ＳＨ１、ＳＨ２、ＳＨ３対応リモートデバッガ

VisualMonitor/SH

Ver2.6

User's Manual

１５版 ２０２３．１０． ２

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。

本製品はルネサス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。

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った場合には、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
- ・ソフトウエア使用許諾書
- ・ＦＡＸサポート依頼書
- ・メンテナンスのご案内
- ・FAX ユーザ登録用紙

お問い合わせ先

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

Fax　(053)401-0035　技術部 担当者宛
E-mail　query@apnet.co.jp

目　次

第１章. 概要

本製品は、ルネサスＳＨシリーズ用に開発されたＷｉｎｄｏｗｓ７／８／８．１／１０／１１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	既存のハードウエ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）
●ＳＨＣ・ｇｃｃ両対応	ＳＨＣ（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もしくはルネサス純正の書き込みツールが別途必要です。
注２）ＧＣＣのＥＬＦ形式には対応しておりません。

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	Ｗｉｎｄｏｗｓ７／８／８．１／１０／１１
	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	ルネサス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。

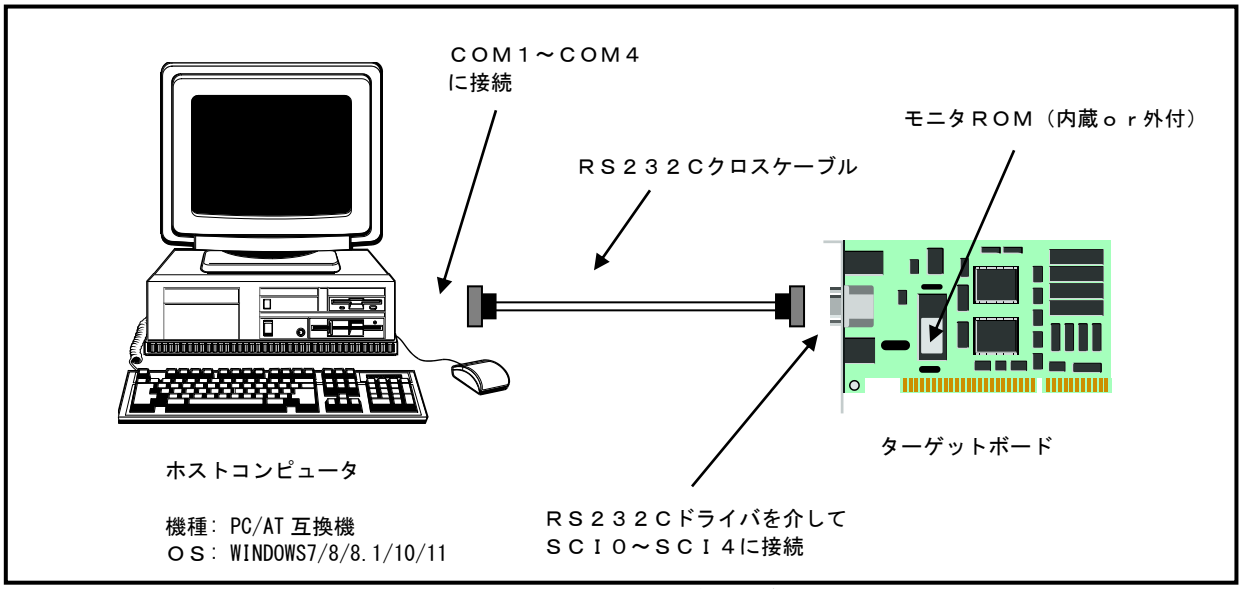


図１．３ VisualMonitor を使用したデバッグ環境の構成例

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がです。	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がです。	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

ＳＨ１

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	○

ＳＨ２

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	AP-SH2F-2A、SF-7044F	SH7044F	○
	SH7045F	AP-SH2F-0A	SH7045F	○
	SH7046F	AP-SH2F-4A	SH7046F	○
	SH7047F	AP-SH2F-5A	SH7047F	○
ＳＨ７０５０シリーズ	SH7050F		SH7050F	○
	SH7051F	AP-SH2F-1A	SH7051F	○
	SH7052F		SH7052F	○
	SH7053F		SH7053F	○
	SH7054F		SH7054F	○
	SH7055F		SH7055F	○
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

ＳＨ３

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。

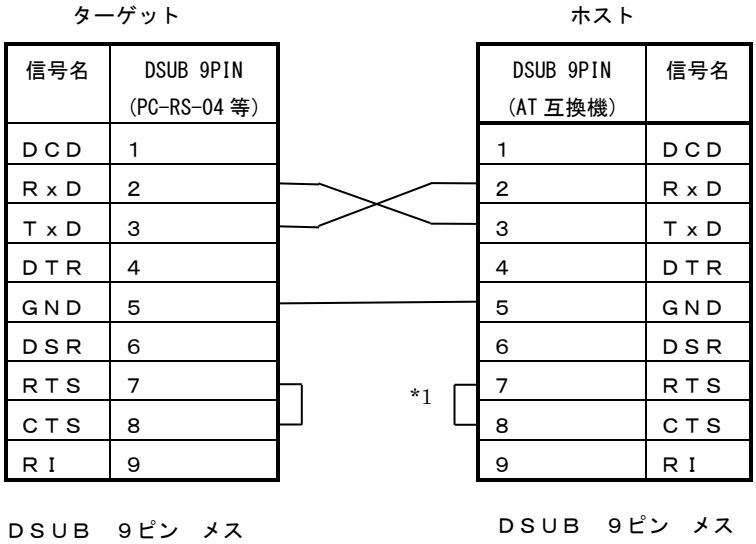
② ターゲット
前項の使用環境を御覧下さい。

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、**「D:￥INSTALL¥SETUP.EXE」**[*1]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。

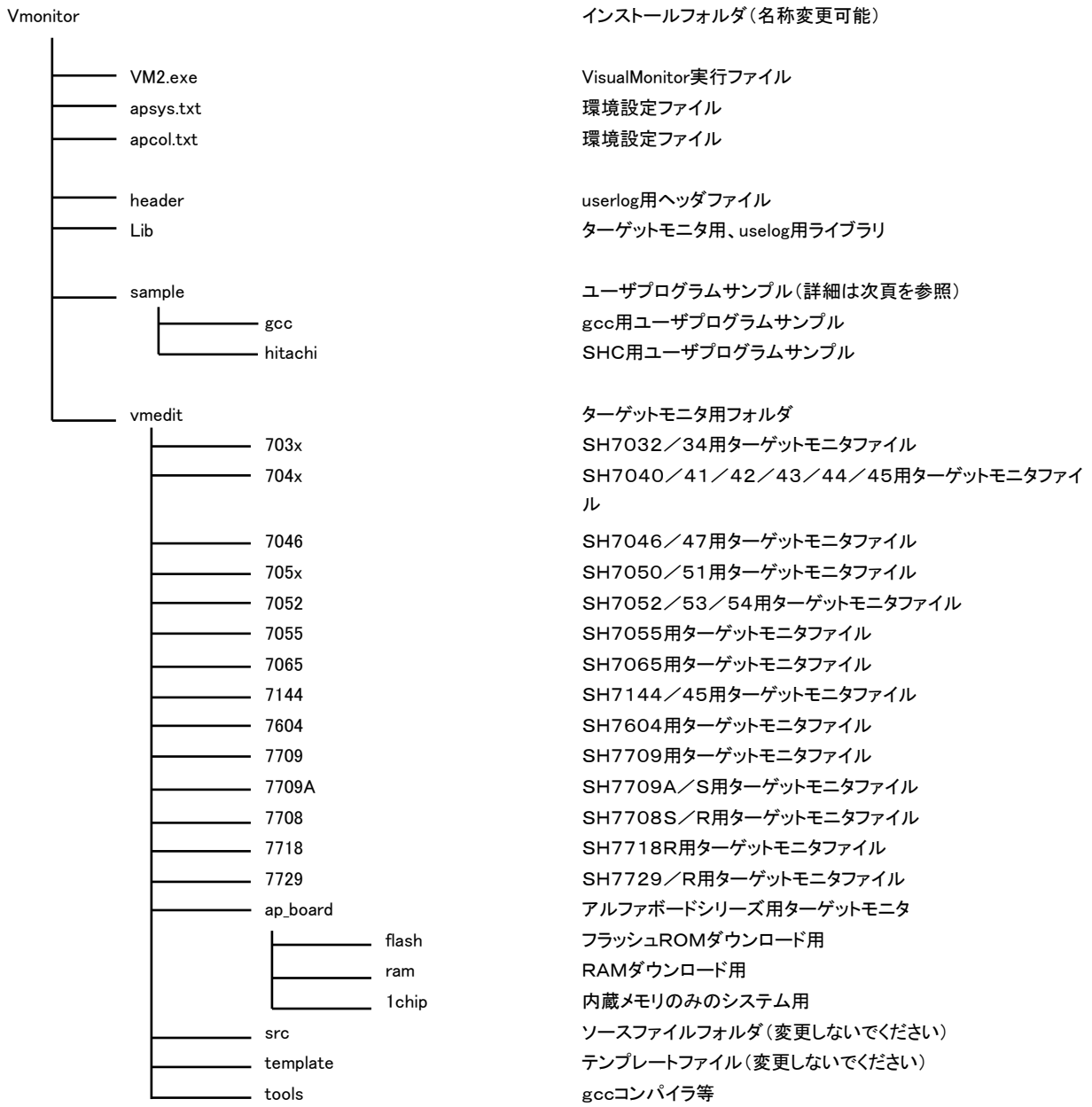
注意

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】



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	ＳHC

<ボード名、またはＣＰＵ名> （ flash、ram、1chip 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	1chip
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、ジャンパ設定によって、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。

<エンディアン>

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

<モニタのタイプ>

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。
1chip	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。

＊ＲＥＡＬｉ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　　￥ＲＥＡＬｉ￥＜ＲＥＡＬｉ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＞

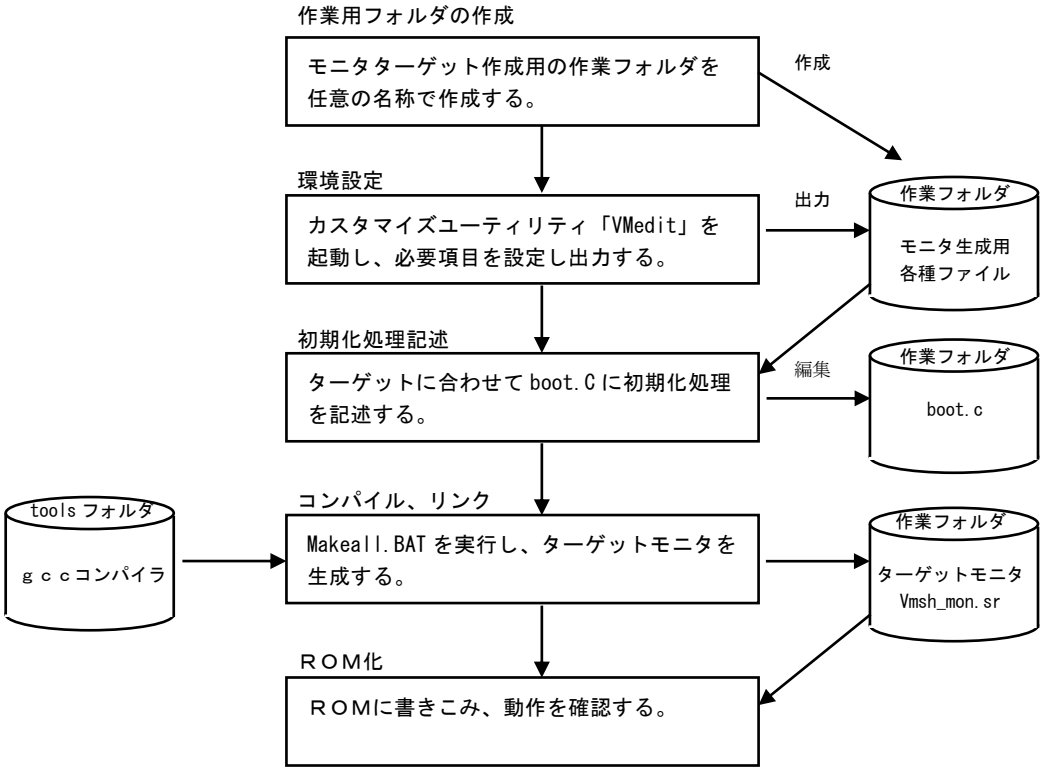
注意）ＲＥＡＬｉはＳＨ２Ｅ、ＳＨ２ＤＳＰ、ＳＨ３、ＳＨ３Ｅ、ＳＨ３ＤＳＰ、ＳＨ４には対応していません。

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ＳＨＣの場合でも同様です。

３．２　ＶＭｅｄｉｔの使用方法

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下さい。（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チャネル	VisualMonitor で使用するＳＣＩのチャネルを設定します。	該当するチャネルを選択
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。	１００～５０００ｍｓｅｃ
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 通信レートの目安

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)
φ：動作周波数（ＭＨｚ）
Ｎ：ボーレートジェネレータのＢＲＲの設定値（0≦N≦255）
ｎ：ボーレートジェネレータ入力クロック（n=0,1,2,3）

ＶＭｅｄｉｔ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ＶＭｅｄｉｔ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      /* ﾎﾞｰﾚｰﾄｼﾞｪﾈﾚｰﾀ入力ｸﾛｯｸ          */
#define         SIO_BRR             22          /* ﾎﾞｰﾚｰﾄｼﾞｪﾈﾚｰﾀ(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。

