🂋 エーワン株式会社

Rev 1.40.00

# DS-5 (ARMC) ツールチェインの設定と必要事項の説明

(ルネサス RZ/A1H用)

DS-5(ARMC)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。

- 1. ツールチェイン設定画面を開きます。
  - 1) 【プロジェクト】 【プロパティ】を選択します。



### 2)「C/C++ビルド」-「設定」を選択します。

TD/(F1: USER_Debug		
フィルタ入力	עע-ג 🔅 ד בעע	•
リソース     CIC++ ビルド     Change Toolchain Ver Tool chain エディター ビルド変数 ロギング 単立 設定 CIC++一般 ビルター プロシェクト参照 案行/デバッグ設定	/(ス(P): /USER_Debug タイプ(Y): プロジェクト ロケーション(L): C:¥05-5 Workspace¥USER_Debug 翻検更目時(M): 2015年1月6日 13:18:23 デキスト・ファイル・エンコード(T) ※ コンデナーから掲来(I)(M5921) ● その燃(O): MERE ■ 算生リソースのエンコードを開始保留(S) 新橋デキスト・ファイルの対応切り文字(F) ※ コンデナー (Windows) から最単度) ● その燃(H): Windows =	
×	ec	+
3	0K #1/2/2/4	1

- 2. ARMC コンパイラの設定
  - 1) ARMC コンパイラのすべてのオプションを表示



### 2) ターゲット設定画面





ターゲット CPU(cpu)	Cortex-A9
バイト順序	リトルエンディアン(littleend)を選択
命令セット	ARM(arm)を選択
インターワーク	Thmb 命令のコードを使用する可能性があ
	る場合は、☑して下さい。
アンアライドアクセスをディセーブル	
ターゲット FPU(-fpu)	vfpv3_fp16

その他は「デフォルト」設定

# 3) プリプロセッサ設定画面

プロパティ: NET_NORTI_US	В		3
フィルタ入力	設定	φ•φ••	-
> リソース C/C++ ビルド Change Toolchain Ver Tool chain エディター ビルド変数 ロギング	模式: Debug [アクティブ]	▼ 構成の管理	•
環境	ARM Cコンパイラ	前処理のみ (-E)	Ш
設定	ターゲット	マクロの定義 (-D) 🗿 🌒 🗑 🤴 😓	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	<li>グリプロセッサ         <ul> <li>グリプロセッサ</li> <li>インクルード</li> <li>ソース言語</li> <li>最速化</li> <li>デバッグ</li> <li>警告およびエラー</li> <li>その他</li> </ul> </li> <li>ARM アセンブラ         <ul> <li>ターゲット</li> <li>ブリプロセッサ</li> <li>デバッグ</li> <li>警告およびエラー</li> <li>その他</li> </ul> </li>	ARMC ITF_LIB CH2 RTOS	
* m +	▲ 勧 ARM リンカ ◎ ターゲット ◎ イメージレイアウト	マクロの定義を解除 (-U) 創 船 街 日 日	-
0		ОК <b>+</b> P>セル	

C ソースに#ifdef 等のマクロ定義している場合に使用します。		
<mark>注*1</mark>		
USED_DEFnano=x	x = DEFnanoを使用[1]する・[0]しない。	
RTOS	NORTi 使用時に定義	
CH2	NORTi 使用時に定義	
ITF_LIB	USB-Function 使用時に定義	
ARMC	USB-Function 使用時に定義	

サンプルプロジェクト別に必要なマクロ定義例					
USER_Debug	USED_DEFnano=1				
EVrxRZ_Sample	USED_DEFnano=1				
EVrxRZ_Sample_USB	USED_DEFnano=1			ITF_LIB	ARMC
EVrxRZ_Norti	USED_DEFnano=1	RTOS	CH2		
EVrxRZ_Norti_USB	USED_DEFnano=1	RTOS	CH2	ITF_LIB	ARMC
EVRZ_Sample	USED_DEFnano=1				
EVRZ_Sample_USB	USED_DEFnano=1			ITF_LIB	ARMC
EVRZ_Norti	USED_DEFnano=1	RTOS	CH2		
EVRZ_Norti_USB	USED_DEFnano=1	RTOS	CH2	ITF_LIB	ARMC

