

お客様各位

---

## カタログ等資料中の旧社名の扱いについて

---

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願ひ申し上げます。

ルネサスエレクトロニクス ホームページ (<http://www.renesas.com>)

2010年4月1日

ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社 (<http://www.renesas.com>)

【問い合わせ先】 <http://japan.renesas.com/inquiry>

## ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したのですが、誤りが無いことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。  
標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット  
高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）  
特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関し、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社とその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

# SuperH RISC engine C/C++ コンパイラパッケージ

アプリケーションノート：＜統合開発環境活用ガイド＞

テスト自動化支援機能編

本ドキュメントでは、High-performance Embedded Workshop のマクロ生成支援機能およびテスト支援機能について紹介します。

## 目次

1. マクロ生成支援機能.....	2
1.1 マクロの記録／実行／編集／削除.....	2
1.1.1 マクロの記録.....	3
1.1.2 マクロの編集.....	4
1.1.3 マクロの実行.....	4
1.1.4 既存のマクロファイルのインポート.....	4
1.1.5 マクロ・マクロファイルの削除.....	5
1.2 マクロ生成支援機能のサポート範囲.....	5
1.2.1 ルネサス統合開発環境共通で記録可能な操作.....	5
1.2.2 デバッグングプラットフォーム依存で記録可能なコマンド.....	9
2. テスト支援機能.....	12
2.1 テストスイート.....	13
2.1.1 テストスイートの作成.....	14
2.1.2 テストスイートの開き方／閉じ方.....	14
2.1.3 テストスイートの編集.....	15
2.2 テストイメージファイル.....	17
2.2.1 テストイメージファイルの編集.....	17
2.2.2 テストイメージファイルの保存.....	17
2.3 テストの実行.....	18
2.4 テスト結果の確認.....	19
2.4.1 テスト結果ブラウザの表示内容.....	19
2.5 テスト結果の比較.....	20
3. チュートリアル.....	21
3.1 サンプルプロジェクトについて.....	21
3.2 テストの準備.....	23
3.2.1 マクロファイルの生成.....	23
3.2.2 マクロの記録.....	24
3.2.3 テストスイートの作成.....	26
3.2.4 テストイメージファイルの作成.....	30
3.3 回帰テスト.....	33
3.3.1 プログラム変更後の動作確認.....	33
3.3.2 正常動作の確認.....	34
ホームページとサポート窓口<website and support,ws>.....	35

### 1. マクロ生成支援機能

マクロ生成支援機能は、High-performance Embedded Workshop(以下、ルネサス統合開発環境)システムのアプリケーション関連(\*1)、ビルド関連(\*2)、およびデバッグ関連(\*3)などの一連の操作をマクロとしてファイルに記録する機能です。マクロはルネサス統合開発環境のコマンドラインに相当します。

- \*1. プロジェクトの変更、セッションの変更、コンフィグレーションの変更など。
- \*2. コンパイル、ビルドなど。機能のサポートは、デバッグプラットフォームに依存します。
- \*3. モジュールのダウンロード、メモリ値の変更、レジスタ値の変更、ソフトウェアブレイクの設定/解除、プログラム実行など。

#### 1.1 マクロの記録／実行／編集／削除

マクロを保存するファイルを「マクロファイル」と呼びます。マクロファイルは拡張子が “.hdc” のテキストファイルです。マクロファイルはルネサス統合開発環境のインストールディレクトリ下の「Macro」ディレクトリに保存されます。デフォルトでは Default.hdc が使用されます。例えば、C:\Program Files\Renesas\Hew がルネサス統合開発環境のインストールディレクトであれば、

C:\Program Files\Renesas\Hew\Macro\Default.hdc がデフォルトで使用されます。

新規マクロファイルを作成するには、[ツール]メニューの[マクロの設定]で表示される[マクロの設定]ダイアログボックス(図 1-1)で[新規]ボタンをクリックしてください。[新規]ボタンをクリックすると、[マクロファイルの新規追加]ダイアログボックスが表示されます。その[マクロファイルの新規追加]ダイアログボックスに新規に作成するマクロファイル名(64文字以内、半角英数字、半角下線のみ)を入力し、[OK]ボタンをクリックすると、[使用中のマクロファイル]ドロップダウンリストに新規マクロファイル名が追加されます。

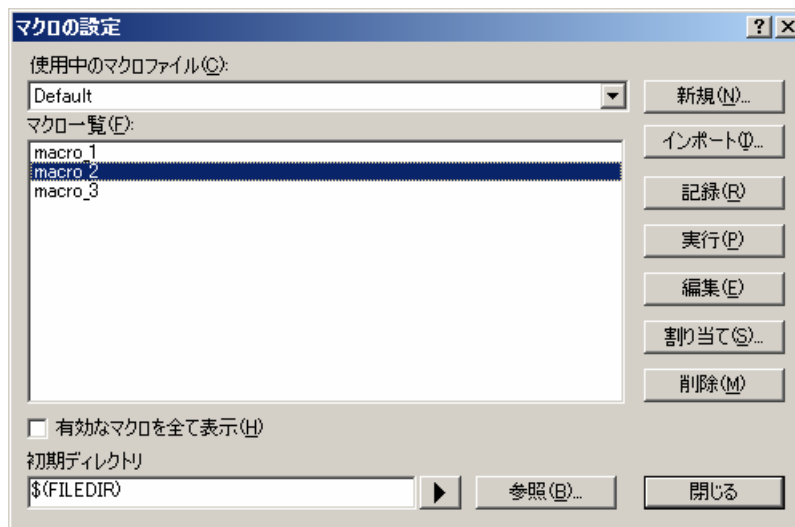


図 1-1

使用するマクロファイルを切り替えるには、[マクロの設定]ダイアログボックス(図 1-1)の[使用中のマクロファイル]ドロップダウンリストで行ってください。

[初期ディレクトリ]はマクロに相対パスが含まれる場合に、基点となるディレクトリを指定するためのものです。マクロの直接編集により相対パスを用いた場合は、本項目の設定が必要です。

## 1.1.1 マクロの記録

マクロの記録をするには、[ツール]メニューの[マクロの記録]、もしくはツールバーのマクロボタンを選択してください。

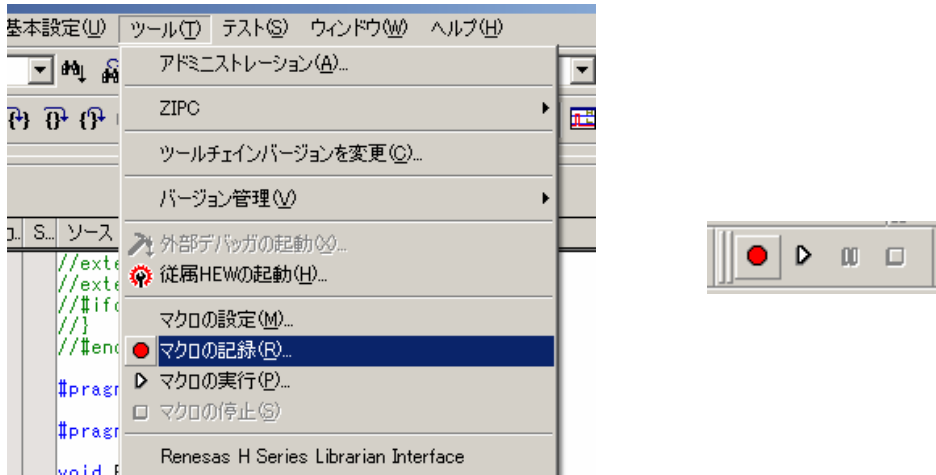



図 1-2

マクロ記録中は、マウスカーソルが  のようになります。

記録している内容はアウトプットウィンドウ([表示]メニューの[アウトプット])の[Macro]タブにリアルタイムに反映されます。マクロの記録を停止するときは、[ツール]メニューの[マクロの停止]を選択してください。マクロの記録の停止を行うと、[マクロの新規追加]ダイアログボックスが表示されます(図 1-3)。



図 1-3

新規マクロ名を入力して[OK]ボタンをクリックしてください。記録を停止するとマウスカーソルは元に戻ります。下記、(1)~(3)の操作を記録した例を図 1-4に示します。

- (1) c:\workspace\hew\_test\cmn\_src\hew\_test.c の 31 行目にブレークポイントを挿入 ([編集]メニューの[ブレークポイントの挿入/削除]を選択)
- (2) リセット後実行([デバッグ]メニューの[リセット後実行]を選択)
- (3) ステップオーバー([デバッグ]メニューの[ステップオーバー]を選択)

使用しているマクロファイルに操作した内容が指定したマクロ名で書き込まれます。

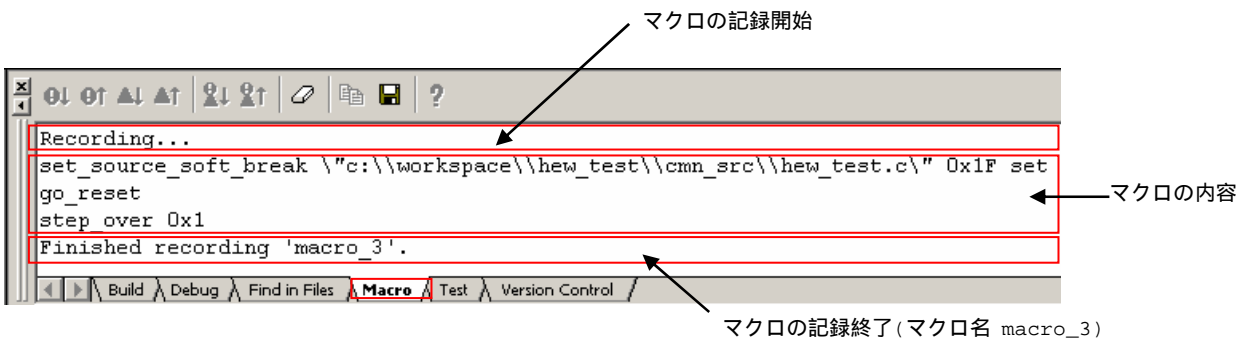


図 1-4

### 1.1.2 マクロの編集

マクロの編集をするには、[ツール]メニューの[マクロの設定]を選択し[マクロの設定]ダイアログボックス(図 1-1)を表示してください。

図 1-1のダイアログボックスで、[マクロ一覧]より編集したいマクロを選択すると、[編集]ボタンが有効になります。[編集]ボタンをクリックすると、エディタウィンドウにマクロファイルが開きます。マクロはルネサス統合開発環境のコマンドラインと同じ書式です。必要に応じて直接マクロファイルを編集してください。

例えば、下記図のようにブレークポイントを設定するソースファイル名を絶対パスから相対パスに変更したい場合は、開いたマクロファイルを直接編集してください。

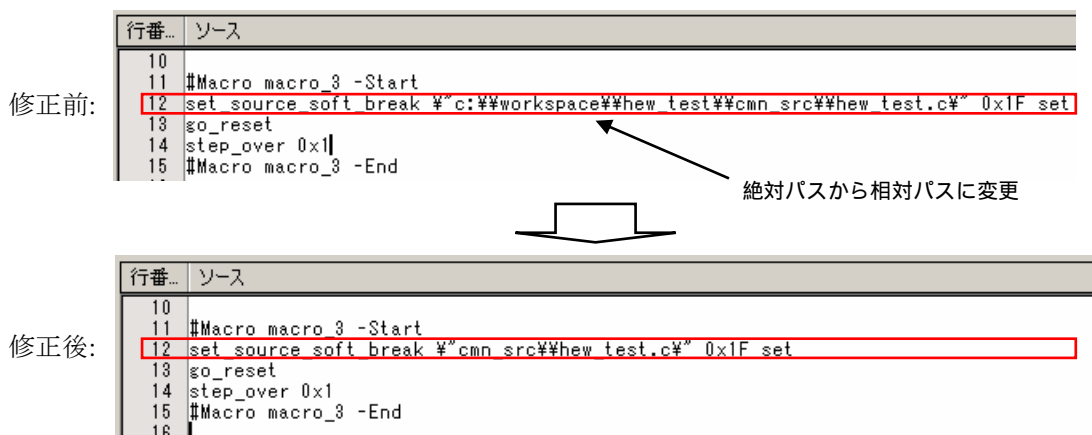


図 1-5

### 1.1.3 マクロの実行

マクロの実行は、[ツール]メニューの[マクロの実行]で行います。ツールメニューの[マクロの実行]から表示される[マクロの選択]ダイアログボックスより、実行するマクロを選択します。このダイアログボックスにはマクロの設定ダイアログボックス(図 1-1)の[使用中のマクロファイル]に含まれるマクロのみが表示されます。



図 1-6

### 1.1.4 既存のマクロファイルのインポート

他のパソコン等で作成したマクロファイルもインポートして使う事ができます。マクロファイルのインポートは[マクロの設定]ダイアログボックス(図 1-1)で[インポート]ボタンをクリックし、インポートしたいファイルを指定してください。インポートされたマクロファイルはルネサス統合開発環境のインストールディレクトリ下の「Macro」ディレクトリにコピーされます。

### 1.1.5 マクロ・マクロファイルの削除

マクロの削除は[マクロの設定]ダイアログボックス(図 1-1)で、[マクロ一覧]より削除したいマクロを選択し、[削除]ボタンをクリックし削除してください。

マクロファイルの削除はルネサス統合開発環境のインストールディレクトリ下の「Macro」ディレクトリにあるマクロファイルを直接削除してください。

## 1.2 マクロ生成支援機能のサポート範囲

ルネサス統合開発環境のすべての操作がマクロファイルに記録できるわけではありません。マクロファイルに記録できない操作もあります。記録できる操作について次の分類で記載します。

