目次

S1C17Familyソフトウェア開発ツール(GNU17)　リリース履歴

以下は、各リリースバージョンの前回リリースバージョンとの改善及び、変更点の履歴です。
(リリース時期の新しいものから記述しています。)

ツール名	Ver 2.1.0　2011/03/04
統合開発環境(IDE)	・ライブラリ生成用のプロジェクトを作成可能にしました。 ・プロジェクト[プロパティ]-[GNU17 リンカスクリプト設定]でVMA≠LMAの設定のときプルダウンリストに、LMAのセクションを指定可能にしました。 ・コマンドファイルの雛形に機種固有のコマンドファルを使用することを可能としました。 ・既存のプロジェクトをインポートするとき、プロジェクトの機種別情報ファイルが存在しなかった場合、他機種を選択できるようにしました。 ・プロジェクトのビルドに未使用関数の検出機能を追加しました。 ・プロジェクトのビルドで1パス目のリンクではセクションが重複してもエラーにしないようにしました。これは、2 パス目のリンクでターゲットのメモリ領域にプログラムが収まるのに1パス目でエラーにしてしまっていたことの対策です。
C コンパイラ(gcc)	―
アセンブラ(as)	・-mc17_ext が指定されている時の、シンボル名登録に失敗した場合のエラーメッセージを修正
リンカ(ld)	・セクションの重複を許可するオプション –c17-overlap-noerr を追加
ライブラリ	―
デバッガ(gdb)	・c17 df 機能追加 ・c17 chgclkmd 機能追加 ・不具合(GDB-09 参照)の修正
マスクROM作成ツール(Winfog/Winmdc)	―
LCDパネルカスタマイズツール(LCDUtil17)	―
その他	―

ツール名	Ver 2.0.0 2010/02/26
統合開発環境(IDE)	・ GDBと Eclipse の統合 ・ IDEの日本語化 ・ コンパイラオプション追加(プロトタイプワーニング) ・ シンボル登録の連動 C/C++ Project Paths -> path Containers と GNU17 Build Options -> Build Options -> Symbol の登録/削除 が連動するようにしました。 ・ ブート処理で設定する%sp の初期値を変数化 ・ GDB MONOS Flash 対応コマンド(c17 flv,c17 flvs)の対応 ・ プロジェクトプロパティからターゲット CPU を変更できるように改善 ・ Flash メモリプロテクトビット設定機能の追加
C コンパイラ(gcc)	・ -Werror-implicit-function-declaration オプションにデフォルトで対応 これにより、C ソースファイルで、プロトタイプ宣言されていない関数を使用した場合にはエラーが出力されるようになります。
アセンブラ(as)	・ 2 パスメイク処理の高速化 ・ long long 型のデバッグ情報のフィルター処理を追加
リンカ(ld)	―
ライブラリ	・ libstdio.a と libc.a をリンク時のサイズが最小限になるように修正
デバッガ(gdb)	・ メモリウィンドウでヘキサ表示形式の場合に"0x"表示を削除 ・ ショートカットのファンクションキーへの割り当て。 ・ Help メニューにコマンドリファレンスを追加 ・ c17 flv(MONOS フラッシュメモリの書き込み・消去電圧の設定)コマンドを追加 ・ c17 flvs(MONOS フラッシュメモリの書き込み・消去電圧の解除)コマンドを追加 ・ c17 chgclkmd(ブレークモード時のクロックソース選択)コマンドを追加 ・ c17 fls コマンドのパラメータに転送データサイズを追加し、RAM 容量の小さい機種に対応 ・ c17 コマンドのパラメータで空白を含む文字列は、ダブルクオーテーションで囲むことにより認識出来るように変更。(例: c17 fwlp コマンドの Comment パラメータなど) ・ 低速クロック(OSC1)デバッグ時の応答速度の改善 ・ ICDmini ハードウェアバージョン 2.0 の対応 ・ 不具合(GDB-08 参照)の修正
マスク ROM 作成ツール(Winfog/Winmdc)	・ 不具合の修正 Vista のユーザアカウント制御機能(UAC)が有効な場合、DevTools(winfog と winmdc)の設定ファイル(winfog17.ini、fog_sel17.ini など)が保存されない。
LCDパネルカスタマイズツール(LCDUtil17)	・ 機種名が起動引数として渡された場合、雛形のドットマトリクスを作成する機能を追加
その他	・ 周辺 I/O シミュレータ(ESSIM) ― クロック計測値バグ修正 ― S1C17705 シミュレータの修正

ツール名	Ver 1.5.0 2009/2/20
統合開発環境(IDE)	・－O3 オプションに対応 ・新漢字フィルタに対応 ・不具合(IDE-05 参照)の修正
C コンパイラ(gcc)	・－O3 オプションに対応 ・double 型 / long long 型の処理の高速化 ・多次元配列の処理を高速化 ・漢字フィルタ処理をプリプロセッサ/コンパイラに実装 この実装により、漢字フィルタ処理が有効な場合(-mno-sjis-filt オプションが指定されていない)は、シングルクォテーション(‘) で全角文字 1 文字を囲っているコードの出力結果がVer1.5.0 より前のバージョンと変わります。 詳細は、コンパイラパッケージマニュアルの「Shift JIS コードのフィルタ機能」を参照して下さい。 また、この実装により「GNU17 C コンパイラ既知の問題」の No.3 は、解決されました。 ・漢字フィルタ処理を無効にする -mno-sjis-filt オプションを追加
アセンブラ(as)	―
リンカ(ld)	―
ライブラリ	・エミュレーションライブラリ(libgcc.a / libgccM.a / libgccMD.a) を高速化 ・ANSI ライブラリ(libc.a) の pow() 関数を高速化 ・ANSI ライブラリ(libc.a) から中身が空のダミー関数を全て削除
デバッガ(gdb)	・コマンド “commands”の追加 ・メモリウィンドウの Address 項目のデフォルト値の変更を可能にしました。 ・不具合(GDB-07 参照)の修正
マスク ROM 作成ツール(Winfog/Winmdc)	―
LCDパネルカスタマイズツール(LCDUtil17)	・ビットマップファイルから取り込めるセグメント数の上限値を引き上げ
その他	・不具合(Kanji-02 参照)の修正 ・サンプルプログラムの VECTOR の定義の不具合を修正 ・moto2ff.exe を更新(チェックサイズ引数追加)