# <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。

# 4) インクルード設定画面

フィルタ入力	設定	\$ • \$ • •
<ul> <li>&gt; リソース</li> <li>C/C++ ビルド Change Toolchain Ver Tool chain エディター ビルド変数 ロギング 環境 設定</li> <li>&gt; C/C++ 一般 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>ツール投定</li> <li>ビルド・スティ</li> <li>シ ARM C コンパイラ</li> <li>シ ターゲット</li> <li>ジ ブリプロセッサ</li> <li>ジ インクルード</li> <li>ジ ソース言語</li> <li>※ 母連化</li> <li>※ デバッグ</li> <li>※ 蓄告およびエラー</li> <li>※ その他</li> <li>※ ARM アセンブラ</li> <li>※ ARM アセンブラ</li> <li>※ ターゲット</li> <li>※ ブリプロセッサ</li> <li>※ デバッグ</li> <li>※ 蓄告およびエラー</li> <li>※ その他</li> <li>※ ARM リンカ</li> <li>※ ターゲット</li> <li>※ イメージレイアウト</li> <li>※ ライブラリ</li> </ul>	yブ ・ ビルド成果物 () バイナリー・パーサー () エラー・パーサー インクルードパス(-1) () () () () () () () () () () () () ()
<ul> <li>         登録化         <ul> <li>                        通加情報</li></ul></li></ul>	インクルード済み (preinclude) 🔕 🗟 🖗 🎘	

サンプル	プロジェクト別に必要なインクルードパスの設定例			
USER_Debug	"\${workspace_loc'/\${ProjName}/src_app/inc}"			
	"\${workspace_loc'/\${ProjName}/src_sys/inc}"			
	"\${workspace_loc:/\${ProjName}/src_sys/inc/iodefines}"			
EVrxRZ_Sample	追加+ "\${workspace_loc;/\${ProjName}/src_eva/inc}"			
EVrxRZ_Sample_USB	追加+ "\${workspace_loc;/\${ProjName}/src_eva/inc}"			
	追加+ "\${workspace_loc;/\${ProjName}/ITF_LIB/Include}"			
	追加+ "\${workspace_loc;/\${ProjName}/ITF_LIB/ITF_Include}"			
EVrxRZ_Norti	追加+ "\${workspace_loc;/\${ProjName}/src_eva/inc}"			
	追加+ "\${workspace_loc;/\${ProjName}/NORTi/INC}"			
	追加+ "\${workspace_loc;/\${ProjName}/NORTi_smp/NETSMP/inc}"			
	追加+ "\${workspace_loc;/\${ProjName}/NORTi_smp/SMP/inc}"			
EVrxRZ_Norti_USB	追加+ "\${workspace_loc;/\${ProjName}/ITF_LIB/Include}"			
	追加+ "\${workspace_loc;/\${ProjName}/ITF_LIB/ITF_Include}"			
EVRZ_Sample	追加+ "\${workspace_loc;/\${ProjName}/src_eva/inc}"			
	追加+ "\${workspace_loc;/\${ProjName}/src_evb/inc}"			
	追加+ "\${workspace_loc;/\${ProjName}/src_vdc/inc}"			
EVRZ_Sample_USB	追加+ "\${workspace_loc;/\${ProjName}/src_eva/inc}"			
	追加+ "\${workspace_loc;/\${ProjName}/src_evb/inc}"			
	追加+ "\${workspace_loc;/\${ProjName}/src_vdc/inc}"			
	追加+ "\${workspace_loc;/\${ProjName}/ITF_LIB/Include}"			
	追加+ "\${workspace_loc;/\${ProjName}/ITF_LIB/ITF_Include}"			
EVRZ_Norti	追加+ "\${workspace_loc:/\${ProjName}/src_eva/inc}"			
	追加+ "\${workspace_loc;/\${ProjName}/src_evb/inc}"			
	追加+ "\${workspace_loc;/\${ProjName}/src_vdc/inc}"			