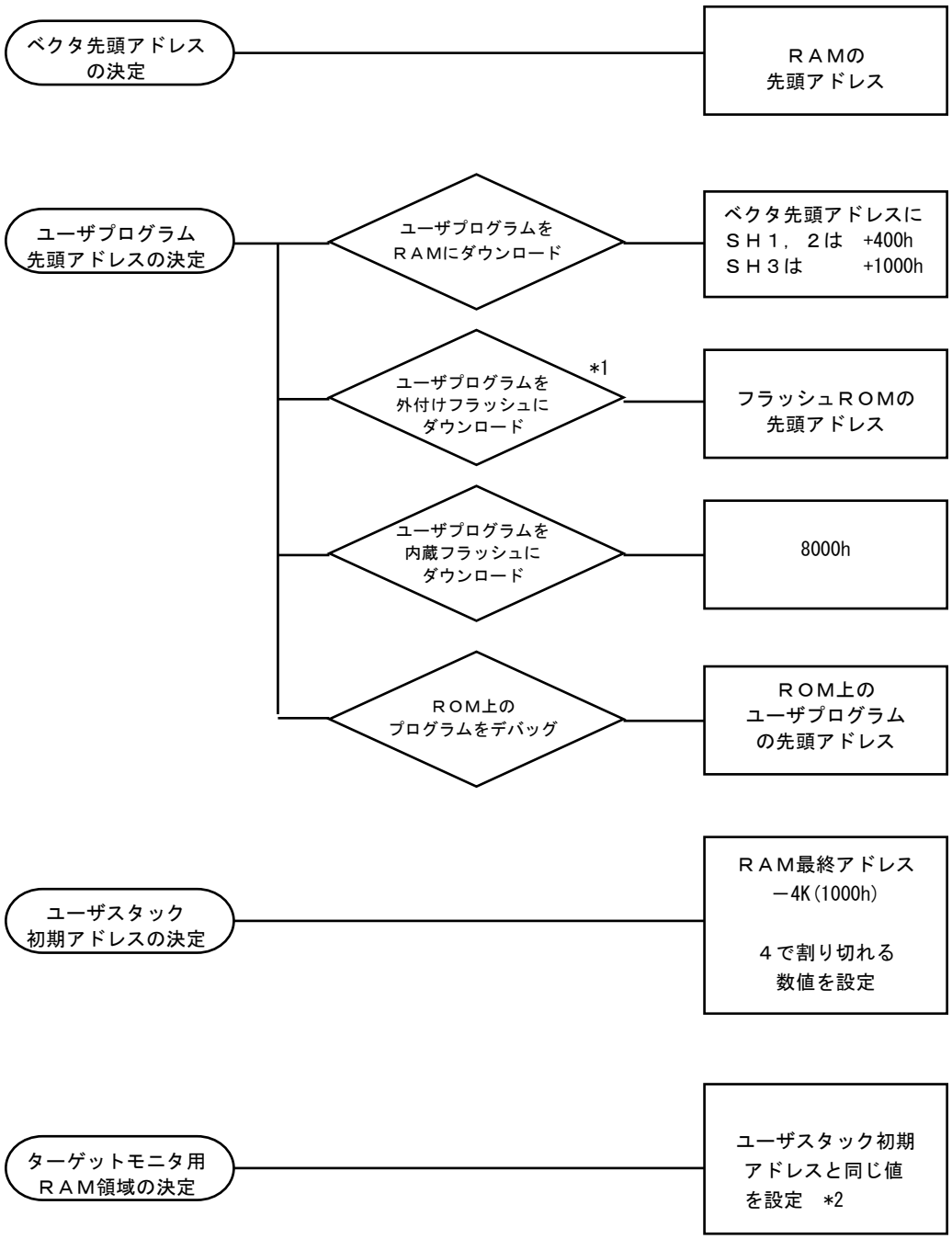
３．２．２　アドレス設定

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ます。	

＜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。

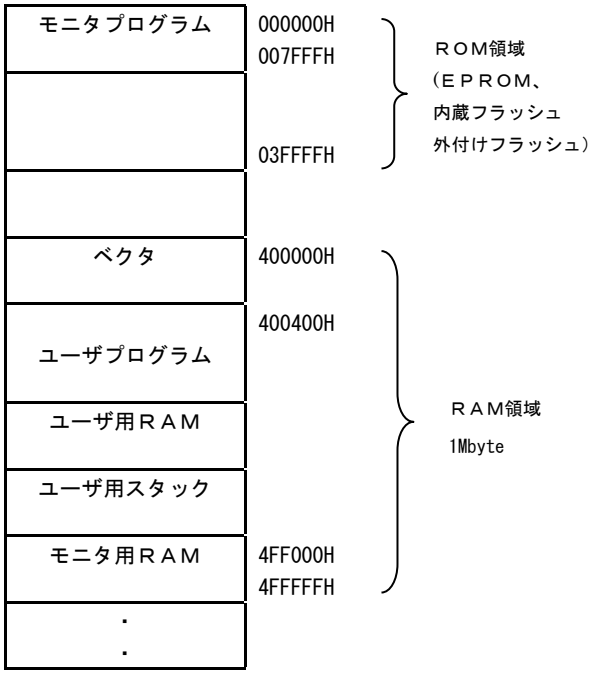


＊１　フラッシュＲＯＭの配置は後述の設計例を参照してください。

＊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



この例は、ＥＰＲＯＭ上にモニタプログラムを配置し、ＲＡＭにユーザプログラムをダウンロードする場合の配置例です。

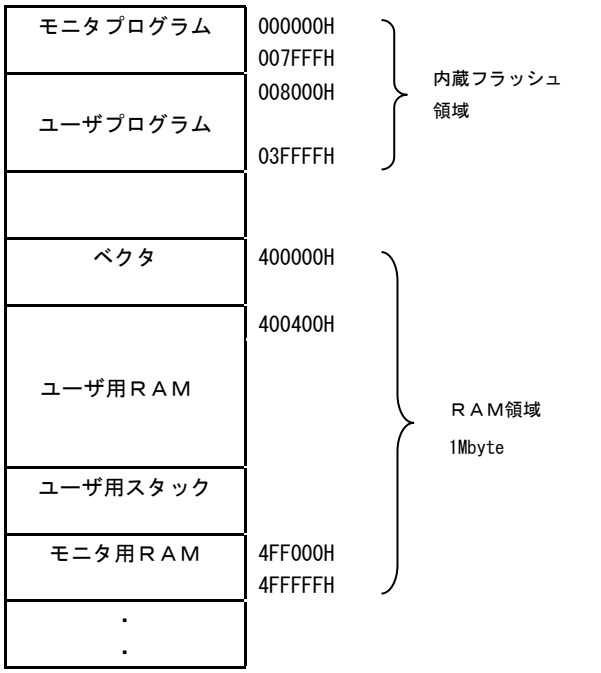
設定値は

ベクタ先頭アドレス	400000h
ユーザプログラム先頭アドレス	400400h
ユーザスタック初期アドレス	4FF000h
ターゲットモニタ用 RAM 領域	4FF000h

となります。

この構成は弊社ＣＰＵボードでは
ＡＰ－ＳＨ－０Ａ（ＳＨ７０３２）
ＡＰ－ＳＨ２－０Ａ（ＳＨ７０４３）
ＡＰ－ＳＨ３－０Ａ（ＳＨ７７０９）
等　が該当します。
（アドレスはそれぞれ異なります）

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



この例は、内蔵フラッシュ上にモニタプログラムを配置し、同じく内蔵フラッシュにユーザプログラムをダウンロードする場合の配置例です。

設定値は

ベクタ先頭アドレス	400000h
ユーザプログラム先頭アドレス	008000h
ユーザスタック初期アドレス	4FF000h
ターゲットモニタ用 RAM 領域	4FF000h

となります。

この構成は弊社ＣＰＵボードでは
ＡＰ－ＳＨ２Ｆ－０Ａ（ＳＨ７０４５Ｆ）
ＡＰ－ＳＨ２Ｆ－１Ａ（ＳＨ７０５１Ｆ）
ＡＰ－ＳＨ２Ｆ－２Ａ（ＳＨ７０４４Ｆ）
等が該当します。
いずれも内蔵ＲＯＭ有効モードです。
（アドレスはそれぞれ異なります）

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

この例では、ＥＰＲＯＭ上にモニタプログラムを配置し、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の配置例です。

設定値は
ベクタ先頭アドレス　400000h
ユーザプログラム先頭アドレス　200000h
ユーザスタック初期アドレス　4FF000h
ターゲットモニタ用 RAM 領域　4FF000h
となります。

弊社ＣＰＵボード
ＡＰ－ＳＨ２－０Ａ（ＳＨ７０４３）
ＡＰ－ＳＨ３－０Ａ（ＳＨ７７０９）
等　が該当します。
（アドレスはそれぞれ異なります）

＊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。

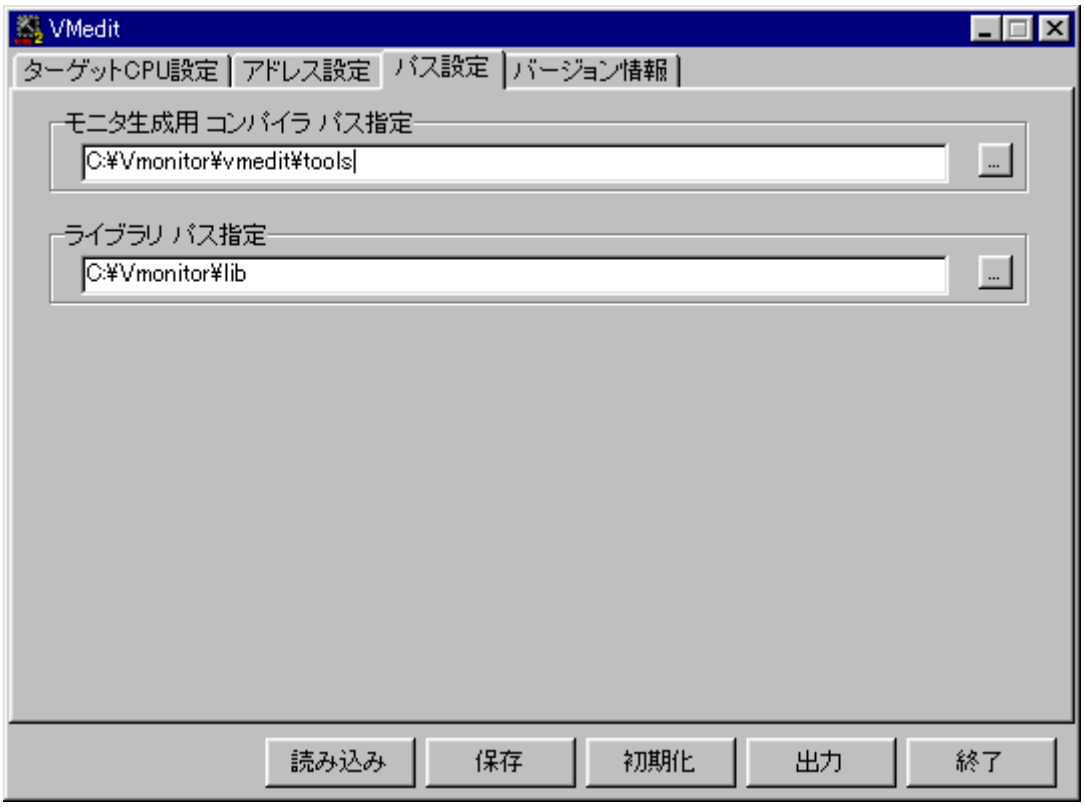
＜内蔵フラッシュを使用する場合の注意事項＞

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	0xffffe800 ～ 0xffffedff

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

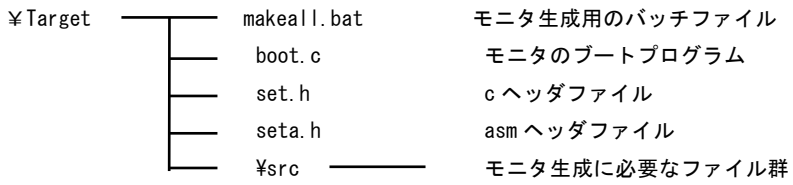
設定ファイルを保存します。
保存ボタンをクリックすると、ファイル保存ダイアログが表示されますので任意の名前で設定ファイル(*.vme)を保存できます。

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）



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¥ﾃﾞｽｸﾄｯﾌﾟ¥VisualMonitor¥vmedit¥lib¥sh704x.a)sio.o
(C:¥WINDOWS¥ﾃﾞｽｸﾄｯﾌﾟ¥VisualMonitor¥vmedit¥lib¥sh704x.a)aplib.o
(C:¥WINDOWS¥ﾃﾞｽｸﾄｯﾌﾟ¥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¥ccIiZJ3R.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¥ccIiZJ3R.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」もしくはルネサス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ｇｃｃ（COFF）またはＳＨＣ（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（ｽﾃｰﾀｽﾚｼﾞｽﾀ）の割り込みマスクを１４に設定します。 **※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。**
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（ＳＨＣ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