ツール名	Ver 1.4.0 2009/1/6
統合開発環境(IDE)	・IDE のベースを Eclipse 3.4/CDT5.0/JavaVM5.0 へ更新 ・プロジェクト新規作成・インポート時、.cdtproject ファイルが.cproject ファイルに置き換わります ・新規機種(S1C17705)に対応 ・mak ファイルで objcopy 起動時に引数に-I elf32-little を追加 ・elf ファイルのコンテキストメニューから Object file conversion を行ったときの objcopy の引数に-I elf32-little を追加 ・不具合(IDE-04 参照)の修正
C コンパイラ(gcc)	―
アセンブラ(as)	―
リンカ(ld)	―
ライブラリ	―
デバッガ(gdb)	・シュミレータモードにプロファイラ/カバレッジ機能を追加 ・c17 stdout コマンドでファイル名を指定した場合、simulatedIO ウィンドウにも表示するよう変更。 ・周辺 I/O シュミレータ(Essim17)に対応機種追加 (S1C17705) ・COM/SEG 未設定状態の LCD ファイルを読み込んだ場合、ES-Sim17 が異常終了する問題を修正 ・不具合(GDB-06 参照)の修正
マスク ROM 作成ツール (Winfog/Winmdc)	・新機種(S1C17705)に対応
その他	・上記ツール全て、Windows Vista 対応 ・以下のツールを cygwin-1.5.25 ベースに更新 cygwin1.dll/cygiconv-2.dll/cygintl-3.dll/cygintl-8.dll/ar.exe/cp.exe/make.exe/ objcopy.exe/rm.exe/sed.exe/sh.exe ・S1C17704 用 fls17 モジュールを更新(¥mcu_model¥17704¥fls¥fls17704.elf)

ツール名	Ver 1.3.0 2008/9/4
統合開発環境(IDE)	・ 乗除算コプロライブラリ使用に対応 ・ コプロ用ベクタチェッカ起動・設定画面追加 ・ ビルド正常終了時、コンソールに終了メッセージを表示 ・ GNU17v1.2.0 以前に作成されたプロジェクトをインポートしたとき、コンパイル時の最終生成物を elf に設定 ・ 新規プロジェクト作成時のパラメータファイル設定値変更 ・ Navigator ビューに*.dump/*.sa/*.saf/*.out ファイルのフィルタを追加
C コンパイラ(gcc)	―
アセンブラ(as)	・2PASS MAKE の高速化
リンカ(ld)	・メモリ配置のオーバーラップチェックの強化
ライブラリ	・乗除算コプロ命令に対応したライブラリ(libgccMD.a)の追加
デバッガ(gdb)	・Watch Window に登録したシンボルの保存・再設定機能の追加 ・コアシュミレータに除算コプロ命令を追加 ・周辺 I/O シュミレータ(Essim17)で S1C17702、S1C17602 より、PSR を読み出せるレジスタが用意されたので、この機能を実装 ・不具合(GDB-05 参照)の修正
マスク ROM 作成ツール (Winfog/Winmdc)	―
その他	・ サンプルプログラムのブート部分の最適化 ・ コプロ用ベクタチェッカ(vecChecker.exe)をパッケージに追加

ツール名	Ver 1.2.1 2008/6/30
統合開発環境(IDE)	・Linked Resources に対応 ・プロジェクトの GNU17 Build Options 内で、コンパイル時の最終生成物(elf か psa か)選択機能追加 ・新規機種(S1C17003)に対応 ・不具合(IDE-03 参照)の修正
C コンパイラ(gcc)	・不具合(GCC-01 を参照)の修正 また、GCC-01 の<内容-1> の修正の結果、構造体及び共用体のサイズは必ず偶数バイト になるように調整されます。奇数バイトにならないように、最後に 1 バイト分の 未使用領域が 追加される場合がありますので、注意して下さい。
アセンブラ(as)	―
リンカ(ld)	―
ライブラリ	・ANSI C ライブラリのヘッダのプロトタイプ宣言をANSIーC準拠に変更。 ・不具合(LIB-02 を参照)の修正
デバッガ(gdb)	・ハードウェアブレークの設定可能本数を機種により最大4本まで対応。 ・ブレークポイントの保存、再設定機能の追加 ・Local/Watch Window でバイナリ形式で表示するとき、シンボルサイズ分表示するように変更。 ・不具合(GDB-04参照)の修正 ・周辺 I/O シュミレータ(Essim17)に対応機種追加 (S1C17003)
マスク ROM 作成ツール (Winfog/Winmdc)	・新規機種(S1C17003)に対応
その他	・PSR リード/ライト用のサンプルを削除 ・S1C17602 の内臓フラッシュ書き込みプログラムの差し替え(mcu_model¥17602¥fls¥fls17602.elf) フラッシュ消去が出来ないことがあるのを修正 ・漢字フィルターで/cygdrive/パス形式のソースファイルを認識するように修正

ツール名	Ver 1.2.0 2008/4/28
統合開発環境(IDE)	・-Wall、-O0 オプションに対応 ・新規機種(17001・17002・17501・17602・17702・17704・17801)に対応 ・プロジェクト生成時に essim17_user.ini ファイルを生成 ・LcdUtil17 の起動ボタン追加 ・マスク ROM 作成ツール(Winfog・Winmdc)の起動ボタン追加 ・Make ファイルに、elf ファイルから psa ファイル(S2 レコードファイル)を生成するコマンドを追加 ・デバッガ用コマンドファイルに psa ファイル(S2 レコードファイル)をロードするコマンドを追加 ・デバッガ用コマンドファイルにフラッシュ書込コマンドを追加 ・プロジェクト生成時に、コプロ対応ライブラリをリンクする設定を追加 ・プロジェクトプロパティからの CPU 機種変更機能の削除