	追加+ "\${workspace_loc;/\${ProjName}/NORTi/INC}"
	追加+ "\${workspace_loc;/\${ProjName}/NORTi_smp/NETSMP/inc}"
	追加+ "\${workspace_loc;/\${ProjName}/NORTi_smp/SMP/inc}"
EVRZ_Norti_USB	追加+ "\${workspace_loc:/\${ProjName}/src_eva/inc}"
	追加+ "\${workspace_loc;/\${ProjName}/src_evb/inc}"
	追加+ "\${workspace_loc:/\${ProjName}/src_vdc/inc}"
	追加+ "\${workspace_loc;/\${ProjName}/NORTi/INC}"
	追加+ "\${workspace_loc;/\${ProjName}/NORTi_smp/NETSMP/inc}"
	追加+ "\${workspace_loc;/\${ProjName}/NORTi_smp/SMP/inc}"
	追加+ "\${workspace_loc;/\${ProjName}/ITF_LIB/Include}"
	追加+ "\${workspace_loc'/\${ProjName}/ITF_LIB/ITF_Include}"

5) ソース言語設定画面 (デフォルト設定)

● プロパティ: USER_Debug					-	x
フィルタ入力	設定				<b>⇔</b> •⇔•	-
<ul> <li>&gt; リソース</li> <li>&gt; C/C++ ビルド Change Toolchain Ver Tool chain エディター ビルド交数 ロギング 環境 設定</li> <li>&gt; C/C++ 一般 ビルダー プロジェクト参照 実行/デパッグ設定</li> </ul>	<ul> <li>③ ARM C コンパイラ</li> <li>③ プリプロセッサ</li> <li>④ プリプロセッサ</li> <li>④ プリプロセッサ</li> <li>④ インクルード</li> <li>④ Palet</li> <li>④ コード生成</li> <li>② デバック</li> <li>● 雪市およびエラー</li> <li>③ その物</li> <li>● ARM アセンブラ</li> <li>④ コード生成</li> <li>⑤ プリプロセッサ</li> <li>② コード生成</li> <li>③ デバック</li> <li>● 雪市およびエラー</li> <li>※ その物</li> <li>● ARM リンカ</li> <li>※ 全般</li> <li>※ イメージレイアウト</li> <li>※ ライブラリ</li> <li>※ 急速に</li> <li>※ 追加情報</li> <li>※ 警告およびエラー</li> <li>※ その物</li> </ul>	ソース書語モード 図NU 拡張機能を 言語への敏密な準規 ■ C++ 例外をイネ	デフォルト 有効化 (gnu) デフォルト ーブル (exceptions)		•	* E
?	ide de			ОК	キャンセル	1
						J

6) 最適化設定画面



#### 7) デバッグ設定画面

●プロパティ: USER_Sample_U	JSB	
フィルタ入力	設定	$\phi \bullet \phi \bullet \bullet \bullet$
<ul> <li>&gt; リソース</li> <li>C/C++ ビルド         <ul> <li>Change Toolchain Ver</li> <li>Tool chain エディター</li> <li>ビルド変数</li> <li>ロギング</li> <li>環境</li> <li>設定</li> </ul> </li> <li>C/C++ 一般</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>	<ul> <li>※ ARM C コンパイラ</li> <li>ダーゲット</li> <li>ブリプロセッサ</li> <li>インクルード</li> <li>ソース言語</li> <li>最適化</li> <li>デバッグ</li> <li>警告およびエラー</li> <li>ターゲット</li> <li>ブリプロセッサ</li> <li>ブリプロセッサ</li> <li>デバッグ</li> <li>夢告およびエラー</li> <li>愛 ターゲット</li> <li>ブリプロセッサ</li> <li>デバッグ</li> <li>警告およびエラー</li> <li>その也</li> <li>※ ARM アセンブラ</li> <li>ダーケット</li> </ul>	PZ)
< m +		
0	OK.	キャンセル

### 8) 警告およびエラー設定画面



#### 9) その他設定画面



#### 3. ARMアセンブラの設定

1) ARM アセンブラのすべてのオプションを表示



# 2) ターゲット設定画面

● プロパティ: USER_Norti_US	в			-	×
フィルタ入力	設定			\$ • \$ •	-
<ul> <li>リソース</li> <li>C/C++ ビルド         <ul> <li>Change Toolchain Ver</li> <li>Tool chain エディター</li> <li>ビルド変数</li> <li>ロギング</li> <li>環境</li> <li>設定</li> </ul> </li> <li>C/C++ 一般</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>	<ul> <li> ARM C コンパイラ ターゲット ブリブロセッサ インクルード ソース言語 ※ 根温化 ※ デバッグ ※ 暫吉およびエラー ※ その他 </li> <li> ※ ARM アセンブラ ※ クーゲット ※ ブリブロセッサ ※ デバッグ ※ 暫吉およびエラー ※ その他 ※ ARM リンカ ※ ターゲット ※ イメージレイアウト ※ ライブラリ ※ 母連化 ※ 追加情報 ※ 暫吉およびエラー </li> </ul>	ターゲット CPU (cpu) バイト順序 命セット ロインターワーク (apcs=/m マアンアラインドアクセスをラ ターゲット FPU (fpu) 浮動小数点モード (fpmode) 浮動小数点 PCS (apcs)	Cortex-A9 リトルエンディアン (littleend) ARM (arm) nterwork) デイセーブル (no_unaligned_access) vfpv3_fp16 デフォルト デフォルト		·
3			ОК ‡т	ンセル	

