# 第1章　はじめに

本書では、SuperHファミリ用C/C++コンパイラ(以降、SHC)から、RH850ファミリ用Cコンパイラ(以降、CC-RH)へ移行する際に主に注意すべき以下の点について説明します。

- オプション
- 組み込み関数
- 拡張言語仕様
- プログラムの互換性
- アセンブラ制御命令とマクロ機能
- その他

基本注意事項：
CC-RH では、
- SHC 用のスタートアップルーチン及び IO ヘッダは使用できません。
- SHC の標準ライブラリ構築ツールはありません。
- DSP-C 言語仕様、C++言語仕様には対応していません。

本書が対象とするコンパイラのバージョンを以下に示します。

| コンパイラ | バージョン |
|---|---|
| CC-RH | V1.03.00 |
| SHC | V9.04 Release03 |

# 第2章　オプション

CC-RH における SHC オプションの対応を以下に示します。
なお、SHC ではオプションの大文字と小文字を区別しませんが、CC-RH では、コンパイルオプション、アセンブルオプションの大文字と小文字を区別します。リンクオプションの大文字と小文字は区別しません。

## 2.1　コンパイルオプション

### ● ソースオプション

| No | 機能 | SHC | CC-RH |
|---|---|---|---|
| 1 | オプションの説明を表示 | コマンドライン上で shc を実行 | -h |
| 2 | インクルードファイルパス名を指定 | -include | -I |
| 3 | 指定したファイルをコンパイル単位の先頭にインクルード | -preinclude | -Xpreinclude |
| 4 | マクロの定義 | -define | -D |
| 5 | インフォメーションメッセージ | -message<br>-nomessage | なし |
| 6 | ファイル間インライン展開対象のファイルの取り込み先パス名指定 | -file_inline_path | なし |
| 7 | メッセージレベルの変更 | -change_massage | なし |

### ● オブジェクトオプション

| No | 機能 | SHC | CC-RH |
|---|---|---|---|
| 1 | プリプロセッサ展開後ソースの出力 | -preprocessor<br>-noline | -P<br>-Xpreprocess<br>デフォルトでは#lineを出力しない。(SHCのnoline)<br>#lineを出力する場合は、<br>-Xpreprocess=lineを指定。 |
| 2 | オブジェクト形式 | -code=machinecode<br>-code=asmcode | -c<br>-S |
| 3 | デバッグ情報 | -debug<br>-nodebug | -g |
| 4 | セクション名 | -section | -Xsection |
| 5 | 文字列出力領域 | -string | なし<br>常に const 属性のセクションに出力します。 |
| 6 | オブジェクトファイル名指定 | -objectfile | -o |
| 7 | テンプレートインスタンス生成機能 | -template | なし |
| 8 | 指定したセクションに属するラベルアドレスもしくはランタイムライブラリを配置するメモリ空間を指定 | -abs16/20/28/32 | なし |
| 9 | 除算方式の選択 | -division=cpu | なし |
| 10 | 浮動小数点レジスタの退避・回復を抑止 | -ifunc | なし |
| 11 | ラベルの 16/32 バイト整合 | -align16<br>-align32<br>-noaligne | なし |
| 12 | TBR 相対関数呼び出し指定 | -tbr | なし |
| 13 | 未初期化関数の出力順 | -bss_order | なし |
| 14 | 変数の配置指定 | -stuff | なし |
| 15 | $G0,$G1 セクションの変数の配置指定 | -stuff_gbr | なし |
| 16 | 分岐先アドレスへの境界調整 | -align4 | -Xalign4 |
| 17 | const_volatile 属性変数の割り付け | -const_volatile | 常に const 属性のセクションに出力します。 |

## ● リストオプション

| No | 機能 | SHC | CC-RH |
|---|---|---|---|
| 1 | リストファイルの出力 | -listfile | なし<br>SHC のコンパイル・リスト・ファイルと同等の情報をアセンブル・リスト・ファイルで出力します。<br>-Xasm_option=-Xprn_path をご参照ください。 |
| 2 | リスト内容と形式 | -show | なし<br>アセンブル・リスト・ファイルはカウンタ値、コード、行番号、ソース・プログラム、セクションの種類、サイズ<br>を出力します。 |

## ● 最適化オプション

| No | 機能 | SHC | CC-RH |
|---|---|---|---|
| 1 | 最適化レベル | -optimize | 「●最適化オプション詳細」をご参照ください。 |
| 2 | 実行速度・サイズ最適化 | -speed<br>-size<br>-nospeed | 「●最適化オプション詳細」をご参照ください。 |
| 3 | モジュール間最適化 | -goptimize | なし |
| 4 | 外部変数アクセス最適化（モジュール間） | -map | -Omap |
| 5 | 外部変数アクセス最適化（モジュール内） | -smap | -Osmap |
| 6 | GBR 相対アクセスコード自動生成 | -gbr | なし |
| 7 | switch 文展開方式 | -case | -Xswitch |
| 8 | シフト演算展開 | -shift | なし |
| 9 | 転送コード展開 | -blockcopy | なし |
| 10 | 非アライメントデータ転送 | -unaligned | なし |
| 11 | 自動インライン展開の制御 | -inline<br>-noinline | -Oinline |
| 12 | ファイル間インライン展開 | -file_inline | なし |
| 13 | 外部変数の volatile 化 | -global_volatile | -Xvolatile |
| 14 | 外部変数最適化範囲指定 | -opt_range | なし |
| 15 | 空ループ削除 | -del_vacant_loop | なし |
| 16 | ループ展開 | -loop<br>-noloop | -Ounroll |
| 17 | ループ展開の最大展開数を指定 | -max_unroll | -Ounroll |
| 18 | 無限ループ前の式削除 | -infinite_loop | なし |
| 19 | 外部変数のレジスタ割り付け | -global_alloc | なし |
| 20 | 構造体/共用体メンバのレジスタ割り付け | -struct_alloc | なし |
| 21 | const定数伝播 | -const_var_propagate | なし |
| 22 | 定数ロードの命令展開 | -const_load | なし |
| 23 | 命令並べ替え | -schedule | -Opipeline |
| 24 | ソフトウェアパイプライニング | -softpipe | なし |
| 25 | 最適化範囲分割 | -scope | なし |
| 26 | GBR相対論理演算生成 | -logic_gbr | なし |
| 27 | C++インライン関数の展開抑止 | -cpp_noinline | なし |
| 28 | ポインタ指示先の型を考慮した最適化 | -alias | -Xalias |