C コンパイラ(gcc)	・以下の最適化をおこなうことにより実行速度を高速化
	(1)ループ処理の最適化
	ループ処理(for 文、while 文など) の最適化を行っています。
	これにより、ループ処理でのコード生成が変更される可能性があります。
	(2)ディレイド分岐命令に対応
	ディレイド分岐命令(call.d / jpr.d など) を生成するようになりました。
	ディレイド分岐命令は、通常の分岐命令よりも少ないサイクル数で実行できます。
	(3)不要な cmp 0 命令を削除
	以下のパターンでは、cmp %rd, 0 命令は削除可能な為、削除されます。
	パターン 1)
	and %rd, %rs/sign7
	cmp %rd, 0
	jreq / jrne / jrgt / jrge / jrlt / jrle
	パターン 2)
	add %rd, %rs/sign7
	cmp %rd, 0
	jreq / jrne
	・－O0 オプション(最適化をしない)に対応。
	・－Wall オプション(全ての警告を出力する)に対応。

アセンブラ(as)	―
リンカ(ld)	・-mpointer16 および-mpointer24 で生成されたオブジェクトファイルを混在してリンクしたときの エラーメッセージを修正
ライブラリ	・乗算コプロ命令に対応したライブラリ(libgccM.a)の追加 ・浮動小数点の比較、符号反転、加算、64bit 整数乗算、16bit シフト演算の高速化 ・ANSI C ライブラリの不具合(LIB-01 参照)の修正
デバッガ(gdb)	・c17 hbreakmd コマンド追加 ・set output-radix コマンド追加 ・Cソースコードの asm(“brk”); 対応 ・コアシュミレータに積和算コプロ命令を追加 ・周辺 I/O シュミレータ(Essim17)の追加。 対応機種は,001,602,701,702,704 ・不具合(GDB-03 参照)の修正
マスク ROM 作成ツール (Winfog/Winmdc)	・新規機種(17001・17002・17501・17602・17702・17704・17801)に対応
その他	・LcdUtil17.exe(LCD ファイル作成ユーティリティ)追加 ・sconv32.exe の出力メッセージを抑止 ・moto2ff で、指定したアドレスの範囲外にプログラムが含まれている場合、エラー ・PSR リード/ライト用のサンプルを追加

	・コプロライブラリ(libgccM.a) を使用したサンプルを追加

ツール名	Ver 1.1.5　2008/3/27	Ver 1.1.4　2008/3/17
統合開発環境(IDE)	—	・不具合(IDE-02 参照)の修正
C コンパイラ(gcc)	—	—
アセンブラ(as)	—	—
リンカ(ld)	—	—
ライブラリ	—	—
デバッガ(gdb)	—	—
マスク ROM 作成ツール	—	—
その他	・漢字フィルタ(日本語対応ツール)の不具合(Kanji-01 参照)の修正	—

ツール名	Ver 1.1.2　2007/11/29	Ver 1.1.1 2007/10/24	Ver 1.1.0 2007/9/29
統合開発環境(IDE)	—	—	・S1C17701 用パラメータファイルの wait 設定変更 ・不具合(IDE-01 参照)の修正
C コンパイラ(gcc)	—	—	—
アセンブラ(as)	—	—	—
リンカ(ld)	—	—	—
ライブラリ	—	・ANSI ライブラリの説明文(¥utility¥lib_src¥ansilib¥doc)の修正	—
デバッガ(gdb)	・不具合(GDB-02 参照)の修正	—	・Flash ROM ロード速度の向上 ・Essim レジスタ初期値、SVD 比較レベル変更 ・不具合(GDB-01 参照)の修正
マスク ROM 作成ツール	—	—	・新規追加
その他	—	—	・漢字フィルタの一行の文字数が 512 文字から無制限へ変更 ・S1C17701 用フラッシュロード/消去プログラムを修正

不具合内容のご説明

項番	不具合内容

GCC-01	<内容-1>
	構造体の配列の要素を値渡しで関数コールするプログラムをコンパイルすると、アドレス不整割り込みが発生するコードが生成されます。
	（意図しないアドレス不整割り込みが発生します。）
	
	<発生条件>
	以下、全ての条件を満たすケースで不具合が発生します。
	
	・構造体を配列で宣言していること。
	・構造体の配列を引数として値渡ししていること。
	・構造体の型のサイズは 7 バイト以下であること。
	・構造体の型のサイズは 奇数であること。
	・構造体のメンバ変数の型は unsigned char / char のみであること。
	
	サンプルコード：
	`// 構造体の型のサイズが 7 バイトかつ、奇数バイトかつ、unsigned char / char のみ`
	`// で定義されています。`
	`typedef struct s_tag {`
	`char m1 [3];`
	`char m2 ;`
	`unsigned char m3 ;`
	`}STR ;`
	
	`void sub( STR arg );`
	
	`int main( void )`
	`{`
	`// 構造体を配列で宣言しています。`
	`STR s[2] = { { 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0 } } ;`
	`int i ;`
	
	`for( i = 0 ; i < 2 ; i++ ) sub( s[i] ) ;// 構造体の配列を値渡ししています。`
	
	・・・次ページへ続く

	・・・前ページより <回避方法> 上記の条件をどれか一つでも満たさないことにより回避できます。 対策例: ・構造体の型のサイズを偶数にすること。(char 型のメンバ変数を構造体に追加) ・構造体の型のサイズが 8 バイト以上になるようにすること。 ・構造体をアドレス渡し(&) でサブルーチンに渡すこと。

<内容-2>
グローバル配列の添え字内で即値を用いた演算を行った時、正しいアドレスにアクセス
できません。

<発生条件>
以下、全ての条件を満たすケースで不具合が発生します。

・SMALL モデル(-mpointer16)でコンパイルすること。
・配列の添え字に即値と変数の演算があること。
・配列はグローバル配列であること。
・配列の型は以下であること。
  unsigned char / unsigned short / unsigned int / unsigned long / char / short
  int / long / float / enum / 構造体 / 共用体

