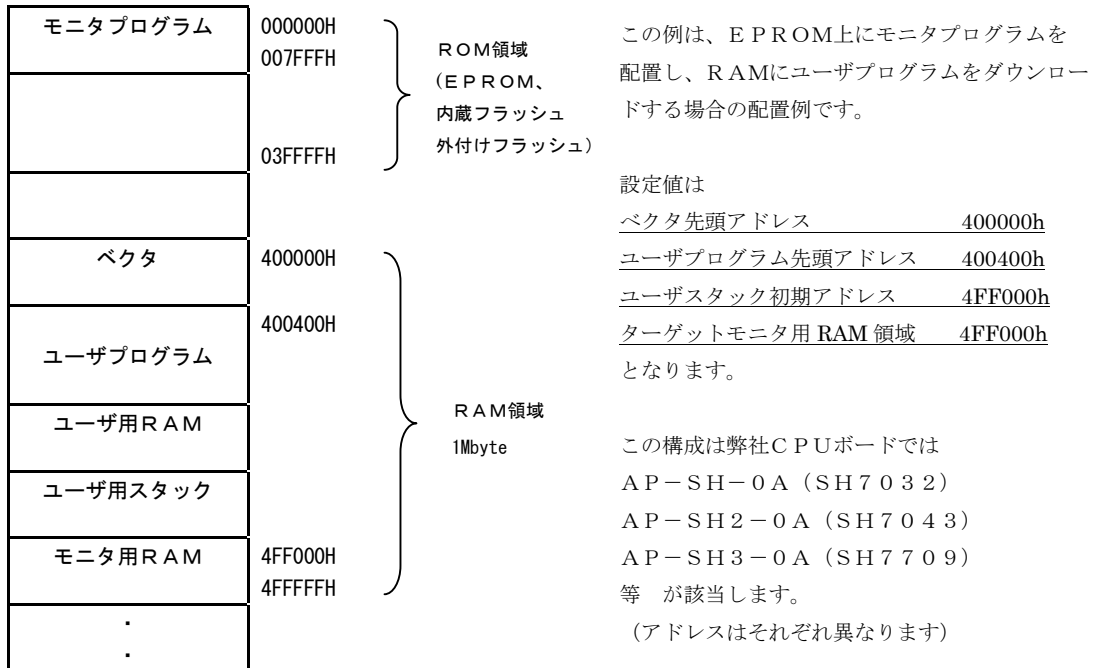


\* 1 フラッシュROMの配置は後述の設計例を参照してください。

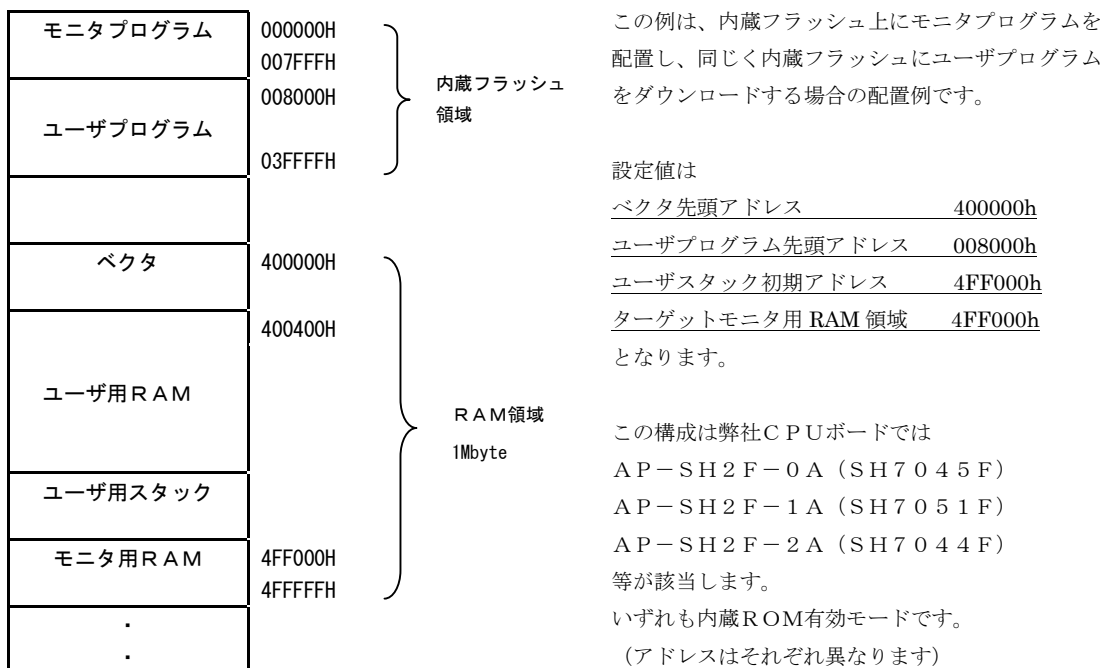
\* 2 ターゲットモニタ用RAM領域とユーザスタック初期アドレスは重複していてもスタックは破壊されません。  
SHの場合、実際に使用されるスタックは（ユーザスタック初期アドレス-4）番地からとなるからです。

前述のフローにしたがって配置した具体例を上げます。

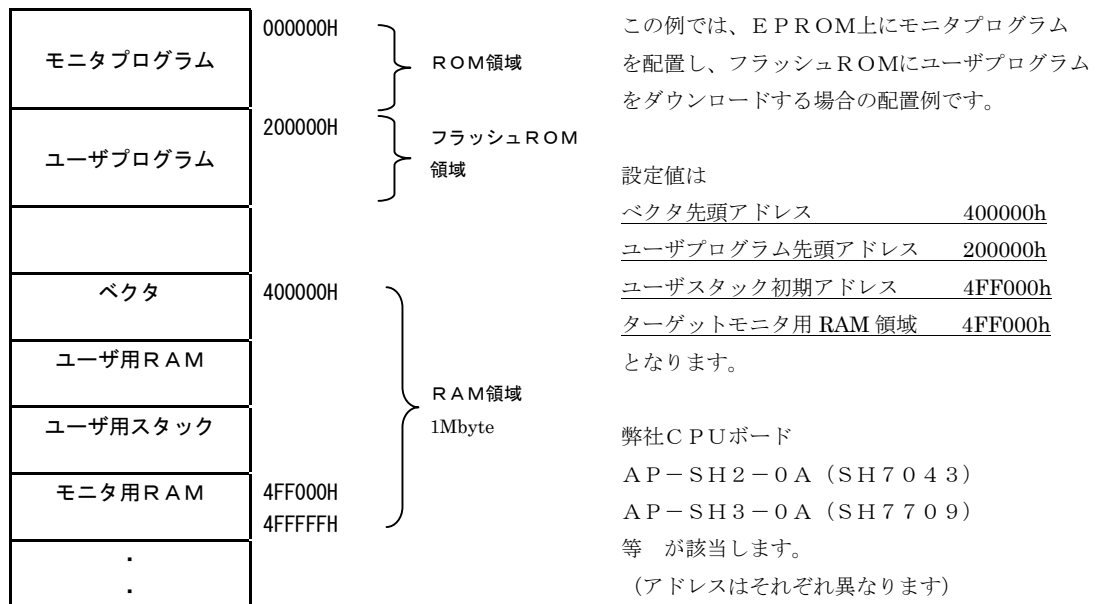
例 1) ROMとRAMを搭載したターゲットの場合



例 2) 内蔵フラッシュROMとRAMを搭載したターゲットの場合



例3) ROMとフラッシュROMとRAMを搭載したターゲットの場合



\*注意

外付けフラッシュにモニタとユーザプログラムを配置する方法は、現在のところサポートされていません。

<内蔵フラッシュを使用する場合の注意事項>

VisualMonitor は、内蔵フラッシュROMへユーザプログラムをダウンロードをする際に、内蔵RAM上にフラッシュ書き込み/消去プログラムを展開します。そのため、そのプログラムが展開される領域に、ユーザプログラムのコード領域とベクタテーブル、ターゲットモニタのワーク領域が重ならないように各エリアを設定する必要があります。書き込み/消去プログラムを展開する領域を以下の表に示します。なお、この領域をユーザプログラムの変数領域、スタック領域として使用することは問題ありません。

CPU	書き込み、消去プログラム展開領域
SH7044F、7045F	0xfffff000 ~ 0xfffff4ff
SH7046F、7047F	0xffffe000 ~ 0xffffe5ff
SH7050F、7051F	0xfffff000 ~ 0xfffff4ff
SH7052F、7053F、7054F	0xffff9000 ~ 0xffff95ff
SH7055F	0xffff6000 ~ 0xffff65ff
SH7065F	0xffff8000 ~ 0xffff85ff
SH7144F、SH7145F	0xfffff000 ~ 0xfffff5ff

### 3. 2. 3 コンパイラパス設定

ターゲットモニタを生成するためのコンパイラとライブラリのパスを指定します。

VisualMonitorのターゲットモニタは、全て付属のgccコンパイラで生成されます。

以下の設定は、VisualMonitorをC:\Vmonitorディレクトリにインストールした場合のパス設定です。



設定項目	設定内容
モニタ生成用 コンパイラパス指定	gccコンパイラ等が格納されているフォルダを指定します。 標準では%インストールディレクトリ%\vmedit%\toolsになります。
ライブラリパス指定	ターゲットモニタ用ライブラリが格納されているフォルダを指定します。 標準では%インストールディレクトリ%\libになります。



### 3. 2. 4 共通操作

共通操作ボタンは、全ての設定画面から操作可能なボタンです。

設定ファイルの操作や、出力など設定値をターゲットモニタに反映するための操作項目が割り付けられています。



#### 読み込みボタン

設定値は設定ファイルとして保存したり、読み込むことができます。

読み込みボタンをクリックすると、ファイル選択ダイアログが表示されますので任意のファイルを選択してパラメータを読み込みます。

弊社製CPUボードをご使用になる場合には、パラメータファイルが添付されておりますので、それらをご使用ください。

#### 保存ボタン

設定ファイルを保存します。

保存ボタンをクリックすると、ファイル保存ダイアログが表示されますので任意の名前で設定ファイル(\*.vme)を保存できます。

#### 初期化ボタン

設定値をデフォルトの初期値に戻します。

#### 出力ボタン

全てのパラメータが設定できたところで出力ボタンをクリックします。

フォルダ参照ダイアログが表示されるので、任意のフォルダを指定してOKをクリックしてください。

フォルダ内には以下のファイルが作成されます。(¥Target 以下に出力した場合)

¥Target	—	makeall.bat	モニタ生成用のバッチファイル
	—	boot.c	モニタのブートプログラム
	—	set.h	c ヘッダファイル
	—	seta.h	asm ヘッダファイル
	—	¥src	モニタ生成に必要なファイル群

### 3. 3 初期化処理のカスタマイズ

#### 3. 3. 1 初期化処理の記述

各ターゲットに合わせて初期化処理を記述します。

設定値を出力したフォルダ内に作成された boot.c をメモ帳などのテキストエディタで編集してください。

初期化処理では、モニタが動作するのに最低限必要な処理を記述します。

例えば、バスコントローラの設定やピンファンクションコントローラの設定やDRAMのダミーリフレッシュ処理、キャッシュメモリの設定等を記述します。記述式は一般的なCの書式で記述します。

ここに記述した処理はリセット直後に実行される処理です。

弊社製CPUボードを使用する場合の処理が記述されておりますので参照してください。

ターゲットモニタプログラム boot.C

```
#include "def.h"

/*+++++++*/
/* <カスタマイズ> */
/* アルファボードシリーズを使用する場合、該当するCPUボードを1にして */
/* ください。 */
/* その他の場合は全て0にして、ターゲットに合わせて初期化処理を記述して */
/* ください */
/*+++++++*/
#define AP_SH2_0A 0 /*7043 使用 AP 製 CPU ボード*/
#define AP_SH2F_0A 1 /*7045 使用 AP 製 CPU ボード*/
#define AP_SH2F_2A 0 /*7044 使用 AP 製 CPU ボード*/
/*+++++++*/
. (途中省略)
.
/*7045 使用 AP 製 CPU ボード*/
#if (AP_SH2F_0A)

    WordWrite(0xFFFF8620, 0x202f); /*BSC BCR1*/
                                     /*CS0:16bit CS1:32bit CS2:16bit CS3:16bit*/
    WordWrite(0xFFFF8622, 0x5500); /*BSC BCR2*/
                                     /*CS0~CS3 3アイドルサイクル*/

    WordWrite(0xFFFF8624, 0x0002); /*BSC WCR1*/
                                     /*CS0 1ウェイト*/

#endif

. (以下 続く)
```

boot.cには任意のアドレスに任意の値を書きこむためのマクロとして以下のマクロを用意しています。これらのマクロを使用すれば、簡単にコーディングできます。

```
ByteWrite(addr, data) 1バイト書き込み
WordWrite(addr, data) 2バイト書き込み
LongWrite(addr, data) 4バイト書き込み
```

### 3. 4 モニタの生成

前述の 3. 1～3. 3 までの処理を終了した後、モニタを生成します。

モニタの生成は、設定値の出力で選択したフォルダ内の `makeall.bat` を実行します。

アイコンをクリックするか、MS-DOS プロンプトから該当ディレクトリに移動して、`MAKEALL` を実行してください。

エラーがなければ、同じフォルダ内にターゲットモニタのROM化ファイル(`vmsh_mon. sr`)が生成されます。

エラーがあった場合には、`vmsh_mon. err` にエラー内容が出力されますので内容を確認して修正ください。

Vmsh\_mon. err の内容 (正常終了時の一例)

```
GNU assembler version 2.9.1 (sh3)
GNU assembler version 2.9.1 (sh3)
Using builtin specs.
gcc version egcs-2.91.60 19981201 (egcs-1.1.1 release)
D:\GCC\BIN\cpp.exe -lang-c -v -iprefix d:/gcc/bin/./lib/gcc-lib/sh3\2_91_60\ -isystem D:\GCC\BIN\include -
undef -D __GNUC__=2 -D __GNUC_MINOR__=91 -D __sh__ -D __sh__ -Acpu(sh) -Amachine(sh) -Wall -D __sh2__
boot.c C:\WINDOWS\TEMP\ccp8loW7.i
GNU CPP version egcs-2.91.60 19981201 (egcs-1.1.1 release) (Hitachi SH)
#include "..." search starts here:
#include <...> search starts here:
D:\GCC\BIN\include
End of search list.
D:\GCC\BIN\cc1.exe C:\WINDOWS\TEMP\ccp8loW7.i -quiet -dumpbase boot.c
D:\VMTEST\src>D:\GCC\BIN\gcc -c -v -Wall -m2 -O0 -omain.o main.c

D:\VMTEST\src>D:\GCC\BIN\gcc -c -v -Wall -m2 -O0 -oflash.o flash.c

D:\VMTEST\src>
D:\VMTEST\src>D:\GCC\BIN\ld @vmsh_mon.lnk -Tvmsh_mon.cmd
GNU ld version 2.9.1 (with BFD 2.9.1)
d:/gcc/bin/ld.exe: mode sh
vector.o
int_asm.o
boot.o
main.o
flash.o
(C:\WINDOWS\src\VisualMonitor\vmedit\lib\sh704x.a)sio.o
(C:\WINDOWS\src\VisualMonitor\vmedit\lib\sh704x.a)aplib.o
(C:\WINDOWS\src\VisualMonitor\vmedit\lib\sh704x.a)monimain.o

D:\VMTEST\src>
D:\VMTEST\src>D:\GCC\BIN\objcopy -O srec vmsh_mon.out vmsh_mon.sr

      .
      .          (途中省略)
      .

2.91.60 19981201 (egcs-1.1.1 release) (Hitachi SH)
#include "..." search starts here:
#include <...> search starts here:
D:\GCC\BIN\include
End of search list.
D:\GCC\BIN\cc1.exe C:\WINDOWS\TEMP\cc1DS0su.i -quiet -dumpbase flash.c -m2 -O0 -Wall -version -o
C:\WINDOWS\TEMP\cccliZJ3R.s
GNU C version egcs-2.91.60 19981201 (egcs-1.1.1 release) (sh3) compiled by GNU C version 2.8.1.
D:\GCC\BIN\as.exe -oflash.o C:\WINDOWS\TEMP\cccliZJ3R.s
```

### 3. 5 モニタROMの準備

- ① カスタマイズ後、作成されたモトローラSフォーマットのROM化ファイル (vmsh\_mon. sr) をROMライターにてPROMに書き込んで下さい。内蔵フラッシュに配置する場合には弊社製書き込みツール「FLASH WRITER」もしくは日立製の書き込みツール等で書き込んでください。
- ② 書き込んだPROMをターゲット上に搭載して下さい。

### 3. 6 モニタROMの動作確認

- ① PCとターゲットをRS232Cクロスケーブルで接続します。
- ② 後述の「チュートリアル」の基本操作手順で、リセットメッセージが表示されることを確認してください。

### 3. 7 アルファボードシリーズでの使用について

#### 3. 7. 1 カスタマイズ済みターゲットモニタ

弊社製のCPUボードをご使用になる場合には、ボード毎にカスタマイズ済みのターゲットモニタと設定ファイルを用意しておりますので、それらを使用すればターゲットモニタを生成する必要はありません。

フォルダ名 : ¥インストールフォルダ名¥vmedit¥ap\_board¥ダウンロード領域名¥ボード名¥

AP-SH2F-0AのRAMダウンロード用の場合、

¥インストールフォルダ名¥vmedit¥ap\_board¥ram¥ap\_sh2f\_0a¥ となります。

詳細は前述の「インストール」の「ディレクトリ構成」を参照してください。

フォルダ内の以下のファイルをEPROMに焼くか、内蔵ROMに書き込んで使用してください。

ターゲットモニタファイル名 : vmsh\_mon. sr

## 第4章. ユーザプログラムの作成

VisualMonitor でユーザプログラムをデバッグするためには、ユーザプログラム上でおこなわなければならない処理があります。本章では、サンプルプログラムを例にコーディング方法と注意点を説明します。

### 4. 1 デバッグに必要なファイル

ダウンロードに必要なファイルは、デバッグ情報ファイルです。

デバッグ情報ファイルは g c c (COFF) または日立 C (SYSROF、ELF) でコンパイルされたものに対応しています。また、ソースプログラムを表示するために、同じフォルダにソースファイルが収録されている必要があります。

### 4. 2 ユーザプログラムの作成上の注意

#### 4. 2. 1 ユーザプログラムの制限事項

VisualMonitor はユーザプログラムを実行する時にターゲットモニタで指定したアドレスから実行します。従ってプログラムはターゲットモニタで指定したアドレスから実行できるようにマッピングしてください。また、VisualMonitor で使用する割り込みはユーザプログラムでは使用できませんので注意してください。

ユーザプログラム作成時のその他の制限事項

項目	制限事項
扱えるソースファイル数	300ファイル
プログラムの大きさ	パソコンのメモリ容量に依存します。
セクション数	100セクションまでです。
割り込みレベル	割り込みレベル14以下を使用して下さい。 割り込みレベル15はVisualMonitorの割り込みで使用します。 VisualMonitorは、モニタ起動時にSR(ステータスレジスタ)の割り込みマスクを14に設定します。 <b>※ユーザプログラム内で割り込みマスクは15に設定しないで下さい。</b>
REALiモニタのタスク情報数	100タスク
最適化オプション	基本的には最適化オプションを外してコンパイルして下さい。 ステップ実行時などにプログラム実行位置とCソースが一致しない場合が発生します。

## 4. 3 サンプルプログラム

### 4. 3. 1 サンプルプログラムの構成

VisualMonitor では各種ターゲットに合わせたサンプルプログラムを添付しております。

サンプルプログラムをお客様のシステムにあわせてカスタマイズし、ユーザプログラムとリンクすれば簡単にご使用いただけます。

サンプルプログラムは基本的に弊社製のCPUボードで動作するサンプルを用意しておりますが、アドレスを変更すれば任意のターゲットで動作させることができます。弊社製CPUボードの緒元は付録資料をご覧ください。

以下に、コンパイラをgcc、ターゲットをAP-SH2F-0Aで動作させるときのサンプルプログラムのファイル構成を説明します。（基本的にサンプルプログラムの構成は全て同じです）

ファイル名	処理内容
sh7040s.h	SH7040シリーズレジスタ定義
gmachine.h	SH2 MPU 依存命令ヘッダファイル定義
gmachine.c	SH2 MPU 依存命令
crt0.s	スタートアップルーチン (GCCのみ)
section.src	セクション定義ファイル (日立Cのみ)
main.c	メイン処理ルーチン
UserLog.h	USERLOG 関数ライブラリヘッダ
Sh2ul.a(lib)	USERLOG 組み込み関数ライブラリ

\* REAL i 用のサンプルプログラムは、インストールされませんのでCD-ROMから必要なプログラムをコピーしてご使用ください。

フォルダ名    ¥REAL i ¥<REAL i バージョン番号> ¥<コンパイラ名> ¥<ボード名> ¥<ダウンロードメモリ領域名>

## 4. 4 ベクタテーブル

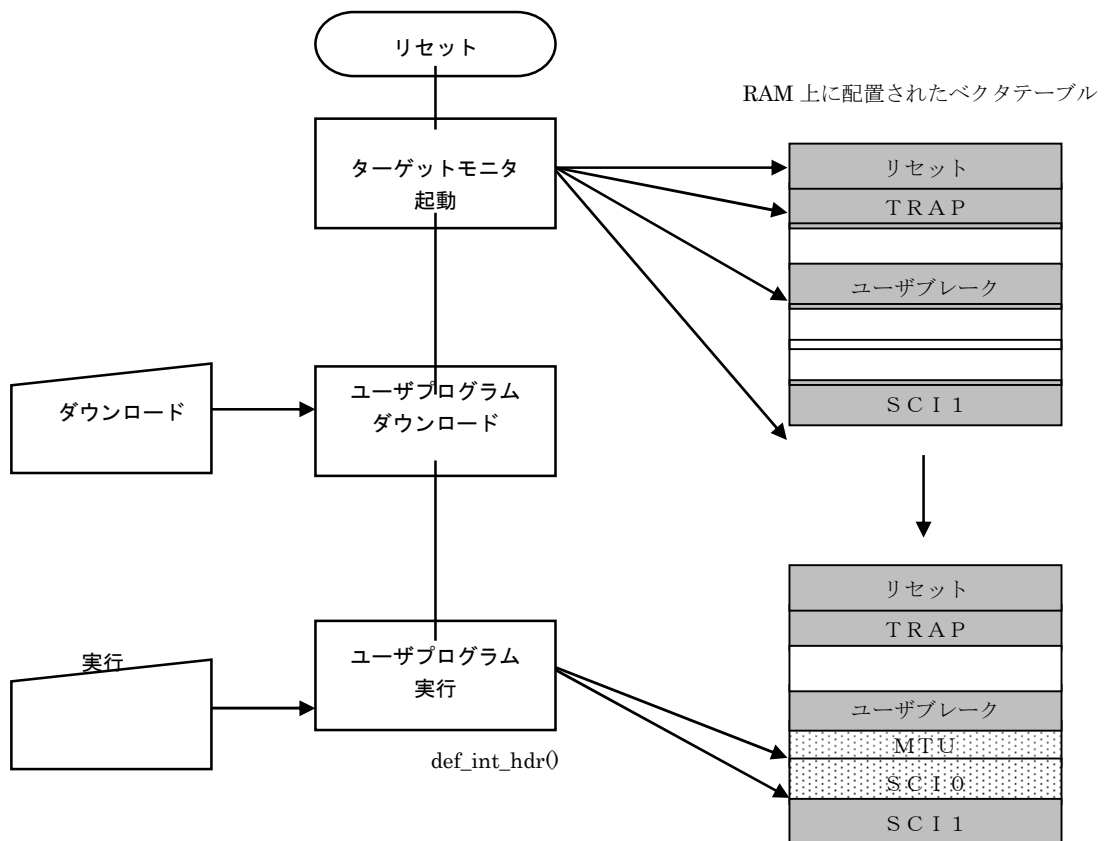
### 4. 4. 1 ベクタテーブルの配置と構造

通常のROM化プログラムの場合、プログラムの先頭にベクタを配置するのが一般的と考えられますが、VisualMonitorは例外処理ベクタをユーザプログラムと共有するためにベクタテーブルをRAM上に配置して使用します。

したがって、ユーザプログラムでは、プログラムの先頭にベクタを配置するのではなく、プログラムの先頭で必要なベクタ情報をRAM上に転送する処理が必要となります。

以下に VisualMonitor の起動時からユーザプログラムの実行にいたるまで流れと、ベクタテーブルの状況をイメージで示します。

ターゲットモニタの環境 : SH2でSCI1をモニタ用に使用  
 ユーザプログラム : MTU、SCIOをプログラムで使用



上記のようにターゲットモニタは起動した時点で、モニタが使用するベクタアドレスをベクタテーブル上に転送します。ユーザプログラムでは、使用するベクタをプログラムの先頭でベクタテーブル上に登録します。VisualMonitorで使用するベクタは書き換えないように注意してください。

#### 4. 4. 2 ベクタテーブルの登録方法 (SH1、SH2)

実際にユーザプログラム上でベクタを登録する方法を説明します。

##### <ベクタ登録関数>

ソースファイル : GMACHINE.H GMACHINE.C	
記述式	機能
void def_int_hdr(unsigned short intno, void *inthdr(void));	ベクタテーブルにベクタアドレスを登録します。
<引数> (ベクタ番号、ベクタアドレス) 例 : def_int_hdr( 88, (void*)int_mtu0);	
<戻り値> なし	
<使用方法> 使用する割り込みのアドレスを割り込み処理に先立ち、ユーザプログラム内で登録してください。	