● **最適化オプション詳細**

SHCの最適化オプションは、CC-RHとデフォルト値や形式が異なるため、指定方法を変更する必要があります。ただし、最適化の機能は同等ではありませんのでご注意ください。
以下にSHCの最適化オプションの組み合わせに相当するCC-RHのオプションを示します。

| No | SHC | | CC-RH |
|---|---|---|---|
| 1 | -optimize=debug_only | -size | -Onothing |
| | | -speed | |
| | | -nospeed | |
| 2 | -optimize=0 | -size | -Odefault |
| | | -speed | |
| | | -nospeed | |
| 3 | -optimize=1 | -size | -Osize |
| | | -speed | -Ospeed |
| | | -nospeed | -Ospeed,-Ounroll=1,-Oinline=1 |

● **その他オプション**

| No | 機能 | SHC | CC-RH |
|---|---|---|---|
| 1 | EC++言語使用に基づいてシンタックスチェック | -ecpp | なし |
| 2 | DSP-C言語使用に基づいてシンタックスチェック | -dspc | なし |
| 3 | コメントのネスト(/**/) | -comment | なし |
| 4 | MACレジスタ保証 | -macsave | なし |
| 5 | SSR/SPCレジスタ退避・回復 | -save_cont_reg | なし |
| 6 | リターン値の拡張 | -rtnext | なし |
| 7 | 浮動小数点定数除算の乗算化 | -approxdiv | なし |
| 8 | SH7055不具合回避 | -patch=7055 | なし |
| 9 | FPSCRレジスタの切り替え | -fpscr | なし |
| 10 | ループ判定式の最適化抑止 | -volatile_loop | なし |
| 11 | 列挙型サイズの自動選択 | -auto_enum | -Xenum_type=auto |
| 12 | register記憶クラス変数優先割り付け | -enable_register | なし |
| 13 | 以下をANSI準拠で行う<br>・unsigned intとlong型演算<br>・浮動小数点演算の結合則 | -strict_ansi | なし |
| 14 | 整数除算を浮動小数点演算に変換する | -fdiv | なし |
| 15 | 浮動小数点定数を固定小数点定数として扱う | -fixed_const | なし |
| 16 | 1.0r(R)を(long)__fixed型最大値として扱う | -fixed_max | なし |
| 17 | __fixed乗算結果の型変換省略 | -fixed_noround | なし |
| 18 | DSPリピートループ | -repeat | なし |
| 19 | 浮動小数点数-整数変換時の範囲チェック省略 | -simple_float_conv | なし |
| 20 | 除算、除余算をDIVS.DIVU命令で生成しない | -nouse_div_inst | なし |
| 21 | 浮動小数点演算の演算順序変更 | -float_order | なし |

● マイコンオプション

| No | 機能 | SHC | CC-RH |
|---|---|---|---|
| 1 | マイコン/動作モード | -cpu | -Xcpu<br>-Xcommon |
| 2 | メモリのバイト並び順 | -endian | なし |
| 3 | 浮動小数点演算モード | -fpu = { single \| double }<br>-double=float | -Xdbl_size<br>注意：double 型および long double 型の精度を指定します。 |
| 4 | 丸め方式 | -round | -Xround |
| 5 | 非正規化数の扱い | -denormalize | なし |
| 6 | プログラムセクションポジションインディペンデント | -pic | なし |
| 7 | ビットフィールド並び順指定 | -bit_order | -Xbit_order |
| 8 | 構造体、共用体、クラスメンバのアライメント数 | -pack | -Xpack |
| 9 | 例外処理機能 | -exception | なし |
| 10 | 実行時型情報 | -rtti | なし |
| 11 | 除算方式選択 | -division | なし |

● 残りのオプション

| No | 機能 | SHC | CC-RH |
|---|---|---|---|
| 1 | C/C++言語の選択 | -lang | なし |
| 2 | コピーライト出力抑止 | -logo<br>-nologo | なし |
| 3 | 文字列内の文字コード | -euc<br>-sjis<br>-latin1 | -Xcharacter_set |
| 4 | オブジェクト内漢字コード指定 | -outcode | なし |
| 5 | サブコマンドファイルの指定 | -subcommand | @ |

## 2.2 アセンブルオプション

● ソースオプション

| No | 機能 | SHC | CC-RH |
|---|---|---|---|
| 1 | オプションの説明を表示 | コマンドライン上で asmsh を実行 | -h |
| 2 | インクルードファイルパス名を指定 | -include | -I |
| 3 | 置換シンボルの定義 | -define | -D |
| 4 | 整数型プリプロセッサ変数の定義 | -assigna | なし |
| 5 | 文字型プリプロセッサ変数の定義 | -assignc | なし |

● オブジェクトオプション

| No | 機能 | SHC | CC-RH |
|---|---|---|---|
| 1 | デバッグ情報 | -debug | -g |
| 2 | プリプロセッサの展開結果出力 | -expand | なし |
| 3 | リテラルプール出力ポイントの指定 | -literal | なし |
| 4 | オブジェクトモジュールの出力制御 | -object | -o |
| 5 | 未解決シンボルのサイズ指定 | -dispsize | なし |