サンプルコード:

```
unsigned int INPUT[20];
float f_Val;

void main(void)
{
    int i, y;

    for(i = 0; i <2; i++) {
        f_Val = INPUT[y + 16 - i]; // 配列 INPUT[] に正しくアクセスできません。
    };
```

<回避方法>
SMALL モデル(-mpointer16)で配列を使用する場合には、添え字の中で即値を含む演算を
しないようにします。
演算が必要な場合は、一度グローバルのワーク変数に演算結果を代入してから、それを
添え字内で使用することにより回避できます。

項番	不具合内容
IDE-01	IDE からのビルド時、リンカで out of range エラーが発生します。 2pass ビルド後のオブジェクトファイル(最適済)を 1pass 目のビルドで使用していたため、out of range が発生しておりました。 IDE は、1pass 目ビルドで生成されたオブジェクトファイル(未最適化)を、2pass ビルド後に復元するコマンドを make ファイルに出力します。 これにより、out of range は発生せず、正常にビルドできるようになります。
IDE-02	ライブラリ内(libgcc.a)から別ライブラリ内(libc.a)のシンボルを参照しているとき、 リンカスクリプトファイルで指定したマッピングの通りにリンクされず、リンカでオーバーラップエラーが発生します。オーバーラップエラーを回避するため、 IDE は、セクションの終了ラベル(_END_text など)をで、セクション定義の中括弧の外に定義するリンカスクリプトファイルを出力します。
IDE-03	S1C17702 のプロジェクト設定ファイルで、RAM および STACK サイズが違っているのを修正しました。 (誤)0x000000-0x001FBF(8KB) (正)0x000000-0x002FBF(12KB)
	C/C++ Project、Navigator、からプロジェクトのコピー＆ペースト後にリネームしたとき、以下の設定が変更されない不具合を修正。 ・ GNU17 Build Options 内のリンカオプションのリンカスクリプトファイル名が変更されない。(-T new_project_gnu17IDE.lds)。 ・ External Tools の"GDB17 launch for new_project"が作成されない。 なお、プロジェクトのコピー&ペーストでは設定は変更されません。ペースト後にプロジェクトのリネームを行ってください。
IDE-04	S1C17701 で 57 文字以上のプロジェクトファイルを作成し、Pack 処理を実行した場合、IDE が異常終了する不具合を修正。
IDE-05	Windows2000、XP で作成したプロジェクトを Vista の GNU17 にインポートしてビルドすると、cp コマンドでエラーが発生してビルドが完了しない不具合を修正。 mak ファイル内で以下のように cp.exe を使用している箇所を、copy を使用するように変更。 旧) for NAME in $(OBJS) ; do ¥ $(CP) -pf $$NAME obj1pass/$$NAME ; done ¥ 新) for NAME in $(subst /,¥¥,$(OBJS)) ; do ¥ cmd /c "copy /y $$NAME obj1pass¥¥$$NAME" >nul ; done ¥ これにより、Windows Vista 上でビルドが終了します。

項番	不具合内容	GDB コネクトﾓｰﾄﾞ
GDB-01	フラッシュメモリへ書き込み後、ソフトPCブレークを設定すると、デバッガが操作不能になることがあります。	ICDモード
	プログラムの RUN 中にメモリウィンドウをマウスクリックすると、「CPU is running」を表示しデバッガが操作不能になることがあります。	ICDモード
	unsigned long の配列の表示(print コマンド とウォッチウィンドウ)が 16bit サイズの表示になってしまいます。	シュミレータモード/ICDモード
	プログラムの RUN 中にお使いのPCのCPU使用率が高くなります。	ICDモード
	RAM に転送したプログラムにソフトブレーク設定をして、GOしてブレーク後、設定したアドレスの命令が 0xAAAA になってしまいます。	シュミレータモード
	enum タイプのシンボルのサイズを 16bit でなく、32bit 扱いで表示してしまいます。	シュミレータモード/ICDモード
GDB-02	シュミレーテッド I/O で RUN 中、ソースウィンドウにマウスを移動もしくはクリック操作すると"CPU is running"表示となるか、デバッガの操作不能になることがあります。	ICDモード
	Step コマンド実行後、ツールバーが無効のままになることがあります。	シュミレータモード
	PCレジスタ値がソフトウェアPCブレークポイントと同じのとき、プログラムを continue コマンドで実行しても実行せずに停止してしまうことがあります。	ICDモード
	PC レジスタ値と until コマンドで指定したアドレスの間にハードウェアブレークが設定されているとき、until コマンドを実行しても途中のハードウェアブレークポイントで停止しません。	ICDモード
	コマンドファイル(*.cmd)実行後、マウスカーソルが砂時計のままになりデバッガが操作不能になることがあります。	シュミレータモード/ICDモード
	ソースウィンドウが MIXED 表示の時、コンソールウィンドウから“set $pc=xxx”コマンﾄﾞを実行したとき、バックカラーが緑の位置がずれます。	ICDモード
GDB-03	OSC1 など低速クロックのとき、メモリウィンドウを大きめに開くと、タイムアウトエラーになることがあります。	ICDモード
	ソースウィンドウの漢字文字列を選択すると、gdb が落ちる。 選択した漢字をペーストしても落ちることがあります。	シュミレータモード/ICDモード
	gdb を起動直後、Continue を実行し、ソースウィンドウにカーソルを持っていくと、"CPU is running"が出て、デバッガ操作不能になることがあります。(再現性は低いです。)	ICDモード
	STOPボタンによる強制ブレーク後、メモリウィンドウから入力しても書き込みが行われません。	シュミレータモード

	PC値＝ソフトブレーク位置のときで、直後にハードブレークが設定されているとき実行すると、ハードブレークで停止しないときがあります。 例： int main() { ←① char str[256]; str[0] = 0; ←② ここで停止しません。 int p=0; ←③ break ① hbreak ② break ③ continue すると、①で停止後、continue しても②で停止せず、 ③で停止します。	ICDモード

	C ソース上でブレークを貼れない行にコンソールからブレークを貼ると、エラーにならず、同一アドレスに設定できてしまいます。
	例：
	while(p<10)
	{ ←ブレークの貼れない行①(″-″が無い)
	p++; ←②
	sub2();
	