##### プログラムコーディング例

```

#include "sh7040.h"
#include "gmachine.h"
#include "userlog.h"

省略

/*****/
/*      CMT0 初期化      */
/*****/
void cmt0_init(void)
{
    def_int_hdr(144, (void*)int_cmi0); /*割り込みハンドラの設定 */
    INTC. IPRG. WORD |= 0x0070;      /*CMT0 InterruptLevel = 7 */

    CMT. CMSTB. WORD &= ~0x0001;    /*CMCNT カウント停止 */
    省略
}

/*****/
/*      CMT0 割り込み      */
/*****/
#pragma interrupt
void int_cmi0(void)
{
    static char      flag;

    CMT0. CMCSR. WORD &= ~0x0080;    /*CMF クリア */
    省略
}

/*****/
/*      MTU0 初期化      */
/*****/
void mtu0_init(void)
{
    def_int_hdr(88, (void*)int_mtu0); /*割り込みハンドラの設定 */
    INTC. IPRD. WORD | = 0x7000;     /*MTU0 InterruptLevel = 7 */

    MTU0. TCR. BYTE = 0x23;         /*T G R A のコンペアマッチ */
    MTU0. TSR. BYTE = 0x00;         /*ステータスクリア */
    省略
}

```

ベクタ番号 : 144 に int\_cmi0() を登録

ベクタ番号 : 88 に int\_mtu0() を登録



#### 4. 4. 3 VisualMonitorが使用する割り込み (SH1、SH2)

VisualMonitorでは以下の割り込みを使用していますので、ユーザプログラムで設定しないでください。

SCIは指定した1つのチャンネルしか使用されませんのでVisualMonitorで使用しないチャンネルはユーザプログラムで使用できません。下表以外のベクタ番号は各CPUのデータシートをご覧ください

##### <SH1 SH703xの場合>

例外要因		ベクタ番号	ベクタテーブルアドレスオフセット
パワーオンリセット	PC	0	H'00000000-H'00000003
	SP	1	H'00000004-H'00000007
マニュアルリセット	PC	2	H'00000008-H'0000000B
	SP	3	H'0000000C-H'0000000F
ユーザブ레이크		12	H'00000030-H'00000033
トラップ命令 (20H)		32	H'00000080-H'00000083
トラップ命令 (21H)		33	H'00000084-H'00000087
SCIO	ERIO	100	H'00000190-H'00000193
	RXIO	101	H'00000194-H'00000197
	TXIO	102	H'00000198-H'0000019B
SCI1	ERI1	104	H'000001A0-H'000001A3
	RXI1	105	H'000001A4-H'000001A7
	TXI1	106	H'000001A8-H'000001AB

1CHのみ使用

##### <SH2 704xの場合>

例外要因		ベクタ番号	ベクタテーブルアドレスオフセット
パワーオンリセット	PC	0	H'00000000-H'00000003
	SP	1	H'00000004-H'00000007
マニュアルリセット	PC	2	H'00000008-H'0000000B
	SP	3	H'0000000C-H'0000000F
ユーザブ레이크		12	H'00000030-H'00000033
トラップ命令 (20H)		32	H'00000080-H'00000083
トラップ命令 (21H)		33	H'00000084-H'00000087
SCIO	ERIO	128	H'00000200-H'00000203
	RXIO	129	H'00000204-H'00000207
	TXIO	130	H'00000208-H'0000020B
SCI1	ERI1	132	H'00000210-H'00000213
	RXI1	133	H'00000214-H'00000217
	TXI1	134	H'00000218-H'0000021B

1CHのみ使用

## &lt;SH2 7046 の場合&gt;

例外要因		ベクタ番号	ベクタテーブルアドレスオフセット
パワーオンリセット	PC	0	H'00000000—H'00000003
	SP	1	H'00000004—H'00000007
予約	PC	2	H'00000008—H'0000000B
	SP	3	H'0000000C—H'0000000F
ユーザブ레이크		12	H'00000030—H'00000033
トラップ命令 (20H)		32	H'00000080—H'00000083
トラップ命令 (21H)		33	H'00000084—H'00000087
SCIO	ERI2	168	H'000002A0—H'000002A3
	RXI2	169	H'000002A4—H'000002A7
	TXI2	170	H'000002A8—H'000002AB
SCI1	ERI3	172	H'000002B0—H'000002B3
	RXI3	173	H'000002B4—H'000002B7
	TXI3	174	H'000002B8—H'000002BB

} 1CHのみ使用

## &lt;SH2 7047 の場合&gt;

例外要因		ベクタ番号	ベクタテーブルアドレスオフセット
パワーオンリセット	PC	0	H'00000000—H'00000003
	SP	1	H'00000004—H'00000007
予約	PC	2	H'00000008—H'0000000B
	SP	3	H'0000000C—H'0000000F
ユーザブ레이크		12	H'00000030—H'00000033
トラップ命令 (20H)		32	H'00000080—H'00000083
トラップ命令 (21H)		33	H'00000084—H'00000087
SCIO	ERI2	168	H'000002A0—H'000002A3
	RXI2	169	H'000002A4—H'000002A7
	TXI2	170	H'000002A8—H'000002AB
SCI1	ERI3	172	H'000002B0—H'000002B3
	RXI3	173	H'000002B4—H'000002B7
	TXI3	174	H'000002B8—H'000002BB
SCI2	ERI4	176	H'000002C0—H'000002C3
	RXI4	177	H'000002C4—H'000002C7
	TXI4	179	H'000002C8—H'000002CB

} 1CHのみ使用

## &lt;SH2 7050、7051 の場合&gt;

例外要因		ベクタ番号	ベクタテーブルアドレスオフセット
パワーオンリセット	PC	0	H'00000000-H'00000003
	SP	1	H'00000004-H'00000007
予約	PC	2	H'00000008-H'0000000B
	SP	3	H'0000000C-H'0000000F
ユーザブ레이크		12	H'00000030-H'00000033
トラップ命令 (20H)		32	H'00000080-H'00000083
トラップ命令 (21H)		33	H'00000084-H'00000087
SC10	ERIO	152	H'00000260-H'00000263
	RXIO	153	H'00000264-H'00000267
	TXIO	154	H'00000268-H'0000026B
SC11	ERI1	156	H'00000270-H'00000273
	RXI1	157	H'00000274-H'00000277
	TXI1	158	H'00000278-H'0000027B
SC12	ERI2	160	H'00000280-H'00000283
	RXI2	161	H'00000284-H'00000287
	TXI2	162	H'00000288-H'0000028B

1CHのみ使用

## &lt;SH2 7052、7053、7054、7055 の場合&gt;

例外要因		ベクタ番号	ベクタテーブルアドレスオフセット
パワーオンリセット	PC	0	H'00000000-H'00000003
	SP	1	H'00000004-H'00000007
予約	PC	2	H'00000008-H'0000000B
	SP	3	H'0000000C-H'0000000F
ユーザブ레이크		12	H'00000030-H'00000033
トラップ命令 (20H)		32	H'00000080-H'00000083
トラップ命令 (21H)		33	H'00000084-H'00000087
SC10	ERIO	200	H'00000320-H'00000323
	RXIO	201	H'00000324-H'00000327
	TXIO	202	H'00000328-H'0000032B
SC11	ERI1	204	H'00000330-H'00000333
	RXI1	205	H'00000334-H'00000337
	TXI1	206	H'00000338-H'0000033B
SC12	ERI2	208	H'00000340-H'00000343
	RXI2	209	H'00000344-H'00000347
	TXI2	210	H'00000348-H'0000034B
SC13	ERI3	212	H'00000350-H'00000353
	RXI3	213	H'00000354-H'00000357
	TXI3	214	H'00000358-H'0000035B
SC14	ERI4	216	H'00000360-H'00000363
	RXI4	217	H'00000364-H'00000367
	TXI4	218	H'00000368-H'0000036B

1CHのみ使用

## &lt;SH2 7065 の場合&gt;

例外要因		ベクタ番号	ベクタテーブルアドレスオフセット
パワーオンリセット	PC	0	H'00000000-H'00000003
	SP	1	H'00000004-H'00000007
予約	PC	2	H'00000008-H'0000000B
	SP	3	H'0000000C-H'0000000F
ユーザブ레이크		13	H'00000034-H'00000037
トラップ命令 (20H)		32	H'00000080-H'00000083
トラップ命令 (21H)		33	H'00000084-H'00000087
SC I 0	ER I 0	188	H'000002F0-H'000002F3
	RX I 0	189	H'000002F4-H'000002F7
	TX I 0	190	H'000002F8-H'000002FB
SC I 1	ER I 1	192	H'00000300-H'00000303
	RX I 1	191	H'00000304-H'00000307
	TX I 1	194	H'00000308-H'0000030B
SC I 2	ER I 2	196	H'00000310-H'00000313
	RX I 2	197	H'00000314-H'00000317
	TX I 2	198	H'00000318-H'0000031B

} 1CHのみ使用

## &lt;SH2 7604 の場合&gt;

例外要因		ベクタ番号	ベクタテーブルアドレスオフセット
パワーオンリセット	PC	0	H'00000000-H'00000003
	SP	1	H'00000004-H'00000007
マニュアルリセット	PC	2	H'00000008-H'0000000B
	SP	3	H'0000000C-H'0000000F
ユーザブ레이크		12	H'00000030-H'00000033
トラップ命令 (20H)		32	H'00000080-H'00000083
トラップ命令 (21H)		33	H'00000084-H'00000087
SC I	ER I	72*	H'00000120-H'00000123
	RX I	73*	H'00000124-H'00000127
	TX I	74*	H'00000128-H'0000012B

\*SC I の割り込みベクタは、VisualMonitor で上記のベクタ番号に設定されます。変更しないでください。

&lt;SH2 7144、7145 の場合&gt;

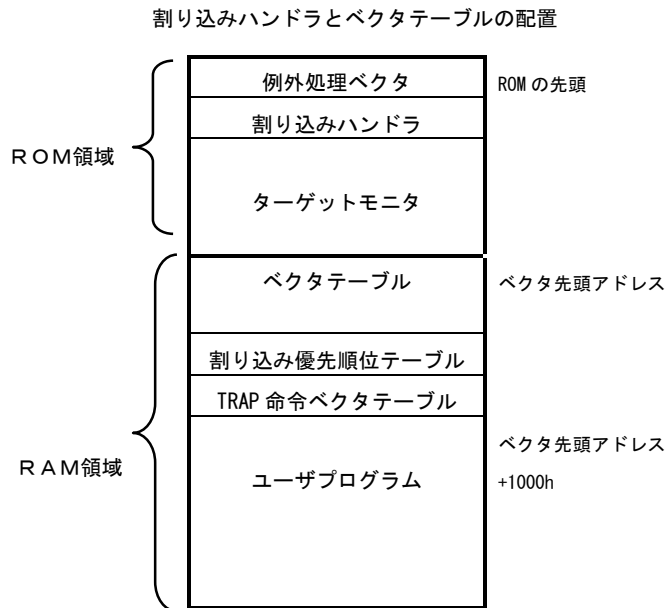
例外要因		ベクタ番号	ベクタテーブルアドレスオフセット
パワーオンリセット	PC	0	H'00000000-H'00000003
	SP	1	H'00000004-H'00000007
マニュアルリセット	PC	2	H'00000008-H'0000000B
	SP	3	H'0000000C-H'0000000F
ユーザブレーク		12	H'00000030-H'00000033
トラップ命令 (20H)		32	H'00000080-H'00000083
トラップ命令 (21H)		33	H'00000084-H'00000087
SC10	ERI0	128	H'00000200-H'00000203
	RXI0	129	H'00000204-H'00000207
	TXI0	130	H'00000208-H'0000020B
SC11	ERI1	132	H'00000210-H'00000213
	RXI1	133	H'00000214-H'00000217
	TXI1	134	H'00000218-H'0000021B
SC12	ERI2	168	H'000002A0-H'000002A3
	RXI2	169	H'000002A4-H'000002A7
	TXI2	170	H'000002A8-H'000002AB
SC13	ERI3	172	H'000002B0-H'000002B3
	RXI3	173	H'000002B4-H'000002B7
	TXI3	174	H'000002B8-H'000002BB

} 1CHのみ使用

#### 4. 4. 4 SH3の割り込みについて

SH3の場合、周辺例外処理は全て同じベクタアドレスになるため、プログラムで割り込みハンドラと個別のベクタテーブルを用意する必要があります。

VisualMonitorではこれらの機能を実現するためにターゲットモニタに割り込みハンドラが組み込まれており、SH1、SH2と同様の扱いで割り込みベクタを設定できるようになっています。



VisualMonitor で用意されている割り込みハンドラは、全てのレジスタを退避し、割り込み要因を解析した後、RAM上に展開されたベクタテーブルより該当する割り込み処理のアドレスを取得してコールします。

その後、割り込み処理が終了して割り込みハンドラに処理が戻ると、レジスタを全て復帰し、ユーザプログラムに戻ります。

したがって、ユーザプログラムでは、ベクタテーブルに、使用する割り込みのベクタアドレスを登録するだけで個別の割り込み処理が実現できません。

ただし、以下の制限があります。

VBR (ベクタベースレジスタ) は必ず 00000000h (初期値) としてください。  
ベクタテーブルは VMedit で指定したベクタ先頭アドレスから作成されます。(サイズは 1000h です。)

#### 割り込み優先順位の設定について

SH3 (7709) は、割り込みがかかった場合に一部の割り込み要因については優先順位を取得することができません。そのため、VisualMonitor の割り込みハンドラは、多重割り込みを高速に処理するために割り込み優先順位テーブルをRAM上に展開して処理をおこないます。設定はベクタ登録関数で同時におこないます。

初期値は VisualMonitor が使用する割り込み以外は 0 になっています。

#### ユーザプログラムで無条件 TRAP を使用する場合

SH3 では 0 ~ 255 の無条件 TRAP を使用できますが、ベクタアドレスは一つしかないため、VisualMonitor の割り込みハンドラは、前述の割り込みベクタとは別に TRAP 命令のベクタテーブルを用意しています。

ユーザプログラムで無条件 TRAP を使用する場合には、前述のベクタテーブル以外に TRAP 命令ベクタテーブルに使用する TRAP 命令番号のベクタアドレスを登録する必要があります。

登録は無条件 TRAP ベクタ登録関数でおこないます。

なお、TRAP 20h、21h は、ターゲットモニタで使用しますのでユーザプログラムでは使用しないで下さい。

## SH3のベクタ番号

SH3には決められたベクタ番号は存在しませんが、VisualMonitorでは、便宜上ベクタ番号を割り当てています。def\_int\_hdr関数を使用する場合には下記のベクタ番号で設定できます。

VisualMonitorの割り込みベクタテーブル (SH3 7709/09A/09S/29/29R用 IRQモード)

D)

例外要因		ベクタ番号	ベクタテーブルアドレス (ベクタ先頭アドレス)
リセット	パワーオン	0	H'00000000-H'00000003
	マニュアルリセット	1	H'00000004-H'00000007
一般例外	TLBミス/無効 (読み出し)	2	H'00000008-H'0000000C
	TLBミス/無効 (書き込み)	3	H'0000000C-H'0000000F
	初期ページ書き込み例外	4	H'00000010-H'00000013
	TLB保護例外 (読み出し)	5	H'00000014-H'00000017
	TLB保護例外 (書き込み)	6	H'00000018-H'0000001B
	アドレスエラー (読み出し)	7	H'0000001C-H'0000001F
	アドレスエラー (書き込み)	8	H'00000020-H'00000023
	無条件トラップ *1	11	H'0000002C-H'0000002F
	一般不当命令例外	12	H'00000030-H'00000033
	スロット不当命令例外	13	H'00000034-H'00000037
	ユーザブレイクポイントトラップ	15	H'0000003C-H'0000003F
	NMI	14	H'00000038-H'0000003B
	IRQ	IRQ0	48
IRQ1		49	H'000000C4-H'000000C7
IRQ2		50	H'000000C8-H'000000CB
IRQ3		51	H'000000CC-H'000000CF
IRQ4		52	H'000000D0-H'000000D3
IRQ5		53	H'000000D4-H'000000D7
PINT	PINT0~7	56	H'000000E0-H'000000E3
	PINT8~15	57	H'000000E4-H'000000E7
DMAC	DEI0	64	H'00000100-H'00000103
	DEI1	65	H'00000104-H'00000107
	DEI2	66	H'00000108-H'0000010C
	DEI3	67	H'0000010C-H'0000010F
IrDA	ERI1	68	H'00000110-H'00000113
	RXI1	69	H'00000114-H'00000117
	BRI1	70	H'00000118-H'0000011B
	TXI1	71	H'0000011C-H'0000011F
SCIF	ERI2	72	H'00000120-H'00000123
	RXI2	73	H'00000124-H'00000127
	BRI2	74	H'00000128-H'0000012B
	TXI2	75	H'0000012C-H'0000012F
ADC	ADI	76	H'00000130-H'0000003F
TMU0	TUNI0	32	H'00000080-H'0000003F
TMU1	TUNI1	33	H'00000084-H'0000003F
TMU2	TUNI2	34	H'00000088-H'0000003F
	TICPI2	35	H'0000008C-H'0000008F
RTC	ATI	36	H'00000090-H'00000093
	PRI	37	H'00000094-H'00000097
	CUI	38	H'00000098-H'0000009B
SCI	ERI	39	H'0000009C-H'0000009F
	RXI	40	H'000000A0-H'000000A3
	TXI	41	H'000000A4-H'000000A7
	TEI	42	H'000000A8-H'000000AC
WDT	ITI	43	H'000000AC-H'000000AF
REF	RCMI	44	H'000000B0-H'000000B3
	ROVI	45	H'000000B4-H'000000B7

\*1 無条件トラップ例外については前述の「ユーザプログラムで無条件TRAPを使用する場合」を参照して下さい。

VisualMonitorの割り込みベクタテーブル (SH3 SH7709/09A/09S/29/29R用 IRLモード)

例外要因		ベクタ番号	ベクタテーブルアドレス (ベクタ先頭アドレス)	
リセット	パワーオン	0	H'00000000-H'00000003	
	マニュアルリセット	1	H'00000004-H'00000007	
一般例外	TLBミス/無効 (読み出し)	2	H'00000008-H'0000000C	
	TLBミス/無効 (書き込み)	3	H'0000000C-H'0000000F	
	初期ページ書き込み例外	4	H'00000010-H'00000013	
	TLB保護例外 (読み出し)	5	H'00000014-H'00000017	
	TLB保護例外 (書き込み)	6	H'00000018-H'0000001B	
	アドレスエラー (読み出し)	7	H'0000001C-H'0000001F	
	アドレスエラー (書き込み)	8	H'00000020-H'00000023	
	無条件トラップ *1	11	H'0000002C-H'0000002F	
	一般不当命令例外	12	H'00000030-H'00000033	
	スロット不当命令例外	13	H'00000034-H'00000037	
	ユーザブレイクポイントトラップ	15	H'0000003C-H'0000003F	
	割り込み要求	NMI		14 H'00000038-H'0000003B
		IRL 3~0	レベル15	16 H'00000040-H'00000043
レベル14			17 H'00000044-H'00000047	
レベル13			18 H'00000048-H'0000004B	
レベル12			19 H'0000004C-H'0000004F	
レベル11			20 H'00000050-H'00000053	
レベル10			21 H'00000054-H'00000057	
レベル9			22 H'00000058-H'0000005B	
レベル8			23 H'0000005C-H'0000005F	
レベル7			23 H'00000060-H'00000063	
レベル6			24 H'00000064-H'00000067	
レベル5			25 H'00000068-H'0000006B	
レベル4			26 H'0000006C-H'0000006F	
レベル3			27 H'00000070-H'00000073	
レベル2			28 H'00000074-H'00000077	
レベル1			29 H'00000078-H'0000007B	
レベル0			30 H'0000007C-H'0000007F	
IRQ		IRQ4	52 H'000000D0-H'000000D3	
		IRQ5	53 H'000000D4-H'000000D7	
PINT		PINT0~7	56 H'000000E0-H'000000E3	
		PINT8~15	57 H'000000E4-H'000000E7	
DMAC		IRQモードと同じ (前ページ参照)		
IrDA				
SCIF				
ADC				
TMU0				
TMU1				
TMU2				
RTC				
SCI				
WDT				
REF				

\* 1 無条件トラップ例外については前述の「ユーザプログラムで無条件TRAPを使用する場合」を参照して下さい。



VisualMonitorの割り込みベクタテーブル (SH3 SH7708S/R, SH7718R用)

例外要因		ベクタ番号	ベクタテーブルアドレス (ベクタ先頭アドレス)	
リセット	パワーオン	0	H'00000000-H'00000003	
	マニュアルリセット	1	H'00000004-H'00000007	
一般例外	TLBミス/無効 (読み出し)	2	H'00000008-H'0000000C	
	TLBミス/無効 (書き込み)	3	H'0000000C-H'0000000F	
	初期ページ書き込み例外	4	H'00000010-H'00000013	
	TLB保護例外 (読み出し)	5	H'00000014-H'00000017	
	TLB保護例外 (書き込み)	6	H'00000018-H'0000001B	
	アドレスエラー (読み出し)	7	H'0000001C-H'0000001F	
	アドレスエラー (書き込み)	8	H'00000020-H'00000023	
	無条件トラップ *1	11	H'0000002C-H'0000002F	
	一般不当命令例外	12	H'00000030-H'00000033	
	スロット不当命令例外	13	H'00000034-H'00000037	
	ユーザブレイクポイントトラップ	15	H'0000003C-H'0000003F	
	NMI	14	H'00000038-H'0000003B	
	割り込み 要求	I R Q	レベル15	16
レベル14			17	H'00000044-H'00000047
レベル13			18	H'00000048-H'0000004B
レベル12			19	H'0000004C-H'0000004F
レベル11			20	H'00000050-H'00000053
レベル10			21	H'00000054-H'00000057
レベル9			22	H'00000058-H'0000005B
レベル8			23	H'0000005C-H'0000005B
レベル7			23	H'00000060-H'00000063
レベル6			24	H'00000064-H'00000063
レベル5			25	H'00000068-H'00000067
レベル4			26	H'0000006C-H'0000006B
レベル3			27	H'00000070-H'00000073
レベル2		28	H'00000074-H'00000073	
レベル1		29	H'00000078-H'00000077	
レベル0		30	H'0000007C-H'0000007B	
TMU0		TUN I 0	32	H'00000080-H'0000003F
TMU1		TUN I 1	33	H'00000084-H'0000003F
TMU2		TUN I 2	34	H'00000088-H'0000003F
		T I C P I 2	35	H'0000008C-H'0000008F
R T C	A T I	36	H'00000090-H'00000093	
	P R I	37	H'00000094-H'00000097	
	C U I	38	H'00000098-H'0000009B	
S C I	E R I	39	H'0000009C-H'0000009F	
	R X I	40	H'000000A0-H'000000A3	
	T X I	41	H'000000A4-H'000000A7	
	T E I	42	H'000000A8-H'000000AC	
W D T	I T I	43	H'000000AC-H'000000AF	
R E F	R C M I	44	H'000000B0-H'000000B3	
	R O V I	45	H'000000B4-H'000000B7	

\* 1 無条件トラップ例外については前述の「ユーザプログラムで無条件T R A Pを使用する場合」を参照して下さい。

#### 4. 4. 5 ベクタテーブルの登録方法 (SH3)

実際にユーザプログラム上でベクタを登録する方法を説明します。

ソースファイル : GMACHINE.H GMACHINE.C

##### <ベクタテーブルアドレス登録関数>

記述式	<code>void vect_table_address(unsigned long addr);</code>
機能	割り込みベクタテーブルのアドレスを宣言します。
<引数>	(割り込みベクタの先頭アドレス) 例: <code>vect_table_address(0x4000000);</code>
<戻り値>	なし
<使用方法>	ベクタ登録関数に先立って、ベクタテーブルのアドレスを宣言します。

##### <ベクタ登録関数>

記述式	<code>def_int_hdr(unsigned short intno, void (*inthead)(void), unsigned long level);</code>
機能	ベクタテーブルにベクタアドレスと割り込みレベルを登録します。
<引数>	(ベクタ番号、ベクタアドレス、割り込みレベル) 例: <code>def_int_hdr(33, (void *)tmul_5ms, 7);</code>
<戻り値>	なし
<使用方法>	使用する割り込みのアドレスを割り込み処理に先立ち、ユーザプログラム内で登録してください。

##### <無条件 TRAP ベクタ登録関数>

記述式	<code>void def_trap_hdr(unsigned short intno, void (*hdr)(void));</code>
機能	TRAP 命令ベクタテーブルにベクタアドレスを登録します。
<引数>	<code>def_trap_hdr</code> (TRAP 番号、ベクタアドレス) 例: <code>def_trap_hdr(22, (void *)trap22);</code>
<戻り値>	なし
<使用方法>	使用する TRAP 割り込みのアドレスを割り込み処理に先立ち、ユーザプログラム内で登録してください。