## ● リストオプション

| No | 機能 | SHC | CC-RH |
|---|---|---|---|
| 1 | リストファイルの出力 | -list | -Xprn_path |
| 2 | ソースプログラムリストの出力制御 | -source | なし |
| 3 | ソースプログラムリストの部分出力の制御およびタブサイズの設定 | -show | なし |
| 4 | クロスリファレンスリストの出力制御 | -cross_reference | なし |
| 5 | セクション情報リストの出力制御 | -section | なし |

## ● その他オプション

| No | 機能 | SHC | CC-RH |
|---|---|---|---|
| 1 | リテラルプール自動生成機能のサイズモードを指定 | -auto_literal | なし |
| 2 | 未参照外部参照シンボル情報の出力抑止 | -exclude | なし |
| 3 | 特権モード命令チェックを指定 | -chkmd | なし |
| 4 | LDTLB 命令チェックを指定 | -chktlb | なし |
| 5 | キャッシュ関連命令チェックを指定 | -chkcache | なし |
| 6 | DSP関連命令チェックを指定 | -chkdsp | なし |
| 7 | FPU関連命令チェックを指定 | -chkfpu | なし |
| 8 | .FDATAの8byteALIGNチェックを指定 | -chkalign8 | なし |

## ● マイコンオプション

| No | 機能 | SHC | CC-RH |
|---|---|---|---|
| 1 | マイコン種別の指定 | -cpu | -Xcpu<br>-Xcommon |
| 2 | メモリのバイト並び順の指定 | -endian | なし |
| 3 | 浮動小数点定数の丸め方式を指定 | -round | なし |
| 4 | 非正規化数となる浮動小数点定数の取扱いを指定 | -denormalize | なし |

## ● 残りのオプション

| No | 機能 | SHC | CC-RH |
|---|---|---|---|
| 1 | 異常終了とするエラーレベルの変更 | -abort | なし |
| 2 | ソース・ファイル内の文字コード | -latin1<br>-sjis<br>-euc | -Xcharacter_set |
| 3 | オブジェクトファイルに出力する漢字コードの指定 | -outcode | なし |
| 4 | アセンブルリストの行数指定 | -lines | なし |
| 5 | アセンブルリストの桁数指定 | -columns | なし |
| 6 | コピーライト出力抑止 | -logo<br>-nologo | なし |
| 7 | サブコマンドファイルの指定 | -subcommand | @ |

## 2.3 リンクオプション

リンクオプションは、SHCと同様に短縮形を指定することができます。オプション、およびパラメータの大文字は短縮形を表します。

### ● 入力オプション

| No | 機能 | SHC | CC-RH |
|---|---|---|---|
| 1 | 入力ファイルを指定 | -Input | -Input |
| 2 | 入力ライブラリファイルを指定 | -LIBrary | -LIBrary |
| 3 | 入力バイナリファイルを指定 | -Binary | -Binary |
| 4 | 未定義シンボルの定義 | -DEFine | -DEFine |
| 5 | 実行開始アドレスを指定 | -ENTry | -ENTry |
| 6 | プレリンカの起動を抑止 | -NOPRElink | なし |

### ● 出力オプション

| No | 機能 | SHC | CC-RH |
|---|---|---|---|
| 1 | 出力形式を指定 | -FOrm | -FOrm |
| 2 | デバッグ情報 | -DEBug | -DEBug |
| 3 | レコードサイズ統一 | -REcord | -RECord |
| 4 | ROM化支援 | -ROm | -ROm |
| 5 | 出力ファイルを指定 | -OUtput | -OUtput |
| 6 | 外部シンボル割り付け情報ファイル | -MAp | -MAp |
| 7 | 空きエリアへの出力値の指定 | -SPace | -SPace |
| 8 | インフォメーションメッセージ | -Message<br>-NOMessage | -Message<br>-NOMessage |
| 9 | 参照されない定義シンボルの通知 | -MSg_unused | -MSg_unused |
| 10 | セクション内データの詰め込み配置 | -DAta_stuff | なし |
| 11 | データレコードのバイト数指定 | -BYte_count | -Byte_count |
| 12 | CRC演算 | -CRc | -CRc |
| 13 | セクション終端にパディング | - PADDING | -PADDING |

### ● リストオプション

| No | 機能 | SHC | CC-RH |
|---|---|---|---|
| 1 | リストファイル出力を指定 | -LISt | -LISt |
| 2 | リストファイルの出力情報を指定 | -SHow | -SHow |

### ● 最適化オプション

| No | 機能 | SHC | CC-RH |
|---|---|---|---|
| 1 | 最適化 | -OPtimize | なし |
| 2 | 共通コード統合対象の最小サイズの指定 | -SAMESize | なし |
| 3 | プロファイル情報ファイルの指定 | -PROfile | なし |
| 4 | キャッシュサイズ | -CAchesize | なし |
| 5 | 最適化部分抑止 | -SYmbol_forbrid<br>-SAMECode_forbrid<br>-Variable_forbrid<br>-FUnction_forbrid<br>-SEction_forbrid<br>-Absolute_forbrid | なし |

● セクションオプション

| No | 機能 | SHC | CC-RH |
|---|---|---|---|
| 1 | セクションの開始アドレス指定 | -STARt | -STARt |
| 2 | 外部定義シンボルアドレスの定義ファイル出力 | -FSymbol | -FSymbol |
| 3 | セクションアライメントを16バイトに変更 | -ALIGNED_SECTION | -ALIGNED_SECTION |

