Rev 1.10.00

## e2studio(GNU)ツールチェインの設定と必要事項の説明

# (ルネサス RZ/T1用)

e2studio(GNU)ツールチェインの設定方法とサンプルプロジェクトに必要な設定を説明します。

下記5通りのサンプルプロジェクトがありますが、tookhainの設定例は「RZT1\_Sample\_BARE」をもとに説明 を進めます。なお、他のサンプルの相違は、各サンプル用アプリケーションノートを参照して下さい。

サンプル名	用途	ワークスペース名	Ц	プロジェクト名
			ア	
RZT1_Sample_BARE	ベアメタル版サ	RZT1_Sample_BARE	M3	Sample_BARE_M3
	ンプル		R4F	Sample_BARE_R4F
RZT1_Sample_HWRTOS	M3側で	RZT1_Sample_HWRTOS	M3	Sample_HWRTOS_M3
	HWRIOS を使用 したサンプル		R4F	Sample_HWRTOS_R4F
RZT1_Sample_ECAT	EtherCAT 制御サ ンプル	RZT1_Sample_ECAT	M3	Sample_ECAT_M3.mot を 使用 (ソース非公開)
	• , , ,		R4F	Sample_ECAT_R4F
RZT1_Demo_BARE	ベアメタル版デ モソフト	RZT1_Demo_BARE	M3	Sample_ECAT_M3.mot を 使用 (ソース非公開)
			R4F	Demo_BARE_R4F
RZT1_Demo_NORTi	RTOS-NORTi 版 デモソフト	RZT1_Demo_NORTi	M3	Sample_ECAT_M3.mot を 使用 (ソース非公開)
			R4F	Demo_NORTi_R4F

注記

コア【M3】 側 EtherCAT®制御用サンプルのソース公開を希望される場合は、JSL Technology 社とのライセンス契約が必要です。

- 1. e2studio を起動する。
- 1-1. ワークスペースを選択する。

ワークスペースの選択         e2 studio は、ワークスペースと呼ばれるフォルダにプロジェクトを保存します。         このセッションに使用するワークスペース・フォルダを選択してください。         ワークスペース(W):         N:¥UsrAp¥C_H28_AICHI¥RZT1¥Sample_e2¥RZT1_Sample_BARE         ワークスペース(W):         ワークスペース名「RZT1_Sample_BARE」を選択         □ この選択をデフォルトとして使用し、今後この質問を表示しない(U)	 <sup>●</sup> ワークスペース・ランチャー	×
e2 studio は、ワークスペースと呼ばれるフォルダにプロジェクトを保存します。 このセッションに使用するワークスペース・フォルダを選択してください。 ワークスペース(W): N:¥UsrAp¥C_H28_AICHI¥RZT1¥Sample_e2¥RZT1_Sample_BARE ・ 参照(B) ワークスペース名「RZT1_Sample_BARE」を選択	ワークスペースの選択	
ワークスペース(W): N:¥UsrAp¥C_H28_AICHI¥RZT1¥Sample_e2¥RZT1_Sample_BARE	e2 studio は、ワークスペースと呼ばれるフォルダにプロジェクトを保存します。 このセッションに使用するワークスペース・フォルダを選択してください。	
□ この選択をデフォルトとして使用し、今後この質問を表示しない(U)	ワークスペース(W): N:¥UsrAp¥C_H28_AICHI¥RZT1¥Sample_e2¥RZT1_Sample_BARE ・ 参照(B) ワークスペース名「RZT1_Sample_BARE」を選択	
	□ この選択をデフォルトとして使用し、今後この質問を表示しない(U)	
	OK         キャンセル	





- 2. 各ツールの設定内容を確認する。
  - 2-1. コア【R4F】側の確認





- 2-1-1. リソース
  - 1) テキスト・ファイル・エンコード

e <sup>2</sup> プロパティ: Sample_BARE_R	!4F	- • •
フィルタ入力	リソース	↓ ↓ ↓ ↓
<ul> <li>リソース・フィルター リンクされたリソース</li> <li>▷ C/C++ ビルド</li> <li>▷ C/C++ 一般</li> <li>▷ タスク・リポジトリー ビルダー</li> <li>プロジェクト参照</li> <li>リファクタリング履歴 実行/デバッグ設定</li> </ul>	パス(P): /Sample_BARE_R4F タイプ(Y): プロジェクト ロケーション(L): N:¥USrAp¥C_H28_AICHI¥RZT1¥Sample_e2¥RZT1_Sample_BARE¥Sample 最終変更日時(M): 2017年4月21日 11:54:44 テキスト・ファイル・エンコード(T) ○コンテナーから継承(I) (SJIS) ● その他(o): SJIS ▼ 「SJIS」と直接 Key 入力 ■ 派生リソースのエンコードを別途保管(S) 新規テキスト・ファイルの行区切り文字(F) ● コンテナー (Windows) から継承(E) ● その他(H): Windows ▼	_BARE_R4F
	< III.	- F
?	ОК <b>+</b> +	ャンセル



-

#### 2-1-2. C/C++ビルド (TOP)

Ē

1) ビルダー設定 (デフォルト)

フィルタ入力	c/c++ ビルド
フィルタ入力         リソース         リソース・フィルター         リンクされたリソース         C/C++ ビルド         Device         Settings         Tool chain エディター         ツールチェーン・バー:         ビルド変数         ロギング         低存閣係スキャン         環境         ▷ C/C++ 一般	構成:     HardwareDebug [アクティブ]     ▼     構成の管理       構成:     HardwareDebug [アクティブ]     ▼       ビルダー設定     ●     振る舞い     ◆     ポリシーを更新       ビルダー     ビルダー      ●       ビルダー     ジー     ●     ボリシーを更新       ビルダー     ジー     ●     ●       ビルダー     ・     ●     ●       With ・     ビルド・コマンドを使用(U)     ●     ●       ビルド・コマンド(C):     make     ●       Makefile 生成     ●     ●       I 自動的に     Makefile を生成(G)     I Makefile に環境変数参照を展開(E)
ビルダー プロジェクト参照 リファクタリング履歴 実行/デバッグ設定	ビルド・ロケーション ビルド・ディレクトリー(D): \${workspace_loc:/Sample_BARE_R4F}/HardwareDebug ワークスペース ファイル・システム 変数
	デフォルトの復元(T) 適用(L)
?	OK キャンセル

2) 振る舞い (デフォルト)

	C/C++ ビルド		
▲ リソース			
リソース・フィルター リンクされたリソース	構成: HardwareDebug [アクティブ]		▼ 構成の管理.
▲ C/C++ ビルド			
Device			
Settings Tool chain エディク			
いールチェーン・バー・	ビルド設定		
ビルド変数	■ 最初のビルド・エラーで得止	<ul> <li></li></ul>	
ロギング		● Ose optimal jobs (4)	]
依存関係スキャン		○ 並列ショブを使用, 4	
環境			
C/C++ 一般	ワークベンチ・ビルドの振る舞い		
タスク・リポジトリー	ワークベンチ・ビルド・タイプ:	Make ビルド・ターゲット:	
ビルダー	□ リソース保管時にビルド (自動ビルド)	all	変数
プロジェクト参照 リファクタリング履歴	注: ワークベンチの自動ビルド設定を参照		
ラファクタラフク福祉 実行/デバッグ設定	☑ ビルド (インクリメンタル・ビルド)	all	変数
	▼ クリーン	clean	変数
b		デフォルトの復元(T)	適用(L)
	I		
?		ОК	キャンセル

3) ポリシーを更新 (デフォルト)

### 2-1-3. C/C++ビルト (Device)

フィルタ入力	Device	
▶ リソース		
▲ C/C++ ビルド	Current Device: R75910018	
Device		
Settings	Change Device: R7S910018	
Tool chain エディター		
ツールチェーン・バー:		
ビルド変数		
ロギング		
依存関係スキャン		
環境		
▷ C/C++ 一般		
▷ タスク・リポジトリー		
ビルダー		
プロジェクト参照		
リファクタリング履歴		
実行/デバッグ設定		
		デフォルトの復元(T) 適用(L)
	1	



## 2-1-4. C/C++ビルド (Settings)

2-1-4-1. Tool Settings (Library Generator)

● <sup>2</sup> プロパティ: Sample_BARE_R	4F	
フィルタ入力	Settings	(→ ▼ ⊂) ▼ ▼
▶ UV-Z	構成: HardwareDebug [アクティブ]	▼ 構成の管理 ^
<ul> <li>(-C++ ビルド) Device Settings Tool-ehan エディター ツールチェーン・バー: ビルド変数 ロギング 低存閣係スキャン 環境</li> <li>C/C++ 一般</li> <li>C/C++ 一般</li> <li>C/C++ 一般</li> <li>C/C++ 一般</li> <li>グスク・リボジトリー ビルダー プロジェクト参照 リファクタリング履歴 実行/デバッグ設定</li> </ul>	<ul> <li>♥ Tool Settings )  PULF・ステップ ♥ ビル</li> <li>● Library Generator</li> <li>○ スッダー・ファイル</li> <li>● その他のオプション</li> <li>● その他のオプション</li> <li>● その他の目中</li> <li>● S Assembler</li> <li>■ S Assembler</li> <li>■ Sunker</li> <li>コマンド行</li> <li>● Objoopy</li> <li>● 一般</li> </ul>	ド成果物 論 パイナリー・パーサー ③ エラー・パーサー         arm-none-eabi-libgen         /ション:
?		ОК =+7>72/L

1) 設定 (デフォルト)

члияд <u>л</u>	Settings	← → → →
<ul> <li>&gt;&gt; リノース</li> <li>(-/-ス)</li> <li>(-/-ス)</li> <li>(-/-ス)</li> <li>(-/-+</li> <lp>(-/-+ <lp>(-/</lp></lp></ul>	構成: HardwareDebug [アクティブ]	<ul> <li>構成の管理</li> <li>、</li> <li>、</li> <li>、</li> <li>、</li> </ul>
•	ок	キャンセル



2) ヘッダー・ファイル



3) その他のオプション (デフォルト)

. Sample_BARE_R4P		
Settings		▼ <
ビルド ビルド Ge ings chain エディター ルチェーン・バー: ・ ジョン Library Generator	ケティブ] ド・ステップ   😤 ビルド成果物   🗟 バイナリー・パーサー   🥹 エラー・パーサー 最適化のタイプ Code size optimization	構成の智
ド変数 ング 避底スキャン 一般 リボジトリー クト参照 (ッグ設定) (○ クロののオブシ (○ クロののオブシ (○ 名osembler (○ 登 Compiler ) (○ Assembler (○ 登 Linker (○ 登 Dip(cpy) (○ 一般	ユーザ定義のコンパイラー・オブション	<b>ඩ</b> ඬි ම ලි
	ユーザ定義のアセンブラー・オブション	<b>ඩ</b> ඩ බ රූ
	デフォルト	の復元(T) 適用(L)
		K ±17\/7/L

4) その他(1/2)

	۲۰ ۲۰ ۲۰
ス ビルド 構成: HardwareDebug [アクティ	ブ] ・ 構成の管
ings chain エディター 🛞 Tool Settings 🎤 ピルド・ス	テップ 🙅 ビルド成果物 🔛 バイナリー・パーサー 🔕 エラー・パーサー
ルチェーン・バー:	■ 関数呼び出しによる値の書き換えを有効にする (-fcaller-saves)
ト変数 👋 設定	■条件式における型の不一致を許可する (-fcond-mismatch)
シンク  過 ヘッダー・ファイル	<ul> <li>共通式の削除 (-fcse-follow-jumps)</li> </ul>
■ 🖓 その他のオプション	条件付きジャンプの後に共通式の削除を行う (-fcse-skip-blocks)
一般 その他	🔲 遅延分岐命令後のスロットを使用する (-fdelayed-branch)
レポジトリー ▷ See Compiler	🔲 いくつかの小規模な最適化を行う (-fexpensive-optimizations)
- Assembler	■ ANSI または IEEE ルールに違反してもスピードを優先する(-ffast-math)
クト参照 シ	□ 浮動小数点変数をレジスタに格納しない (-ffloat-store)
タリング履歴	☑ 各関数を、出力ファイル内の対応するセクションに配置する (-ffunction-sections)
グ設定	☑ 各データを、出力ファイル内の対応するセクションに配置する (-fdata-sections)
	<ul> <li>すべての単純な関数を呼び出し例に組み込む (-finiline-functions)</li> </ul>
	■実行時に呼び出し可能なパージョンの関数を別途出力する (-fkeep-inline-functions)
	── 各関数呼び出しに対して、関数のリターン直後に引数をポップする (-fno-defer-pop)
	☑ 関数のアドレスをレジスターに置かない (-fno-function-cse)
	🔄 'inline' キーワードを無視する (-fno-inline)
	□ マシン固有のビープホール最適化を禁止する (-fno-peephole)
	□ 不必要な関数フレーム・ポインターを保持しない (-fomit-frame-pointer)
	コンパイル中に arc をインストルメントする (-fprofile-arcs)
	□ ループ最適化の後に再び共通式の削除を行う (-frerun-cse-after-loop)
•	□ 実行が遅滞しないよう、命令の順番を入れ替える (-fschedule-insns)

### 5) その他 (2/2)



2-1-4-2. Tool Settings(Compiler)

ルタ入力	Settings		
リソース C/C++ ビルド Device Settings Tool chain エディター ツールチェーン・バー: ビルド変数 ロギング 依存閣係スキャン 環境 C/C++ 一般 タスク・リポジトリー	<ul> <li>Tool Settings PULK</li> <li>Ulbrary Generator</li> <li>Compiler</li> <li>フィス</li> <li>オブジェクト</li> <li>リスト</li> <li>ど 標準</li> <li>ご 標準</li> <li>ご 構準</li> <li>ご 構築</li> </ul>	ステップ 🔮 ビルド成果物 コマンド: すべてのオプション: エキスパート設定: コマンド行	
ビルダー プロジェクト参照 リファクタリング履歴 実行/デバッグ設定	▲ 200 その他	パターン:	
			OK         キャンセル

## 1) ソース

フィルタ入力 ▶ リソース	Settings 🗘 🗘	⇒ • •
<ul> <li>2 C/C++ ビルド Device Settings Tool chain エディター ツールチェーン・パー: ビルド変数 ロギング 依存聞係スキャン 環境</li> <li>▶ C/C++ 一般</li> <li>▶ 4スク・リポジトリー ビルダー プロジェクト参照 リファクタリング履歴 実行/デバッグ設定</li> </ul>	構成: HardwareDebug [アクティブ] ● Tool Settings  ● ビルド・ステップ  ● ビルド成果物  ● バイナリー・パーサー ● エラー・パーサー  ● S Library Generator ● S Compiler  ● Compiler  ● Compiler  ● Compiler  ● Compiler  ● S Co	<u>∲</u>
	マクロ定義 USED_DEFnano_=1 注*1 DEFnano使用/未使用の定義 「使用」」USED_DEFnano_=1	₽I
() () ()	 「 、 で 、 、 、 、 、 、 、 、 、 、 、 、 、 、 、 、 、	

### <mark>注\*1</mark>

「\_\_USED\_DEFnano\_=0」と使用しない側に定義しても内蔵 RAM へのダウンロードとシリアルフラッシュ ROM への書き込み操作は可能です。ただし、再操作する場合はターゲット側のリセット操作が必要 になります。

2) オブジェクト



## 3) リスト(デフォルト)



4) 警告 (デフォルト)

ルタ入力	Settings	$\leftarrow$ $\bullet$ $\Rightarrow$ $\Rightarrow$ $\bullet$
リソース C/C++ ビルド Device Settings Tool chain エディター ツールチェーン・パー: ビルド変数 ロギング 低存関係スキャン 環境 C/C++ 一般 タスク・リポジトリー ビルダー プロジェクト参照 リファクタリング履歴 実行/デパッグ設定	<ul> <li>▶ 後 Library Generator</li> <li>▲ ③ Compiler</li> <li>※ ソース</li> <li>※ オブジェクト</li> <li>※ ゴジェクト</li> <li>※ 国本</li> <li>※ 否の他</li> <li>※ その他</li> <li>※ その他</li> <li>※ Con</li> <li>※ Coult</li> <li>※ Coult</li></ul>	

5) 警告-標準 (デフォルト)



### 6) 警告-拡張 (デフォルト)



7) その他



8) その他-その他 (1/2)



8) その他-その他 (2/2)





# 9) CPU

e <sup>2</sup> フロパティ: Sample_BARE_R フィルタ入力 ▷ リソース ■ C/C++ ピルド Device Settings Tool chain エディター ツールチェーン・パー:	4F Settings	CPU タイプ アーキテクチャー エンディアン 命令セット	cortex-r4f armv7-r Little-endian
ビルド変数 ロギング 依存関係スキャン 環境 ▷ C/C++ 一般 ▷ タスク・リポジトリー ビルダー プロジェクト参照 リファクタリング履歴 実行/デバッグ設定	<ul> <li>▲ 図 言言</li> <li>※ 信仰</li> <li>※ 石 の他</li> <li>※ その他</li> <li>※ その他</li> <li>※ CPU</li> <li>※ Assembler</li> <li>※ Linker</li> <li>※ Objcopy</li> <li>※ 一般</li> </ul>	<ul> <li>インター・ワーキング (-mthumb)</li> <li>ARM Procedure Call Standard ()</li> <li>Thumb Procedure Call Standard ()</li> <li>leaf 関数に対するスタック・ブレー 浮動小数点 ABI (-mflost-abi= #ame)</li> <li>関数プロローグシスケシューリング</li> <li>図 Disable unargned word and halfw</li> <li>関数名をパブジェクトコードに格納 ターゲ・ト FPU (-mfpu)</li> </ul>	terwork) 接性のあるスタック・フレームを生成する と互換性のあるスタック・フレームを生成する ムを生成する Softfp で有効にする word accesses to packed data する Vrfp
⑦ ⑦ DEFnanoを使用する	11 5場合は、命令セット	「ARM」固定	0K キャンセル

2-1-4-3. Tool Settings(Assembler)

7ルタ入力 5	Settings			← → → →
リソース C/C++ ビルド Device Settings Tool chain エディター	<ul> <li></li></ul>	コマンド: すべてのオプション:	arm-none-eabi-as -I"N:¥UsrAp¥C_H28_AICHI ¥RZT1¥Sample_e2¥RZT1_Sample_BARE ¥Sample_BARE_R4F/src"defsym	
ッールチェーン・バー: ビルド変数 ロギング 依存関係スキャン 環境 C/C++ 一般 タスク・リポジトリー ビルダー プロジェクト参照 リファクタリング履歴	 ◎ スト ◎ その他 ▷ 勁 Linker ▲ 勁 Objcopy ◎ 一般	エキスパート設定: コマンド行 パターン:	\${COMMAND} \${FLAGS} \${OUTPUT_FLAG} \$	(OUTPUT_
実行/デバッグ設定 Ⅲ ▶			ОК <b>‡</b> т;	ンセル