## ユーザプログラムのコーディング例

```

#include    "Gmachine3.h"                /*ヘッダファイルの定義*/
#define    VectorTableAdd    0x10000000 /* ベクタテーブルの先頭アドレス*/
void main()
{
    vect_table_address(VectorTableAdd); /*ベクタテーブルアドレスの宣言*/
    TRAP_INIT();                       /*T R A P 割り込み初期化    */
    : (途中省略)
    :
    CMT_timer_init();                  /*CMT タイマ初期化    */
    : (途中省略)
    :
}
/*****/
/*      CMT タイマインテリヤル      */
/*****/
void CMT_timer_init(void)
{
    def_int_hdr( 32, (void*)mtu0_5ms, 7); /* 割り込みハンドラの設定 */
    CMT0.CMCOR = 0x45ea-1;              /* 5 m s タイマ S E T    */
    . (省略)
}
/*****/
/*      T R A P 割り込み初期化      */
/*****/
void TRAP_INIT(void)
{
    def_trap_hdr(22, (void *)trap22);
}
/*****/
/*      M T U 0 タイマ割り込み (5 m s)  */
/*****/
void mtu0_5ms(void)
{
    static char    flag;
    unsigned char  wk;
    . (省略)
}
/*****/
/*      T R A P 2 2 割り込み      */
/*****/
void trap22(void)
{
    static char    flag;
    unsigned char  wk;
    . (省略)
}

```

ベクタ番号 : 32 に mtu0\_5ms ()  
のアドレスを設定

ベクタ番号 : TRAP22 に trap22 ()  
のアドレスを設定

#### 4. 4. 6 VisualMonitorが使用する割り込み (SH3)

VisualMonitorでは以下の割り込みを使用していますので、ユーザプログラムで設定しないでください。

SCIは指定した1つのチャンネルしか使用されませんのでVisualMonitorで使用しないチャンネルはユーザプログラムで使用可能です。(下表以外のベクタ番号は前述の「SH3のベクタ番号」をご覧ください)

<SH3 SH7709、SH7709A/09S、SH7729/29Rの場合>

例外要因		ベクタ番号	ベクタテーブルアドレス (ベクタ先頭アドレス)
ユーザブ레이크ポイントトラップ		15	H'0000003C-H'0000003F
無条件トラップ (20H、21H)		11	H'0000002C-H'0000002F
SCI	ERI	39	H'0000009C-H'0000009F
	RXI	40	H'00000100-H'00000103
	TXI	41	H'00000104-H'00000107
IrDA	ERI1	68	H'00000110-H'00000113
	RXI1	69	H'00000114-H'00000117
	TXI1	71	H'0000011C-H'0000011F
SCIF	ERI2	72	H'00000120-H'00000123
	RXI2	73	H'00000124-H'00000127
	TXI2	75	H'0000012C-H'0000012F

1CHのみ使用

<SH3 SH7708、SH7718Rの場合>

例外要因		ベクタ番号	ベクタテーブルアドレス (ベクタ先頭アドレス)
ユーザブ레이크ポイントトラップ		15	H'0000003C-H'0000003F
無条件トラップ (20H、21H)		11	H'0000002C-H'0000002F
SCI	ERI	39	H'0000009C-H'0000009F
	RXI	40	H'00000100-H'00000103
	TXI	41	H'00000104-H'00000107

## 第5章. デバッグ時の使用ファイル

### 5. 1 フォルダ構成

インストールからユーザプログラムの作成までをおこなうと基本的に以下のフォルダが作成されます。



インストール時に作成される。  
VM2.EXE (デバッガ本体) と  
関連ファイルが格納されている。



ターゲットモニタ作成時に用意  
する作業フォルダ。  
ターゲットモニタをROM化した  
後は使用しない。



ユーザプログラムを格納する  
フォルダ。  
ユーザが任意に作成する。

### 5. 2 デバッグ用ファイル

ユーザプログラムをデバッグするためには、デバッグ用フォルダに●のファイルは必ず収録されている必要があります。

- ソースファイル      ユーザプログラムでリンクされている全てのソースファイル  
フォルダ内にない場合には、デバッグ時にソースファイルが表示されません。
- デバッグ情報ファイル      日立Cの場合は、拡張子が .abs、g c c の場合は、.out および .x (任意に変更可能) です。  
いずれもコンパイル時にデバッグオプションを付けてコンパイルすると生成されます。
- コンフィグレーション  
ファイル      VisualMonitor が参照する設定ファイル(\*.CFG)です。  
ダウンロード時にフォルダ内にない場合には自動的に生成されます。
- HEXファイル      VisualMonitor がダウンロード時に生成する一時ファイルです。

### 5. 3 コンフィグレーションファイル (\*.CFG)

コンフィグレーションファイルは、ソース上でブレーク指定する場合のブレーク方法設定、REALiを使用する場合のREALiウィンドウの有効/無効設定、REALiのタスク名設定を指定するファイルです。

コンフィグレーションファイルは、デバッグ情報ファイルが格納されているフォルダに作成され、ダウンロードする時に参照されます。

#### 5. 3. 1 コンフィグレーションファイルウィンドウ

デバッグ情報ファイルをダウンロードする時に、そのフォルダにコンフィグレーションファイルが存在しない場合にはコンフィグレーションファイルウィンドウが開きます。そして必要項目設定後にOKボタンをクリックすると新規にコンフィグレーションファイルが作成され、ダウンロードが開始されます。

<標準のコンフィグレーションファイルウィンドウ>



- ① ソース上でブレーク指定する場合のブレーク方法を設定します。  
ターゲットのRAM上にダウンロードする場合はTRAP優先として下さい。  
フラッシュメモリを使用する場合はUBC優先として下さい。
- ② REALi モニタウィンドウの有効/無効を設定します。
- ③ REALi のタスク名の変更をします。  
「TASK1=MAINTASK」などのようにキー入力に変更して下さい。

<REALi 使用時のコンフィグレーションウィンドウ>



REALi 使用時に、REALi ウィンドウを有効にするとブレーク時などに大量のタスク情報をターゲットから読み込む為に、処理が若干かかります。  
REALi のモニタが不要な時は REALi ウィンドウのチェックを外して下さい。

REALi モニタウィンドウの有効/無効設定

### 5. 3. 2 コンフィグレーションファイルの変更

コンフィグレーションの設定を変更したい場合には、テキストエディタでコンフィグレーションファイルを変更して下さい。以下にコンフィグレーションファイルのフォーマットを記述します。

#### <コンフィグレーションファイルのフォーマット>

BREAK = UBC	←	ソース上でのブレーク方法指定 (UBCもしくはTRAPを指定)
REALi = ON	←	REALiウィンドウの有効/無効指定 (ONもしくはOFFを指定)
TASK1 = TASK_A		
TASK2 = TASK_B	←	タスク名の指定
TASK3 = TASK_C		指定できるタスク数は最大100タスクまでです。

## 第6章. チュートリアル

### 6. 1 基本操作手順

VisualMonitor の基本操作手順を説明します。

- ① VisualMonitor (VM2. EXE) を起動します。

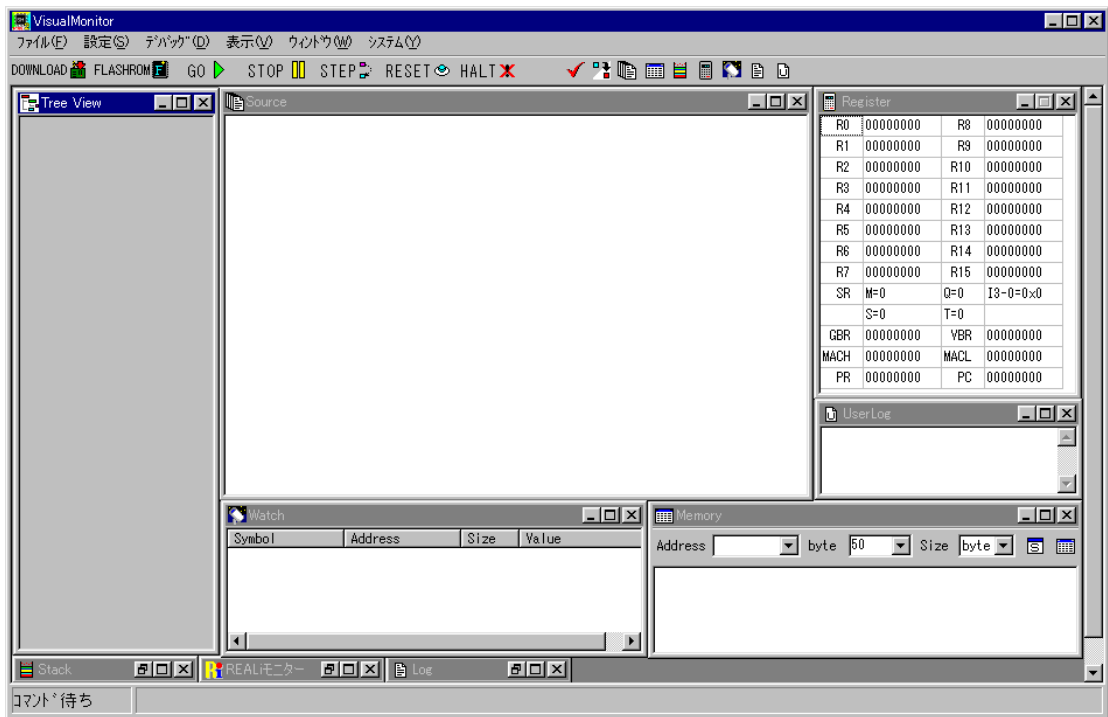
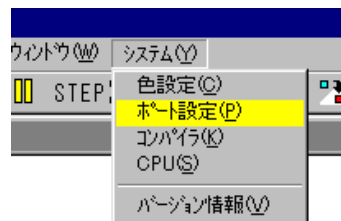
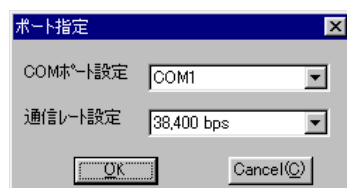


図 6. 1. 1 起動画面 (ウィンドウ位置は前回起動時と同じ位置となります)

- ② PC側のCOMポートの設定をおこないます。次回起動時は、設定する必要はありません。



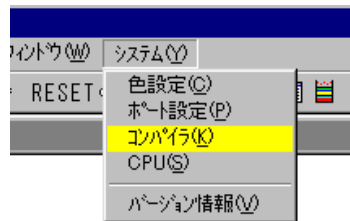
1. システムメニューよりポート設定を選択



2. ポート設定ダイアログが表示されるので使用するCOMポート番号と通信レートを設定し、OKをクリックします。通信レートはターゲットモニタと同じ値に設定してください。



- ③ デバッグ対象となるプログラムのコンパイラを設定します。  
次回起動時は、設定する必要はありません。

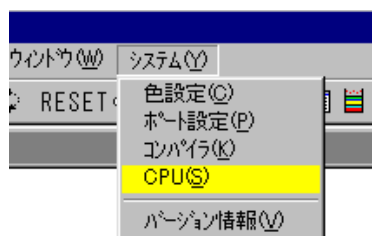


1. システムメニューよりコンパイラを選択します。

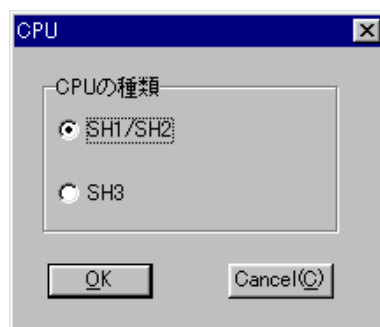


2. GCCか日立Cのいずれか該当する方を選択し、OKをクリックします。

- ② ターゲットのCPUを選択します。  
次回起動時は、設定する必要はありません。

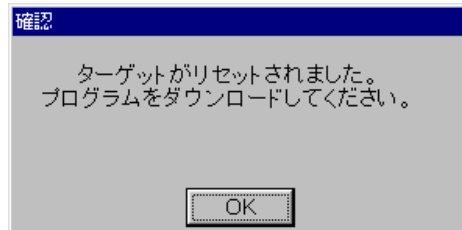


1. システムメニューよりCPUを選択します。



2. SH1 / SH2かSH3の該当する方を選択しクリックします。

- ③ ターゲットをリセット（電源を投入）します。  
リセットすると、以下の確認ダイアログが表示されますのでOKをクリックしてください。



#### ターゲットモニタが起動しない場合

ターゲットをリセットしても、上記のダイアログが表示されない場合には、②～④までの設定項目を再度確認してください。それでも表示されない場合には以下の内容を再度確認してください。

##### ①CPU動作モードの確認

CPUの動作モード端子が正しく設定されているか確認してください。

##### ②ハードウェアの動作確認

RESET信号のタイミングや立ち上がり特性に問題がないか確認してください。  
また、簡単な動作確認プログラムを作成してROMやRAMアクセスの動作を確認してください。

##### ③モニタプログラムの確認

モニタプログラムの作成において設定項目や初期化の間違いがないか確認してください。  
特にアドレスの重複等がないか確認してください。

##### ④RS232Cケーブルの確認

RS232Cケーブルの接続が正しいか確認してください。  
特にPC側のCTS、RTSは正しく処理されているか確認してください。

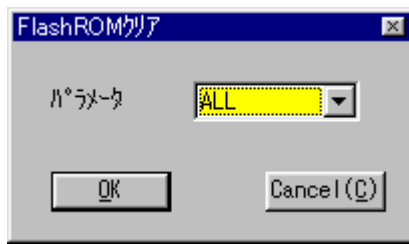
- ④ フラッシュROMをクリアします。  
RAMにユーザプログラムをダウンロードする場合にはこの操作は必要ありません。

外付けのフラッシュROMや内蔵フラッシュROMにユーザプログラムをダウンロードしてデバッグする場合には、ダウンロードする前にフラッシュメモリの内容をクリアする必要があります。

また、内蔵フラッシュROMにダウンロードする場合には、CPUの動作モードをユーザプログラムモードに設定し、FWP端子を解除状態にしておく必要があります。



1. ファイルメニューからフラッシュROMクリア  
もしくはツールバーからFLASHROMを選択  
します。



- FlashROMクリアのダイアログが表示されますので、ALLを選択してOKをクリックします。  
ALLを選択してもモニタ領域は消去されません。  
特定のセクタのみを消去したい場合には、消去するセクタ番号を選択してOKをクリックします。

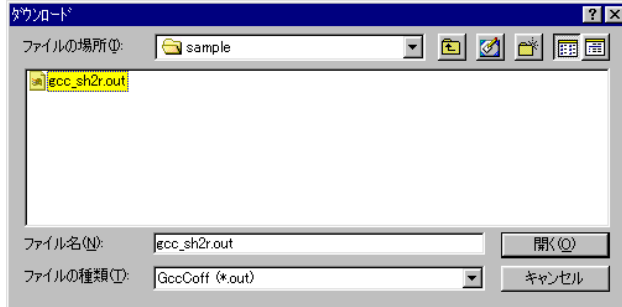
⑦ ユーザプログラムをダウンロードします。



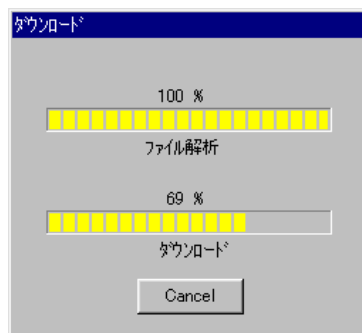
- ファイルメニューもしくはツールバーからダウンロードを選択します。

この時、コンフィグレーションダイアログが表示された場合は、設定をおこなってください。

前述の「コンフィグレーションファイル」を参照



- ファイル選択ダイアログが表示されますのでデバッグするプログラムを選択してOKをクリックします。  
ダウンロード可能なファイルはデバッグ情報ファイルです。  
拡張子のデフォルトは  
日立は \*.ABS  
GCCは \*.OUT  
となっています。



- ダウンロードが開始されると、ダウンロードの状況が表示されます。  
ダウンロード中にCANCELボタンをクリックするとダウンロードを中止します。

## ⑧ ユーザプログラムを実行します。

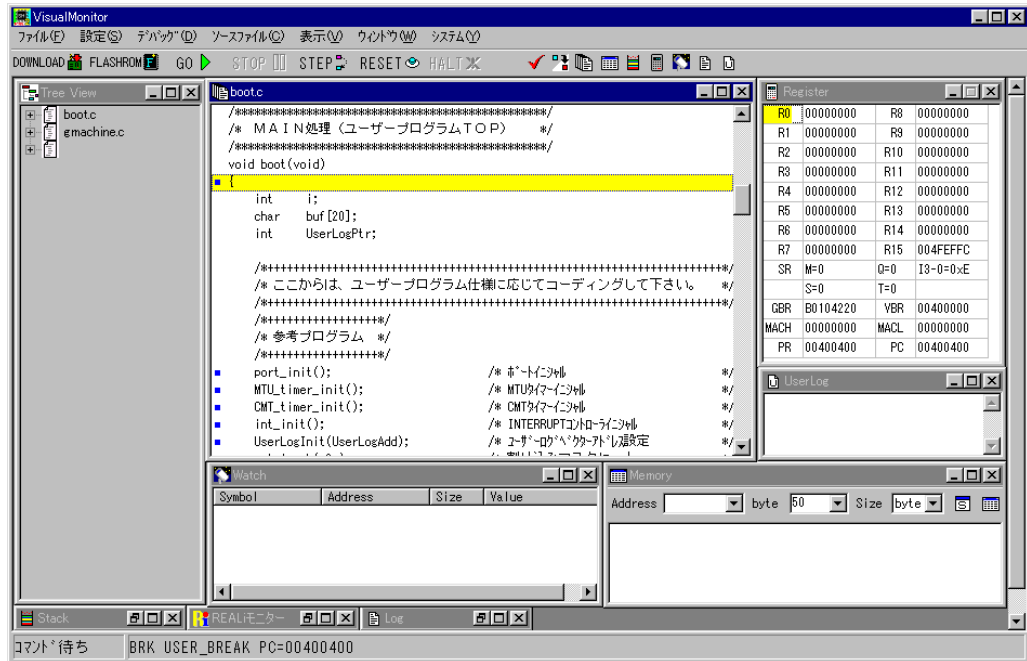


図 6. 1. 8 ダウンロード後の画面

1. ダウンロードが正常に終了すると、PCの値がユーザプログラムの先頭になり、ソースプログラムが表示されます。
2. GOボタンをクリックするとユーザプログラムが実行されます。
3. STOPボタンをクリックするとユーザプログラムが停止します。  
このとき、レジスタウィンドウには停止時の値が表示され、ウォッチウィンドウやメモリウィンドウの設定がある場合には、停止時の値を表示します。

## ⑨ VisualMonitor を終了します。



1. ファイルメニューから終了を選択します。

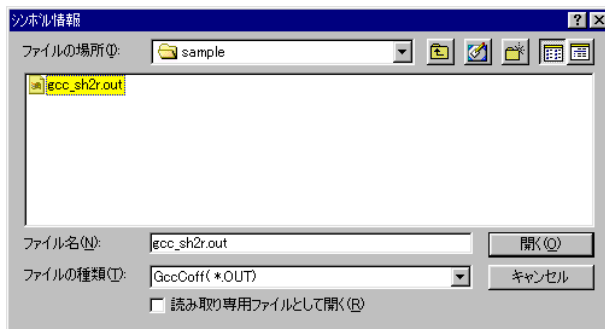
## 6. 2 ROM上のプログラムのデバッグ

既にEPROMなどでプログラムがターゲット上に実装されている場合や、ダウンロード済みの場合には、以下の手順でデバッグをおこないます。

- ① 前項の基本操作手順①～⑤まで（ダウンロードの前まで）は同じ手順を操作します。
- ② シンボル情報を読み込みます。



1. ファイルメニューよりシンボル情報を選択します。



2. ファイル選択ダイアログが表示されますのでデバッグするプログラムを選択してOKをクリックします。  
ダウンロード可能なファイルはデバッグ情報ファイルです。  
デフォルトの拡張子は  
日立Cの場合は \* . ABSファイル  
GCCの場合は \* . OUTファイル  
となっています。

- ③ その後の操作は、基本操作手順と同じです。

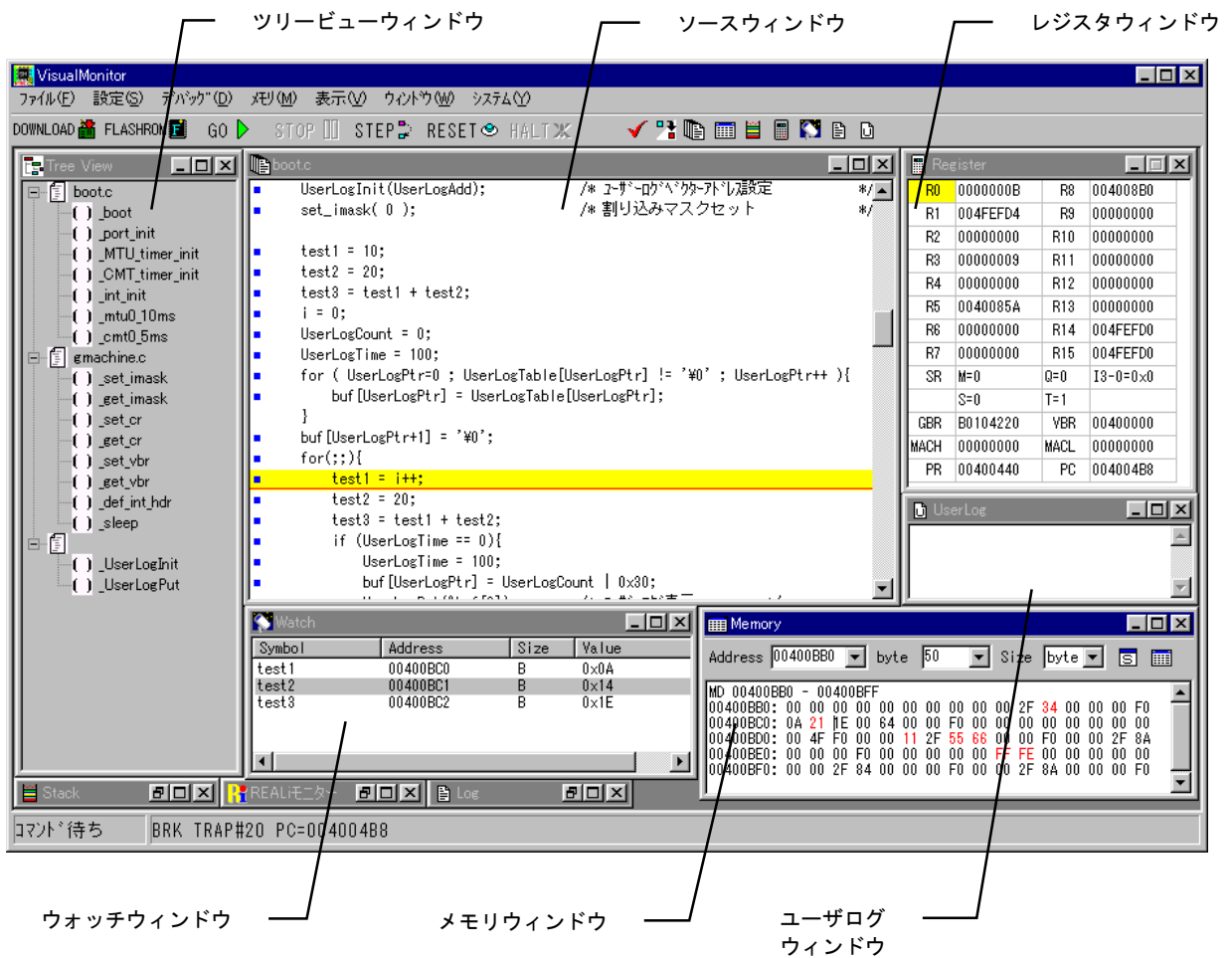
### デバッグ機能の制限

1. ブレークポイント機能はアクセスブレークしか使用できません、したがって、ブレークポイントは1点しか指定できません。  
アクセスブレークについては後述の「ブレークポイント機能」を参照してください。

