

SHマイコン開発支援パッケージ

---

## *SH2/SH3 Starter KIT*

---

リモートデバッグ編

8版 2003/9/1

**ALPHA PROJECT Co., LTD**

# 目次

## 第1章 概要 1

1. 1	特徴	1
1. 2	使用環境	1
1. 3	構成	2
1. 4	機能概要	3
1. 5	対応チップ	4

## 第2章 ユーザプログラムの作成 5

2. 1	デバッグに必要なファイル	5
2. 2	ユーザプログラムの作成上の注意	5
2. 2. 1	ユーザプログラムの制限事項	5
2. 3	サンプルプログラム	6
2. 3. 1	サンプルプログラムの構成	6
2. 4	ベクタテーブル	7
2. 4. 1	ベクタテーブルの配置と構造	7
2. 4. 2	ベクタテーブルの登録方法(SH1, SH2)	8
2. 4. 3	VisualMonitor が使用する割り込み(SH1, SH2)	9
2. 4. 4	SH3の割り込みについて	10
2. 4. 5	ベクタテーブルの登録方法(SH3)	13
2. 4. 6	VisualMonitor が使用する割り込み(SH3)	14

## 第3章 デバッグ時の使用ファイル 16

3. 1	フォルダ構成	16
3. 2	デバッグ用ファイル	17
3. 3	コンフィグレーションファイル (* CFG)	17
3. 3. 1	コンフィグレーションファイルウィンドウ	17
3. 3. 2	コンフィグレーションファイルの変更	18

4. 1	ウィンドウ構成	19
4. 2	メニュー	20
4. 2. 1	ファイルメニュー	21
4. 2. 2	設定メニュー	21
4. 2. 3	デバッグメニュー	21
4. 2. 4	ローカルメニュー	21
4. 2. 5	表示メニュー	22
4. 2. 6	ウィンドウメニュー	22
4. 2. 7	システムメニュー	22
4. 3	ツールバー	23
4. 4	ステータスバー	23
4. 5	確認ダイアログ	23
4. 6	シンボル入力	24
4. 7	システム設定	25
4. 7. 1	色設定	25
4. 7. 2	ポート設定	26
4. 7. 3	コンパイラ設定	26
4. 7. 4	CPU設定	27
4. 8	プログラムのダウンロード	28
4. 9	フラッシュROMのクリア	29
4. 10	プログラムの実行と停止	30
4. 10. 1	実行	30
4. 10. 2	停止	30
4. 10. 3	ステップ実行	30
4. 10. 4	RESET	31
4. 10. 5	強制停止	31
4. 11	メモリの設定	32
4. 12	設定値の保存と読み込み	33
4. 12. 1	設定値の保存	34
4. 12. 2	設定値の読み込み	34

4. 13	ツリービューウィンドウ	.....	35
	4. 13. 1 ツリービューウィンドウの機能	.....	35
4. 14	ソースウィンドウ	.....	36
	4. 14. 1 ソースウィンドウの機能	.....	36
	4. 14. 2 ブレークポイントの設定方法	.....	37
	4. 14. 3 ダイレクト変数表示機能	.....	38
4. 15	レジスタウィンドウ	.....	39
	4. 15. 1 レジスタビューウィンドウの機能	.....	39
4. 16	ブレークポイントウィンドウ	.....	40
	4. 16. 1 ブレークポイントウィンドウの機能	.....	40
	4. 16. 2 詳細なブレーク条件の設定	.....	41
	4. 16. 3 ブレークポイントの注意と制限	.....	42
4. 17	ウォッチウィンドウ	.....	43
	4. 17. 1 ウォッチウィンドウの機能	.....	43
	4. 17. 2 ウォッチ変数の値変更	.....	44
4. 18	メモリウィンドウ	.....	45
	4. 18. 1 メモリウィンドウの機能	.....	45
	4. 18. 2 メモリイメージの編集	.....	46
4. 19	スタックウィンドウ	.....	47
	4. 19. 1 スタックウィンドウの機能	.....	47
4. 20	ユーザログウィンドウ	.....	48
	4. 20. 1 ユーザログウィンドウの機能	.....	48
	4. 20. 2 ユーザログ出力関数	.....	48
	4. 20. 3 コーディング例	.....	49
4. 21	ログウィンドウ	.....	50
	4. 21. 1 ログウィンドウの機能	.....	50
4. 22	REAL i モニタウィンドウ	.....	51
	4. 22. 1.2 REAL i モニタウィンドウの機能	.....	51

## 第5章 その他の機能

52

5. 1	自動ブート機能	.....	52
------	---------	-------	----

## 第6章. ハードウェア

53

6. 1	ハードウェアの注意点	.....	53
6. 1. 1	ウォッチドッグタイマ	.....	53

## 第7章. 製品サポートと使用上の注意

54

7. 1	製品サポートのご案内	.....	54
7. 1. 1	弊社ホームページのご利用について	.....	54
7. 1. 2	製品サポートの方法	.....	54
7. 1. 3	製品サポートの範囲	.....	54
7. 2	使用上の注意	.....	54

## 第1章. 概要

VisualMonitor/SHは、日立SHシリーズ用に開発されたWindows 95/98/NT4.0/2000対応リモートデバッグで、従来のリモートモニタ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。(注1)       |
| ●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。(注1)            |
| ●フィールドデバッグが可能     | ターゲットに組み込むことでフィールドデバッグが可能です。         |
| ●接続が容易            | 既存のハードウェアに手を加えることなくデバッグが可能です。        |
| ●リアルタイムOS対応       | μITRON仕様準拠の弊社製リアルタイムOS「REALi」に対応します。 |
| ●F-ZTAT対応         | F-ZTATマイコンのオンボード書き込みに対応します。(注1、注2)   |
| ●日立C・GCC両対応       | 日立CとGCCの両方に対応しています。(注3)              |

注1) KIT限定版のSH7045F、SH7709A、SH7709S版では対応していません。

注2) 弊社製FLASH WRITERもしくは日立純正の書き込みツールが別途必要です。

注3) GCCはCOFFに対応しています。ELF形式(exeGCC Ver2等)には対応していません。

### 1.2 使用環境

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

### 1. 3 構成

VisualMonitorは以下の2つのソフトウェアから構成されます。

●ターゲットモニタ

ターゲットのハードウェアに搭載するソフトウェアです。

各ターゲットの構成に合わせてカスタマイズし、ROM化してターゲットに搭載します。(注1)

●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。

注1) KIT限定版ではカスタマイズはできません。

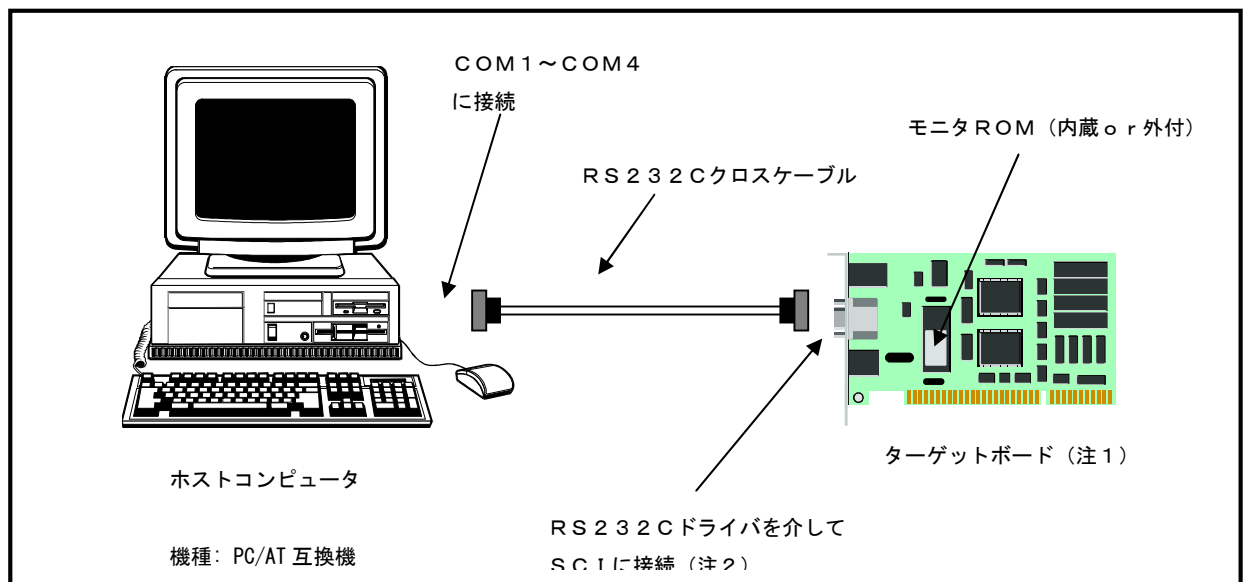


図1. 3 VisualMonitor を使用したデバッグ環境の構成例

注1) KIT限定版では、ターゲットボードは同梱の各CPUボードに固定されています。

注2) KIT限定版では、SH7045FはSCI1に、SH7046FはSCI3に固定されています。

SH7709A、SH7709SはSCIF (SCI2) に固定されています。

正規版では、任意のSCIを選択することができます。

## 1. 4 機能概要

以下にVisualMonitorのデバッグ機能一覧を示します。

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



## 1.5 対応チップ

## 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	○
SH7050シリーズ	SH7050F	AP-SH2F-1A	SH7050F	○
			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