## 1) ソース

7イルタ入力	Settings	
<ul> <li>リソース</li> <li>C/C++ ビルド Device</li> <li>Settings Tool chain エディター ツールチェーン・パー: ビルド変数 ロギング</li> </ul>	<ul> <li>               Eibrary Generator      </li> <li>             Compiler         </li> <li>             Assembler         </li> <li>             ダノース         </li> <li>             ガブジェクト         </li> <li>             ジリスト         </li> <li>             ぞの他         </li> <li>             We Linker         </li> </ul>	インクルード・ファイル・ディレクトリー 全 会 谷 公 小 *\${workspace_loc:/\${ProjName}}/src" "\${workspace_loc:/\${ProjName}}/src"
<ul> <li></li></ul>	ヤン A	注*1 DEFnano使用/未使用の定義 「使用 」_USED_DEFnano_=1 「未使用」_USED_DEFnano_=0 デバッグ環境によって設定する。
<mark>主*1</mark> 「_USED_DEFnar ノリアルフラッシュ デット側のリセット	o_=0」と使用しない側 ROM への書き込み操作 操作が必要になります。	ルに定義しても内蔵 RAM へのダウンロードと 乍は可能です。ただし、再操作する場合はター



2) オブジェクト

<ul> <li>&gt; リソース</li> <li>C/C++ ビルド</li> <li>Device</li> <li>Settings</li> <li>Tool chain エディター</li> <li>ツールチェーン・バー:</li> <li>ビルド変数</li> <li>ロギング</li> <li>低存開係スキャン</li> <li>環境</li> <li>C/C++ 一般</li> <li>タスク・リボジトリー</li> <li>ビルドタ</li> <li>マー般</li> <li>グラスク・リボジトリー</li> <li>ビルター</li> <li>ブロジェクト参照</li> <li>リファクタリング履歴</li> <li>実行/デバッグ投定</li> <li>エラー後もオブジェクト・ファイルを生成する (-2)</li> <li>エラー後もオブジェクト・ファイルを生成する (-2)</li> <li>エラー後もオブジェクト・ファイルを生成する (-2)</li> <li>エラー後もオブジェクト・クット・ファイルを生成する (-2)</li> <li>エラー後もオブジェクト・ファイルを生成する (-2)</li> <li>エラー後もオブジェクト・ファイルを生成する (-2)</li> <li>デオスト・セクションを結合する (-R)</li> <li>アノース</li> <li>ジレスト</li> <li>アールチーマット</li> <li>レドな数</li> <li>ロギング</li> <li>(アレチープ)</li> <li>ロギング</li> <li>マー般</li> <li>ロギング</li> <li>マー般</li> <li>ロギング</li> <li>マー般</li> <li>ロギング</li> <li>ロギング</li> <li>ロギング</li> <li>ロギング</li> <li>マー般</li> <li>ロギング</li> <li>ロギング</li> <li>ロギング</li> <li>ロギング</li> <li>マール</li> <li>マール</li> <li>マール</li> <li>マール</li> <li>ロギング</li> <li>ロギン</li> <li>ロギング</li> <li>ロギン</li> <li>ロギン</li> <li>ロギン</li> <li>ロギン</li> <li>ロギン</li> <li>ロ</li></ul>	リソース       C/C++ ビルド         Device       Settings         Tool chain エディター       ● ⑤ Compiler         ツールチェーン・バー:       ビルド変数         ロギング       ● ⑥ Linker         マの他       ● ⑥ Linker         ● ⑧ Linker       ○ ワレ タイブ         ● ⑧ Objcopy       □ キキクチャレ         ジルチ       ● ⑩ Objcopy         ビルド変数       □ ポング         成存開協系スキャン       - 般         ワレタイブ       Contex-r4f         ● ⑨ Objcopy       □ キキクチャー         ● ⑦ N InfishU-       ● ⑦ N InfishU-         ビルター       ● ⑦ N InfishU-         リファクタリング履歴       ● ⑦ N FIPU (-nmpu)         ● ⑦ N FIPU (-nmpu)       ● ⑦ N FIPU (-nmpu)         ● ⑦ N FIPU (-nmpu)       ● ⑦ N FIPU (-nmpu)         ● ⑦ N FIPU (-nmpu)       ● ⑦ N FIPU (-nmpu)	フィルタ入力	Settings		↓ ↓ ↓ ▼
	⑦ OK #ヤンセル	<ul> <li>&gt; リソース</li> <li>&gt; リソース</li> <li>&gt; C/C++ ビルド Device</li> <li>Settings</li> <li>Tool chain エディター ツールチェーン・バー: ビルド変数</li> <li>ロギング 依存関係スキャン 環境</li> <li>&gt; C/C++ 一般</li> <li>&gt; タスク・リポジトリー ビルダー</li> <li>プロジェクト参照</li> <li>リファクタリング履歴 実行/デバッグ設定</li> </ul>	<ul> <li>&gt; いにしていたいのでは、</li> <li>&gt; いたいのでは、</li> <l< td=""><td>オブジェクト・ディレクトリー 「エラー後もオブジェクト・ファイル 「テキスト・セクションとデータ・セ デバッグ・フォーマット エンディアン CPU タイプ アーキテクチャー 命令セット 「インター・ワーキング (-mthum) 浮動小数点 ABI (-mfloat-pti=nzme) ターゲット FPU (-pmpu)</td><td>\${CONFIGDIR} を生成する (-Z) クションを結合する (-R) Dwarf2 Little-endian cortex-r4f armv7-r ARM nterwork) Softfp vfp</td></l<></ul>	オブジェクト・ディレクトリー 「エラー後もオブジェクト・ファイル 「テキスト・セクションとデータ・セ デバッグ・フォーマット エンディアン CPU タイプ アーキテクチャー 命令セット 「インター・ワーキング (-mthum) 浮動小数点 ABI (-mfloat-pti=nzme) ターゲット FPU (-pmpu)	\${CONFIGDIR} を生成する (-Z) クションを結合する (-R) Dwarf2 Little-endian cortex-r4f armv7-r ARM nterwork) Softfp vfp

## 3) リスト(デフォルト)

	Settings 🗢	• => • •
<ul> <li>C/C++ ビルド         Device         Settings             Tool chain エディター             ツールチェーン・バー:             ビルド変数             ロギング             依存閣係スキャン             環境         C/C++一般          C/C++一般          タクク・リポジトリー         ビルダー         プロジェクト参照             リファクタリング履歴             実行/デパッグ設定     </li> </ul>	<ul> <li>構成: HardwareDebug [アクティブ]</li> <li>● Tool Settings</li></ul>	D管理
?	OK キャンセノ	L D

4) その他(1/2)

イルタ入力	Settings		
> リソース → C/C++ ビルド Device Settings Tool chain エディタ	構成: HardwareDebug [アクテ・ 酸 Tool Settings 🎤 ビルド・	ィブ] ステップ   🙅 ビルド成果物   🗟 バイナリー・ノ	▼ 構成の管理 パーサー Q エラー・パーサー
ツールチェーン・パ ビルド変数 ロギング 依存関係スキャン 環境 C/C++ 一般 タスク・リポジトリー ビルダー プロジェクト参照	<ul> <li></li></ul>	ユーザー定義オプション -g -g -g	· 원 원 원 장I 와I
?		ОК	キャンセル

5) その他(2/2)(デフォルト)

イルタ入力 Settings	
<ul> <li>&gt; リソース ▲</li> <li>C/C++ ビルド Device</li> <li>Settings</li> <li>Tool chain エディタ ツールチェーン・バ ビルド変数</li> <li>ロギング 依存関係スキャン 環境</li> <li>&gt; C/C++ 一般</li> <li>&gt; タスク・リポジトリー</li> <li>ビルダー</li> </ul>	<ul> <li>              ・             ・</li></ul>
フロジェクト参照 113-5-20-2015-2月家区 *	デフォルトの復元(T) 適用(L)

2-1-4-4. Tool Settings(Linker)

パルタ入力	Settings		<del>ب</del> ج	-
リソース C/C++ ビルド Device Settings Tool chain エディター ツールチェーン・バー: ビルド変数 ロギング	構成: HardwareDebug [アクテ ③ Tool Settings 🎤 ビルド・ ト 🛞 Library Generator	・イブ] ステップ   🙅 ビルド成!   コマンド:	◆ 構成の 果物 I 励 パイナリー・パーサー 🧿 エラー・パーサー arm-none-eabi-ld	·管理)
ロキシク 依存関係スキャン 環境 C/C++ 一般 タスク・リポジトリー	▶  Solve: Compiler ▶  Solve: Compiler ●  Sol	すべてのオプション:	-M=Sample_BARE_R4F.map @"N:/UsrAp/C_H28_AICHI/RZT1/Sample_e2/RZT Sample_BARE/Sample_BARE_R4F¥HardwareDebug	1_ [ g
ビルダー プロジェクト参照 リファクタリング履歴 実行/デバッグ設定	<ul> <li>              登 セクション</li></ul>	エキスパート設定: コマンド行 パターン:	\${COMMAND} \${OUTPUT_FLAG}\${OUTPUT_PREF	=IX} \$
•			ОК +т>ти	,

# 1)入力 (デフォルト)

フィルタ入力 > リソース	Settings		
<ul> <li> <ul> <li>C/C++ ビルド Device         </li> <li>Settings             Tool chain エディター ツールチェーン・バー: ビルド変数             ロギング             依存関係スキャン             環境         </li> <li>C/C++ 一般         </li> <li>タスク・リポジトリー ビルダー         </li> <li>ブロジェクト参照             リファクタリング履歴             実行/デバッグ設定         </li> </ul> </li> </ul>	構成: HardwareDebug [アクティブ]	ピルド成果物     「読 バイナリー・パーサー     カファイル     カファイル	<ul> <li>● 構成の管理</li> <li>● エラー・パーサー</li> <li>● 毛 宮 주! ♪!</li> </ul>
۰ ( m )		1 7 54 1 0	
?		ОК	キャンセル

\_

2) 出力 (デフォルト)

リソース C/C++ ビルド Device       構成:       HardwareDebug [アクティブ]       ・       構成の管理.         Settings       Tool chain エディター ツールチェーン・バー:       ・       構成の管理.         ジレド交数 ロギング 依存開係スキャン 環境       ・       ビルド・ステップ ・       ビルド成果物       ・       バイナリー・パーサー       ●       エラー・パーサー         レド交数 ロギング 依存開係スキャン 環境       ・       ・       ●       Ubrary Generator       ●       Ubrary Generator       ●       ●       Ubrary Generator       <	ィルタ入力	Settings	↓ ↓ ↓
····································	リソース C/C++ ビルド Device Settings Tool chain エディター ツールチェーン・パー: ビルド変数 ロギング 依存関係スキャン 環境 C/C++ 一般 タスク・リポジトリー ビルダー プロジェクト参照 リファクタリング履歴 実行/デバッグ設定	構成: HardwareDebug [アクティブ]	<ul> <li>▼ 構成の管理…</li> <li>・サー ③ エラー・パーサー</li> <li>追加のセクションを作成する:</li> <li>マンドを使用する</li> </ul>

3) セクション (設定なし)

タ入力	Settings	← < <>
ノース C++ ビルド Device Settings Tool chain エディター	構成: HardwareDebug [アクティブ]	<ul> <li>▼ 構成の管理.</li> </ul>
ツールチェーン・バー: ビルド変数	③ Tool Settings デビルド・ステップ 学ビルド成果物 品 バイナリー・パーサー @	エラー・パーサー
ロギング 依存関係スキャン 環境 C++ 一般 スク・リポジトリー レダー コジェクト参照 ファクタリング履歴 〒/デバッグ設定	<ul> <li>▶ S Compiler</li> <li>▶ S Assembler</li> <li>▲ Linker</li> <li>▲ Linker</li> <li>● 出力</li> <li>● 出力</li> <li>● セクション・ビューアー:</li> <li>セクション・ビューアー:</li> <li>レーン・ビューアー:</li> <li>レーン・ビューアー:</li> <li>レーン・ビューアー:</li> <li>レーン・ビューン・ビューアー:</li> <li>レーン・ビューアー:</li> <li>レーン・ビューン・ビューアー:</li> <li>セク地・ビューン・ビーン・ビューアー:</li> <li>セクション・ビューン・ビーン・ビューン・ビーン・ビーン・ビーン・ビーン・ビーン・ビーン・ビーン・ビーン・ビーン・ビ</li></ul>	<ul> <li>ウションの詳細:</li> <li>前: .flash_contents</li> <li>開始アドレス:</li> <li>回意アドレス</li> <li>前のセクションから続</li> <li>ラベル</li> </ul>
4	▶ 🖗 .eh_frame_hdr	◎ 変数

### <mark>注\*1</mark>

サンプルのセクション設定は、後記「その他」のLinker Script ファイルにて設定

4) その他(1/2)(デフォルト)

21/10%X01     Settings       > UV-ス     C(C++ ビルド       C/C++ ビルド     ●       Device     Settings       Tool chain エディター     ●       ツールチェーン・バー:     ●       ビルド交数     □       ロギング     係存間係スキャン       感見     ●       ③     Library Generator       ●     ③       ●     ③       ○     C++       ●     ③       ●     ③       ○     ○       ○     ○       ○     ○
⑦ OK キャンセル

5) その他(2/2)

e <sup>2</sup> プロパティ: Sample_BARE_R	4F		
フィルタ入力	Settings		(
▷ リソース ▲ C/C++ ビルド			
Device Settings Tool chain エディター		シンボル定義	🗐 🔊 🗟 취 灯
ツールチェーン・バー: ビルド変数			
<ul> <li>ロキンク</li> <li>依存関係スキャン</li> <li>環境</li> </ul>			
▷ C/C++ 一般 ▷ タスク・リポジトリー			
ビルダー プロジェクト参照 リファクタリング履歴		External Linker script(-T)	
実行/デバッグ設定			=
		コマンド・ファイルの上書き External Linker script(-T)	▼ 参昭(B)
۰ III. ۲			
(?)		ок	キャンセル
	"\$	{ProjDirPath}/script file/Locate.ld"	
	[	Locate.ld Linker Script File	

[Locate.ld]

```
MEMORY{
  ROM
            (rx) : ORIGIN = 0x00000000, LENGTH = 0x00060000
  RAM
           (rw) : ORIGIN = 0x00060000, LENGTH = 0x00018000
  STACK
            (rw) : ORIGIN = 0x00078000, LENGTH = 0x00008000
}
SECTIONS
{
  .text:{}
   = 0 \times 00000000;
   * (.fvectors)
                       /* asm */
   = 0 \times 00000100;
   *(.loader text)
   *(.text.text.*)
   * (.rodata.rodata.*)
 }>ROM
  .data:{
   __data_load = .;
     _data_start = LOADADDR(.data) + (___data_load - ADDR(.data));
   * (.data.data.*)
    __data_end = LOADADDR(.data) + (. - ADDR(.data));
 }>ROM
  .bss:{
    \_bss\_start = .;
   * (.bss.bss.*)
   *(COMMON)
    \_bss\_end = .;
   * (_ebss_end)
   end = .;
 }>RAM
  .stack:{
   .=ALIGN(0x10);
   _sys_stack_top = .;
   .+=0x0002000;
   _sys_stack = .;
   .=ALIGN(0x10);
   _svc_stack_top = .;
   .+=0x00000100;
   _svc_stack = .;
   =ALIGN(0x10);
   _irq_stack_top = .;
   .+=0x00002000;
   _irq_stack=.;
// Next page
```



.=ALIGN(0x10); _und_stack_top=.; .+=0x00000100; _und_stack=.;		
.=ALIGN(0x10); _abt_stack_top=.; .+=0x00000100; _abt_stack=.; }>STACK		
} # EOF		

6) その他-その他 (デフォルト)

7) アーカイブ (1/2)



8) アーカイブ (2/2)



2-1-4-5. Tool Settings(Objcopy)

7ィルタ入力	Settings			(-, -, -, -, -, -, -, -, -, -, -, -, -, -
<ul> <li>&gt; リソース</li> <li>▲ C/C++ ビルド</li> <li>Device</li> <li>Settings</li> <li>Tool chain エディタ</li> </ul>	構成: HardwareDebug [アクラ ※ Tool Settings デビルド・	ティブ] ステップ   🙅 ビルド成!	果物 │ 読 バイナリー・パーサー │ 🧿 エラー・パーサ	▼ 構成の管理…
ツールチェーン・バ ビルド変数 ロギング 依存関係スキャン 環境 ▷ C/C++ 一般		コマンド: すべてのオプション:	arm-none-eabi-objcopy -O srec	4 
<ul> <li>タスク・リポジトリー ビルダー プロジェクト参照</li> <li>リファクタリング履歴</li> </ul>		エキスパート設定: コマンド行 パターン:	\${COMMAND} \${FLAGS} \${OUTPUT_FLAG}\${(	DUTPUT_PREFIX]
?			ОК	キャンセル

# 1) 一般

リソース ^ 構成: HardwareDebug [アクティブ]	11
Device Settings Tool chain エディタ ツールチェーン・バ ビルド変数 ロギング 低存関係スキャン 環境 C/C++ 一般 タスク・リポジトリー ビルダー プロジェクト参照 リファクタリング履歴 ・ ・ * * * * * * * * * * * * *	<ul> <li>▼ 構成の管理</li> <li>エラー・パーサー</li> <li>▼</li> </ul>

\_

2-1-4-6. ビルド・ステップ (デフォルト)

(カ	Settings 🗘 🕆 🖒
·ス + ビルド vvice ttings	構成: HardwareDebug [アクティブ]  ・  構成の管理
I chain エディター	🛞 Tool Settings 🗶 ビルド・ステップ 🍨 ビルド成果物 🗟 バイナリー・パーサー 😣 エラー・パーサー
デエーン・ハー. 変数	ビルド前のステップ
r	コマンド:
係スキャン	
·铅	說明:
ポジトリー	·····
クト参照	ービルド後のステップ
レング 履歴 # 50-111	
/ BX AC	
	說明:
	▼
•	

