

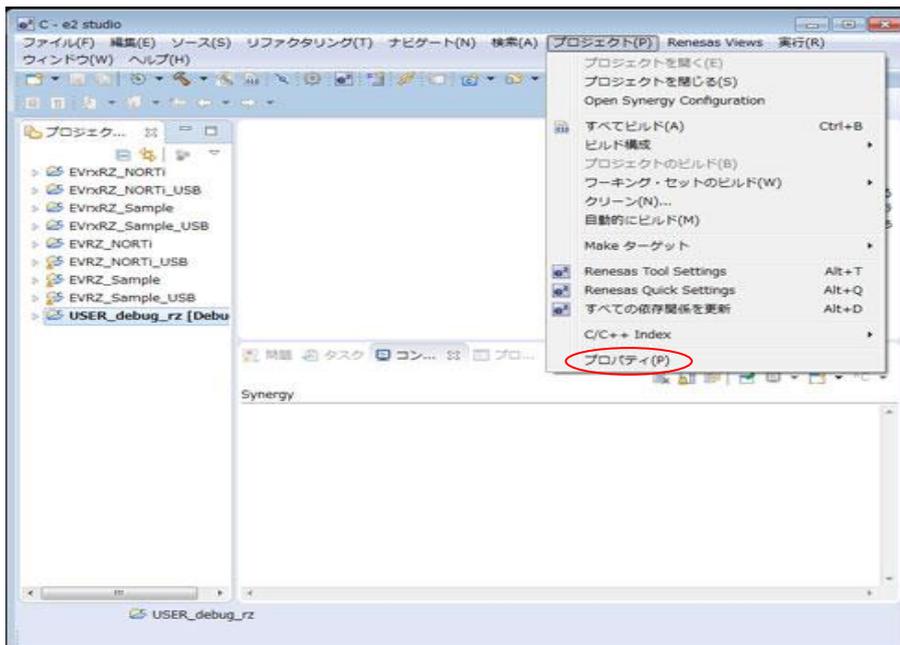
e2studio (GNU Tools) ツールチェーンの設定と必要事項の説明

(ルネサス RZ/A1H用)

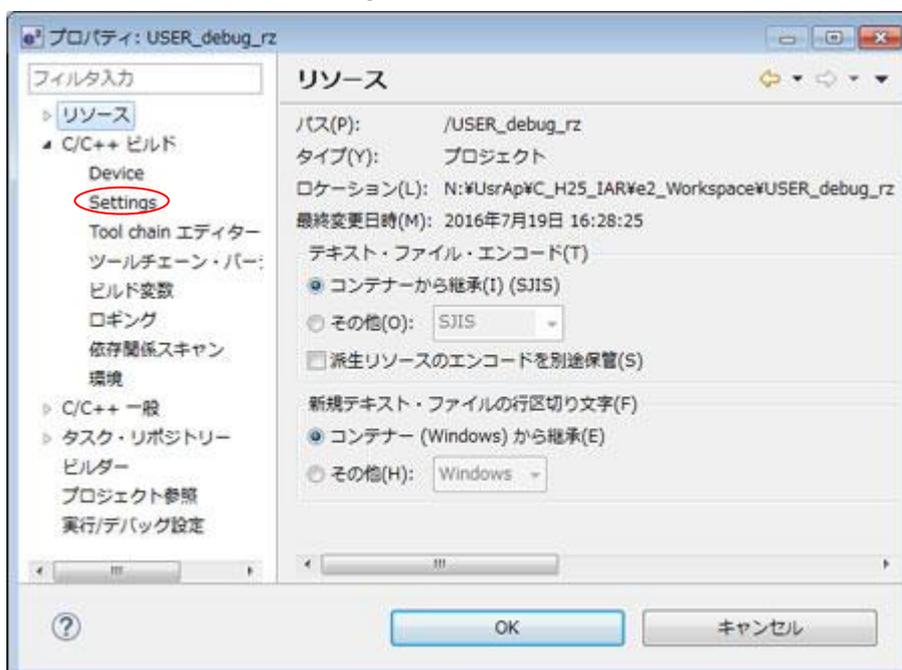
e2studio(GNU Tools)ツールチェーンの設定方法とサンプルプロジェクトに必要な設定を説明します。