## 第7章. メニューとウィンドウの操作

### 7. 1 ウィンドウ構成

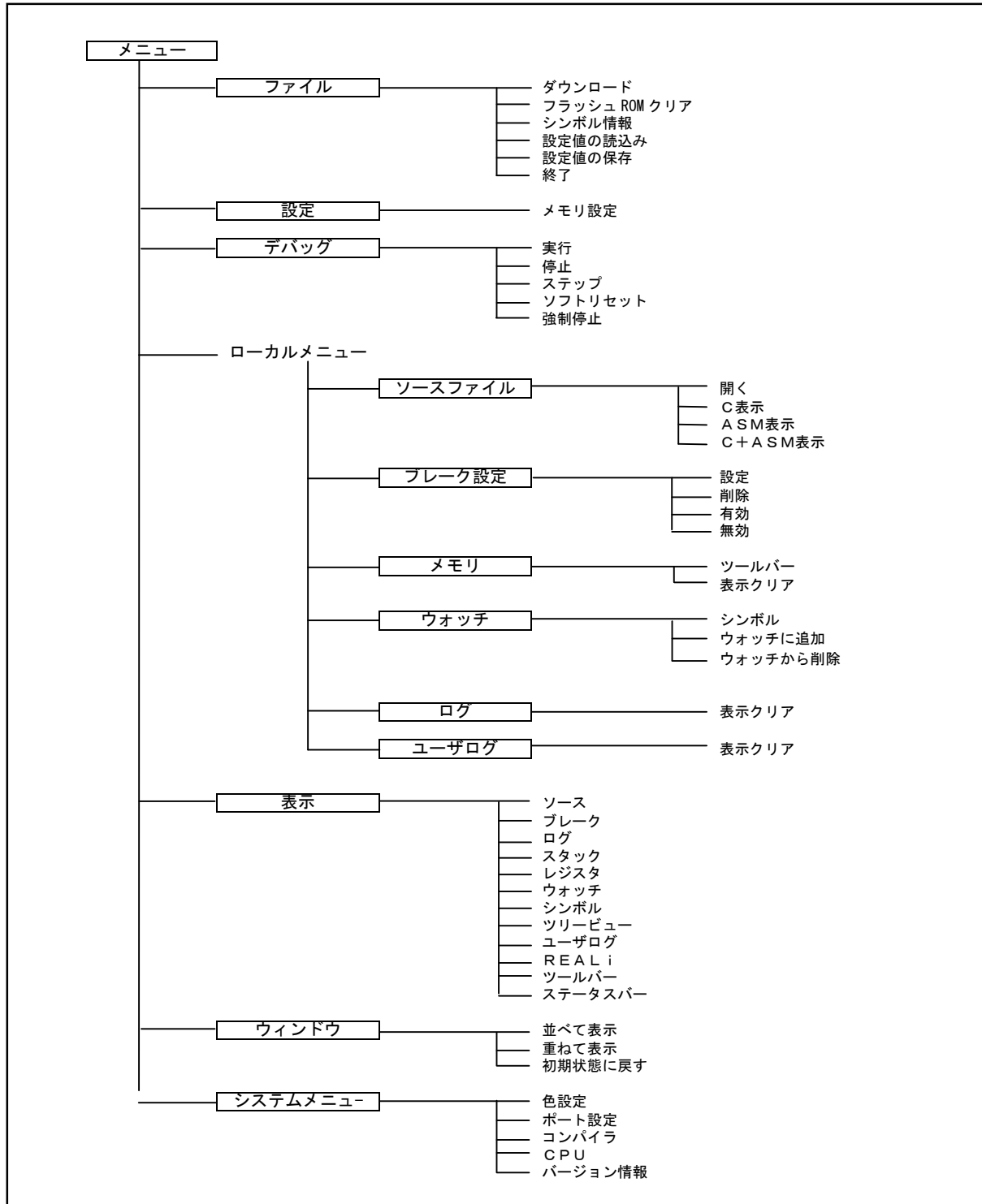
VisualMonitor の主なウィンドウには以下のものがあります。



## 7.2 メニュー

メニューバーに、各コマンドがグループ化されています。以降に各コマンドの説明を記述します。

図 7.2 メニューコマンド一覧



### 7. 2. 1 ファイルメニュー (Alt+F)

ファイルメニューには、ファイル関連項目とフラッシュROM消去の項目が用意されています。

メニュー項目	ショートカット	機能概要
ダウンロード (L)	—	ユーザが作成したデバッグ情報ファイルを読み込む為のダイアログボックスを表示する。
フラッシュROMクリア (F)	—	フラッシュROMエリアをクリアする為のダイアログボックスを表示する。 チップ消去/セクタ消去の指定が可能。
シンボル情報 (S)	—	シンボル情報を読み込む。 既にプログラムはダウンロード済み、もしくはROM上にある場合に使用する。
設定値の読み込み		設定値をファイルから読み込む。
設定値の保存		設定値をファイルに保存する。
終了 (X)	—	VisualMonitor を終了する。

### 7. 2. 2 設定メニュー (Alt+S)

設定メニューには、メモリ設定の設定項目が用意されています。

メニュー項目	ショートカット	機能概要
メモリ設定 (M)	—	メモリ設定ダイアログボックスを表示する。

### 7. 2. 3 デバッグメニュー (Alt+D)

デバッグメニューには、ユーザプログラムの実行/停止項目が用意されています。

メニュー項目	ショートカット	機能概要
実行 (R)	F 5	プログラムの実行を行う。
停止 (A)	E S C	プログラムの強制停止を行う。
ステップ (T)	F 9	プログラムのステートメントをステップ実行させる。
ソフトリセット (I)	F 1	PCとSPを初期値に戻し、ソフト的にリセット状態にする。
強制停止	F 1 2	プログラムを強制停止する。

### 7. 2. 4 ローカルメニュー

ローカルメニューとは、各ウィンドウがアクティブになったときだけ表示されるメニューです。

詳細については、各ウィンドウの操作説明を参照してください。



### 7. 2. 5 表示メニュー (Alt+V)

表示メニューには、各ウィンドウの表示指定関連の項目が用意されています。

メニュー項目	ショートカット	機能概要
ブレーク (B)	—	ブレークウィンドウを表示する。
ソース (S)	—	ソースウィンドウを表示する。
ログ (L)	—	ログウィンドウを表示する。
メモリ (D)	—	メモリダンプウィンドウを表示する。
スタック (T)	—	スタックウィンドウを表示する。
レジスタ (R)	—	レジスタウィンドウを表示する。
ウォッチ (W)	—	ウォッチウィンドウを表示する。
シンボル (Y)	—	シンボル情報指定ウィンドウを表示する。
ツリービュー (V)	—	ツリービューウィンドウを表示する。
ユーザログ	—	ユーザログウィンドウを表示する。
REALi (I)	—	REALi モニタウィンドウを表示する。
ツールバー (O)	—	ツールバーの表示/非表示を行う。 チェックマークが付いている場合にはツールバー表示が有効である。
ステータスバー (A)	—	ステータスバーの表示/非表示を行う。 チェックマークが付いている場合にはステータスバー表示が有効である。

### 7. 2. 6 ウィンドウメニュー (Alt+W)

ウィンドウメニューには、ウィンドウの制御の項目が用意されています。

メニュー項目	ショートカット	機能概要
並べて表示 (T)	—	ウィンドウを並べて表示させる。
重ねて表示 (C)	—	ウィンドウを重ねて表示させる。
初期状態に戻す (R)	—	ウィンドウ状態を初期状態に戻す。

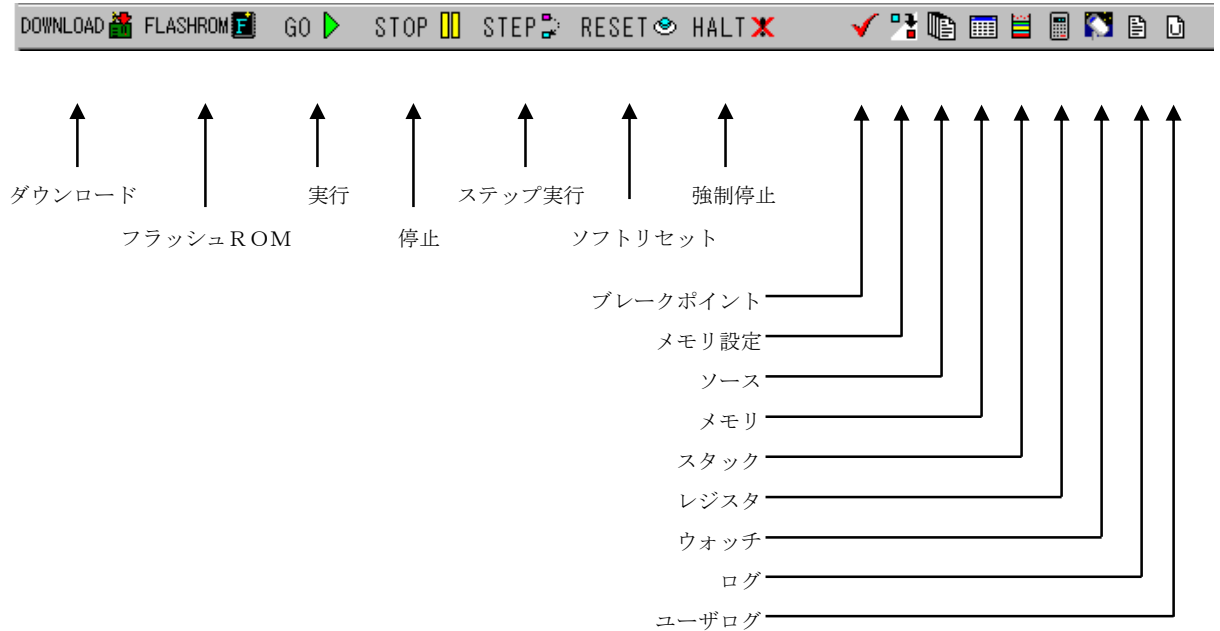
### 7. 2. 7 システムメニュー (Alt+F)

システムメニューには、設定項目及び VisualMonitor のバージョン情報表示の項目が用意されています。

メニュー項目	ショートカット	機能概要
色設定 (C)	—	各ウィンドウの色及びフォント指定ダイアログボックスを表示する。
ポート設定 (P)	—	COMポート設定のダイアログボックスを表示する。
コンパイラ (K)	—	コンパイラを選択する
CPU (S)	—	CPUを選択する。
バージョン情報 (V)	—	VisualMonitor のバージョン情報ダイアログを表示する。

### 7. 3 ツールバー

ツールバーには、VisualMonitor でよく使用されるコマンドをボタン化したものが用意されています。



### 7. 4 ステータスバー

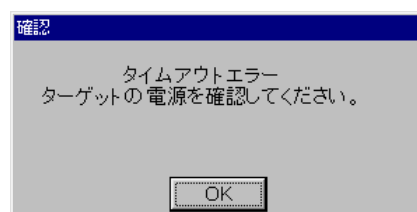
ステータスバーには、各コマンドの実行状態が表示されます。

ステータスバーは、[表示]メニューの[ステータスバー]にて表示/非表示の選択が可能です。



### 7. 5 確認ダイアログ

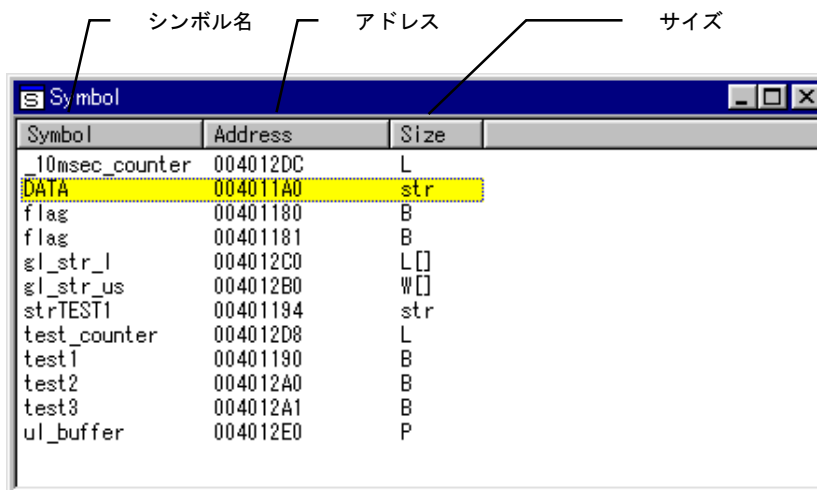
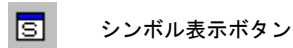
動作確認メッセージやエラーメッセージなどを表示するダイアログです。



## 7. 6 シンボル入力

シンボル入力はウォッチウィンドウやブレイクポイントダイアログ等のアドレス入力が必要な機能に用意されており、アドレスの入力補助に使用します。

各ウィンドウのシンボル表示ボタンをクリックするとシンボル情報の一覧が表示されます。



シンボル情報ダイアログ

### <表示>

項目	表示内容
Symbol	シンボル名を表示します。 グローバルシンボルのみ表示されます。
Address	シンボルのアドレスを表示します。
size	シンボルのサイズを表示します。

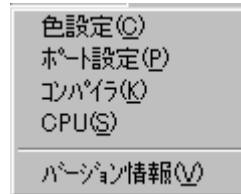
### <操作>

入力したシンボルの行をダブルクリックすると、該当するアドレスが各操作ウィンドウに入力されます。

## 7.7 システム設定

デバッグをおこなうためには、基本的な環境を設定する必要があります。

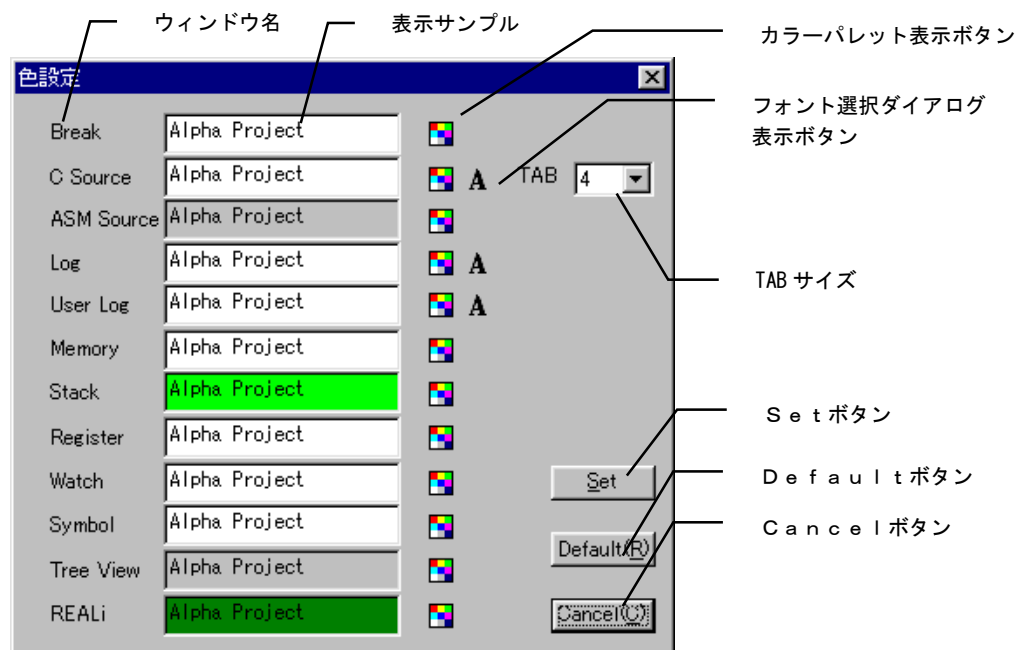
それらの設定はシステムメニューからおこないます。



システムメニュー

### 7.7.1 色設定

色設定では各ウィンドウの背景色やフォント等を指定します。



色設定ダイアログ

#### <表示>

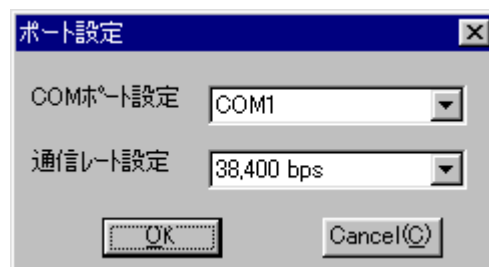
コントロール	機能
ウィンドウ名	対応する各ウィンドウ名を表示します。
表示サンプル	選択した設定でのサンプルが表示されます。

## &lt;設定項目&gt;

コントロール	機能
カラーパレット 表示ボタン	色選択用のカラーパレットを表示します。
フォント選択ダイアログ 表示ボタン	フォント選択用のダイアログを表示します。
TAB サイズ	ソースウィンドウに表示する際のTABサイズを指定します。
Set ボタン	設定を有効にします。
Default ボタン	全ての設定を初期値に戻します。
Cancel ボタン	設定を無効にし、色設定ダイアログを閉じます。

## 7. 7. 2 ポート設定

ホストコンピュータの使用するCOMポートの設定をおこないます。



ポート設定ダイアログ

## &lt;設定項目&gt;

コントロール	機能
COMポート設定	使用するCOMポートを選択します。
通信レート設定	通信レートを選択します。必ずターゲットモニタと同じ値にしてください。

## 7. 7. 3 コンパイラ設定

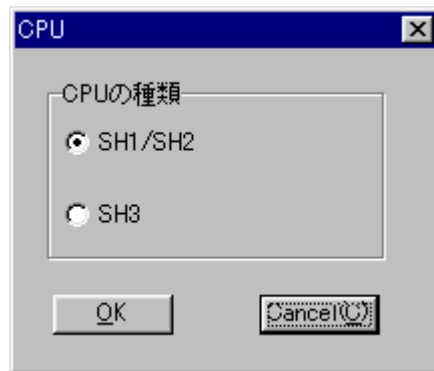
ユーザプログラムのコンパイラを設定します。



コンパイラ設定ダイアログ

#### 7. 7. 4 CPU設定

ターゲットのCPUの種類を指定します。



CPU設定ダイアログ

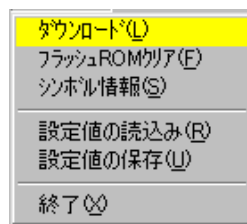
## 7.8 プログラムのダウンロード

ユーザプログラムをターゲットのRAMもしくはフラッシュROMにダウンロードする機能です。

シンボルデバッグが可能なデバッグ情報ファイルダウンロードが可能です。

### <操作方法>

- ① メニューもしくはツールバーからダウンロードを選択します。

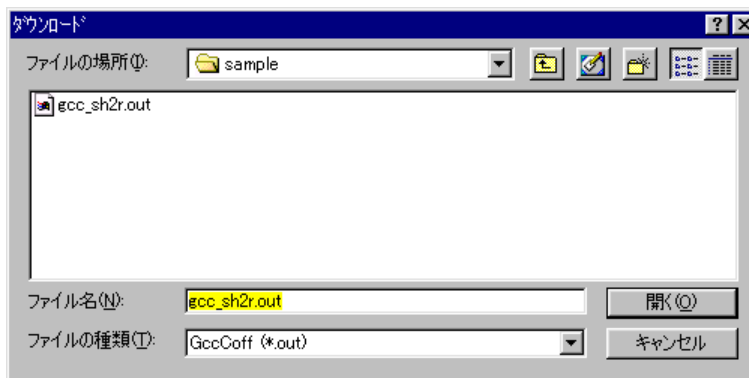


ファイルメニュー



ツールバー

- ② ファイル選択ダイアログが表示されますので、デバッグするプログラムファイルを選択します。

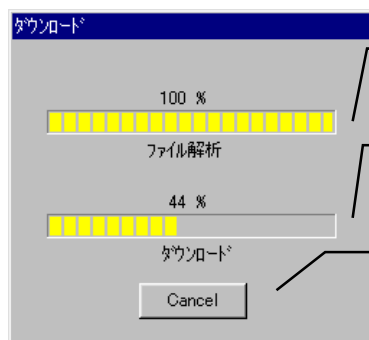


ダウンロード可能なファイルは、デバッグ情報ファイルです。

日立Cは .ABS、gccは .out および .x の拡張子がデフォルトになっていますが、名称は特に関係ありません。

コンパイル時には必ずデバッグオプションを付けてください。

- ② ダウンロードが開始されると状況が表示されます。



解析状況の表示

ダウンロードの  
進捗表示

キャンセルボタン

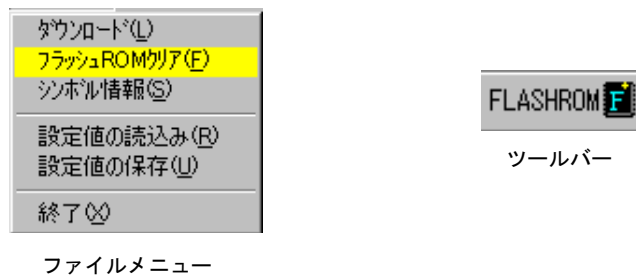
VisualMonitor は選択されたファイルのデバッグ情報を解析した後にダウンロードを開始します。ダウンロード時間は、デバッグ環境によって変化しますが、100K以上のプログラムの場合には1分以上かかる場合もあります。ダウンロードを高速におこなうため、できるだけ高い通信レートでデバッグされることをお勧めします。(ターゲットモニタの作成を参照)

## 7.9 フラッシュROMのクリア

フラッシュROM操作機能は、指定した範囲のフラッシュROMもしくは内蔵フラッシュを消去します。

### <操作方法>

- ① ファイルメニューもしくはツールバーからフラッシュROMを選択します。



- ② 消去単位を選択します。ALLの場合はチップ全体、番号の場合は対応する番号のセクタ (F-ZTATの場合はEB) が消去されます (注)。OKをクリックすると消去が始まります。



注) F-Z T A Tを使用する場合、通常、E B 0 (SH7055 はEB0~EB7) に VisualMonitor を配置します。  
したがって、A L Lを指定した場合はE B 0は消去されません。  
ただし、番号指定で0を指定した場合には消去されますので注意してください。

### フラッシュROMの使用条件

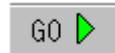
- ① 外付フラッシュROMは対応チップのみ使用可能です。  
後述のハードウェア「9. ハードウェアの制限と設計例」をご覧ください。
- ② 内蔵フラッシュ (F-Z T A T) を使用するには、あらかじめ VisualMonitor のターゲットモニタを書込ツール (弊社製 FLASH WRITER もしくは日立製書き込みツール等) を使用して書き込んでおきます。  
書き込んだ後、MODE 端子及び FWP 端子の設定をユーザプログラムモードに設定してデバッグしてください。



## 7. 10 プログラムの実行と停止

ユーザプログラムの実行や停止を操作する基本機能です。

### 7. 10. 1 実行



実行ボタン

プログラムを実行します。STOPボタンがクリックされるまでユーザプログラムは実行されます。

### 7. 10. 2 停止



ストップボタン

実行中のプログラムを停止します。

### 7. 10. 3 ステップ実行



ステップボタン

プログラムのCソース及びアセンブラソースの各ステートメントをステップ実行する機能です。ステップ実行単位はソースウィンドウの表示状態によって切り替わります。

#### <Cソース表示でのステップ実行>

- Cソースの各ステートメントを実行します。
- 関数の実行は、関数内に入りステップ実行していきます。
- Cソースファイルがない関数の実行は、アセンブラソース表示に自動的に切り換わり関数内のステップ実行を行います。関数を抜け出すとCソース表示に戻ります。
- 最適化されているプログラムでは、Cソース表示上でステップ実行されないステートメントが発生する場合があります。
- 組み込み関数（マクロ等）は、その関数を1ステップとしてステップ実行します。

#### <アセンブラ表示でのステップ実行>

- アセンブラの各ステートメントを実行します。

#### <混在表示でのステップ実行>

- アセンブラの各ステートメントを実行します。

### ステップ実行使用時の注意

ステップ実行は、U B Cを使用する為、幾つかの制限があります。

- ・非遅延分岐命令の次の命令はU B Cの制約上ブレークがかからない為、ステップ実行を行った時に2命令以上のステップ実行をする場合があります。
- ・割り込み禁止命令のステップ実行はブレークがかからない為、ステップ実行を行った時に2命令以上のステップ実行をする場合があります。

※ U B Cの詳細は各CPUのハードウェアマニュアルを参照して下さい。

## 7. 10. 4 RESET



リセットボタン

ソフトウェアリセット機能はソフト的にP CとS Pの値を初期値に戻す機能です。パワーオンリセットではないため、周辺の内蔵レジスタなどは初期値には戻りません。その点に留意すれば、プログラムを最初から実行し直す場合などに便利です。

## 7. 10. 5. 強制停止



強制停止ボタン

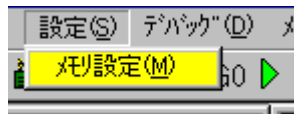
実行中のプログラムを強制停止します。実行中やステップ実行の際にプログラムの停止に時間がかかる場合や、停止しなかった場合に強制的にプログラムを停止させます。

## 7. 11 メモリの設定

メモリの設定は、指定範囲を指定値で書き換える機能です。  
メモリの初期化や、広範囲な空間を書き換える場合に有効です。

### <操作方法>

- ① 設定メニューからメモリ設定を選択します。



- ② Memory Fill か Memory Set を選択し、アドレスと値とサイズを入力し、OKをクリックします。  
Memory Fill は範囲内のメモリを全て指定値で書き換え、Memory Set は指定アドレスのメモリを書き換えます。



### <メモリ設定ダイアログ>

項目	入力
Memory Fill/Memory Set	範囲指定(Memory Fill)か単一アドレス指定(Memory Set)か選択します。
Start Address (Address)	範囲の開始アドレスを16進数で指定します。
End Address	範囲の最終アドレスを16進数で指定します。(Memory Set時はなし)
Value	書き込む値を指定します。 16進数の場合は0xから、それ以外は10進数とみなされます。
Size	書き換え単位を指定します。(B:1バイト、W:2バイト、L:4バイト) 例えば、Value=FFでSize=Wの場合には、00FFで書き換えられます。

## 7. 12 設定値の保存と読み込み

設定値の読み込みと保存とは、デバッグ中の各ウィンドウの設定状態をファイルから読み込んだり、保存する機能です。同じプログラムを繰り返しデバッグをする場合などに、便利な機能です。