### 2-1-4-7. ビルド成果物

フィルタ入力	Settings 🗘 🕆 🗸 🗸
<ul> <li>&gt; リソース</li> <li>a C/C++ ビルド</li> <li>Device</li> <li>Settings</li> <li>Tool chain エディター</li> <li>ツールチェーン・バー:</li> <li>ビルド変数</li> <li>ロギング</li> <li>成存関係スキャン</li> <li>環境</li> <li>&gt; C/C++ 一般</li> <li>タスク・リポジトリー</li> <li>ビルダー</li> <li>プロジェクト参照</li> <li>リファクタリング履歴</li> <li>実行/デバッグ設定</li> </ul>	構成: HardwareDebug [アクティブ] ・ 構成の管理 ● Tool Settings ♪ ビルド・ステップ ● ビルド成果
< <u> </u>	・ OK キャンセル

2-1-4-8. バイナリー・パーサー (デフォルト)

7ィルタ入力	Settings	↓ ↓ ↓ ↓
<ul> <li>リソース</li> <li>∠(C++ ビルド Device</li> <li>Settings) Tool chain エディター ツールチェーン・バー ビルド変数</li> </ul>	構成: HardwareDebug [アクティブ]	構成の管理)
<ul> <li>ロギング</li> <li>依存関係スキャン</li> <li>環境</li> <li>&gt; C/C++ 一般</li> <li>&gt; タスク・リポジトリー</li> <li>ビルダー</li> <li>プロジェクト参照</li> <li>リファクタリング履歴</li> <li>実行/デバッグ設定</li> </ul>	バイナリー・パーサー: ② Elf パーサー ③ PE Windows パーサー ③ Mach-O 64 パーサー ③ Cygwin PE パーサー ⑤ Cygwin PE パーサー ⑤ Mach-O Parser (Deprecated) ⑤ AIX XCOFF32 パーサー ⑤ GNU Elf パーサー ⑤ HP-UX SOM パーサー	上へ移動
?	ОК +	- ד>דנו

## 2-1-4-8. エラー・パーサー (デフォルト)

イルタ入力	Settings	<
ロソソース C/C++ ビルド Device	構成: HardwareDebug [アクティブ]   ▼	構成の管理
1001 chain エティター ツールチェーン・バー:	🛞 Tool Settings 🎤 ビルド・ステップ 🍨 ビルド成果物 🗟 バイナリー・パーサー 👰 エラー・	パーサー
ヒルド変数	🖉 🗢 GNU Assembler Error Parser	追加
山テンン 依存関係スキャン	🖉 🖛 GNU gmake Error Parser 7.0	A=./JH
環境	🖉 🏍 GNU Linker Error Parser	編集
C/C++ 一般		削除
タスク・リポジトリー	☐ ← GHS C/C++ Error Parser	133192
ビルダー	🔲 🖛 GNU gmake Error Parser 6.0 (Deprecated)	上へ珍動
プロジェクト参照	🔲 🕽 🛏 IAR Error Parser	
リファクタリング履歴	🔲 🚈 Microsoft Visual C Error Parser	下へ移動
実行/デバッグ設定	■ �=Renesas C/C++ エラー・パーサー	
	🔲 🕽 🖛 Renesas Make エラー・パーサー	
• III	Renesas RXC エラー・パーサー	
?	ОК <b>‡</b> т	ンセル

2-1-5. C/C++ビルド (Tool Chain エディター) (デフォルト)

e <sup>2</sup> プロパティ: Sample_BARE_F	4F	
フィルタ入力	Tool chain エディター	← → ⇒ → →
▷ リソース ▲ C/C++ ピルド Device Settings	構成(HardwareDebug [アクティブ]	▼ 構成の管理…
Tool chain エディタ ツールチェーン・バー: ビルド変数 ロギング 依存即係スキャン	☑ 互換 toolchain のみ表示 現在の toolchain (KPIT GNUARM-NONE-EABI Toolchain)	•
環境 ▷ C/C++ 一般 ▷ タスク・リポジトリー ピリガー	現在のビルダー: Builder 使用ツール	<b></b>
プロジェクト参照 リファクタリング履歴 実行/デバッグ設定	Library Generator Compiler Assembler Linker Objcopy	▲ <u>ツールの選択</u> E
4 111	デフォルトの復元(T)	適用(L)
?	ок	キャンセル

## 2-1-6. C/C++ビルド (ツールチェーン・バージョン) (デフォルト)

フィルタ入力	ツールチェーン・	バージョンの変更	i		← → ⇒ →
▶ リソース					
⊿ C/C++ ビルド	プロジェクト名:	Sample_BARE_R4F			
Device Settings	ツールチ…ン名:	KPIT GNUARM-NONE	-EABI Toolchain		
Tool chain エディター	現在のバ ヨン・	v16.01			
ツールチェーン・バー	SELON (				
ビルド変数	有効なバ…ョン:	v16.01 -			
ロギング					
依存関係スキャン					
環境					
▷ C/C++ 一般					
▷ タスク・リポジトリー					
ビルダー					
プロジェクト参照					
リファクタリング履歴					
実行/デバッグ設定					
			デフォルトの復	π(T)	適用(L)
4 III +					~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
		-			
			OK		キャンセル

2-1-7. C/C++ビルド(ビルド変数)(デフォルト)

e <sup>2</sup> プロパティ: Sample_BARE_F フィルタ入力	<sup>R4F</sup> ビルド変数				- • • • •
▷ リソース ▲ C/C++ ピルド Device Settings	構成: Hardwar	reDebug [アクテ	ィブ]		▼ 構成の管理
Tool chain エディター ツールチェーン・バー: ビルド変数 ロギング 依存関係スキャン 環境 ▷ C/C++ 一般 ▷ タスク・リポジトリー ビルダー プロジェクト参照 リファクタリング履歴 実行/デバッグ設定	名前	タイプ	値		追加         編集         削除
۰ III. ا				デフォルトの復元(T)	適用(L)
?				ОК	キャンセル

2-1-8. C/C++ビルド(ロギング)(デフォルト)

フィルタ入力	ロギング	← → → → →
<ul> <li>&gt; リソース</li> <li>△ C/C++ ビルド Device Settings Tool chain エディター ツールチェーン・バー: ビルド変数 ロギンク 依存関係スキャン 環境</li> <li>&gt; C/C++ 一般</li> <li>&gt; タスク・リポジトリー ビルダー プロジェクト参照 リファクタリング履歴 実行/デバッグ設定</li> </ul>	マビルドのロギングを有効にするLL ログファイルの場所(F): N:¥UsrAp¥C_H28_AICk 必要に応じてチェック	HI¥RZT1¥Sample_e2¥RZT1_Sample_BARE¥.metadat (
<	<	デフォルトの復元(T) 適用(L)

2-1-9. C/C++ビルド(依存関係スキャン)(デフォルト)

フィルタ入力	依存関係スキャン	$\langle \neg \bullet \neg \neg \bullet $
<ul> <li>&gt; リソース</li> <li>∠ C/C++ ビルド Device Settings Tool chain エディター ツールチェーン・パー: ビルド変数 ロギング     <li><b>び</b>存開係スキャン 環境     <li>▷ C/C++ 一般     <li>タスク・リポジトリー ビルダー     <li>プロジェクト参照 リファクタリング履歴 実行/デバッグ設定</li> </li></li></li></li></ul>	<ul> <li>依存関係スキャン</li> <li>○ 依存関係をスキャンしてプロジェクトをビルド</li> <li>● 低存関係をスキャンしてプロジェクトをビルド</li> <li>○ 既存の依存関係を使用してプロジェクトをビルド</li> </ul>	
4	デフォルト	トの復元(T) 適用(L)
?	ОК	キャンセル

## 2-1-10. C/C++ビルド(環境)(デフォルト)

フィルタ入力	環境			<b>\</b>	
▷ リソース					
▲ C/C++ ビルド Device Settings	構成: HardwareDebug [フ	"クティブ]		▼ 構成	成の管理
Tool chain エディター ツールチェーン・バー:	設定する環境変数				追加
ビルド変数	変数	値	由来		
ロキング	AMS_KEEP_FILE	\${synergyKeepFile}	ユーザー:構成		J至1/、···
低存関係スキャン	AMS_LICENSE_PATH	\${synergyLicenseFile}	ユーザー:構成		編集
<b>填</b> 現	CONFIGDIR	\${workspace_loc:/\${ ビルド・シス		=	8000
▷ C/C++ 一般	CWD	N:¥UsrAp¥C_H28_AI	ビルド・シス		門际
▷ タスク・リボジトリー	GCC_VERSION	5.2-GNUARM-NONE	ビルド・シス		定義解除
ビルター プロジェクト共昭	PATH	C:¥PROGRA~1¥KPIT	ビルド・シス		
ノロシェクト参照 リファカタロトが展歴	PWD	N:¥UsrAp¥C_H28_AI	ビルド・シス		
ラファウラウフラ値座 宇行/デバッグ設定	TCINSTALL	C·¥PROGRA~1¥KPIT	ビルド・シス	-	
±11/7/19/2822	◎ ネイティブ環境へ変数を	追加			
	◎ ネイティブ環境を指定さ	れた環境と置換			
4		デフ	7ォルトの復元(T)	適用(L	.)
2			ОК	キャンセル	_

\_

2-1-11.【説明省略】C/C++ 一般(デフォルト)
2-1-12.【説明省略】タスク・リポジトリー(デフォルト)
2-1-13.【説明省略】ビルダー(デフォルト)
2-1-14.【説明省略】プロジェクト参照(デフォルト)
2-1-15.【説明省略】リファクタリング履歴(デフォルト)

2-1-16. 実行/デバッグ設定

1)新規登録

e <sup>2</sup> プロパティ: Sample_BARE_F	14F		
フィルタ入力	実行/デバッグ設定		↓ ↓ ↓ ↓
▷ リソース ▷ C/C++ ビルド ▷ C/C++ 一般	このページで現在選択されているリソ 'Sample_BARE_R4F' の起動構成(F):	ースに関連する起動構成を智	言理できます。 ^
▷ タスク・リポジトリー ビルダー		$\leq$	新規(N)
プロジェクト参照			複製(P)
リファクタリング履歴 (実行/デバッグ設定)			編集(E)
			削除(L) ■
			•
?		ок	キャンセル

2)構成タイプの選択

e <sup>2</sup> 構成タイプの選択	
作成する構成のタイプを選択(S):	
C/C++ アプリケーション	
GDB Simulator Debugging (SH, RH850)	
Renesas GDB Hardware Debugging	
Renesas GDB Simulator Running	
💽 Renesas Simulator Debugging (RX, RL78)	
	×1711.

3) 起動構成プロパティの編集(メイン) (デフォルト)

記動構成プロパティの編集 名前(N): Sample_BARE_R4F HardwareDebug スイン 珍 Debugger ▶ Startup □ 共通(C) ▷ ソース プロジェクト(P): Sample_BARE_R4F (C/C++ アブリケーション: HardwareDebug¥Sample_BARE_R4F.x 変数(V) プロジェクトの検索(H) 参照(R) 変数(v) プロジェクトの検索(H) 参照(R) 更加に必要に応じてビルド ビルド構成: Select Automatically ① 自動ビルドを無効にする ③ つークスペース設定の構成 前回保管した状態に戻す(V) 適用(Y) ② OK 主ヤンセル	構成の編集		<b>EX</b>
Ami(N): Sample_BARE_R4F HardwareDebug メイン 参 Debugger ▶ Startup ■ 共通(C) ▶ ソース プロジェクト(P): Sample_BARE_R4F C/C++ アプリケーション: HardwareDebug¥Sample_BARE_R4F.x 変数(V) プロジェクトの検索(H) 参照(R) 起動前に必要に応じてビルド ビルド構成: Select Automatically ● 自動ビルドを有効にする ④ ワークスペース設定の使用 ⑦ OK キャンセル	己動構成プロパティの編集		Ť.
<ul> <li>スイン 称 Debugger ▶ Startup □ 共通(C) ▶ ソース</li> <li>プロジェクト(P):</li> <li>Sample_BARE_R4F</li> <li>C/C++ アプリケーション:</li> <li>HardwareDebug¥Sample_BARE_R4F.x</li> <li>変数(V)</li> <li>プロジェクトの検索(H)</li> <li>参照(R)</li> <li>ごしジェクトの検索(H)</li> <li>参照(R)</li> <li>ごしいド構成: Select Automatically</li> <li>自動ビルドを有効にする</li> <li>ワークスペース設定の使用</li> <li>ワークスペース設定の構成</li> <li>①</li> <li>①</li> <li>①</li> <li>○</li> <li></li></ul>	前(N): Sample_BARE_R4F HardwareDebug		
プロジェクト(P): Sample_BARE_R4F C/C++ アプリケーション: HardwareDebug¥Sample_BARE_R4F.x 変数(V) プロジェクトの検索(H) 参照(R) 起動前に必要に応じてビルド ビルド構成: Select Automatically ● 自動ビルドを有効にする ● ワークスペース設定の使用 ① ークスペース設定の構成 前回保管した状態に戻す(V) 適用(Y)	📔 メイン 🏇 Debugger 🕨 Startup 🔲 共道	▲(C) 陸 ソース	
Sample_BARE_R4F       参照(B)         C/C++ アブリケーション:          HardwareDebug¥Sample_BARE_R4F.x          室数(V)       プロジェクトの検索(H)       参照(R)         起動前に必要に応じてビルド          ビルド構成:       Select Automatically          ● 自動ビルドを有効にする       ● 自動ビルドを無効にする          ⑦ ワークスペース設定の使用       ワークスペース設定の構成          前回保管した状態に戻す(V)       適用(Y)         ⑦       OK       キャンセル	プロジェクト(P):		<u>^</u>
C/C++ アプリケーション: HardwareDebug¥Sample_BARE_R4F.x を数(V) プロジェクトの検索(H) 参照(R) 起動前に必要に応じてビルド ビルド構成: Select Automatically ● 自動ビルドを有効にする ● フークスペース設定の構成 前回保管した状態に戻す(V) 適用(Y) ② OK キャンセル	Sample_BARE_R4F		参照(B)
HardwareDebug¥Sample_BARE_R4F.x       変数(V)       プロジェクトの検索(H)       参照(R)         起勤前に必要に応じてビルド             ビルド構成:       Select Automatically            ● 自動ビルドを有効にする       ● 自動ビルドを無効にする            ⑨ ワークスペース設定の使用       ワークスペース設定の構成            ⑦ 回帰管した状態に戻す(V)       適用(Y)           ⑦	C/C++ アプリケーション:		
変数(V)     プロジェクトの検索(H)     参照(R)       起動前に必要に応じてビルド       ビルド構成:       Select Automatically       ● 自動ビルドを有効にする       ● フークスペース設定の使用       ワークスペース設定の使用       ワークスペース設定の使用       第回保管した状態に戻す(V)       適用(Y)	HardwareDebug¥Sample_BARE_R4F.x		
起動前に必要に応じてビルド ビルド構成: Select Automatically ● 自動ビルドを有効にする ● ワークスペース設定の使用 可ークスペース設定の構成 前回保管した状態に戻す(V) 適用(Y) ② OK キャンセル	2	変数(V) プロジェクトの検索(H)	参照(R) ⊨
ビルド構成:       Select Automatically         ● 自動ビルドを有効にする       ● 自動ビルドを無効にする         ● ワークスペース設定の使用       ワークスペース設定の構成         前回保管した状態に戻す(V)       適用(Y)         ⑦       OK       キャンセル	起動前に必要に応じてビルド		
<ul> <li>○ 自動ビルドを有効にする</li> <li>○ ワークスペース設定の使用</li> <li>ワークスペース設定の構成</li> <li>前回保管した状態に戻す(V) 適用(Y)</li> <li>②</li> <li>○ OK</li> </ul>	ビルド構成: Select Automatically		<b>-</b>
・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	◎ 自動ビルドを有効にする	◎ 自動ビルドを無効にする	
前回保管した状態に戻す(V)     適用(Y)       ⑦     OK     キャンセル		ワークスペース設定の構成	
前回保管した状態に戻す(V)     適用(Y)       ⑦     OK     キャンセル			•
<ul> <li>OK キャンセル</li> </ul>		前回保管した状態に戻す(V)	適用(Y)
<ul> <li>OK キャンセル</li> </ul>			
	?	ок	キャンセル

4) 起動構成プロパティの編集(Debugger[GDB Settings])

<ul> <li>●<sup>2</sup> 構成の編集</li> </ul>
起動構成プロパティの編集 J-Link ARM
名前(N): Sample_BARE_R4F HardwaryDebog
○ メイン ② Debugger ▶ Star pp □ 共通(C) ⇒ ソース
Debug hardwares 1-Link ARM ・ Target Device R7S910018 R7S910018 R7S910018 R7S910018
③ ローカル GDB サーバーを自動起動       ホスト名または IP アドレス:       localhost         ③ リモート GDB サーバーへ接続       GDB ボート番号:       61234         ADM ボート番号:       61236
GDB コマンド: \${eclipse_home}/DebugComp/arm-none-eabi-gdb
□ 詳細モードを有効にする
OK         キャンセル

\_

5) 起動構成プロパティの編集(Debugger[Connection Settings]) (デフォルト)

記動構成プロパティの編集 1	TO T
名前(N): Sample_BARE_R4F HardwareDebug	
Dy A. (* Dohuggor) Chartup E #3(0) 5. V	-7)
Debug hardware: J-Link ARM   Target Device:	R75910018
GDB Settings Connection Settings テパック・ツール設立	£
Type	USB
1-l ink Serial	(Auto)
設定ファイル	\${workspace loc:\u00e4\${ProiName}}\u00e4\${LaunchConfigName}
Script File	+ c
Low Power Handling	No
Interface	
Type	JTAG
Speed (kHz)	Auto
▲ JTAG Scan Chain	
Multiple Devices	いいえ
IRPre	0
DRPre	0
⊿ 接続	
レジスターの初期化	いいえ
接続時にリセット	いいえ
実行前にリセット	いいえ
ID ⊐−ド	FFFFFFFFFFFFFFFFFFFFFFFFF
接続時にリセット状態を維持する	いいえ
⊿ SWV	
Core clock (MHz)	0
•	
?	前回保管した状態に戻す(V) 適用(Y) OK キャンセル

6) 起動構成プロパティの編集(Debugger[デバッグ・ツール設定])

記動構成プロパティの編集		ñ
3前(N): Sample_BARE_R4F HardwareDebug		
■ メイン 森 Debugger ト Startun 同 共通(C) い	×	
Debug hardware: J-Link ARM 🔹 Target Devi	ice: R7S910018	
GDB Settings Connection Settings デバッグ・ツール	設定	
4 IO		
デフォルト IO ファイル名を使用	はい	=
IO ファイル名	\${eclipse_home}¥internal¥IoFiles¥RZ¥R7S910018.sfrx	
▲ General Debug		
Reset After Reload	いいえ	
RTOS Integration in Debug View	いいえ	
▲ Xモリー		
エンディアン	リトル・エンディアン	
⊿ 中断		
フラッシュ・ブレークポイントを使用する	いいえ	
シミュレーションを許可する	いいえ	
⊿ Flash		
Use CFI-Flash	いいえ	
WorkRam Start	0x0	
WorkRam End	0x0	
CFI Start	0x0	
CFI End	0x0	
Semihosting		_
Semihosting breakpoint address		
		-
		-
•	4	
	前回保管した状態に戻す(V) 適用(Y)	