ARMCコンパイラのターゲット設定と同じにする。

ターゲット CPU(cpu)	Cortex-A9
バイト順序	リトルエンディアン(littleend)を選択
命令セット	ARM(arm)を選択
インターワーク	Thmb 命令のコードを使用する可能性があ
	る場合は、☑して下さい。
アンアライドアクセスをディセーブル	
ターゲット FPU(-fpu)	vfpv3_fp16

その他は「デフォルト」設定

3) プリプロセッサ設定画面 (デフォルト設定)

ンブルする前に入力を削処理する (cpreproc) ・ Iセッサオプション (cpreproc_opts)

# 4) デバッグ設定画面



5) 警告およびエラー設定画面



# 6) その他設定画面

● プロパティ: USER_Sample_I	USB	- • • •X
フィルタ入力	股定	0-0- <b>-</b>
<ul> <li>&gt; リソース</li> <li>C/C++ ビルド Change Toolchain Ver Tool chain エディター ビルド変数 ロギング 環境 設定</li> <li>&gt; C/C++ 一般 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>・ ● ARM C コンパイラ</li> <li>※ ターゲット</li> <li>※ ブリプロセッサ</li> <li>※ ブリプロセッサ</li> <li>※ インクルード</li> <li>※ ソース言語</li> <li>※ 最適化</li> <li>※ 予パッグ</li> <li>※ 香島およびエラー</li> <li>※ その地</li> <li>・ ● ARM アセンブラ</li> <li>※ ARM アセンブラ</li> <li>※ デバッグ</li> <li>※ 予パッグ</li> <li>※ 予パッグ</li> <li>※ 音告およびエラー</li> <li>※ テパッグ</li> <li>※ 香島およびエラー</li> <li>※ ARM リンカ</li> </ul>	
• m •		,
?	ОК \$	r>tu

サンプルプロジェクトの追加オプション				
USER_Debug	arm-only	ARM_ONLY コード		
EVrxRZ_Sample	-i./src_sys/inc/	インクルードパスの設定		
EVrxRZ_Sample_USB				
EVRZ_Sample	<mark>注*1</mark>			
EVRZ_Sample_USB	pd="USED_DEFnanoEQU x"	x= DEFnano を		
		使用[1]する・[0]しない。		
EVrxRZ_Norti	arm-only	ARM_ONLY コード		
EVrxRZ_Norti_USB	pd=''MMU SETL {TRUE}''	MMU と各キャッシュを有		
EVRZ_Norti	pd="CACHE_ISETL {TRUE}"	効・無効の指定用フラグ		
EVRZ_Norti_USB	pd="CACHE_DSETL{TRUE}"			
	pd="CACHE_L2SETL{TRUE}"			
	pd='CACHE_WT SETL {TRUE}"			
	<u> </u>			
	pd='USED_DEFnanoEQU x''	x= DEFnano を		
		使用[1]する・[0]しない。		

# <mark>注\*1</mark>

「\_\_USED\_DEFnano\_\_ EQU 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。



- 4. ARM リンカの設定
  - 1) ARM リンカのすべてのオプションを表示

フィルタ入力	設定			φ.•.φ.•	-
> リソース ■ C/C++ ビルド Change Toolchain Ver Tool chain エディター ビルド変数	構成: Debug [アクティブ]			▼ 構成の管理…	] [
ロギング	ツール設定     ドルド・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	
設定	ARM C コンパイラ	コマンド:	armlink		E
<ul> <li>&gt; C/C++ 一般</li> <li>ビルダー</li> <li>プロジェクト参照</li> </ul>		すべてのオプション:	cpu=Cortex-A9fpu=vfpv3_fp16entry=Start userlibpath="N:¥UsrAp¥C_H25¥DS-5 Workspace ¥USER_Sample_USB¥ITF_LIB"no_remove		
リファクタリング爆空 実行/デバッグ設定	<ul> <li>※ ライブラリ</li> <li>※ 母連化</li> <li>※ 追加情報</li> </ul>	エキスパート設定: コマンド行			
	※ 警告およびエラー   ※ その他	パターン:	S(COMMAND) S(FLAGS)	2(001901_FD4G) 2(001	
e [ +					-
?		C	ОК	キャンセル	