## 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	○		

注意) KITのパッケージ限定版では同梱の各CPUボードのみに対応しています。

## 第2章. ユーザプログラムの作成

### 2. 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) (注1)または日立C (SYSROF、ELF)でコンパイルされたものに対応しています。

また、ソースプログラムを表示するために、同じフォルダにソースファイルが収録されている必要があります。

注1) ELF形式のGCC (exeGCC (京都マイコン製)等)については対応していません。

### 2. 2 ユーザプログラムの作成上の注意

#### 2. 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 (ステータレジスタ)の割り込みマスクを14に設定します。 <b>※ユーザプログラム内で割り込みマスクは15に設定しないで下さい。</b>
REALiモニタの タスク情報数	100タスク
最適化オプション	基本的には最適化オプションを外してコンパイルして下さい。 ステップ実行時などにプログラム実行位置とCソースが一致しない場合が発生します。

## 2. 3 サンプルプログラム

### 2. 3. 1 サンプルプログラムの構成

以下に、コンパイラを `gcc`、ターゲットを `AP-SH2F-0A (SH7045F)` 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。(他のCPUボードのサンプルも基本的な構成は同じです)

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

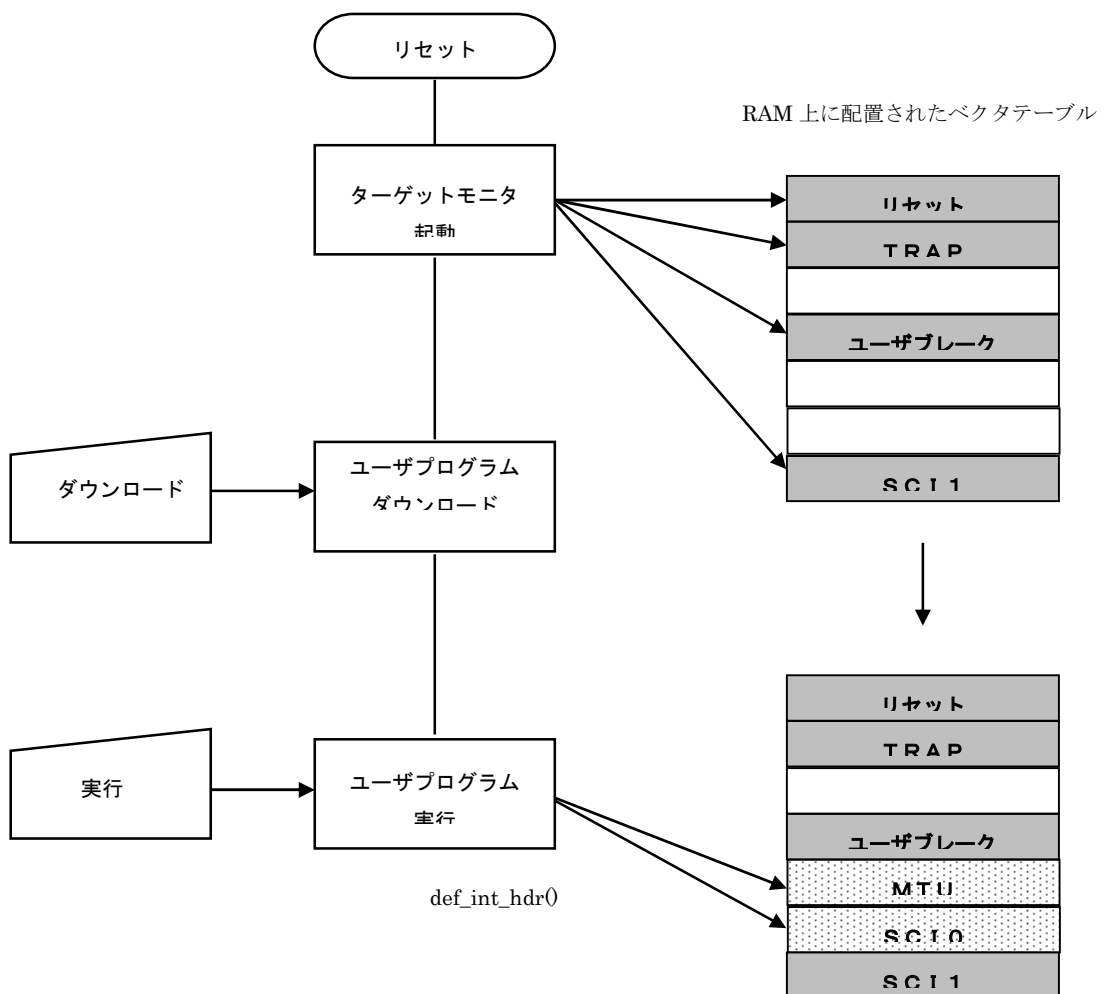
\* REAL i 用のサンプルプログラムは、インストールされませんのでCD-ROMから必要なプログラムをコピーしてご使用ください。(¥VM¥REAL i AP-SH2F-0Aのみ)

## 2. 4 ベクタテーブル

### 2. 4. 1 ベクタテーブルの配置と構造

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AP-SH2F-0Aでの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 : SC11をモニタ用に使用  
 ユーザプログラム : MTU、SC10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。

## 2. 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。

## &lt;ベクタ登録関数&gt;

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

<pre> #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&amp;= ~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 &amp;= ~0x0080;    /*CMF クリア */     省略 } /*****/ /*      MTU0 初期化      */ /*****/ void mtu0_init(void) {     def_int_hdr(88, (void*)int_mtu0); /*割り込みハンドラの設定 */     INTC. IPRD. WORD  = 0x7000;      /*MTU0 InterruptLevel = 7 */      MTU0. TCR. BYTE = 0x23;          /*T G R A のコンペアマッチ */     MTU0. TSR. BYTE = 0x00;          /*ステータスクリア */      省略 } </pre>		<div style="border: 1px solid black; padding: 5px; width: fit-content;">           ベクタ番号 : 144 に int_cmi0()  <small>を登録</small> </div>
<pre> } </pre>		<div style="border: 1px solid black; padding: 5px; width: fit-content;">           ベクタ番号 : 88 に int_mtu0()  <small>を登録</small> </div>

## 2. 4. 3 VisualMonitorが使用する割り込み (SH1, SH2)

VisualMonitorのモニタプログラムは以下の割り込みを使用しますので、ユーザプログラムで設定しないでください。  
 S C Iは指定した1つのチャンネルしか使用されませんので VisualMonitor で使用しないチャンネルはユーザプログラムで  
 使用可能です。(下表以外のベクタ番号は各CPUのデータシートをご覧ください)

## &lt;SH2 7045 の場合&gt;

例外要因		ベクタ番号	ベクタテーブルアドレスオフセット
パワーオンリセット	P C	0	H'00000000-H'00000003
	S P	1	H'00000004-H'00000007
マニュアルリセット	P C	2	H'00000008-H'0000000B
	S P	3	H'0000000C-H'0000000F
ユーザブレーク		1 2	H'00000030-H'00000033
トラップ命令 (20H)		3 2	H'00000080-H'00000083
トラップ命令 (21H)		3 3	H'00000084-H'00000087
S C I 0	E R I 0	1 2 8	H'00000200-H'00000203
	R X I 0	1 2 9	H'00000204-H'00000207
	T X I 0	1 3 0	H'00000208-H'0000020B
S C I 1	E R I 1	1 3 2	H'00000210-H'00000213
	R X I 1	1 3 3	H'00000214-H'00000217
	T X I 1	1 3 4	H'00000218-H'0000021B

## &lt;SH2 7046 の場合&gt;

例外要因		ベクタ番号	ベクタテーブルアドレスオフセット
パワーオンリセット	P C	0	H'00000000-H'00000003
	S P	1	H'00000004-H'00000007
マニュアルリセット	P C	2	H'00000008-H'0000000B
	S P	3	H'0000000C-H'0000000F
ユーザブレーク		1 2	H'00000030-H'00000033
トラップ命令 (20H)		3 2	H'00000080-H'00000083
トラップ命令 (21H)		3 3	H'00000084-H'00000087
S C I 2	E R I 0	1 6 8	H'000002A0-H'000002A3
	R X I 0	1 6 9	H'000002A4-H'000002A7
	T X I 0	1 7 0	H'000002A8-H'000002AB
S C I 3	E R I 1	1 7 2	H'000002B0-H'000002B3
	R X I 1	1 7 3	H'000002B4-H'000002B7
	T X I 1	1 7 4	H'000002B8-H'000002BB

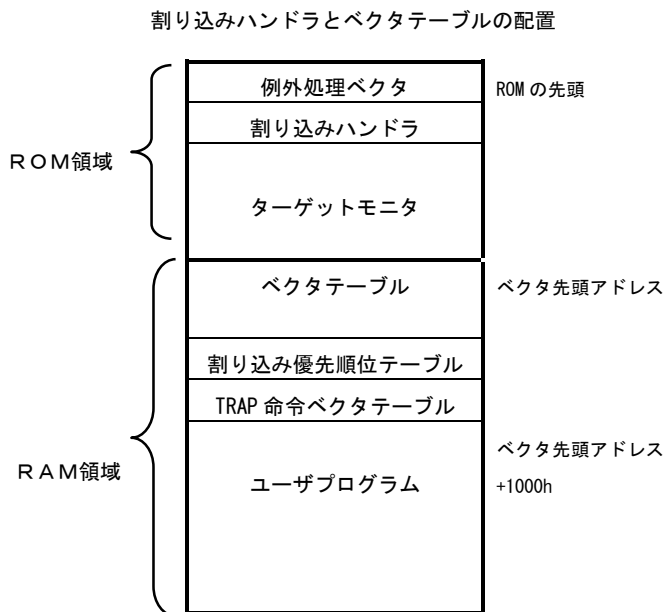
K I T 限定版ではモニタプログラムで使用する S C I のチャンネルは S H 7 0 4 5 F 版は S C I 1 に、S H 7 0 4 6 F 版は  
 S C I 3 になります。