7) 起動構成プロパティの編集(Startup)

	/柚朱					)
tj(N): Sample_BARE_R	4F HardwareDebug					
メイン 🕸 Debugger 🕻	Startup 🔲 共通(C)	<b>₩</b> ν-ス				
の期化コマンド						
🔲 リセットと遅延 (秒): —	3					
Halt						
						-
イメージとシンボルをロ-	-۲					
ファイル名	ロード・タイプ	オフセット	接続時			
🔽 プログラム・バイナ	イメージとシンボル	0	Yes			
						編集
						除去
						上へ
ランタイム・オプション-						
] プログラム・カウンタ-	-設定先(16進):		20	ヨットック	<=n,,-++	7
2 ブレークボイント設定的	t: main			要に応じて	.設定	
一一円用						
コマンドを実行						
						-
			前回	保管した状態に原	冥す(V)	適用(Y)



8) 起動構成プロパティの編集(共通)

H 3/AV A 2/H BHC			
≧動構成プロパティ	の編集		1
			200
站前(N): Sample_BARE_	R4F HardwareDebug		
📄 メイン 🏇 Debugger	▶ Startup (二共通(C)	▶ ソース	
	2)		
	) 		(4 TT (- 1
◎ 共用ファイル(H):	¥Sample_BARE_K4F		参照(B)
お気に入りメニューに表	示(R)	エンコード	
■ ☆デバッグ		◎ デフォルト - 継承(U) (SJIS)	
		④ その他(E) UTF-8	-
		「UTF-8」を選	択
		l	J
標準入出力			
標準入出力 ▼コンソールに割り当て	:(A) (入力に必要)		
- 標準入出力 ▼ コンソールに割り当て 同 入力ファイル(F):	(A) (入力に必要)		
標準入出力 ☑ コンソールに割り当て □ 入力ファイル(F):	:(A) (入力に必要) ワークスペース	ファイル・システム…	<u> </u>
標準入出力 ▼コンソールに割り当て □入力ファイル(F): □出力ファイル(L):	:(A) (入力に必要) ワークスペース	) ファイル・システム…	亦粉 实数…
標準入出力 ☑ コンソールに割り当て □ 入力ファイル(F): □ 出力ファイル(L):	:(A) (入力に必要) ワークスペース	) ファイル・システム	<u>奕</u> 数
標準入出力 ☑ コンソールに割り当て □ 入力ファイル(F): □ 出力ファイル(L):	:(A) (入力に必要) ワークスペース ワークスペース(W)	) ファイル・システム… ) 参照(R)…	变数 变数
<ul> <li>標準入出力</li> <li>✓ コンソールに割り当て</li> <li>□ 入力ファイル(F):</li> <li>□ 出力ファイル(L):</li> <li>□ 追加(P)</li> </ul>	:(A) (入力に必要) ワークスペース ワークスペース(W)	) ファイル・システム 参照(R)	<u>変</u> 数 変数
標準入出力 ☑ コンソールに割り当て □ 入力ファイル(F): □ 出力ファイル(L): □ 追加(P) ☑ パックグラウンドでの誘	:(A) (入力に必要) ワークスペース ワークスペース(W) 己動(K)	) ファイル・システム… ) 参照(R)…	变数 变数
標準入出力 ▼コンソールに割り当て □入力ファイル(F): □出力ファイル(L): □追加(P) ▼バックグラウンドでの表	:(A) (入力に必要) ワークスペース ワークスペース(W) 己動(K)	) ファイル・システム… ) 参照(R)…	变数 变数
標準入出力 ☑ コンソールに割り当て □ 入力ファイル(F): □ 出力ファイル(L): □ 追加(P) ☑ パックグラウンドでの読	:(A) (入力に必要) ワークスペース ワークスペース(W) 己動(K)	) ファイル・システム 参照(R) 前回保管した状態に戻す(V)	<u>変数</u> 変数 適用(Y)
標準入出力 ▼コンソールに割り当て □入力ファイル(F): □出力ファイル(L): □追加(P) ▼ノバックグラウンドでの読	:(A) (入力に必要) ワークスペース ワークスペース(W) 2動(K)	) ファイル・システム… 参照(R)… 前回保管した状態に戻す(V)	変数 変数 変数 適用(Y)
<ul> <li>標準入出力</li> <li>□ コンソールに割り当て</li> <li>□ 入力ファイル(F):</li> <li>□ 出力ファイル(L):</li> <li>□ 追加(P)</li> <li>マ バックグラウンドでのあ</li> </ul>	:(A) (入力に必要) ワークスペース ワークスペース(W) 記動(K)	) ファイル・システム ) 参照(R) 前回保管した状態に戻す(V)	<u>変数</u> 変数 変数 適用(Y) キャンセル

9) 起動構成プロパティの編集(ソース) (デフォルト)

動構成プロパティの編集	
前(N): Sample_BARE_R4F HardwareDebug	
) メイン (参 Debugger 🌘 Startup 🔲 共通(C) 👰	V-X
/ース・ルックアップ・パス(0):	
▷ 🗁 デフォルト	追加(A)
	福集(E)
	除去(M)
	上へ(P)
	デフォルトの復元(U)
]パス上の重複ソース・ファイルを検索(H)	
	前回保管した状態に戻す(V) 適用(Y)

\_
### 2-2. コア【M3】 側の確認





- 2-2-1. リソース
  - 1) テキスト・ファイル・エンコード

e <sup>2</sup> プロパティ: Sample_BARE_I	из	- • •
フィルタ入力	リソース	← ← ⇒ → ▼
<ul> <li>リソース</li> <li>&gt; C/C++ ビルド</li> <li>&gt; C/C++ 一般</li> <li>&gt; タスク・リポジトリー ビルダー</li> <li>プロジェクト参照 実行/デバッグ設定</li> </ul>	パス(P): /Sample_BARE_M3 タイブ(Y): プロジェクト ロケーション(L): N:¥UsrAp¥C_H28_AICHI¥RZT1¥Sample_e2¥RZT1_Sample_BARE¥S 最終変更日時(M): 2017年6月12日 9:28:40 テキスト・ファイル・エンコード(T) ◎ コンテナーから継承(I) (SJIS) ◎ その他(O): SJIS ・ □派生リソースのエンコードを別途保管(S) 新規テキスト・ファイルの行区切り文字(F) ◎ コンテナー (Windows) から継承(E) ◎ その他(H): Windows ▼	Gample_BARE_M3
	< III	F
?	ОК ‡	マンセル

\_



#### 2-2-2. C/C++ビルド (TOP)

1) ビルダー設定 (デフォルト)

フィルタ入力       C/C++ ビルド         > リソース       C/C++ ビルド         C/C++ ビルド       構成の管理         > タスク・リポジトリー       増成:         ビルダー       プロジェクト参照         実行/デバッグ設定       ● 振る舞い ◇ ポリシーを更新         ビルダー       ビルダー         ビルダー       ビルダー         ビルダー       ビルダー         ビルダー・タイプ(T)       外部ビルダー         マデフォルト・ビルド・コマンドを使用(U)
<ul> <li>&gt; リソーム</li> <li>★ リソーム</li> <li>★ ロング・ロー</li> <li>★ ロング・ロー<!--</th--></li></ul>
ビルダー プロジェクト参照 実行/デバッグ設定 ・ ビルダー・タイプ(T)、 外部ビルダー ・ マ マ デフォルト・ビルド・コマンドを使用(U)
ビルダー・タイプ(T)(外部ビルダー・ 『デフォルト・ビルド・コマンドを使用(U)
ビルド・コマンド(C): make 変数
Makefile 生成 『 自動的に Makefile を生成(G) 『 Makefile に環境変数参照を展開(E)
ビルド・ロケーション ビルド・ディレクトリー(D): \${workspace_loc:/Sample_BARE_M3}/HardwareDebug
「ワークスペース…」「ファイル・システム…」「変数…」
デフォルトの復元(T) 適用(L)
のK         キャンセル

2) 振る舞い (デフォルト)

e <sup>2</sup> プロパティ: Sample_BARE_R	4F		
フィルタ入力	C/C++ ビルド		← ▼ ⇒ ▼
▲ リソース リソース・フィルター リンクされたリソース ▲ C/C++ ビルド Device	構成: HardwareDebug [アクティブ]		▼ 構成の管理…
Settings Tool chain エディター ツールチェーン・パー: ビルド変数 ロギング 依存関係スキャン	<ul> <li>■ ビルダー設定 ● 振る舞 ● ポリミ</li> <li>ビルド設定</li> <li>✓ 最初のビルド・エラーで停止</li> </ul>	ンーを更新 □ 並列ビルドを有効にする ⑥ Use optimal jobs (4) ○ 並列ジョブを使用: ④ 無制限のジョブを使用	
環現 ▷ C/C++ 一般 ▷ タスク・リポジトリー ビルダー プロジェクト参照 リファクタリング履歴 実行/デバッグ設定	ワークベンチ・ビルドの振る舞い ワークベンチ・ビルド・タイプ: リソース保管時にビルド(自動ビルド) 注:ワークベンチの自動ビルド設定を参照 I ビルド(インクリメンタル・ビルド) I クリーン	Make ビルド・ターゲット: all all	変数 変数 変数
< )		デフォルトの復元(T) OK 4	適用(L) Eヤンセル

3) ポリシーを更新 (デフォルト)

e <sup>2</sup> プロパティ: Sample_BARE_R	14F
	C/C++ ビルド (> ▼ ⇒ ▼ ▼
<ul> <li>&gt; リソース</li> <li>&gt; C/C++ ビルド</li> <li>&gt; C/C++ 一般</li> <li>&gt; タスク・リボジトリー ビルダー プロジェクト参照 リファクタリング履歴 実行/デバッグ設定</li> </ul>	構成: HardwareDebug [アクティブ] ・ 構成の管理 ■ ビルダー設定 ● 振る舞い
?	OK         キャンセル

## 2-2-3. C/C++ビルド (Device)

<sup>2</sup> プロパティ: Sample_BARE_N	13		
フィルタ入力	Device		← → ⇒ →
▶ リソース			
▲ C/C++ ビルド	Current Device: R7S910018_M3		
Device	Chapter Dovice: P7C010010 M2		
Settings	Change Device. R/S910018_M3		
Nールチェーン・パー・			
ビルド変数			
ロギング			
依存関係スキャン			
環境			
▷ C/C++ 一般			
▶ タスク・リポジトリー			
ビルダー			
ノロシェクト参照 実に/ポルガ設会			
关1)//パック設定			
• III • •		デフォルトの復元(T)	週用(L)
(?)		ОК	キャンセル



## 2-2-4. C/C++ビルド (Settings)

2-2-4-1. Tool Settings(Library Generator)

カ Settings		↓
トビルド + ビルド 構成: HardwareDebug ttings	[アクティブ]	<ul> <li>√ 構成の管</li> </ul>
ol chain エディター ールチェーン・バー: レド変数	アルド・ステップ 💮 ビルド成果物	■ パイナリー・パーサー <b>③</b> エラー・パーサー
ング 製紙スキャン 一般 後 この他のオフ 一般 後 この他のオフ の しの の 、 、 の 、 の 、 の 、 の 、 の 、 の 、 、 の 、 の 、 の 、 、 の 、 の 、 、 の 、 の 、 の 、 、 の 、 の 、 の 、 の 、 の 、 の 、 、 の 、 の 、 の 、 の 、 の 、 の 、 の 、 の 、 の 、 の 、 、 の 、 の 、 の 、 の 、 の 、 の 、 の 、 の の 、 の の 、 の の 、 の 、 の の 、 、 の 、 の 、 の 、 の 、 の 、 の 、 、 の の 、 、 の の 、 、 の 、 の 、 の 、 の 、 の 、 、 の 、 の 、 の 、 、 、 の 、 、 、 の 、 、 、 、 、 の 、 、 、 、 の 、 の 、 、 、 、 、 、 、 、 、 、 、 、 、	すべてのオブション: アイル ジョン	select-lib=optibheader-files=allcompiler-options=-mcpu=cortex-m3,- march=armv7-m,-mlittle-endian,-mthumb,-mfloat-abi=soft,-Os,-ffunction- sections,-fdata-sections,-fno-function-cse,-funit-at-a-time,-falign-jumps
ホジトリー	エキスパート設定: コマンド行 パターン:	<pre>\${COMMAND} \${FLAGS} \${OUTPUT_FLAG}\${OUTPUT_PREFIX} \${INPUTS}\${OUTPU</pre>
🖄 一般		
•		

1) 設定 (デフォルト)

】 プロパティ: Sample_BARE_R フィルタ入力	4F Settings	
<ul> <li>&gt; リソース</li> <li>2 C/C++ ビルド Device Settings Tool chain エディター ツールチェーン・バー: ビルド変数 ロギング 依存閣係スキャン 環境</li> <li>&gt; C/C++ 一般</li> <li>&gt; タスク・リポジトリー ビルダー プロジェクト参照 リファクタリング履歴 実行/デバッグIB定</li> </ul>	構成: HardwareDebug [アクティブ] ③ Tool Settings ▲ じしド・ステップ ④ ビ ● ひにド・ステップ ● ビ ● ひにだ・ステップ ● ビ ● ひたうファイル ● ③ たの他のガジョン ● その他のガジョン ● その他のガジョン ● そのでのオジョン ● そのでの ● ③ Compiler ● ③ Compiler	<ul> <li>▼ 構成の管理 ▲</li> <li>S成果物 論 パイナリー・パーサー ③ エラー・パーサー</li> <li>型 Project-Built ・</li> <li>の避沢 Optimized ・</li> <li>Build library (only when options change) ・</li> <li>libgcc along with project-built library</li> </ul>
?		OK         キャンセル



2) ヘッダー・ファイル



3) その他のオプション (デフォルト)

4) その他(1/2)

Settings	
-ス + ビルド 構成: HardwareDebug [アクティン	7] ・ 構成の
wice wittings ol chain エディター ールチェン・バー: ルド変数 ギング 存留紙スキャン 境 + 一般 ア・リポジトリー テー 江クト参照 ぐクタリング履歴 デバッグ設定 W Tool Settings	・シブ       ・シレド成果物       ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

## 5) その他 (2/2)



2-2-4-2. Tool Settings(Compiler)

	Settings	\$ • \$ •
<ul> <li>■ C/C++ ビルド Device Settings Tool chain エディター ツールチェーン・バー: ビルド変数 ロギング 依存関係スキャン 環境</li> <li>&gt; C/C++ 一般</li> <li>&gt; タスク・リポジトリー ビルダー プロジェクト参照 実行/デバッグ設定</li> </ul>	構成: HardwareDebug [アクティブ] ③ Tool Settings ♪ ビルド・ステップ ① ビルド成果物 ඛ パイナリー・パー ③ Library Generator ③ Compile ③ フラス ④ オブジェクト ② リスト ④ 選覧 ④ 振振 ④ 振振 ④ 変形 ④ その他 ⑥ CPU ▷ ③ Assembler ▷ ③ Dipcopy ⑧ Objcopy ⑧ 一般	<ul> <li>● 構成の管理</li> <li>サー ② エラー・パーサー</li> <li>e \$(notdir \$&lt;)).lst" -nostdinc -I"C: ARM~1.01-¥ARM-NO~1¥ARM- itlibinc" -I"N:¥USrAp¥C_H28_AICHI *</li> <li>\${OUTPUT_FLAG} \${OUTPUT_PREFIX}\${</li> </ul>
?		ОК <b>キャンセル</b>

1) ソース

タ入力	Settings 🗘 🕆 🖒
アーへ i++ ビルド Device Settings Tool chain エディター ツールチェーン・パー:	構成: HardwareDebug [アクティブ]  ◆ 構成の管理  No Settings
ビルド変数 ロギング 依存関係スキャン 環境 :++ 一般 :ク・リポジトリー ダー Jジェクト参照 5/デバッグ設定	Source ProjName}/src/Renesas/CMSIS/Include}"     Source ProjName}/src/Renesas/RZT1_RIN_Engine/Include}"     Source ProjName}/src/Renesas/RZT1_RIN_Engine/Include}"
III •	マクロ定義 RZT1_REGISTER_CORTEX_M3_ _RZT1_REGISTER_CORTEX_M3_
	OK         キャンセル

2) オブジェクト



## 3) リスト(デフォルト)



4) 警告 (デフォルト)

ルタ入力	Settings	$\leftarrow$ $\bullet$ $\Rightarrow$ $\Rightarrow$ $\bullet$
リソース C/C++ ビルド Device Settings Tool chain エディター ツールチェーン・バー: ビルド変数 ロギング 低存関係スキャン 環境 C/C++ 一般 タスク・リポジトリー ビルダー プロジェクト参照 リファクタリング履歴 実行/デバッグ設定	<ul> <li>▶ 後 Library Generator</li> <li>▲ ③ Compiler</li> <li>※ ソース</li> <li>※ オブジェクト</li> <li>※ ゴジェクト</li> <li>※ 国本</li> <li>※ 否の他</li> <li>※ その他</li> <li>※ その他</li> <li>※ Con</li> <li>※ Coult</li> <li>※ Coult</li></ul>	

5) 警告-標準 (デフォルト)



### 6) 警告-拡張 (デフォルト)



### 7) その他



## 8) その他-その他(1/2)



## 8) その他-その他 (2/2)





# 9) CPU

アルタ人力 Settings		
<ul> <li>リソース</li> <li>C/C++ ビルド</li> <li>Device</li> <li>Settings</li> <li>Tool chain エディター</li> <li>ツールチェーン・バー:</li> <li>ビルド変数</li> <li>ロギング</li> <li>依存関係スキャン</li> <li>環境</li> <li>C/C++ 一般</li> <li>タスク・リポジトリー</li> <li>ビルダー</li> <li>プロジェクト参照</li> <li>実行/デバッグ設定</li> </ul>	CPU タイプ アーキテクチャー エンディアン 命令セット インター・ワーキング (-mthu ARM Procedure Call Standar Thumb Procedure Call Standar I hamb Procedure Call Standar leaf 関数に対するスタック・フ 浮動小数点 ABI (-mfloat-abi=na 関数プロローグのスケジューリ Disable unaligned word and 関数名をオブジェクトコード(a ターゲット FPU (-mfpu)	cortex-m3       ・         armv7-m       ・         Little-endian       ・         Thumb       ・         umb-interwork)       ・         rdと互換性のあるスタック・フレームを生成する       ・         dardと互換性のあるスタック・フレームを生成する       ・         Jレームを生成する       ・         Iame)       Soft         リングを有効にする       ・         Ihalfword accesses to packed data       ・         に格納する       ・

2-2-4-3. Tool Settings(Assembler)

иядл <b>s</b>	ettings		< → <
Jソース C(C++ ビルド Device Settings Tool chain エディター	<ul> <li>&gt;</li></ul>	コマンド: すべてのオプション:	arm-none-eabi-as -I"N:¥UsrAp¥C_H28_AICHI ¥RZT1¥Sample_e2¥RZT1_Sample_BARE ¥Sample_BARE_M3/src" -adlhn="\$(basename
ッールチェーン・パー: ビルド変数 ロギング 依存関係スキャン 環境 :/C++ 一般 シスク・リポジトリー 2)レダー	● リスト ● その他 ▶ 歌 Linker ▲ 歌 Objcopy ● 一般	エキスパート設定: コマンド行 パターン:	\${COMMAND} \${FLAGS} \${OUTPUT_FLAG} \${OUTPUT_
プロジェクト参照 眞行/デバッグ設定 Ⅲ →			OK キャンセル

## 1) ソース

7ィルタ入力 ▶ リソース	Settings	$\langle \neg \bullet $
<ul> <li></li></ul>	構成: HardwareDebug [アクティ ③ Tool Settings ● ③ Library Generator ● ③ Compiler ④ ③ Assembler ④ ③ Assembler ④ ジース ④ オブジェクト ④ リスト ⑧ Compile ● ③ Library Generator ● ③ Compiler	:ブ] ・ 構成の管理 ボテップ  ・ ビルド成果物  ・ バイナリー・パーサー  ・ エラー・パーサー インクルード・ファイル・ディレクトリー ・ そ  ・ ※ {workspace_loc:/\$ {ProjName}}/src" 「 \$ {workspace_loc:/\$ {ProjName}}/src"
		シンボル定義 🕢 🗟 🖗 🖗
?		