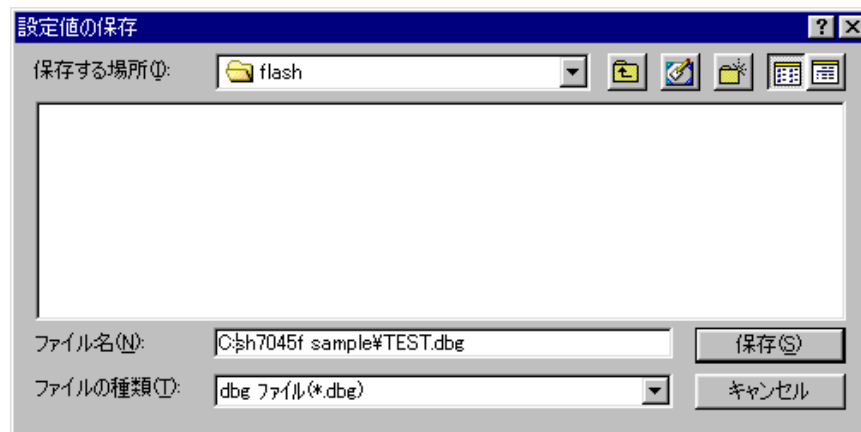
### 7. 12. 1 設定値の保存

<操作方法>

- ① ファイルメニューから設定値の保存を選択します。



- ② ファイル選択ダイアログが表示されるので、保存場所とファイル名を入力し保存をクリックします。ファイル名の拡張子は\*.dbgです。



保存される設定値は以下の設定値です。

機能	保存内容
ソースウィンドウ	ソースの表示状態
ブレークポイント	全ブレークポイントの設定
ウォッチウィンドウ	ウォッチ変数の登録状態
メモリウィンドウ	表示範囲のアドレス

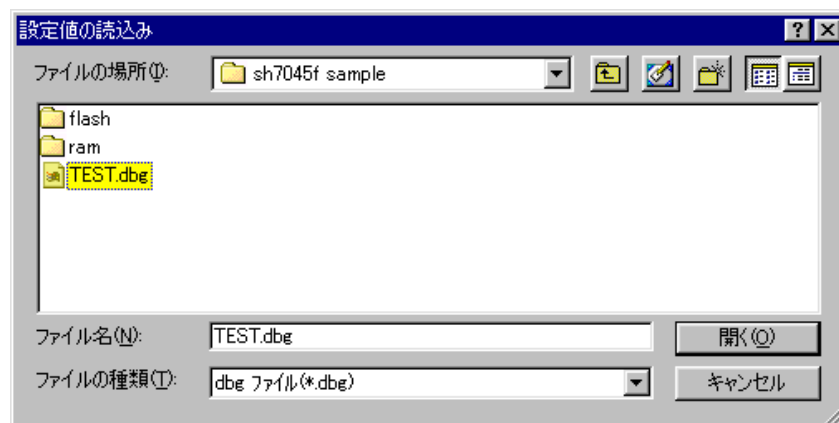
## 7. 12. 2 設定値の読み込み

<操作方法>

- ① ファイルメニューから設定値の保存を選択します。



- ② ファイル選択ダイアログが表示されるので、ファイルを選択し開くをクリックします。  
ファイル名の拡張子は\*.dbgです。



### 設定値の注意

基本的に設定値が有効なのは、プログラムに変更が加えられていない場合です。  
プログラムに変更が加えられている場合や、アドレスが変更されている場合などは、正しく動作しませんので注意してください。

## 7. 13 ツリービューウィンドウ

### 7. 13. 1 ツリービューウィンドウの機能

ツリービュー機能は、ソースファイル情報やそのソースファイル内で定義された関数の情報をツリー表示する機能です。

ツリー上のファイル名や関数名をクリックするとソースウィンドウに該当するソース位置が表示されます。

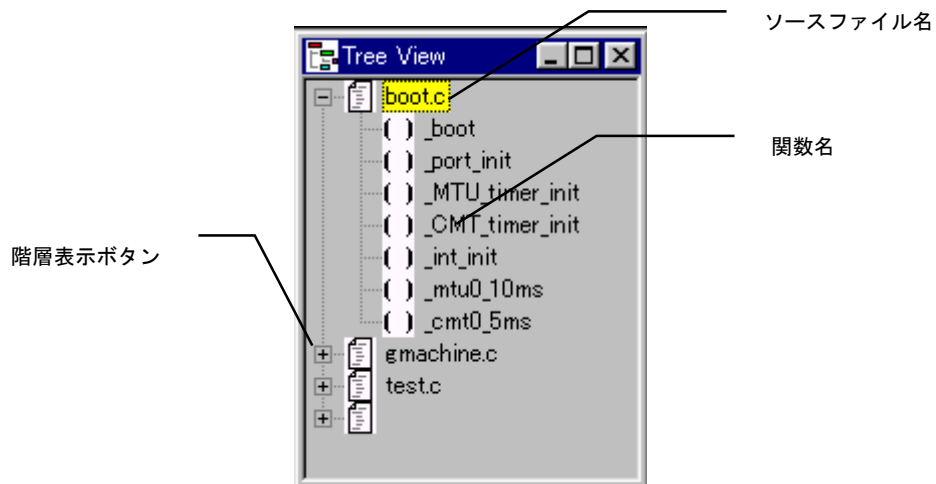


図7. 13 ツリービューウィンドウ

#### <表示>

ダウンロードしたプログラムのファイルと関数をツリー構造で一覧表示します。

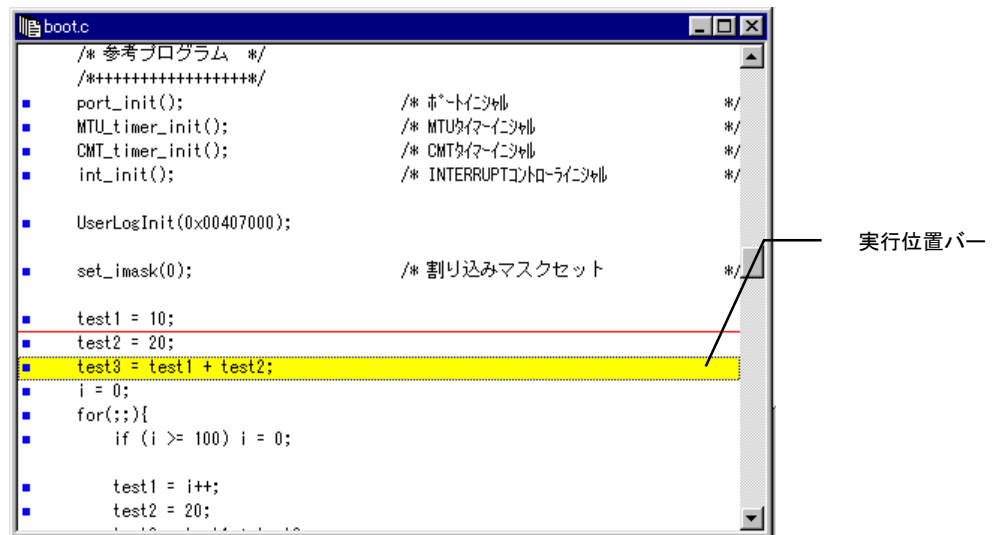
#### <マウス操作>

コントロール	機能
階層表示ボタン	クリックで下の階層（関数）を展開表示します。
関数名	ダブルクリックで該当する関数をソースウィンドウに表示します。

## 7. 14 ソースウィンドウ

### 7. 14. 1 ソースウィンドウの機能

ソースウィンドウには、Cソースファイル表示、アセンブラソースファイル表示、混在表示が選択できます。ソースウィンドウ上でブレークポイントの設定や変数の表示などの操作をおこないます。



#### <表示>

デバッグ対象となるプログラムのソースファイルを表示します。

表示は、Cソース表示、アセンブラ表示、C+A S M表示を選択できます。

停止直後は自動的に停止位置が参照されますが、停止時には任意のソースファイルを表示することができます。

#### <ローカルメニュー操作>

メニュー項目	ショートカット	機能
開く (O)		任意のファイルを表示します。
C表示 (C)	—	Cソースを表示します。
A S M表示 (A)	—	アセンブラソースを表示します。
C+A S M表示 (M)	—	Cとアセンブラソースを混在表示します。

#### <マウス操作>

ポインタ位置	機能
ソース行	右クリックでブレークポイント設定のプルダウンメニューが表示され、ブレークポイントの設定/削除/有効/無効が操作できます。 詳細は「ブレークポイントの設定方法」を参照してください。
変数名	変数名をダブルクリックすると、変数の値が表示されます。 詳細は「ダイレクト変数表示機能」を参照してください。

## 7. 14. 2 ブレークポイントの設定方法

ソースウィンドウ上でブレークポイントの設定、削除、有効、無効の指定できます。

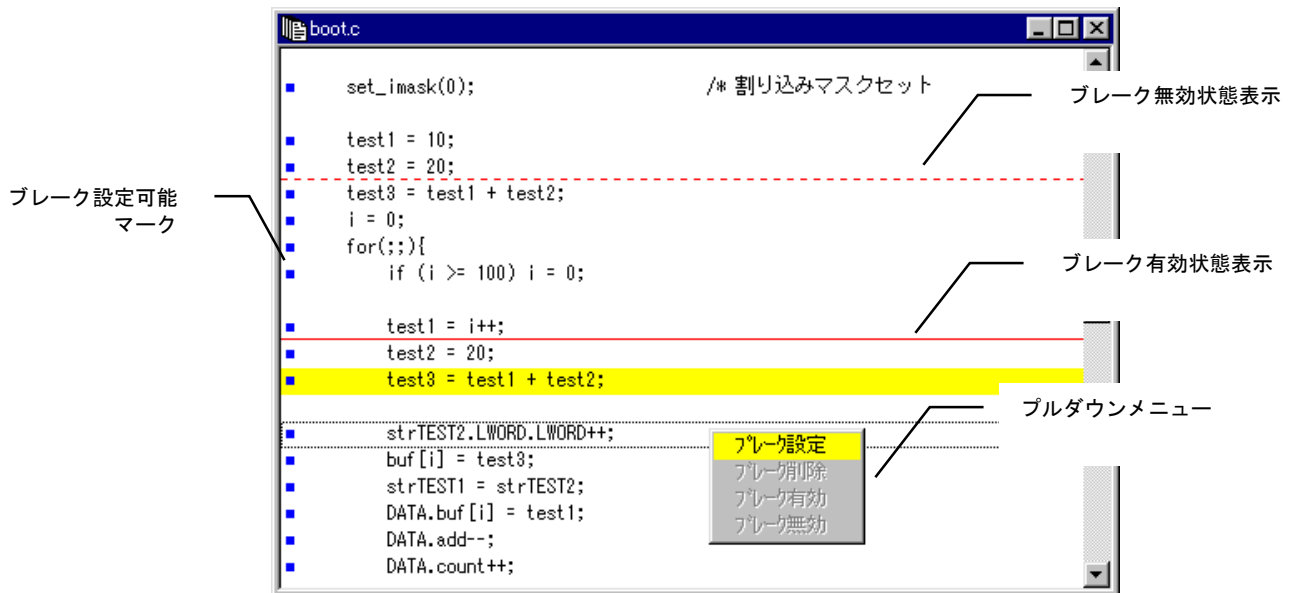


図 7. 14. 2 ソースウィンドウのブレーク設定

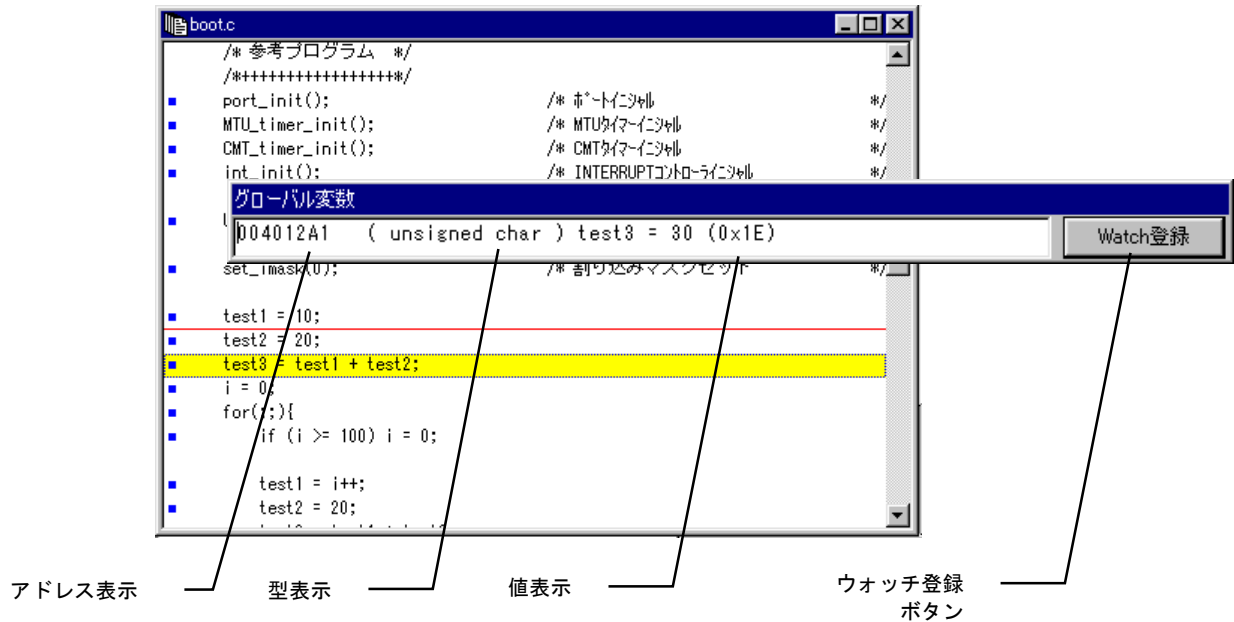
### <操作方法>

- ①ソース行の左側にブレーク可能マークが表示されている行にマウスカーソルを合わせ、右クリックします。
- ②プルダウンメニューが表示されるので、ブレーク設定を選択しマウスボタンを離すとブレークポイントが設定され、赤いアンダーラインが表示されます。
- ③既に設定されている行を右クリックすると、削除、有効、無効が選択できます。
- ④無効にした場合は、一時的にブレークポイントが解除され赤の点線に変わります。
- ⑤コンフィグレーションでブレーク設定がU B C優先になっている場合には、1箇所しかブレークポイントを有効にできないため、新たにブレークポイントを設定した場合には、他のブレークポイント箇所は全て無効状態になります。
- ⑥設定可能なブレークポイントは最大で10箇所です。



### 7. 14. 3 ダイレクト変数表示機能

ソースウィンドウ上の変数をダブルクリックして変数の内容を表示することができます。



#### <操作方法>

変数名にポインタを合わせダブルクリックすると変数の値を表示します。

グローバル変数とローカル変数に対応しています。

配列変数は対応していません（アドレス表示のみ）のでメモリ表示／編集機能と組み合わせてお使いください。

表示は他の操作をすると自動的に閉じます。

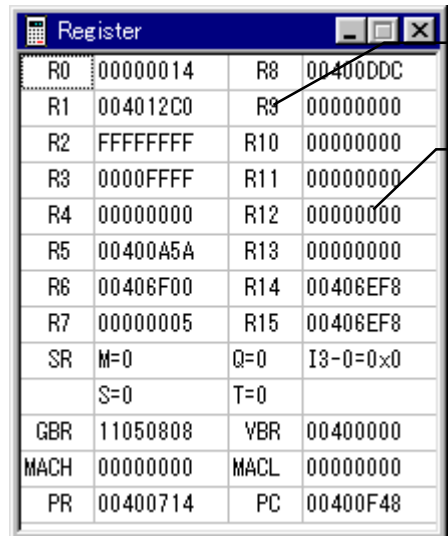
#### <表示>

コントロール	機能
変数名	ダブルクリックで変数の型と値を表示します（10進と16進）。
ウォッチ登録ボタン	クリックでウォッチウィンドウに変数を登録します。 グローバル変数の場合だけ表示されます。

## 7. 15 レジスタウィンドウ

### 7. 15. 1 レジスタウィンドウの機能

レジスタウィンドウには、現在のレジスタ値が表示されます。  
また各レジスタの値を書き換えることが可能です。



Register			
R0	00000014	R8	00400DDC
R1	004012C0	R9	00000000
R2	FFFFFFFF	R10	00000000
R3	0000FFFF	R11	00000000
R4	00000000	R12	00000000
R5	00400A5A	R13	00000000
R6	00406F00	R14	00406EF8
R7	00000005	R15	00406EF8
SR	M=0	Q=0	I3-0=0x0
	S=0	T=0	
GBR	11050808	YBR	00400000
MACH	00000000	MACL	00000000
PR	00400714	PC	00400F48

レジスタ名

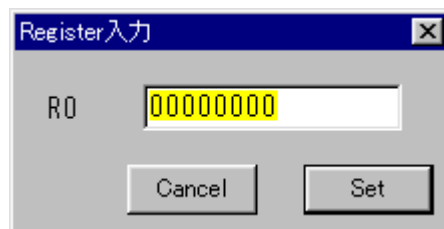
レジスタ値

#### <表示>

CPU内部のレジスタ値を16進表示します。常に最新の状態を自動的に表示します。  
SH1/2とSH3とはレジスタ構成が違うため、表示が異なります。  
FPU内蔵のSH2E、SH3Eの場合は、FR0～FR15も表示されます。  
DSP内蔵のCPUの場合は、DSPレジスタも表示されます。

#### <マウス操作>

ポインタ位置	機能
レジスタ値	ダブルクリックでレジスタ値設定ウィンドウが開き、値を変更できます。



レジスタ値を16進で入力します。

レジスタ値設定ウィンドウ

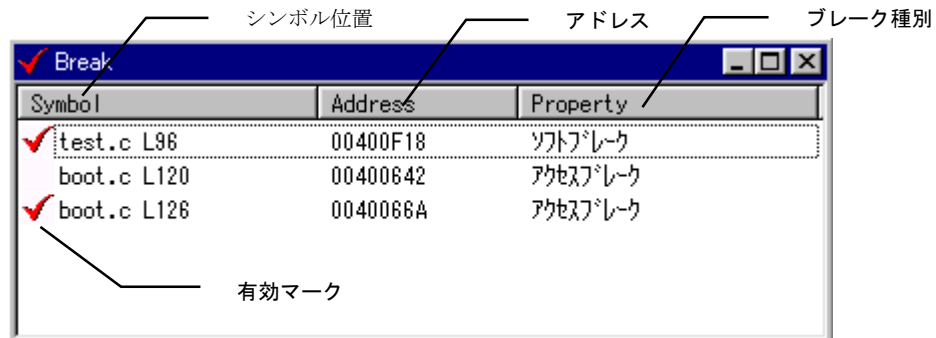
## 7. 16 ブレークポイントウィンドウ

### 7. 16. 1 ブレークポイントウィンドウの機能

現在のブレークポイントの設定を一覧表示します。

また、さらに詳細なブレークポイント条件の設定やブレークポイントの無効化や削除などもおこなえます。

ブレークポイントの設定はソースウィンドウ上で可能です。詳細はソースウィンドウを参照してください。



#### <表示>

項目	表示内容
有効マーク	ブレークが有効になっている場合は赤いチェックマークが表示します。
Symbol	ブレークが設定されているシンボルがコード部の場合、該当するソースファイル名と行番号を表示します。
Address	ブレークが設定されているシンボルのアドレスを表示します。
Property	ブレーク種別を表示します。

#### <ローカルメニュー操作>

メニュー項目	ショートカット	機能
設定 (A)	—	ブレークポイント設定のダイアログボックスを表示され、新規にブレークポイントを追加できます。
削除 (C)	—	設定されたブレークポイントの削除します。
有効 (E)	—	設定されたブレークポイントを有効にします。
無効 (D)	—	設定されたブレークポイントを無効にします。

#### <マウス操作>

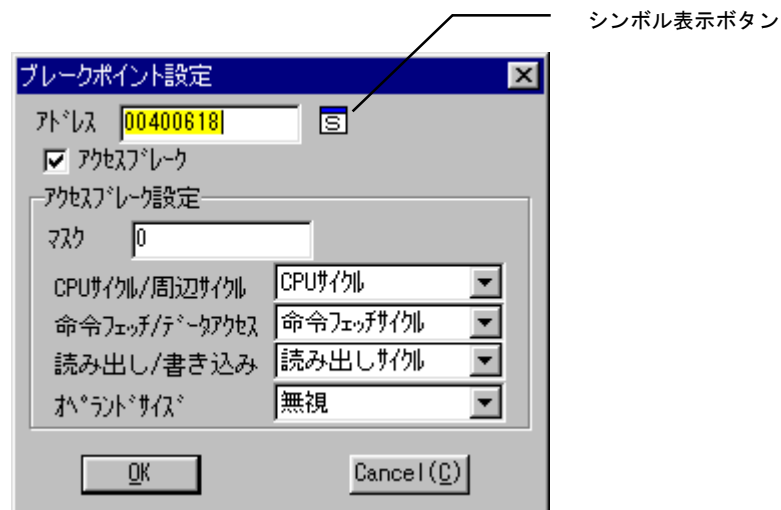
ポインタ位置	機能
表示行	既に表示されている行をダブルクリックするとブレークポイント設定ダイアログが表示されブレーク条件を変更できます。 右クリックするとプルダウンメニューが表示され設定/削除/有効/無効とソースが選択できます。設定/削除/有効/無効はローカルメニューと同じ機能です。ソースはソースウィンドウに該当行を表示します。
空白行	空白の行をダブルクリックするとブレークポイント設定ダイアログが表示されブレークポイントを追加できます。

## &lt;キーボード操作&gt;

操作	機能
DELキー	設定されている行を選択し、DELキーを押下すると設定が削除されます。

## 7. 16. 2 詳細なブレイク条件の設定

ブレイクポイントウィンドウで、より詳細なブレイクの条件を設定する場合にはブレイクポイント設定ダイアログで設定します。ローカルメニューの「設定」もしくはマウス操作で変更、追加操作をおこなった場合に表示されます。



## &lt;シンボル表示ボタン&gt;

前述の「シンボル表示入力」を参照してください。

## &lt;設定項目&gt;

項目	表示内容	
アドレス	ブレイクポイントを設定するアドレスを入力します。	
アクセスブレイク	アクセスブレイクを使用する場合にチェックします。	
設定	マスク	CPU内部のUBCのユーザブレイクアドレスマスクレジスタの値を16進で入力します。
	CPUサイクル/周辺サイクル	CPUサイクル/周辺サイクルの条件を選択します。
	命令フェッチ/データアクセス	命令フェッチ/データアクセスの条件を選択します。
	読み出し/書き込み	読み出し/書き込みサイクルの条件を選択します。
	オペランド	オペランドサイズの条件を選択します。

詳細は、各CPUデータシートの「ユーザブレイクコントローラ」を参照してください。

### 7. 16. 3 ブレークポイントの注意と制限

ブレークポイントには、幾つか注意と制限事項があります。

ブレークは、アクセスブレーク（UBC使用）とソフトウェアブレーク（TRAP#20使用）の2つがあります。

#### <ブレークポイントの制限>

- ・ROM、フラッシュROM上に存在するユーザプログラムは、アクセスブレークのみ使用可能です。
- ・ブレークポイントの登録は最大10ポイントまで可能です。
  - ・アクセスブレークは、登録されたブレークポイントの内の1ポイントしか有効になりません。
  - ・ソフトウェアブレークは、登録されたブレークポイントが全て有効になります。

#### <アクセスブレーク使用時の注意>

アクセスブレークは、UBCを使用する為、以下の制限があります。

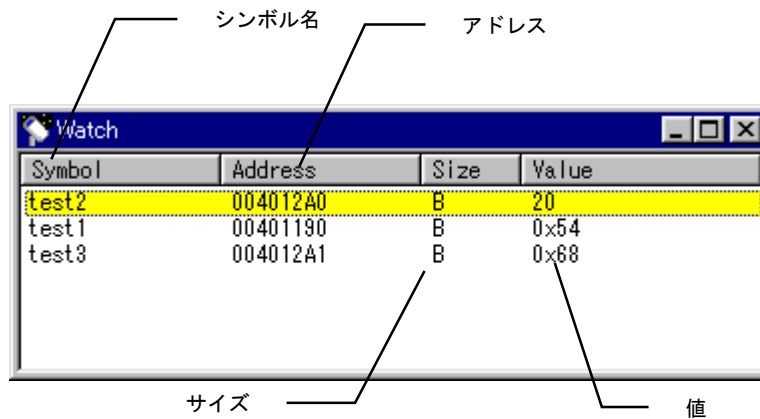
- ・BT, BF命令（非遅延分岐命令）の次の命令フェッチにブレークを指定しても停止しません。
- ・データアクセスをブレーク条件に設定した場合、ブレークするデータアクセスが発生した付近で停止する為停止位置は特定できません。
- ・命令フェッチをブレーク条件に設定した場合、指定位置の手前で停止する場合があります。これはSHがパイプライン方式を採用している為です。
- ・SH1のアセンブラレベルでのデバッグ時において、現在停止している命令の2命令後にアクセスブレークを設定して実行をかけた場合、タイマーなどの割り込み要因が保持されているとアクセスブレークの割り込みと同時に発生します。  
ここで実際はアクセスブレークを設定した所で停止しなくてはならないのですが、SH1のUBCの制約上割り込み処理を実行してからアクセスブレークを実行する為、割り込み処理の先頭で停止してしまいます。
- ・SH7065をデバッグ時において、アクセスブレークと使用した場合は、ブレークポイントの次の行でプログラムが停止します。  
これは、UBCの仕様のためです。SH7065以外のCPUの場合ブレークポイントとして指定したアドレスにある命令の実行前に停止しますが、SH7065は、命令の実行後に停止してしまうためです。

※ UBCの詳細は各CPUのハードウェアマニュアルを参照して下さい。

## 7. 17 ウォッチウィンドウ

### 7. 17. 1 ウォッチウィンドウの機能

ウォッチウィンドウは、指定されたウォッチ変数の内容を常に最新の値で表示します。ステップ実行時やブレーク時に、その都度、変数値を確認したい場合などには非常に便利な機能です。ウォッチ変数の値を変更することも可能です。