2) ターゲット設定画面

フィルタ入力	設定				9-9-
<ul> <li>リソース</li> <li>C/C++ ビルド</li> <li>Change Toolchain Ver Tool chain エディター ビルドの歌</li> </ul>	構成: Debug [	アクティブ]			▼ 構成の管理
ロギング 環境 設定	<ul> <li>ジール設定</li> <li>シ (※) ARM C</li> </ul>	ビルド・スティ コンパイラ	ップ 🤗 ビルド成果物	<ul> <li>□ /(イナリー・パーサー)</li> <li>Cortex-A9</li> </ul>	3 15- · パーサー
設定 ▷ C/C++ 一般 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	● (ARM ) ● (1) ARM ) ◎ ター ◎ フィ ◎ ライ ◎ 通動 ◎ 通動 ◎ 登録	センフラ リンカ - ゲット ページレイアウト (ブラリ 動化 ロ情報 話よびエラー )地	ターゲット FPU (fpu)	vfpv3_fp16	
( m. ) )					

ARMC コンパイラと ARM アセンブラのターゲット設定と同じにする。

ターゲット CPU(cpu)	Cortex-A9
ターゲット FPU(-fpu)	vfpv3_fp16

3) イメージレイアウト設定画面



# 4) ライブラリ設定画面



サンプルプロジェクト別のライブラリ検索パス			
USER_Debug	なし		
EVrxRZ_Sample			
EVRZ_Sample			
EVrxRZ_Sample_USB	"\${workspace_loc;/\${ProjName}/ITF_LIB}"		
EVRZ_Sample_USB			
EVrxRZ_Norti	"\${workspace_loc;/\${ProjName}/NORTi_LIB/CA9/DS5}"		
EVRZ_Nort			
EVrxRZ_Norti_USB	"\${workspace_loc;/\${ProjName}/ITF_LIB}"		
EVRZ_Norti_USB	"\${workspace_loc;/\${ProjName}/NORTi_LIB/CA9/DS5}"		

サンプルプロジェクト別のライブラリファイル				
USER_Debug	なし			
EVrxRZ_Sample				
EVRZ_Sample				
EVrxRZ_Sample_USB	ITFUSBLib_RZA1H_A1.a <mark>(別売り)</mark>			
EVRZ_Sample_USB				
EVrxRZ_Norti	n4d7anol.a <mark>(別売り)</mark>			
EVRZ_Norti	n4e7anol.a <mark>(別売り)</mark>			
EVrxRZ_Norti_USB	ITFUSBLib_RZA1H_A1.a <mark>(別売り)</mark>			
EVRZ_Norti_USB	n4d7anol.a <mark>(別売り)</mark>			
	n4e7anol.a <mark>(別売り)</mark>			

5) 最適化設定画面 (デフォルト設定)

● プロパティ: USER_Norti_US	В	- 0 💌
フィルタ入力	設定	φ·φ·•
<ul> <li>リソース</li> <li>C/C++ ビルド</li> <li>Change Toolchain Ver</li> <li>Tool chain エディター</li> <li>ビルド変数</li> </ul>	> リソース a C/C++ ビルド Change Toolchain Ver Tool chain エディター ビルド変数 ロボング 図 ツール投定 ♪ ビルド・ステ・	
ビルド変数 ロギング 環境 設定 > C/C++ 一般 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>※ ARM C コンパイラ</li> <li>※ ARM アセンブラ</li> <li>※ ARM リンカ</li> <li>※ ターケット</li> <li>※ イメージレイアウト</li> <li>※ ライブラリ</li> <li>※ 最速化</li> <li>※ 追加情報</li> <li>※ 警告およびエラー</li> <li>※ その他</li> </ul>	<ul> <li>☑ 未使用セクションを削除する (remove)</li> <li>□ リンカのインライン化を可能にする (Inline)</li> <li>1 特定のセクションを保持する (keep)</li> </ul>
·		・ OK キャンセル