- ルネサス統合開発環境共通で記録可能な操作
- デバッグプラットフォーム依存で記録可能な操作

### 1.2.1 ルネサス統合開発環境共通で記録可能な操作

以下の操作は、すべてのデバッグプラットフォームに共通で記録可能です。

- メニュー、ショートカットキー、およびツールバーボタンの操作

表 1-1 メニュー、ショートカットキー、またはツールバーボタン で記録可能な操作

メニュー	メニューオプション	対応するコマンド
ファイル	ワークスペースを開く	OPEN_WORKSPACE
	ワークスペースの保存	SAVE_WORKSPACE
	ワークスペースを閉じる	CLOSE_WORKSPACE
	新規セッション	CHANGE_SESSION
	セッションのインポート	CHANGE_SESSION
	セッションの保存	SAVE_SESSION
	セッションのリフレッシュ	REFRESH_SESSION
	ダウンロードモジュールの追加	「ダイアログボックス」を参照してください。
	最近使ったワークスペース	OPEN_WORKSPACE
編集	最近ダウンロードしたモジュール	FILE_LOAD
	ブレークポイントの挿入/削除	SET_DISASSEMBLY_SOFT_BREAK SET_SOURCE_SOFT_BREAK
表示	ブレークポイントの有効化/無効化	STATE_DISASSEMBLY_SOFT_BREAK STATE_SOURCE_SOFT_BREAK
	ワークスペース	「ウィンドウ」を参照してください。
	逆アセンブリ	
	CPU	
	メモリ	
プロジェクト	IO	
	アクティブプロジェクトに設定	CHANGE_PROJECT
	プロジェクトの挿入	CHANGE_PROJECT
ビルド *2	構成の編集 *2	SAVE_SESSION
	コンパイル	BUILD_FILE
	ビルド	BUILD
	すべてをビルド	BUILD_ALL
	複数ビルド	「ダイアログボックス」を参照してください。
	クリーンアクティブプロジェクト	CLEAN
	すべてをクリーン	CLEAN
	ビルドの構成	「ダイアログボックス」を参照してください。
デバッグ	ビルドの構成	
	デバッグセッション	
	CPU のリセット	RESET
	実行	GO
	リセット後実行	GO_RESET
	カーソル位置まで実行	GO_TILL
	カーソル位置を PC 値に設定	REGISTER_SET *1
	条件を指定して実行	GO_TILL
ステップイン	STEP	

	ステップオーバー		STEP_OVER
	ステップアウト		STEP_OUT
	ステップ		「ダイアログボックス」を参照してください。
	ステップ モード	自動	STEP_MODE
		アセンブリ	STEP_MODE
		ソース	STEP_MODE
	プログラムの停止		HALT
	初期化		INITIALIZE
	接続 *2		CONNECT
	接続解除 *2		DISCONNECT
	メモリの保存		FILE_SAVE
	メモリのベリファイ *2		FILE_VERIFY
	ダウンロード	ダウンロードモジュール名	FILE_LOAD
		All Download Modules	FILE_LOAD_ALL
	アンロード	ダウンロードモジュール名	FILE_UNLOAD
		All Download Modules	FILE_UNLOAD_ALL
基本設定	基数	16 進数	RADIX
		10 進数	RADIX
		8 進数	RADIX
		2 進数	RADIX

\*1. エミュレータまたはシミュレータのヘルプを参照してください。

\*2. 機能のサポートは、デバッグプラットフォームに依存します。



- ウィンドウで記録可能な操作

表 1-2 ウィンドウで記録可能な操作

ウィンドウ	対象	機能/動作	記録するコマンド
ワークスペース ウィンドウの [Projects]タブ	ワークスペース のポップアップ メニュー	クリーン 全プロジェクト *2	CLEAN
	プロジェクトの ポップアップ メニュー	ビルド *2	BUILD
		すべてをビルド *2	BUILD_ALL
		クリーンアクティブプロジェクト *2	CLEAN
		アクティブプロジェクトに設定	CHANGE_PROJECT
	プロジェクト ファイルのポップ アップメニュー	コンパイル<ファイル名> *2	BUILD_FILE
	Download modules フォルダのポップ アップメニュー	全てダウンロード	FILE_LOAD
		ダウンロードモジュールの追加	「ダイアログボックス」を参照してください。
	Download modules のポップアップ メニュー	ダウンロード	FILE_LOAD
		ダウンロード (debug 情報のみ)	FILE_LOAD
アンロード		FILE_UNLOAD	
ダウンロードモジュールの追加		「ダイアログボックス」を参照してください。	
ダウンロード モジュール名	ダブルクリックによるダウンロード	FILE_LOAD	
エディタおよび 逆アセンブリ (ソースモード)	ポップアップ メニュー	コンパイル<ファイル名> *3	BUILD_FILE
		ブレークポイントの挿入/削除	SET_SOURCE_SOFT_BREAK
		ブレークポイントの有効化/無効化	STATE_SOURCE_SOFT_BREAK
		カーソル位置まで実行	GO_TILL
		カーソル位置に PC 値を設定	REGISTER_SET *1
S/W ブレーク ポイントカラム	ダブルクリックによるブレークポイントの挿入/削除	SET_SOURCE_SOFT_BREAK	
エディタおよび 逆アセンブリ (混合モード/ 逆アセンブリ モード)	ポップアップ メニュー	カーソル位置まで実行	GO_TILL
		カーソル位置に PC 値を設定	REGISTER_SET *1
		ブレークポイントの挿入/削除	SET_DISASSEMBLY_SOFT_BREAK
		ブレークポイントの有効化/無効化	STATE_DISASSEMBLY_SOFT_BREAK
	S/W ブレークポイント - ASM カラム	ダブルクリックによるブレークポイントの挿入/削除	SET_DISASSEMBLY_SOFT_BREAK
レジスタ	値	インプレース編集	REGISTER_SET *1
	フラグレジスタ *2	クリックによる編集	REGISTER_SET *1
	ポップアップ メニュー	編集	REGISTER_SET *1
メモリ	値	インプレース編集	MEMORY_EDIT
	ポップアップ メニュー/ ツールボタン	設定	MEMORY_EDIT
		フィル	MEMORY_FILL
		コピー	MEMORY_MOVE
		比較 *2	MEMORY_COMPARE
		保存	FILE_SAVE
		書き込み	FILE_LOAD
IO	値	インプレース編集	MEMORY_EDIT
		ダブルクリックによるブレークポイントの挿入/削除	MEMORY_EDIT

- \*1. エミュレータまたはシミュレータのヘルプを参照してください。  
 \*2. 機能のサポートは、デバッグングプラットフォームに依存します。  
 \*3. エディタウィンドウのみ表示します。

- ダイアログボックスで記録可能な操作

表 1-3 ダイアログボックスで記録可能な操作

ダイアログボックス	機能/動作	記録するコマンド
ダウンロードモジュール	[オフセット]、[フォーマット]、[ファイル名]、[アクセスサイズ]、[ダウンロード時のメモリペリファイ]オプションの入力	FILE_LOAD
複数のビルド *1	[ビルド]または[すべてをビルド]ボタンのクリック	BUILD_MULTIPLE
	[クリーン]ボタンのクリック	CLEAN
ビルドコンフィグレーション	[現在のコンフィグレーション]ドロップダウンリストの切り替え	CHANGE_CONFIGURATION
デバッグセッション	[現在のセッション]ドロップダウンリストの切り替え	CHANGE_SESSION
プログラムステップ	[ステップオーバー]チェックボックスのオン	STEP_OVER
	[ステップオーバー]チェックボックスのオフ	STEP

\*1. 機能のサポートは、デバッグプラットフォームに依存します。

### 1.2.2 デバッグプラットフォーム依存で記録可能なコマンド

以下の操作はデバッグプラットフォームに依存して記録の可否が異なります。

- SuperH RISC engine および H8 ファミリー用エミュレータ/シミュレータデバッガ のみ記録可能な操作

表 1-4 メニュー、ショートカットキー、またはツールバーボタン で記録可能な操作

メニュー	メニューオプション		記録するコマンド
表示	シンボル	ラベル	「ウィンドウ」を参照してください。
		ウォッチ	
		ローカル	

表 1-5 ウィンドウで記録可能な操作

ウィンドウ	対象	機能/動作	記録するコマンド	
ラベル	ポップアップメニュー /ツールバーボタン	追加	SYMBOL_ADD	
		削除	SYMBOL_CLEAR	
		すべてを削除	SYMBOL_CLEAR	
		ロード	SYMBOL_ADD	
	BP カラム	ダブルクリックによるブレークポイントの 挿入/削除	SET_DISASSEMBLY_SOFT_BREAK	
ウォッチ	ポップアップ メニュー/ ツールバーボタン	自動更新有効化	WATCH_AUTO_UPDATE	
		全シンボル自動更新有効化	WATCH_AUTO_UPDATE	
		自動更新無効化	WATCH_AUTO_UPDATE	
		全シンボル自動更新無効化	WATCH_AUTO_UPDATE	
		値更新記録	記録開始	WATCH_RECORD
			記録終了	WATCH_RECORD
		シンボル登録	WATCH_ADD	
		値の編集	WATCH_EDIT	
		削除	WATCH_DELETE	
		全シンボル削除	WATCH_DELETE	
		基数	16 進数表示	WATCH_RADIX
			10 進数表示	WATCH_RADIX
			8 進数表示	WATCH_RADIX
			2 進数表示	WATCH_RADIX
+/-マーク	クリックによる配列の展開/縮小	WATCH_EXPAND		
値	インプレース編集	WATCH_EDIT		
ローカル	ポップアップ メニュー/ ツールバーボタン	編集	WATCH_ADD WATCH_EDIT WATCH_DELETE	
		値	WATCH_ADD WATCH_EDIT WATCH_DELETE	

- SuperH RISC engine および H8 ファミリー用シミュレータデバッガのみ記録可能な操作

表 1-6 メニュー、ショートカットキー、またはツールバーボタン で記録可能な操作

メニュー	メニューオプション		記録するコマンド
表示	CPU	I/O シミュレーション	「ウィンドウ」を参照してください。
	コード	カバレッジ	
		トレース	
		イベントポイント	

表 1-7 ウィンドウで記録可能な操作

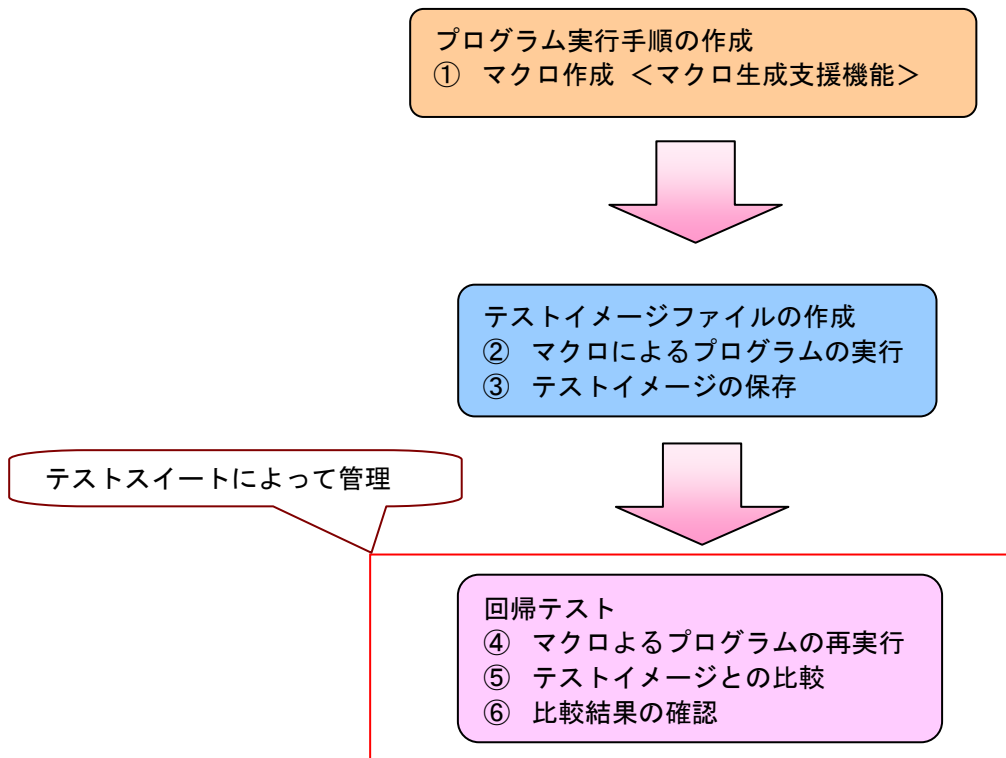
ウィンドウ	対象	機能/動作	記録するコマンド			
I/O シミュレーション	ポップアップメニュー/ ツールバーボタン	削除	SIMULATEDIO_CLEAR			
カバレッジ	ポップアップメニュー/ ツールバーボタン	すべて有効	COVERAGE			
		すべてクリア	COVERAGE			
		測定範囲追加	COVERAGE_RANGE			
		測定範囲編集	COVERAGE_RANGE			
		有効	COVERAGE			
		クリア	COVERAGE			
		保存	COVERAGE_SAVE			
		ロード	COVERAGE_LOAD			
トレース	ポップアップメニュー/ ツールバーボタン	設定	TRACE_ACQUISITION			
イベント - Software Break	ポップアップメニュー/ ツールバーボタン	追加	PC ブレークポイント	BREAKPOINT		
			ブレークアクセス	BREAK_ACCESS		
			ブレークデータ	BREAK_DATA		
			ブレークレジスタ	BREAK_REGISTER		
			ブレークシーケンス	BREAK_SEQUENCE		
			ブレークサイクル	BREAK_CYCLE		
		編集	PC ブレークポイント	BREAK_CLEAR BREAKPOINT		
			ブレークアクセス	BREAK_CLEAR BREAK_ACCESS		
			ブレークデータ	BREAK_CLEAR BREAK_DATA		
			ブレークレジスタ	BREAK_CLEAR BREAK_REGISTER		
			ブレークシーケンス	BREAK_CLEAR BREAK_SEQUENCE		
			ブレークサイクル	BREAK_CLEAR BREAK_CYCLE		
		有効	BREAK_ENABLE			
		無効	BREAK_ENABLE			
		削除	BREAK_CLEAR			
		全て削除	BREAK_CLEAR			
		イベント - Software Event	ポップアップメニュー/ ツールバーボタン	追加	PC ブレークポイント	BREAKPOINT
					ブレークアクセス	BREAK_ACCESS
					ブレークデータ	BREAK_DATA
					ブレークレジスタ	BREAK_REGISTER
ブレークシーケンス	BREAK_SEQUENCE					
ブレークサイクル	BREAK_CYCLE					
編集	PC ブレークポイント			BREAK_CLEAR BREAKPOINT		
	ブレークアクセス			BREAK_CLEAR BREAK_ACCESS		
	ブレークデータ			BREAK_CLEAR BREAK_DATA		