● ベリファイオプション

| No | 機能 | SHC | CC-RH |
|---|---|---|---|
| 1 | アドレス整合性のチェック | -CPu | -CPu |
| 2 | 物理空間上の重複チェック | -PS_check | なし |
| 3 | セクション分割対象外指定 | -CONTIGUOUS_SECTION | なし |

● その他オプション

| No | 機能 | SHC | CC-RH |
|---|---|---|---|
| 1 | S9レコードを常に出力 | -S9 | -S9 |
| 2 | スタック情報ファイル | -STACk | -STACk |
| 3 | デバッグ情報圧縮 | -COmpress<br>-NOCOmpress | -COmpress<br>-NOCOmpress |
| 4 | メモリ使用量削減指定 | -MEMory | -MEMory |
| 5 | シンボル名変更 | -RENname | -RENname |
| 6 | シンボル名削除 | -DELete | -DELete |
| 7 | モジュールの置き換え | -REPlace | -REPlace |
| 8 | モジュールの抽出 | -EXTract | -EXTract |
| 9 | デバッグ情報削除 | -STRip | -STRip |
| 10 | メッセージレベルの変更 | -CHange_message | -CHange_message |
| 11 | ローカルシンボル名秘匿指定 | -Hide | -Hide |
| 12 | 合計セクションサイズの表示 | -Total_size | -Total_size |
| 13 | エミュレータ向けの情報ファイル | -RTs_file | なし |

● サブコマンドファイルオプション

| No | 機能 | SHC | CC-RH |
|---|---|---|---|
| 1 | サブコマンドファイル | -SUbcommand | -SUbcommand |

● マイコンオプション

| No | 機能 | SHC | CC-RH |
|---|---|---|---|
| 1 | SBRアドレス指定 | -SBr | なし |

● 残りのオプション

| No | 機能 | SHC | CC-RH |
|---|---|---|---|
| 1 | コピーライトの出力 | -LOgo<br>-NOLOgo | -Logo<br>-NOLOgo |
| 2 | 継続指定 | -END | -END |
| 3 | 終了指定 | -EXIt | -EXIt |

# 第3章　組み込み関数

SHCの組み込み関数を使用した場合、以下のコンパイルエラーが出力されます。

E0562310: “ファイル”内で未定義の”シンボル”を参照しています。

また、<machine.h> <smachine.h> <umachine.h>をインクルードした場合、以下のコンパイルエラーが出力されます。なおCC-RHでは、組み込み関数使用時にヘッダファイルのインクルードは不要です。

F0520005:ソース・ファイル "ファイル" を開くことができません。

SHC の組み込み関数と同等の処理を行う CC-RH の組み込み関数を以下に示します。

| No | 機能 | SHC | CC-RH |
|---|---|---|---|
| 1 | ステータスレジスタ（SR）の設定 | void set_cr(int cr) | なし |
| 2 | ステータスレジスタ（SR）の参照 | int get_cr(void) | なし |
| 3 | 割り込みマスクの設定 | void set_imask(int mask) | void __set_il_rh(int NUM, void* ADDR); |
| 4 | 割り込みマスクの参照 | int get_imask(void) | unsigned long __stsr_rh(long regID, long selID);<br>__stsr_rh(11, 2)で、PMRを読込 |
| 5 | ベクタベースレジスタ（VBR）の設定 | void set_vbr(void *base) | void __ldsr_rh(long regID, long selID, unsigned long a);（注1） |
| 6 | ベクタベースレジスタ（VBR）の参照 | void *get_vbr(void) | unsigned long __stsr_rh(long regID, long selID);（注1） |
| 7 | グローバルベースレジスタ（GBR）の設定 | void set_gbr(void *base) | なし（注2） |
| 8 | グローバルベースレジスタ（GBR）の参照 | void *get_gbr(void) | なし |
| 9 | グローバルベースレジスタ（GBR）ベースのバイト参照 | unsigned char gbr_read_byte(int offset) | なし |
| 10 | グローバルベースレジスタ（GBR）ベースのワード参照 | unsigned short gbr_read_word(int offset) | なし |
| 11 | グローバルベースレジスタ（GBR）ベースのロングワード参照 | unsigned long gbr_read_long(int offset) | なし |
| 12 | グローバルベースレジスタ（GBR）ベースのバイト設定 | void gbr_write_byte(int offset, unsigned char data) | なし |
| 13 | グローバルベースレジスタ（GBR）ベースのワード設定 | void gbr_write_word(int offset, unsigned short data) | なし |
| 14 | グローバルベースレジスタ（GBR）ベースのロングワード設定 | void gbr_write_long (int offset, unsigned long data) | なし |
| 15 | グローバルベースレジスタ（GBR）ベースのバイトAND | void gbr_and_byte(int offset, unsigned char mask) | なし |
| 16 | グローバルベースレジスタ（GBR）ベースのバイトOR | void gbr_or_byte(int offset, unsigned char mask) | なし |
| 17 | グローバルベースレジスタ（GBR）ベースのバイトXOR | void gbr_xor_byte(int offset, unsigned char mask) | なし |
| 18 | グローバルベースレジスタ（GBR）ベースのバイトTEST | void gbr_tst_byte(int offset, unsigned char mask) | なし |
| 19 | SLEEP命令 | void sleep(void) | void __halt() |
| 20 | TAS命令 | int tas(char *addr) | なし |
| 21 | TRAPA 命令 | int trapa (int trap_no) | なし（注3） |