1. ツールチェー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。

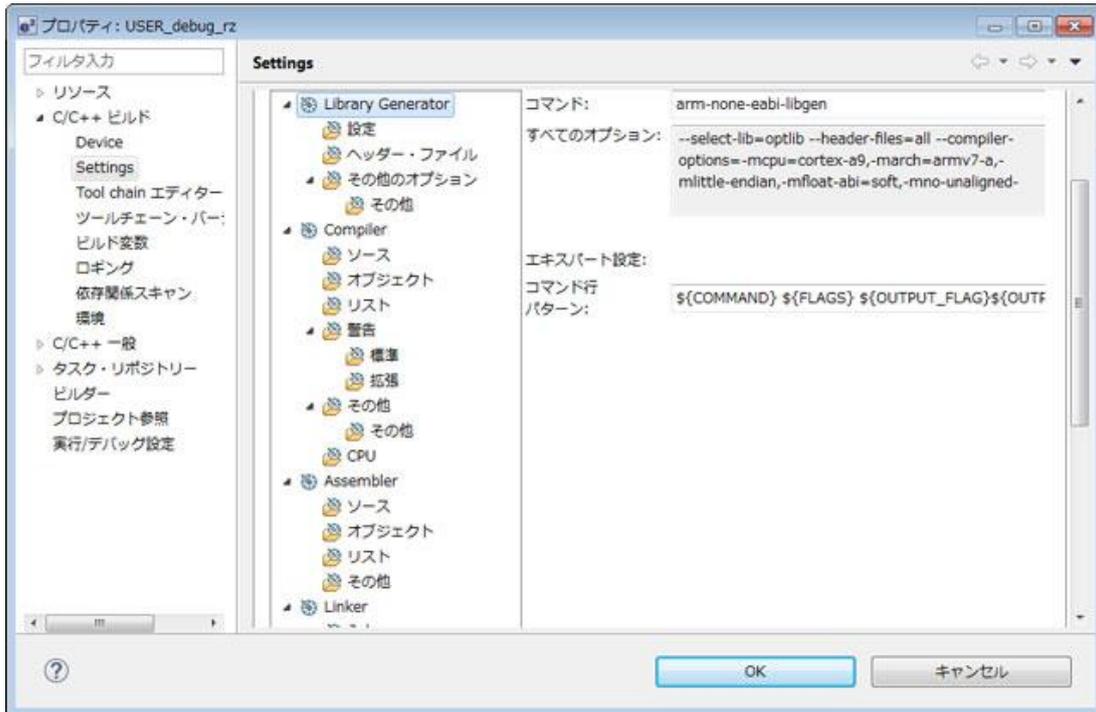


2) 「C/C++ビルド」 - 「settings」 を選択します。

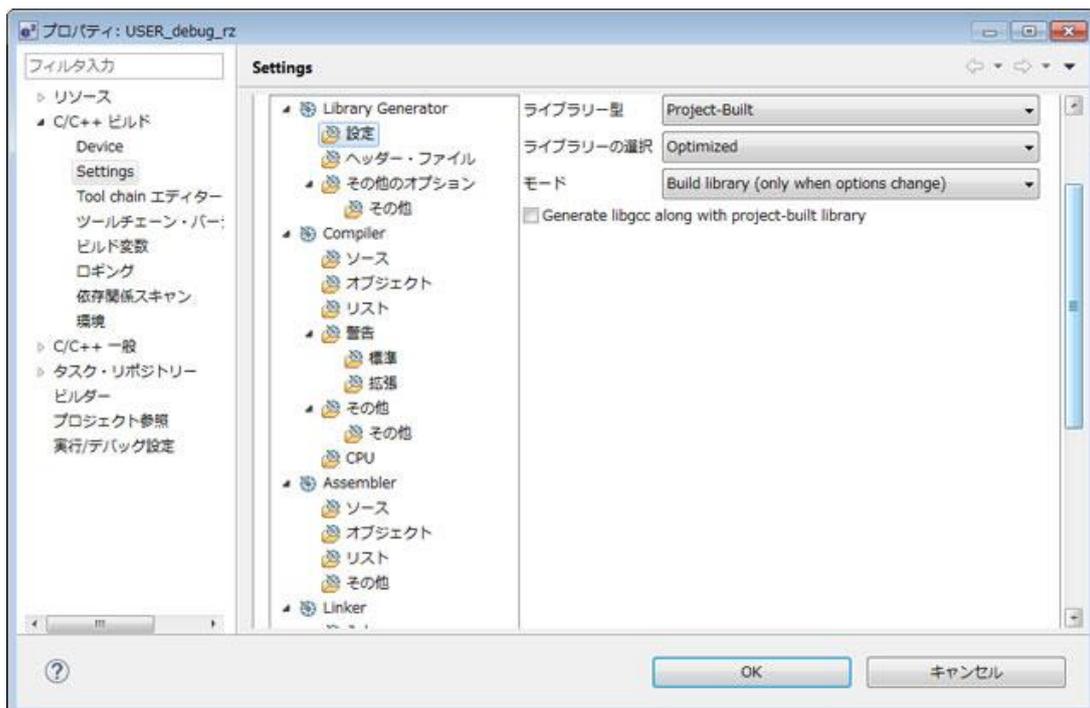


2. ライブラリジェネレータの設定