#### <表示>

項目	表示内容
Symbol	シンボル名を表示します。
Address	シンボルのアドレス値を表示します。
Size	シンボルのサイズを表示します。(B: 1バイト、W: 2バイト、L: 4バイト)
Value	シンボルの値を表示します。(16進と10進の切り替え可能)

#### <ローカルメニュー操作>

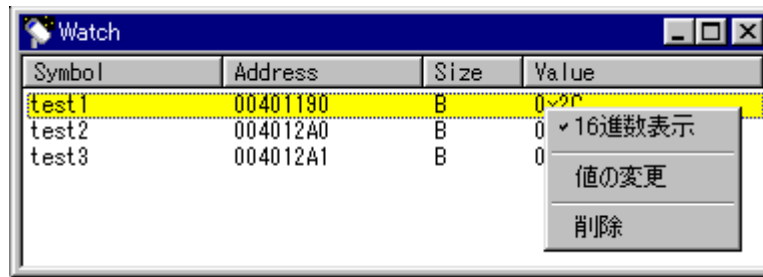
メニュー項目	ショートカット	機能
シンボル (S)	—	シンボル情報ダイアログを表示する。
ウォッチに追加 (A)	—	ウォッチウィンドウに登録を追加する。
ウォッチから削除 (D)	—	ウォッチウィンドウから登録を削除する。

#### <マウス操作>

ポインタ位置	機能
表示行	既に表示されている行を右クリックするとプルダウンメニューが表示され表示形式、値の変更、削除ができます。
空白行	空白号をダブルクリックすると、設定ダイアログが表示されウォッチ変数を追加できます。

#### <キーボード操作>

操作	機能
削除	設定されている行を選択し、DELキーを押下すると登録が削除されます。



プルダウンメニュー（右クリックで表示）

## &lt;プルダウンメニュー&gt;

項目	表示内容
16進表示	チェック時に16進数、未チェック時に10進数で値を表示します。
値の変更	値変更ダイアログを表示します。
削除	ウォッチ変数の登録を削除します。

## 7. 17. 2 ウォッチ変数の値変更

登録されたウォッチ変数の内容を変更することができます。

プルダウンメニューから値の変更を選択した時に表示される変更ダイアログで変更します。



変数値変更ダイアログ

## &lt;変更ダイアログ&gt;

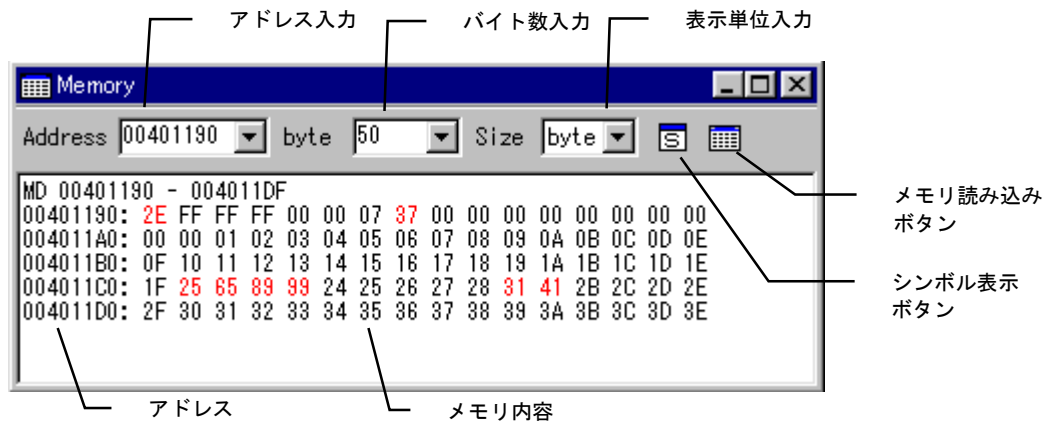
項目	表示内容
Symbol	シンボル名を表示します。
Address	シンボルのアドレスを表示します。
Size	サイズを表示します。（B：1バイト、W：2バイト：L：4バイト）
Value	値を入力します。 16進数で入力する場合は0xから入力し、それ以外は10進数とみなされます。

## 7. 18 メモリウィンドウ

### 7. 18. 1 メモリウィンドウの機能

メモリウィンドウは、指定された範囲のメモリ内容を表示します。

表示はステップ実行時やブレーク時に、自動的に最新の情報を表示します。メモリの内容を直接編集することもできます。



#### <表示>

項目	表示内容
アドレス	表示している行の先頭アドレスを表示します。(16進数)
メモリ内容	現在のメモリの内容を表示します。(16進数) 値が変化した箇所は赤文字で表示します。

#### <入力項目>

項目	表示内容
Address	表示するメモリ先頭アドレスを入力します。
byte	表示するメモリのサイズを入力します。 任意のサイズを指定できます。16進数指定です。
Size	表示する単位を選択します(バイト、ワード、ロング)

#### <ローカルメニュー操作>

メニュー項目	ショートカット	機能
ツールバー (T)	—	ツールバーの表示/非表示を行う。 チェックマークが付いている場合にはツールバー表示が有効である。
表示クリア (C)	—	メモリ内容の表示をクリアします。

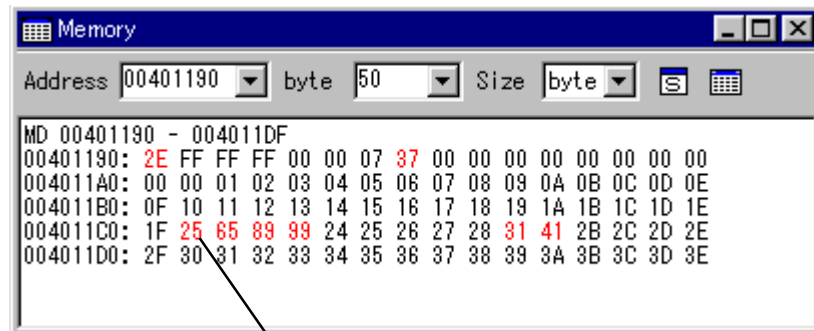
#### <マウス操作>

ポインタ位置	機能
メモリ読み込みボタン	メモリの内容を読み込みます。リターンキーを押しても表示されます。
シンボル表示ボタン	シンボル情報ダイアログを表示します。 詳細は前述の「シンボル入力」を参照してください。



## 7. 18. 2 メモリイメージの編集

メモリウィンドウでは、メモリイメージを直接書き換えて編集できます。



変更したい箇所をクリックし、カーソルを位置付けて数値を入力します。

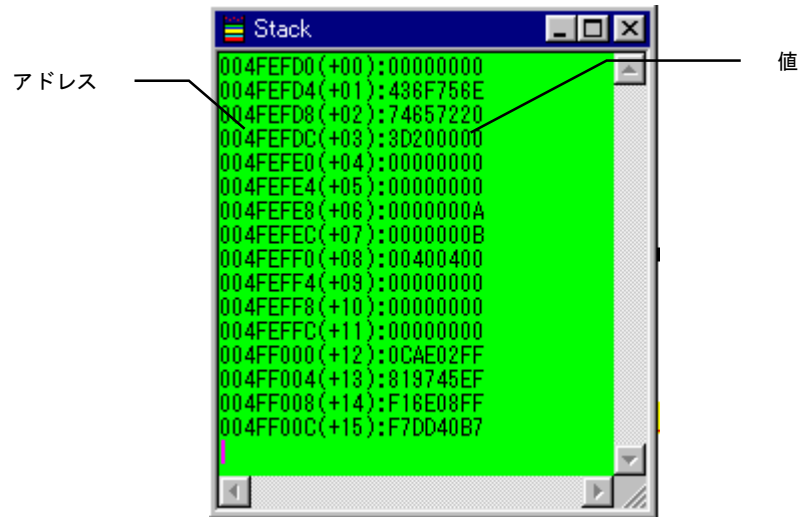
### <操作手順>

- ① 編集したいメモリのアドレス等を入力し、メモリイメージを表示させます。
- ② 編集したい箇所をクリックすると、カーソルが表示されますので数値を入力します。  
入力値は16進数で入力します。
- ③ 実メモリの内容は、入力した時点で変更されていきます。  
変更された箇所は表示が赤色に変わります。

## 7. 19 スタックウィンドウ

### 7. 19. 1 スタックウィンドウの機能

スタックウィンドウは、スタックの内容を常に最新の情報で表示します。



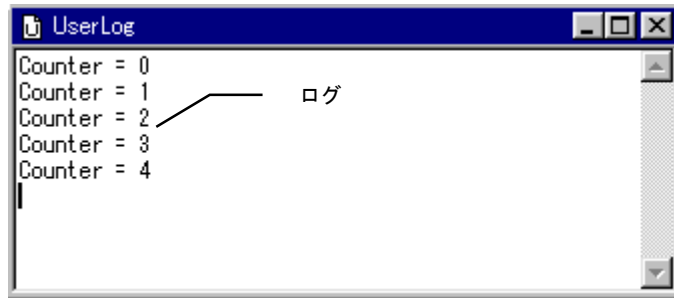
#### <表示>

項目	表示内容
アドレス	スタックのアドレスを表示します。 + 0 が現在のスタックポインタで、相対アドレスで + 1 5 ( 6 4 バイト ) まで表示します。
値	スタックの内容を表示します。 値は常に最新の内容を表示します。

## 7. 20 ユーザログウィンドウ

### 7. 20. 1 ユーザログウィンドウの機能

ユーザログウィンドウは、ユーザプログラムから出力された文字列を表示するウィンドウです。ユーザプログラム実行中の状態を確認する場合などに非常に便利な機能です。ログ出力には、VisualMonitor で用意されている専用関数を使用する必要があります。



#### <表示>

項目	表示内容
ログ	ユーザプログラムから出力された文字列を表示します。

### 7. 20. 2 ユーザログ出力関数

ユーザログ出力機能を使用するためには、VisualMonitor で用意された専用関数を使用する必要があります。

#### <ユーザログ出力関数>

関数名	機能
void UserLogInit(unsigned long addr)	ユーザログ出力用バッファを初期化します。
<引数> ターゲットモニタのワーク領域のアドレスを指定します。(VMedit で指定したアドレス) 例: UserLogInit(0x4ff000);	
<戻り値> なし	
<使用方法> ユーザログを使用する為の初期設定を行ないます UserLogPut 関数を使用する前に、1回だけ設定します。	

関数名	機能
void UserLogPut(char * s);	ユーザログを出力します。
<引数> 出力する文字列を指定します。(最大80文字) 例: UserLogPut("UserLog Out");	
<戻り値> なし	
<使用方法> ユーザログへの出力を行ないます。	

### 7. 20. 3 コーディング例

ライブラリ ユーザプログラムにリンクしてください。

SH1用	SH1UL. LIB	(日立C用)
	SH1ULD. LIB	(日立CVer6用)
	SH1UL. A	(gcc用)
SH2用	SH2UL. LIB	(日立C用)
	SH2ULD. LIB	(日立CVer6用)
	SH2UL. A	(gcc用)
SH3用	SH3UL. LIB	(日立C ビッグエンディアン用)
	SH3ULD. LIB	(日立CVer6 ビッグエンディアン用)
	SH3UL. A	(gcc ビッグエンディアン用)
	SH3LUL. LIB	(日立C リトルエンディアン用)
	SH3LULD. LIB	(日立CVer6 リトルエンディアン用)
	SH3LUL. A	(gcc リトルエンディアン用)

ヘッダファイル ユーザプログラムでインクルードしてください。

```
userlog.h
```

#### コーディング例

```
#include "userlog.h" /*ヘッダファイルの定義*/

void main()
{
    : (省略)
    :
    UserLogInit(0x004ff000); /*ユーザログ初期化*/
    for (;;)
    {
        : (省略)
        :
        UserLogPut("UserLog Out"); /*ユーザログ出力*/
        :
        : (省略)
    }
}
```

#### 注意

ユーザログ出力関数 (UserLogPut ()) は文字列を全て出力するまで終了しませんので、その点に留意して使用してください。

また、自動ブートモードでユーザプログラムが起動した場合にも文字列は出力されません。

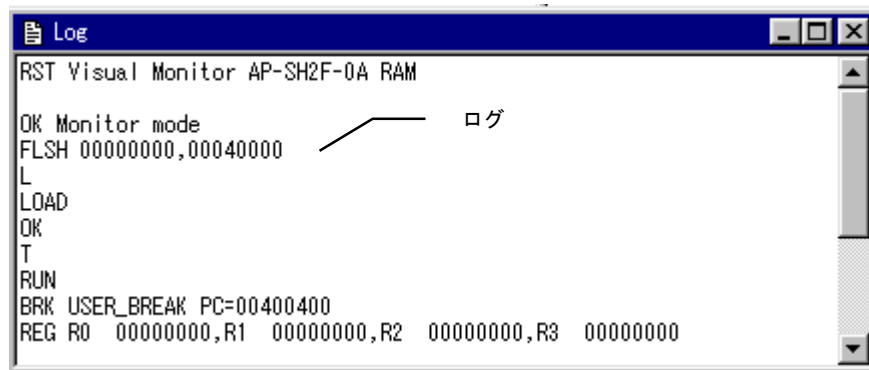
## 7. 21 ログウィンドウ

### 7. 21. 1 ログウィンドウの機能

ログウィンドウは、VisualMonitor の通信状況を表示ウィンドウです。

VisualMonitor の動作に問題がある場合などに、これらのログによって問題を把握します。

通常のデバッグ時には必要ありません。



<表示>

項目	表示内容
ログ	ターゲットモニタとの通信記録を表示します。

## 7. 22 REALi モニタウィンドウ

### 7. 22. 1 REALi モニタウィンドウの機能

REALi モニタウィンドウはREALi の実行状態を表示するウィンドウです。  
ユーザプログラムでREALi を使用していない場合には、この機能は無効になります。



#### <表示>

項目	表示内容
システム表示	REALi のシステム定義項目を表示します。
タスク表示	各タスクの状態及び詳細情報を表示します。
レディキュー表示	レディキューにつながっているタスクを表示します。

#### <マウス操作>

項目	機能
表示切替タグ	各情報表示画面を切り替えます。

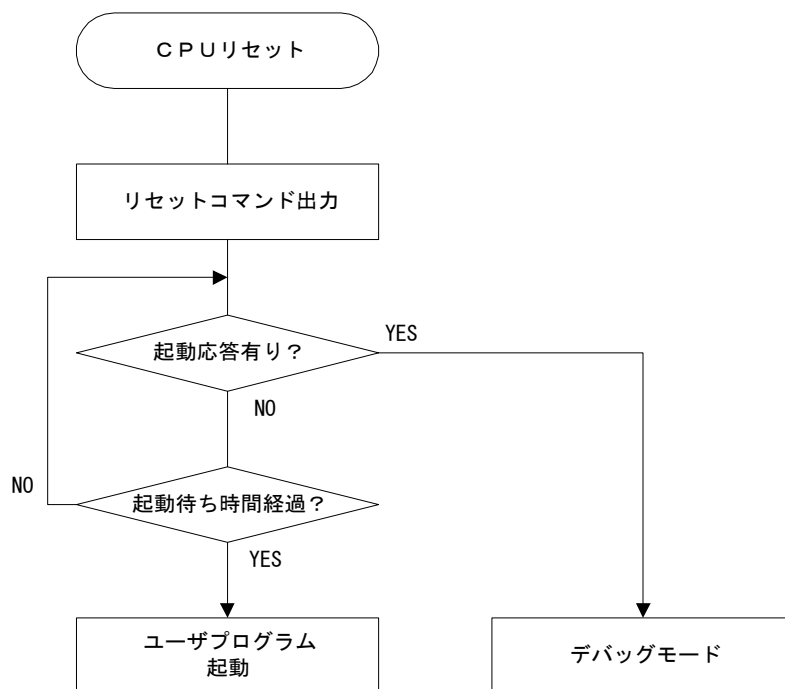
注意) REALi のシステム初期化がされていない状態では、REALi の情報項目は不定値となります。  
REALi の情報項目の詳細はREALi のマニュアルを参照して下さい。  
またREALi はデバッグ付きオプションでコンパイルされている必要があります。  
REALi は、SH7055F、SH7065F、SH3には対応していません (2002年10月時点)

## 8. その他の機能

### 8. 1 自動ブート機能

EPROM、フラッシュROMやバックアップされたRAM上にプログラムがある場合には、デバッグで使用したユーザプログラムをそのまま機器内に組み込んで自動的に起動させる事ができます。

<起動フロー>



<説明>

ターゲットに搭載されたモニタプログラムは、リセット時にリセットコマンドをシリアルインターフェースより出力します。

このリセットコマンドに対して、PC側コントロールソフトは起動応答を返しますが、PCと接続されていない、もしくはコントロールソフトが立ち上がっていない場合には、起動応答がないため起動待ち時間のタイムアウト後にユーザプログラムをジャンプします。

したがって、この機能を利用して機器内にそのまま組み込むことが可能です。

なお、モニタプログラムはユーザプログラム実行中にはまったく関与しないためプログラムの速度低下等はありません。

また、ユーザログ出力関数が使用されている場合でも文字列は出力されません。(無効になります)

注意) 機器内に組み込んだモニタプログラムおよび周辺機器、ソフトウェア等につきましては弊社では一切保証しませんので十分テストを行ってください。

## 第9章. ハードウェアの制限と設計例

### 9. 1 ターゲットシステムの構成

VisualMonitor は特別なハードウェアを必要としないという反面、全ての動作はターゲットシステム上のハードウェアに依存するため、ターゲットシステムを設計する際には、幾つかの注意事項があります。以降に設計例を示しますので参考にしてください。

#### 9. 1. 1 CPUクロック

VisualMonitor は内蔵S C Iを使用して、ホストコンピュータと9600bps～115.2Kbpsで通信します。S C Iの原クロックはC P Uクロックを使用するため、できるだけ通信速度の倍数に近いクロックでC P Uを動作させる必要があります。（詳細は前述の「ターゲットモニタの作成」を参照してください。

#### 9. 1. 2 シリアルインターフェース

VisualMonitor はC P Uの内蔵S C Iを使用してリモート動作します。したがって、内蔵S C Iを1チャンネル、デバッグ用に確保する必要があります。尚、自動ブートで動作する場合には、ブート後に全てのハードウェア資源は開放されるのでユーザプログラムで 사용할ことが可能です。

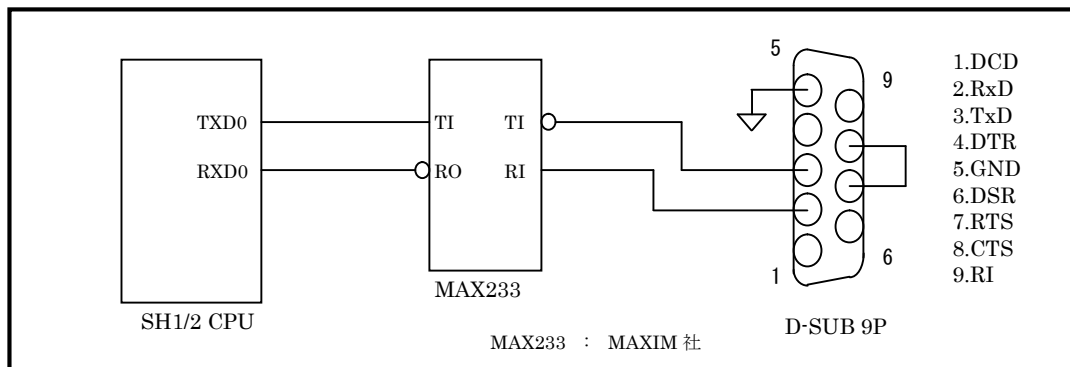


図9. 1. 2 S C I Oを使用した設計例

#### 9. 1. 3 ウォッチドッグタイマ

VisualMonitor は、ユーザプログラムが停止している状態では全ての割り込みを禁止にします。したがって、タイマ割り込み等を使用してウォッチドッグタイマを構成した場合は正しく動作しない場合がありますので、デバッグ時には回路を無効にしてください。



### 9. 1. 4 フラッシュROM

VisualMonitor はフラッシュROMへの書き込みに対応しています。

Ver. 2で標準対応しているデバイス及び接続構成は以下になります。なお、アドレスについてはモニタのカスタマイズにより変更可能です。

設計条件： フラッシュメモリ MBM29F800T/B 互換品

ビット幅：16ビット

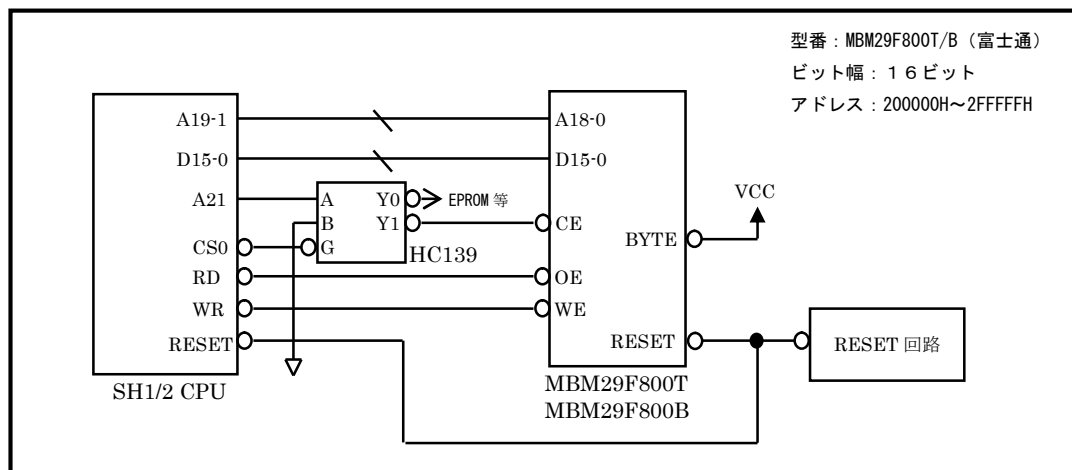


図9. 1. 4 ①フラッシュROMの接続例

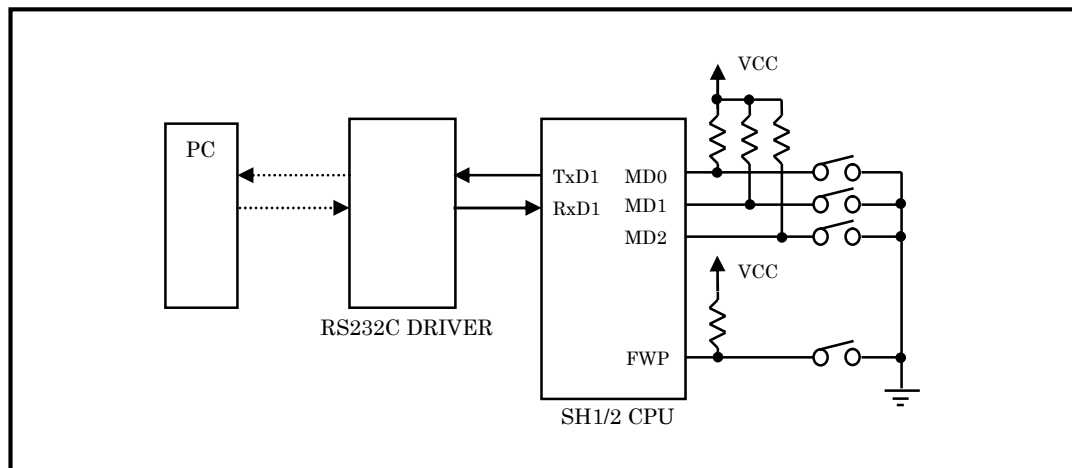


図9. 1. 4 ②F-ZTAT版CPUの設計例

F-ZTAT版CPUをデバッグする場合には、どのチャンネルを使用してもデバッグは可能ですが、内蔵フラッシュのブート書きこみはCPUの制限によりSCI1からしかできません。

したがって、SCI1をデバッグ用に使用することをお勧めします。

また、内蔵フラッシュROMにユーザプログラムをダウンロードする場合には、内蔵ROM有効モードとし、FWPもしくはFWE端子を書きこみ許可にしてデバッグします。(ユーザプログラムモード)

## 第10章. ROM化の方法

### 10.1 ユーザプログラムのROM化

デバッグ後のプログラムをROM化する方法について説明します。

フラッシュROMや内蔵フラッシュROMにダウンロードしてデバッグした場合には、自動ブート機能によってユーザプログラムが起動しますので特にROM化の必要はありません。（「自動ブート機能」を参照）ただし、モニタ部を削除してコードサイズを小さくしたい場合などには有効です。また、本製品で提供するROM化プログラムは、ROM化の手法一例であり、ユーザがブートプログラムや割り込みベクタ等を記述することによってもROM化は可能です。