		ブ레이크レジスタ	BREAK_CLEAR BREAK_REGISTER
		ブ레이크シーケンス	BREAK_CLEAR BREAK_SEQUENCE
		ブ레이크サイクル	BREAK_CLEAR BREAK_CYCLE
	有効	BREAK_ENABLE	
	無効	BREAK_ENABLE	
	削除	BREAK_CLEAR	
	全て削除	BREAK_CLEAR	

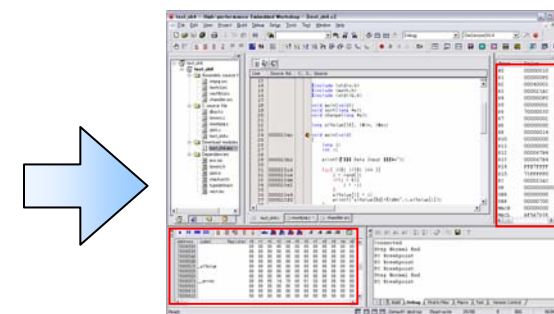
### 2. テスト支援機能

テスト支援機能とは、ユーザアプリケーションの回帰テストを自動化する機能です。回帰テストとは、ある手順を経て実行されたプログラムの実行結果を”正しい結果”とし(これをテストイメージと呼びます)、プログラムの修正やビルドオプションに変更を加えた後に同じ手順でプログラムを再実行し、そのときの実行結果が”正しい結果”と同じ結果であるか否かを確認するテストです。

テスト支援機能は、マクロファイルを用いてプログラムを自動的に実行する機能と、予め保存しておいたプログラムの実行結果とマクロにより再実行されたプログラムの実行結果を比較する機能から構成されています。2つの機能の組み合わせ方は”テストスイート”と言う単位で管理されます。



#### ① マクロ作成 <マクロ生成支援機能>



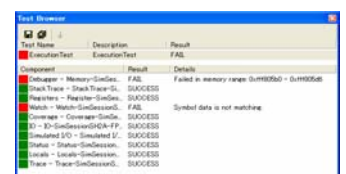
マクロによる、  
② プログラム実行  
④ プログラム再実行

#### ③ テストイメージの保存

R0	00000010
R1	000000F0
R2	00040001
R3	000023AC
R4	000000F0
R5	00000002
R6	70000030
R7	00000002
R8	00000000

00	00	00	00	00	00
00	00	00	00	00	00
00	00	00	00	00	00
00	00	00	00	00	00
00	00	00	00	00	00
00	00	00	00	00	00
00	00	00	00	00	00
00	00	05	14	70	00
00	00	00	00	01	C0

#### ⑤ テストイメージとの比較



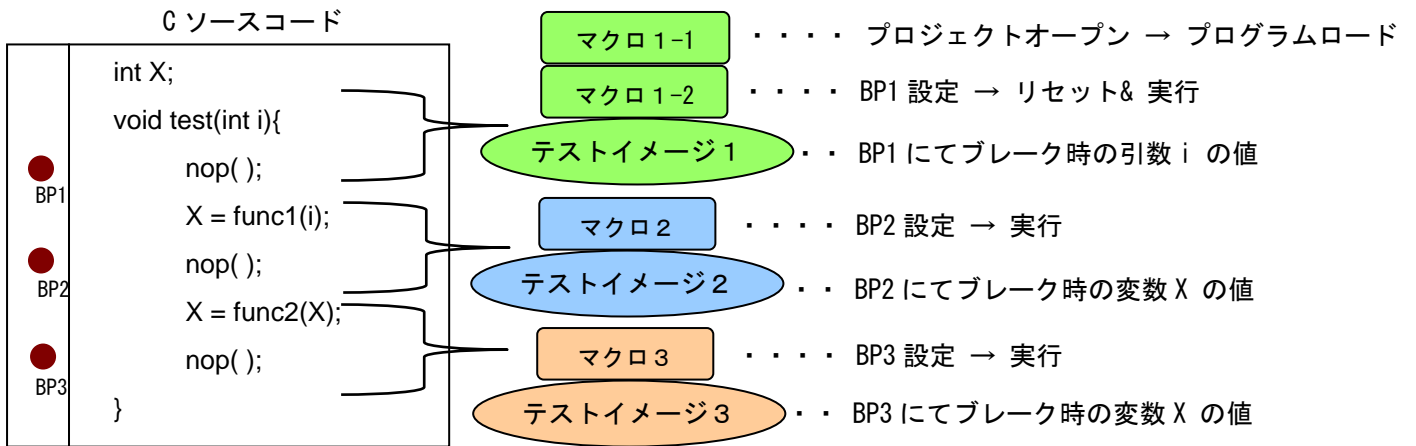
#### ⑥ 比較結果の確認

図 2-1

## 2.1 テストスイート

テスト支援機能に必要なマクロとテストイメージは”テストスイート”により管理されます。テストスイートは1つ以上の”テストケース”から構成されており、”テストケース”は1つ以上のマクロと1つのテストイメージから構成されます。アプリケーションのテストでは、プログラムが特定の状態にあるときの変数の値を確認する必要があります。そのため、テストは特定の状態を作り出すためのマクロとそのときのテストイメージから構成されます。テストスイートの構成例を以下に示します。

テスト内容 : test 関数呼び出し時の引数、func1 の実行結果、func2 の実行結果を確認する。



BP = ブレークポイント

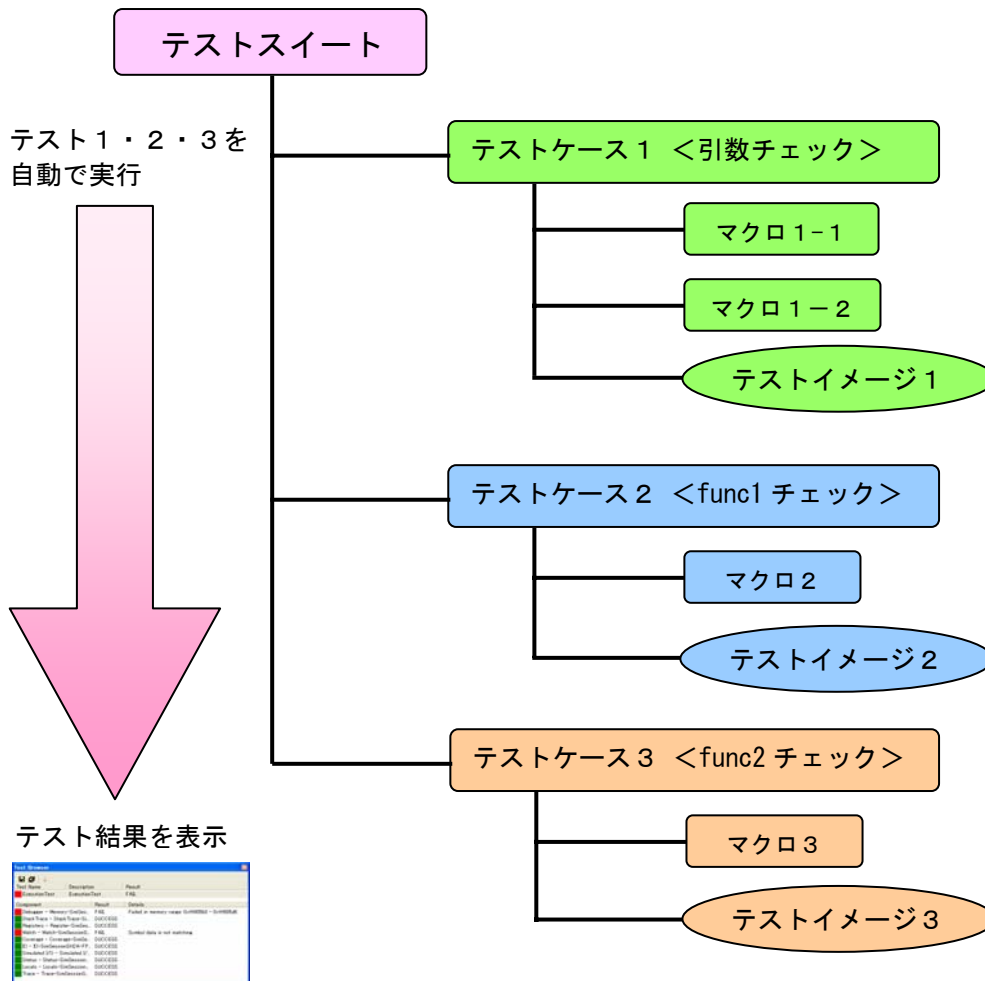


図 2-2

### 2.1.1 テストスイートの作成

テストスイートを作成するには、[テスト]メニューの[テストスイートの作成]を選択してください。[新規テストスイート作成]ダイアログボックスが表示されます。

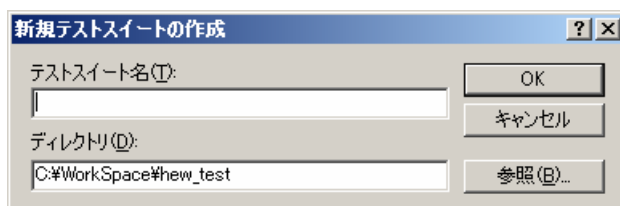


図 2-3

[テストスイート名]フィールドにテストスイート名を入力します。[ディレクトリ]フィールドにはあらかじめワークスペースディレクトリが表示されています(ワークスペースを開いていない場合はルネサス統合開発環境インストールディレクトリが表示されます)。必要に応じて編集してください。

[OK]ボタンをクリックするとテストスイートが作成されます。テストスイートが作成されるとワークスペースウィンドウの[Test]タブにテストスイートが追加されます。このタブにより、テストスイートの操作およびテストケースへ容易にアクセスできます。

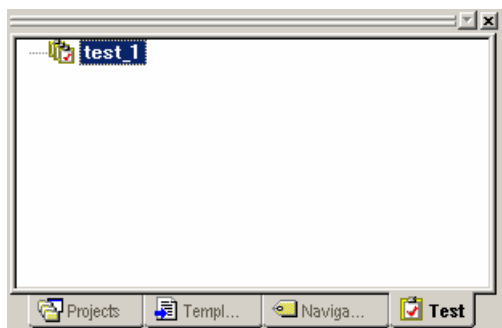


図 2-4

テストスイートファイルは、保存先の場所に拡張子 “.hts” の付いたファイルとして保存されます。

### 2.1.2 テストスイートの開き方/閉じ方

- テストスイートを開く

[テスト]メニューの[テストスイートを開く]を選択してください。[テストスイートを開く]ダイアログボックスでテストスイートファイルを選択し、[選択]ボタンをクリックしてください。テストスイートを開くと[Test]メニューにある他のアイテムが使用可能になります。また、ワークスペースウィンドウの[Test]タブに、テストスイートの内容が表示されます。

最近開いた4個までのテストスイートファイルを[ファイル]メニューの[最近使ったテストスイート]のサブメニューに表示されます。最近使ったテストスイートを再び開きたいときに使用してください。

- テストスイートを閉じる

[テスト]メニューの[テストスイートを閉じる]を選択してください。現在のテストスイートが閉じ、ワークスペースウィンドウの[Test]タブからすべてのアイテムが削除されます。

ワークスペースウィンドウの[Test]タブのポップアップメニューからも、テストスイートを閉じることができます。



### 2.1.3 テストスイートの編集

テストスイートには複数の”テストケース”を登録できます。テストスイートにテストケースを追加する方法、テストスイートに登録されているテストケースを編集する方法について説明します。