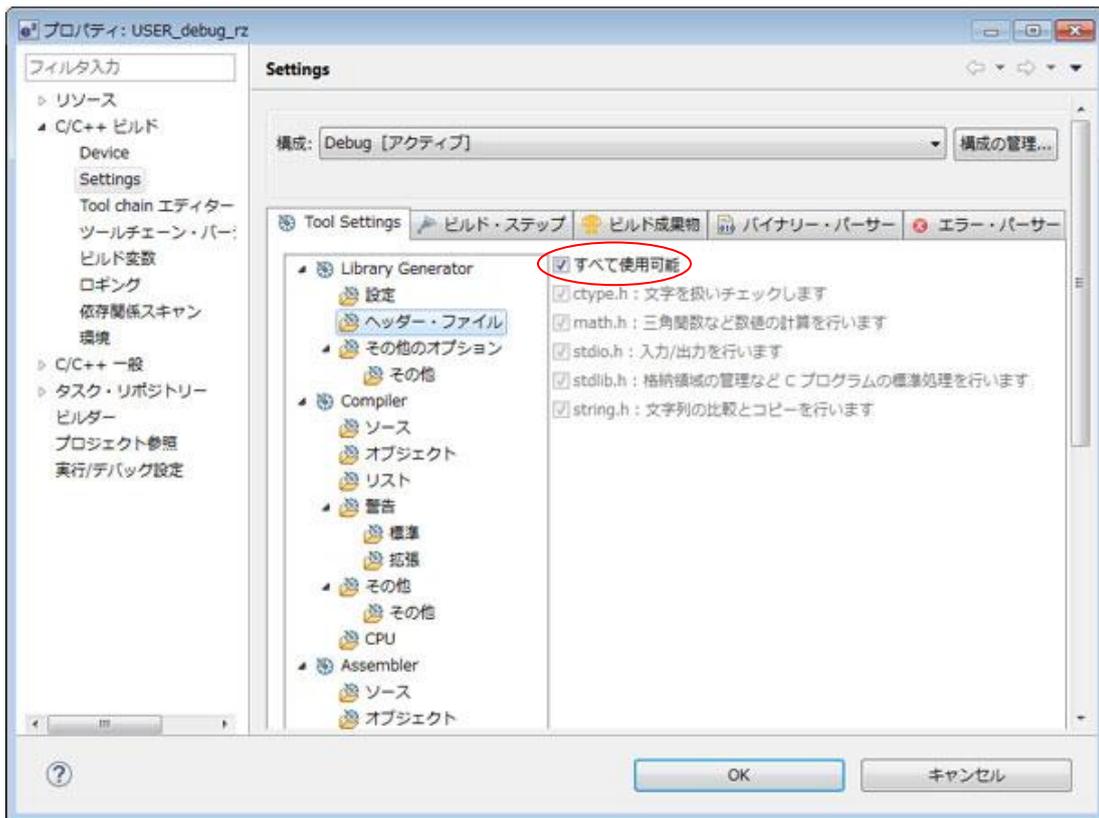
1) ライブラリジェネレータのすべてのオプションを表示



2) 「設定」設定画面 (デフォルト設定)



3) 「ヘッダー・ファイル」 設定画面



4) 「その他のオプション」 設定画面 (デフォルト)



5) 「その他のオプションーその他」 設定画面