その他の S C I はユーザプログラムで使用できます。

## 2. 4. 4 SH3の割り込みについて

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

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



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

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

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

### 割り込み優先順位の設定について

SH3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 SH7709A/S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	
	ユーザブレイクポイントトラップ	15	H'0000003C-H'0000003F	
	割り込み要求	NMI		14 H'00000038-H'0000003B
		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	
IrDA		DEI3	67 H'0000010C-H'0000010F	
		ERI1	68 H'00000110-H'00000113	
		RXI1	69 H'00000114-H'00000117	
		BRI1	70 H'00000118-H'0000011B	
SCIF		TXI1	71 H'0000011C-H'0000011F	
		ERI2	72 H'00000120-H'00000123	
		RXI2	73 H'00000124-H'00000127	
		BRI2	74 H'00000128-H'0000012B	
ADC		TXI2	75 H'0000012C-H'0000012F	
		ADI	76 H'00000130-H'0000003F	
TMU0		TUNI0	32 H'00000080-H'0000003F	
TMU1		TUNI1	33 H'00000084-H'0000003F	
		TUNI2	34 H'00000088-H'0000003F	
TMU2		TICPI2	35 H'0000008C-H'0000008F	
		ATI	36 H'00000090-H'00000093	
RTC		PRI	37 H'00000094-H'00000097	
		CUI	38 H'00000098-H'0000009B	
		ERI	39 H'0000009C-H'0000009F	
SCI		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 SH7709A/S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	T L Bミス/無効 (読み出し)	2	H'00000008-H'0000000C	
	T L B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	
	T L B保護例外 (読み出し)	5	H'00000014-H'00000017	
	T L B保護例外 (書き込み)	6	H'00000018-H'0000001B	
	アドレスエラー (読み出し)	7	H'0000001C-H'0000001F	
	アドレスエラー (書き込み)	8	H'00000020-H'00000023	
	無条件トラップ *1	11	H'0000002C-H'0000002F	
	一般不当命令例外	12	H'00000030-H'00000033	
	スロット不当命令例外	13	H'00000034-H'00000037	
	ユーザブレイクポイントトラップ	15	H'0000003C-H'0000003F	
	割り込み要求	N M I		14
I R L 3~0		レベル15	16	H'00000040-H'00000043
		レベル14	17	H'00000044-H'00000047
		レベル13	18	H'00000048-H'0000004B
		レベル12	19	H'0000004C-H'0000004F
		レベル11	20	H'00000050-H'00000053
		レベル10	21	H'00000054-H'00000057
		レベル9	22	H'00000058-H'0000005B
		レベル8	23	H'0000005C-H'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	
I R Q		I R Q 4	52	H'000000D0-H'000000D3
		I R Q 5	53	H'000000D4-H'000000D7
P I N T		P I N T 0~7	56	H'000000E0-H'000000E3
		P I N T 8~15	57	H'000000E4-H'000000E7
D M A C		I R Qモードと同じ (前ページ参照)		
I r D A				
S C I F				
A D C				
T M U 0				
T M U 1				
T M U 2				
R T C				
S C I				
W D T				
R E F				