	①に break コマンドでブレークを貼る。
	次に②にソースウィンドウ上でマウスでブレークを貼る。
	結果、info breakpoint コマンドでブレーク情報を確認すると、
	同じアドレスにブレークが貼られてしまっています。
		シュミレータモード/ICDモード

	テンポラリブレークポイントを設定し、その先のアドレスまで step,finish や Until コマンドを実行させたとき、テンポラリブレークポイントで停止しますがその設定が消去されずに残ったままになります。	シュミレータモード/ICDモード

	C ソースで、関数内からの finish→step にて、step 先が同じ関数の場合の飛び先位置が不正になることがあります。
	例：
	int main()
	{
	char str[256];
	str[0] = 0;
	int p=0;
	write_str("*** Test gdb simulated IO ***¥n"); ←①
	write_str("Please enter any string and <CR>¥n"); ←②
	▪
	▪
	
	void write_str(char *str)
	{
	int i;
	char *c;
	
	i = 0; ←③
	c = str; ←④
	while (*c != 0) { /* Check string length */
	▪
	▪
	}①から step 実行します。PC値は、write_str()関数の③で停止します。
	次に、finish を実行すると②に行きます。
	ここで step 実行すると、今度は、write_str()関数の③で停止せずに④で停止してしまいます。
		シュミレータモード/ICDモード

	関数コール行に Until でジャンプし、next を実行したときの結果(停止位置)が、
	デバッガ起動直後とソフトウェアリセット後で異なってしまいます。
	int main()
	{
	char str[256];
	str[0] = 0;
	int p=0;
	write_str("*** Test gdb simulated IO ***¥n"); ←①
	write_str("Please enter any string and <CR>¥n"); ←②
	write_str("Egggggg¥n"); ←③
	
	until ①
	next
	ここでPC値は、②になります。
	再び、
	c17 rst (リセットにより PC 値が変わります。)
	until ①
	next
	ここでPC値は、③になってしまいます。
		シュミレータモード

	アセンブラソースで ld 命令、call命令と続いている場合、回数付き next コマンドを 実行すると、step 動作になってしまいます。 例: ld %r0,%r1 ←① call func nop ←③ ・ func: nop ←② ①にPC値があるとき、next 2 を実行すると、PC値が ③ではなく、②になってしまいます。	シュミレータモード /ICDモード

	アセンブラソースのサブルーチンコール命令("call")行にソフトブレークまたは、ハードブレークが設定されている状態で、その箇所から finish を実行すると、break point の次の行で停止してしまう。 例: boot: xld.a %sp, 0xfc0 xcall init_lib ←①ここにブレーク設定 nop ←② boot から実行して、①で停止します。 次に、Finish を実行すると②で停止してしまいます。	ICDモード
	ツールバーのリセットボタンを実行すると、ソースウィンドウ上ではマウスポインタが砂時計になってしまいます。	ICDモード
	コンパイラが 24 ビットポインタモードのとき、 void 型ポインタ配列を print コマンドで正しい値を表示しません。 例: void *pData[10]; pData[1] = 0x123456; //ポインタを設定 print /x pData[1] $1 = 0x3456aa ←期待値は、0x123456 ですが、上位 8 ビットが正しくありません。 上位 8bit が正しくないのは、Watch ウィンドウ、Local ウィンドウで表示した場合も同様です。	シュミレータモード/ICDモード
	シミュレーテッド入力(simulated Input)でキー入力時、ツールバーの STOP ボタン以外も有効になってしまいます。	ICDモード
	Breakpoints ウィンドウからブレークポイントを無効に設定した場合、Source ウィンドウ上のブレークポイントも無効(黒)にならないことがあります。	シュミレータモード/ICDモード
	同じアドレスにテンポラリハードウェアブレーク設定を2回行い、エラーになったあと、プログラムを実行後、テンポラリハードウェアブレークが解除されたのにも拘わらず、再度テンポラリハードウェアブレーク設定を行うと、無効(黒)設定になってしまいます。	シュミレータモード/ICDモード

	print コマンドでポインタ値を表示した場合、ポインタアドレスの上位 8bit に無効な値が付加されてエラーになることがあります。 例： l_pcDataのポインタアドレスが 0x2cc0 のとき、上位 8bit の"0xa5"は誤り。 (GDB)print *l_pcData" Cannot access memory at address 0xa5002cc0"	ICDモード

	while 文中の最終行にある関数コール文を finish で while 文の先頭行まで実行しません。
	例：
	<C ソース、アセンブラMIX表示>
	.L5:
	while(p<10)
	{
	p++;
	add %r4,0x1 ←①
	sub2();
	xcall sub2
	sub3();
	xcall sub3
	sub4();
	xcall sub4
	cmp %r4,0x9 ←②
	jrle .L5
	}
	
	sub4()関数内で finish コマンド実行後、①ではなく、②で停止してしまいます。
		シュミレータモード/ICDモード

	パラメータファイルにより設定された領域以外にブレークポイントを設定してもエラーになりません。	シュミレータモード
GDB-04	ignore コマンドで指定したブレーク番号以外のブレークポイントを通過しても停止しません。	シュミレータモード
	フラッシュ ROM へ PSA ファイルのロード(書き込み)が正常に行われないことがあります。	ICDモード
	ソースウィンドウに表示されている漢字をマウスでクリックすると GDB がフリーズし、終了してしまうことがあります。	シュミレータモード/ICDモード
	ソースウィンドウに漢字(シフトJISコード)のうちEUCコードと一致するものが空白になってしまいます。	シュミレータモード/ICDモード