＊ＲＥＡＬｉ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。

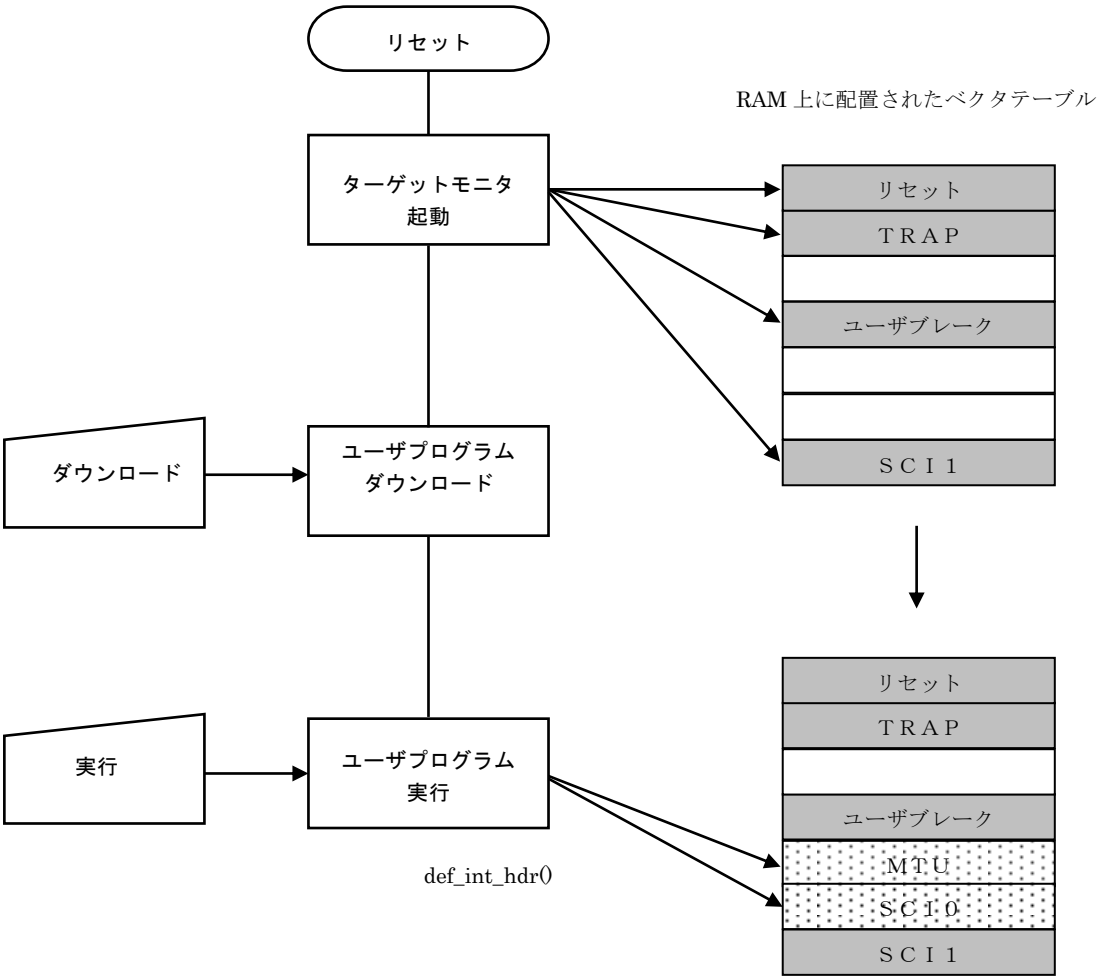
フォルダ名　　￥ＲＥＡＬｉ￥＜ＲＥＡＬｉバージョン番号＞￥＜コンパイラ名＞￥＜ボード名＞￥＜ダウンロードメモリ領域名＞

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、ＳＣＩ０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.CMSTR.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;                      /*ＴＧＲＡ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つのチャネルしか使用されませんので VisualMonitor で使用しないチャネルはユーザプログラムで使用できます。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	
ＳＣＩ０	ＥＲＩ０	１００	H'00000190－H'00000193	１ＣＨのみ使用
	ＲＸＩ０	１０１	H'00000194－H'00000197	
	ＴＸＩ０	１０２	H'00000198－H'0000019B	
ＳＣＩ１	ＥＲＩ１	１０４	H'000001A0－H'000001A3	
	ＲＸＩ１	１０５	H'000001A4－H'000001A7	
	ＴＸＩ１	１０６	H'000001A8－H'000001AB	

＜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	
ＳＣＩ０	ＥＲＩ０	１２８	H'00000200－H'00000203	１ＣＨのみ使用
	ＲＸＩ０	１２９	H'00000204－H'00000207	
	ＴＸＩ０	１３０	H'00000208－H'0000020B	
ＳＣＩ１	ＥＲＩ１	１３２	H'00000210－H'00000213	
	ＲＸＩ１	１３３	H'00000214－H'00000217	
	ＴＸＩ１	１３４	H'00000218－H'0000021B	

＜ＳＨ２　７０４６ 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
予約	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
ＳＣＩ０	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

＜ＳＨ２　７０４７ 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
予約	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
ＳＣＩ０	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

１ＣＨのみ使用

＜ＳＨ２　７０５０、７０５１　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	
予約	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	
ＳＣＩ０	ＥＲＩ０	１５２	H'00000260－H'00000263	１ＣＨのみ使用
	ＲＸＩ０	１５３	H'00000264－H'00000267	
	ＴＸＩ０	１５４	H'00000268－H'0000026B	
ＳＣＩ１	ＥＲＩ１	１５６	H'00000270－H'00000273	
	ＲＸＩ１	１５７	H'00000274－H'00000277	
	ＴＸＩ１	１５８	H'00000278－H'0000027B	
ＳＣＩ２	ＥＲＩ２	１６０	H'00000280－H'00000283	
	ＲＸＩ２	１６１	H'00000284－H'00000287	
	ＴＸＩ２	１６２	H'00000288－H'0000028B	

＜ＳＨ２　７０５２、７０５３、７０５４、７０５５　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	
予約	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	
ＳＣＩ０	ＥＲＩ０	２００	H'00000320－H'00000323	１ＣＨのみ使用
	ＲＸＩ０	２０１	H'00000324－H'00000327	
	ＴＸＩ０	２０２	H'00000328－H'0000032B	
ＳＣＩ１	ＥＲＩ１	２０４	H'00000330－H'00000333	
	ＲＸＩ１	２０５	H'00000334－H'00000337	
	ＴＸＩ１	２０６	H'00000338－H'0000033B	
ＳＣＩ２	ＥＲＩ２	２０８	H'00000340－H'00000343	
	ＲＸＩ２	２０９	H'00000344－H'00000347	
	ＴＸＩ２	２１０	H'00000348－H'0000034B	
ＳＣＩ３	ＥＲＩ３	２１２	H'00000350－H'00000353	
	ＲＸＩ３	２１３	H'00000354－H'00000357	
	ＴＸＩ３	２１４	H'00000358－H'0000035B	
ＳＣＩ４	ＥＲＩ４	２１６	H'00000360－H'00000363	
	ＲＸＩ４	２１７	H'00000364－H'00000367	
	ＴＸＩ４	２１８	H'00000368－H'0000036B	

＜ＳＨ２　７０６５ 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
予約	ＰＣ	２	H'00000008－H'0000000B
	ＳＰ	３	H'0000000C－H'0000000F
ユーザブレーク		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
ＳＣＩ０	ＥＲＩ０	１８８	H'000002F0－H'000002F3
	ＲＸＩ０	１８９	H'000002F4－H'000002F7
	ＴＸＩ０	１９０	H'000002F8－H'000002FB
ＳＣＩ１	ＥＲＩ１	１９２	H'00000300－H'00000303
	ＲＸＩ１	１９１	H'00000304－H'00000307
	ＴＸＩ１	１９４	H'00000308－H'0000030B
ＳＣＩ２	ＥＲＩ２	１９６	H'00000310－H'00000313
	ＲＸＩ２	１９７	H'00000314－H'00000317
	ＴＸＩ２	１９８	H'00000318－H'0000031B

｝１ＣＨのみ使用

＜ＳＨ２　７６０４ 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
ＳＣＩ	ＥＲＩ	７２*	H'00000120－H'00000123
	ＲＸＩ	７３*	H'00000124－H'00000127
	ＴＸＩ	７４*	H'00000128－H'0000012B

*ＳＣＩ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。

＜ＳＨ２　７１４４、７１４５　の場合＞

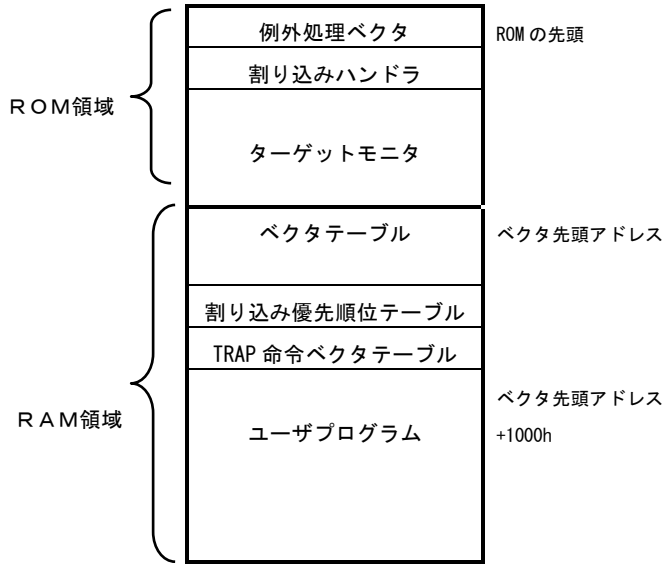
例外要因		ベクタ番号	ベクタテーブルアドレスオフセット
パワーオンリセット	ＰＣ	０	H'00000000－H'00000003
	ＳＰ	１	H'00000004－H'00000007
マニュアルリセット	ＰＣ	２	H'00000008－H'0000000B
	ＳＰ	３	H'0000000C－H'0000000F
ユーザプレーク		１２	H'00000030－H'00000033
トラップ命令（２０Ｈ）		３２	H'00000080－H'00000083
トラップ命令（２１Ｈ）		３３	H'00000084－H'00000087
ＳＣＩ０	ＥＲＩ０	１２８	H'00000200－H'00000203
	ＲＸＩ０	１２９	H'00000204－H'00000207
	ＴＸＩ０	１３０	H'00000208－H'0000020B
ＳＣＩ１	ＥＲＩ１	１３２	H'00000210－H'00000213
	ＲＸＩ１	１３３	H'00000214－H'00000217
	ＴＸＩ１	１３４	H'00000218－H'0000021B
ＳＣＩ２	ＥＲＩ２	１６８	H'000002A0－H'000002A3
	ＲＸＩ２	１６９	H'000002A4－H'000002A7
	ＴＸＩ２	１７０	H'000002A8－H'000002AB
ＳＣＩ３	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。

割り込みハンドラとベクタテーブルの配置



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す。

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、ＴＲＡＰ２０ｈ、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）

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
	ユーザブレー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
	ＩＲＱ	ＩＲＱ０	４８	H'000000C0－H'000000C3
		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
	ユーザブレー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
	ＩＲＬ３～０	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'00000053
		レベル９	２２	H'00000058－H'00000057
		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
	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
	ユーザブレー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
	ＩＲＱ	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'00000053
		レベル９	２２	H'00000058－H'00000057
		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	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。

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　void vect_table_address(unsigned long addr);
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：　vect_table_address(0x4000000); ＜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　def_int_hdr(unsigned short　intno,void (*inthdr)(void),unsigned long level);
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：　def_int_hdr(33,(void *)tmu1_5ms,7); ＜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　void def_trap_hdr(unsigned short　intno,void (*hdr)(void));
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＞　　def_trap_hdr（TRAP 番号、ベクタアドレス） 例：　def_trap_hdr(22,(void *)trap22); ＜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();                           /*ＴＲＡＰ割り込み初期化      */
          :      (途中省略)
          :
          CMT_timer_int();                       /*CMT タイマ初期化            */
          :      (途中省略)
          :
}
/***********************************************/
/*         CMT ﾀｲﾏｰｲﾆｼｬﾙ                       */
/***********************************************/
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;                 /* ５ｍｓタイマーＳＥＴ    */
          ・          (省略)
}
/***********************************************/
/*         ＴＲＡＰ割り込み初期化              */
/***********************************************/
void TRAP_INIT(void)
{
          def_trap_hdr(22,(void *)trap22);
}
/***********************************************/
/*         ＭＴＵ０タイマー割り込み（５ｍｓ）  */
/***********************************************/
void mtu0_5ms(void)
{
static char         flag;
unsigned char       wk;
                    ・          (省略)
}
/***********************************************/
/*         ＴＲＡＰ２２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つのチャネルしか使用されませんので VisualMonitor で使用しないチャ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）	
ユーザブレークポイントトラップ		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	１ＣＨのみ使用
	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	

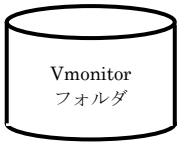
＜ＳＨ３　ＳＨ７７０８、ＳＨ７７１８Ｒ の場合＞

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）
ユーザブレークポイントトラップ		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

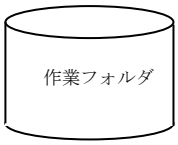
第５章. デバッグ時の使用ファイル

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　ＳＨＣの場合は、拡張子が　.abs、ｇｃｃ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で変更して下さい。

＜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。

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。

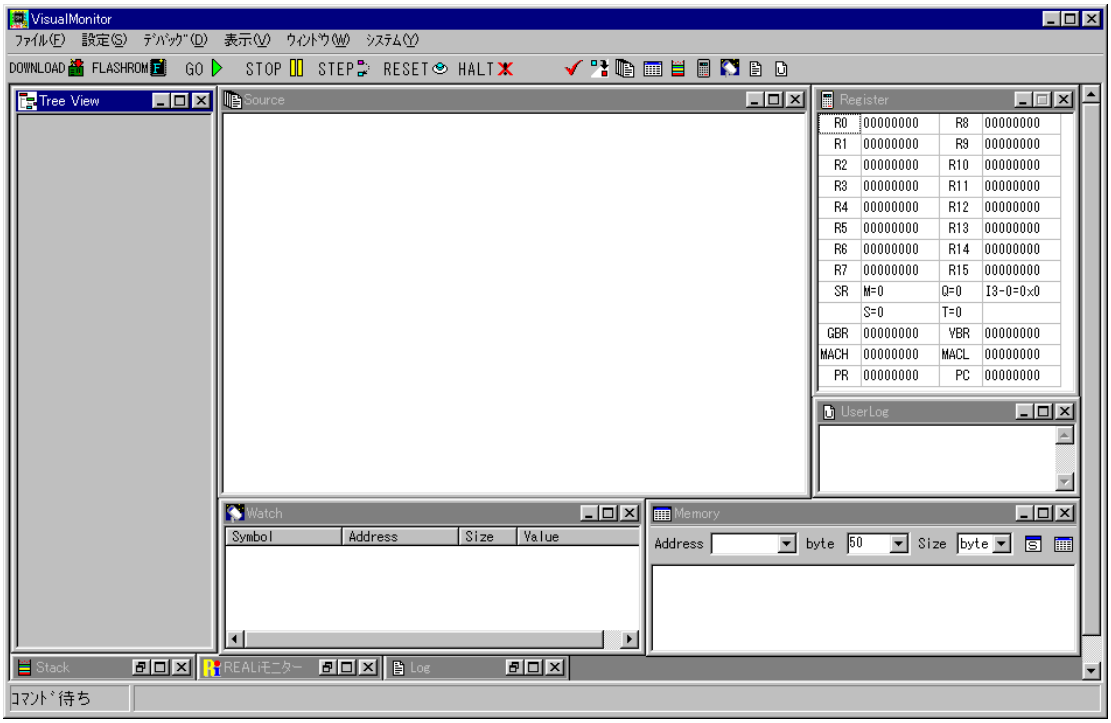
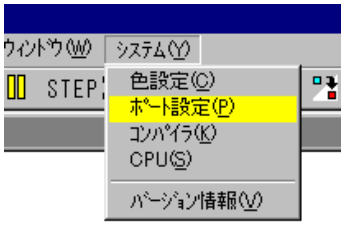
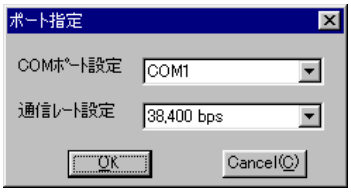