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

## 2. 4. 5 ベクタテーブルの登録方法 (SH3)

実際にユーザプログラム上でベクタを登録する方法を説明します。

ソースファイル : GMACHINE.H GMACHINE.C

### <ベクタテーブルアドレス登録関数>

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

### <ベクタ登録関数>

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

### <無条件 TRAP ベクタ登録関数>

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

## ユーザプログラムのコーディング例

```

#include "Gmachine.h" /*ヘッダファイルの定義*/
#define VectorTableAdd 0x0C000000 /*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_init(); /*CMT タイマ初期化 */
    : (途中省略)
    :
}

/*****
/* CMT タイマイニシャル */
*****/
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; /* 5 m s タイマ SET */
    . (省略)
}

/*****
/* TRAP割り込み初期化 */
*****/
void TRAP_INIT(void)
{
    def_trap_hdr(22, (void *)trap22);
}

/*****
/* MTU0タイマ割り込み (5 m s) */
*****/
void mtu0_5ms(void)
{
    static char flag;
    unsigned char wk;
    . (省略)
}

/*****
/* TRAP 22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

## 2.4.6 VisualMonitorが使用する割り込み (SH3)

VisualMonitor のモニタプログラム以下の割り込みを使用しますので、ユーザプログラムで設定しないでください。  
 S C I は指定した1つのチャンネルしか使用されませんので VisualMonitor で使用しないチャンネルはユーザプログラムで  
 使用可能です。(下表以外のベクタ番号は前述の「SH3のベクタ番号」をご覧ください)

### <SH3 SH7709A/Sの場合>

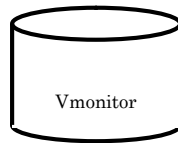
例外要因	ベクタ番号	ベクタテーブルアドレス (ベクタ先頭アドレス)
ユーザブレークポイントトラップ	15	H'0000003C-H'0000003F
無条件トラップ (20H、21H)	11	H'0000002C-H'0000002F
S C I	ER I	H'0000009C-H'0000009F
	R X I	H'00000100-H'00000103
	T X I	H'00000104-H'00000107
I r D A	ER I 1	H'00000110-H'00000113
	R X I 1	H'00000114-H'00000117
	T X I 1	H'0000011C-H'0000011F
S C I F	ER I 2	H'00000120-H'00000123
	R X I 2	H'00000124-H'00000127
	T X I 2	H'0000012C-H'0000012F

K I T版では、モニタプログラムはS C I Fを使用しています。S C I、I r D Aはユーザプログラムで使用できます。

## 第3章. デバッグ時の使用ファイル

### 3. 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 (デバッグ本体) と  
関連ファイルが格納されている。



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。

### 3. 2 デバッグ用ファイル

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

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

### 3. 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。

#### 3. 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を変更して下さい。

<REALi 使用時のコンフィグレーションウィンドウ>



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

REALi モニタウィンドウの有効/無効設定

### 3. 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。

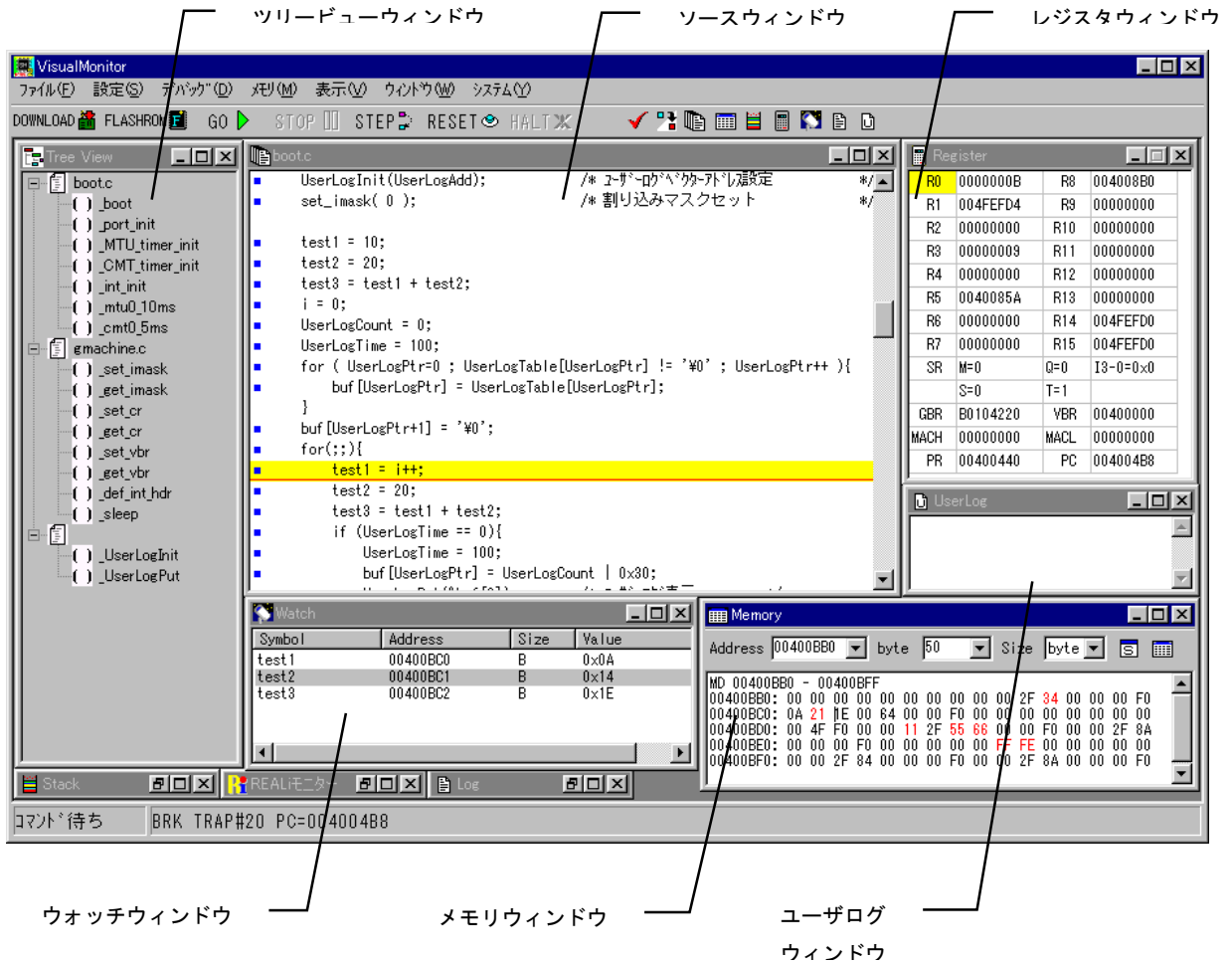
<コンフィグレーションファイルのフォーマット>

BREAK = UBC	←	ソース上でのブレイク方法指定 (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4章. メニューとウィンドウの操作

### 4. 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。

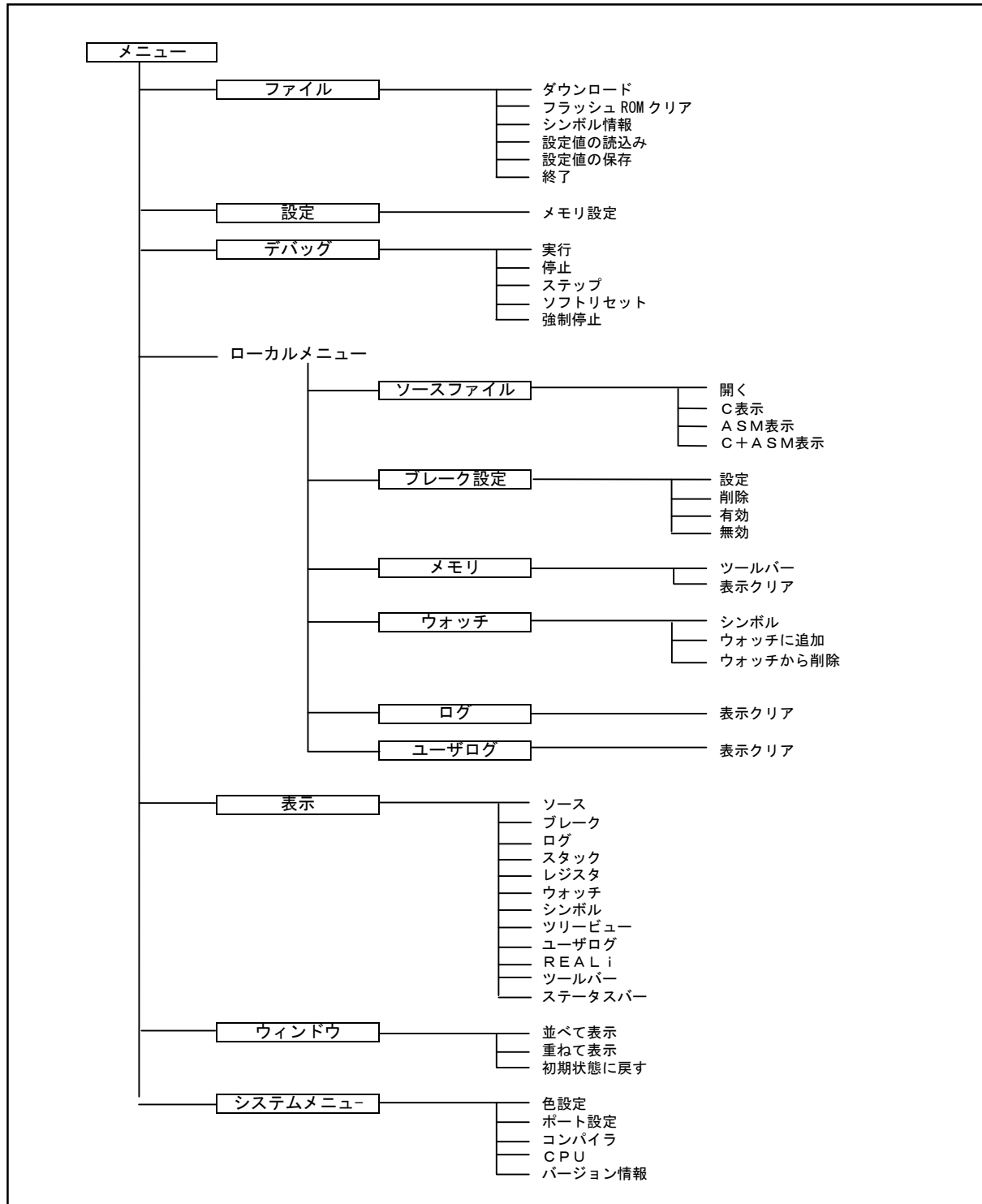




## 4.2 メニュー

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 4.2 メニューコマンド一覧



#### 4. 2. 1 ファイルメニュー (Alt+F)

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