2) オブジェクト

イルタ入力	Settings		$\langle \neg \bullet \circ \rangle \bullet \bullet$
<ul> <li>・ リソース</li> <li>・ C/C++ ビルド</li> <li>Device</li> <li>Settings</li> <li>Tool chain エディター</li> <li>・ ツールチェーン・バー:</li> <li>ビルド変数</li> <li>ロギング</li> <li>依存関係スキャン</li> <li>環境</li> <li>・ O/C++ 一般</li> <li>・ タスク・リポシトリー</li> <li>ビルダー</li> <li>プロジェクト参照</li> <li>実行/デバッグ設定</li> </ul>	<ul> <li>&gt; いにはでは、</li> <li>&gt; いには、</li> <li>&gt; いたい、</li> <li>&gt; いたいいいいいいいいいいいいいいいいいいいいいいいいいいいいいいいいいい</li></ul>	オブジェクト・ディレクトリー コース エラー後もオブジェクト・ファ・ テキスト・セクションとデータ デバッグ・フォーマット エンディアン CPU タイプ アーキテクチャー 命令セット コンター・ワーキング (-mthun 浮動小数点 ABI (-mfloat-abi=nan ターゲット FPU (-mfpu)	\${CONFIGDIR} イルを生成する (-Z) ・セクションを結合する (-R) Dwarf2 ・ Little-endian ・ cortex-m3 ・ armv7-a ・ Thumb ・ mb-interwork) me) Soft ・ None ・
?			0K キャンセル

# 3) リスト(デフォルト)

マルタ入力	Settings $\Leftrightarrow \checkmark \diamond \checkmark \checkmark$
<ul> <li>&gt; リジーム</li> <li>&gt; ノジーム</li> <li>&gt; C/C++ ビルド Device</li> <li>Settings</li> <li>Tool chain エディター</li> <li>&gt; ツールチェーン・パー:</li> <li>ビルド変数</li> <li>ロギング</li> <li>依存関係スキャン</li> <li>環境</li> <li>&gt; C/C++ 一般</li> <li>&gt; タスク・リポジトリー</li> <li>ビルダー</li> <li>プロジェクト参照</li> <li>リファクタリング履歴</li> <li>実行/デバッグ設定</li> </ul>	構成: HardwareDebug [アクティブ]  ◆ 構成の管理
?	OK キャンセル

4) その他(1/2)(デフォルト)

иудр	Settings		$\langle \neg \bullet \neg \neg \rangle \bullet \bullet$
リソース C/C++ ビルド Device Settings Tool chain エディター	構成: HardwareDebug [アクティン ③ Tool Settings 🎤 ビルド・スラ	7] =ップ   🙅 ビルド成果物   🗟 バイナリー・パ	<ul> <li>▼ 構成の管理</li> <li>ーサー (3 エラー・パーサー)</li> </ul>
ツールチェーン・バー: ビルド変数 ロギング 依存関係スキャン 環境 C/C++一般 タスク・リポジトリー ビルダー プロジェクト参照 実行/デバッグ設定	<ul> <li>※ Library Generator</li> <li>※ Compiler</li> <li>※ Assembler</li> <li>※ ソース</li> <li>※ オブジェクト</li> <li>※ リスト</li> <li>※ セの他</li> <li>※ Linker</li> <li>※ Objcopy</li> <li>※ 一般</li> </ul>	ユーザー定義オプション	<b>1</b> 图 图 初 21
2		ОК	キャンセル

5) その他(2/2)(デフォルト)

フィルタ入力 Settings	
<ul> <li>&gt; リソース ▲</li> <li>C/C++ ビルド Device</li> <li>Settings Tool chain エディち ツールチェーン・バ ビルド変数 ロギング 依存関係スキャン 環境</li> <li>&gt; C/C++ 一般</li> <li>&gt; タスク・リポジトリー ビルダー</li> </ul>	<ul> <li>         警告メッセージを抑止する (-W)         ローカルラベルをインクルードする (-L)         差分テーブルが変更された場合に警告する (-k)         高速で作業する (-f)         アセンブリー統計を表示する (statistics)     </li> </ul>
	デフォルトの復元(T) 適用(L)
(?)	

2-2-4-4. Tool Settings(Linker)

フィルタ入力	Settings		<	•
<ul> <li>リソース</li> <li>C/C++ ビルド         Device         Settings         Tool chain エディター             ツールチェーン・バー:             ビルド変数             ロギング             依存関係スキャン             現境         </li> </ul>	構成: HardwareDebug [アクティ 鬱 Tool Settings デビルド・ス > い Library Generator > い Compiler > い Assembler - い Linker ご 入力	・ブ] 、テップ 🙅 ビルド成身 コマンド: すべてのオブション:	<ul> <li>▼ 構成の管理</li> <li>果物 □ パイナリー・パーサー ③ エラー・パーサー</li> <li>arm-none-eabi-ld</li> <li>-T"N:¥UsrAp¥C_H28_AICHI</li> <li>¥RZT1¥Sample_e2¥RZT1_Sample_BARE</li> <li>¥Sample_BARE_M3¥HardwareDebug</li> </ul>	
<ul> <li>&gt; タスク・リポジトリー ビルダー プロジェクト参照 実行/デバッグ設定</li> </ul>	<ul> <li>避 出力</li> <li>逆 セクション</li> <li>2 その他</li> <li>逆 その他</li> <li>逆 アーカイブ</li> <li>S Objcopy</li> <li>逆 一般</li> </ul>	エキスパート設定: コマンド行 パターン:	\${COMMAND} \${OUTPUT_FLAG}\${OUTPUT_PREFI	
?			0K キャンセル	

1)入力 (デフォルト)



2) 出力(デフォルト)

Settings       Tool chain エディター         ツールチェーン・バー:       ビルド マステップ <ul> <li>ビルド マステップ              <li>ビルド 広果物              <li>バイナリー・パーサー              <li>エラー・パーサー</li> <li>ロー</li> <li>ロー</li></li></li></li></ul>	フィルタ入力 ▶ リソース ▲ C/C++ ビルド Device	Settings 構成: HardwareDebug [アクティブ] 構成の管理	
実行/デバッグ設定 避 アーカイブ ▲  ③ Objcopy 避 一般	Seturings Tool chain エディター ツールチェーン・バー: ビルド変数 ロギング 依存関係スキャン 環境 ▷ C/C++ 一般 ▷ タスク・リポジトリー ビルダー プロジェクト参照 実行/デバッグ設定	<ul> <li>※ Tool Settings ・ ビルド・ステップ ・ ビルド・ステップ ・ ビルド成果物 ・ パイナリー・パーサー ・ エラー・パーサー         ・         ・ エラー・パーサー         ・         ・         ・</li></ul>	]]

# 3) セクション

Dy-ス     Jy-ス     Jy-ス     Jy-ス     Jy-ス     Jy-ス     Jy-ス     Jy-ス     Jy-ス     Jy-ス     Jy-z     Jy-z	ルタ入力	Settings	÷ ج	\$
回リンカー・スクリプトの上書き:       参照         再適用       Import       エクスポート	Uソース (/C++ ビルド Device Settings Tool chain エディター ツールチェーン・バー: ビルド変数 ロギング 低存関係スキャン 環境 C/C++ 一般 タスク・リボジトリー ビルダー プロジェクト参照 実行/デパッグ設定	構成: HardwareDebug [アクテ・ で Tool Settings P ビルド・: Source Compiler Source Compiler Sourc	・イブ ・ 構成の1 ステップ 堂 ビルド成果物 論 バイナリー・パーサー ② エラー・パーサー ステップ 堂 ビルド成果物 論 パイナリー・パーサー ② エラー・パーサー セクション・ビューアー: セクション・ジェーアー: セクション・マッピング  シー (0, 100000000) > INST_RAM  ○ 編, eh_frame_hdr <li>○ 編, eh_frame</li> <li>○ 編, eh_frame</li> <li>○ 編, itrs  ○ 編, tors  ○ 編集, tors  ○ ポーレー  ○</li>	<b>管理</b>
	III		回リンカー・スクリプトの上書き:       参照         再適用       Import	

4) セクション (メモリー領域) (デフォルト)



5) セクション(セクション・マッピング)(デフォルト)

ルタ入力	Settings			<
リソース C/C++ ビルド Device Settings	構成: HardwareDebug [アイ	<i>ウ</i> ティフ <u>]</u>		▼ 構成の管理
Settings Tool chain エディター ツールチェーン・バー ビルド変数 ロギング 依存閣係スキャン 環境 C/C++ 一般 タスク・リポジトリー ビルダー プロジェクト参照 実行/デバッグ設定	<ul> <li>Tool Settings 上レル</li> <li>じibrary Generator</li> <li>Compiler</li> <li>Compiler</li> <li>Scompiler</li> <li>Scompiler</li> <li>Linker</li> <li>入力</li> <li>出力</li> <li>ビクション</li> <li>その他</li> <li>その他</li> <li>アーカイブ</li> <li>Objcopy</li> <li>一般</li> </ul>	ド・ステップ 👚 ビルド成果物 🔜 バイナリ- セクション ×モリー領域 セクション 定葉済みセクション・マッピング: セクション名 .data	- · パーサー ② エラー · パーサー · · マッピンク 予約されたロケーショ _mdata	э> 
4 11		□ リンカー・スクリプトの上書き:	再適用	Import
		[	ОК	キャンセル

6) その他 (デフォルト)



6) その他-その他 (デフォルト)

e <sup>2</sup> プロパティ: Sample_BARE_N	3	
フィルタ入力	Settings	⟨¬ ▼ ¬
<ul> <li>フィルタ入力</li> <li>リソース</li> <li>C/C++ ビルド Device Settings Tool chain エディター ツールチェーン・バー: ビルド変数 ロギング 依存開係スキャン 環境</li> <li>C/C++ 一般</li> <li>タスク・リボジトリー ビルダー プロジェクト参照 実行/デバッグ設定</li> </ul>	Settings         ● Tool Settings       ● ビルド・ステップ       ● ビルド成果物       ● パイナリ         ● ③ Library Generator       ● 共用ライブラリーに対してリンクしな         ● ③ Compiler       ● ダイナミック・ライブラリーに対して         ● ③ Compiler       ● グローバレ・シンボルへの参照を、井         ● ③ Linker       ● ガローバレ・シンボルには常にスペースを書         ● ③ 入力       ● デキスト・セクションとデータ・セク         ● ※ セクション       ● デキスト・セクションを読み出し専用         ● その他       ● リンカーの動作に関する続計を算出し         ● ○ ひjcopy       ● グローバレ・ンスボルが特定を中可能         ● ○ ひjcopy       ● プローバレ・シンボルにおけして「使用         ● 一般       ● 一時的なローカル・シンボルをすべて         ● 可のしへシン・シンボルをすべて       ● マイのローカル・シンボルを書体すく	(・・パーサー ◎ エラー・パーサー     (*) (-Bstatic)     (ワンクする (-Bdynamic)     (相ライブラリー内の定義にパインドする (-Bsymbolic)     引り当てる (-d)     フションを読み出し/書き込み可能に設定する (-N)     相に設定する (-n)     をたわれば疾持する (-noinhibit-exec)     表示する (-stats)     的は書告する (-warn-common)     思されている場合は書告する (-warn-constructors)     皆する (-x)     方る (-x)     う     (*)
< <u> </u>	<ul> <li>リロケータブルは出力を生成し、コン</li> <li>出力ファイルからすべてのシンボル様</li> <li>名入力ファイルのための新規の出力せ</li> <li>既存の形式で出力する(-traditional-1)</li> <li>未使用の入力セクションのガページ・</li> <li>メモリー使用を最遠化する(-no-kee)</li> <li>出力ファイルからデバッガー・シンオ</li> </ul>	ストラクターへの参照を開決する(C++)(-UP) 輸税を除去する(-s) 2クションを作成する(-split-by-file) format) - コレクションを有効にする(-gc-sections) p-memory) 抗レ情報を削除する(-S) 

7) アーカイブ



2-2-4-5. Tool Settings(Objcopy)

フィルタ入力	Settings			<
> リソース A C/C++ ビルド Device Settings Tool chain エディタ	構成: HardwareDebug [アクラ	=ィブ] ステップ	裏物 📾 バイナリー・パーサー 😡 エラー・パー	<ul> <li>▼ (構成の管理) ▲</li> <li>-サー =</li> </ul>
<ul> <li>ツールチェーン・パ</li> <li>ビルド変数</li> <li>ロギング</li> <li>低存関係スキャン</li> <li>環境</li> <li>▷ C/C++ 一般</li> </ul>		コマンド: すべてのオプション:	arm-none-eabi-objcopy -O srec	
> タスク・リポジトリー ビルダー プロジェクト参照 リファクタリング履歴 ・		エキスパート設定: コマンド行 パターン:	\${COMMAND} \${FLAGS} \${OUTPUT_FLAG}\$	{OUTPUT_PREFIX]
?			ОК	キャンセル

# 1) 一般

イルタ入力	Settings	
<ul> <li>&gt; リソース</li> <li>&gt; ロジース</li> <li>&gt; C/C++ ビルド</li> <li>Device</li> <li>Settings</li> <li>Tool chain エディち</li> <li>&gt; ツールチェーン・バ</li> <li>ビルド変数</li> <li>ロギング</li> <li>広存期係スキャン</li> <li>環境</li> <li>&gt; C/C++ 一般</li> <li>&gt; タスク・リポジトリー</li> <li>ビルダー</li> <li>ブロジェクト参照</li> <li>リファクタリング履歴</li> <li>- ・</li> </ul>	構成: HardwareDebug [アクティブ]	<ul> <li>▼ 構成の管理</li> <li>▲ エラー・パーサー</li> <li>▲ エラー・パーサー</li> </ul>
?	ОК	キャンセル

2-2-4-6. ビルド・ステップ (デフォルト)

Ь	Settings 🔶 🔻
と ビルド ice	構成: HardwareDebug [アクティブ] ・ 構成の
hain エディター チェーン・バー:	🛞 Tool Settings 🖉 ビルド・ステップ 🍨 ビルド成果物 🔜 バイナリー・パーサー 🔇 エラー・パーサ
2数 7	ビルド前のステップ
マキャン	
トリー	גנייטעק -
参照	- ビルド後のステップ
グ履歴	
E	
	説明:
•	

## 2-2-4-7. ビルド成果物

ЧЛИ <b>9</b> ДЛ	Settings $(\neg \bullet \bullet$
<ul> <li>マ/C++ ビルド         Device     </li> <li>Cetting:         Tool chain エディター         ツールチェーン・パー:         ビルド変数         ロギング         依存関係スキャン         環境         C/C++ 一般          タスク・リポジトリー         ビルダー         プロジェクト参照         実行/デバッグ設定         </li> </ul>	構成: HardwareDebug [アクティブ] ・ 構成の管理 ● Tool Settings  ● ビルド・ステップ ● ビルド成果物  ● バイナリー・パーサー ③ エラー・パーサー
(?)	OK         キャンセル

2-2-4-8. バイナリー・パーサー (デフォルト)

ルタ入力	Settings	<
リソース C/C++ ビルド Device Settings Tool chain エディター	構成: HardwareDebug [アクティブ]	<ul> <li>▼ 構成の管理…</li> <li>→ パーサー</li> </ul>
ッールテェージ・ハー: ビルド変数 ロギング 依存関係スキャン 環境 C/C++ 一般 タスク・リポジトリー ビルダー プロジェクト参照 リファクタリング履歴 実行/デバッグ設定	パイナリー・パーサー:         Image:	上へ移動 下へ移動
) 	ОК	キャンセル

## 2-2-4-8. エラー・パーサー (デフォルト)

イルタ入力	Settings	<
ロソソース C/C++ ビルド Device	構成: HardwareDebug [アクティブ]   ▼	構成の管理
1001 chain エティター ツールチェーン・バー:	🛞 Tool Settings 🎤 ビルド・ステップ 🍨 ビルド成果物 🗟 バイナリー・パーサー 👰 エラー・	パーサー
ヒルド変数	🖉 🗢 GNU Assembler Error Parser	追加
山テンン 依存関係スキャン	🖉 🖛 GNU gmake Error Parser 7.0	A=./JH
環境	🖉 🏍 GNU Linker Error Parser	編集
C/C++ 一般		削除
タスク・リポジトリー	☐ ← GHS C/C++ Error Parser	133192
ビルダー	🔲 🖛 GNU gmake Error Parser 6.0 (Deprecated)	上へ珍動
プロジェクト参照	🔲 🕽 🛏 IAR Error Parser	
リファクタリング履歴	🔲 🚈 Microsoft Visual C Error Parser	下へ移動
実行/デバッグ設定	■ �=Renesas C/C++ エラー・パーサー	
	🔲 🕽 🖛 Renesas Make エラー・パーサー	
• III	Renesas RXC エラー・パーサー	
?	ОК <b>‡</b> т	ンセル