（注 1）SH の VBR は、RH850 の RBASE・EBASE・INTBP に該当します。
（注 2）SH の GBR は、RH850 の GP・EP に該当します。CC-RH では GP・EP はアセンブラで設定して下さい。
（注 3）SH の TRAPA 命令は、RH850 の TRAP 命令に該当します。

| | | | |
|---|---|---|---|
| 22 | OSシステムコール | int trapa_svc (int trap_no, int code, type1 para1, type2 para2, type3 para3, type4 para4) | なし（注4） |
| 23 | PREF命令 | void prefetch(void *p) | なし |
| 24 | TRACE命令 | void trace(long v) | なし |
| 25 | LDTLB命令 | void ldtlb(void) | なし |
| 26 | NOP命令 | void nop(void) | void __nop(void) |
| 27 | 符号付き64bit乗算の上位32bit | long dmuls_h(long data1, long data2) | long __mul32(long data1, long data2) |
| 28 | 符号付き64bit乗算の下位32bit | unsigned long dmuls_l(long data1, long data2) | なし |
| 29 | 符号なし64bit乗算の上位32bit | unsigned long dmulu_h(unsigned long data1,unsigned long data2) | unsigned long __mul32u(unsigned long data1, unsigned long data2) |
| 30 | 符号なし64bit乗算の下位32bit | unsigned long dmulu_l(unsigned long data1,unsigned long data2) | なし |
| 31 | SWAP.B命令 | unsigned short swapb(unsigned short data) | なし（注5） |
| 32 | SWAP.W命令 | unsigned long swapw(unsigned long data) | long __hsw(long data) |
| 33 | 4バイトデータの上位・下位交換 | unsigned long end_cnvl(unsigned long data) | long __bsw(long data) |
| 34 | MAC.W命令 | int macw(short *ptr1, short *ptr2, unsigned int count)<br>int macwl(short *ptr1, short *ptr2, unsigned int count, unsigned int mask) | なし |
| 35 | MAC.L命令 | int macl(int *ptr1, int *ptr2, unsigned int count)<br>int macll (int *ptr1, int *ptr2, unsigned int count, unsigned int mask) | なし |
| 36 | FPSCRの設定 | void set_fpscr(int cr) | void __ldsr_rh(long regID, long selID, unsigned long a);<br>__ldsr_rh(6, 0, x)で、FPSRをxに設定 |
| 37 | FPSCRの参照 | int get_fpscr(void) | なし |
| 38 | FIPR命令 | float fipr(float vect1[4], float vect2[4]) | なし |
| 39 | FTRV命令 | void ftrv(float vec1[4], float vec2[4]) | なし |
| 40 | 4次元ベクタの4×4行列による変換と4次元ベクタとの和 | void ftrvadd(float vec1[4], float vec2[4], float vec3[4]) | なし |
| 41 | 4次元ベクタの4×4行列による変換と4次元ベクタとの差 | void ftrvsub(float vec1[4], float vec2[4], float vec3[4]) | なし |
| 42 | 4次元ベクタの和 | void add4(float vec1[4], float vec2[4], float vec3[4]) | なし |
| 43 | 4次元ベクタの差 | void sub4(float vec1[4], float vec2[4], float vec3[4]) | なし |
| 44 | 4×4行列の乗算 | void mtrx4mul(float mat1[4][4], float mat2[4][4]) | なし |
| 45 | 4×4行列の乗算と和 | void mtrx4muladd(float mat1[4][4], float mat2[4][4], float mat3[4][4]) | なし |
| 46 | 4×4行列の乗算と差 | void mtrx4mulsub(float mat1[4][4], float mat2[4][4], float mat3[4][4]) | なし |
| 47 | 拡張レジスタへのロード | void ld_ext(float mat[4][4]) | なし |
| 48 | 拡張レジスタからのストア | void st_ext(float mat[4][4]) | なし |
| 49 | 絶対値 | long _ _fixed pabs_lf(long _ _fixed data)<br>long _ _accum pabs_la(long _ _accum data) | なし |
| 50 | MSB検出 | _ _fixed pdmsb_lf(long _ _fixed data)<br>_ _fixed pdmsb_la(long _ _accum data) | なし |

（注 4）SH の OS システムコールは、RH850 の SYSCALL 命令に該当します。
（注 5）CC-RH の組み込み関数「long __bsh(long data)」によって代用は可能です。

| | | | |
|---|---|---|---|
| 51 | 算術シフト | long _ _fixed psha_lf(long _ _fixed data,int count)<br>long _ _accum psha_la(long _ _accum data,int count) | なし |
| 52 | 丸め演算 | _ _accum rndtoa(long _ _accum data)<br>_ _fixed rndtof(long _ _fixed data) | なし |
| 53 | ビットパターンコピー | long _ _fixed long_as_lfixed(long data)<br>long lfixed_as_long(long _ _fixed data) | なし |
| 54 | モジュロアドレッシング設定 | void set_circ_x(_ _X _ _circ _ _fixed array[],<br>size_t size)<br>void set_circ_y(_ _Y _ _circ _ _fixed array[],<br>size_t size) | なし |
| 55 | モジュロアドレッシング解除 | void clr_circ(void) | なし |
| 56 | CSビット（DSRレジスタ）設定 | void set_cs(unsigned int mode) | なし |
| 57 | 正弦・余弦の計算 | void fsca(long angle, float *sinv, float *cosv) | なし |
| 58 | 平方根の逆数 | float fsrra(float data) | なし |
| 59 | 命令キャッシュブロックの無効化 | void icbi(void *p) | なし |
| 60 | キャッシュブロックの無効化 | void ocbi(void *p) | なし |
| 61 | キャッシュブロックのパージ | void ocbp(void *p) | なし |
| 62 | キャッシュブロックの書き戻し | void ocbwb(void *p) | なし |
| 63 | 命令キャッシュブロックのプリフェッチ | void prefi(void *p) | なし |
| 64 | データ操作の同期 | void synco(void) | なし |
| 65 | Tビットの参照 | int movt(void) | なし |
| 66 | Tビットのクリア | void clrt(void) | なし |
| 67 | Tビットのセット | void sett(void) | なし |
| 68 | 連結した64ビットの内容から中央の32ビットを切り出し | unsigned long xtrct(unsigned long data1,<br>unsigned long data2) | なし |
| 69 | 2つのデータとTビットを加算し、キャリーをTビットに反映 | long addc(long data1, long data2) | なし |
| 70 | 2つのデータとTビットを加算し、キャリーを参照 | int ovf_addc(long data1, long data2) | なし |
| 71 | 2つのデータを加算し、キャリーをTビットに反映 | long addv(long data1, long data2) | なし |
| 72 | 2つのデータを加算し、キャリーを参照 | int ovf_addv(long data1, long data2) | なし |
| 73 | data1からdata2とTビットを減算し、ボローをTビットに反映 | long subc(long data1, long data2) | なし |
| 74 | data1からdata2とTビットを減算し、ボローを参照 | int unf_subc(long data1, long data2) | なし |
| 75 | data1からdata2を減算し、ボローをTビットに反映 | long subv(long data1, long data2) | なし |
| 76 | data1からdata2を減算し、ボローを参照 | int unf_subv(long data1, long data2) | なし |
| 77 | 0からdataとTビットを減算し、ボローをTビットに反映 | long negc(long data) | なし |
| 78 | data1/data2の1ステップ除算を行い、結果をTビットに反映 | unsigned long div1(unsigned long data1, unsigned long data2) | なし |
| 79 | data1/data2の符号付き除算の初期設定をし、Tビットを参照 | int div0s(long data1, long data2) | なし |
| 80 | 符号なし除算の初期設定 | void div0u(void) | なし |
| 81 | データを1ビット左方向に回転し、オペランドの外へ出たビットをTビットに反転 | unsigned long rotl(unsigned long data) | なし |