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

#### 4. 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。

#### 4. 2. 3 デバッグメニュー (Alt+D)

デバッグメニューには、ユーザプログラムの実行/停止項目が用意されています。

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

#### 4. 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。

#### 4. 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。

#### 4. 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。

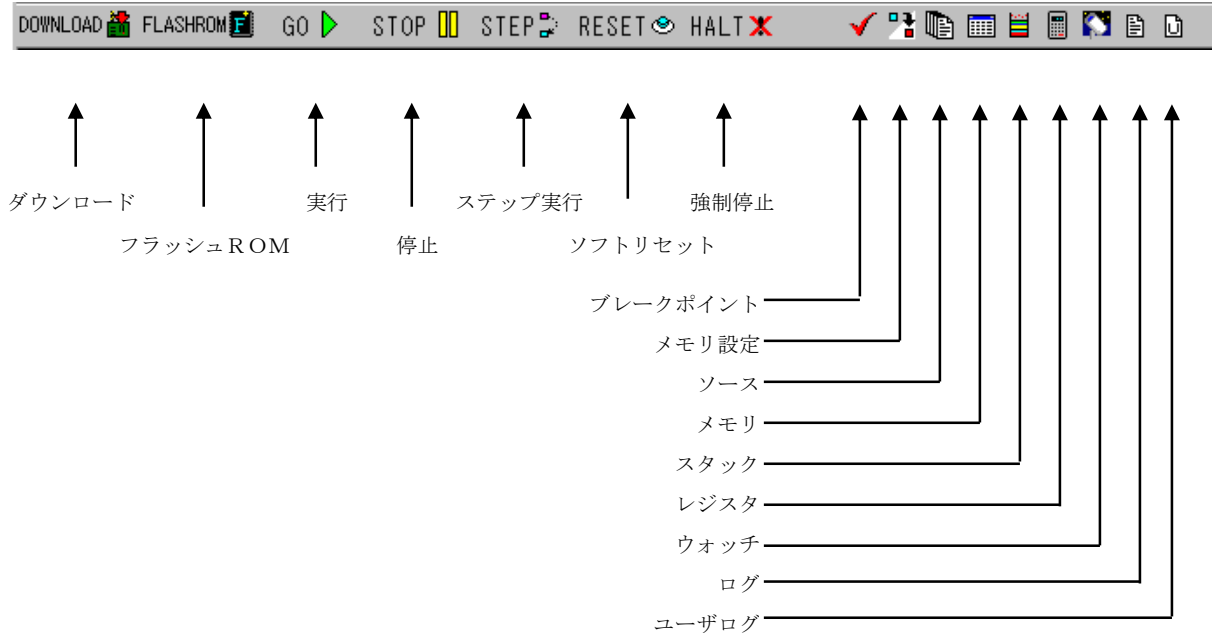
#### 4. 2. 7 システムメニュー (Alt+F)

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。

### 4. 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。



### 4. 4 ステータスバー

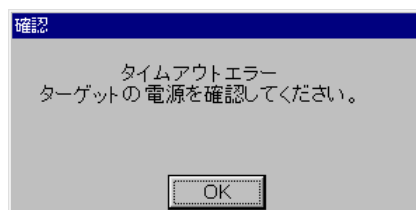
ステータスバーには、各コマンドの実行状態が表示されます。

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。



### 4. 5 確認ダイアログ

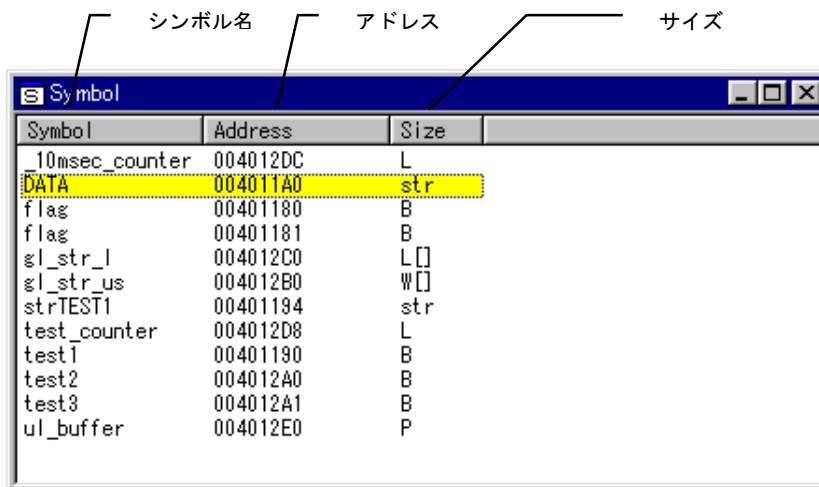
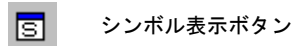
動作確認メッセージやエラーメッセージなどを表示するダイアログです。



## 4. 6 シンボル入力

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

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



シンボル情報ダイアログ

### <表示>

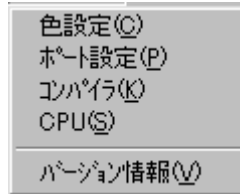
項目	表示内容
Symbol	シンボル名を表示します。 グローバルシンボルのみ表示されます。
Address	シンボルのアドレスを表示します。
size	シンボルのサイズを表示します。

### <操作>

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

## 4. 7 システム設定

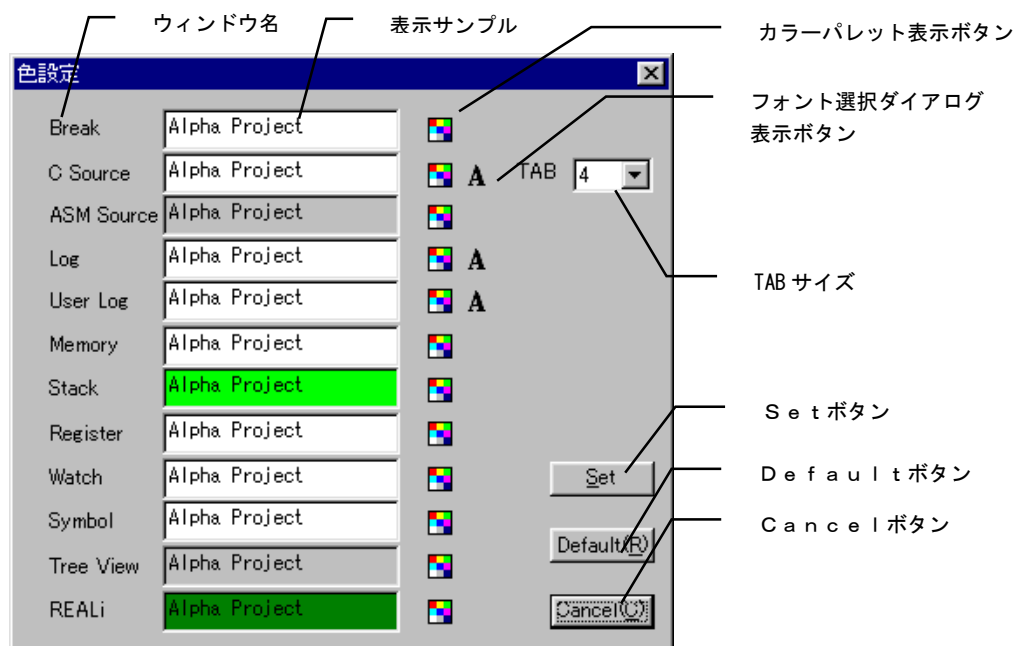
デバッグをおこなうためには、基本的な環境を設定する必要があります。  
それらの設定はシステムメニューからおこないます。



システムメニュー

### 4. 7. 1 色設定

色設定では各ウィンドウの背景色やフォント等を指定します。



色設定ダイアログ

#### <表示>

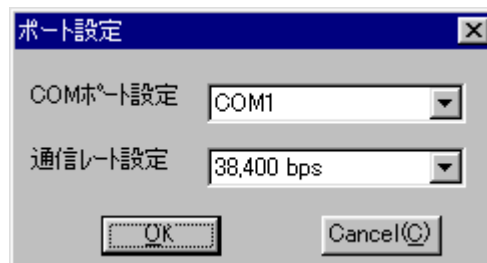
コントロール	機能
ウィンドウ名	対応する各ウィンドウ名を表示します。
表示サンプル	選択した設定でのサンプルが表示されます。

## &lt;設定項目&gt;

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

## 4. 7. 2 ポート設定

ホストコンピュータの使用するCOMポートの設定をおこないます。



ポート設定ダイアログ

## &lt;設定項目&gt;

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

## 4. 7. 3 コンパイラ設定

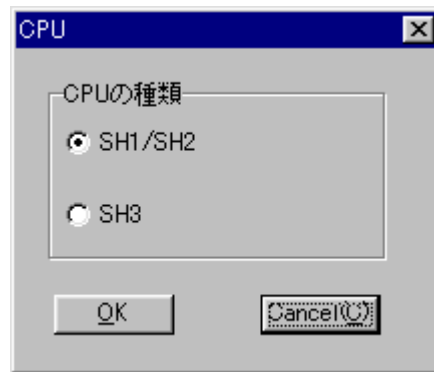
ユーザプログラムのコンパイラを設定します。



コンパイラ設定ダイアログ

#### 4. 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

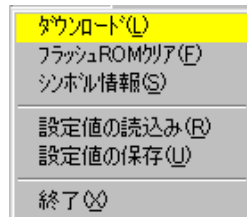


## 4. 8 プログラムのダウンロード

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。

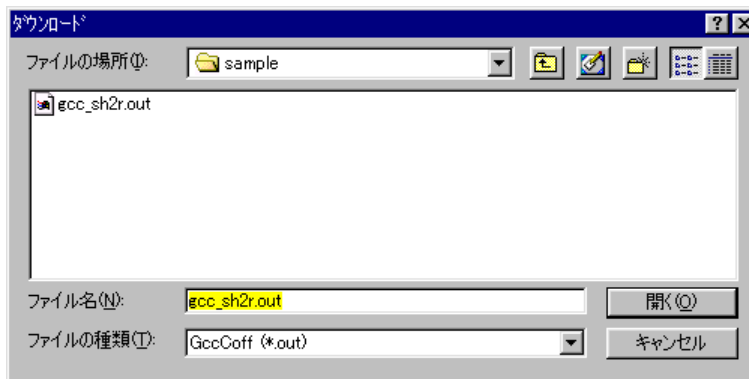


ファイルメニュー



ツールバー

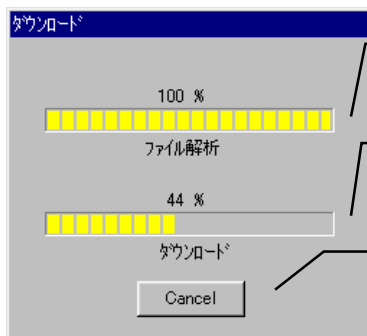
- ② ファイル選択ダイアログが表示されますので、デバッグするプログラムファイルを選択します。



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。

日立Cは .ABS、gccは .out および .x の拡張子がデフォルトになっていますが、名称は特に関係ありません。  
コンパイル時には必ずデバッグオプションを付けてください。

- ② ダウンロードが開始されると状況が表示されます。



解析状況の表示

ダウンロードの  
.....

キャンセルボタン

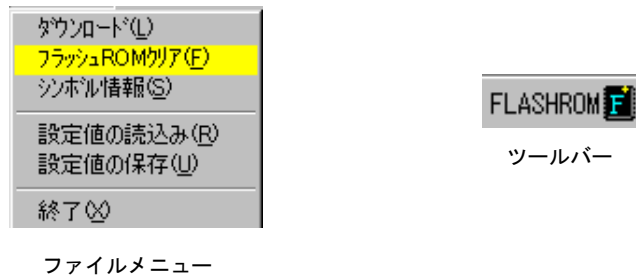
VisualMonitor は選択されたファイルのデバック情報を解析した後にダウンロードを開始します。ダウンロード時間は、デバック環境によって変化しますが、100K以上のプログラムの場合には1分以上かかる場合もあります。

## 4.9 フラッシュROMのクリア

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

### <操作方法>

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