2-2-5. C/C++ビルド (Tool Chain エディター) (デフォルト)

e <sup>2</sup> プロパティ: Sample_BARE_F	14F	
フィルタ入力	Tool chain エディター	← → ⇒ → →
▷ リソース ▲ C/C++ ビルド Device Settings	構成 [HardwareDebug [アクティブ]	▼ 構成の管理
Tool chain エティター ツールチェーン・バー: ビルド変数 ロギング	▼ 互換 toolchain のみ表示 現在の toolchain、KPIT GNUARM-NONE-EABI Toolchain	
14.1+ 闽1th スキャン 環境 ▷ C/C++ 一般 ▷ タスク・リポジトリー	現在のビルダー: Builder	-
ビルダー プロジェクト参照 リファクタリング履歴 実行/デバッグ設定	Library Generator Compiler Assembler Linker Objcopy	▲ ジールの選択 = +
4	デフォルトの復元(T)	適用(L)
?	ОК	キャンセル

2-2-6. C/C++ビルド (ツールチェーン・バージョン) (デフォルト)

<sup>2</sup> ] フロバティ: Sample_BARE_I	13	
フィルタ入力	ツールチェーン・バージョンの変更	
▷ リソース		
⊿ C/C++ ビルド	プロジェクト名: Sample_BARE_M3	
Device	ツールチェーン名 · KRIT CNU ARM_NONE-EABI Toolchain	
Settings		
1001 chain エティター	現在のバージョン: v16.01	
ビルド変数	有効かパージョン: 16.01	
ロギング		
依存関係スキャン		
環境		
▷ C/C++ 一般		
▷ タスク・リポジトリー		
ビルダー		
プロジェクト参照		
美行/デバック設定		
	デフォルトの復示	ī(T) 適用(L)
4		
(?)	ок	キャンセル

2-2-7. C/C++ビルド(ビルド変数)(デフォルト)

e <sup>2</sup> プロパティ: Sample_BARE_F フィルタ入力	AF ビルド変数					
> リソース <ul> <li>C/C++ ビルド Device Settings</li> </ul>	構成: HardwareD	ebug [アクテ	イブ]		•	横成の管理
Settings Tool chain エディター ツールチェーン・パー: ビルド変数 ロギング 依存関係スキャン 環境 ▷ C/C++ 一般 ▷ タスク・リポジトリー ビルダー プロジェクト参照 リファクタリング履歴 実行/デパッグ設定	名前	タイプ	<b>値</b>			<b>追加</b> 編集 削除
۰				デフォルトの復元(T)		適用(L)
?				ок	<b>+</b> 7	ンセル

2-2-8. C/C++ビルド (ロギング) (デフォルト)

フィルタ入力	ロギング	↓ ↓ ↓ ▼
<ul> <li>&gt; リソース</li> <li>△ C/C++ ビルド Device Settings Tool chain エディター ツールチェーン・バー: ビルド変数 ロギンク 依存関係スキャン 環境</li> <li>&gt; C/C++ 一般</li> <li>&gt; タスク・リポジトリー ビルダー プロジェクト参照</li> </ul>	ビルドのロギングを有効にする(L) ログファイルへ場所(F): N:¥UsrAp¥C_H28_AICHI¥RZT1¥Sample 必要に応じてチェック	e_e2¥RZT1_Sample_BARE¥.metadat
()) 実行/デバッグ設定 ( )	デフ ・ のK	'ォルトの復元(T) 適用(L ・ ・ キャンセル

2-2-9. C/C++ビルド(依存関係スキャン)(デフォルト)

フィルタ入力	依存関係スキャン	$\langle \neg \bullet \neg \neg \bullet $
<ul> <li>&gt; リソース</li> <li>∠ C/C++ ビルド Device Settings Tool chain エディター ツールチェーン・パー: ビルド変数 ロギング     <li><b>び</b>存開係スキャン 環境     <li>▷ C/C++ 一般     <li>タスク・リポジトリー ビルダー     <li>プロジェクト参照 リファクタリング履歴 実行/デバッグ設定</li> </li></li></li></li></ul>	<ul> <li>依存関係スキャン</li> <li>○ 依存関係をスキャンしてプロジェクトをビルド</li> <li>● 低存関係をスキャンしてプロジェクトをビルド</li> <li>○ 既存の依存関係を使用してプロジェクトをビルド</li> </ul>	
4	デフォルト	トの復元(T) 適用(L)
?	ОК	キャンセル

## 2-2-10. C/C++ビルド (環境) (デフォルト)

ノイルタ入力	環境			<b>\</b>	
▷ リソース					
▲ C/C++ ビルド Device Settings	構成: HardwareDebug [アクティブ] ・ 構成の管理				
Tool chain エディター ツールチェーン・バー:	設定する環境変数				追加
ビルド変数	変数	値	由来	*	RF
ロキング	AMS_KEEP_FILE	\${synergyKeepFile}	ユーザー:構成		JE1/(
低存関係スキャン	AMS_LICENSE_PATH	\${synergyLicenseFile}	ユーザー:構成		編集
境現	CONFIGDIR	\${workspace_loc:/\${	ビルド・シス	=	
> C/C++ 一般	CWD	N:¥UsrAp¥C_H28_AI	ビルド・シス		削际
▶ タスク・リポジトリー	GCC_VERSION	5.2-GNUARM-NONE	ビルド・シス		定義解除
ビルター プロミュクト券昭	PATH	C:¥PROGRA~1¥KPIT	ビルド・シス		
フロジェクト参照	PWD	N:¥UsrAp¥C_H28_AI	ビルド・シス		
リファクタリンク履歴 実行/デバッグ設定	TOINSTALL	C-YPROGRA~1¥KPIT	ビルド・シス	-	
关11/7/19/21222	◎ ネイティブ環境へ変数を	追加			
	◎ ネイティブ環境を指定さ	れた環境と置換			
4		デフ	'ォルトの復元(T)	適用(L	.)
2			OK	キャンセル	

2-2-11.【説明省略】C/C++ 一般(デフォルト)
2-2-12.【説明省略】タスク・リポジトリー(デフォルト)
2-2-13.【説明省略】ビルダー(デフォルト)
2-2-14.【説明省略】プロジェクト参照(デフォルト)
2-2-15.【説明省略】リファクタリング履歴(デフォルト)

2-2-16. 実行/デバッグ設定

1)新規登録

e <sup>2</sup> プロパティ: Sample_BARE_I	13	
フィルタ入力	実行/デバッグ設定	← → □ → ▼
▷ リソース ▲ C/C++ ピルド Device	このページで現在選択されているリソースに関連する起動構成を智 'Sample_BARE_M3' の起動構成(F):	言理できます。
Settings		新規(N)
Tool chain エディター ツールチェーン・バー:		複製(P)
ビルド変数 ロギング		編集(E)
依存関係スキャン		= 削除(L)
環境 ▷ C/C++ 一般 ▷ タスク・リポジトリー ビルダー プロジェクト参照 実行/デバッグ設定		
< <u> </u>	ОК	キャンセル

2)構成タイプの選択

●     構成タイプの選択     □	×
作成する構成のタイプを選択(S):	
C/C++ アプリケーション C <sup>™</sup> GDB Simulator Debugging (SH, RH850)	
Renesas GDB Hardware Debugging	
Renesas Simulator Debugging (RX, RL78)	
ОК #ヤンセル	

3) 起動構成プロパティの編集(メイン) (デフォルト)

e <sup>2</sup> 構成の編集
起動構成プロパティの編集
名前(N): Sample_BARE_M3 HardwareDebug
( メイン 参 Debugger ) ► Startup □ 共通(C) ↓ ソース
プロジェクト(P):
Sample_BARE_M3 参照(B)
C/C++ アプリケーション:
HardwareDebug¥Sample_BARE_M3.x
変数(V) プロジェクトの検索(H) 参照(R) ≡
起動前に必要に応じてビルド
ビルド構成: Use Active 🔹
◎ 自動ビルドを有効にする ◎ 自動ビルドを無効にする
の ワークスペース設定の使用     ワークスペース設定の構成…     ・
前回保管した状態に戻す(V) 適用(Y)
OK         キャンセル

# 4) 起動構成プロパティの編集(Debugger[GDB Settings])

e <sup>2</sup> 構成の編集
起動構成プロパティの編集 J-Link ARM
名前(N): Sample_BARE_M3 Hardy are Debug
□ メイン 登 Debugger ▶ Aartup □ 共通(C) ↓ ソース
Debug hardware: (J-Link ARM  Target Device: R75910018_M3 GDB Settings Connection Settings デバッグ・ツール設定 R7S910018_M3
GDB 接続設定:
<ul> <li>◎ ローカル GDB サーバーを自動起動 ホスト名または IP アドレス: localhost</li> <li>◎ リモート GDB サーバーへ接続 GDB ポート番号: 61234</li> <li>ADM ポート番号: 61236</li> </ul>
GDB コマンド: feclipse_home}/DebugComp/arm-none-eabi-gdb
□ 詳細モードを有効にする
前回保管した状態に戻す(V) 適用(Y)
OK         キャンセル

5) 起動構成プロパティの編集(Debugger[Connection Settings])

こ動構成プロパティの編集	
	Jor
高前(N): Sample_BARE_M3 HardwareDebug	
🗎 メイン (参 Debugger 🔪 🕨 Startup) 🥅 共通(C) )	<b>レ</b> ソース
ebug hardware: J-Link ARM 🔹 Target De	evice: R7S910018_M3
GDB Settings Connection Settings デバッグ・ツー	ル設定
⊿ J-Link	
Туре	USB
J-Link Serial	(Auto)
設定ファイル	\${workspace_loc:¥\${ProjName}}¥\${LaunchConfigNan
Script File	
Low Power Handling	No
▲ Interface	
Туре	SWD
Speed (kHz)	Auto
▲ JTAG Scan Chain	
Multiple Devices	いいえ
IRPre	0
DRPre	0
⊿ 接続	
レジスターの初期化	いいえ
接続時にリセット	いいえ
実行前にリセット	いいえ
ID コード	FFFFFFFFFFFFFFFFFFFFFFFFFFFFF
接続時にリセット状態を維持する	いいえ
⊿ SWV	
Core clock (MHz)	0
< [	
	前回保管した状態に戻す(V) 適用(Y)
٢	
(?)	OK キャンセル

6) 起動構成プロパティの編集(Debugger[デバッグ・ツール設定])

	X.
前(N): Sample_BARE_M3 HardwareDebug	
メイン 🏇 Debugger 🕒 Startup 🥅 共通(C) 🎚	a V-Z
bug hardware: J-Link ARM 🔹 Target Dev	rice: R7S910018_M3
DB Sottings Connection Sottings デバッグ・ソール	設定
DB Settings Connection Settings 27(99197)	/52.42
テノオルト 10 ノアイル名を使用	t (aclinea hama) VinternalVIaFilaaVDZVDZC010010 M
10 ファイル名	\${ecipse_nome}+internal+toPiles+KZ+K/S910018_M
General Debug	1.11.2
Reset After Reload	
RTOS Integration in Debug view	0.002
エンティアン	リトル・エンティアン
	1.11.2
フラッシュ・フレークホイントを使用する	
シミュレーションで計画する Flach	VIVIZ
	1.11.) 7
WorkBam Start	020
WorkRam End	0×0
	0×0
CEI End	0×0
Semihosting	0.0
Semihosting breaknoint address	
Seminoscing breakpoint address	
	III
	r r
	前回保管した状態に戻す(ハ) 適田(ハ)

\_

7) 起動構成プロパティの編集(Startup)

					) (	2
前(N): Sample_BARE_M	3 HardwareDebug					
) メイン (参 Debugger 🤇	Startup 🔲 共通(C)	<b>☞</b> ソース				
初期化コマンド IJセットと遅延(秒):   Halt	3					
イマージレンデルをロー	ĸ				Ŧ	
コマイルタ	ロード・タイプ	オフセット	接待時			
✓ プログラム・バイナ	··· イメージとシンボル	0	Yes		追加	
					編集	
					除去	Ξ
ランタイム・オプション つプログラム・カウンター マプレークポイント設定が 同 再開 コマンドを実行	-設定先(16進): t: main			こ応じて設定		
			前回保管した	犬態に戻す(V)	適用(Y)	



8) 起動構成プロパティの編集(共通)

動伸成ノロハナイの無	集		1
			200
前(N): Sample_BARE_M3 H	lardwareDebug		
〕メイン (参 Debugger ( ▶	Startup 🦳 共通(C) 🌾 🏷	ノース	
保管 の ローカル・ファイル(0)			
◎ 二 リジル リソ いい(=) ◎ 共用ファイル(H):	¥Sample_BARE_M3		参照(B)
お気に入りメニューに表示(R	)		
□ ☆デバッグ		◎ デフォルト - 継承(U) (SJIS)	
		④ その他(E) UTF-8	•
			۲
		「UTF-8」を選択	J
標準入出力			
標準入出力 ▼コンソールに割り当て(A)	(入力に必要)		
標準入出力 ▼ コンソールに割り当て(A) ■ 入力ファイル(F):	(入力に必要)		
標準入出力 ▼ コンソールに割り当て(A) ■ 入力ファイル(F):	<b>(入力に必要)</b> ワークスペース		変数
標準入出力 ▼ コンソールに割り当て(A) ■ 入力ファイル(F):	(入力に必要) ワークスペース		交数
標準入出力 ▼ コンソールに割り当て(A) ■ 入力ファイル(F):	(入力に必要) ワークスペース ワークスペース(W)	. ファイル・システム 参照(R)	变数 变数
標準入出力 ▼ コンソールに割り当て(A) □ 入力ファイル(F): □ 出力ファイル(L):	(入力に必要) ワークスペース ワークスペース(W)		変数
標準入出力 ✓ コンソールに割り当て(A) □ 入力ファイル(F): □ 出力ファイル(L): □ 追加(P) ] バックグラウンドでの起動(	(入力に必要) ワークスペース ワークスペース(W) K)		変数 変数
標準入出力 マコンソールに割り当て(A) □入力ファイル(F): □出力ファイル(L): □追加(P) ]バックグラウンドでの起動(	(入力に必要) ワークスペース ワークスペース(W) K)	. ファイル・システム 	変数
標準入出力 ダ コンソールに割り当て(A) 「入力ファイル(F): 「出力ファイル(L): 」追加(P) リバックグラウンドでの起動(	(入力に必要) ワークスペース ワークスペース(W) K)		<u>変数</u> 変数 変数 道用(Y)
標準入出力 ▼ コンソールに割り当て(A) ■ 入力ファイル(F): ■ 出力ファイル(L): ■ 追加(P) ■ パックグラウンドでの起動(	(入力に必要) ワークスペース ワークスペース(W) K)		変数 変数 変数 適用(Y)

\_

9) 起動構成プロパティの編集(ソース) (デフォルト)

構成の編集	
己動構成プロパティの編集	- The second sec
前(N): Sample_BARE_M3 HardwareDebug	
📄 メイン 🏇 Debugger 🌘 Startup 🔲 共通(C) 👰	y-X
ソース・ルックアップ・パス(0):	
▷ 🗁 デフォルト	〕 追加(A)
	編集(E)
	除去(M)
	上へ(P)
	下へ(D)
	デフォルトの復元(U)
■パス上の重複ソース・ファイルを検索(H)	
	前回保管した状態に戻す(V) 適用(Y)
?	OK         キャンセル

- 3. サンプルプロジェクト「Sample\_BARE」をビルドする。
  - 3-1. コア【R4F】 側サンプル「Sample\_BARE\_R4F」をビルドする。
    - 1) すべてビルド

アイル(F) 編集(E) タース(S) マー ①プロジェク 「Sample BA	リファクタリング(T) ナビゲート(N) 横 へ名 RE R4F1 クリック	<ul> <li>菜(A) クロジェクト(P) Renesas Views</li> <li>プロジェクトを開く(E)</li> <li>プロジェクトを開くる(S)</li> <li>Open Synergy Configuration</li> </ul>	実行(R) ウィンドウ(W) ○ ▼ Q ・ なデバック
Sample_BARE_M3 Sample_BARE_M3 Sample_BARE_R4F [Har Sample_BARE_R4F [Har R アーカイブ Includes Common Commo	I ● I/***     2 プロジェクト     3 「すべてビルド」選択     3 「すべてビルド」選択     10 // 〈基板〉 MF     10 //     11 // 〈Debug> リ     12 //     13 // 注意事項 CMT     14 // 「     15 //     16 //     17 #include "main     18 #include "Umoni     19     20 ● //     21 // define     22 //     24 ● //     25 // インボート     17     16     1/     15     1/     16     1/     10	<ul> <li>         ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・</li></ul>	Ctrl+B W) , Alt+Q Alt+D ,
e [] }	<ul> <li>問題 </li> <li>タスク</li> <li>コン ※</li> <li>CDT ビルド・コンソール [Sample_BARE_</li> <li></li> </ul>	ブロ 🔋 Mem 🐚 スタ 🧇 🔑 😚 🗐 🔛 🚮 F [R4F]	スマ ☆ デ/ □ □ = ■   d □ • d • ^C •

2) ビルド結果

CDT ビルド・コンソール [Sample_BARE_R4F]	
'Building target:' arm-none-eabi-ld -o "Sample_BARE_R4F.x" 'Finished building:'	'@"N:/UsrAp/C_H28_AICHI/RZT1/Sample_e2/RZT1_Sample_I
'Invoking: Objcopy' 'Building target:' arm-none-eabi-objcopy -O srec Sample_ 'Finished building target:'	BARE_R4F.x "Sample_BARE_R4F.mot"
'Build complete.'	「ビルドが完」しました」確認
15:33:01 ビルドが完了しました(所要時間 45s.344m	n5)