<input checked="" type="checkbox"/> char 型を符号なしとして扱う(funsigned-char)
--

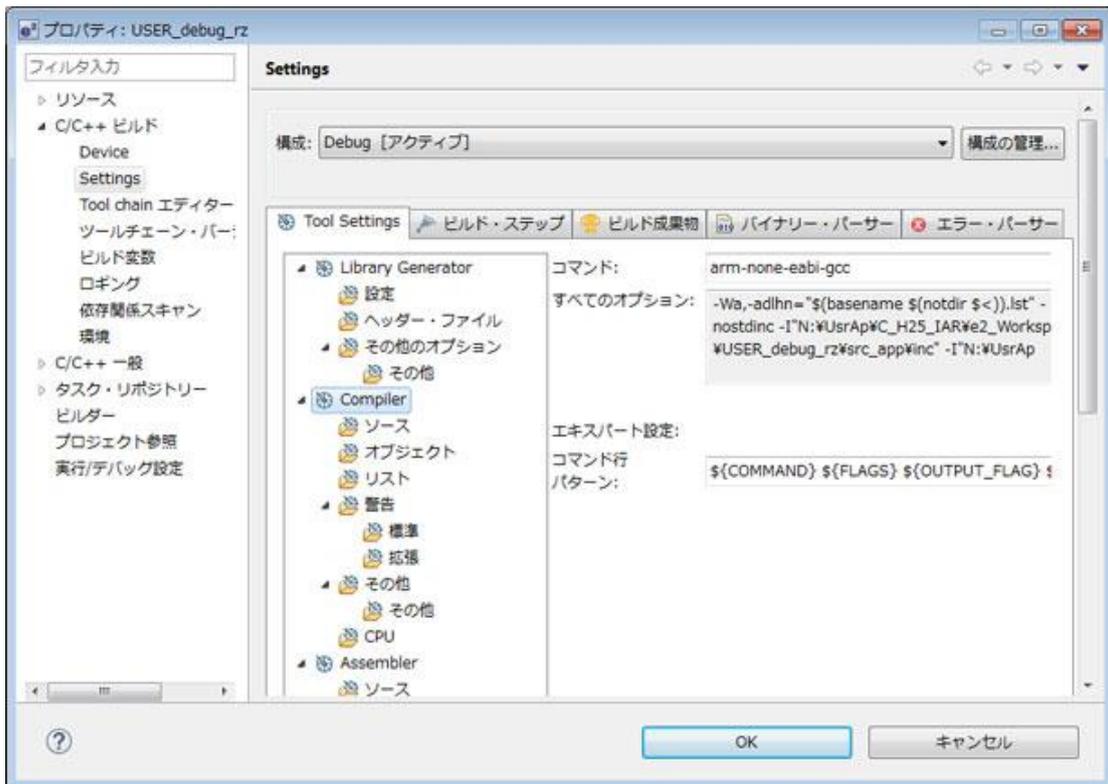
<input checked="" type="checkbox"/> Parse the whole compilation unit before starting to produce code(-funit-at-a-time)
--