	以下のような while 文を STEP 実行すると、STOP ボタンによる強制ブレークが出来なくなることがあります。 例： while (*(unsigned long*)0x400 == 0){ }	シュミレータモード
GDB-05	ソースウィンドウのテンポラリブレーク設定の右クリックにより表示されるメニューに[Disable breakpoint]が無い。	シュミレータモード/ICDモード
	フラッシュメモリへロード時、ベリファイエラーが発生してもエラー表示して停止しない。	ICDモード
	パラメーラファイルで設定したスタック領域が複数あるとき、最初の1つしか有効にならない。	シュミレータモード
	コマンドファイルで continue コマンドを実行したとき、マウスカーソルが砂時計にならない。また、ICD モードでは STOP ボタンをクリックしても停止せず、GDB がフリーズする。	シュミレータモード/ICDモード
	コアシュミレータで、未定義の命令コードを実行したときエラーにならない。	シュミレータモード
	Watch/Local Window でレジスタ割り当ての long 型のシンボルの値が下位 16bit しか表示されない。	シュミレータモード/ICDモード
GDB-06	until コマンドで存在しない行番号を指定したとき、[Source]ウィンドウのメニューバーとツールバーのボタンが無効のままになってしまう。	ICDモード
	Source Window と Simulated I/O Window の改行箇所に■が表示されることがある。	シュミレータモード/ICDモード
	Simulated I/O Window で getc 関数などでファイル入力するとき、ファイルの改行が CRLF の場合、2文字として入力してしまう。	シュミレータモード/ICDモード
	Set コマンドと x コマンドでアドレスが 0xffffff を超えてもエラーにならない。	シュミレータモード/ICDモード
	Memory Window のセルに表示指定サイズを超えたデータを入力したとき、指定サイズ分の値が入力されない。	シュミレータモード/ICDモード
	Until コマンドでブレーク後、Local Window に volatile long 型変数の値が空白になる。	シュミレータモード/ICDモード
	Watch ウィンドウへ 1024 バイト以上の構造体を表示したとき、メンバ変数の値が正しくないときがある。	ICDモード
	Watch/Local ウィンドウの変数を編集した状態で SourcePreferences の OK/Apply を押下すると、登録していた変数が表示されなくなる。	シュミレータモード/ICDモード
GDB-07	Watch ウィンドウに大きなサイズ(1024Byte 以上)の構造体を表示したとき、 メンバ変数の値が正しくないことがある。	ICDモード
	RUN中、開放されないリソースハンドルが増えて、長時間RUNしているとリソースリークを発生することがある。	ICDモード
	カレントPCアドレスと until コマンドで指定したアドレスが一致するとき、プログラムを実行しない。	ICDモード
GDB-08	Watch ウィンドウにシンボルを登録した状態で、Console ウィンドウから x コマンドを実行すると、値とアドレスが正しく表示されない。	シュミレータモード/ICDモード

	load 命令で フラッシュメモリ へロードするときに、同じアドレスに2重に書き込みを行う。	ICDモード
GDB-09	Terminate and relunch ボタンの実行時にデバッガが2重起動となって正しく動かない。	シュミレータモード/ICDモード
	逆アセンブルウィンドウが開いているとき、エラーになることがある。	シュミレータモード/ICDモード

項番	不具合内容
LIB-01	ANSI ライブラリの関数のプロトタイプを一部修正しました。 <string.h>と<string.h>に属するCソース char *memcpy(/* char *, char *, int */) → void *memcpy(/* char *, char *, int */) ; char *memmove(/* char *, char *, int */) ;→ void *memmove(/* char *, char *, int */) ; char *memset(/* char *, int, int */) ; → void *memset(/* char *, int, int */) ;
	<stdio.h>と<stdio.h>に属するCソース int sscanf(char *, const char *, ...); → int sscanf(const char *, const char *, ...); int puts(const char *); → int puts(char *); int fputs(const char *, FILE *); → int fputs(char *, FILE *); <ctype.h>に属するCソース int isalnum(char c) ; → int isalnum(int c) ; int isalpha(char c) ; →int isalpha(int c) ; int iscntrl(char c) ; →int iscntrl(int c) ; int isdigit(char c) ; →int isdigit(int c) ; int isgraph(char c) ; →int isgraph(int c) ; int islower(char c) ; →int islower(int c) ; int isprint(char c) ; →int isprint(int c) ; int ispunct(char c) ; →int ispunct(int c) ; int isspace(char c) ; →int isspace(int c) ; int isupper(char c) ; →int isupper(int c) ; int isxdigit(char c) ; →int isxdigit(int c) ; int tolower(char c) ; →int tolower(int c) ; int toupper(char c) ; →int toupper(int c) ;
LIB-02	malloc()でヒープ領域の確保が正しく出来ないことがあります。
	ANSI ライブラリ の gmtime() 関数の定義部の型を修正しました。 struct tm *gmtime(time_t *t); →struct tm *gmtime(const time_t *t);

項番	不具合内容
Kanji-01	IDE で漢字フィルター機能が有効時にビルドしたとき、漢字フィルター処理が失敗するときがあります。
Kanji-02	IDE からビルド中にキャンセルしたとき、漢字フィルターの処理が中断されソースファイルが削除されることがあります。 このとき、ソースファイル(*.c)、フィルタされたファイル(*.kanji_filt)も残りません。

GNU17 C コンパイラ既知の問題

以下に GNU17 C コンパイラで認識されている不具合のケースを記載します。

No.1	**不具合の内容**
	巨大なサイズの配列(数十万バイト) が定義されているコードをコンパイルすると、 以下のコンパイルエラーが発生します。 cc1.exe: out of memory allocating mmmmmmmm bytes after a total of nnnnnnnn bytes
	対応方法
	配列やソースコードを分割するなどして、一度にコンパイラが確保するメモリ領域が小さくて 済むようにして下さい。
	再現コード
	unsigned char uc_array[] = { 0x00,0x01, ... }; int main() { ※配列のサイズは数十万バイト以上です。
	原因
	コンパイル時にコンパイラが確保しているメモリ領域が足りなくなるエラーです。 添え字なしの配列のサイズが大きすぎる為に発生しています。 同様のエラーは、ソースコードの行数の多いファイルで発生する場合もあります。
No.2	**不具合の内容**
	char 型の変数の符号拡張と、他の型との加減算処理が最適化により 一つにまとめられることにより、演算結果が正しい値になりません。 この不具合は、以下の条件を全て満たした時に発生します。 ・ 最初に 128(= 0x80)以上の値を、char 型より大きい型の変数に設定しておきます。 次のこの char 型より大きい型の変数を加減算した結果を、char 型の変数に代入します。 最後に、この char 型の変数を加減算した結果を char 型より大きい型の変数に代入します。 このとき、不具合が発生します。 ・ いずれかの代入の結果が、0 ～ 127 の範囲内である必要があります。
	対応方法
	最適化により符号拡張と加減算が一度に行われないようにする為に、char 型変数に volatile をつけて宣言することで回避して下さい。