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



注) F-ZTATを使用する場合、通常、EB0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。  
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もしくは日立製書き込みツール等) を使用して書き込んでおきます。  
書き込んだ後、MODE端子及びFWP端子の設定をユーザプログラムモードに設定してデバッグしてください。

注意) KIT限定版のSH7045F、SH7709A、SH7709S版では、フラッシュROMにユーザプログラムはダウンロードできません。

## 4. 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。

### 4. 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。

### 4. 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。

### 4. 10. 3 ステップ実行



ステップボタン

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

#### <Cソース表示でのステップ実行>

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

#### <アセンブラ表示でのステップ実行>

- ・ アセンブラの各ステートメントを実行します。

#### <混在表示でのステップ実行>

- ・ アセンブラの各ステートメントを実行します。

### ステップ実行使用時の注意

ステップ実行は、U B Cを使用する為、幾つかの制限があります。

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

※ U B Cの詳細は各CPUのハードウェアマニュアルを参照して下さい。

#### 4. 10. 4 RESET



リセットボタン

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

#### 4. 10. 5. 強制停止



強制停止ボタン

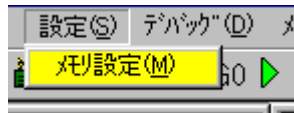
実行中のプログラムを強制停止します。実行中やステップ実行の際にプログラムの停止に時間がかかる場合や、停止しなかった場合に強制的にプログラムを停止させます。

## 4. 1 1 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。

## 4. 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。

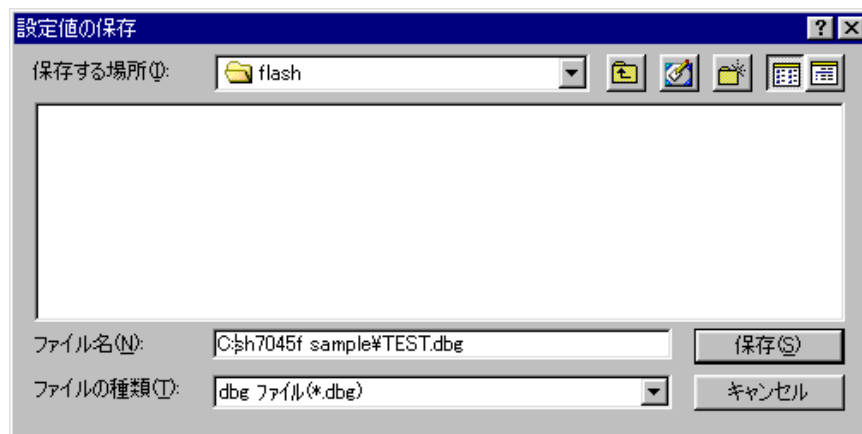
### 4. 12. 1 設定値の保存

<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

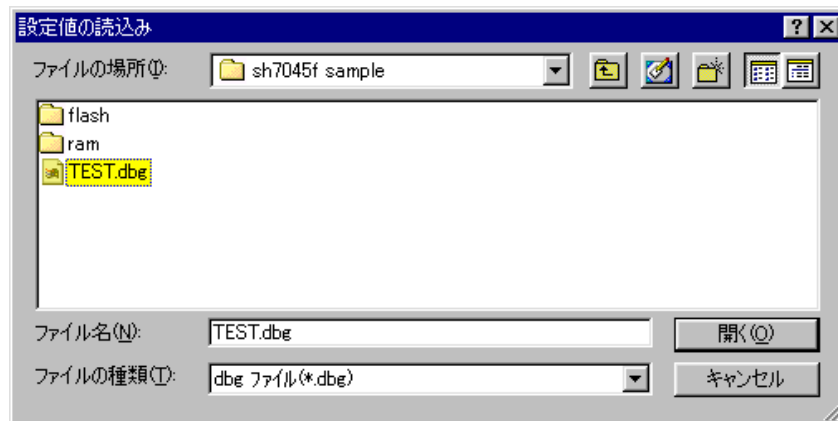
#### 4. 1 2. 2 設定値の読み込み

<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。

## 4. 13 ツリービューウィンドウ

### 4. 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。

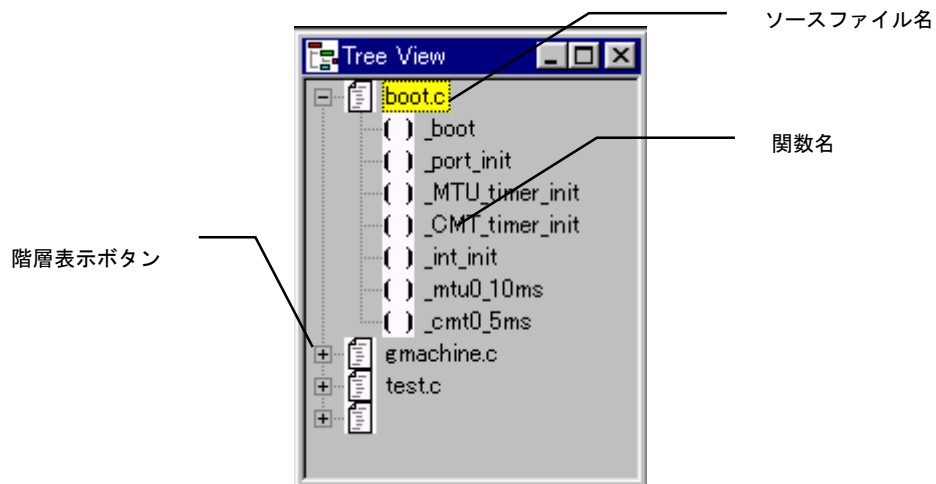


図 4. 13 ツリービューウィンドウ

#### <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>

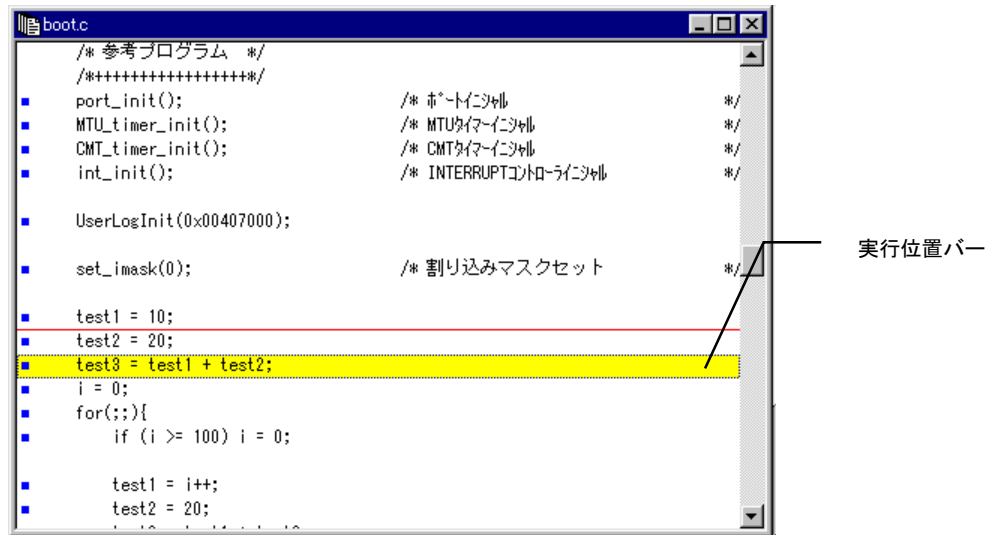
コントロール	機能
階層表示ボタン	クリックで下の階層（関数）を展開表示します。
関数名	ダブルクリックで該当する関数をソースウィンドウに表示します。



## 4. 14 ソースウィンドウ

### 4. 14. 1 ソースウィンドウの機能

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



#### <表示>

デバッグ対象となるプログラムのソースファイルを表示します。

表示は、Cソース表示、アセンブラ表示、C+A S M表示を選択できます。

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

#### <ローカルメニュー操作>

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

#### <マウス操作>

ポインタ位置	機能
ソース行	右クリックでブレークポイント設定のプルダウンメニューが表示され、ブレークポイントの設定/削除/有効/無効が操作できます。 詳細は「 <a href="#">ブレークポイントの設定方法</a> 」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「 <a href="#">ダイレクト変数表示機能</a> 」を参照してください。

#### 4. 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。

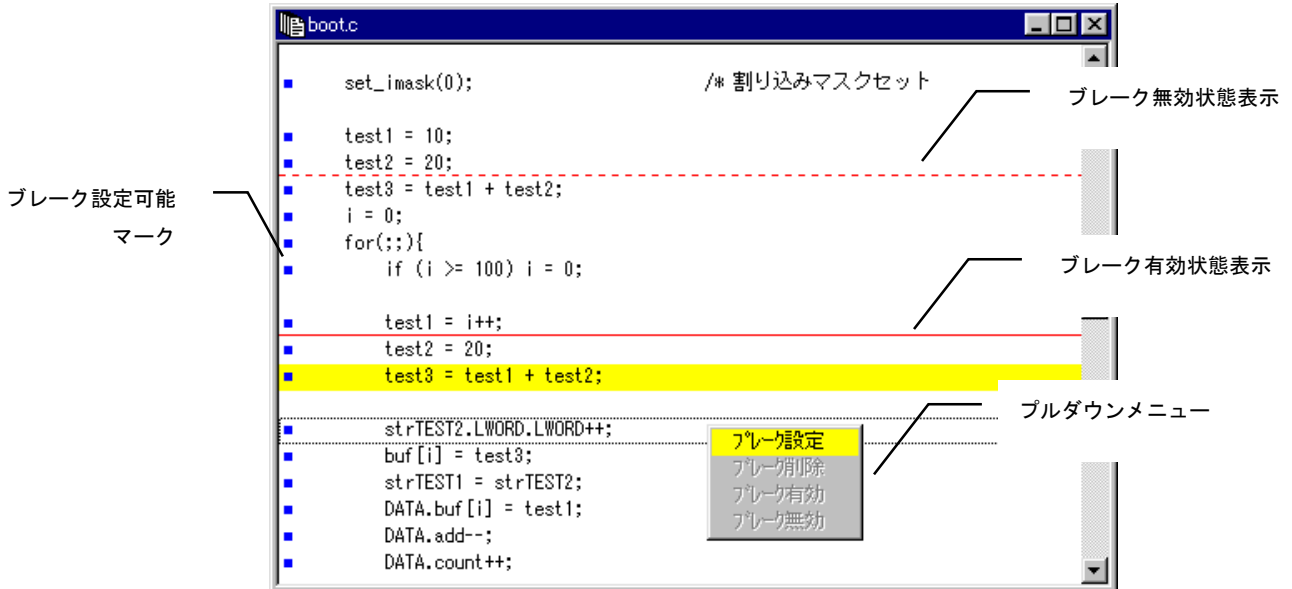


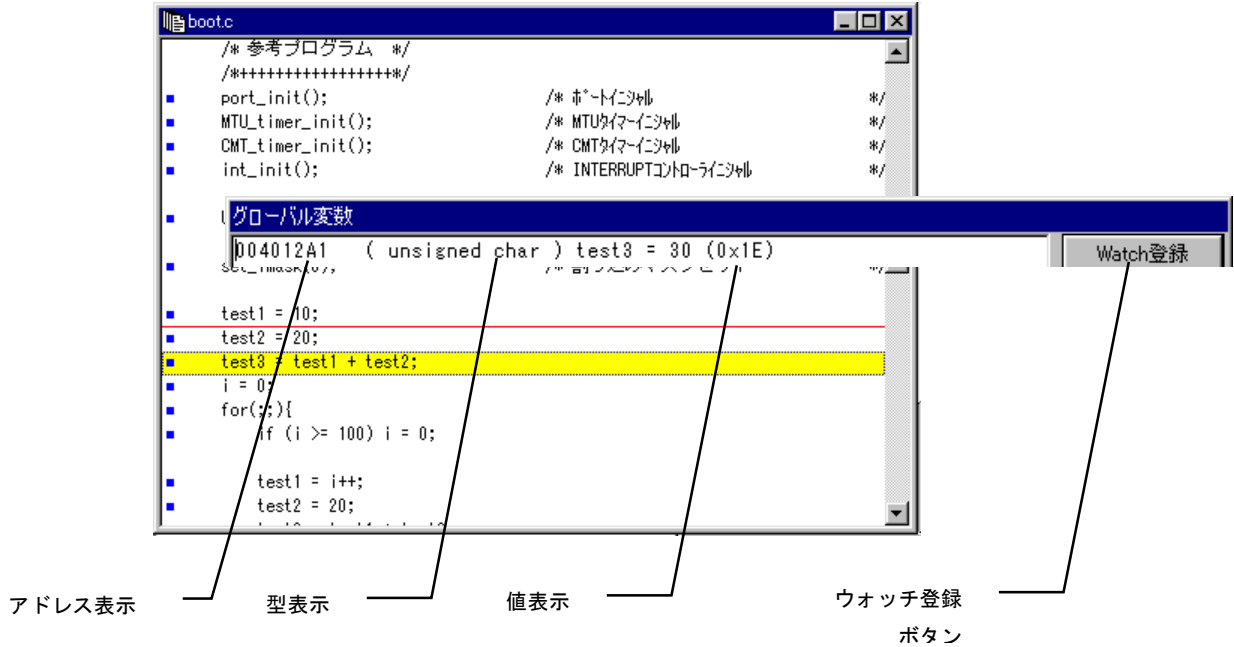
図 4. 14. 2 ソースウィンドウのブレーク設定

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

### 4. 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。

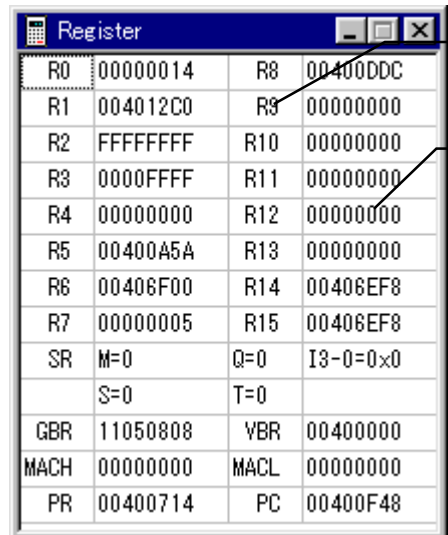