<テストケースの作成>

”テストケース”は、プログラムを実行する手順(マクロ)とテストイメージ(2.2章で詳しく説明します)から構成されます。

テストケースの作成は、まず、[テスト]メニューの[テストスイートの編集]を選択し[テストスイートの編集]ダイアログボックスを表示してください。

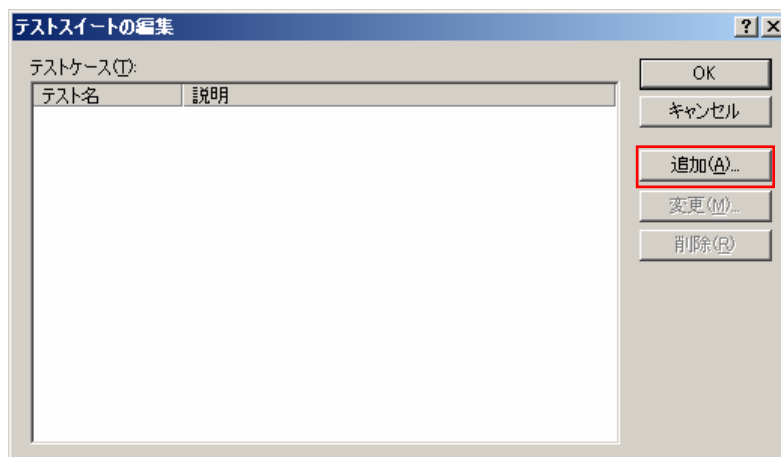


図 2-5

[テストスイートの編集]ダイアログボックスで[追加]ボタンをクリックすると、[テストの追加]ダイアログボックスが表示されます(図 2-6)。テスト名とテストの説明を入力してください。テスト名にはスペースが含まれないようにしてください。テストの説明には後からテスト内容が理解できるよう、詳細な説明を記入してください。

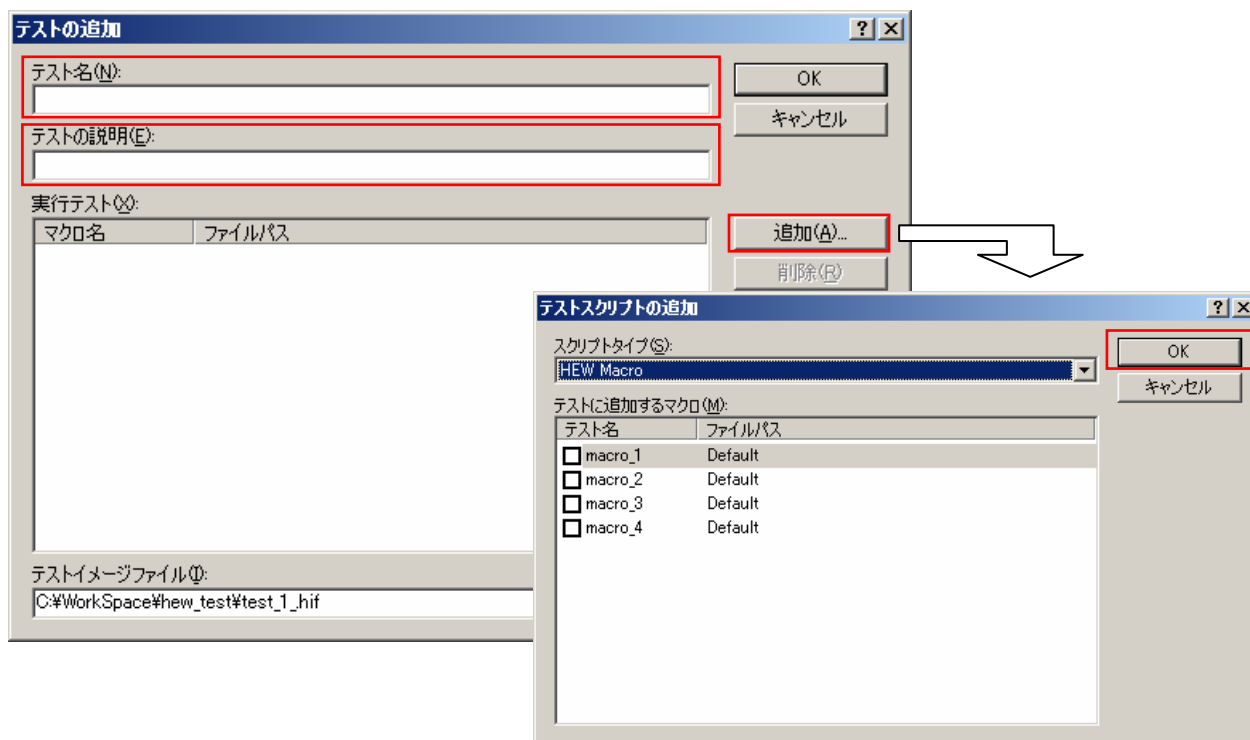


図 2-6

テストで使用するマクロは、[テストの追加]ダイアログボックスの[追加]ボタンから登録します。[追加]ボタンをクリックすると、[テストスクリプトの追加]ダイアログボックスが表示されます(図 2-6)。[スクリプトタイプ]ドロップダウンリストで“HEW Macro”を選択すると、すべての登録済みのマクロが[テストに追加するマクロ]リストに表示されます。使用したいマクロ名の横にあるチェックボックスをオンにしてください。マクロを追加した後、[テストの追加]ダイアログボックスの[実行テスト]で、登録したマクロの順番を変更することができます。テストが実行されるときは上から順番にマクロが処理されます。[上端へ]、[上へ]、[下へ]、[下端へ]のボタンで順番を変更してください(図 2-7)。尚、[スクリプトタイプ]ドロップダウンリストで“Tcl command line batch file”を選択すると、ユーザ定義のスクリプトファイルを使用することが出来ます。

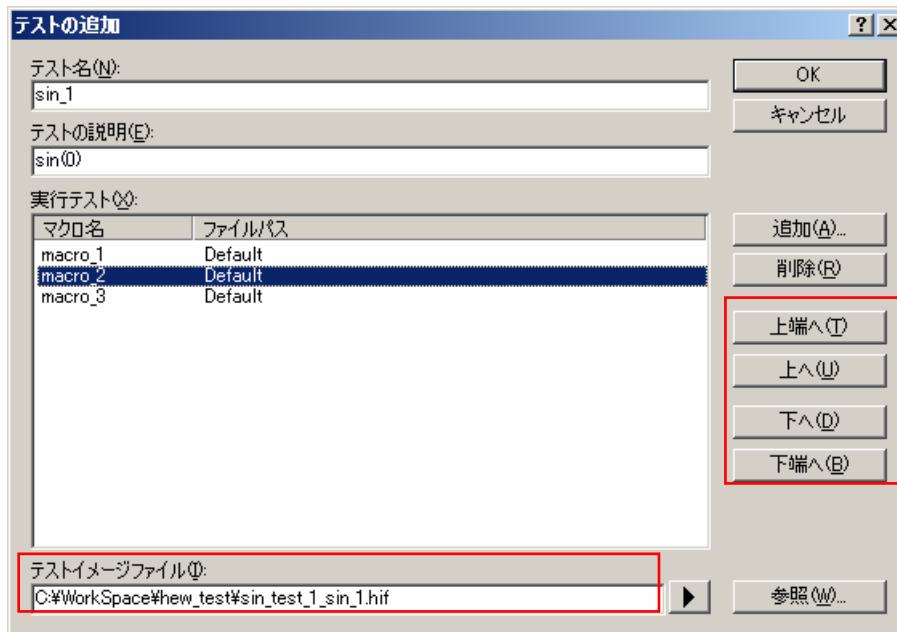


図 2-7

テストで使用するテストイメージファイルは、[テストの追加]ダイアログボックスの[テストイメージファイル]フィールドで指定します。デフォルトのファイル名はテストスイートと同じディレクトリに “テストスイート名 + テスト名.hif” です。[テストスイートの編集]ダイアログボックス(図 2-5)で[OK]ボタンをクリックすると、空のテストイメージファイルが作成されます。テストイメージファイルについては「2.2テストイメージファイル」で詳細を説明します。

<テストケースの編集／削除>

既存のテストケースを編集 / 削除したい場合は、[テストスイートの編集]ダイアログボックス(図 2-5)で、変更したいテストケースを選択し、[編集] / [削除]ボタンをクリックしてください。

既存のテスト編集する場合は、[テストケースの編集]ダイアログボックスが表示されます。[テストの追加]ダイアログボックスと同じ内容のダイアログボックスです(図 2-7)。テストケースの作成と同じ要領で変更してください。

## 2.2 テストイメージファイル

回歸テストでは、プログラムの正しいの実行結果を予め”正しい結果”として用意する必要があります。この正しい結果を”テストイメージ”と呼び、テストイメージが保存されたファイルを”テストイメージファイル”と呼びます。1つのテストケースに対して1つのテストイメージファイルが作成されます。テストイメージファイルの拡張子は “.hif” です。

### 2.2.1 テストイメージファイルの編集

テストイメージの対象項目はテストイメージファイルの編集から設定します。ワークスペースウィンドウの[Test]タブより、テストイメージファイルを作成したいテストケースを右クリックしてください。表示されたポップアップメニュー(図 2-8)より[テストイメージファイルの編集]を選択することで、テストイメージファイルの編集ダイアログボックスが表示されます(図 2-9)。テストイメージファイルの編集を終了したときに、テストイメージファイルの編集で設定した各項目の値がテストイメージファイルに保存されます。

取得可能なテストイメージはプラットフォームに依存します。詳しくは、「High-performance Embedded Workshop V.4.02 ユーザーズマニュアル 16.6 テストイメージファイルにテストイメージデータとして保存できる機能」を参照してください。

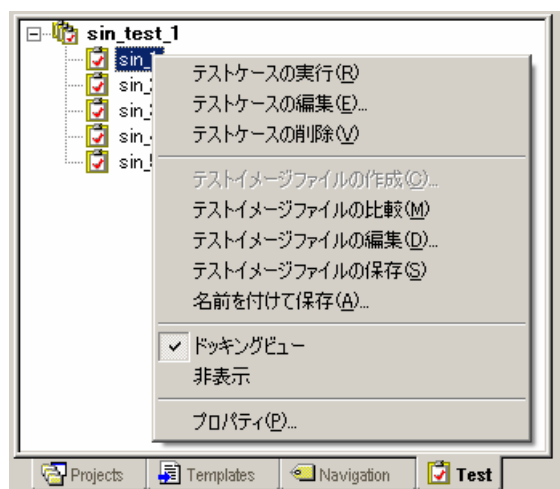


図 2-8



図 2-9

### 2.2.2 テストイメージファイルの保存

[テストイメージファイルの編集] にて選択した項目は[テストイメージファイルの保存]より、テストイメージファイルに保存することができます。

ワークスペースウィンドウの[Test]タブより、テストイメージを保存したいテストケースを右クリックし、表示されたポップアップメニューより[テストイメージファイルの保存]を選択してください。[テストイメージファイルの保存]を選択した時点での各項目の値がテストイメージファイルに保存されます。

### 2.3 テストの実行

テストを実行するには、[テスト]メニューの[テストを実行]を選択してください。[テストを実行]ダイアログボックスが表示されます。

[テストケース]リストにはテストスイートに登録されているテストケースが表示されます。実行したいテストケースのチェックボックスをオンにしてください。テストケースを選択して[上へ]/[下へ]ボタンをクリックすることで、テストケースの実行順序を変更することができます。

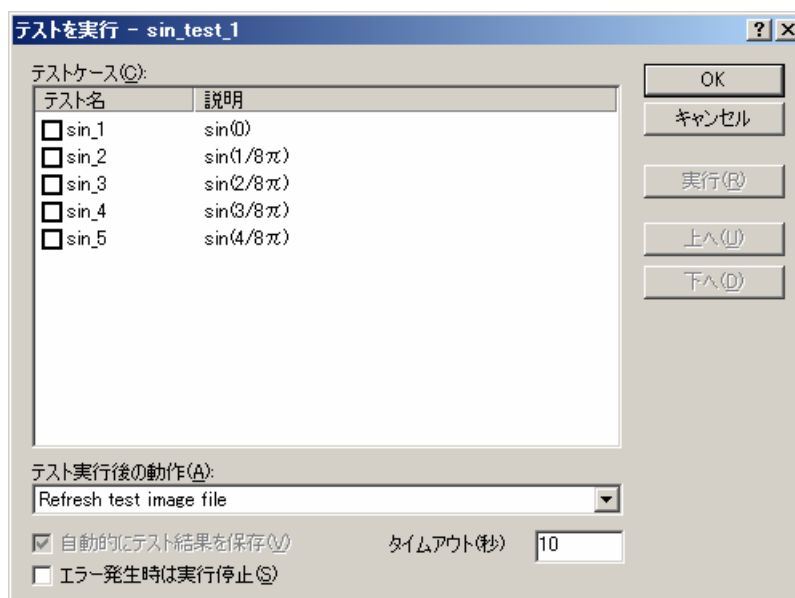


図 2-10

- [テスト実行後の動作]ドロップダウンリストには2つのオプションがあります。
  - “Compare system against saved test image file”オプションは、通常の動作で、関連するテストケースに添付されたテストイメージファイル(\*.HIF)と現在のシステムを比較します。これらの結果はテストブラウザに追加され、テストの成功/失敗、失敗した理由が情報として提供されます。前回のテスト実行結果と比較したい場合に指定してください。
  - “Refresh test image file”オプションは、テストケースを実行後に関連するテストケースに添付されたテストイメージファイル(\*.HIF)ファイルを更新します。最初のテスト(前回のデータがないテスト)の場合に指定してください。
- [自動的にテスト結果を保存]チェックボックスをオンにすると、各テストの実行結果を自動的にテキストファイルに保存することができます。このファイルの場所は、テストスイートと同じディレクトリになります。ファイル名は、現在のテストスイート名とテスト実行時の日付が付加されています。
- [エラー発生時は実行停止]チェックボックスをオンにすると、最初にエラーが発生した時点でテストの実行を停止します。最初のテストが原因で他のテストも失敗してしまう場合、何度もテストを実行せずに済みます。
- [タイムアウト]に入力された秒数よりもテストが長くかかった場合、テストは中止され失敗したとみなされます。処理時間が長いテストを行うときは、タイムアウト時間を長めに設定してください。