| | | | |
|---|---|---|---|
| 82 | データを1ビット右方向に回転し、オペランドの外へ出たビットをTビットに反転 | unsigned long rotr(unsigned long data) | なし |
| 83 | データを1ビット左方向にTビットを含めて回転し、オペランドの外へ出たビットをTビットに反転 | unsigned long rotcl(unsigned long data) | なし |
| 84 | データを1ビット右方向にTビットを含めて回転し、オペランドの外へ出たビットをTビットに反転 | unsigned long rotcr(unsigned long data) | なし |
| 85 | データを1ビット左シフトし、オペランドの外へ出たビットをTビットに反映 | unsigned long shll(unsigned long data) | なし |
| 86 | データを論理的に、1ビット右シフトし、オペランドの外へ出たビットをTビットに反映 | unsigned long shlr(unsigned long data) | なし |
| 87 | データを算術的に、1ビット右シフトし、オペランドの外へ出たビットをTビットに反映 | long shar(long data) | なし |
| 88 | 符号付き1バイトデータ飽和演算 | long clipsb(long data) | なし |
| 89 | 符号付き2バイトデータ飽和演算 | long clipsw(long data) | なし |
| 90 | 符号なし1バイトデータ飽和演算 | unsigned long clipub(unsigned long data) | なし |
| 91 | 符号なし2バイトデータ飽和演算 | unsigned long clipuw(unsigned long data) | なし |
| 92 | TBRにdataを設定 | void set_tbr(void *data) | なし |
| 93 | TBRの値を参照 | void *get_tbr(void) | なし |
| 94 | SR.RBとSR.BLを0クリアし、SR.IO-I3にimaskを設定し、func関数を呼び出す | void sr_jsr(void *func, int imask) | なし |
| 95 | 指定アドレス(addr)の指定ビット(bit_num)に1を設定 | void bset(unsigned char *addr, unsigned char bit_num) | void __set1(unsigned char *a, long bit) |
| 96 | 指定アドレス(addr)の指定ビット(bit_num)に0を設定 | void bclr(unsigned char *addr, unsigned char bit_num) | void __clr1(unsigned char *a, long bit) |
| 97 | 指定アドレス1(from_addr)の指定ビット1(from_bit_num)を、T ビットと指定アドレス2(to_addr)の指定ビット2(to_bit_num)に設定 | void bcopy(unsigned char *from_addr, unsigned char from_bit_num, unsigned char *to_addr, unsigned char to_bit_num) | なし |
| 98 | 指定アドレス1(from_addr)の指定ビット1(from_bit_num)の反転を、T ビットと指定アドレス2(to_addr)の指定ビット2(to_bit_num)に設定 | void bnotcopy(unsigned char *from_addr, unsigned char from_bit_num, unsigned char *to_addr, unsigned char to_bit_num) | なし |

# 第4章　拡張言語仕様

以下のSHCの#pragma指令は、SHCに依存した拡張仕様です。
記述されている場合は、警告メッセージを出力し、#pragma指令を無効とします。

```
#pragma abs16
#pragma abs20
#pragma abs28
#pragma abs32
#pragma regsave
#pragma noregsave
#pragma noregalloc
#pragma ifunc
#pragma tbr
#pragma gbr_base
#pragma gbr_base1
```

これらの#pragma指令をCC-RHで使用している場合、コンパイル時に以下の警告メッセージが出力されます。
-Xcheck=shcオプションを指定することで、警告メッセージの出力を抑止することができます。

W0520161:認識されない #pragma です。

また、プログラムの動作に影響を及ぼす可能性がある以下のSHCの#pragma指令に対しては、-Xcheck=shc
オプションの指定により、警告メッセージが出力されます。メッセージが出力された場合は、該当する
#pragma指令をご確認ください。

```
#pragma section
#pragma stacksize
#pragma entry
#pragma global_register
#pragma address
```