3-2. コア【M3】側サンプル「Sample\_BARE\_M3」をビルドする。

1) すべてビルド

ファイル(F) 編集(E) ソース(S) いプ(H)	リファクタリング(T) ナビゲート(N) 検索(A) フロジェクト(P) Renesas Views 実行(R)	ウィンドウ(W)
		0.0.
(1)プロジェクト名	W フロシェクトを閉じる(5) Open Synergy Configuration	
Sample BARE N	[3] クリック	なテノシシ
		rl+B □ □
	1	******
Sample_BARE_M3 [Have		
▷ 総 バイナリー	(2)プロジェクト ワーキング・セットのビルド(W)	•
▷ 毘 アーカイブ	③「すべてビルド」選択 <sup>クリーン(N)…</sup>	
⊳ 🗊 Includes	目動的にビルド(M)	E
🔺 🐸 src	7 // <cpu> RZ/T1 ( 8 // <mod> main.c Make ターゲット</mod></cpu>	↓ 150MHz)
common	9 // 《基板》 MP-RZT1 Repease Quick Settings A	lt+o
🖻 🔁 Renesas		It+Q
b 👝 sample	11 77 KUEBUGS リトルエンナ 12 77 KUEBUGS リトルエンナ 12 77 KUEBUGS リトルエンナ 12 77 KUEBUGS リトルエンナ 12 77 KUEBUGS リトルエンナ	IC+D
b 👝 startup_GCC	13 // 注意事項 CMT0ユニット C/C++ Index	•
👂 😝 HardwareDebug	14 // R4F側の( ゴロパティ/D)	\$3.
a custom.bat		******
📄 makefile.init	17 <b>#include "main.h</b> "	
Sample_BARE_M3 Harc	18	
Sample_BEAR_M3 Harc	19 ビ//===================================	
Sample_BEAR_M3 Harc	21 //	
Sample_BARE_R4F	<pre>22 extern void (* const vectors_tbl[])(void);</pre>	
	23 24	
	25 //* グローバル変数宣言	
	26 //	
	27 uint32_t LoopCnt = 0; 28	-
	۲. III	•
	💛 🕐 👔 🔛 🗮 📑 🖬 🔤 🖬 🖬 🖬	🖬 🖌 🛄 🔺 VC 🔸
v <u></u>		
		P.
	書き込み可能 スマート挿入 3:1	L

2) ビルド結果

	우 순 😨 📰 🖅 = 🚉 📑 🔫 🔽 🛪 ^0
DI Eルト・コンソール [Sample_B/	ARE_M3J
'Invoking: Linker'	
'Building target:'	
arm-none-eabi-ld -o "Sample B	ARE M3 x" -T"N·\UscAn\C H28 ATCHT\R7T1\Sample e2\R7T1 Sample
'Einiched huildings'	and 1913 A 1914 (0313) (C_120_ATCH1 (M211 (Sumpto_C2 (M211_Sumpto
Finished building.	
I Taught and Obderson'	
Invoking: Objcopy	
Building target:	
arm-none-eabi-objcopy -O srec	Sample_BARE_M3.x "Sample_BARE_M3.mot"
'Finished building target:'	
'Build complete.'	「ビルドが完了しました」 確認
- 4. サンプルプロジェクト「Sample\_BARE」をデバッグする。
  - 4-1. MP-RZT1-01 のデバッグ準備
    - 1) コア【R4F】 側とコア【M3】 側を J-Link BASE のみ使用してデバッグする場合



2) コア【R4F】側をDEFnano、コア【M3】側をROM化してデバッグする場合



4-2. コア【R4F】側のデバッッガ初期起動

4-2-1. J-Link BASE を使用(詳細なデバッグ操作は SEGGER 提供「J-Link USER Guide」を参照)

1)「デバッグの構成」を選択

C - Sample_BARE_M3/src/sample/app/m	nain.c - e2 studio				
ファイル(F) 編集(E) ソース(S) リファクタ	タリング(T) ナビゲー	ト(N) 検索(A) プロジェクト(P) Renesas Views	寒	〒(R) ウィンドウ(W) ヘルプ(H)	
📸 🕈 🔚 👘 🚫 🕶 🚫 🐨 🚫 👘 🔌 🗆			₿₽.	再開(M)	
1 ()プロジークト	々		00	中断(S)	
	'口			終了(T)	
Sample_BAI	RE R4F」ク	リック / /	8-8	切断	
I I	2			TraceX	•
Sample BARE RAE [Hardware0)	3	// RZ/T1/5ループ	2.	ステップイン(I)	
Sample_DARC_RATE [nardwareDA	(2)実行	J	9	ステップ・オーバー(0)	
			.e	ステップ・リターン(U)	
	3 7	パックの構成」選択 🏻 🎬	H =⊅]	指定行まで実行(L)	
			P	ステップ・フィルタの使用(F)	シフト+F5
	10	// Cheburgs II billT'r (T'r)	0	実行(R)	Ctrl+F11
	12	//	to	デバッグ(D)	F11
	13	// 注意事項 CMT0ユニットは使用に注意が必要!!	7	実行履歴(T)	•
	15	//	2	実行(S)	•
	16	//************************************	*	実行構成(N)	
	18	#Include main.n		デバッグ履歴(日)	,
	19 😁	//	=	デバッグ(G)	,
	20	//~ 1.2水一下旦言 //	-	デバッグの構成(B).	
	22	extern void (* const vectors_tbl[])(void)	;	ゴームポイントの切り共同化	Obdus Thurp
	23			ノレークパイノトの切り含え(K)	CU1+>>1+B
	-			11.20 シバイントの切り音ん(L) メリッド・ブレークポイントの切り替う(M)	
	● 問題 ● タスク	🖳 コンソール 🕮 🖂 フロバティー 🍯 Memory U	60	いっていていていていていていていていていい。 監視ポイントの切り替え(W)	
			8	すべてのブレークポイントをスキップ(I)	Ctrl+Alt+B
	Invoking: Linker	-70 [Sample_BARE_R4F]	Sig	すべてのブレークポイントを削除(V)	
	'Building target:			ブレークポイント型(R)	•
	arm-none-eabi-ld	-o "Sampie_BARE_R4F.x" @"N:/UsrAp/C_H28_A:		の(朝い)_1(5)	
			-	VIEV /V(L)	
Sample_BARE_R4F					

● デバック構成 構成の作成、管理、および実行		<u>ب</u>
<ul> <li>○ (C++ アブリケーション)</li> <li>○ (C++ アブリケーション)</li> <li>○ (C++ リモート・アブリケーション)</li> <li>○ Debug-only</li> <li>○ EASE Script</li> <li>○ GDB Simulator Debugging</li> <li>○ GDB Simulator Debugging (SH, RH8</li> <li>○ GDB (N-ドウェア・デバッギング)</li> <li>○ GBB (N-F)</li> <li>○ Renesas GDB Hardware Debugging (RX, I)</li> <li>○ Renesas Simulator Debugging (RX, I)</li> <li>○ Renesas Simulator Debugging (RX, I)</li> <li>○ E Renesas Simulator Debugging (RX, I)</li></ul>	名前(N): Sample_BARE_R4F HardwareDebug          メイン	参照(B) 参照(R) マ マク 適用(Y) 閉じる

## 2) デバッガ起動画面

ファイル(F) 編集	(E) ナビゲート(N) 検 § ▼ ≪ ▼ ≪ 品 ☆	(本(A) プロジェクト(P) Renesa (▶ 11 ■ ぱ 3. ⑦ ☆ i→	is Views 実行(i 同志 武   徽   『	R) ウィンドウ(W) ヘルス 3]]@] Ø  ②  禄 ▼(	ブ(H) ◑ ▾ ▾¦ᢄ	) クイ:	<i>☆</i> • ね • 神 ック・アクセス	- ← ← → → ∰ 0⊡ [%	デバッグ
<ul> <li>              テバック ☆             Sample_B             ▲             ご</li></ul>	ARE_R4F HardwareDeb a_BARE_R4F.x [1] ead #11 (single core) ( () at g_vector_robin.asr esas/e2_studio/Debug(	『 智 む #   ♪ ⇒   ↓   ↓ ug [Renesas GDB Hardware Deb Suspended : シグナル : SIGTRAP n:16 0x0 Comp/arm-none-eabi-gdb (7.8.2)	♀ ▽ □ □ ugging] h:Trace/breakpo	(*)- 変数 🍨 ブ 出けし. 式 🍦 新しい式を追加	… ☆ C ⇒	€ 6	然式 22 ● イ 約 →4 回   中 3 値	■ I 器 P ※ 後   13 ぜ アドレス	€ .
				1					
main.c     S       5     6       7     8       9     10       11     12       13     14       15     16       16     0000000       17     0000000       19     0000000       20     00000010       21     0000001       22     00000012       24     25       26     26	<pre>) g_vector_robin.asm %</pre>	<pre>3 3 3 3 3 3 3 3 5 3 5 5 5 5 5 5 5 5 5 5</pre>	<pre>@ Start+000 @ Start+000 @ Start+000 @ Start+001 @ Start+001 @ Start+001 @ Start+001 @ Start+001</pre>	<ul> <li>98 : リセット</li> <li>94 : 未定義命令</li> <li>83 : ソフトウェア書別込み</li> <li>95 : フリフェクチアボート</li> <li>14 : Reserved</li> <li>18 : IRQ</li> <li>16 : FIQ(NMI)</li> </ul>	*		P プトラ № Sample_BA ▷ ▷ Sample_B/ ▷ ▷ Sample_B/	プロジェ 窓 ■ な RE_M3 RE_R4F [Hard	wareDebu
📴 🗆 🕺 🔊 夕 💹 Cu 丧 Re 🔋 Me 🛞 Pe 🕐 Pe 🧐 Pr 🂱 Re 👒 Tr 🔿 Vis 🖄 AR 🗞 Zu 🖹 問題 🕥 実 🔋 メ 🔋 メ 🖓 🕮									
Sample_BARE R4	F HardwareDebug [Rer	iesas GDB Hardware Debugging]	C:/Renesas/e2	_studio/DebugComp/arm-	none-eabi-gdb	<b>a</b> (7.8.2)		🛃 📮 🕇 📑	▼ ^C ▼
	2.5				J				

4-2-2. DEFnano を使用(詳細なデバッグ操作は Aone 提供「DEFnano\_Vx\_xx.pdf」を参照)

1) サンプル「Sample\_BARE\_R4F」をダウンロードする。

7747U(F) 7-9(D) ¥11(G)					
ダウンロード(D)	View CF	PU RZ/T1(R7S910018)	Advance25.0000 Ix2	4 TOP 0×00000000	SIZ 0×00080000 😱
アップロード(U)	FIC Src	C Mix 🗭 Asm	<b>新</b> ▼	5速 🗾	
ベリファイ(V)		00400			
シンボル読込み(Y)	2	②【ファイル】	- 【ダウンロ	ード】を選択	5
アブソリュートファイル設定(/	N)	=0x00003258			
CPU設定読込み(S)	S	pc, lr, #0x4			
CPU設定登録(R)		=0x0000031F0			
スクリプト実行(1)	EQ	r0,r0,r0			
x() )) ( x()(-)	EQ EQ	r0,r0,r0			
オフライン作業(0)	EQ	r0, r0, r0			
オフライン環境設定(M)	EQ	r0,r0,r0			
終了(E)	0	rU,rU,rU			
• pooooosc: oooooooo	ANDEQ	r0, r0, r0			
. 00000040: 00000000	ANDEQ	r0, r0, r0			
00000044: 00000000 00000048: 0000000		r0,r0,r0			
• 0000004C: 00000000 🦯		2			
• 00000050: 00000000	①Start をクリ	ック			
		00			
SB1 000000000	SLR 💌 🤆 SB2	000000000	🔹 CLR 💌	実行回数 1	7757
	- ((tr) )	▼ Ichar	<b>▼</b> 16j≇		
Go Brg RstR4F RstN	3 Win Reg	Watch Sym	Trace Step 1	CTrac CStep	IntFlg
Go Bro RstR4F RstN	3 Win Reg	Watch Sym	Trace Step	CTrac CStep	IntFlg
Go Bre RstR4F RstN	3 Win Reg	Watch Sym	Trace Step	CTrac CStep	IntFlg
Go Bre RstR4F RstN	3   Win   Reg	Watch Sym	Trace Step I	CTrac CStep	IntFlg Sto
Go Bre RstR4F RstM Start	3   Win   Reg ファイル指定でのダウン	Watch Sym  Info L >□−ド	Trace Step	OTrac   CStep	IntFlg Esc Sto
Go Bree RstR4F RstM Start	3 Win Reg ファイル指定でのダウ: HardwareDebug 、	J Watch Sym Info L >□-ド ▼ 49 Hardv	Trace Step 1	CTrac   CStep	IntFlg Esc Sto
Go Bre RstR4F RstM Gtart DEFnano アブソリュート/ヘキサ シーマ M Sample、 整理 マ 新しいフォルダー	3 Win Reg ファイル指定でのダウン HardwareDebug 、	Watch Sym Info L >□-ド → ↔ Hardv	Trace Step 1 Dg vareDebugの検索 跳 ~ []	OTrac OStep	IntFlg Esc Sto
Go Bre RstR4F RstM Start DEFnano アブソリュート/ヘキザ シーマ M Sample 、 整理 マ 新しいフォルダー RZT1_Sample_	3 Win Reg ファイル指定でのダウン HardwareDebug 、 BARE A 名詞	J Watch Sym Info L >□-ド + +y Hardv	uareDebugの検索	CTrac CStep	IntFlg Esc Sto
Go Bra RstR4F RstM Start DEFnano アブソリュート/ヘキサ シーマ W Sample 、 整理 マ 新しいフォルダー W RZT1_Sample_I W metadata	3 Win Reg ファイル指定でのダウ: HardwareDebug ト BARE 名詞	Watch Sym Info L >□-ド + + Hardv	uareDebugの検索	CTrac CStep	IntFlg Esc Sto
Go Bra RstR4F RstM Start DEFnano アブソリュート/ヘキサ シーマ W Sample 、 整理 マ 新しいフォルダー W RZT1_Sample_I W RZT1_Sample_I Sample_BARE	3 Win Reg 3 Win Reg ファイル指定でのダウ: HardwareDebug ト BARE 名語	Watch Sym Info L >□-ド ↓ ↓ Hardv	Trace Step U Dg □ vareDebugの検索 III ▼ □	CTrac CStep CTrac CStep の 更新日時 2017/0 2017/0	Sto
Go Bra RstR4F RstM Start DEFnano アブソリュート/ヘキサ シーマ M Sample ) 整理 マ 新しいフォルダー RZT1_Sample_ M RZT1_Sample_ M Sample_BARE Sample_BARE	3 Win Reg 3 Win Reg ファイル指定でのダウ: HardwareDebug 、 3ARE 名詞 _M3 _R4F	Watch Sym Info L Info L >□-F + + Hardv src Sample_BARE_R4F,x	Trace Step   1 Dg   □ vareDebugの検索 IIII ▼ □	CTrac CStep CTrac CStep の 更新日に 2017/0 2017/0	IntFlg Esc Sto
Go Bra RstR4F RstM Start	3 Win Reg 3 Win Reg ファイル指定でのダウン HardwareDebug ト BARE 名詞 _M3 _R4F	Watch Sym Info L >□-ド + + Hardv src Sample_BARE_R4F.x	Trace Step   1 Dg   □ VareDebugの検索 III マ □	CTrac CStep CTrac CStep の 更新日に 2017/0 2017/0	IntFlg Esc Sto
Go Bra RstR4F RstM Go Bra RstR4F RstM Start DEFnano アブソリュート/ヘキサ シーマ シーマ Sample 、 整理 新しいフォルダー 原ZT1_Sample_ 	3 Win Reg 3 Win Reg ファイル指定でのダウン HardwareDebug ・ BARE 名詞 _M3 _R4F	Watch Sym Info L >□-ド + + Hardv src Sample_BARE_R4F,x	Trace Step   1 pg   □ vareDebugの検索 III マ □	CTrac CStep CTrac CStep の 更新日程 2017/0 2017/0	IntFlg
Go Bro RstR4F RstM Go Bro RstR4F RstM Start	3 Win Reg 3 Win Reg ファイル指定でのダウ: HardwareDebug ・ BARE 名語 _M3 _R4F	Watch Sym Info L >□-ド ▼ ← ← Hardv src Sample_BARE_R4F.x	Trace Step   1 pg   □ wareDebugの検索 III ▼ □	CTrac CStep	IntFlg Esc Sto
Go Bro RstR4F RstM Go Bro RstR4F RstM Btart DEFnano アブソリュート/ヘキサ マ W Sample 、 整理 マ 新しいフォルダー RZT1_Sample_ Mathing Sample_BARE Sample_BARE Sample_BARE Sample_BARE	3 Win Reg 3 Win Reg ファイル指定でのダウ: HardwareDebug ・ BARE 名語 _M3 _R4F	Watch Sym Info L >□-ド ▼ 47 Hardv src Sample_BARE_R4F.x	Trace Step U Dg wareDebugの検索 III マ □	CTrac CStep	IntFlg
Go Bro RstR4F RstM Go Bro RstR4F RstM Start DEFnano アブソリュート/ヘキサ シーマ ※ Sample 、 整理 ボレいフォルダー RZT1_Sample_ 。 RZT1_Sample_ 。 Sample_BARE 。 Sample_BARE 。 Sample_BARE 。 Script_file 。 src Sample BARE R4E	3 Win Reg 3 Win Reg ファイル指定でのダウン HardwareDebug ト 3ARE 名語 _M3 _R4F bug くく	Watch Sym Info L >□-ド ▼ 4y Hardv src Sample_BARE_R4F.x	Trace Step U Dg vareDebugの検索 IIII ▼ □	CTrac CStep	IntFlg
Go Bro RstR4F RstM Go Bro RstR4F RstM Start	3 ¥in Reg 3 ¥in Reg ファイル指定でのダウ: HardwareDebug → BARE 4 Bug 4 × 状況: 33 共有 更新日時: 2017/06/	Watch Sym Info L >□-ド + + Hardv src Sample_BARE_R4F,x "" 12 15:33	Trace Step   1 pg   □ vareDebugの検索 III ▼ □	CTrac CStep	IntFlg

<RZT1\_Sample\_BARE>-<Sample\_BARE\_R4F>-<HardeareDebug>[Sample\_BARE\_R4F.x]を選択後、 「開く」をクリック