## 2.4 テスト結果の確認

テスト完了後、自動的にテスト結果ブラウザが表示されテスト結果が表示されます。

### 2.4.1 テスト結果ブラウザの表示内容

テスト結果ブラウザではテスト結果を確認することができます。

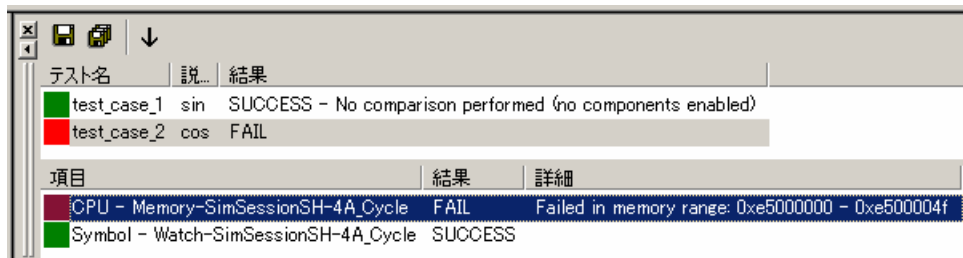


図 2-11

テスト結果ブラウザの上のペインには、今回実行した各テストケースの実行結果が表示されます。テストが最後まで正常に実行し、今回のテスト結果とテストイメージとの間に差分が無い場合は、テストが成功したことを示す緑のアイコンが表示され、差分がある場合は、テストが失敗したことを示す赤のアイコンが表示されます。

上のペインでテストケースを選択すると、そのテストケースのテストイメージに設定された、各テスト項目の実行結果が下のペインに表示されます。上のペイン同様、下のペインでは、今回のテスト結果とテストイメージファイルに差分がない場合は緑のアイコンが表示され、差分がある場合は赤のアイコンが表示されます。また、結果に差分のあるテスト項目は、[詳細]カラムにその内容が表示されます。

下のペインで赤のアイコンで表示されたテスト項目をダブルクリックすると、不一致データの詳細を確認できます(図 2-12)。

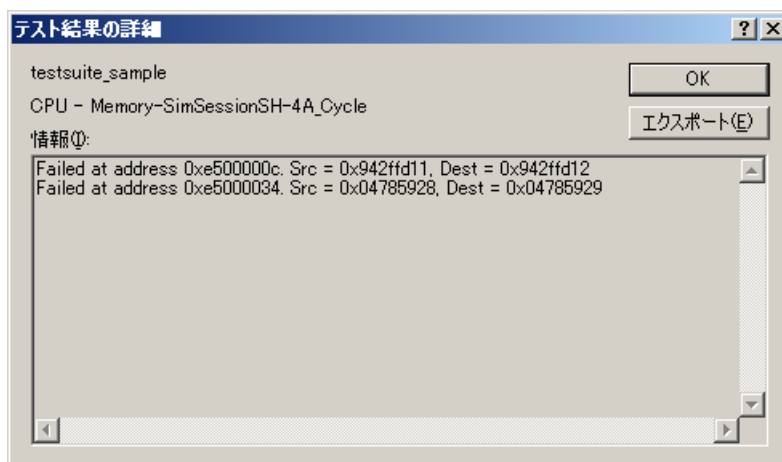


図 2-12

## 2.5 テスト結果の比較

テスト実行後には自動的にテスト結果ブラウザが表示されますが、テスト実行直後以外であっても、テスト結果ブラウザを用いてテスト結果を確認することができます。[テスト]メニューの[テストイメージファイルの比較]を選択してください。[テストイメージファイルの比較]ダイアログボックスが表示されます。

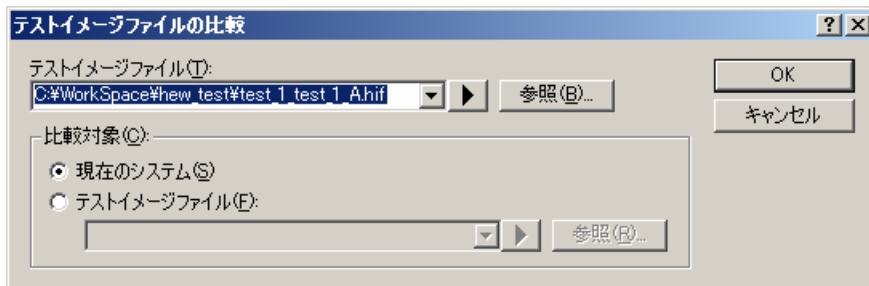


図 2-13

[テストイメージファイル]フィールドには、比較元になるテストイメージファイルを入力してください。

次に、[テストイメージファイル]に指定したファイルと比較する対象を[比較対象]に指定してください。[比較対象]では、現在のシステム、または以前に保存された別のテストイメージファイルを選択できます。テストを手動で実行し、現在のテストイメージと過去に保存したテストイメージファイルを比較して確認したい場合は、[現在のシステム]オプションが便利です。

### 3. チュートリアル

テスト支援機能はデグレードの確認に有用な機能です。本章では、具体的な例を用いてテスト支援機能の使用方法を説明します。本章で作成するサンプルプロジェクトは本ドキュメントと共にダウンロードサイトより提供しています。サンプルプロジェクトは C:\¥WorkSpace¥sample に置いてください。

テストスイートの作成について以下の手順に従って説明します。

- (1) マクロファイルの新規作成
- (2) テスト用のマクロの記録
- (3) マクロの編集
- (4) テストスイートの作成
- (5) 生成したマクロを使用したテストをテストスイートに登録
- (6) 作成したそれぞれのテストに対してテストイメージを作成

テスト結果の確認については以下の手順に従って説明します。

- (a) サンプルプログラムが正しく実行されるかを確認。
- (b) サンプルプログラムを変更し、デグレードが無いかを確認。
- (c) (b)で検出された不具合を修正し、正しく修正されたかを確認。

#### 3.1 サンプルプロジェクトについて

説明で使用するサンプルプロジェクトは、以下の環境で作成されています。この環境より古いバージョンではサンプルプロジェクトを開く事が出来ませんのでご注意ください。

- Renesas SuperH ファミリー用 C/C++コンパイラパッケージ      .....V.9.01Release00
- High-performance Embedded Workshop                      .....V. 4.02.00
- ツールチェーン                                                      .....V. 9.1.0.0

サンプルプロジェクトは、ルネサス統合開発環境のプロジェクト作成機能で生成されるファイルに対して、関数を追加しています。

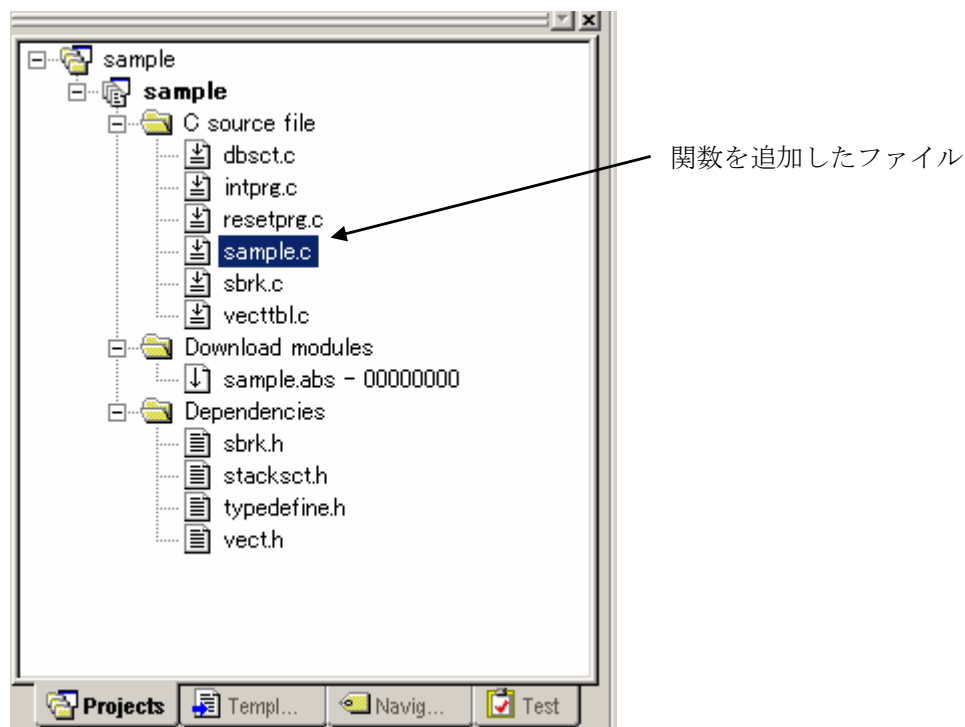


図 3-1

サンプルプログラム(図 3-2)には、テスト対象となる func()関数を定義しています。func()関数は、引数が 0 以外の場合には変数 x、y に 0 を代入し、引数が 0 の場合には変数 x、y はそのままの値を保持します。このプログラムに対するテ

スト項目は以下の(i)(ii)の2項目です。(a)~(c)のケースで、それぞれ(i)(ii)の2項目のテストを行います。

- (i) func()関数の引数が0のとき、変数 x、y の値が変わらないことを確認。
- (ii) func()関数の引数が0以外のとき、変数 x、y の値が0となることを確認。

行番...	ソース
28	#include <machine.h>
29	
30	int x,y;
31	
32	void func(int a);
33	
34	void main(void)
35	{
36	x = y = 1;
37	nop();
38	
39	func(0);
40	nop();
41	
42	func(1);
43	nop();
44	}
45	
46	void func(int a)
47	{
48	if (a)
49	x = 0;
50	if (a)
51	y = 0;
52	}

図 3-2



### 3.2 テストの準備

#### 3.2.1 マクロファイルの生成

まず始めに、マクロファイルを新規に作成します。マクロファイルの作成は、[ツール]メニューの[マクロの設定]で表示される[マクロの設定]ダイアログボックスで[新規]ボタンをクリックしてください。[マクロファイルの新規追加]ダイアログボックスで、新規マクロファイル名(例では、sample\_macro)を入力し、[OK]ボタンをクリックしてください。

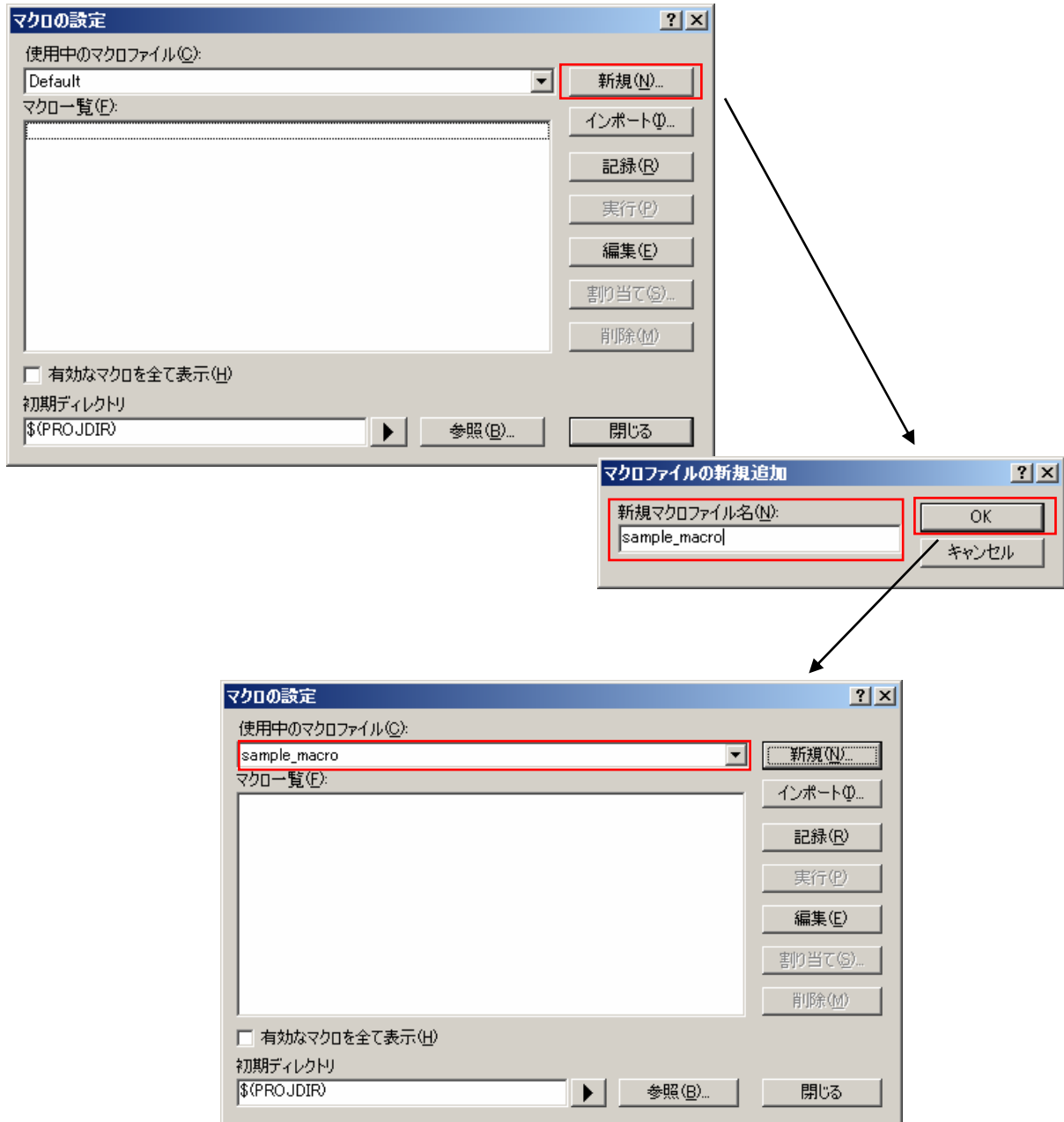


図 3-3

### 3.2.2 マクロの記録

テストで自動的にプログラムを実行するための手順をマクロに記録します。マクロの記録方法は「1.1.1マクロの記録」を参照してください。ここでは、次の2つのマクロを作成します。