図６．１．１　起動画面（ウィンドウ位置は前回起動時と同じ位置となります）

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。

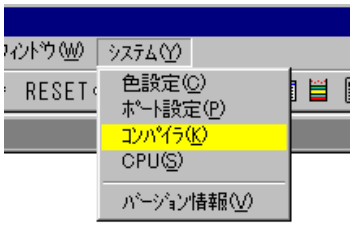


１．システムメニューよりポート設定を選択



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。

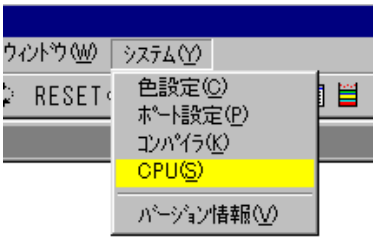
③　デバッグ対象となるプログラムのコンパイラを設定します。
次回起動時は、設定する必要はありません。



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。

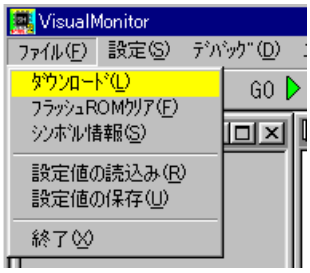
④　フラッシュＲＯＭをクリアします。
ＲＡＭにユーザプログラムをダウンロードする場合にはこの操作は必要ありません。

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。

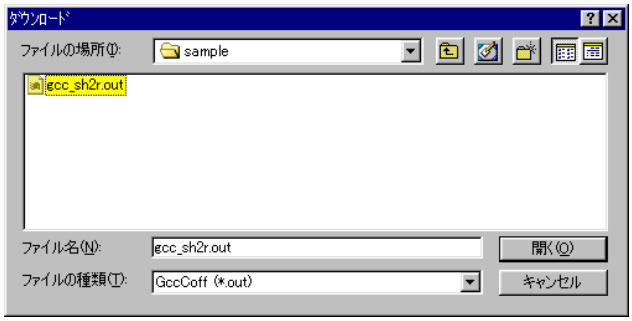
２．ＦｌａｓｈＲＯＭクリアのダイアログが表示されますので、ＡＬＬを選択してＯＫをクリックします。
ＡＬＬを選択してもモニタ領域は消去されません。
特定のセクタのみを消去したい場合には、消去するセクタ番号を選択してＯＫをクリックします。

⑦　ユーザプログラムをダウンロードします。

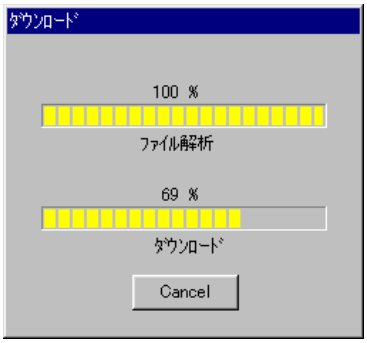


１．ファイルメニューもしくはツールバーからダウンロードを選択します。

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



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。
拡張子のデフォルトは
　日立は　　＊．ＡＢＳ
　ＧＣＣは　＊．ＯＵＴ
となっています。



３．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。

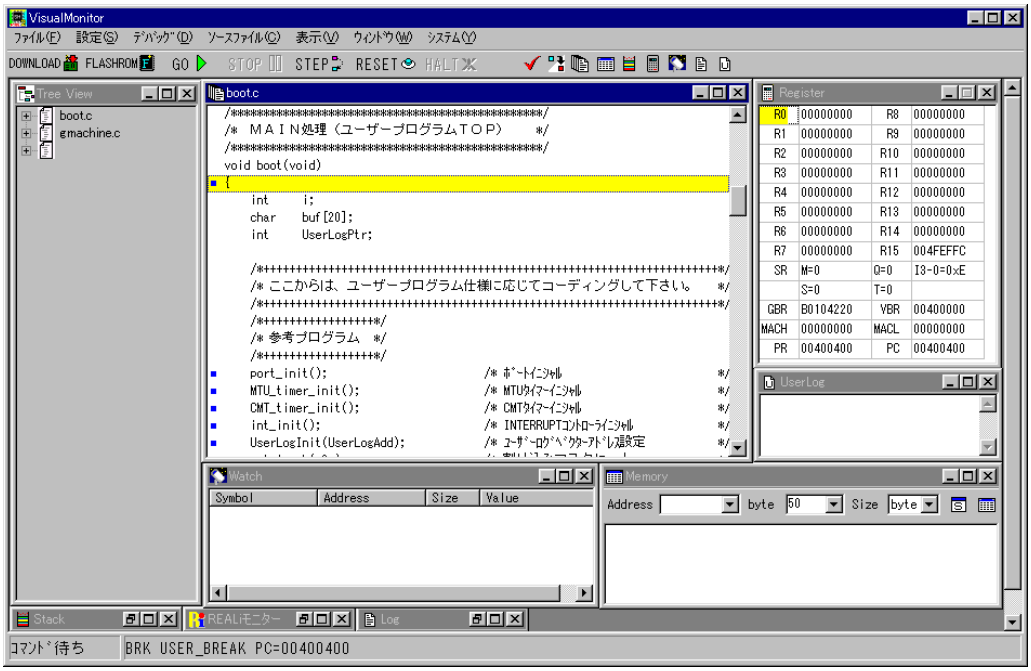
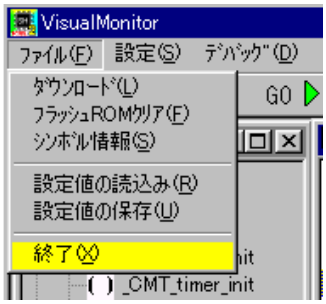


図６．１．８　ダウンロード後の画面

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。

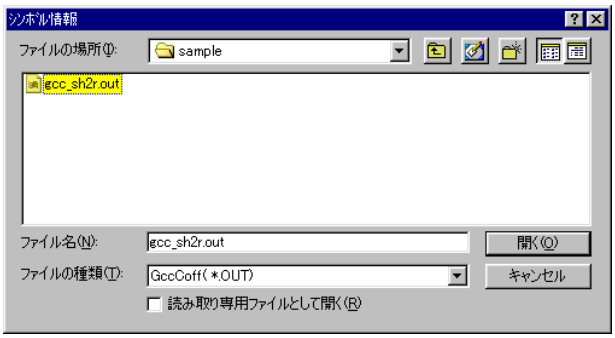
６．２　ＲＯＭ上のプログラムのデバッグ

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。

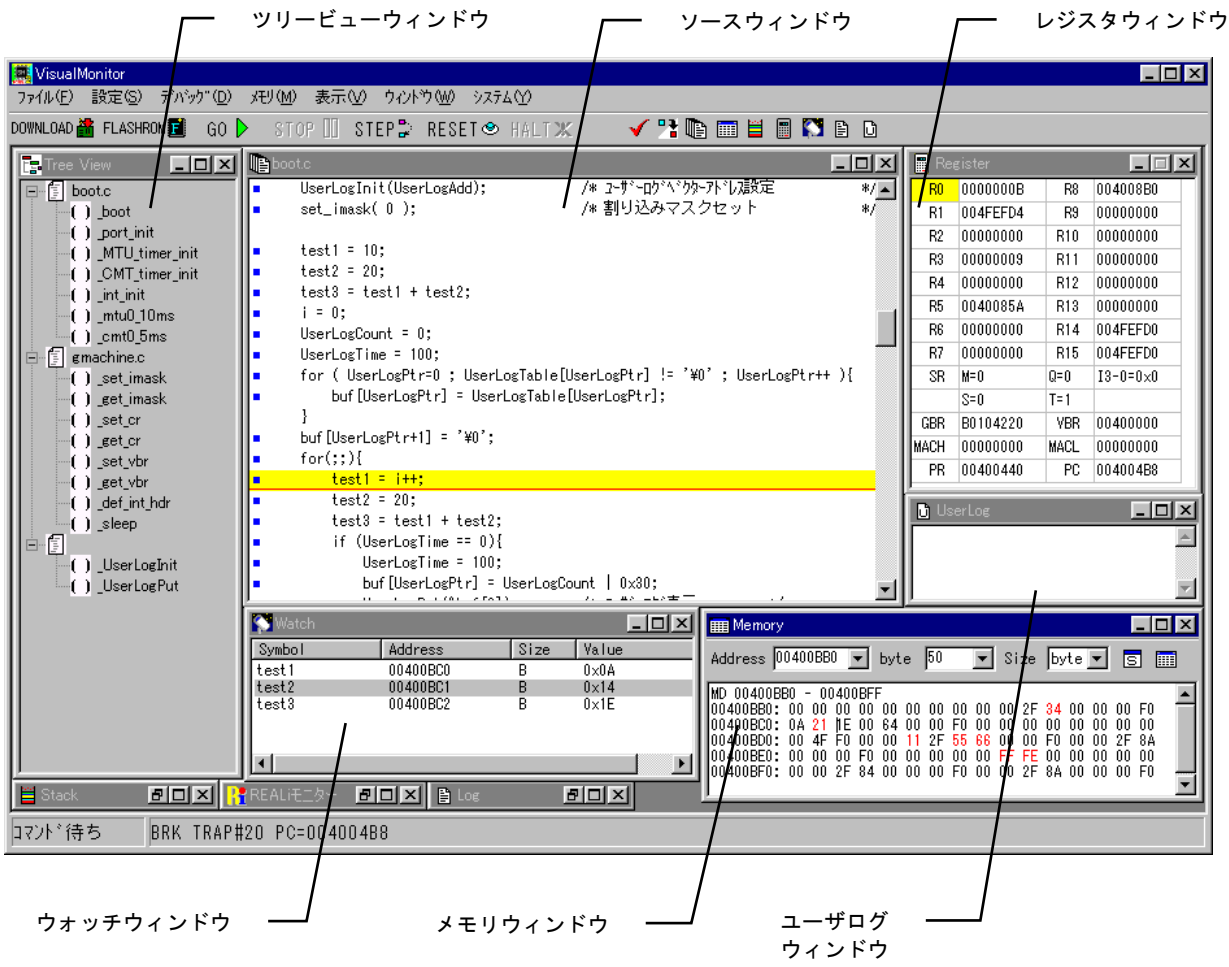
デバッグ機能の制限

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

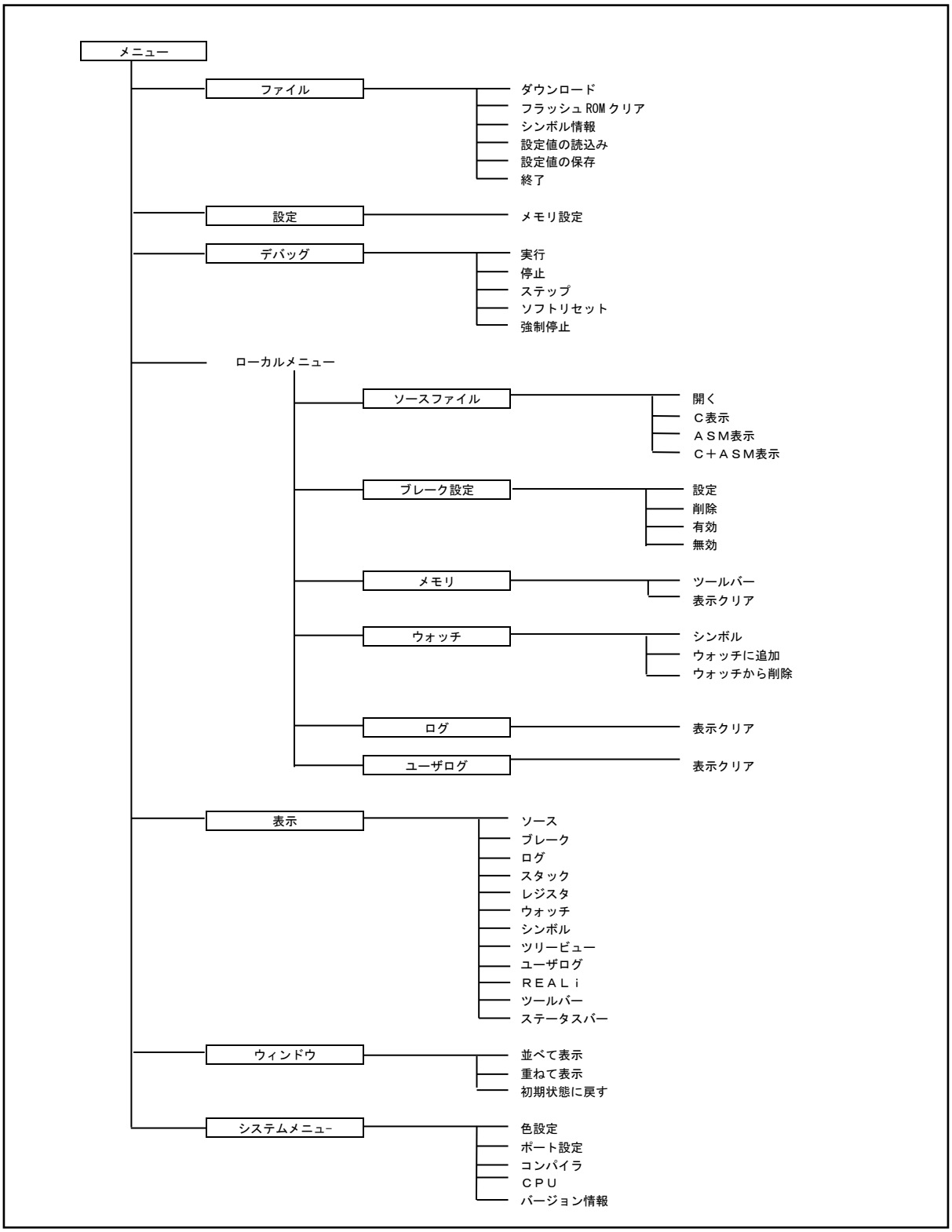
VisualMonitor の主なウィンドウには以下のものがあります。



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）	Ｆ５	プログラムの実行を行う。
停止（Ａ）	ＥＳＣ	プログラムの強制停止を行う。
ステップ（Ｔ）	Ｆ９	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）	Ｆ１	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	Ｆ１２	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。