**注意** お客様の個別アプリケーションのROM化の方法についての質問等につきましてはサポート外とさせていただきます。

#### 10.1.1 ROM化用ファイル

本製品には、デバッグ後のユーザプログラムをROM化するためのファイル群を添付しています。

ファイルは、CD-ROM内の以下のフォルダに収められています。添付のサンプルプログラムと対応したフォルダのファイル

を使用してください。

gcc用ROM化ファイル                      ¥ROM化¥gcc¥cpu名¥  
日立C用ROM化ファイル                   ¥ROM化¥hitachi¥cpu名¥

cpu名 (フォルダ名)	対象CPU	CPUシリーズ
ap_sh_0a	SH7032, 7034	SH 1
ap_sh2_0a	SH7040, 7041, 7042, 7043	SH 2
ap_sh2f_0a	SH7045	
ap_sh2f_1a	SH7050, 7051	
ap_sh2f_2a	SH7044	
ap_sh2f_3a	SH7065	
ap_sh2f_4a	SH7046	
ap_sh2f_5a	SH7047	
ap_sh2f_6a_32	SH7145 (CS1=32BIT BUS)	
ap_sh2f_6a_16	SH7145 (CS1=16BIT BUS)	
ap_sh2f_7a_16	SH7144 (CS1=16BIT BUS)	
ap_sh2f_7a_8	SH7144 (CS1=8BIT BUS)	
sf_7044f	SH7044	
sf_7144f	SH7144	
ap_sh3_0a	SH7709	
ap_sh3_1a	SH7709A	
ap_sh3_2a	SH7709S	
ap_sh3d_0a	SH7729	
ap_sh3d_1a	SH7729R	
sh7604	SH7604	SH 2
sh7052	SH7052	
sh7053	SH7053	
sh7054	SH7054	
sh7055	SH7055	
sh7708	SH7708S/R	SH 3
sh7718	SH7718R	

## フォルダ内のソースファイル

ファイル名	処理内容
cpu.h	ダミーヘッダファイル
rom.c	ブートプログラム
rom_asm.s(src)	割り込み処理
rom_int.c	割り込みハンドラ
set.h	各種設定ファイル

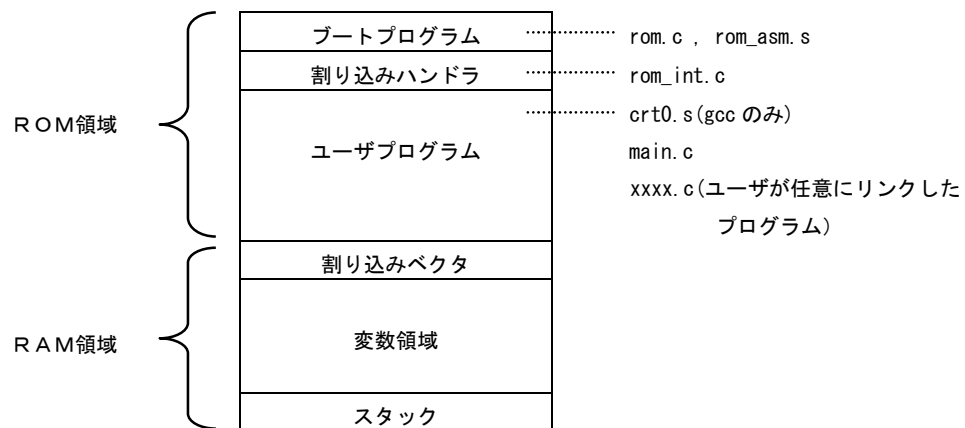
## フォルダ内のコンパイル用サンプルファイル

XXXXXXXXX.bat	生成用バッチファイル (GCC)
XXXXXXXXX.x	生成用リンクスクリプトファイル (GCC)
makefile	make定義バッチファイル (GCC)
makeall.bat	生成用バッチファイル (日立SHC Ver6 以前)
makeall16.bat	生成用バッチファイル (日立SHC Ver6 以降)

これらのファイル群を利用すれば、デバッグしたユーザプログラムを変更することなくROM化が可能です。ROM化用ファイルは全てソースコードで提供しておりますので参考にしてください。

## 10. 1. 2 ROM化プログラムの構造

本製品に添付されているROM化プログラムは、基本的にデバッグしたユーザプログラムを変更することなくROM化することを前提としています。添付のサンプルプログラムを使用した場合を例に、構造を以下に示します。



## デバッグ用プログラムとの違い

- ① ブートプログラム モニタプログラムの代わりとなるブートプログラムを配置します。サイズは1Kバイト以下になります。(ブートプログラム+割り込みハンドラ)
- ② ユーザプログラム ROM領域に配置します。
- ③ スタック ユーザプログラム用のスタックのみとなります。
- ④ ユーザログ ユーザログは出力されません。

### 10. 1. 3 ROM化の手順

SH7045F（弊社製ボードAP-SH2F-0A）のサンプルプログラムを例にROM化手順を説明します。

- ① ユーザプログラムのフォルダをコピーし、前述のROM化ファイルと使用したターゲットモニタの作業フォルダから Set.h をコピーします。



アルファボードシリーズを使用していて、作業フォルダを作成していなかった場合には、VMedit を使用して以下のフォルダから設定ファイルを読み込んで、適当な作業用フォルダに出力して Set.h を作成してください。詳細は「3. 7 アルファボードシリーズでの使用について」をご覧ください。

**フォルダ名：** ¥インストールフォルダ名¥medit¥ap\_board¥ダウンロード領域名¥ボード名¥

- ② 各ソースファイルをターゲットに合わせてカスタマイズします。  
変更が必要なファイルは次のファイルです。

SH1, 2系	.....	rom.c
SH3系	.....	rom.c、rom_asm.s(src)

また、ユーザプログラムを作成するために使用していたコンパイル用ファイルにも変更を加えます。こちらも変更方法は次ページ以降を参照してください。またサンプルの変更済みコンパイル用ファイルがROM化フォルダ内に収録されておりますので参考にしてください。

（日立CでHIMを使用している場合には、サンプルの変更内容を参考に適宜設定してください）

- ③ make や makeall.bat を実行してターゲットのROM化プログラムを生成します。

## &lt;各ファイルの変更点&gt;

rom\_asm.s (gcc) SH3系のみ

```

/*****
/* <カスタマイズ>
/* スタックポインタの初期値を設定してください
/*****
Stack:  .long    0x1000f000          /* スタックポインタの初期値    */
Boot:   .long    _rom_boot
.org    0x00000100

```

スタックポインタ設定  
基本的にはRAMの最終アドレス

rom\_asm.src (日立C、日立CVer6) SH3系のみ

```

/*****
/* <カスタマイズ>
/* スタックポインタの初期値を設定してください
/*****
Stack:  .DATA.L  H' 1000f000          /* スタックポインタの初期値    */
Boot:   .DATA.L  _rom_boot

        .SECTION P1, CODE, ALIGN=4

```

スタックポインタ設定  
基本的にはRAMの最終アドレス

rom.c (gcc、日立C、日立CVer6)

```

void rom_boot(void)
{
/*****
/* <カスタマイズ>
/*   ターゲットモータ作成時に初期設定を以下にコピーし
/*****
        WordWrite(0xFFFFF60, 0x3010);
        WordWrite(0xFFFFF62, 0x35f0);
        . (省略)

```

初期化処理を記述します。  
基本的にはターゲットモータ作成時に記述した boot.c の初期化処理と同じ処理となります。

/\* BSC BCR1\*/  
/\* エリア 3 を EDO DRAM アクセス \*/  
/\* エリア 4 を SRAM アクセス \*/

/\* BSC BCR2\*/  
/\* CS5 のバス幅 8 ビット \*/

## GCC用バッチファイル(.bat)の変更(sh2f\_0a.bat)

rem 環境に合わせて各実行ファイルやライブラリのパスの設定を変更してください

```
echo OFF
echo -v -m2 -nostartfiles          > sh2f_0a.lnk
echo -Wl,-t,-Map sh2f_0a.map      >> sh2f_0a.lnk
echo -o sh2f_0a.out               >> sh2f_0a.lnk
echo crt0.o                       >> sh2f_0a.lnk
echo main.o                       >> sh2f_0a.lnk
echo gmachine.o                   >> sh2f_0a.lnk
```

ROM化用プログラムを追加する。

```
rem ROM化用プログラムを付け加える (ここから↓)
echo rom_asm.o                   >> sh2f_0a.lnk
echo rom.o                        >> sh2f_0a.lnk
echo rom_int.o                    >> sh2f_0a.lnk
```

※1

rem ROM化用プログラムを付け加える (ここまで↑)

※1

```
echo -Tsh2f_0a.x                  >> sh2f_0a.lnk
```

User log 用ライブラリを外す。

```
rem UserLog 用のライブラリはリンクから外してください
rem echo c:\¥VMonitor¥lib¥sh2ul.a    >> sh2f_0a.lnk
```

※2

echo ON

```
sh-hms-gcc -c -v -g -m2 main.c -o main.o
sh-hms-gcc -c -v -g -m2 crt0.s -o crt0.o
sh-hms-gcc -c -v -g -m2 gmachine.c -o gmachine.o
```

ROM化用プログラムを追加する。

```
rem ROM化用プログラムを付け加える (ここから↓)
```

※3

```
sh-hms-gcc -c -v -g -m2 rom_asm.s -o rom_asm.o
sh-hms-gcc -c -v -g -m2 rom.c -o rom.o
sh-hms-gcc -c -v -g -m2 rom_int.c -o rom_int.o
```

```
rem ROM化用プログラムを付け加える (ここまで↑)
```

※3

```
sh-hms-gcc @sh2f_0a.lnk
```

```
rem S-FORMAT へのコンバート
```

```
sh-hms-objcopy -O srec sh2f_0a.out sh2f_0a.mot
```

※4

Sフォーマット変換処理を追加する。  
exeGCC の場合は COFF2S を使用。

## GCC用スクリプトファイル (.x) の変更 (sh2f\_0a.x)

```

OUTPUT_FORMAT("coff-sh")
OUTPUT_ARCH(sh)
MEMORY
{
/* メモリのマッピングを変更する(ここから↓)           ※5
rom    : o = 0x00400400, l = 0x0007fc00
ram    : o = 0x00480000, l = 0x0007f000
stack  : o = 0x004feff0, l = 0x00000010
*/
}
/*ROM のアドレス、サイズ*/
rom    : o = 0x00000000, l = 0x00040000
/*RAM のアドレス、サイズ
ベクタテーブルの領域は外して下さい*/
ram    : o = 0x00400400, l = 0x0007fbf0
/*スタックのアドレス、サイズは固定*/
stack  : o = 0x004ffff0, l = 0x00000010
/* メモリのマッピングを変更する(ここまで↑)           ※5*/
}
SECTIONS
{
.text :
{
    _stext = . ;
}
/* 割り込み処理用のセクションを追加する               ※6*/
*(.vector)

*(.text)
*(.strings)
_etext = . ;
} > rom
.data : AT ( ADDR (.text) + SIZEOF (.text) )
{
    _sdata = . ;
    *(.data)
    _edata = . ;
} > ram
.bss :
{
    _bss_start = . ;
    *(.bss)
    *(COMMON)
    _end = . ;
} > ram
.stack :
{
    _stack_top = . ;
    *(.stack)
} > stack
}

```

コメントアウト

メモリマッピングを変更する。  
rom → ROMのアドレス  
ram → 開始アドレスはRAMの先頭  
アドレス+ベクタ領域サイズ  
stack → 基本的にはRAMの最終ア  
ドレス 注1)

割り込み処理用のセクションを追加。

注1) 上記のスクリプト例の場合、リンカの制限によりサイズ指定が最小 0x00000010 となるため、このような設定となっています。

## makefileの変更(AP-SH2F-0A用 makefile)

```

INCPATH =
LIBPATH = C:/VMonitor/lib/
AS      = sh-hms-gcc
CC      = sh-hms-gcc
LD      = sh-hms-gcc
OBJCOPY = sh-hms-objcopy

TARGET = sh2f_0a

CFLAGS = -c -v -g -m2
LDFLAGS = -v -m2 -nostartfiles -Wl,-t,-Map ${TARGET}.map -Tsh2f_0a.x

# ROM 化用プログラムを付け加える (ここから↓)
#SRCS  = crt0.s main.c gmachine.c
#OBJJS  = crt0.o main.o gmachine.o
SRCS   = crt0.s main.c gmachine.c rom_asm.s rom.c rom_int.c
OBJJS  = crt0.o main.o gmachine.o rom_asm.o rom.o rom_int.o
# ROM 化用プログラムを付け加える (ここまで↑)

# UserLog 用のライブラリはリンクから外してください
#LIBS  = ${LIBPATH}SH2UL.a
LIBS   =

${TARGET}.out:${OBJJS} ${TARGET}.x makefile
    ${LD} ${LDFLAGS} -o ${TARGET}.out ${OBJJS} ${LIBS}
# S-FORMAT へのコンバート
    ${OBJCOPY} -O srec ${TARGET}.out ${TARGET}.mot

crt0.o:crt0.s
    ${CC} ${CFLAGS} crt0.s -o crt0.o

main.o:main.c
    ${CC} ${CFLAGS} main.c -o main.o

gmachine.o:gmachine.c
    ${CC} ${CFLAGS} gmachine.c -o gmachine.o

# ROM 化用プログラムを付け加える (ここから↓)
rom_asm.o:rom_asm.s
    ${CC} ${CFLAGS} rom_asm.s -o rom_asm.o

rom.o:rom.c
    ${CC} ${CFLAGS} rom.c -o rom.o

rom_int.o:rom_int.c
    ${CC} ${CFLAGS} rom_int.c -o rom_int.o
# ROM 化用プログラムを付け加える (ここまで↑)

```

※1 ROM化用プログラムを追加する。

※2 User log 用ライブラリを外す。

※4 Sフォーマット変換処理を追加する。  
exeGCC の場合は COFF2S を使用。

※3 ROM化用プログラムを追加する。

**注意** makefile は、GNU ツール (レッドハット社 GNU Pro) に含まれる make.exe にて動作確認されています。それ以外の make.exe を使用する場合には、適宜修正してください。



## 日立C用のバッチファイルの変更 (AP-SH2F-0A 用 makeall.bat)

rem 環境に合わせて各実行ファイルやライブラリのパスの設定を行なってください

```

echo off
echo debug                                > sh2f_0a.sub

rem ROM 化用プログラムを付け加える (ここから↓)
echo input rom_asm.obj                    >> sh2f_0a.sub
echo input rom_obj                        >> sh2f_0a.sub
echo input rom_int.obj                    >> sh2f_0a.sub
rem ROM 化用プログラムを付け加える (ここまで↑)

echo input section.obj                    >> sh2f_0a.sub
echo input main.obj                       >> sh2f_0a.sub
echo input gmachine.obj                   >> sh2f_0a.sub
echo library %shc_lib%¥..¥lib¥shcnpic.lib >> sh2f_0a.sub

rem UserLog 用のライブラリはリンクから外してください
rem echo library C:¥VMonitor¥lib¥SH2UL.lib >>> sh2f_0a.sub

rem スタート関数の指定を変更する
rem echo entry _boot                       >> sh2f_0a.sub
echo entry _rom_boot                       >> sh2f_0a.sub

echo output sh2f_0a.abs                    >> sh2f_0a.sub
echo print sh2f_0a.map                     >> sh2f_0a.sub
echo rom (D,R)                             >> sh2f_0a.sub

rem メモリのマッピングを変更する(ここから↓)
rem echo start P,C,D (00400400)            >> sh2f_0a.sub
rem echo start R,B (00480000)              >> sh2f_0a.sub
echo START P0,P,C,D(00000000)              >> sh2f_0a.sub
echo START R,B(00400400)                   >> sh2f_0a.sub
rem メモリのマッピングを変更する(ここまで↑)

echo exit                                  >> sh2f_0a.sub
echo on

shc /op=0 /cpu=SH2 /debug /show=ob,noso main.c > err
shc /op=0 /cpu=SH2 /debug /show=ob,noso gmachine.c >>err
asmsh section.src -debug -NOLIST -CR -SE >>err

rem ROM 化用プログラムを付け加える (ここから↓)
shc /op=0 /cpu=SH2 /debug /show=ob,noso rom.c
shc /op=0 /cpu=SH2 /debug /show=ob,noso rom_int.c
asmsh rom_asm.src -debug -NOLIST -CR -SE
rem ROM 化用プログラムを付け加える (ここまで↑)

lnk /subcommand=sh2f_0a.sub

rem S-FORMAT へのコンバート
cnvs sh2f_0a

```

※1 ROM化プログラムを追加する。

※1

※2 User log 用ライブラリを外す。

※3 スタート関数を指定する。

※4 コメントアウト

※4 メモリマッピングを変更する。  
P,C,D → ROMのアドレス  
R,B → 開始アドレスはRAMの  
先頭アドレス+ベクタ領域サイズ

※5 ROM化プログラムを追加する。

※5

※6 Sフォーマット変換処理を追加する。

## 日立CVer6用のバッチファイルの変更 (AP-SH2F-0A用 makeall6.bat)

rem 環境に合わせて各実行ファイルやライブラリのパスの設定を行なってください

echo off

ROM 化ファイルを出力するため  
debug オプションを外す。

rem S-FORMAT ファイルをリンカが直接出力するため不要 ※6-1  
rem echo debug > sh2f\_0a.sub

rem ROM 化用プログラムを付け加える (ここから↓) ※1  
echo input=rom\_asm.obj > sh2f\_0a.sub }  
echo input=rom.obj >> sh2f\_0a.sub }  
echo input=rom\_int.obj >> sh2f\_0a.sub }  
rem ROM 化用プログラムを付け加える (ここまで↑) ※1

ROM化プログラムを追加する。

echo input=main.obj >> sh2f\_0a.sub  
echo input=section.obj >> sh2f\_0a.sub  
echo input=gmachine.obj >> sh2f\_0a.sub  
echo library=%shc\_lib%¥.¥lib¥shcnpic.lib >> sh2f\_0a.sub

rem UserLog 用のライブラリはリンクから外してください ※2  
rem echo library C:¥VMonitor¥lib¥SH2ULD.lib >>> sh2f\_0a.sub

User log 用ライブラリを外す。

rem スタート関数の指定を変更する ※3  
rem echo entry=\_boot >> sh2f\_0a.sub }  
echo entry=\_rom\_boot >> sh2f\_0a.sub }

スタート関数を指定する。

rem S-FORMAT ファイルをリンカが直接出力するため変更 ※6-2  
rem echo output=sh2f\_0a.abs >> sh2f\_0a.sub }  
echo output=sh2f\_0a.mot >> sh2f\_0a.sub }

出力ファイル名を変更する。

echo list=sh2f\_0a.map >> sh2f\_0a.sub  
echo show=symbol >> sh2f\_0a.sub  
echo rom=D=R >> sh2f\_0a.sub

rem メモリのマッピングを変更する(ここから↓) ※4  
rem echo start=P, C, D/00400400 >> sh2f\_0a.sub }  
rem echo start=R, B/00480000 >> sh2f\_0a.sub }  
echo start=P0, P, C, D/00000000 >> sh2f\_0a.sub }  
echo start=R, B/00400400 >> sh2f\_0a.sub }  
rem メモリのマッピングを変更する(ここまで↑) ※4

ROM領域のマッピングを変更  
する。

rem 出力形式を S-FORMAT に指定する ※6  
echo FORM=stype >> sh2f\_0a.sub

出力フォーマットをSフォー  
マットに指定する。

echo exit >> sh2f\_0a.sub  
echo on

shc /op=0 /cpu=SH2 /debug /show=ob,noso main.c > err  
shc /op=0 /cpu=SH2 /debug /show=ob,noso gmachine.c >> err  
asmsh section.src -DEBUG -NOLIST -CR -SE >>err

rem ROM 化用プログラムを付け加える (ここから↓) ※5  
shc /op=0 /cpu=SH2 /debug /show=ob,noso rom.c }  
shc /op=0 /cpu=SH2 /debug /show=ob,noso rom\_int.c }  
asmsh rom\_asm.src -DEBUG -NOLIST -CR -SE }  
rem ROM 化用プログラムを付け加える (ここまで↑) ※5

ROM化プログラムを追加する。

optlnk -subcommand=sh2f\_0a.sub

## 第11章. gccコンパイラ

VisualMonitorには、ターゲットモニタ生成用にgccが添付されています。

お客様がユーザプログラムを作成することもできますが、本製品の目的とは異なりますので特に詳細なご説明はいたしません。使用方法等については、弊社ホームページで参考文献やリンクをご紹介しますので、そちらをご覧ください。また、トランジスタ技術2001年6月号、もしくは弊社KIT製品にレッドハット社のGNUProが添付されておりますので、それらの使用をお勧めいたします。

なお、弊社ではgccに関する質問やサポートは一切受け付けておりませんのでご了承ください。

gccはフリーソフトですが、使用するにあたってはGPL規約及びLGPL規約に従う必要があります。本CD-ROM内に収録されたGNU使用許諾書をご覧ください。

## 第12章. Q&A

Q&Aは、弊社ホームページで随時更新しておりますので、トラブルシューティングにお役立てください。

Q 1. モニタプログラムを機器に組み込んだままで使用可能ですか。?

A 1. 可能です。モニタプログラムはブートした時点で、PC側ソフトから所定のコマンドを受け取るによりデバッグモードとなります。設定時間以内に所定のコマンドを受けとらなかった場合にはRUNモードとなり、制御をユーザプログラムに移します。これを利用すればデバッグ時の状態のままで、製品に組み込むことが可能です。  
また、組み込んだままでメンテナンス等が可能となります。

Q 2. 製品組み込みで使用した場合のロイヤリティは必要ですか。?

A 2. ロイヤリティは必要ありません。ただし、組み込みで使用した場合の製品に対する保証は一切しませんので、十分テストを行ってご利用ください

Q 3. モニタプログラムのソースコードは公開しているのでしょうか。?

A 3. ヘッドファイルおよびスタートアップ等、ユーザカスタマイズが必要な部分のみ公開しています。

Q 4. シリアルインターフェースの条件は。?

A 4. 対応しているシリアルインターフェースは内蔵SCIのみです。  
外付けのSIOなどは使用できません。なお、内蔵SCIはどのチャンネルでも使用可能です。  
また、クロックの選択は内部クロックのみとなります。

Q 5. デバッグモードからユーザモードへダイナミックに移行することは可能ですか。? またその逆は可能ですか。?

A 5. 対応していません。リセット時にデバッグモードかユーザモードか決定してします。

Q 6. 内蔵SCIはデバッグ用として必ず1CH確保しておかなければならないのでしょうか。?

A 6. デバッグ時には必要です。ただし、ユーザプログラムに自動ブートした場合はモニタは全ての資源を開放しますので、使用可能となります。

Q 7. gccのバージョン及び種類は限定されますか?

A 7. gccはさまざまなプラットフォーム上で動作し、またいろいろな方法で移植されています。  
ただし、基本的に生成されるデバッグ情報ファイル(COFFフォーマット)は共通なので問題はありません。  
当社では、最も標準的なGNUCとして、レッドハット社のGNUProにて動作を確認しています。  
なお、ELF形式には現在対応しておりませんので、注意してください。(Q18参照)

Q 8. VisualMonitorをF-ZTAT書き込みツールとして利用可能ですか。?

A 8. VisualMonitorの対応している書き込みモードはユーザプログラムモードです。  
したがって、VisualMonitorを一番最初に書き込むにはブートモード書き込みに対応したツールが必要となります。純粋に書き込みツールが必要な場合には弊社製書き込みツール「FLASH WRITER PRO 2」をご利用ください。

Q 9. EPROMやフラッシュROM上のユーザプログラムをデバッグ可能ですか。?

A 9. 可能です。したがって、製品内に組み込んだ後でもシリアル接続してデバッグモードにすれば、そのままデバッグが可能となります。

Q 1 0. VisualMonitorでタイマを使用しているようですが、ユーザプログラムでは使用できないのでしょうか。?

A 1 0. タイマは、スタートアップ時のデバッグモードへの遷移時間を設定できるように使用しています。  
ただし、使用するのはスタートアップ時（リセット時）だけなので、ユーザプログラムでは自由に使用できます。

Q 1 1. リアルタイムOSなどでユーザプログラムがスタックを切り替えているが、問題なく動作しますか?

A 1 1. リアルタイムOSはスタックポインタを直接操作するものや割り込みベクタを操作するものがありますので、それらについては正しくデバッグできない場合があります。

Q 1 2. gccでサンプルプログラムをコンパイルしたが、コンパイルエラーが大量に出ってしまった。

A 1 2. コンパイルで問題が発生するのは、ほとんどの場合コンパイラオプションの指定ミスによるものです。  
特に-m1, -m2, -m3 指定誤り及び-ansi 指定などは注意してください。  
また、コンパイラオプションは環境変数でも指定されるので、AUTOEXEC.BATなどで間違ったオプションを指定していないかなどもチェックしてください。

Q 1 3. gccでサンプルプログラムをコンパイルしたが、リンクエラーが出てしまう。

A 1 3. ほとんどが環境変数の指定ミスによるものです。  
ご使用の環境に合わせて正しく変更されているかご確認ください。

Q 1 4. REALiのモニタウィンドウがアクティブになりません。

A 1 4. REALiのモニタウィンドウを有効にするためには、REALiのデバッグ情報付きライブラリをリンクする必要があります。REALiにはソースコードが添付されておりますので、デバッグオプション付きで再生成してライブラリを作成してください。わからない場合にはE-mailにて送付しますので、ご連絡ください。

Q 1 5. スタックのアドレスエラーが発生するのですが?