(i) テストケース : func(0)

func()関数の引数が0のとき、変数 x、y の値が変わらないことを確認

次の(1)~(3)の操作をマクロ名 sample\_break\_1 で記録します。

- (1) ビルド([ビルド]メニューの[ビルド]を選択)
- (2) エディタウィンドウでプログラムソースの40行目にブレークポイントを挿入  
([編集]メニューの[ブレークポイントの挿入/削除]を選択)
- (3) プログラムの実行([デバッグ]メニューの[リセット後実行]を選択)

行番...	ソースアド...	S...	ソース
28			#include <machine.h>
29			
30			int x,y;
31			
32			void func(int a);
33			
34	00001000		void main(void)
35			{
36	00001002		x = y = 1;
37	0000100C		nop();
38			
39	0000100E		func(0);
40	00001012	●	nop();
41			
42	00001014		func(1);
43	00001018		nop();
44	0000101C		}
45			
46	0000101E		void func(int a)
47			{
48	0000101E		if (a)
49	00001022		x = 0;
50			if (a)
51	00001026		y = 0;
52	0000102C		}

図 3-4

(ii) テストケース : func(1)

func()関数の引数が0以外のとき、変数 x、y の値が0 となることを確認

次の(1)、(2)の操作をマクロ名 sample\_break\_2 で記録します。

- (1) エディタウィンドウでプログラムソースの 43 行目にブレークポイントを挿入  
([編集]メニューの[ブレークポイントの挿入/削除]を選択)
- (2) 実行([デバッグ]メニューの[実行]を選択)

行番..	ソースアド..	S..	ソース
28			#include <machine.h>
29			
30			int x,y;
31			
32			void func(int a);
33			
34	00001000		void main(void)
35			{
36	00001002		x = y = 1;
37	0000100C		nop();
38			
39	0000100E		func(0);
40	00001012	●	nop();
41			
42	00001014		func(1);
43	00001018	●	nop();
44	0000101C		}
45			
46	0000101E		void func(int a)
47			{
48	0000101E		if (a)
49	00001022		x = 0;
50			if (a)
51	00001026		y = 0;
52	0000102C		}

図 3-5

上記2つのマクロはルネサス統合開発環境のインストールディレクトリ下の「Macro」ディレクトリのマクロファイル“sample\_macro.hdc”に記録されます。記録される内容は下記図の通りです。

行番..	ソース
1	
2	##Macro sample_break_1 -Start
3	build wait
4	set_source_soft_break %"c:%%workspace%%sample%%sample%%sample.c%" 0x28 set
5	go_reset
6	##Macro sample_break_1 -End
7	
8	##Macro sample_break_2 -Start
9	set_source_soft_break %"c:%%workspace%%sample%%sample%%sample.c%" 0x2B set
10	go
11	##Macro sample_break_2 -End
12	

図 3-6

### 3.2.3 テストスイートの作成

テストスイートを作成し、作成したテストスイートにテストケースを追加します。

テストスイートを作成するには、[テスト]メニューの[テストスイートの作成]を選択し、[新規テストスイートの作成]ダイアログボックスを表示してください。[新規テストスイートの作成]ダイアログボックスで、作成するテストスイートの名称を[テストスイート名]フィールドに入力し(例では testsuite\_sample)、[OK]ボタンをクリックしてください。

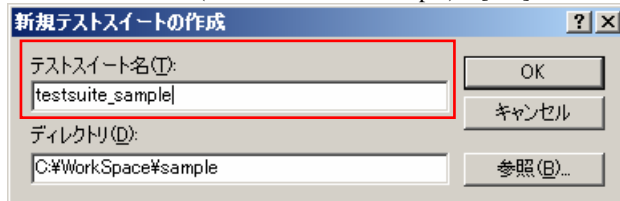


図 3-7

C:\WorkSpace\sample\testsuite\_sample.hts が作成され、ワークスペースウィンドウ([表示]メニューの[ワークスペース])の[Test]タブには、“testsuite\_sample” テストスイートアイコンが表示されます。

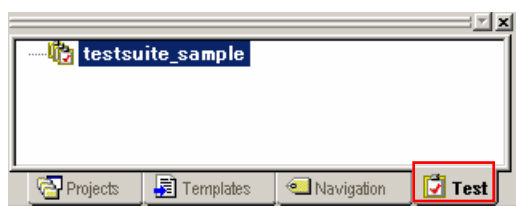


図 3-8

[テスト]メニューの[テストスイートの編集]を選択してください。[テストスイートの編集]ダイアログボックスが表示されます。

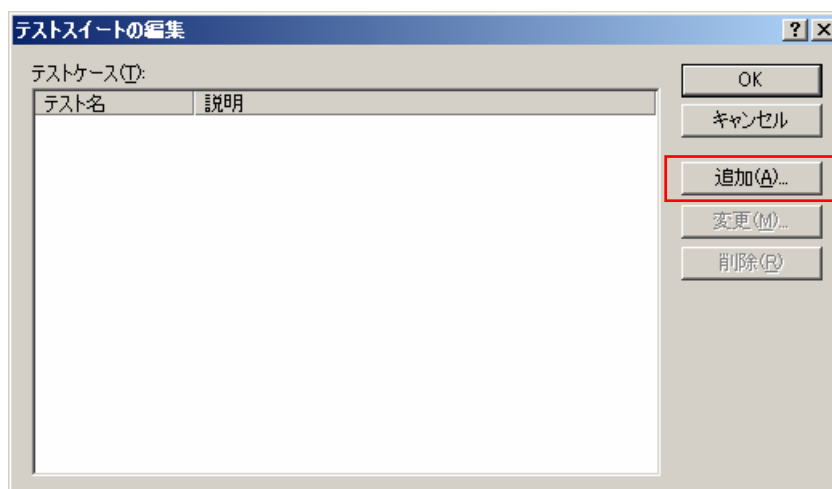


図 3-9

[テストスイートの編集]ダイアログボックスで[追加]ボタンをクリックすると[テストの追加]ダイアログボックスが表示されます。

まず初めに、「(i) テストケース : func(0)」用のテストケース test\_case\_1 を追加します。 [テスト名]フィールドおよび[テストの説明]フィールドを入力します。次に[追加]ボタンをクリックすると、[テストスクリプトの追加]ダイアログボックスが表示されます。

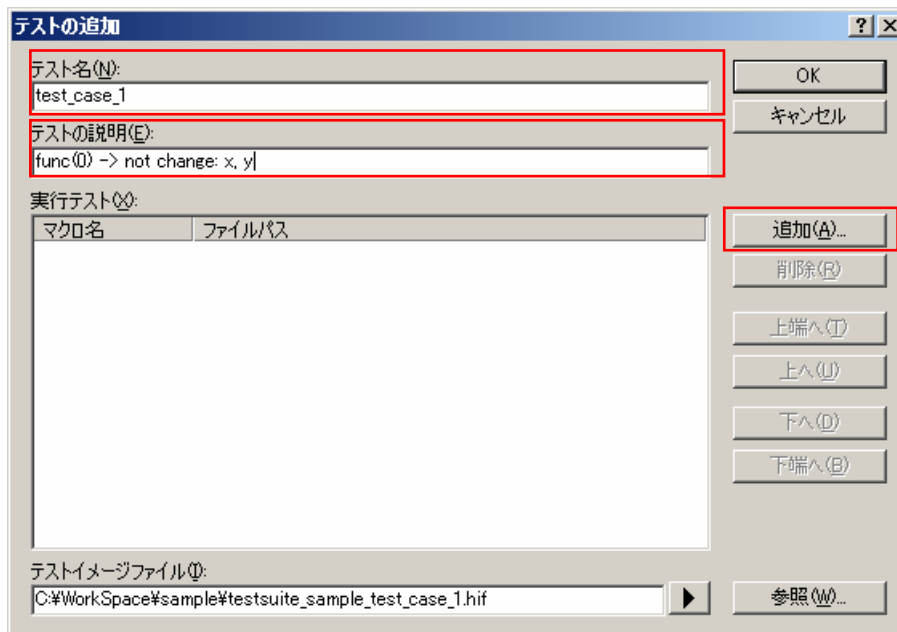


図 3-10

次に先ほど作成したマクロ(sample\_break\_1)の登録を行います。[テストに追加するマクロ]の一覧にて "sample\_break\_1"のチェックボックスをオンにして、[OK]ボタンをクリックします。

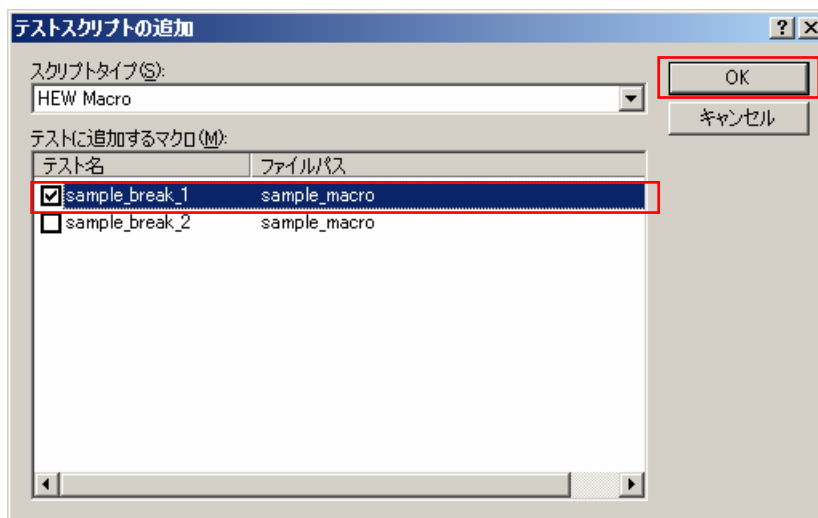


図 3-11

[テストの追加]ダイアログボックスに戻ります。ここで[OK]ボタンをクリックしてください。

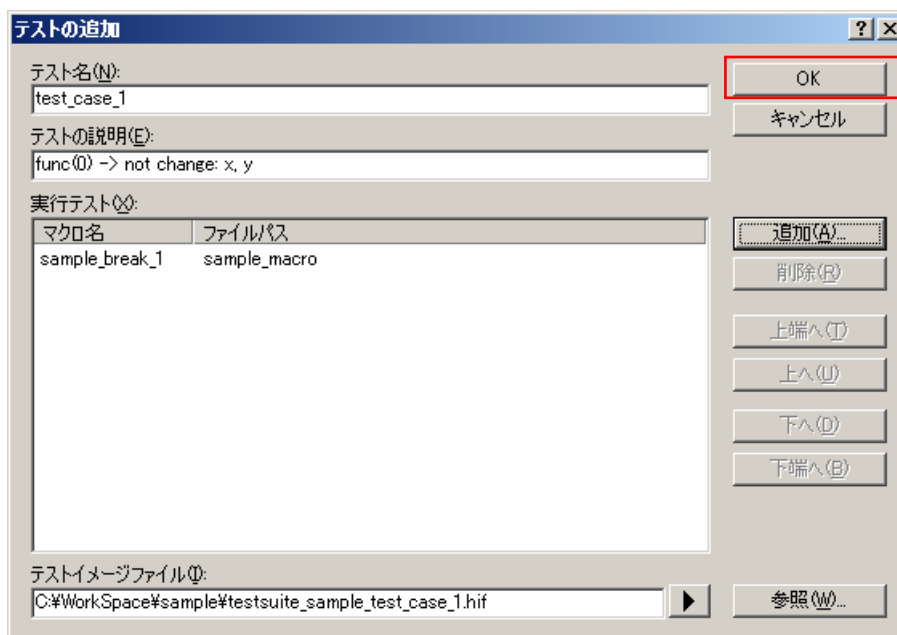


図 3-12

[テストスイートの編集]ダイアログボックスに戻ります。次に「(ii)テストケース : func(1)」用のテストケース test\_case\_2 を以下のように作成します。