#### <表示>

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。

## 4. 15 レジスタウィンドウ

### 4. 15. 1 レジスタウィンドウの機能

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



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

レジスタ名

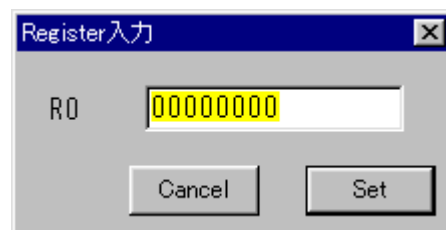
レジスタ値

#### <表示>

CPU内部のレジスタ値を16進表示します。常に最新の状態を自動的に表示します。  
SH1/2とSH3とはレジスタ構成が異なるため、表示が異なります。

#### <マウス操作>

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



レジスタ値を16進で入力します。

レジスタ値設定ウィンドウ

## 4. 16 ブレークポイントウィンドウ

### 4. 16. 1 ブレークポイントウィンドウの機能

現在のブレークポイントの設定を一覧表示します。

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

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



#### <表示>

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

#### <ローカルメニュー操作>

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

#### <マウス操作>

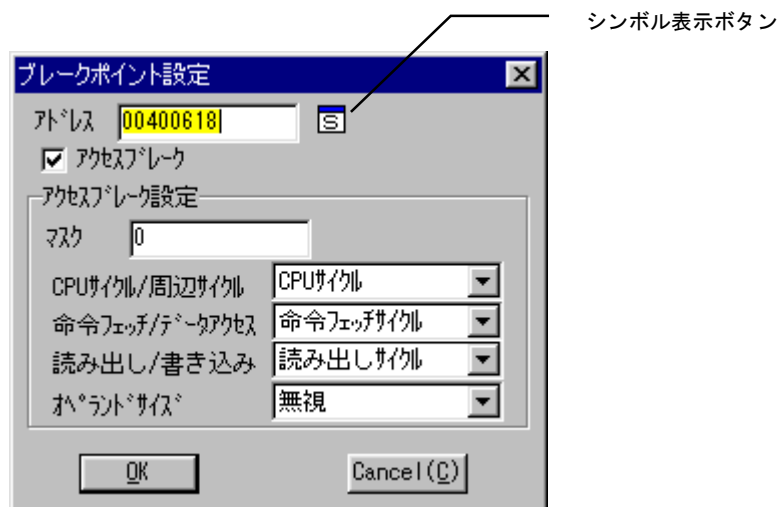
ポインタ位置	機能
表示行	既に表示されている行をダブルクリックするとブレークポイント設定ダイアログが表示されブレーク条件を変更できます。 右クリックするとプルダウンメニューが表示され設定/削除/有効/無効とソースが選択できます。設定/削除/有効/無効はローカルメニューと同じ機能です。 ソースはソースウィンドウに該当行を表示します。
空白行	空白の行をダブルクリックするとブレークポイント設定ダイアログが表示されブレークポイントを追加できます。

## &lt;キーボード操作&gt;

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

## 4. 16. 2 詳細なブレイク条件の設定

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



## &lt;シンボル表示ボタン&gt;

前述の「シンボル表示入力」を参照してください。

## &lt;設定項目&gt;

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

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

#### 4. 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。

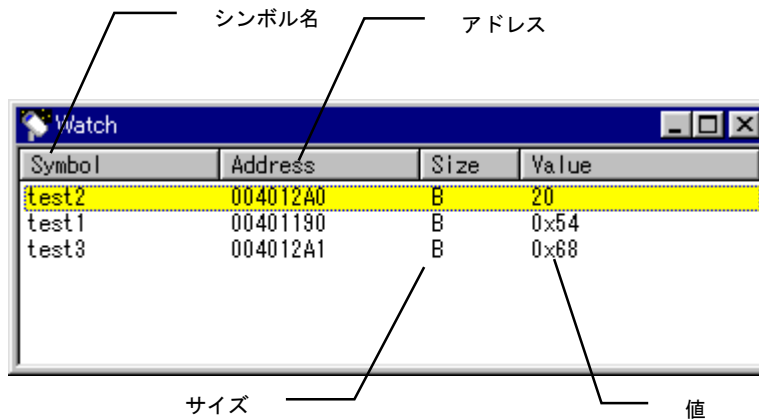
- ・BT, BF命令（非遅延分岐命令）の次の命令フェッチにブレークを指定しても停止しません。
- ・データアクセスをブレーク条件に設定した場合、ブレークするデータアクセスが発生した付近で停止する為停止位置は特定できません。
- ・命令フェッチをブレーク条件に設定した場合、指定位置の手前で停止する場合があります。これはSHがパイプライン方式を採用している為です。
- ・SH1のアセンブラレベルでのデバッグ時において、現在停止している命令の2命令後にアクセスブレークを設定して実行をかけた場合、タイマーなどの割り込み要因が保持されているとアクセスブレークの割り込みと同時に発生します。  
ここで実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。

**※ 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。**

## 4. 17 ウォッチウィンドウ

### 4. 17. 1 ウォッチウィンドウの機能

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



#### <表示>

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

#### <ローカルメニュー操作>

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

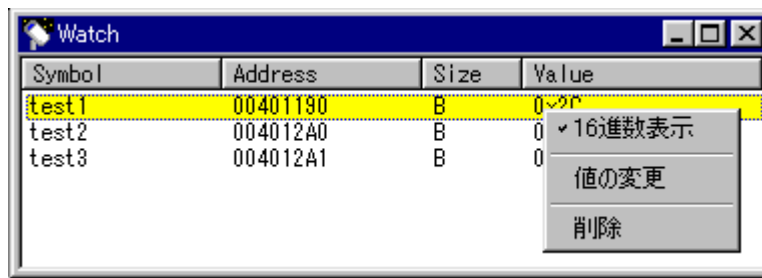
#### <マウス操作>

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

#### <キーボード操作>

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





プルダウンメニュー（右クリックで表示）

## &lt;プルダウンメニュー&gt;

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

## 4. 17. 2 ウォッチ変数の値変更

登録されたウォッチ変数の内容を変更することができます。

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



変数値変更ダイアログ

## &lt;変更ダイアログ&gt;

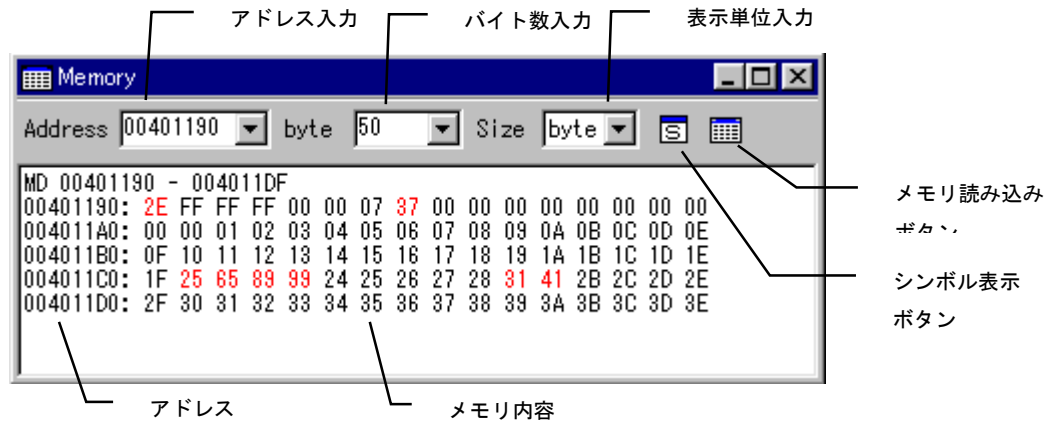
項目	表示内容
Symbol	シンボル名を表示します。
Address	シンボルのアドレスを表示します。
Size	サイズを表示します。（B：1バイト、W：2バイト：L：4バイト）