出力される警告メッセージは以下です。

W0523042:SuperH コンパイラとの互換性に影響ある "#pragma xxxx" が使用されています。

| No | 機能 | SHC | CC-RH |
|---|---|---|---|
| 1 | セクションの切り替え指定 | #pragma section | 書式を拡張していますが、SHCと同じ書式でご使用いただけます。<br><br>注意：<br>SHC と同じ書式で、CC-RH の属性指定文字と同じ名前のセクション名を指定した場合、指定した文字列は、属性指定文字とみなされます。ただし、オプション-Xcheck=shc を指定した場合は 「#pragma section 属性指定文字」の形式とみなした場合にメッセージが出力されます。 |
| 2 | 割り込み関数の作成 | #pragma interrupt | CC-RH の仕様に合わせて適切に変更してください。 |
| 3 | 関数のインライン展開を指定 | #pragma inline | SHC と同じ書式でご使用いただけます。 |
| 4 | アセンブリ記述関数のインライン展開 | #pragma inline_asm | SHC と同じ書式でご使用いただけます。ただし、#pragma inline_asm 内のアセンブラは変更してください。 |
| 5 | 分岐先アドレスの 4 バイト整合 | #pragma align4 | CC-RH の仕様に合わせて適切に変更してください。 |
| 6 | ビットフィールドの並び順指定 | #pragma bit_order | SHC と同じ書式でご使用いただけます。 |
| 7 | 構造体、共用体、クラスのアライメント数指定 | #pragma pack<br>#pragma unpack | 書式を拡張していますが、SHC と同じ書式でご使用いただけます。 |

# 第5章　プログラム互換性

SHC向けにコーディングしたC/C++言語ソース・ファイルをCC-RHへ流用する際、-Xcheck=shcオプションで、互換性に影響するオプション指定、拡張言語仕様をチェックすることができます。未規定及び処理系定義については、-Xmisra2004/-Xmisra2012オプション等を活用し、問題ないことを確認してください。

## 5.1　エンディアン

CC-RH はリトルエンディアンのため、ビッグエンディアンである SHC とデータの配置が異なります。データ配置の詳細は、CC-RH コンパイラユーザーズマニュアル「4.1.6 データの内部表現と領域」をご参照ください。また、以下のようなケースではエンディアンの違いが動作上問題となりますのでご注意ください。

(1) 定義した型とは異なる型へのポインタでアクセスした場合

```
1: int val = 0x12345678;
2: void func( void )
3: {
4:    char *p;
5:    ・・・
6:    p = (char *)&val;
```

※6 行目で int 型の変数“val”を char 型へのポインタでキャストして使用していますので、エンディアンの違いが動作上問題となります。なお、異なる型へのポインタにキャストした場合、CC-RH の -Xmisra2004 オプションによりチェックが可能です。

(2) 共用体で違う型のメンバにアクセスしている場合

```
1: union {
2:     int u1;
3:     struct {
4:         char  s1;
5:         char  s2;
6:         short s3;
7:     }st;
8: }tag;
```

※例えば、tag.u1 で設定した値 0x01020304 を tag.st のメンバで参照すると、
ビッグエンディアンの場合：
tag.st.s1 -> 0x01
tag.st.s2 -> 0x02
tag.st.s3 -> 0x0304
リトルエンディアンの場合：
tag.st.s1 -> 0x04
tag.st.s2 -> 0x03
tag.st.s3 -> 0x0102 と違う値になります。
なお、共用体を使用した場合、CC-RH の-Xmisra2004 オプションによりチェックが可能です。

(3) 異なるエンディアンで配置された初期値を持つ変数にアクセスする場合
ビッグエンディアンで配置された ROM データをリトルエンディアンに変換するか、CC-RH の組み込み関数__bsw()／__bsh() によりエンディアン変換してから使用してください。

## 5.2 ANSI仕様外の記述

SHCでコンパイル可能な以下のプログラムに対して、CC-RHではエラーを出力します。これらはANSI仕様範囲外ですので、内容を確認してプログラムを修正してください。

### (1) 変数の範囲外の定数の代入

・ 例

```
int a = 12345678901234567890123456789012345678901234567890;
```

変数に代入する定数の値は、変数のデータ型で表現可能な範囲に修正してください。

・ 対策例

```
int a = 12345678901234567890;
```

### (2) 拡張表記の文字定数

・ 例

```
char *x = “¥xh”;
```

¥x<16 進数字>は 16 進整数を表現します。16 進数字以外の定数は、¥を削除してください。

・ 対策例

```
char *x = “xh”;
```

### (3) int 型で表現できない値の列挙子の定義

・ 例

```
1: #include <limits.h>
2: enum { A = INT_MAX, B }
```

列挙定数の値は int 型の定数で表現可能な範囲に修正してください。

・ 対策例

```
1: #include <limits.h>
2: enum { A = INT_MAX, B = 0x80000000U };
```

### (4) 前処理指令の不正な文字

・ 例

```
1: #if x ! y == 3
2: #endif
```

前処理指令の不正な文字を削除してください。

・ 対策例

```
1: #if x
2: #endif
```

### (5) 条件式が定数の場合の条件演算子への代入

・ 例

```
1: void main() {
2:     char *p;
3:     * (1 ? (char *)p : (const char *)p) = 0;
4: }
```

代入式の左辺に条件演算子を記述することはできません。以下の対策例のように、間接的に代入してください。

・ 対策例

```
1: void main() {
2:     char *p;
3:     char *p2;
4:     p2 = (1 ? (char *)p : (const char *)p);
5:     *p2 = 0;
6: }
```