# 6) 追加情報の設定画面

UV-2	Internet - A Activity Andrews Internet Activity
<ul> <li>C/C++ ビルド</li> <li>Change Toolchain Ver</li> <li>Tool chain エディター</li> <li>ビルド変数</li> <li>ロギング</li> <li>環境</li> <li>設定</li> <li>C/C++ 一般</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/デノ(ッグ設定</li> </ul>	コールグラフを生成する (callgraph)         コールグラフファイル (callgraph_file)         コールグラフの形式 (callgraph_output)         HTML         マイメーシマップを生成する (map)         詳細な出力 (verbose)         マ出力イメージのコードサイズとデータサイズの合計を一覧表示する (info=siz         マニカイメージのコードサイズとデータサイズの合計を一覧表示する (info=siz         マニカイメージのコードサイズとデータサイズの合計を一覧表示する (info=siz         一川除された未使用セクションを一覧表示する (info=unused)         RW 圧縮情報を印刷する (info=compression)         グローバルシンボルのスタッグ消費量を一覧表示する (info=stack)         リンカによってインライン展開された関数を一覧表示する (info=inline)         診断出力をファイルに転送 (list)       \${ProjName}.map

リンカで生成される「Project.map」に追加される情報の選択画面です。必要に応じて選択する。

7) 警告およびエラー設定画面 (デフォルト設定)

マイルタ入力	設定		Q • Q •	*
<ul> <li>リソース・</li> <li>C/C++ ビルド</li> <li>Change Toolchain</li> <li>Tool chain エディち</li> <li>ビルド変数</li> <li>ロギング</li> <li>環境</li> <li>設定</li> <li>C/C++ 一般</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> </ul>	<ul> <li></li></ul>	エラーの重大度 (diag_error)		

### 8) その他設定画面



追加オプション		
arm-only	標準ライブラリを ARM 命令のみで作成	
symbols	各ローカルとグローバルシンボル、およびそ	
	のアドレス一覧を作成する。	
load_addr_map_info	実行領域のロードアドレスをマップファイル	
	に追加する。	
datacompressor=off	RWデータ圧縮を無効にする。	
library_type=standardlib	リンク時にフルランタイムライブラリを選択	
	します。	
scatter¥scatter_file¥scatter.scat	各セクションのアドレス指定にスキャッタフ	
	ァイルを使用する。	

5. ビルド・ステップの設定画面

フィルタ入力	設定 ゆ・ウ・・
<ul> <li>&gt;&gt; リソース</li> <li>▲ C/C++ ビルド</li> <li>Change Toolchain Ver Tool chain エディター</li> </ul>	③ ツール設定 ♪ ビルド・ステップ 常 ビルド成果物 品 バイナリー・パーサー ④ エラー・パーサー ビルド前のステップ コマンド:
ビルド変数 ロギング 環境 設定 ▷ C/C++ 一般 ビルダー プロジェクト参照	・ 設知月: ・
	ビルド後のステップ コマンド:
リファクタリング履歴 実行/デバッグ設定	"after_build.bat" \${ProjName} • 認知時:
4 ( m ) )	

ビルド実行前と実行後に追加したいコマンドがある場合に設定します。

サンプルプロジェクトでは「after_build.bat	」にて実行後に設定する。
fromelfbinoutput=%1.bin%1.axf	bin ファイルの作成
fromelf m32combined output=%1.mot %1.axf	motファイルの作成

6. ビルド成果物の設定画面 (デフォルト設定)

フィルタ入力	設定	9 • ¢	*
> リソース ▲ C/C++ ビルド Change ToolChain Tool chain エディタ ビルド変数 ロギング 環境 設定	<ul> <li>         ・ツール設定         成果物タイプ:     </li> </ul>	ドルド・ステップ 学 ビルド成果物 前 バイナリー・パーサー 3 エラー・パーサ     東行可能	
	成果物名:	\${ProjName}	
	成果物処据子: axr 出力接頭部:	•	
> C/C++ −R2 +			

サンプルプロジェクトの設定(DEF	Fnanoを使用する場合は、変更不可)
成果物タイプ	実行可能に選択
成果物名	\${ProjName}
成果物拡張子	axf