A 1 5. VisualMonitorでは、カスタマイズ項目としてスタックのアドレス設定がありますが、SHマイコンの場合スタックは4バイト長なので、4の倍数でアドレスを設定する必要があります。

Q 1 6. ダウンロード中にエラーが頻繁に発生する。

A 1 6. 主な原因としては以下の2点が考えられます。  
① ユーザプログラム領域とターゲットモニタのワーク領域が重複している。  
② SCIのボーレートの誤差が大きい。  
両方とも「ターゲットモニタの作成」を参照して、ただしく設定をおこなってください。

Q 1 7. 内蔵フラッシュROMのダウンロード中にエラーが発生する。

A 1 7. 内蔵フラッシュROMにユーザプログラムをダウンロードする場合には、CPUのFWP端子を解除してユーザプログラムモードにしておく必要があります。

Q 1 8. exeGCCのVer.2でデバッグができないのですが?

A 1 8. exeGCC(京都マイクロコンピュータ製)のVer.2はデバッグ情報がELF形式で、VisualMonitorでは使用できません。  
以前のVer.1を使用するか、COFF形式のGCCをご使用ください。

Q 1 9. StarterKitからバージョンアップしたが、うまく動作しない

A 1 9. SH2, SH3/StarterKitのターゲットモニタと正規版のVisualMonitorは互換性がありません。  
したがって、正規版にアップグレードされた場合にはターゲットモニタを入れ替えていただく必要があります。  
ターゲットモニタについては「3章 ターゲットモニタの作成」を参照してください。

Q 2 0. SH704x シリーズで DTC ベクタテーブルを使いたいのですが、使用できますか？

A 2 0. 制限はございますが、使用することは可能です。

DTC ベクタテーブルは、アドレスが固定されています。(0x00000400~)

このアドレスは、通常はターゲットモニタのコード領域となっているため DTC ベクタテーブルは使用できません。

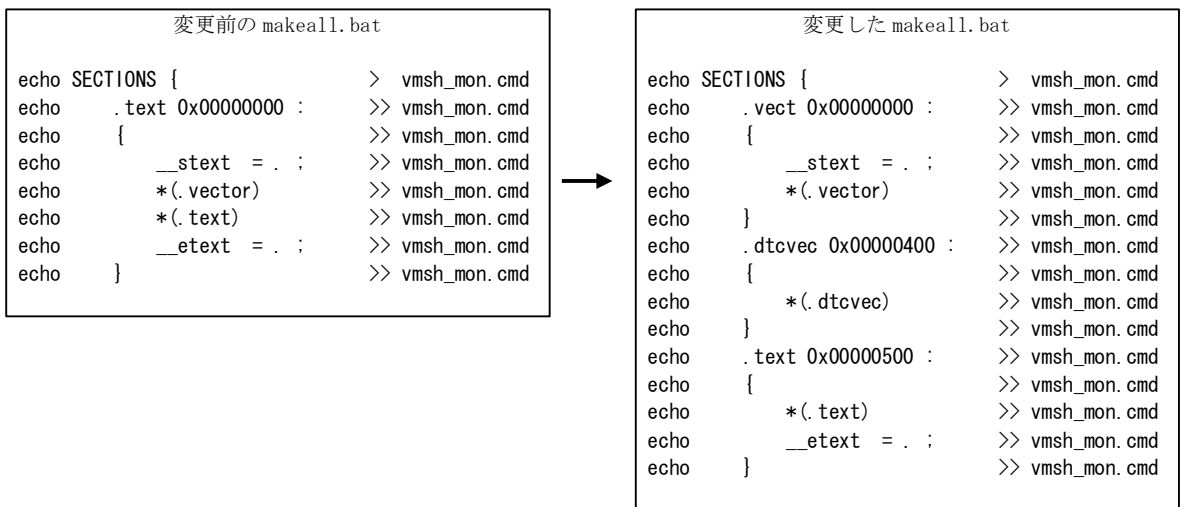
しかし、ターゲットモニタのコード領域をずらし、あらかじめ DTC ベクタテーブルをリンクすることで使うことができます。

このターゲットモニタの作成手順は以下のようになります。

なお、DTC ベクタテーブルを変更する場合には、ターゲットモニタを変更する必要があります。

1. VMedit を使用してターゲットモニタ作成用ファイルを出力し boot.c の変更を行ないます。

2. makeall.bat を以下のように変更します。(リンカスクリプトの変更)



3. src¥vector.s を以下のように変更します。(DTC ベクタテーブルの追加)

```

変更した vector.s

/*===== Vector Table =====*/
:
: (省略)
:
.long STACK

/*===== DTC Vector Table =====*/
.section .dctvec
/*===== DTC Vector Table =====*/
.word 0x0000
:
: (省略)
:
.word 0x0000

.end

```

追加箇所

ユーザプログラムに合わせてベクタ (下位 16bit) を設定して下さい。

4. makeall.bat を実行してターゲットモニタを作成します。

## 第13章. サポート

### 13.1 ユーザ登録

添付のユーザ登録カードに必要な事項を記入し、弊社までFAXでお送りください。

ご登録のお客様にはバージョンアップ等の情報をお送りいたします。

なお、ご登録のないお客様からのサポートはお断りさせていただく場合もありますので、必ずご登録ください。

### 13.2 バージョンアップ

マイナーバージョンアップは、不定期で年2回～4回程度おこなっております。

バージョンアップデータは無償で弊社ホームページからダウンロードできますので、定期的にご覧いただくようお願いいたします。

メジャーバージョンアップの予定は未定となっておりますが、実施時にはご登録ユーザ様にご連絡いたします。

こちらは、有償となりますがご登録ユーザ様は、バージョンアップ費用のみでご購入いただけます。

### 13.3 質問の受け付け

弊社では随時質問を受け付けております。FAXもしくはE-mailにてご連絡ください。

正確な対応をおこなうために、電話による受け付けはお断りしております。

また、ご質問の際にはコンパイル環境、MAKEファイル、プログラムソース、コンパイラ生成物一式を添付していただくと迅速な対応が可能です。

ご質問の回答については、基本的には2、3日中にご返答しますが、内容によってはお時間をいただく場合もございますのでご了承ください。

なお、以下のサポートや質問については半導体メーカーにお問い合わせいただくか、お客様の技術によって解決される問題であり、弊社では受け付けをお断りさせていただきますのでご了承願います。

- ・マイコンの使用法、プログラミング、コンパイラの使用法など、技術指導的なサポート
- ・ユーザターゲットに合わせたサンプルプログラムおよびモニタプログラムの作成提供サービス
- ・ユーザプログラムのROM化サービス
- ・その他、本製品の仕様範囲外の質問

#### 株式会社 アルファプロジェクト

〒431-3114 静岡県浜松市東区積志町 834

FAX (053)401-0035 技術部 担当者宛

ホームページアドレス : <http://www.apnet.co.jp/>

E-mail アドレス : [query@apnet.co.jp](mailto:query@apnet.co.jp)

## 第14章. 付録

### 14.1 アルファボードシリーズの仕様

弊社製CPUボードの諸元値を記載します。添付のサンプルプログラムと合わせてご覧ください。

#### <AP-SH-0A>

構成	仕様
CPU	SH7032
周波数	20MHz
ROM	256K~2Mbyte 16bitバス CS0に接続 アドレス: 08000000~0803FFFFH(081FFFFFH)
SRAM	256Kbyte 16bitバス CS2に接続 アドレス: 0A000000~0A03FFFFH
内蔵RAM	8Kbyte アドレス: FFFFE000~FFFFFFFH

#### <AP-SH2-0A>

構成	仕様
CPU	SH7043
周波数	24.576MHz (6.144MHz水晶振動子)
ROM	256K~2Mbyte 16bitバス CS0をデコード アドレス: 10000000~0003FFFFH(01FFFFFH)
フラッシュROM	1Mbyte 29F800T 16bitバス CS0をデコード アドレス: 00200000~002FFFFFH
SRAM	1Mbyte 16bitバス CS1に接続 アドレス: 00400000~004FFFFFH
内蔵RAM	4Kbyte (キャッシュ兼用) アドレス: FFFFF000~FFFFFFFH

#### <AP-SH2F-0A>

構成	仕様
CPU	SH7045F
周波数	28.636MHz (7.15909MHzクロック入力)
内蔵フラッシュROM	256Kbyte アドレス: 00000000~0003FFFFH
ROM	512Kbyte 16bitバス CS0に接続 アドレス: 内蔵ROM有効モード 00200000~0027FFFFH 内蔵ROM無効モード 00000000~0007FFFFH
SRAM	1Mbyte 32bitバス CS1に接続 アドレス: 00400000~004FFFFFH
内蔵RAM	4Kbyte (キャッシュ兼用) アドレス: FFFFF000~FFFFFFFH



## &lt;AP-SH2F-1A&gt;

構成	仕様
CPU	SH7051F
周波数	19.6608MHz (4.9152MHz水晶振動子)
内蔵フラッシュROM	256Kbyte アドレス: 00000000~0003FFFFH
ROM	512Kbyte 16bitバス CS0に接続 アドレス: 内蔵ROM有効モード 00200000~0027FFFFH 内蔵ROM無効モード 00000000~0007FFFFH
SRAM	256Kbyte 16bitバス CS1に接続 アドレス: 00400000~0043FFFFH
内蔵RAM	10Kbyte アドレス: FFFFD800~FFFFFFFH

## &lt;AP-SH2F-2A&gt;

構成	仕様
CPU	SH7044F
周波数	24.576MHz (6.144MHz水晶振動子)
内蔵フラッシュROM	256Kbyte アドレス: 00000000~0003FFFFH
SRAM	256Kbyte 16bitバス CS1に接続 アドレス: 00400000~0043FFFFH
内蔵RAM	4Kbyte (キャッシュ兼用) アドレス: FFFFF000~FFFFFFFH

## &lt;AP-SH2F-3A&gt;

構成	仕様
CPU	SH7065F
周波数	60MHz (15MHz水晶振動子)
内蔵フラッシュROM	256Kbyte アドレス: 00000000~0003FFFFH
SRAM	1Mbyte 32bitバス CS2に接続 アドレス: 08000000~080FFFFFH
X/Yメモリ	4Kbyte + 4Kbyte アドレス: FFFF8000~FFF8FFFFH (XRAM) アドレス: FFFFA000~FFFAFFFFH (YRAM)

## &lt;AP-SH2F-4A&gt;

構成	仕様
CPU	SH7046F
周波数	49.152MHz (12.288MHz水晶振動子)
内蔵フラッシュROM	256Kbyte アドレス: 00000000~0003FFFFH
内蔵RAM	12Kbyte アドレス: FFFFD000~FFFFFFFH

## &lt;AP-SH2F-5A&gt;

構成	仕様
CPU	SH7047F
周波数	49.152MHz (12.288MHz水晶振動子)
内蔵フラッシュROM	256Kbyte アドレス: 00000000~0003FFFFH
SRAM	256Kbyte X 2BANK 8bitバス CS0に接続 アドレス: 00200000~0023FFFFH
内蔵RAM	12Kbyte アドレス: FFFFD000~FFFFFFFH

## &lt;AP-SH2F-6A&gt;

構成	仕様
CPU	SH7145F
周波数	49.152MHz (12.288MHz水晶振動子)
内蔵フラッシュROM	256Kbyte アドレス: 00000000~0003FFFFH
SRAM ※	2Mbyte 32bitバス CS1に接続 アドレス: 00400000~005FFFFFFH もしくは、 1Mbyte 16bitバス CS1に接続 アドレス: 00400000~004FFFFFFH
内蔵RAM	8Kbyte アドレス: FFFFE000~FFFFFFFH

※CS1のバス幅はジャンパ設定によって選択できます。

## &lt;AP-SH2F-7A&gt;

構成	仕様
CPU	SH7144F
周波数	49.152MHz (12.288MHz水晶振動子)
内蔵フラッシュROM	256Kbyte アドレス: 00000000~0003FFFFH
SRAM ※	1Mbyte 16bitバス CS1に接続 アドレス: 00400000~004FFFFFFH もしくは、 512Kbyte 8bitバス CS1に接続 アドレス: 00400000~0047FFFFH
内蔵RAM	8Kbyte アドレス: FFFFE000~FFFFFFFH

※CS1のバス幅はジャンパ設定によって選択できます。

## &lt;SF-7044F&gt;

構成	仕様
CPU	SH7044F
周波数	24.576MHz (6.144MHz水晶振動子)
内蔵フラッシュROM	256Kbyte アドレス:00000000~0003FFFFH
SRAM	1Mbyte 16bitバス CS1に接続 アドレス:00400000~004FFFFFH
FPAG ※	1Mbyte 8bitバス CS3に接続 アドレス:00C00000~00CFFFFFH
内蔵RAM	4Kbyte (キャッシュ兼用) アドレス:FFFFFF00~FFFFFFFH

※SF-7044F用のサンプルは、ボードに搭載されているFPAGを使用しています。

## &lt;SF-7144F&gt;

構成	仕様
CPU	SH7144F
周波数	49.152MHz (12.288MHz水晶振動子)
内蔵フラッシュROM	256Kbyte アドレス:00000000~0003FFFFH
SRAM	1Mbyte 16bitバス CS1に接続 アドレス:00400000~004FFFFFH
FPAG ※	1Mbyte 8bitバス CS3に接続 アドレス:00C00000~00CFFFFFH
内蔵RAM	8Kbyte アドレス:FFFEE000~FFFFFFFH

※SF-7144F用のサンプルは、ボードに搭載されているFPAGを使用しています。

## &lt;AP-SH3-0A&gt;

構成	仕様
CPU	SH7709
周波数	最大80MHz (10MHz水晶振動子) 注1)
フラッシュROM	1Mbyte 29LV800B 16bitバス CS0かCS5の接続切替可能 アドレス: CS0接続時 00000000~000FFFFFH CS5接続時 14000000~140FFFFFH
ROM	128K~1Mbyte 8bitバス CS0かCS5の接続切替可能 アドレス: CS0接続時 00000000~0001FFFFH (000FFFFFH) CS5接続時 14000000~1401FFFFH (140FFFFFH)
EDO-DRAM	16Mbyte 32bitバス 4K/64ms CBRリフレッシュ CS3に接続 アドレス: 0C000000~0CFFFFFFH
SRAM	128Kbyte 8bitバス CS4に接続 アドレス: 10000000~1001FFFFH
内蔵RAM	4Kbyte (RAMモード時) アドレス: 7F000000~7F000FFFH

注1) サンプルの初期値は以下のようになっています。

クロック動作モード MODE2 FRQCR=H'102 1:B:P = 4:4:1 = 40MHz:40MHz:10MHz キャッシュ OFF

サンプルを参照して適宜変更してください。

## &lt;AP-SH3-1A&gt;

構成	仕様
CPU	SH7709A
周波数	最大128MHz (16MHz水晶振動子) 注1)
フラッシュROM	1Mbyte 29LV800B 16bitバス CS0かCS5の接続切替可能 アドレス: CS0接続時 00000000~000FFFFFH CS5接続時 14000000~140FFFFFH
ROM	128K~1Mbyte 8bitバス CS0かCS5の接続切替可能 アドレス: CS0接続時 00000000~0001FFFFH (000FFFFFH) CS5接続時 14000000~1401FFFFH (140FFFFFH)
SDRAM	16Mbyte 32bitバス 4K/64msリフレッシュ CS3に接続 アドレス: 0C000000~0CFFFFFFH
SRAM	128Kbyte 8bitバス CS4に接続 アドレス: 10000000~1001FFFFH
キャッシュメモリ	命令/データ混在キャッシュ16Kbyte

注1) サンプルの初期値は以下のようになっています。

クロック動作モード MODE2 FRQCR=H'102 1:B:P = 4:4:1 = 64MHz:64MHz:16MHz キャッシュ OFF

サンプルを参照して適宜変更してください。

## &lt;AP-SH3-2A&gt;

構成	仕様
CPU	SH7709S
周波数	最大192MHz (16MHz水晶振動子) 注1)
フラッシュROM	4Mbyte 29LV320B 16bitバス CS0かCS5の接続切替可能 アドレス: CS0接続時 00000000~003FFFFFFH CS5接続時 14000000~143FFFFFFH
ROM	128K~1Mbyte 8bitバス CS0かCS5の接続切替可能 アドレス: CS0接続時 00000000~0001FFFFH (000FFFFFFH) CS5接続時 14000000~1401FFFFH (140FFFFFFH)
SDRAM	32Mbyte 32bitバス 4K/64msリフレッシュ CS3に接続 アドレス: 0C000000~0DFFFFFFH
SRAM	128Kbyte 8bitバス CS4に接続 アドレス: 10000000~1001FFFFH
キャッシュメモリ	命令/データ混在キャッシュ16Kbyte

注1) サンプルの初期値は以下のようになっています。

クロック動作モード MODE2 FRQCR=H'A101 I:B:P = 12:4:2 = 192MHz:64MHz:32MHz キャッシュ OFF  
サンプルを参照して適宜変更してください。

## &lt;AP-SH3D-0A&gt;

構成	仕様
CPU	SH7729
周波数	最大128MHz (16MHz水晶振動子) 注1)
フラッシュROM	1Mbyte 29LV800B 16bitバス CS0かCS5の接続切替可能 アドレス: CS0接続時 00000000~000FFFFFFH CS5接続時 14000000~140FFFFFFH
ROM	128K~1Mbyte 8bitバス CS0かCS5の接続切替可能 アドレス: CS0接続時 00000000~0001FFFFH (000FFFFFFH) CS5接続時 14000000~1401FFFFH (140FFFFFFH)
SDRAM	16Mbyte 32bitバス 4K/64msリフレッシュ CS3に接続 アドレス: 0C000000~0CFFFFFFH
SRAM	128Kbyte 8bitバス CS4に接続 アドレス: 10000000~1001FFFFH
X/Yメモリ	8Kbyte + 8Kbyte アドレス: A5007000~A5008FFFH (XRAM) アドレス: A5017000~A5018FFFH (YRAM)
キャッシュメモリ	命令/データ混在キャッシュ16Kbyte

注1) サンプルの初期値は以下のようになっています。

クロック動作モード MODE2 FRQCR=H'102 I:B:P = 4:4:1 = 64MHz:64MHz:16MHz、キャッシュ OFF  
サンプルを参照して適宜変更してください。

## &lt;AP-SH3D-1A&gt;

構成	仕様
CPU	SH7729R
周波数	最大192MHz (16MHz水晶振動子) 注1)
フラッシュROM	4Mbyte 29LV320B 16bitバス CS0かCS5の接続切替可能 アドレス: CS0接続時 00000000~003FFFFFFH CS5接続時 14000000~143FFFFFFH
ROM	128K~1Mbyte 8bitバス CS0かCS5の接続切替可能 アドレス: CS0接続時 00000000~0001FFFFFFH (00FFFFFFH) CS5接続時 14000000~1401FFFFFFH (140FFFFFFH)
SDRAM	32Mbyte 32bitバス 4K/64msリフレッシュ CS3に接続 アドレス: 0C000000~0DFFFFFFH
SRAM	128Kbyte 8bitバス CS4に接続 アドレス: 10000000~1001FFFFFFH
X/Yメモリ	8Kbyte + 8Kbyte アドレス: A5007000~A5008FFFH (XRAM) アドレス: A5017000~A5018FFFH (YRAM)
キャッシュメモリ	命令/データ混在キャッシュ16Kbyte

注1) サンプルの初期値は以下のようになっています。

クロック動作モード MODE2 FRQCR=H'A101 I:B:P = 12:4:2 = 192MHz:64MHz:32MHz、キャッシュ OFF  
サンプルを参照して適宜変更してください。

\* アルファボードシリーズ以外のサンプルプログラムは以下のターゲットの条件で作成されています。

<SH7604>

構成	仕様
CPU	SH7604
周波数	24.576MHz (6.144MHz水晶振動子)
ROM	128Kbyte 16bitバス ビッグエンディアン アドレス: 00000000~0001FFFFH
フラッシュROM	1Mbyte 29F800T 16bitバス ビッグエンディアン CS2に接続 アドレス: 04000000~040FFFFFH
SRAM	128Kbyte 8bitバス ビッグエンディアン CS1に接続 アドレス: 02000000~0201FFFFH
内蔵RAM	4Kbyte (キャッシュ兼用) アドレス: C0000000~C0000FFFH

<SH7052F>

構成	仕様
CPU	SH7052F
周波数	40MHz (10MHzクロック入力)
内蔵フラッシュROM	256Kbyte アドレス: 00000000~0003FFFFH
SRAM	256Kbyte 16bitバス ビッグエンディアン CS1に接続 アドレス: 00400000~0043FFFFH
内蔵RAM	12Kbyte アドレス: FFFF8000~FFFAFFFH

<SH7053F>

構成	仕様
CPU	SH7053F
周波数	40MHz (10MHzクロック入力)
内蔵フラッシュROM	256Kbyte アドレス: 00000000~0003FFFFH
SRAM	256Kbyte 16bitバス ビッグエンディアン CS1に接続 アドレス: 00400000~0043FFFFH
内蔵RAM	16Kbyte アドレス: FFFF8000~FFFBFFFH

## &lt;SH7054F&gt;

構成	仕様
CPU	SH7054F
周波数	40MHz (10MHzクロック入力)
内蔵フラッシュROM	384Kbyte アドレス 00000000~0005FFFFH
SRAM	256Kbyte 16bitバス ビッグエンディアン CS1に接続 アドレス: 00400000~0043FFFFH
内蔵RAM	16Kbyte アドレス: FFFF8000~FFFFBFFFH

## &lt;SH7055F&gt;

構成	仕様
CPU	SH7055F
周波数	40MHz (10MHz水晶振動子)
内蔵フラッシュROM	512Kbyte アドレス 00000000~0007FFFFH
ROM	512Kbyte 16bitバス ビッグエンディアン CS0に接続 アドレス: 内蔵ROM有効モード 00200000~0027FFFFH 内蔵ROM無効モード 00000000~0007FFFFH
SRAM	512Kbyte 16bitバス ビッグエンディアン CS1に接続 アドレス: 00400000~0047FFFFH
内蔵RAM	32Kbyte アドレス: FFFF6000~FFFFDFFFH

## &lt;SH7708&gt;

構成	仕様
CPU	SH7708R
周波数	100MHz (25MHz水晶発信器)
ROM	128Kbyte 8bitバス ビッグエンディアン アドレス: 00000000~0001FFFFH
SRAM	128Kbyte 8bitバス ビッグエンディアン CS2に接続 アドレス: 08000000~0801FFFFH
SDRAM	16Mbyte 32bitバス ビッグエンディアン CS3に接続 アドレス: 0C000000~0CFFFFFFH
キャッシュメモリ	命令/データ混在キャッシュ 8Kbyte



## &lt;SH7718R&gt;

構成	仕様
CPU	SH7718R
周波数	100MHz (25MHz水晶発信器)
ROM	128Kbyte 8bitバス ビッグエンディアン アドレス: 00000000~0001FFFFH
SRAM	128Kbyte 8bitバス ビッグエンディアン CS2に接続 アドレス: 08000000~0801FFFFH
SDRAM	16Mbyte 32bitバス ビッグエンディアン CS3に接続 アドレス: 0C000000~0CFFFFFFH
キャッシュメモリ	命令/データ混在キャッシュ 8Kbyte

## 第15章. 製品サポートと使用上の注意

### 15.1 製品サポートのご案内

#### 15.1.1 弊社ホームページのご利用について

弊社製品へのよくあるご質問及びご要望については、弊社ホームページ上のFAQに掲載しております。掲載内容につきましては随時更新されておりますので、是非ご利用ください。また、バージョンアップについてもホームページ上より提供しております。

弊社ホームページアドレス <http://www.apnet.co.jp/index.html>

#### 15.1.2 製品サポートの方法

製品サポートについては、FAXもしくはE-MAILでのみ受け付けております。お電話でのお問い合わせは受け付けておりませんのでご了承ください。

##### 製品サポート窓口

- |                |                   |
|----------------|-------------------|
| ■ FAXによるご連絡    | 053-401-0035      |
| ■ E-MAILによるご連絡 | query@apnet.co.jp |

#### 15.1.3 製品サポートの範囲

以下の内容に該当するお問い合わせにつきましては、サポートの対象とはなりませんので、あらかじめご了承ください。

- 本製品を利用したアプリケーションプログラムの作成方法とそれらに関連するご質問
- 本製品のソフトウェア技術に関するご質問
- 本製品を利用して作成された2次生成物の利用者からのご質問
- 一般的なコンピュータに関する事項や他社製品に関するご質問

### 15.2 使用上の注意

- 本製品を改造した場合、一切の保証は適用されません。
- 本製品を仕様範囲を越える条件において使用された場合については、動作は保証しませんのでご了承願います。
- 本製品に組み込まれたプログラム及び添付アプリケーションのリバースエンジニアリング及び本製品以外でのご使用は堅くお断りします。
- 万が一、本製品を使用して事故または損失が発生した場合、弊社では一切その責を負いませんのでご了承ください。

## ご注意

- (1) 本書に記載されている、MPU、コンパイラなどの製品名は各社の登録商標です。
- (2) 本書の内容の一部又は全部を無断で転載することは、一切禁止されています。
- (3) 本書の内容および本資料に記載された製品に関しては、将来予告なしに変更されることがあります。
- (4) 本書の内容については、万全を期して作成いたしましたが、万一ご不審な点、誤りなどお気づきの点がありましたら弊社までご連絡下さい。
- (5) 運用した結果については(4)項にかかわらず責任を負いませんのでご了承下さい。

本製品 (**VisualMonitor**) は日本国内仕様であり、外国の規格等には準拠しておりません。本製品は日本国外で使用された場合、当社では一切責任を負いかねます。また当社は本製品に関し、海外での保守サービス及び技術サポート等はおこなっておりません。

**Alpha Project Co.,LTD**