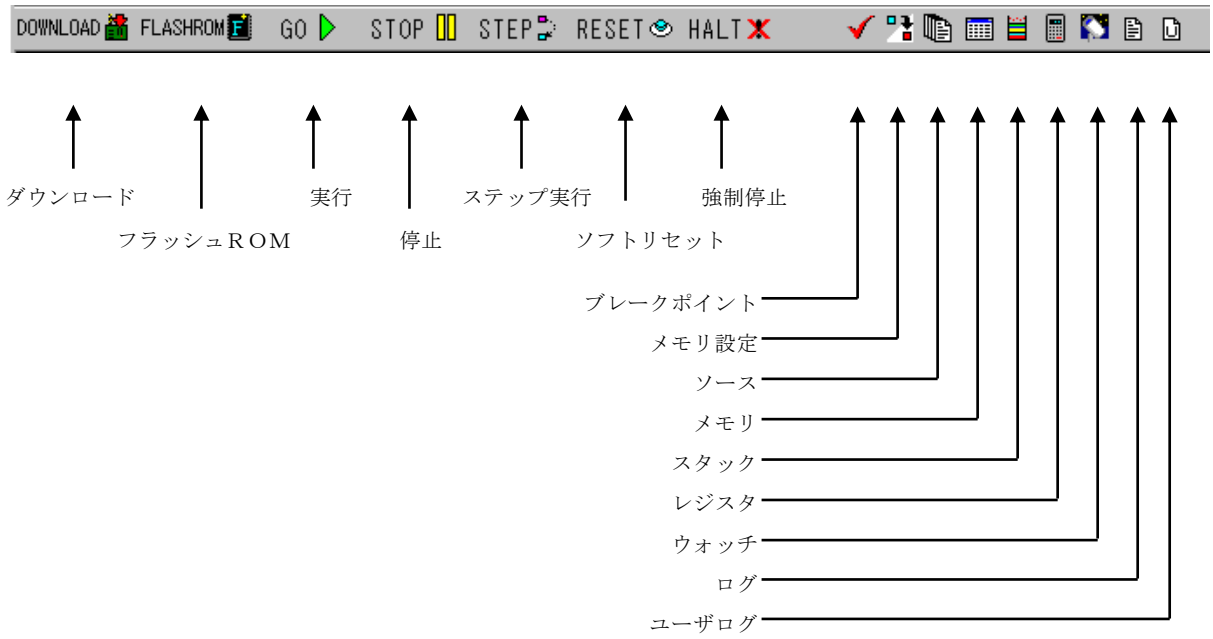
7．2．7　システムメニュー　(Alt+F)

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。

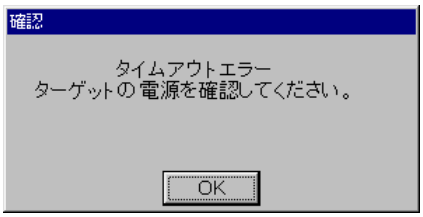


7．4　ステータスバー

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

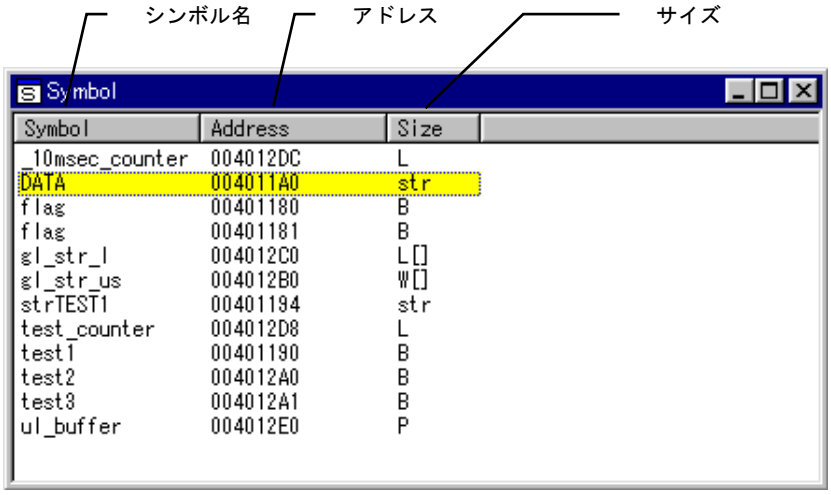
動作確認メッセージやエラーメッセージなどを表示するダイアログです。



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ー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。

シンボル表示ボタン



シンボル情報ダイアログ

＜表示＞

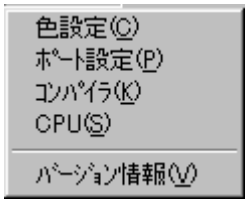
項目	表示内容
Symbol	シンボル名を表示します。 グローバルシンボルのみ表示されます。
Address	シンボルのアドレスを表示します。
size	シンボルのサイズを表示します。

＜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

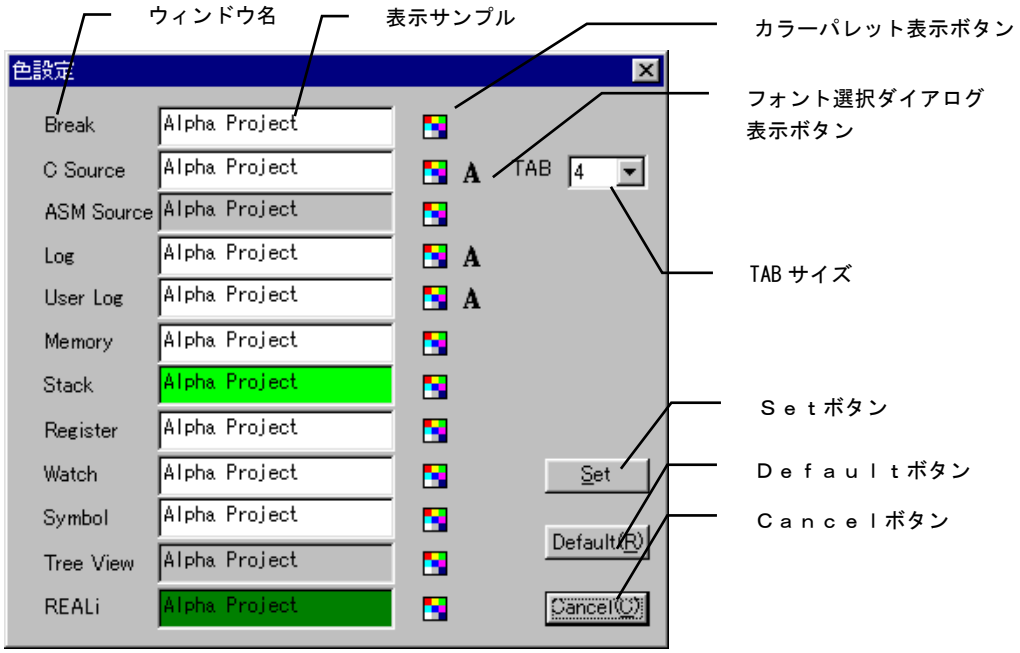
デバッグをおこなうためには、基本的な環境を設定する必要があります。
それらの設定はシステムメニューからおこないます。



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＞

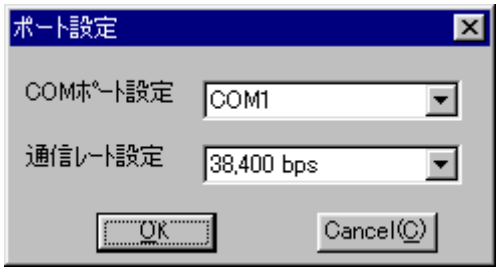
コントロール	機能
ウィンドウ名	対応する各ウィンドウ名を表示します。
表示サンプル	選択した設定でのサンプルが表示されます。

＜設定項目＞

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

＜設定項目＞

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。

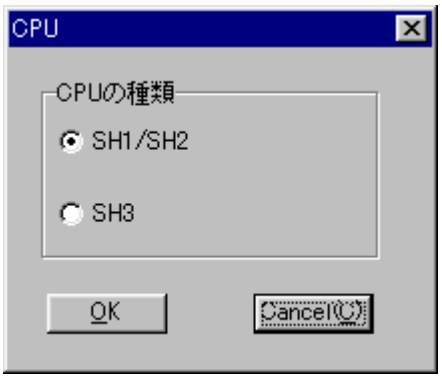
７．７．３　コンパイラ設定

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。



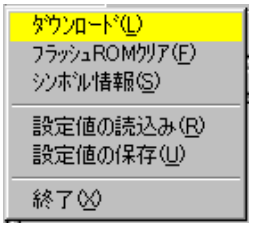
ＣＰＵ設定ダイアログ

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＞

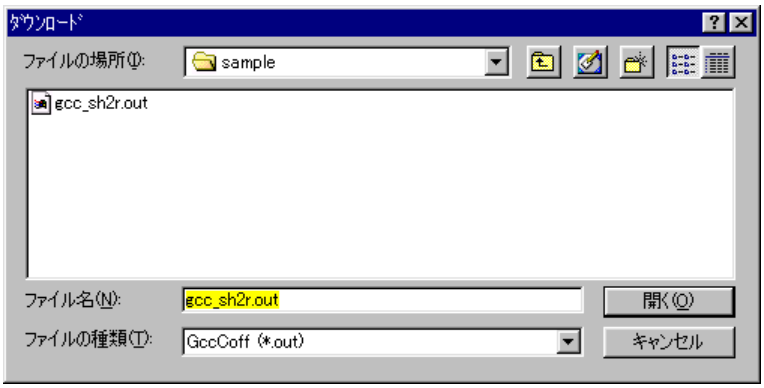
① メニューもしくはツールバーからダウンロードを選択します。



ファイルメニュー

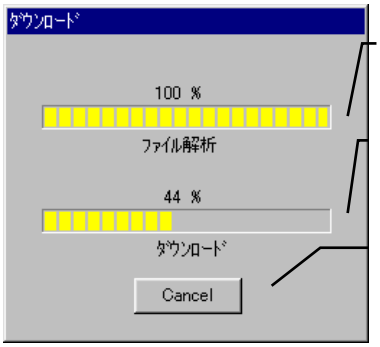
ツールバー

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、デバック情報ファイルです。
ＳＨＣ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

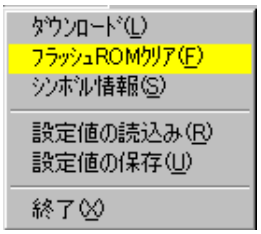
VisualMonitor は選択されたファイルのデバッグ情報を解析した後にダウンロードを開始します。
ダウンロード時間は、デバッグ環境によって変化しますが、１００Ｋ以上のプログラムの場合は１分以上かかる場合もあります。
ダウンロードを高速におこなうため、できるだけ高い通信レートでデバッグされることをお勧めします。（ターゲットモニタの作成を参照）

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。



ファイルメニュー

ツールバー

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－ＺＴＡＴを使用する場合、通常、ＥＢ０（SH7055 は EB0～EB7）に VisualMonitor を配置します。
したがって、ＡＬＬを指定した場合はＥＢ０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－ＺＴＡＴ）を使用する場合には、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 もしくはルネサス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、ＵＢＣ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ＵＢＣ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。

※ 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。

７．１０．４　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ＰＣとＳＰ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

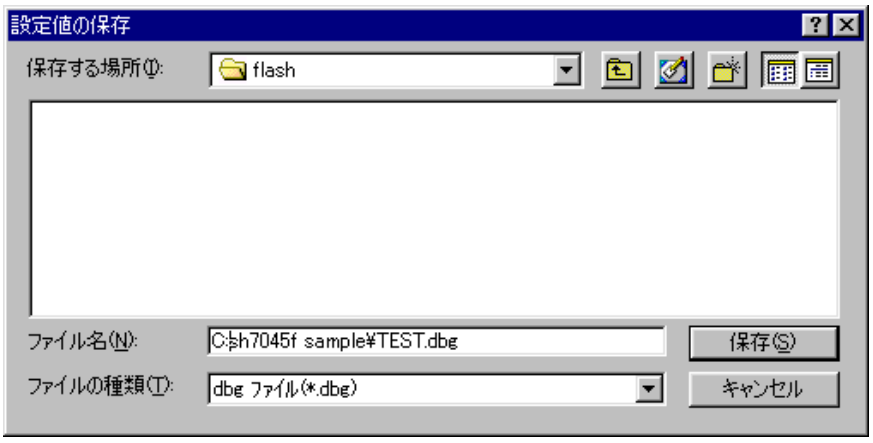
設定値の読み込みと保存とは、デバッグ中の各ウィンドウの設定状態をファイルから読み込んだり、保存する機能です。
同じプログラムを繰り返しデバッグをする場合などに、便利な機能です。

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。

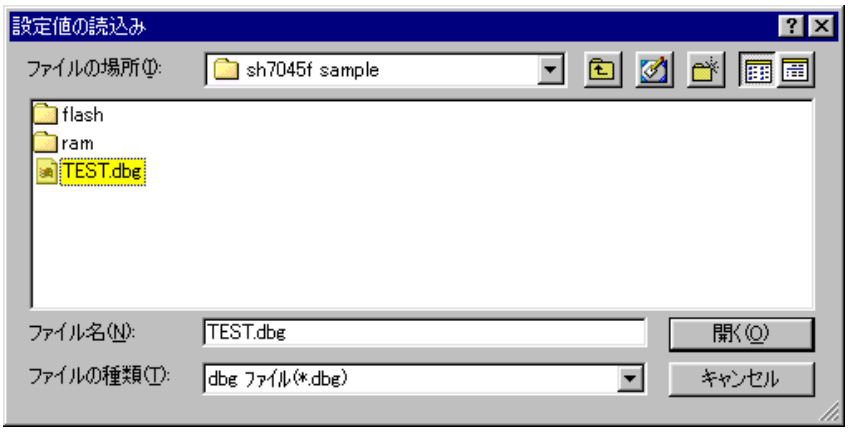
機能	保存内容
ソースウィンドウ	ソースの表示状態
ブレークポイント	全ブレークポイントの設定
ウォッチウィンドウ	ウォッチ変数の登録状態
メモリウィンドウ	表示範囲のアドレス