7. バイナリー・パーサー設定画面 (デフォルト設定)

アイルタ入力	設定	\$ • \$ •
<ul> <li>&gt;&gt; リソース ・</li> <li>C/C++ ビルド Change Toolchain Tool chain エディち ビルド変数 ロギング 環境 設定</li> <li>&gt;&gt; C/C++ 一般 ビルダー</li> </ul>	<ul> <li>③ ツール設定 ・ ビルド・ステップ ・ ビルド成果物 ・ パイナリー・パーサー         ・ ・ パーサー:         <ul> <li>Mach-0 64 パーサー</li> <li>Elf パーサー</li> <li>Elf パーサー</li> <li>PE Windows パーサー</li> <li>HP-UX SOM パーサー</li> <li>Mach-O Parser (Deprecated)</li> <li>GNU Elf パーサー</li> </ul> </li> </ul>	<b>エラ-・パーサー</b> 上へ存動 下へ存動
リファクタリング履歴・	Cygwin PE /(-サ-	

全てノーチェック(使用していません)

8. エラー・パーサー設定画面

	設定	6.4.4.4
リソース C/C++ ビルド Change Toolchain Ver Tool chain エディター ビルド変数 ロボング	構成: Debug [アクティブ] ・	横成の管理
<ul> <li>ビルト装数</li> <li>ロギング 環境 設定</li> <li>C/C++ 一般</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>	✓	道加 編集 相論 上へ移動 下へ移動

「ZARM コンパイラエラーパーサー」のみを使用する。

9. スキャッタファイルについて

スキャッタファイルとは、セグメントごとにアブソリュートアドレスを定義するためのファイ ルです。スキャッタファイル独自の予約語がありますので ARM 社が提供しているドキュメント 「ARM コンパイラツールチェーン リンカの使用:DUI0474GJ\_using\_the\_arm\_linker.pdf」の 「8:スキャッタファイルの使用」を参照して下さい。

サンプルプロジェクト[scatter.scat]の定義例1

#! armcc -E			
;Content-Type: text/plain; charset=utf-8	3		
;/************************************	*******		
;* File Name : scatter.scat			
* Description : Scatter file			
***************************************	***************************************		
:: Definition of On-chip large-capacity	RAM (IRAM) area		
#define LRAM START	(0x20000000)		
#define LRAM STZE	(0x20000000) (0x00000000)		
#define LRAM END	(IRAM STARTAL RAM STZE)		
#define APP_START	(0x20080000)		
#define VECT_SIZE	(0x100)		
#define APP SIZE	(0x300000)		
-			
#define TTB_START	((APP_START+APP_SIZE) AND 0xFFFFC000)		
#define TTB_SIZE	(0x4000)		
#define ARM_LIB_STACK_SIZE	(0x8000)		
#define STACK_SIZE	(0x80000)		
#define ARM_LIB_HEAP_SIZE	(0x8000)		
LOAD_ROM APP_START APP_SIZE	;; CODE/CONST/DATA Area		
{			
VECTOR TABLE +0	FIXED ;; CODE/CONST/DATA Area		
{ * (VECTOR TABLE, +FIRST) }	;; Vector table		
RESET INIT HANDLER +VECT SIZE	FIXED		
{ * (RESET INIT HANDLER) }			
( (			
InRoot +0	FIXED		
{ * (InRoot\$\$Sections) }	:: All (library) code		
CODE +0	FTYED		
{ * (+R0_CODE) }	TALD		
CONST +0	FTXED		
{ * (+RO-DATA) }			
	FTXFD		

```
;; use as RAM Area
                       ;;
  TTB TTB_START EMPTY TTB_SIZE ;; Level-1 Table for MMU(AND 0xFFFFC000)
  { }
  ARM_LIB_STACK +0 EMPTY ARM_LIB_STACK_SIZE ;; Application stack
  { }
  STACK_TOP
             +0 EMPTY STACK_SIZE
  { }
  ARM_LIB_HEAP +0 EMPTY ARM_LIB_HEAP_SIZE ;; Application heap
  { }
  BSS
              +0
  { * (+ZI)
              }
  LAN BSS
               0x60700000
                                     ;; No Cache Area
  { * (LAN_BSS)
               }
  VIDEO_BSS
               0x60800000
  { * (VIDEO_BSS) }
}
```

```
サンプルプロジェクト[scatter.scat]の定義例の一部
```