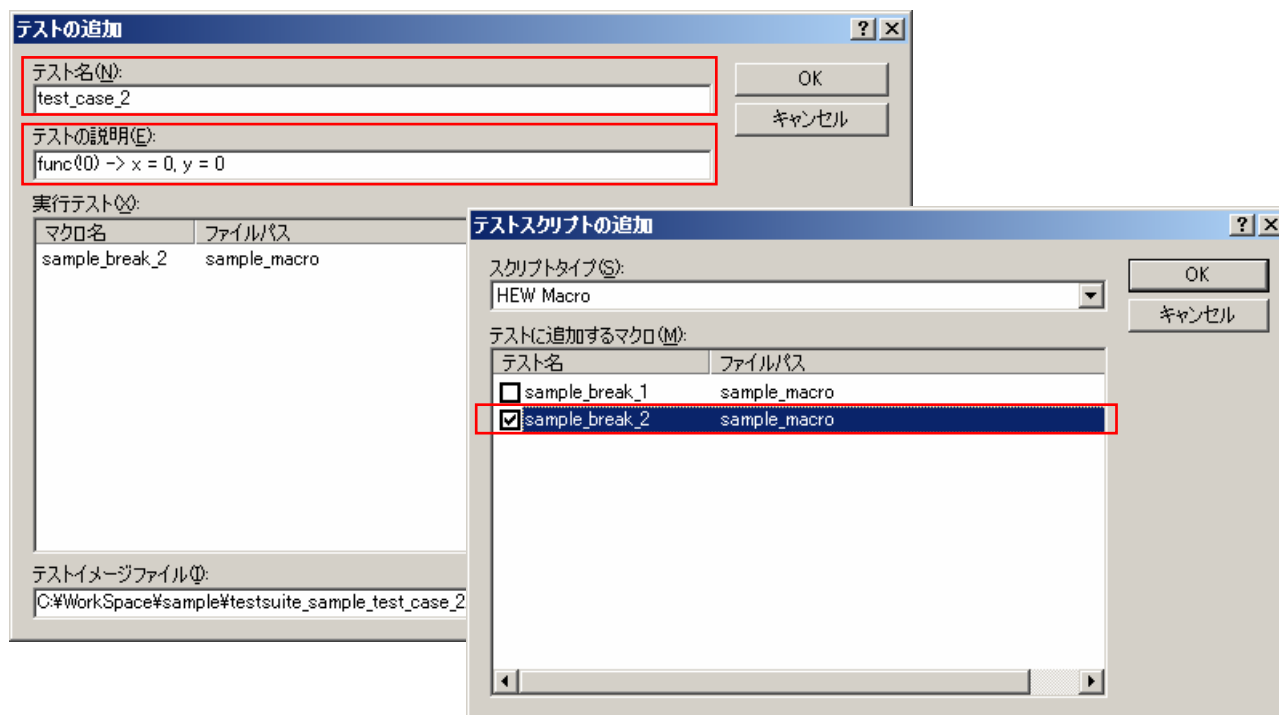


図 3-13

2つのテストケースの編集後、[OK]ボタンをクリックしてください。

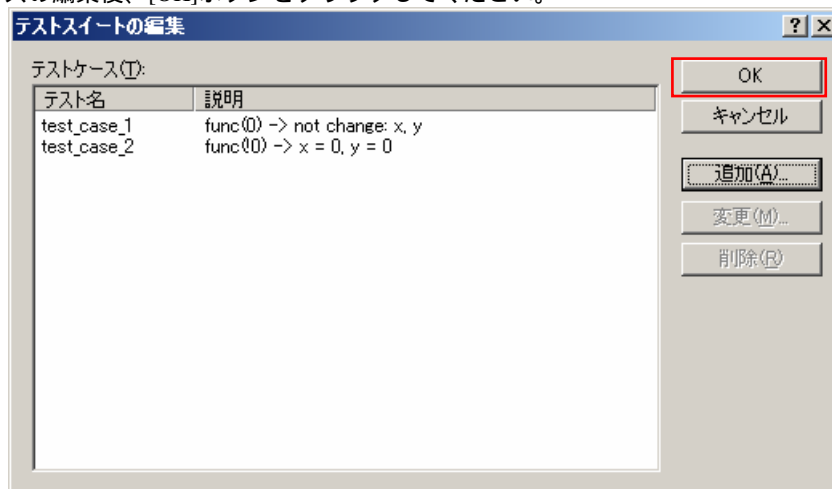


図 3-14

テストイメージファイル、C:\¥Workspace¥sample¥testsuite\_sample\_test\_case\_1.hif、C:\¥Workspace¥sample¥testsuite\_sample\_test\_case\_2.hif が作成されます。また、ワークスペースウィンドウ([表示]メニューの[ワークスペース])の[Test]タブを表示すると、テストスイートにテストケース(test\_case\_1、test\_case\_2)が追加されます。

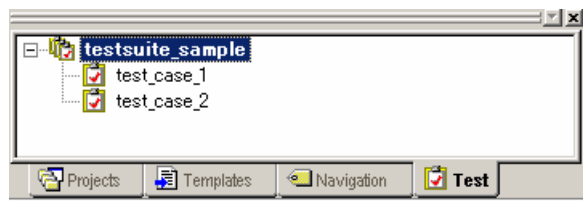


図 3-15

### 3.2.4 テストイメージファイルの作成

テストケースを実行し、プログラムが正しい実行結果を得られたかを確認します。正しい実行結果が得られた場合は、その結果をテストイメージファイルに出力します。確認したいテスト対象(変数の値、レジスタの値、等)の指定は、テストイメージファイルの編集より行います。サンプルプロジェクトではグローバル変数 x、y の値を確認します。変数の値をテストイメージファイルに出力するためには、ウォッチウィンドウ([表示]メニューの[シンボル]の[ウォッチ]を選択)に変数 x、y のシンボル登録をする必要があります。シンボル登録をするために、ウォッチウィンドウで右クリックしポップアップメニュー表示します。表示したポップアップメニューで[シンボル登録]を選択し、[シンボル登録]ダイアログボックスを表示します。そして、[シンボル登録]ダイアログボックスで変数 x および y を登録します。



図 3-16

まず、テストケース test\_case\_1 を実行します。[テスト]メニューの[テストを実行]を選択し、[テストを実行]ダイアログボックスから test\_case\_1 のチェックボックスをオンにしてください。また、テストケース test\_case\_1 には、ビルドを実行するマクロが設定されているため、ビルドが実行されてもタイムアウトしない秒数を[タイムアウト]フィールドに設定する必要があります。

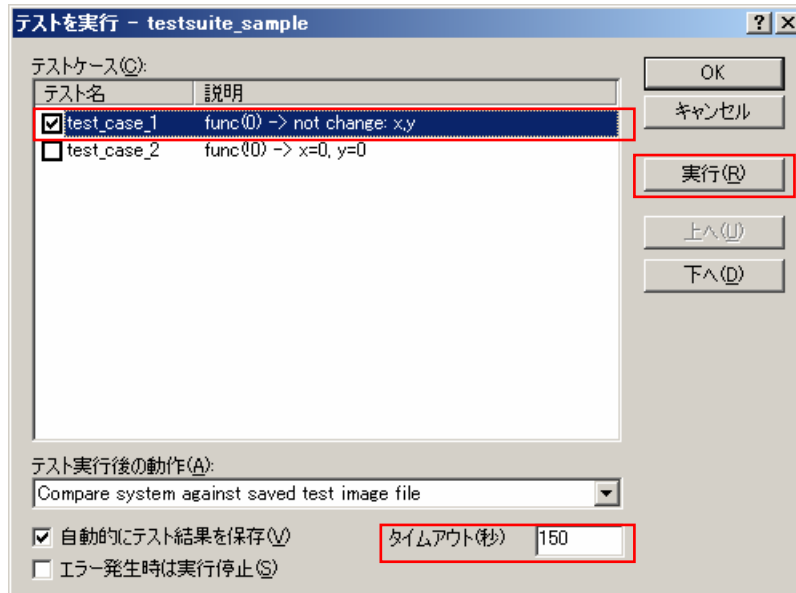


図 3-17

テストを実行すると、test\_case\_1 の確認箇所にブレークします。ここで、ウォッチウィンドウ(図 3-18)で x、y の値が正しいか確認します。ここでは、x=1、y=1 であれば正しい値です。

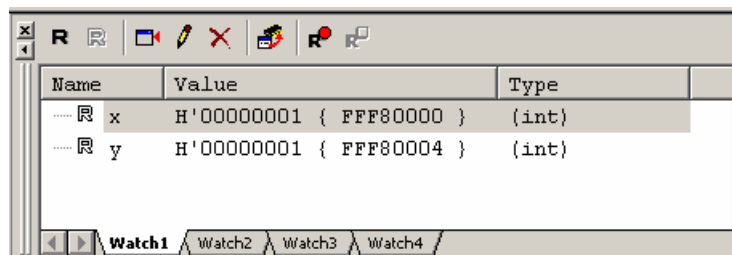


図 3-18



値が正しいことを確認した後、テストイメージファイルの編集を行い、ウォッチウィンドウに表示されている変数  $x$ 、 $y$  の値をテストイメージファイルに指定します。ワークスペースウィンドウの[Test]タブのtest\_case\_1を右クリックして、表示されたポップアップメニューで[テストイメージファイルの編集]を選択します。

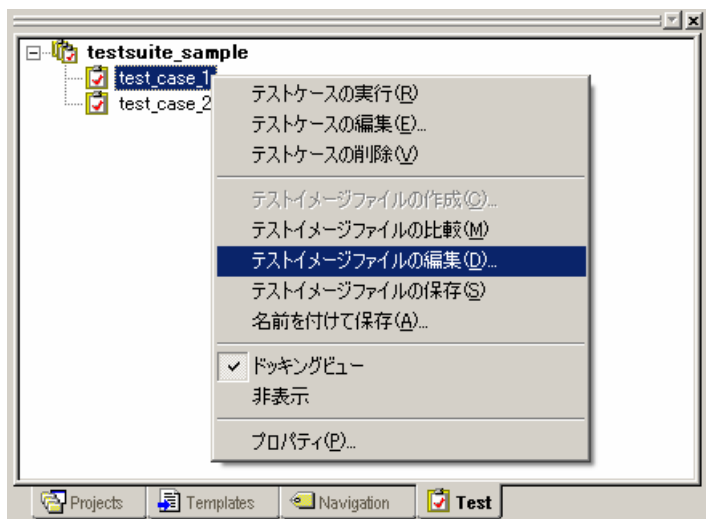


図 3-19

test\_case\_1 で確認したい項目は、ウォッチウィンドウに登録した変数  $x$ 、 $y$  の値です。

まず、[テストイメージファイルの編集]ダイアログボックスにある「Watch-SimSessionSH2A-FPU\_Cycle」のチェックボックスをオンにしてください。

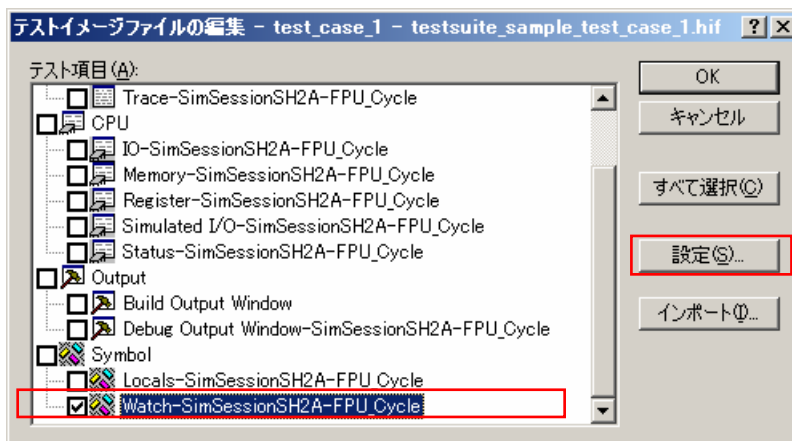


図 3-20

次に [設定] ボタンをクリックしてください。[テスト設定-ウォッチ] ダイアログボックスが表示されます。[テスト設定-ウォッチ] ダイアログボックスのシンボル一覧で変数 x および y のシンボルのチェックボックスをオンにして [OK] ボタンをクリックしてください。