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。

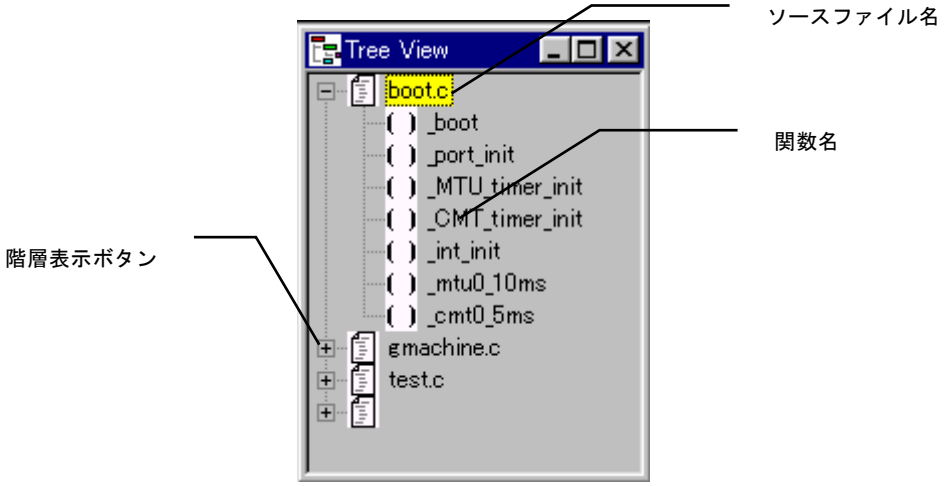


図７．１３　ツリービューウィンドウ

＜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。

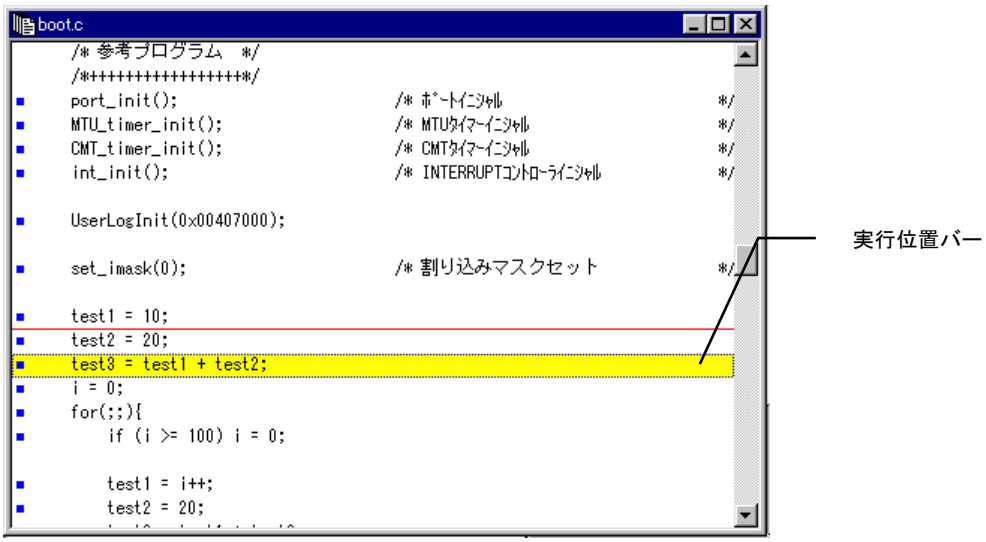
＜マウス操作＞

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＋ＡＳＭ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。
ＡＳＭ表示（Ａ）	－	アセンブラソースを表示します。
Ｃ＋ＡＳＭ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。

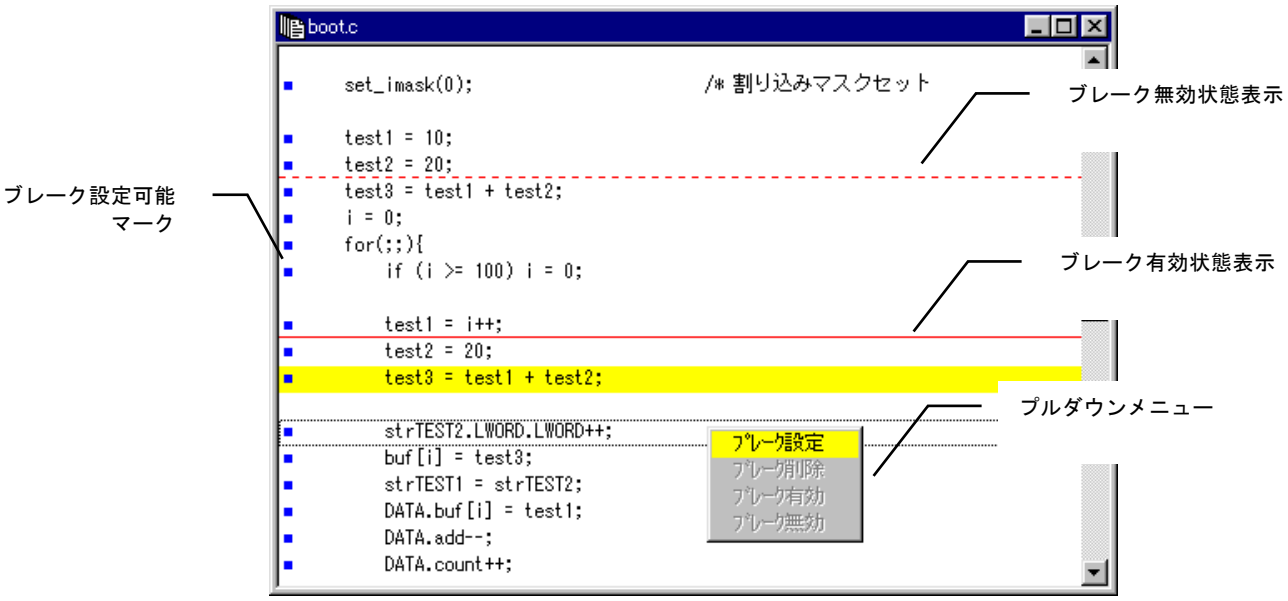


図７．１４．２　ソースウィンドウのブレーク設定

＜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。

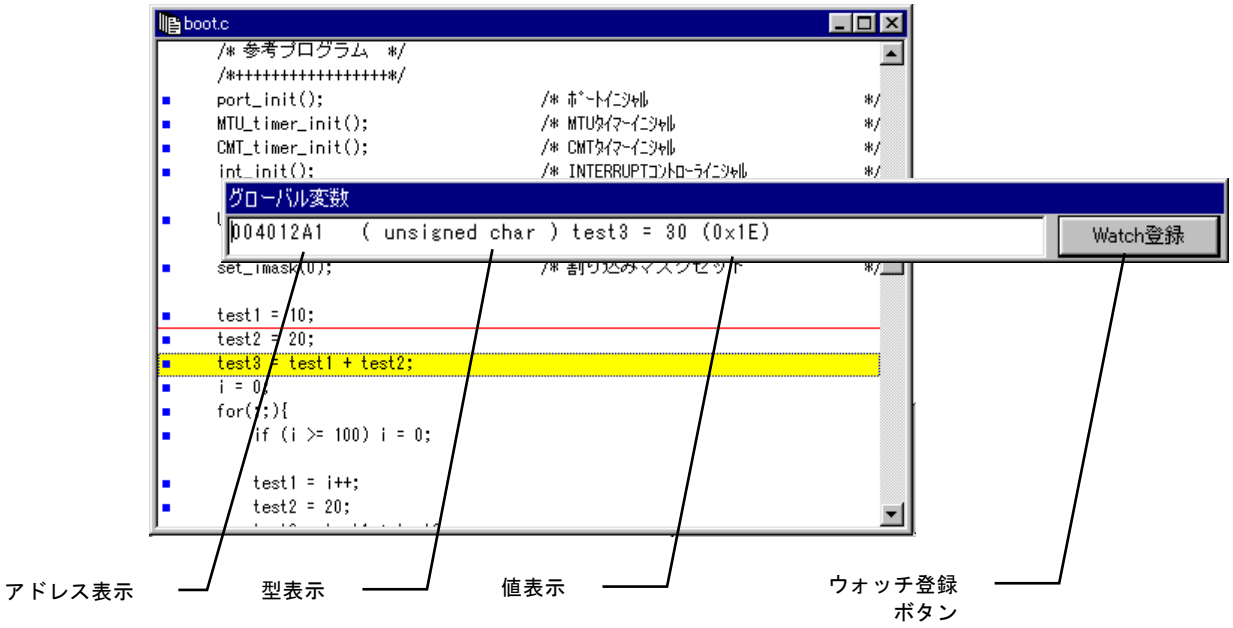
④無効にした場合は、一時的にブレークポイントが解除され赤の点線に変わります。

⑤コンフィグレーションでブレーク設定がＵＢＣ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。

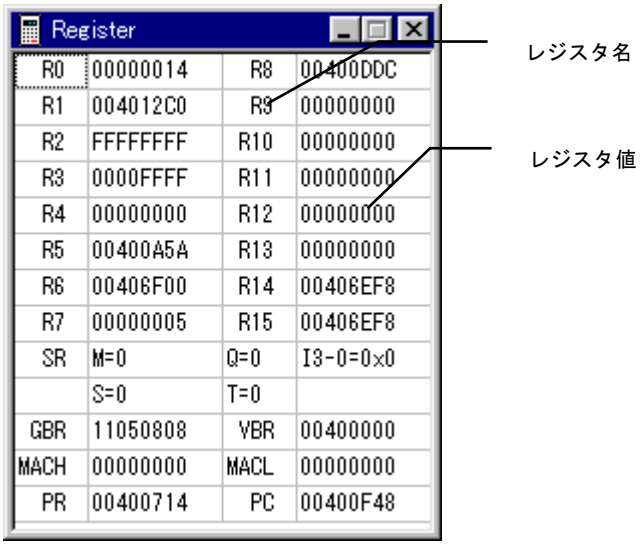
＜表示＞

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。



＜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。

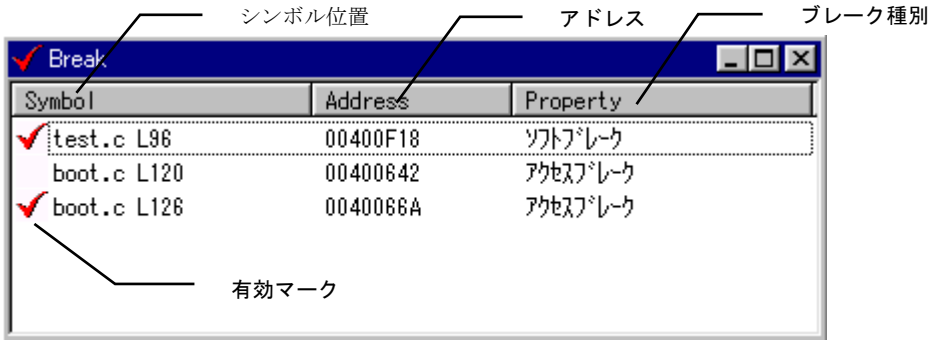
レジスタ値設定ウィンドウ

レジスタ値を１６進で入力します。

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。

＜キーボード操作＞

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　詳細なブレーク条件の設定

ブレークポイントウインドウで、より詳細なブレークの条件を設定する場合にはブレー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。

＜シンボル表示ボタン＞

前述の「シンボル表示入力」を参照してください。

＜設定項目＞

項目		表示内容
アドレス		ブレークポイントを設定するアドレスを入力します。
アクセスブレーク		アクセスブレー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のユーザブレー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/書きこみ	読み出し/書きこ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「ユーザブレークコントローラ」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。

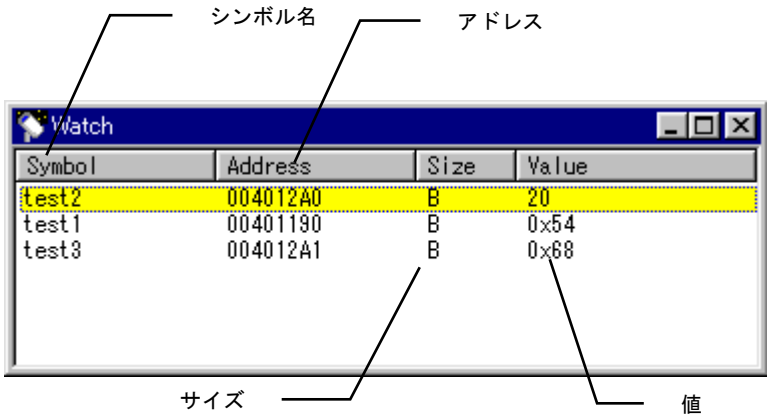
・ＳＨ７０６５をデバッグ時において、アクセスブレークと使用した場合は､ブレークポイントの次の行でプログラムが停止します。
これは、ＵＢＣの仕様のためです。ＳＨ７０６５以外のＣＰＵの場合ブレークポイントとして指定したアドレスにある命令の実行前に停止しますが、ＳＨ７０６５は、命令の実行後に停止してしまうためです。