ARM_LIB_STACK STACK BSS	START	EMPTY	0x8000	;; Application stack
{ }				
IRQ_STACK	+0	EMPTY	0x8000	;; IRQ mode stack
{ }				
FIQ_STACK	+0	EMPTY	0x2000	;; FRQ mode stack
{ }				
SVC_STACK	+0	EMPTY	0x2000	;; SVC mode stack
{ }				
ABT_STACK	+0	EMPTY	0x2000	;; ABT mode stack
{ }				
ARM_LIB_HEAP	+0	EMPTY	0x8000	;; Application heap
{ }			$\sim$	
				EMIPTY Oxnnnn は、SIZEを指
				正していることになります。

スキャッタファイルを編集する場合の注意点

- 1) #define 行にコメント文をいれないで下さい。
- 2) 漢字入力した場合、このファイルに限り「UTF-8」コードになります。
- 3) MP-RZA1H 基板を使用する限り、変更する箇所は#define の[VECT\_SIZE と TTB\_SIZE]以外のxxxx\_SIZE のみにして下さい。

10. ベクターテーブルとローダーの関係について

MP-RZA1H 基板は、シリアルフラッシュ ROM にローダーとアプリケーションプログラムを 記憶させ、電源 ON 時にローダーが RZA1H の内蔵 RAM にアプリケーションプログラムをロー ドして実行させる仕組みになっています。ローダーは内蔵 RAM にロードする時にロード先の先 頭アドレスと最終アドレスと実行開始アドレスを知る必要があります。この情報を得るため独自 の定義が必要なため下記に説明します。

<sup>1)</sup> ローダーが必要な情報はベクターテーブルに登録する。「\_vector\_table\_s.s/\_vevtor\_table\_r.s」

Entry point for the Reset ha	andler
ector_table LDR pc, =Reset_handler LDR pc, =Undefined_handler LDR pc, =Svc_handler LDR pc, =Prefetch_handler LDR pc, =Abort_handler LDR pc, =Reserved_handler LDR pc, =Irg handler	; Start+0x0000 : リセット ; Start+0x0004 : 未定義命令 ; Start+0x0008 : ソフトウェア割り込み ; Start+0x000C : プリフェッチアボート ; Start+0x0010 : データアボート ; Start+0x0014 : Reserved ; Start+0x0018 : IRQ
LDR pc, =Fiq_handler	; Start+0x001C : FIQ(NMI)
D        Image\$\$VECTOR_TABLE\$\$Base         D        Image\$\$DATA\$\$Limit          D       vector_table         D       0	<ul> <li>Start+0x0020:①内蔵RAM転送先の開始アドレス</li> <li>Start+0x0024:②内蔵RAM転送先の終了アドレス(+1)</li> <li>Start+0x0028:③初期PC値</li> <li>Start+0x002C:④デバッグモードフラグ</li> </ul>
	; DEFnanoを未使用にして、 USBOを開放する場合は、
nfo_end 【重要】 ①②③④の情報は、ROM 化するためには必要な情報 テーブルです。必ず、定義 して下さい。	<ul> <li>(ACDEFODEFOZZERTY 5)。</li> <li>【注意事項】</li> <li>④で「DEFnano 未使用」</li> <li>コード「OxDEFODEFO」をセットし、シリアルフラッシュ ROM に登録した場合、二度とDEFnanoを使用することが出来なくなります。</li> <li>復帰したい場合は、JTAG デバッガ等でシリアルフラッシュ ROM アドレス「Ox2_002C」を未使用コード「OxDEFODEFO」以外の数値を直接書</li> </ul>

以上です。

- 11. 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。

12. 商標

- ・ARM DS-5は、ARM 社の登録商標、または商品名称です。
- ・RZ および RZ/A1H 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。

13. 参考文献

- ・「RZ/A1H 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
- ・「armcc ユーザガイド DUI 0472JJ」 ARM 社
- ・「アセンブラの使用 DUI 0473GJ」 ARM 社
- ・「リンカの使用 DUI 0474GJ」 ARM 社
- ・「コンパイラリファレンスガイド DUI 0328BJ」 ARM社
- ・「アセンブラリファレンス DUI 0489GJ」 ARM社
- ・「armkink リファレンスガイド DUI 0804AJ」 ARM 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