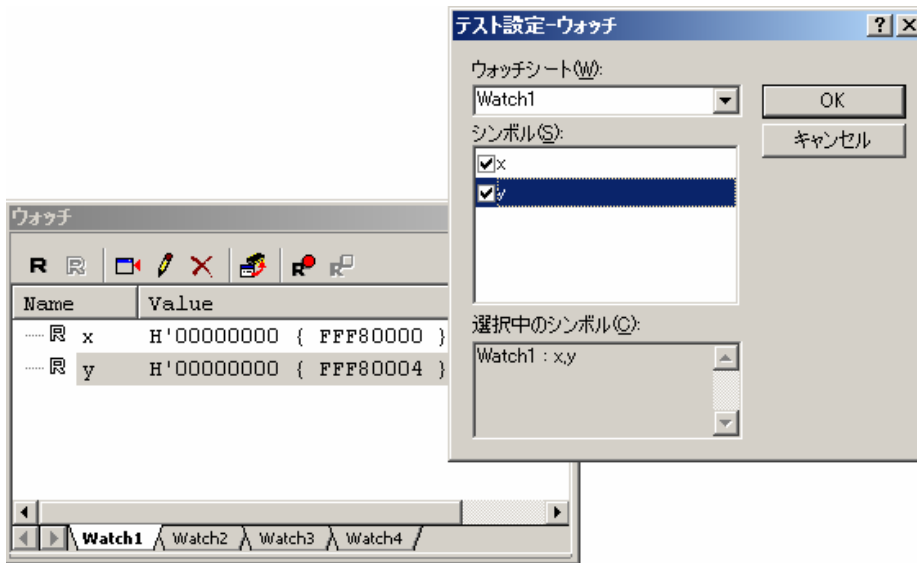


図 3-21

[テストイメージファイルの編集] ダイアログボックスに戻ります。[テストイメージファイルの編集] で [OK] ボタンをクリックすると、ウォッチウィンドウに表示されている変数 x、y の値が、テストイメージファイル C:\¥Workspace¥sample¥testsuite\_sample\_test\_case\_1.hif に保存されます。

test\_case\_1 と同様に、test\_case\_2 のテストを実行します(図 3-22)。

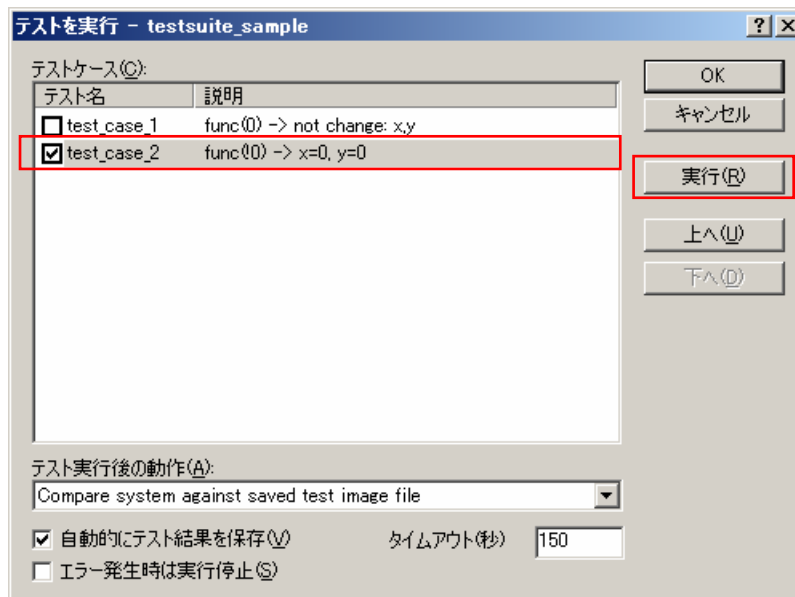


図 3-22

テストを実行すると test\_case\_2 の確認箇所にブレイクします。ここで、ウォッチウィンドウで変数 x、y の値がどちらとも 0 であることを確認します。結果が正しければ、test\_case\_2 に対してテストイメージファイルの編集を行います。test\_case\_1 と同様に、[テストイメージファイルの編集] ダイアログボックスで「Watch-SimSessionSH2A-FPU\_Cycle」のチェックボックスをオンにしてください。テストイメージファイルの編集を終了すると、ウォッチウィンドウに表示されている変数 x、y の値が、テストイメージファイル C:\¥Workspace¥sample¥testsuite\_sample\_test\_case\_2.hif に保存されません。

### 3.3 回帰テスト

テスト支援機能を用いると、先に保存したテスト結果とプログラム変更後のテスト結果を比較することにより、デグレードがないか確認することができます。サンプルプロジェクトに対し、以下の2ケースを例として説明します。

- (1) プログラム変更後の動作確認  
サンプルプログラムを変更し、デグレードが無いかを確認する。
- (2) 正常動作の確認  
(1)で検出された不具合を修正し、正しく修正されたかを確認する。

#### 3.3.1 プログラム変更後の動作確認

func()関数では、変数 x への代入条件と変数 y への代入条件が同じにも関わらず、それぞれの代入に対して if 文を記述している冗長な処理となっています。この if 文を1つに統合するようにプログラムを変更します。ここでは、以下の様にプログラムを変更しました。

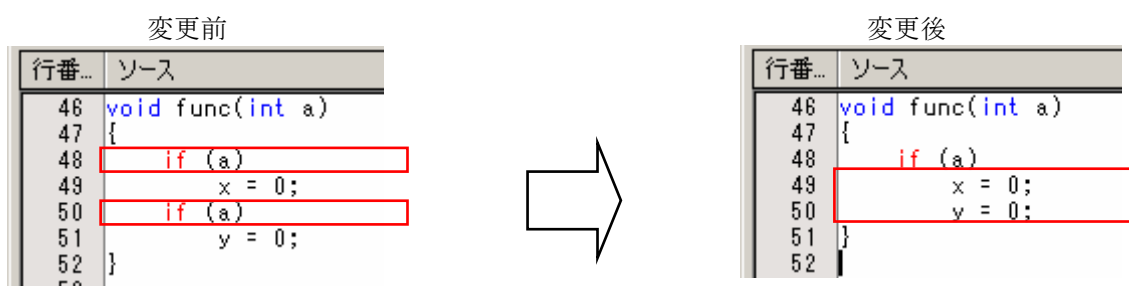


図 3-23

プログラム変更後、次のようにテストを実行し、プログラム変更後のテスト結果がテストイメージと同じであることを確認します。

[テスト]メニューの[テストを実行]を選択し、[テストを実行]ダイアログボックスを表示してください。[テストを実行]ダイアログボックス(図 3-24)のテストケース一覧で、テストケース test\_case\_1、test\_case\_2 のチェックボックスをオンにしてください。

[テスト実行後の動作]ドロップリストで “Compare system against saved test image file” を選択してください。[実行]ボタンをクリックするとテストを実行します。

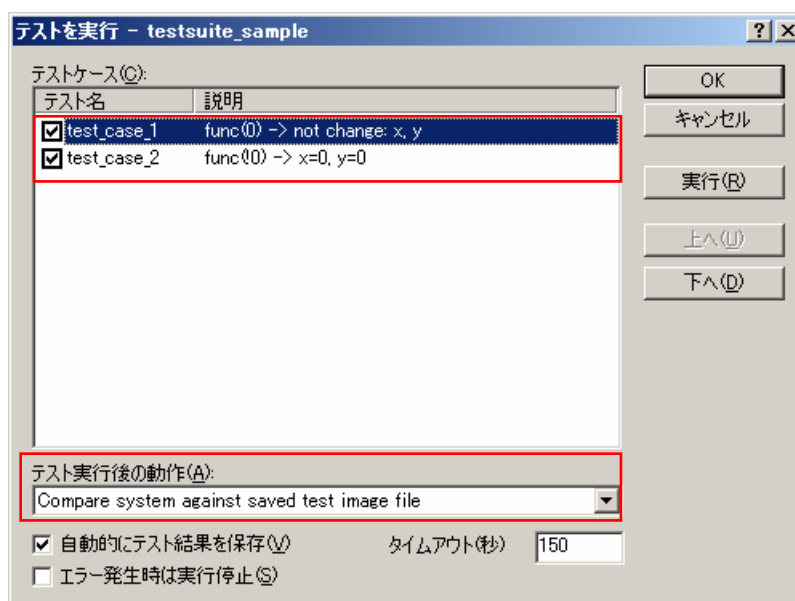


図 3-24

テスト実行後、テスト結果ブラウザ(図 3-25)が表示されます。

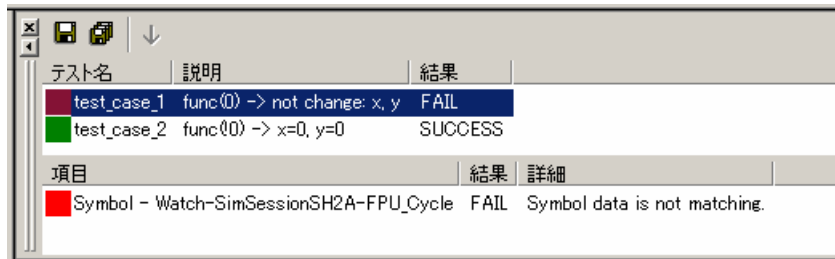


図 3-25

テスト結果ブラウザで赤く表示されている項目は、テストイメージと実行結果が異なる項目です。下側のペインで赤のアイコンで表示されたテスト項目をダブルクリックすると、不一致データの詳細を確認できます。

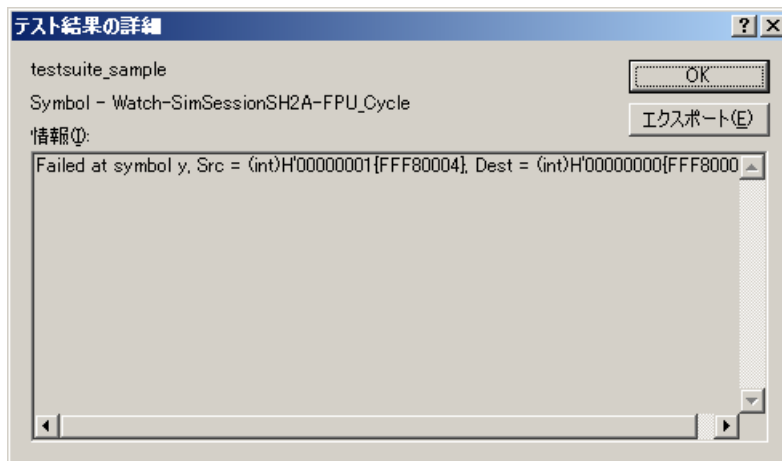


図 3-26

### 3.3.2 正常動作の確認

先のプログラム変更では、if 文の then 節が x への代入式のみとなっており、y には無条件で 0 が代入される処理となっていました。その為、テストケース のテストで、エラーが検出されました。プログラムが正しく動作するように、以下の通り修正を行います。

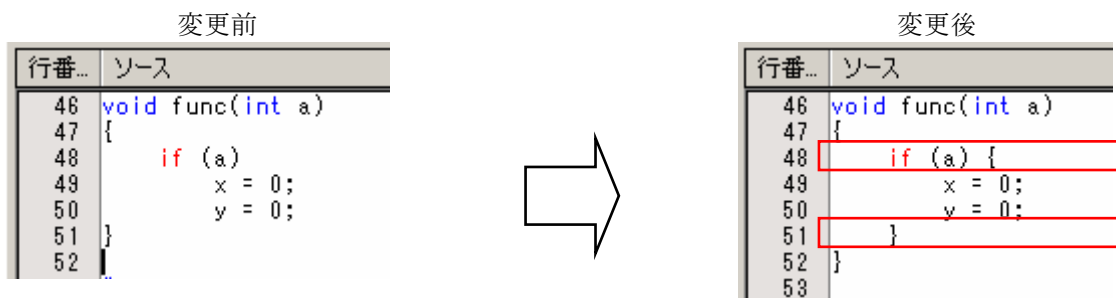


図 3-27

プログラム変更後、再度テストを実行し、不具合が修正されたかを確認します。すべてのテストが成功した場合は、次のようにすべての項目が緑のアイコンで表示されます。

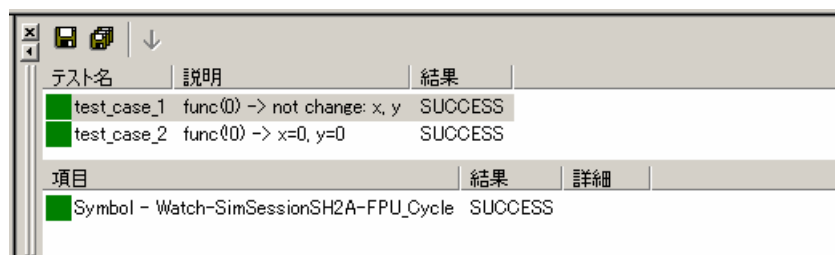


図 3-28

ホームページとサポート窓口<website and support,ws>

ルネサステクノロジホームページ

<http://japan.renesas.com/>

お問合せ先

<http://japan.renesas.com/inquiry>

[csc@renesas.com](mailto:csc@renesas.com)

改訂記録<revision history,rh>

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2008.1.10	—	初版発行

## 安全設計に関するお願い

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

## 本資料ご利用に際しての留意事項

1. 本資料は、お客様が用途に応じた適切なルネサス テクノロジ製品をご購入いただくための参考資料であり、本資料中に記載の技術情報についてルネサス テクノロジが所有する知的財産権その他の権利の実施、使用を許諾するものではありません。
2. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例の使用に起因する損害、第三者所有の権利に対する侵害に関し、ルネサス テクノロジは責任を負いません。
3. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他全ての情報は本資料発行時点のものであり、ルネサス テクノロジは、予告なしに、本資料に記載した製品または仕様を変更することがあります。ルネサス テクノロジ半導体製品のご購入に当たりましては、事前にルネサス テクノロジ、ルネサス販売または特約店へ最新の情報をご確認頂きますとともに、ルネサス テクノロジホームページ(<http://www.renesas.com>)などを通じて公開される情報に常にご注意ください。
4. 本資料に記載した情報は、正確を期すため、慎重に制作したものです。万一本資料の記述誤りに起因する損害がお客様に生じた場合には、ルネサス テクノロジはその責任を負いません。
5. 本資料に記載の製品データ、図、表に示す技術的な内容、プログラム及びアルゴリズムを流用する場合は、技術内容、プログラム、アルゴリズム単位で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断してください。ルネサス テクノロジは、適用可否に対する責任を負いません。
6. 本資料に記載された製品は、人命にかかわるような状況の下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。本資料に記載の製品を運輸、移動体用、医療用、航空宇宙用、原子力制御用、海底中継用機器あるいはシステムなど、特殊用途へのご利用をご検討の際には、ルネサス テクノロジ、ルネサス販売または特約店へご照会ください。
7. 本資料の転載、複製については、文書によるルネサス テクノロジの事前の承諾が必要です。
8. 本資料に関し詳細についてのお問い合わせ、その他お気づきの点がございましたらルネサス テクノロジ、ルネサス販売または特約店までご照会ください。