No.2	**再現コード**
	signed int big_type_val ; int main(void) { 　　signed char char_val ; 　　big_type_val = 128 ; 　　char_val = big_type_val - 1 ; -- ① 　　big_type_val = char_val - 1 ; -- ②　　// big_type_val は 126 になるべきですが、 　　// -130 になります。
	原因
	最適化によるエラーです。 最適化により、① & ②の2行の処理が一つの処理としてコンパイルされます。 その為、符号拡張と演算が一度に行われ、正しい値になりません。
No.3	**不具合の内容**
	文字列化演算子のマクロ(#) で定義した漢字の文字列とダブルクォーテーションで囲んだ漢字の 文字列同士の比較が strcmp() で等しくなりません。 Kanji filter が有効な時にエラーになります。 ➔ Ver1.5.0 以降はこの問題は解決済みであり、エラーになることはありません。
	対応方法
	Kanji filter を無効にして下さい。 コマンドライン上でコンパイルする時は、メイクファイル(*.mak) 内の "CC_KFILT=xgcc_filt" を "CC_KFILT=xgcc" に変更して下さい。 IDE 上でコンパイルする時は、プロジェクトプロパティ内の Kanji filter 使用の項目を無効にして 下さい。
	再現コード

	#include <string.h>
	
	#define str(a) #a　　// 文字列化演算子のマクロ
	
	int main(void)
	{
	　　if(strcmp(str("字"),"¥"字¥"")) {　　// この比較結果は等しいべきですが、そうなりません。

No.3	原因
	Kanji filter は、コンパイル時にシフト JIS コードを ASCII コードに変換するツールでデフォルトで有効になっています。 文字列化演算子のマクロを使用した場合、コンパイル時には以下の順序で文字列が変換されていく為、等しい文字列になりません。 ソースコード　str(″字″)　″¥″字¥″″ ↓ Kanji filter による変換　str(″¥x8e¥x9a″)　″¥″¥x8e¥x9a¥″″ ↓ プリプロセッサ による変換　″¥″¥¥x8e¥¥x9a¥″″　″¥″¥x8e¥x9a¥″″
No.4	不具合の内容
	以下のコンパイルエラーが発生します。 error: unable to find a register to spill in class この不具合は、以下の条件を全て満たした時に発生する場合があります。 ・REGULAR モデルもしくは、MIDDLE モデルでコンパイルすること。 ・関数の引数渡しの時に、ポインタ引数が %r3 レジスタ経由で渡されること。 ・%r3 レジスタ経由で渡されたポインタ引数を、関数内で参照すること。 関数の引数渡しのレジスタ割り当てについては、コンパイラパッケージマニュアルの「6.4.3 レジスタ使用法」の「引数渡し用レジスタ(%r0～%r3)」の箇所を参照して下さい。
	対応方法
	エラーの原因となるポインタ引数が %r3 経由で引数渡しにされないようにします。 その為には、パラメータの順序を変更することや、ダミー引数を追加するという方法で実現できます。

No.4	**再現コード**

	void sub(int arg1, int arg2, int arg3, long *arg4)
	{
	　　static long long int num ;
	
	　　num = *arg4 ;
	}
	
	※このケースでは、ポインタ arg4 が %r3 経由で引数渡しされています。
	　以下のように、例えばダミー引数を追加し、arg4 を %r3 経由にならないように
	　することで、回避できます。
	
	void sub(int arg1, int arg2, int arg3, int dummy, long *arg4)
	{
	　　static long long int num ;
	
	　　num = *arg4 ;
	}

	原因
	コンパイラの処理中に必要なレジスタを確保できなかった為に発生している、 コンパイラの内部エラーです。

No.5	**不具合の内容**
	サブルーチン内でのグローバル変数への値の設定が、インライン展開により正しく行われません。 この不具合は、以下の条件を全て満たした時に発生します。 ・グローバル変数のアドレスをパラメータとしてサブルーチンに渡すこと。 ・サブルーチン内でパラメータであるポインタを経由して、グローバル変数に値を設定 すること。 ・サブルーチンがインライン展開されていること。 インライン展開される為には以下のように、いくつかの条件を満たす必要があります。 ➔ ・inline 宣言をつけて –O1 以上の最適化でコンパイル、もしくは -O3 で コンパイルすること ・サブルーチンの定義部が関数本体よりも前方にあること ・サブルーチンのサイズが小さいこと 実際にインライン展開されているかどうかは、逆アセンブルビュー上からサブルーチンが call 命令されていないことにより確認できます。
	対応方法
	対応方法①）グローバル変数(以下の例では g2) を volatile をつけて宣言します。 対応方法②）サブルーチンの定義部を関数本体よりも後方に配置するなどにより インライン展開されないようにします。
	再現コード

	`int g1, g2;`
	`int i_Val;`
	
	`void write_at ( int *addr, int off )`
	`{`
	`  addr[off] = 1000;  // addr[off] は g2 を指しています。`
	`}`
	
	`int main( void )`
	`{`
	`  g2 = 12;`
	`  write_at ( &g1, &g2 - &g1 );  // この関数がインライン展開されます。`
	`  i_Val = g2;  // i_Val は 1000 になるべきですが、12 になります。`