※ 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＞

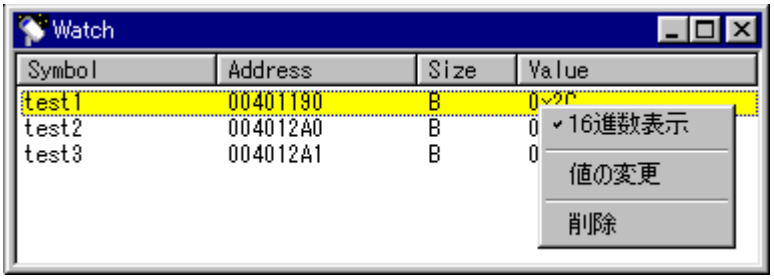
メニュー項目	ショートカット	機能
シンボル（Ｓ）	ー	シンボル情報ダイアログを表示する。
ウォッチに追加（Ａ）	ー	ウォッチウィンドウに登録を追加する。
ウォッチから削除（Ｄ）	ー	ウォッチウィンドウから登録を削除する。

＜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）

＜プルダウンメニュー＞

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

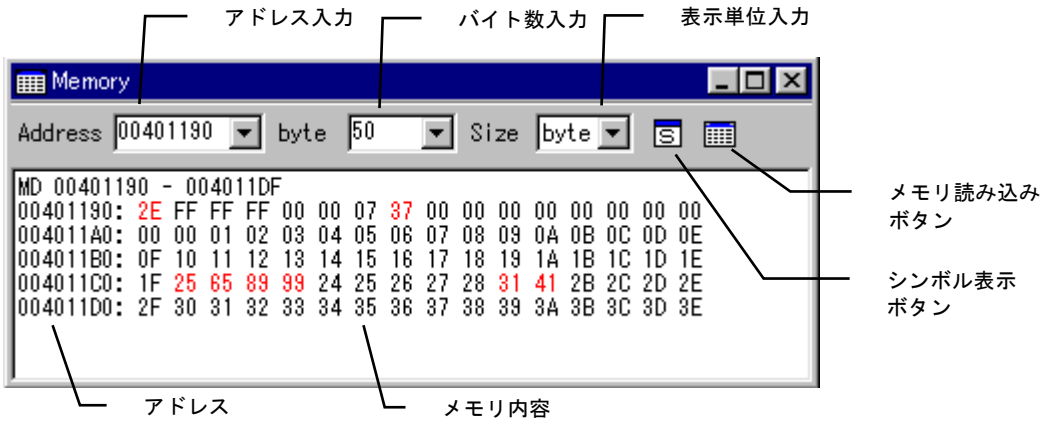
＜変更ダイアログ＞

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	表示するメモリの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＞

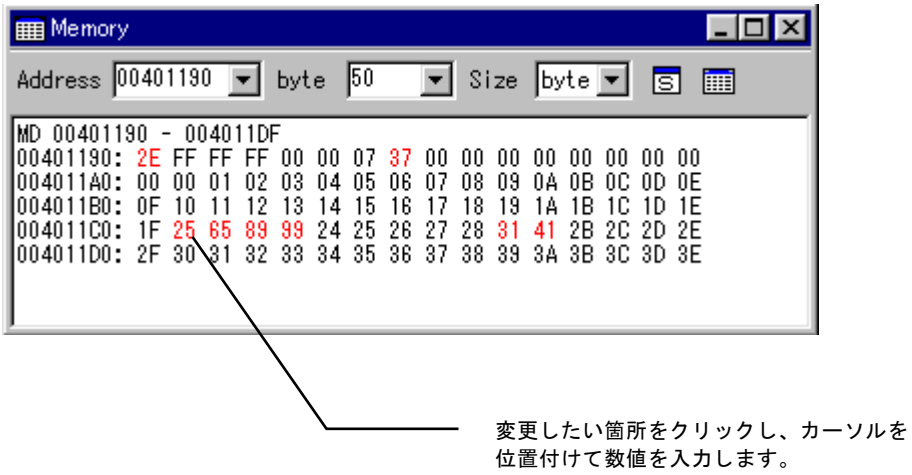
メニュー項目	ショートカット	機能
ツールバー（T）	－	ツールバーの表示／非表示を行う。 チェックマークが付いている場合にはツールバー表示が有効である。
表示クリア（C）	－	メモリ内容の表示をクリアします。。

＜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。



＜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。

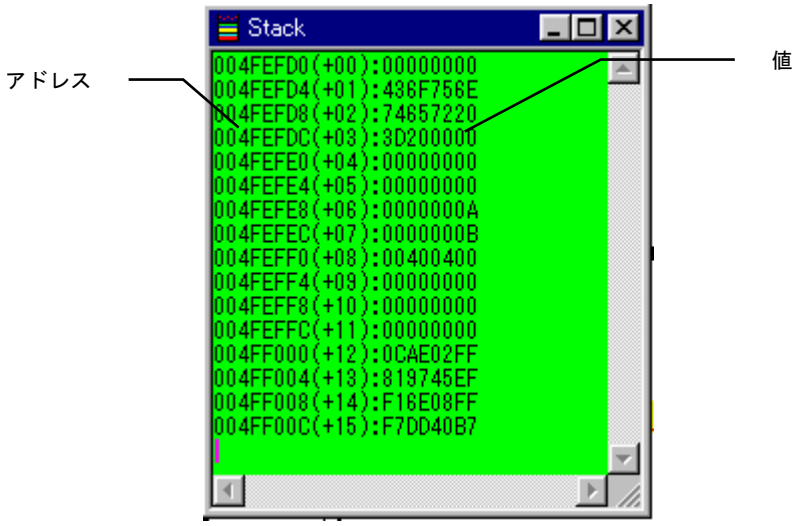
② 編集したい箇所をクリックすると、カーソルが表示されますので数値を入力します。
入力値は１６進数で入力します。

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。



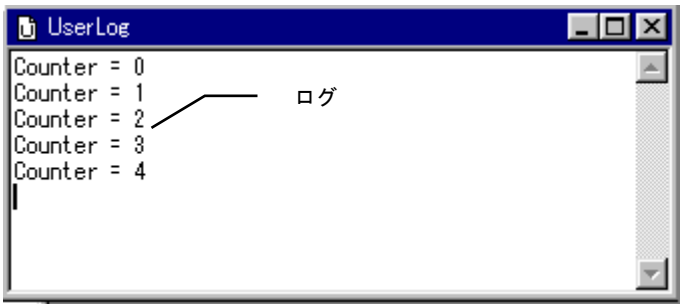
＜表示＞

項目	表示内容
アドレス	スタックのアドレスを表示しまします。 ＋０が現在のスタックポインタで、相対アドレスで＋１５（６４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	（ＳＨＣ用）
	ＳＨ１ＵＬＤ．ＬＩＢ	（ＳＨＣＶｅｒ６用）
	ＳＨ１ＵＬ．Ａ	（ｇｃｃ用）
ＳＨ２用	ＳＨ２ＵＬ．ＬＩＢ	（ＳＨＣ用）
	ＳＨ２ＵＬＤ．ＬＩＢ	（ＳＨＣＶｅｒ６用）
	ＳＨ２ＵＬ．Ａ	（ｇｃｃ用）
ＳＨ３用	ＳＨ３ＵＬ．ＬＩＢ	（ＳＨＣ　ビッグエンディアン用）
	ＳＨ３ＵＬＤ．ＬＩＢ	（ＳＨＣＶｅｒ６　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	（ＳＨＣ　リトルエンディアン用）
	ＳＨ３ＬＵＬＤ．ＬＩＢ	（ＳＨＣＶｅｒ６　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）
    }
}
```

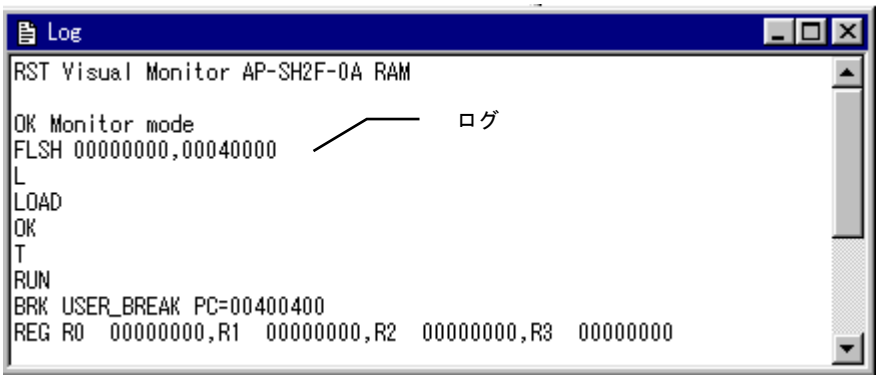
注意

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。

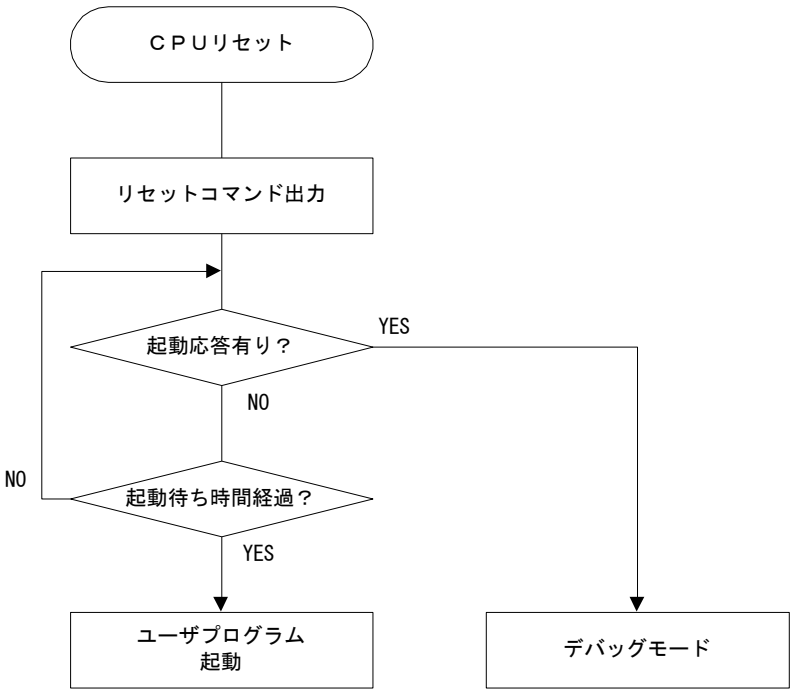
注意）ＲＥＡＬｉのシステム初期化がされていない状態では、ＲＥＡＬｉの情報項目は不定値となります。
ＲＥＡＬｉの情報項目の詳細はＲＥＡＬｉのマニュアルを参照して下さい。
またＲＥＡＬｉはデバッグ付きオプションでコンパイルされている必要があります。
ＲＥＡＬｉは、ＳＨ７０５５Ｆ、ＳＨ７０６５Ｆ、ＳＨ３には対応していません（2002 年 10 月時点）

第８章. 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ＳＣＩを使用して、ホストコンピュータと９６００ｂｐｓ～１１５．２Ｋｂｐｓで通信します。
ＳＣＩの原クロックはＣＰＵ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ＣＰＵ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ＣＰＵの内蔵ＳＣＩを使用してリモート動作します。
したがって、内蔵ＳＣＩを１チャネル、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使用することが可能です。

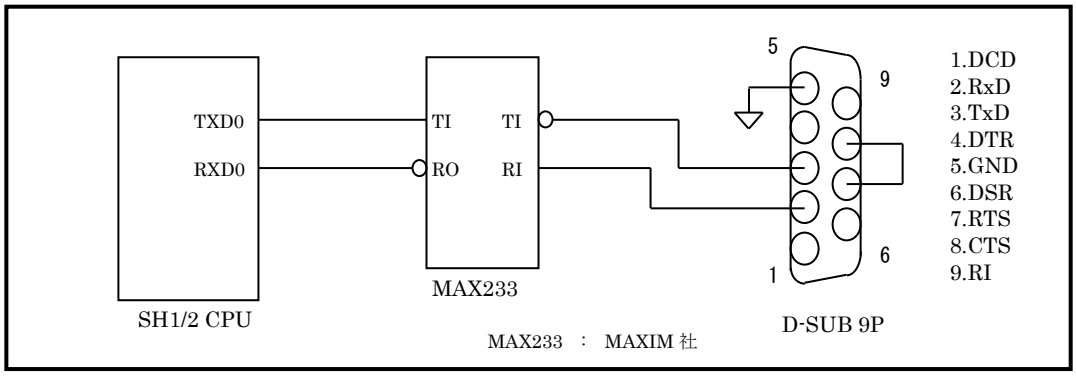


図９．１．２ ＳＣＩ０を使用した設計例

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

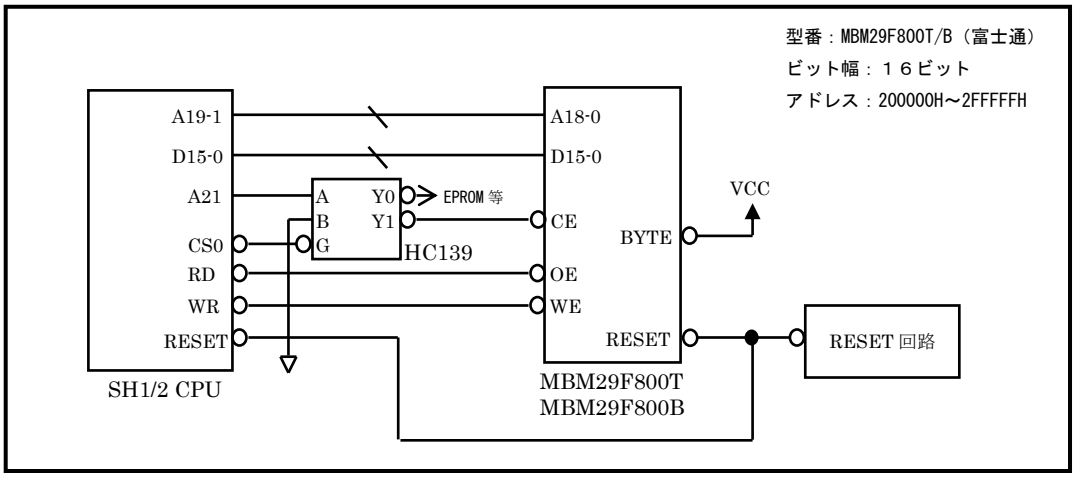


図９．１．４ ①フラッシュＲＯＭの接続例

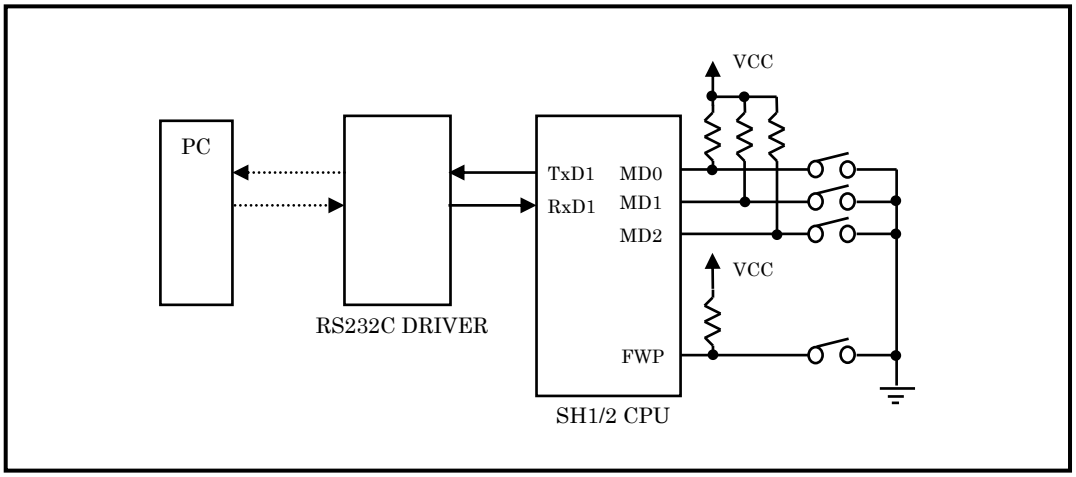


図９．１．４ ②Ｆ－ＺＴＡＴ版ＣＰＵの設計例

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のでご了承ください。**

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*￥
ＳＨＣ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	ＳＨ１
ap_sh2_0a	SH7040、7041、7042、7043	ＳＨ２
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	ＳＨ３
ap_sh3_1a	SH7709A	
ap_sh3_2a	SH7709S	
ap_sh3d_0a	SH7729	
ap_sh3d_1a	SH7729R	
sh7604	SH7604	ＳＨ２
sh7052	SH7052	
sh7053	SH7053	
sh7054	SH7054	
sh7055	SH7055	
sh7708	SH7708S/R	ＳＨ３
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

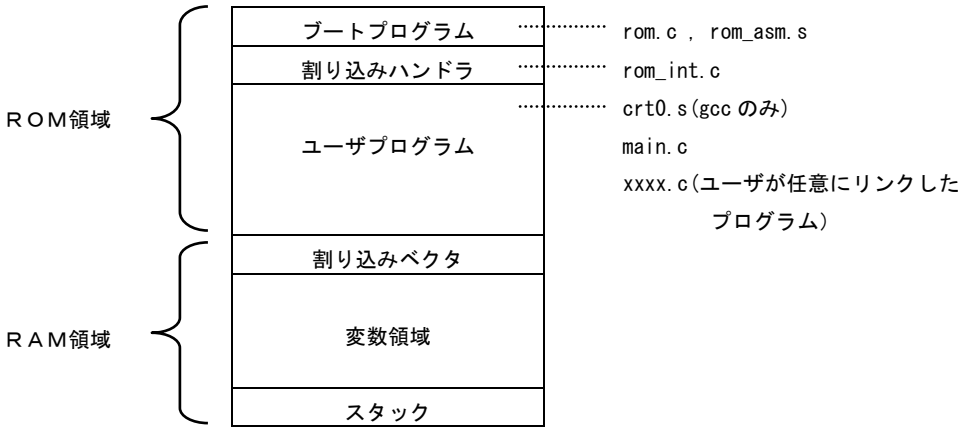
フォルダ内のコンパイル用サンプルファイル

XXXXXXXX.bat	生成用バッチファイル（ＧＣＣ）
XXXXXXXX.x	生成用リンカ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（ＳＨＣ　Ver6以前）
makeall6.bat	生成用バッチファイル（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。



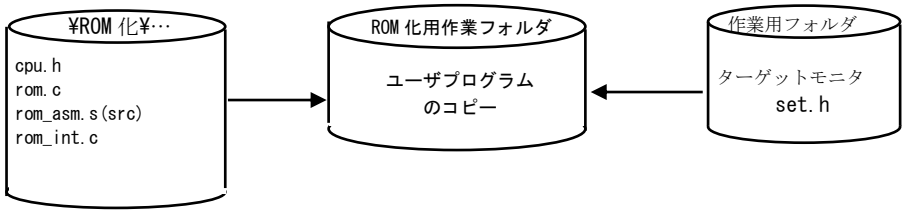
デバッグ用プログラムとの違い

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¥vmedit¥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。
（ＳＨＣ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。

＜各ファイルの変更点＞

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（ＳＨＣ、ＳＨＣＶｅｒ６）　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、ＳＨＣ、ＳＨＣＶｅｒ６）