### (6) 不統一な関数宣言

・ 例

```
1: void main() {
2:     { int f(int, ...); int f(); }
3: }
```

仮引数の個数、型が一致していない同名の関数の宣言を削除してください。

・ 対策例

```
1: void main() {
2:     { int f(int, ...); }
3: }
```

### (7) defined 単項演算子式の不正な識別子

・ 例

```
1: #if defined (x
2: #endif
```

上記のような前処理指令は、「#if defined (識別子)」または「#if defined 識別子」で記述してください。

・ 対策例

```
1: #if defined (x)
2: #endif
```

### (8) 不完全変数の sizeof

・ 例

```
1: void main(){
2:     extern int i[]; {
3:         extern int i[10];
4:     }
5:     sizeof(i);
6: }
```

sizeof演算の引数に不完全型を渡すことはできません。以下の例のように、修正してください。

・ 対策例

```
1: void main()
2: {
3:     extern int i[]; {
4:         extern int i[10];
5:         sizeof(i);
6:     }
7: }
```

# 第6章　アセンブラ制御命令とマクロ機能

## 6.1　アセンブラ制御命令

SH のアセンブラ制御命令と CC-RH のアセンブラ擬似命令の対応表を以下に示します。

| 分類 | 機能 | SHC | CC-RH |
|---|---|---|---|
| マイコンに関するもの | マイコン種別を指定 | .CPU | なし |
| セクションまたはロケーションカウンタに関するもの | セクションの宣言 | .SECTION | .cseg<br>コード・セクションの開始を指示<br>.dseg<br>データ・セクションの開始を指示<br>.org<br>絶対アドレス形式セクションの開始を指示<br><br>セクション開始アドレスのアライメント数指定はできない |
| | ロケーションカウンタ値を指定 | .ORG | なし |
| | ロケーションカウンタ値をアライメント数の倍数に補正 | .ALIGN | .align |
| シンボルに関するもの | シンボルに値を設定する | .EQU | .equ<br>外部参照シンボルは指定できない |
| | 再定義可能なシンボルを設定する。 | .ASSIGN | .set |
| | レジスタ別名の定義 | .REG | なし |
| | 浮動小数点レジスタ別名を定義 | .FREG | なし |
| データまたはデータ領域を確保するもの | 整数データを確保する | .DATA | .db<br>1 バイト領域の確保<br>. db2/.dhw<br>2 バイト領域の確保<br>.db4/.dbw<br>4 バイト領域の確保 |
| | 整数データブロックを確保する | .DATAB | なし |
| | 文字列データをメモリ上に確保 | .SDATA | .db |
| | 文字列データを指定の数だけメモリ上に確保 | .SDATAB | なし |
| | 計数付き文字列データをメモリ上に確保 | .SDATAC | なし |
| | ゼロ終端文字列データをメモリ上に確保 | .SDATAZ | なし |
| | 浮動小数点データ領域を確保する | .FDATA | .float<br>4 バイト領域の確保<br>.double<br>8 バイト領域の確保 |
| | 浮動小数点定数データブロックを確保する | .FDATAB | なし |
| | 固定小数点データを確保する | .XDATA | なし |
| | データ領域を確保する | .RES<br>サイズを指定し、メモリ上に領域を確保 | .ds<br>任意のサイズを指定し、メモリ上に確保 |
| | 文字列用のデータ領域を確保 | .SRES | なし |
| | 計数付き文字列データ領域をメモリ上に確保 | .SRESC | なし |
| | ゼロ終端文字列データ領域をメモリ上に確保 | .SRESZ | なし |

| | | | |
|---|---|---|---|
| データまたはデータ領域を確保するもの | 浮動小数点定数データ領域をメモリ上に確保 | .FRES | なし |
| 外部定義または外部参照に関するもの | 外部定義シンボルを宣言する | .EXPORT | .public |
| | 外部参照シンボルを宣言する | .IMPORT | .extern |
| | 外部定義シンボルまたは外部参照シンボルを宣言する | .GLOBAL | なし |
| オブジェクトモジュールに関するもの | オブジェクトモジュール、デバッグ情報出力制御 | .OUTPUT | なし |
| | シンボルデバッグ情報の部分出力を制御する | .DEBUG | なし |
| | エンディアン種別を指定する | .ENDIAN | なし |
| | 行番号を変更する | .LINE | なし |
| アセンブルリストの出力制御 | アセンブルリストの出力を制御する | .PRINT | なし |
| | ソースプログラムリストの部分出力を制御する | .LIST | なし |
| | アセンブルリストの行数と桁数を設定する | .FORM | なし |
| | ソースプログラムリストのヘッダの設定する | .HEADING | なし |
| | ソースプログラムリストを改ページする | .PAGE | なし |
| | ソースプログラムリストに空行を出力 | .SPACE | なし |
| その他 | オブジェクトモジュール名を設定 | .PROGRAM | なし |
| | 整数定数の基数指定 | .RADIX | なし |
| | ソース・プログラムの終わりとエントリポイントを指定 | .END | なし |
| | シンボルに対するスタック使用量の定義 | .STACK | .stack |

## 6.2 マクロ機能

SHのマクロ機能のアセンブラ制御文とCC-RHのアセンブラ擬似命令の対応表を以下に示します。

| 機能 | SHC | CC-RH |
|---|---|---|
| マクロ命令の定義 | .MACRO<br>～<br>.ENDM | .macro<br>～<br>.endm |
| マクロ命令の展開を中断 | .EXITM | .exitm<br>繰り返しアセンブル擬似命令の１つ外側にスキップ<br>.exitma<br>繰り返しアセンブル擬似命令の一番外側にスキップ |

# 第7章　その他

以下の機能について、SHC と CC-RH では仕様が異なります。

(1) セクションアドレス演算子
SHC では、__sectop、__secend、__secsize を使用できますが、CC-RH の場合は、アセンブラで以下のセクション集合演算子を記述してください。
- STARTOF
- SIZEOF

(2) enum 型
enum 型に未定義の識別子を設定した場合 SHC では未定義の識別子に 0 が設定されますが、CC-RH ではエラーを出力します。

以上