Value	値を入力します。 16進数で入力する場合は0xから入力し、それ以外は10進数とみなされます。

## 4. 18 メモリウィンドウ

### 4. 18. 1 メモリウィンドウの機能

メモリウィンドウは、指定された範囲のメモリ内容を表示します。

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



#### <表示>

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

#### <入力項目>

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

#### <ローカルメニュー操作>

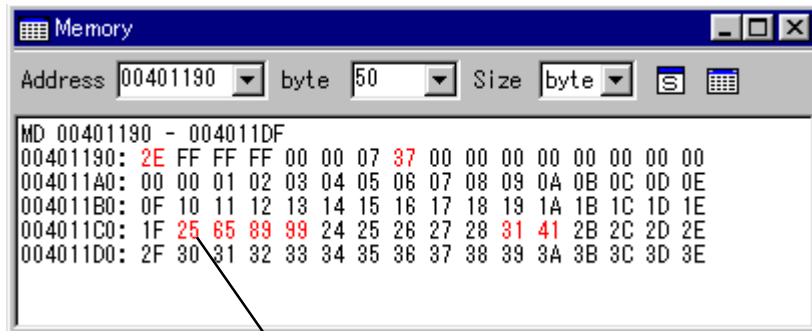
メニュー項目	ショートカット	機能
ツールバー (T)	—	ツールバーの表示/非表示を行う。 チェックマークが付いている場合にはツールバー表示が有効である。
表示クリア (C)	—	メモリ内容の表示をクリアします。

#### <マウス操作>

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

## 4. 18. 2 メモリイメージの編集

メモリウィンドウでは、メモリイメージを直接書き換えて編集できます。



変更したい箇所をクリックし、カーソルを位置付けて数値を入力します。

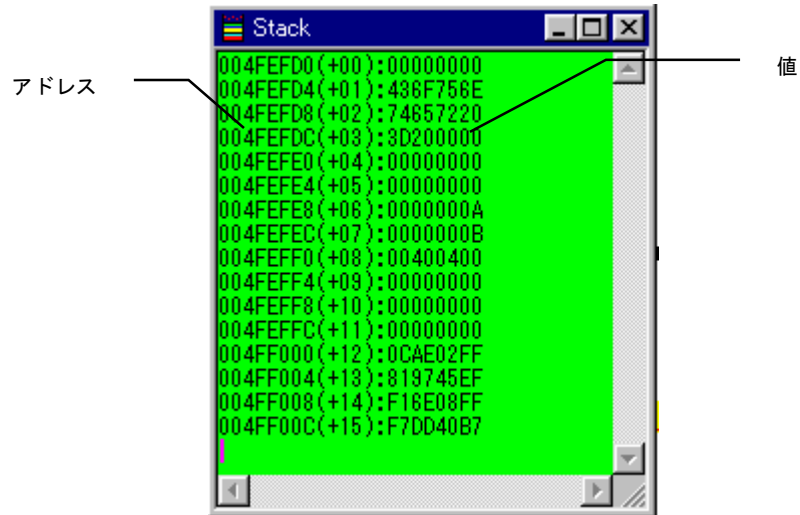
## &lt;操作手順&gt;

- 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。

## 4. 19 スタックウィンドウ

### 4. 19. 1 スタックウィンドウの機能

スタックウィンドウは、スタックの内容を常に最新の情報で表示します。



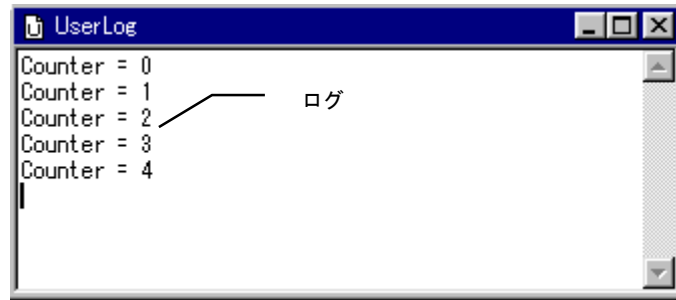
#### <表示>

項目	表示内容
アドレス	スタックのアドレスを表示します。 +0が現在のスタックポインタで、相対アドレスで+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。

## 4. 20 ユーザログウィンドウ

### 4. 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。

### 4. 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いません 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いません。	

## 4. 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。

SH2用	SH2UL. LIB	(日立C用)
	SH2ULD. LIB	(日立C Ver6用)
	SH2UL. A	(gcc用)
SH3用	SH3UL. LIB	(日立C ビッグエンディアン用)
	SH3ULD. LIB	(日立C Ver6 ビッグエンディアン用)
	SH3UL. A	(gcc ビッグエンディアン用)
	SH3LUL. LIB	(日立C リトルエンディアン用)
	SH3LULD. LIB	(日立C Ver6 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。

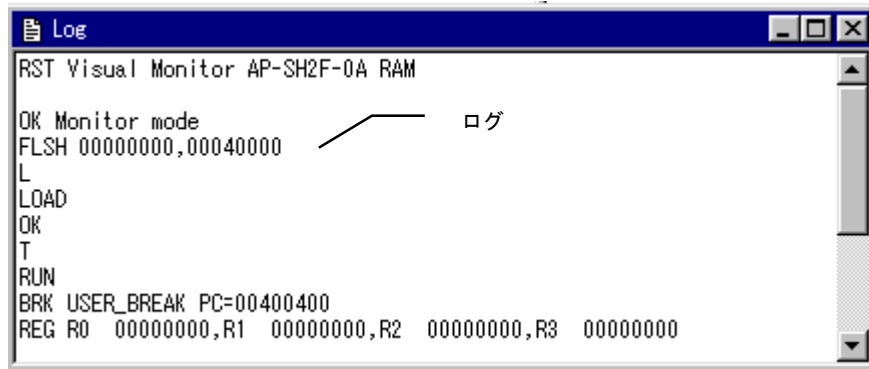
## 4. 21 ログウィンドウ

### 4. 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。

## 4.22 REALi モニタウィンドウ

### 4.22.1 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SH3には対応していません。

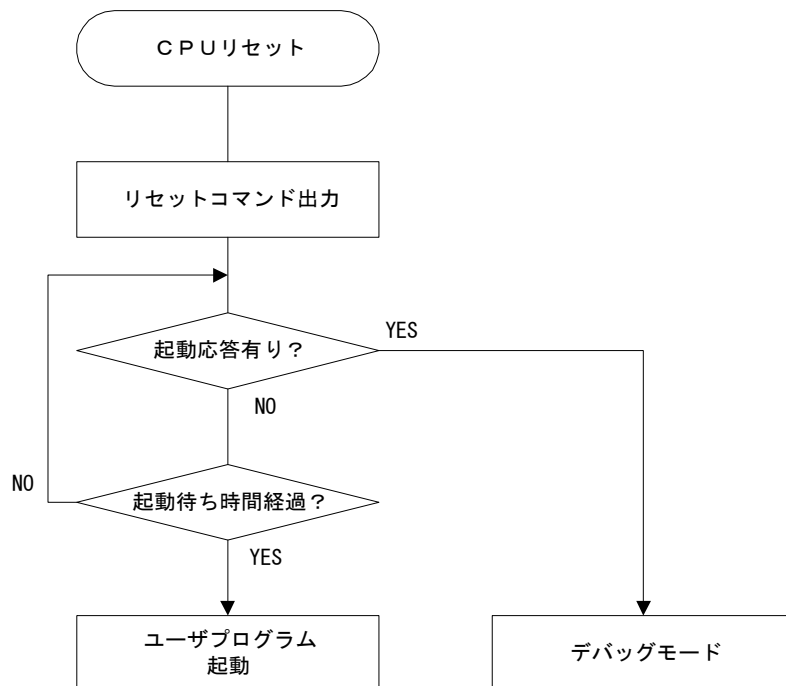


## 第5章. その他の機能

### 5. 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を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6章. ハードウェア

### 6. 1 ハードウェアの注意点

#### 6. 1. 1 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。

## 第7章. 製品サポートと使用上の注意

### 7. 1 製品サポートのご案内

#### 7.1.1 弊社ホームページのご利用について

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

弊社ホームページアドレス <http://www.apnet.co.jp>

#### 7.1.2 製品サポートの方法

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

##### 製品サポート窓口

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

#### 7.1.3 製品サポートの範囲

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

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

### 7. 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ください。
- 本製品に組み込まれたプログラム及び添付アプリケーションのリバースエンジニアリング及び本製品以外でのご使用は堅くお断りします。
- 万が一、本製品を使用して事故または損失が発生した場合、弊社では一切その責を負いませんのでご了承ください。

## ご注意

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

**Alpha Project Co.,LTD**