	原因
	最適化によるエラーです。

No.6	**不具合の内容**
	即値を関数ポインタにキャストしてから関数コールすると、以下のインターナルコンパイルエラーが発生します。 internal compiler error: Segmentation fault
	対応方法
	一度、即値を関数ポインタのグローバル変数に代入してから、グローバル変数を関数コールして下さい。
	再現コード

	typedef void *(*T)(void);
	
	void f(void)
	{
	((T) 10000000)(); // インターナルコンパイルエラーが発生します。
	}
	
	※このケースでは、以下のように関数ポインタのグローバル変数を関数コールすることで回避できます。
	
	typedef void *(*T)(void);
	T p_Pt; // 関数ポインタのグローバル変数
	
	void f(void)
	{
	p_Pt = (void *(*)(void))10000000;
	p_Pt();
	}

	原因
	即値から直接、関数コールする処理に不具合がある為です。

No.7	**不具合の内容**
	パラメータのアドレスを関数ポインタにキャストしてから関数コールすると、不正なアセンブラ命令が生成されます。
	対応方法
	一度、パラメータをグローバル変数に代入してから、グローバル変数のアドレスをキャストして関数コールして下さい。
	再現コード

	`void f( int x )`
	`{`
	`  ( *(void (*)())&x )();  // 不正なアセンブラ命令が生成されます。`
	`}`
	
	※このケースでは、以下のようにグローバル変数のアドレスをキャストして関数コールすることで回避できます。
	
	`int  ip_Pt;  // グローバル変数`
	
	`void f( int x )`
	`{`
	`  ip_Pt = x;`
	`  ( *(void (*)())&ip_Pt )();`
	`}`

	原因
	パラメータのアドレスから直接、関数コールする処理に不具合がある為です。

No.8	**不具合の内容**
	while() 文の条件式内でネストされているサブルーチンとローカル変数との演算が、インライン展開により正しく行われません。 この不具合は、以下の条件を全て満たした時に発生します。 ・-O1 よりも強い最適化(gnu17 でサポートしているのは –O3) でコンパイルすること ・while() 文の条件式内のサブルーチンがインライン展開されること ・9 個以上の関数でネストされていること ・while() 文の条件式内で演算されているローカル変数が、while() 文のブロックの中でも同様の演算をされていること
	対応方法
	while() 文の条件式内で演算されるローカル変数に volatile をつけて宣言して下さい。
	再現コード

	int f(int x)
	{
	  return (x + 1);
	}
	
	int main(void)
	{
	  int a = 1;
	
	  // while() 文の条件式内で、9 個の f() 関数がネストされ、かつローカル変数 a と演算を
	  // 行っています。
	  while ((f(f(f(f(f(f(f(f(f(1)))))))))) - a < 10){
	  a--; // 条件式内と同じ減算の処理を行っています。
	  exit (0); // exit(0) が実行されるべきですが、実行されません。
	  }
	  abort(); // while() 文のブロックの中に入らず、abort() が
	  // 実行されます。

	原因
	最適化によるエラーです。

GNU17 IDE 既知の問題

以下は、GNU17 IDE での既知の問題です。

No.1	**不具合の内容**
	CDT の Scanner Config Builder 実行中にビルドがハングする
	対応方法
	プロジェクトプロパティ->C/C++ Make Project->Discovery Options ->Enable generate scanner info command は OFF になっております。 この設定を ON にするとビルドできなくなる場合があります。
No.2	**不具合の内容**
	C/C++ Projects ビューのフィルタが正常動作しない
	対応方法
	フィルタの設定を変更した場合でも、プロジェクトを一旦閉じて開くか、 IDE を再起動しないと表示が反映されない場合があります。
No.3	**不具合の内容**
	Problems ビューの Description の表示が空になる
	対応方法
	ビルド後に Include ファイル内にワーニングがあるとき、 Problems ビューで Description の表示が空になり、 エラーのアイコンが表示されることがあります。 このときビルドは成功していますが、ワーニングを排除して再ビルドすることをお勧めします。
No.4	**不具合の内容**
	IDE 上のエディタで開いているファイルを外部のエディタで更新し、 File Changed のダイアログで No を選択したが、変更が反映される
	対応方法
	メニューの Window->Preferences->General->Workspace->Refresh automatically がデフォルトで ON のため、No と選択しても変更が反映されます。 IDE 上のエディタと外部のエディタで同時に同じファイルを編集しないようにしてください。
No.5	**不具合の内容**
	IDE で Window->Reset Perspective を選択すると、 Outline ビューでエラーが表示されることがあります。
	対応方法
	この場合は Outline ビューを閉じ、C/C++ Projects ビューから新しいファイルを ダブルクリックして IDE のエディタで開いたのち、 Window->Show View->Outline で再表示させてください。

No.6	不具合の内容
	C/C++Projects ビューでのアセンブラソースファイルのツリーの表示(ラベル等)が正しくありません。
	対応方法
	C/C++Projects ビューでは、アセンブラソースファイルのツリーの表示には対応しておりません。 表示の参照に関してはご注意ください。
No.7	不具合の内容
	ソースウィンドウの現在行の色が 2 行になることがあります。
	対応方法
	ソースウィンドウでソースファイルを開いているとき、現在行を示す色つきの行が 2 行になることがあります。 この場合、エディタで表示中のファイルを開きなおし、表示を更新してください。
No.8	不具合の内容
	エラーを修正してビルドしたのにソースの×印が消えないことがあります。
	対応方法
	ソースファイルにエラーがある状態でビルドを行うと、ソースの左側に、 エラー箇所を示す×印が表示されます。 ソースファイル修正後にビルドしても、この×印が消えない場合があります。 上記の場合には、Problems ウィンドウのコンテキストメニューから、 Delete C/C++ Markers を選択して×印を削除し、ビルドしなおしてください
No.9	不具合の内容
	Windows Vista 上で、[Delete Resources]ダイアログの [Delete project contents on disk]を選択してプロジェクトを削除しようとしたとき、 [An exception has been caught while processing the refactoring 'Delete Resource'] のエラーメッセージが表示され、プロジェクトフォルダが削除できないことがあります。
	対応方法
	プロジェクトをビルドした時、プロジェクトフォルダをカレントディレクトリとして conime.exe(コマンドプロンプトで日本語入力を可能にする)が起動し、ビルド後も終了しないため、プロジェクトフォルダが削除できなくなります。 この場合には、タスクマネージャなどから conime.exe のプロセスを終了させるか、 PC を再起動させてから、プロジェクトフォルダを削除してください。