## 2) デバッガ起動画面

Aone DEF	nano & S	erialFlashWriter Ver3	.00A RZ/T1 AHnano Ver1	.00(2016/11/25)		
ファイル(F)	データ(	D) 実行(G) ブレーク	<sup>ラ(B)</sup> 割り込み(I) オプラ	ション(0) ヘルプ(H)	)	
停止割	此 DI PO	○0×00000000 [ 周期	View CPURZ/T1(R7S	910018) Advance25.00	000 Ix24 TOP 0x000000	100 SIZ 0×00003E54 😱 🖕
0000000	<b>. #</b> 21:	•	⊙ Src ⊂ Mix ⊂ A	sm vector robin.as	▼ 高速 ▼	
. 00000000:	21		b loader init1	: Start+0000 :	リセット	
. 00000004:	22		b Undefined_Interrupt	; Start+0004 :	未定義命令	
. 00000008:	23	svc_handler:	b svc_handler	; Start+0008 :	ソフトウェア割り込み	
0000000C:	24		b_prefetch_handler	; Start+000c :	ブリフェッチアボート	
00000010:	25		b Abort_Interrupt	; Start+0010 :	データアボート	
00000014:	26	reserved_handler:	subs pc,lr, <b>#</b> 4	; Start+0014 :	Reserved	
00000018:	27	irg_handler:	b irg_handler	; Start+0018 :	IRQ	
0000001C:	28		b FigHandler_Interrupt	t ; Start+001c :	FIQ(NMI)	
00000020:	29	;				
	30	END				
	31	; End of File				
• SB1 00000	0000	CLR	✓ C SB2 00000000		2LR ▼ 実行回数 1	 
• SB1 00000	0000	CLR	<ul> <li>C SB2 000000000</li> <li>↓(なし) ↓ ch</li> </ul>	Iar I	CLR ▼ 実行回数 1 6進 ▼ <b>周期</b>	<u>אראראראראראראראראראראראראראראראראראראר</u>
SB1 00000	0000	CLR	<ul> <li>C SB2 000000000</li> <li>(なし) <ul> <li>ch</li> </ul> <li>(なし) <ul> <li>ch</li> </ul> </li></li></ul>	iar I	CLR 💌 実行回数 1 6進 💌 🗖 周期	<u>איזען</u> זיראַגע
SB1 000000	0000	CLR .R4F RstM3 Win	<ul> <li>         ・ SB2 00000000         ・         ・         ・</li></ul>	var ▼ 1 Sym Trace St	CLR ▼ 実行回数 1 6進 ▼ 周期 ep CTrac CStep	<u>yyh</u> 7°l
SB1 000000 Go Bri 88_sys =0x00	0000	CLR .R4F RstM3 ₩in 9_sys =0x0000000 R10	▼ ○ SB2 000000000   ▼ (なし)   ■ Reg Watch S   _sys=0x0000000 R11_sys=	iar v 1 Sym Trace St 0x0000000 R12_sys=1	CLR ▼ 実行回数 1 6進 ▼ <b>周期</b> ep CTrac CStep 0x0000000 SP_sys =0x0	<u>9777°L</u> IntFlg 0000000 LR_sys =0x00

4-3. コア【M3】側のデバッッガ初期起動

4-3-1. J-Link BASE を使用(詳細なデバッグ操作は SEGGER 提供「J-Link USER Guide」を参照)

1)「デバッグの構成」を選択

ファイル(F) 編集(F) ソース(F) リファクタリソク(T) ナビタート(N) 操集(A) プロジェクト(P) Renease Views( 星(R)) ファンドラ(N) ヘルブ(H)         ① プロジェクト名         「Sample_BARE_M3] クリック         Image: Sample_BARE_M3[LandwareD)	e <sup>a</sup> C - Sample_BARE_R4F/src/monitor/g_ve	ector_robin.asm - e2 studio		_		
D プロジェクト名     Sample_BARE_M3] クリック     m 図     # File Name : E.**     Sample_BARE_M3[Hardware:     **     **     Sample_BARE_M3[Hardware:     **	ファイル(F) 編集(E) ソース(S) リファク	タリング(T) ナビゲート(N) 検索(A) プロ	コジェクト(P) Renesas Views	実行	テ(R) ウィンドウ(W) ヘルプ(H)	
Contripute_transformer     Sample_BARE_M3 [Hardwares]     Sample_BARE_M3 [Hardwares]     Sample_BARE_M3 [Hardwares]     Sample_BARE_M3 [Hardwares]     Sample_BARE_M4     Sample_BARE_M3     Sample_BARE_M3     Sample_BARE_M3     Sample_BARE_M4     Sample_BARE_M4     Sample_BARE_M3     Sample	①プロジェクト名 Sample BARI	$\mathbf{E} \mathbf{M}_{3} \mathbf{D}_{3} \mathbf{D}_{3} \mathbf{D}_{3}$			再關(M) 中断(S) 終了(T)	
Sample_BARE_M3 [Hardwared				2-3	切断	
10       10       20       20       20       20       20       20       20       20       11	Sample_BARE_M3 [HardwareDe	2 3 (2)実行 (3)「デバッグの構成	成」選択		TraceX ステップイン(I) ステップ・オーバー(O) ステップ・リターン(U) 指定行まで実行(L) フテップ・フィルタの使用(E)	°√7 b±55
11       ●       Entry point for the Reset handler         14       ●       Entry point for the Reset handler         15       ●       Undefined_Interrupt         16       ●       Undefined_Interrupt         17       svc_handler:       ●       svc_handler         18       svc_handler:       ●       svc_handler         19       >       perfetch_handler       svc_handler         20       reserved_handler:       >       bing_handler         21       reserved_handler:       >       fighandler         22       irq_handler:       >       fighandler         23       e		10 11 @		00 th	実行(R) デバッグ(D)	Ctrl+F11 F11
18       svc_handler:       b       svc_handler         19       b       prefetch_handler         20       reserved_handler:       b inq_handler         21       reserved_handler:       b inq_handler         22       inq_handler:       b inq_handler         23       e       ind indier         24       e       indien         25       .end       e         26       @ End of File       indien         27       indient       indient         27       indient       indient         28       .end       file         29       indient       indient         20       indient       indient         21       indient       indient         22       indient       indient         23       indient       indient         24       indient       indient         25       .end       indient         26       @ End of File       indient         27       indient       indient         28       indient       indient         29       indient       indient         20       indien       indient<		13 @	the Reset handler b loader_init1 b Undefined_Interrupt		実行履歴(T) 実行(S) 実行構成(N)	* *
1       In Q_initial:       b fiqlinalder_Interrupt         23       b fiqlinalder_Interrupt         24       @=         25       .end         26       @ End of File         27       @         28       @ Image: State of the s		18   svc_handler:     19   20     21   reserved_handler:     22   irg_handler:	<pre>b svc_handler b prefetch_handler b Abort_Interrupt subs pc,lr,#4 b icc handler</pre>	•	デバック履歴(H) デバッグ(G) デバッグの構成(B)	•
		23 24 @ 25 .end 26 @ End of File	b FiqHandler_Interrupt	0 0 0	ブレークポイントの切り替え(K) 行ブレークポイントの切り替え(L) メソッド・ブレークポイントの切り替え(M)	Ctrl+シフト+B
CDT ビルド・コンソール [Sample_BARE_M3] *BUILIC complete・ 15:47:31 ビルドが完了しました(所要時間 20s.941ms)		() (1) (1) (1) (1) (1) (1) (1) (1) (1) (	] プロパティー 🔋 Memory Us	2	転換パインドの切り皆え(W) すべてのブレークポイントをスキップ(I) すべてのブレークポイントを削除(V) ブレークポイントを削除(V)	Ctrl+Alt+B
「Build complete.」 15:47:31 ビルドが完了しました(所要時間 20s.941ms)		CDT ビルド・コンソール [Sample_BARE_I	м3]	9	外部ツール(E)	•
		Build complete. 15:47:31 ビルドが完了しました(所要時間 20:	s.941ms)			_
CS Sample BARE M3	Sample BARE M3	•	£1			•

● アハック 備成 構成の作成、管理、および実行		Ť.
C C/C++ アプリケーション C C/C++ アプリケーション C C/C++ リモート・アプリケーシ Debug-only EASE Script GDB OpenOCD Debugging GDB Simulator Debugging (St GDB /\-ドウェア・デ/(ッギン GHS Local C/C++ Launch IAR C-SPY Application Java アプリケーション Java アプリケーション Java アプリケーション Java アプリケーション A Renesas GDB Hardware Debu	名前(N): Sample_BARE_M3 HardwareDebug ③ メイン 参 Debugger ▶ Startup ↓ ソース □ 共通(C) プロジェクト(P): Sample_BARE_M3 C/C++ アプリケーション: HardwareDebug¥Sample_BARE_M3.x 変数(V) プロジェクトの検索(H) 起動前に必要に応じてビルド ビルド構成: Use Active ● 自動ビルドを有効にする ● 自動ビルドを無効にする ● フークスペース設定の使用 フークスペース設定の構成 ① 「Sample_BARE_M3」 選択 ② 「デバックグ」ク デバッグ(D)	参照(B) 参照(R) ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・

## 2) デバッガ起動画面

R = 10 µ H _ Comple_RARE_M2/cm/startum_CCC/start.come3_studie							
「「ファイル(F) 編集(F) +ビゲート(N) 絵素(A) プロジェクト(D) Renesas Views 宇行(R	) ウィンドウ(W) ヘルプ(H)						
	クイック・アクセス 昭 昭 昭 な デバック						
存 デバッグ 🗴 🍓 🛰 🔻 🦠 💷 🐨 😭 🍪 🖉 🐇 🖉 📩 🖘	🗱 変数 💊 ブ 闘 レ 🎎 C 🛋 モ 候 式 🛿 🥵 イ 🔲 I 묾 P 🖳 🗖						
Sample_BARE_M3 HardwareDebug [Renesas GDB Hardware Debugging]	約 →4 🖻 🕂 💥 🔂 🔂 👳 ▽						
a 🔐 Sample_BARE_M3.x [1]	式 タイプ 値 アドレス ^						
🔺 🧬 Thread #1 1 (single core) (Suspended : シグナル : SIGTRAP:Trace/breakpoi	🖕 新しい式を追加						
<pre>stack_init() at start.asm:56 0x139c</pre>							
📕 C:/Renesas/e2_studio/DebugComp/arm-none-eabi-gdb (7.8.2)							
📕 GDB server							
🖻 main.c 🕼 g_vector_robin.asm							
47 * Function Name : loader_init1	* 📄 🛱 💱 🌫						
48 * Description : Initialize sysytem by loader program 49 * Arguments : none	Sample_BARE_M3 [HardwareDebug						
50 * Return Value : none	Bample_BARE_R4F						
51 ************************************	***********						
52 .type _PowerON_Reset, afunction 53 PowerON Reset:							
54 stack_init:							
55 /* Stack setting */							
57 0000139e msr msp,r0							
58	1						
59 000013a2 ldr r0,=_sys_stack+0x800	≡ □						
61							
62 /* use psp */							
63 000013a8 movs r0,#2 64 000013aa msr control r0							
65							
66 data_init:							
6/ /* Initialize variables has initialized value. */							
69 000013b0 ldr r1, =_data_start							
70 00001760 1J0 J-4J	۰						
및 コ 23 刻 タスク 吸 Cur 後 Re							
KI KI							
Sample_BARE_M3 HardwareDebug [Renesas GDB Hardware Debugging] C:/Renesas/e2_s	tudio/DebugComp/arm-none-eabi-gdb (7.8.2)						
	i						

- 4-4. デバッグに関する特記事項
  - J-Link BASE を使用した場合、コア【R4F】・【M3】を同時にデュアルコアデバッグすることはできません。
     必ず、片側のコアを ROM 化してデバッグ作業を進めて下さい。

デバッグ手順の一例として、

①R4F側で使用する I/O の初期化のみを実施する「初期化プログラム」を作成する。
 (M3 側のリセット解除処理を含む)
 ②上記の「初期化プログラム」を ROM 化して、コア【M3】側のデバッグを進め完成させる。
 ③上記の「コア【M3】側プログラム」を ROM 化して、コア【R4F】側のデバッグを進め完成させる。
 ④総合検証で「②<->③」のデバッグ作業を進めて完成品にする。

- Ijet や DEFnano を使用してのデバッグに関する共通事項は、デバッグ時はプログラムコードを RZ/T1の内蔵 RAM にダウンロードしてデバッグ作業を進めます。
   特に JTAG デバッガを使用したサンプル例ですとダウンロード時にシリアル FROM に書き込み後、 アプリ側でロードさせるコードが含まれていますが、MP-RZT1-01を使用する場合は不要です。
- 3) MP-RZT1-01 基板は、オンボードのシリアル FROM にローダーとデバッガ用ファームが書き込み済 みで出荷しています。
- 4) ROM 書き込みに関しては、5項をご覧下さい。

## 5. ROM 書き込み

5-1. DEFnano を使用して ROM 書き込みする。

【オプション】-	【フラッシュ ROM ライタ】	を選択
----------	-----------------	-----

亭止 書記 080000	EI P	C 0×2008000	0 [	周期 「 王	T Vie	w Cl €Src	PURZ/A ⊂ Nix		環境的 CPU的	建(E) 定(C)					00800	00 SIZ ();	×200C7	DE3 Tar	getMen	nory	
20080000:	35	LDR	pc,	=Reset	handi	er		1	フラッ	シュRO	DM 5-	(夕(F)	>								2
20080004:	30	LUN	PC,	= Undef i = Swc. ha	ned_h	andler		~				- (- )	/		24						
2008000C:	38	LDR	pc,	=Prefet	ch ha	odler			ファー	-11-	ジョン	アッフ	7(V)		1.						
20080010:	39	LOR	PC.	=Abort	handl	er		-	: 51	art+0x	0010 :	デー	タアボ	- 1	_						
20080014:	40	LDR	PC.	Reserv	ed ha	ndler			: 51	art+0x	0014 :	Reser	ved								
20080018:	41	LDR	PC.	=Irg ha	ndler				: \$1	art+0x	0018 :	IRO									
2008001C:	42	LDR	pc,	=Fig_ha	ndler				; 51	art+0x	001C :	FIQ(N	(HI)								
20080020:	43	;*******																			
	44	; SFRO	MC3	建緑した	0-4	-12;	度才情報	1													
	45	;=======																			
	46	Info_tab	le																		
	47	DCD	11	nage\$\$V	ECTOR	TABLE	\$\$Base	1	; \$1	art+0x	0020 :	内蔵	AH	き先の	開始アド	レス					
	48	DCD	11	mage\$\$D	ATA\$\$	Linit			; St	art+0xi	0024 :	内蔵	AH	*先の	除了アド	レス(+1	)				
	49	DCD	ve	ctor_ta	ble				; St	art+0x	0028 :	初期	吃值								
	50	Info_end																			
	51	;*******																			
	52	Literals																			
	53	LTOR	G																		
	54																				
	55	END																			
SB1 000000	00 0		_	- 0.8	-	C 992	Inneen	0000	1				-	te de la	124 1			VEL:	191	DDV .	
	1		-		-	ver	Luna		1			Horn	-		wy l.		_	Thur		Lever C	-
1					-	(なし)		-	char		*	16進	19		周期						
																					1
																					1
																					2
																					12
Go Brea	ak Rst	Mon		Win		Reg	Wato	h	Sym	Tra	ce :	Step	CT	rac	CStep		In	tFla	ProF		
	-		-	-	-		100			-			1				-			and some	

『N¥UsrAp¥C_H28_AICHI¥RZT1¥Sample_IAR¥RZT1_Sample_BARE¥M3¥Debug¥ 参照 の Hex     Start 0x0 end 0x977 Size 0x978     Cortex-R4F(*.mot/*.hex)     『N¥UsrAp¥C_H28_AICHI¥RZT1¥Sample_IAR¥RZT1_Demo_BARE¥R4F¥Debug¥ 参照 の Hex     ひを指     S.     『N¥UsrAp¥C_H28_AICHI¥RZT1¥Sample_IAR¥RZT1_Demo_BARE¥R4F¥Debug¥ 参照     Start 0x0 end 0x38aff Size 0x38b00     告込み個数 0     書込み個数 0     書込み開始     の Hex     し Hex     し Hex     し Hex     し Hex     し	 Z後、 M3側	先頭を☑ Cortex-N		<b></b>	チツール	vシュROM書き込み (hex)	RZ/A1H RZ/T1 Cortex-M3(*.mot/
▼ N¥UsrAp¥C_H28_AICHI¥RZT1¥Sample_IAR¥RZT1_Demo_BARE¥R4F¥Debug¥ Start 0x0 end 0x38aff Size 0x38b00   書込み個数 0   書込み開始	ファイ 冒定す	の Hex ルを指 る。		nple_BARE¥M3¥Debugi 参照	¥Sample_IAR¥RZT1_S Size 0x978	2_H28_AICHI¥RZT14 end 0x977 (*hex)	Start 0x0 Cortex-R4F(*.mot
書込み個数 0 書込み開始 Cortex-R	 ]後、	先頭を☑		no_BARE¥R4F¥Debug¥	¥Sample_IAR¥RZT1_D Size 0x38600	-H28_AICHI¥RZT14 end 0x38aff	Vi¥UsrAp¥ Start 0x0
全消去個数 0     全消去開始     の 1 ex       レ を 指	AF側 ファイ 亡 定す	Cortex-R のHexこ ルを指		書込み開始 全消去開始			書込み個数 0 全消去個数 0









シリアルフラッシュ ROM マップ(MP-RZT1-01)						
ローダ・デバッグ用ファーム	0~1セクター					
	0x0~0x1_FFFF					
アプリケーションエリア	2~15セクター					
Cortex-M3	0x2_0000~0x9_FFFF					
	0xA_000~0xF_FFFF(予備)					
アプリケーションエリア	16~31セクター					
Cortex-R4F	0x10_0000~0x17_FFFF					
	0x18_0000~0x1F_FFFF(予備)					

- 6. 注意事項
  - ・本文書の著作権は、エーワン(株)が保有します。
  - 本文書を無断での転載は一切禁止します。
  - ・本文書に記載されている内容についての質問やサポートはお受けすることが出来ません。
  - ・本文章に関して、ARM 社およびルネサス エレクトロニクス社および SEGGER 社およびエンビテック社 への問い合わせは御遠慮願います。
  - ・本文書の内容に従い、使用した結果、損害が発生しても、弊社では一切の責任は負わないものとします。
  - ・本文書の内容に関して、万全を期して作成しましたが、ご不審な点、誤りなどの点がありましたら弊社まで ご連絡くだされば幸いです。
  - ・本文書の内容は、予告なしに変更されることがあります。
- 7. 商標
  - ・J-Link BASE は、SEGGER 社の登録商標、または商品名称です。
  - ARM Cortex, Thumb および ARM Cortex-M3/R4F は ARM Limited の EU およびその他の国における商標および登録商標です
  - ・RZT1は、ルネサス エレクトロニクス株式会社の登録商標、または商品名です。
  - ・その他の会社名、製品名は、各社の登録商標または商標です。
- 8. 参考文献
  - ・「RZ/T1 グループ ユーザーズマニュアル ハードウェア編」 ルネサス エレクトロニクス株式会社
  - ・ルネサス エレクトロニクス株式会社提供のサンプル集
  - ・「J-Link User Guide」 SEGGER 社
  - ・「J-Flash User Guide」 SEGGER 社
  - ・「Flasher User Guide」 SEGGER社
  - ・その他

 $\mp 486-0852$ 

愛知県春日井市下市場町6-9-20 エーワン株式会社 http://www.robin-w.com