```
void rom_boot(void)
{
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
/*  ＜カスタマイズ＞                                                         */
/*        ターゲットモニタ作成時に初期設定を以下にコピーし                   */
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

          WordWrite(0xFFFFFF60,0x3010);       /*  BSC BCR1*/
                                              /* エリア３を EDO DRAM アクセス */
                                              /* エリア４を SRAM アクセス     */

          WordWrite(0xFFFFFF62,0x35f0);       /*  BSC BCR2*/
                                              /* CS5 のバス幅 8 ビット        */
                    ・（省略）
```

初期化処理を記述します。
基本的にはターゲットモニ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。

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

rem ROM 化用プログラムを付け加える （ここから↓）          ※1
echo rom_asm.o                         >> sh2f_0a.lnk
echo rom.o                             >> sh2f_0a.lnk
echo rom_int.o                         >> sh2f_0a.lnk
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

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 c:¥VMonitor¥lib¥sh2ul.a             >> sh2f_0a.lnk

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

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
```

ＲＯＭ化用プログラムを追加する。

Userlog 用ライブラリを外す。

ＲＯＭ化用プログラムを追加する。

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 = 0x000ffbf0
                              /*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

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↓）                    ※1
#SRCS    = crt0.s main.c gmachine.c
#OBJS    = crt0.o main.o gmachine.o
SRCS    = crt0.s main.c gmachine.c rom_asm.s rom.c rom_int.c
OBJS    = 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↑）                    ※1

# UserLog 用のライブラリはリンクから外してください                ※2
#LIBS    = ${LIBPATH}SH2UL.a
LIBS    =

${TARGET}.out:${OBJS} ${TARGET}.x makefile
        ${LD} ${LDFLAGS} -o ${TARGET}.out ${OBJS} ${LIBS}
# S-FORMAT へのコンバート                                        ※4
        ${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↓）                    ※3
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↑）                    ※3
```

※1 ＲＯＭ化用プログラムを追加する。

※2 Userlog 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ＧＮＵＰｒｏ）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。

ＳＨＣ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↓）                    ※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

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               ※2
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                                    ※3
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↓)                         ※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

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

lnk /subcommand=sh2f_0a.sub

rem S-FORMAT へのコンバート                                         ※6
cnvs sh2f_0a
```

ＲＯＭ化プログラムを追加する。

User log 用ライブラリを外す。

スタート関数を指定する。

コメントアウト

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

ＲＯＭ化プログラムを追加する。

Ｓフォーマット変換処理を追加する。

ＳＨＣVer6 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

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

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

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

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

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

rem 出力形式を S-FORMAT に指定する                                 ※５
echo FORM=stype                                 >> sh2f_0a.sub

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

optlnk -subcommand=sh2f_0a.sub
```

ROM 化ファイルを出力するため debug オプションを外す。

ＲＯＭ化プログラムを追加する。

Userlog 用ライブラリを外す。

スタート関数を指定する。

出力ファイル名を変更する。

ＲＯＭ領域のマッピングを変更する。

出力フォーマットをＳフォーマットに指定する。

ＲＯＭ化プログラムを追加する。

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ておりますので、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。

Ｑ１．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。？

Ａ１．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こと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。

Ｑ２．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。？

Ａ２．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

Ｑ３．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。？

Ａ３．ヘッダ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。

Ｑ４．シリアルインターフェースの条件は。？

Ｑ４．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。

Ｑ５．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。？

Ｑ５．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。

Ｑ６．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。？

Ｑ６．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。

Ｑ７．ｇｃｃのバージョン及び種類は限定されますか？

Ａ７．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）

Ｑ８．ＶｉｓｕａｌＭｏｎｉｔｏｒをＦ－ＺＴＡＴ書き込みツールとして利用可能ですか。？

Ａ８．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 ＰＲＯ２」をご利用ください。

Ｑ９．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。？

Ａ９．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。

Ｑ１０．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。？

Ａ１０．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。

Ｑ１１．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？

Ａ１１．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。

Ｑ１２．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てしまった。

Ａ１２．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。

Ｑ１３．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。

Ａ１３．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。

Ｑ１４．ＲＥＡＬｉのモニタウィンドウがアクティブになりません。

Ａ１４．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。

Ｑ１５．スタックのアドレスエラーが発生するのですが？

Ａ１５．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。

Ｑ１６．ダウンロード中にエラーが頻繁に発生する。

Ａ１６．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。

Ｑ１７．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。

Ａ１７．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。

Ｑ１８．exeGCC の Ver.2 でデバッグができないのですが？

Ａ１８．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。

Ｑ１９．StarterKit からバージョンアップしたが、うまく動作しない

Ａ１９．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。

Ｑ２０．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？

Ａ２０．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）

変更前の makeall.bat

```
echo SECTIONS {                  >  vmsh_mon.cmd
echo     .text 0x00000000 :      >> vmsh_mon.cmd
echo     {                       >> vmsh_mon.cmd
echo         __stext  = . ;      >> vmsh_mon.cmd
echo         *(.vector)          >> vmsh_mon.cmd
echo         *(.text)            >> vmsh_mon.cmd
echo         __etext  = . ;      >> vmsh_mon.cmd
echo     }                       >> vmsh_mon.cmd
```

→

変更した makeall.bat

```
echo SECTIONS {                  >  vmsh_mon.cmd
echo     .vect 0x00000000 :      >> vmsh_mon.cmd
echo     {                       >> vmsh_mon.cmd
echo         __stext  = . ;      >> vmsh_mon.cmd
echo         *(.vector)          >> vmsh_mon.cmd
echo     }                       >> vmsh_mon.cmd
echo     .dtcvec 0x00000400 :    >> vmsh_mon.cmd
echo     {                       >> vmsh_mon.cmd
echo         *(.dtcvec)          >> vmsh_mon.cmd
echo     }                       >> vmsh_mon.cmd
echo     .text 0x00000500 :      >> vmsh_mon.cmd
echo     {                       >> vmsh_mon.cmd
echo         *(.text)            >> vmsh_mon.cmd
echo         __etext  = . ;      >> vmsh_mon.cmd
echo     }                       >> vmsh_mon.cmd
```

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

/*============================================================*/
    .section    .dtcvec
/*================= DTC Vector Table ======================*/

    .word       0x0000
                    :
                (省略)
                    :
    .word       0x0000

    .end
```

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。

追加個所

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

ユーザ登録は、弊社ホームページ、またはFAXにて受け付けております。
ユーザ登録をしていただきますと、製品サポートのほか、ご希望のお客様には、新製品やバージョンアップをメールにてご案内させていただきます。

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パ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ける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。

・マイコンの使用方法、プログラミング、コンパイラの使用方法など、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　静岡県浜松市中央区積志町８３４

FAX　(053)401-00033　技術部 担当者宛
ホームページアドレス　：　https://www.apnet.co.jp/
E-mail アドレス　：　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～FFFFFFFFH

＜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～FFFFFFFFH

＜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～FFFFFFFFH

＜ＡＰ－ＳＨ２Ｆ－１Ａ＞

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～FFFFFFFFH

＜ＡＰ－ＳＨ２Ｆ－２Ａ＞

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～FFFFFFFFH

＜ＡＰ－ＳＨ２Ｆ－３Ａ＞

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～FFFF8FFFH（ＸＲＡＭ） アドレス：FFFFA000～FFFFAFFFH（ＹＲＡＭ）

＜ＡＰ－ＳＨ２Ｆ－４Ａ＞

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～FFFFFFFFH

＜ＡＰ－ＳＨ２Ｆ－５Ａ＞

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～FFFFFFFFH

＜ＡＰ－ＳＨ２Ｆ－６Ａ＞

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：00400000～005FFFFFH もしくは、 １Ｍｂｙｔｅ　１６ｂｉｔバス　ＣＳ１に接続 アドレス：00400000～004FFFFFH
内蔵ＲＡＭ	８Ｋｂｙｔｅ アドレス：FFFFE000～FFFFFFFFH

※ＣＳ１のバス幅はジャンパ設定によって選択できます。

＜ＡＰ－ＳＨ２Ｆ－７Ａ＞

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 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～FFFFFFFFH

※ＣＳ１のバス幅はジャンパ設定によって選択できます。

＜ＳＦ－７０４４Ｆ＞

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：FFFFF000～FFFFFFFFH

※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。

＜ＳＦ－７１４４Ｆ＞

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：FFFFE000～FFFFFFFFH

※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。

＜ＡＰ－ＳＨ３－０Ａ＞

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／６４ｍｓＣＢＲ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　I: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。

＜ＡＰ－ＳＨ３－１Ａ＞

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　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。

＜ＡＰ－ＳＨ３－２Ａ＞

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　00000000～003FFFFFH ＣＳ５接続時　14000000～143FFFFFH
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	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。

＜ＡＰ－ＳＨ３Ｄ－０Ａ＞

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　00000000～000FFFFFH 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
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。

＜ＡＰ－ＳＨ３Ｄ－１Ａ＞

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　00000000～003FFFFFH ＣＳ５接続時　14000000～143FFFFFH
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	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
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～FFFFAFFFH

＜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～FFFFBFFFH

＜ＳＨ７０５４Ｆ＞

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

＜ＳＨ７０５５Ｆ＞

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

＜ＳＨ７７０８＞

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

＜ＳＨ７７１８Ｒ＞

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　https://www.apnet.co.jp/

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
■本姓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、誤りなどお気付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処しておりません。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.