<input checked="" type="checkbox"/> Align branch targets to a power-of-two boundary(-falign-jumps)
--

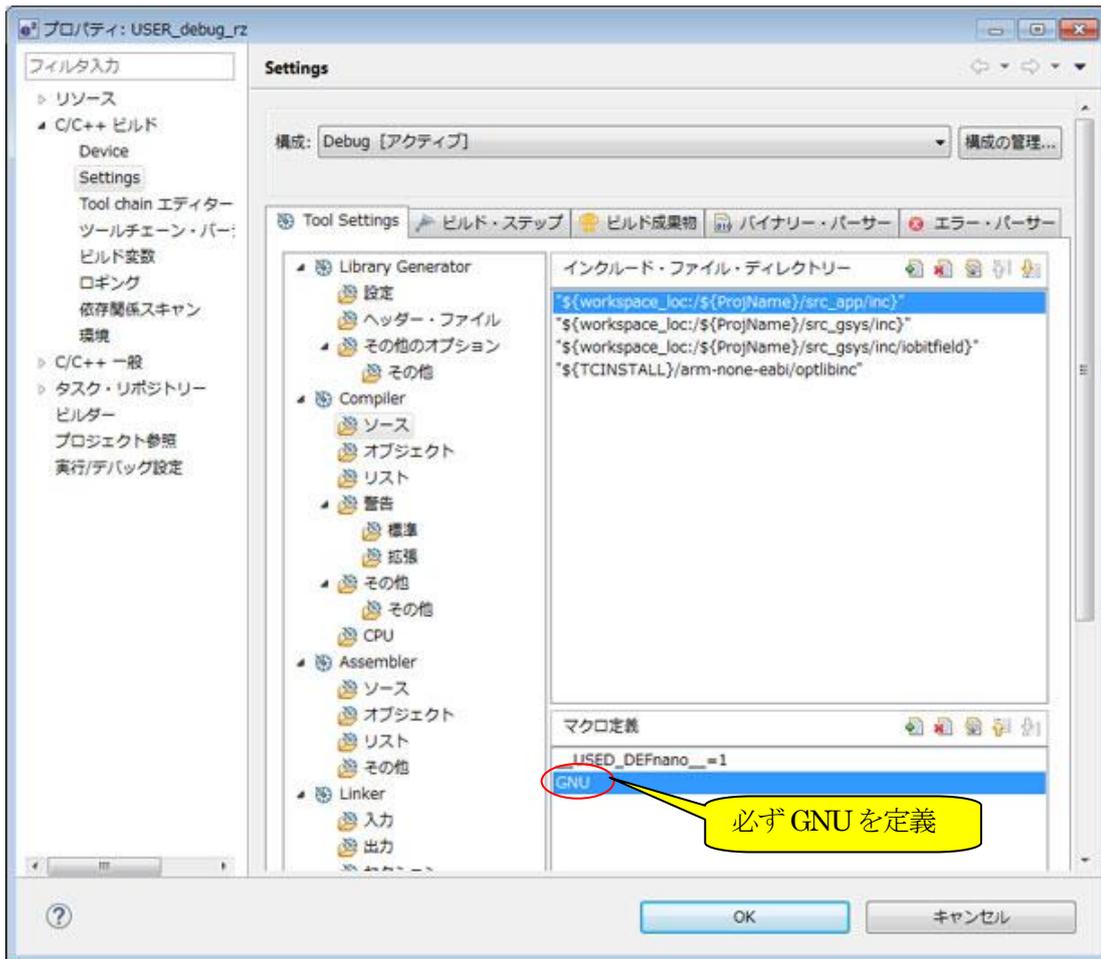
その他は、アプリ側の必要に応じて設定する。

3. コンパイラの設定

1) Compiler のすべてのオプションを表示



2) 「ソース」 設定画面



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_rz	"\${workspace_loc}/\${ProjName}/src_app/inc" "\${workspace_loc}/\${ProjName}/src_gsys/inc" "\${workspace_loc}/\${ProjName}/src_gsys/inc/iodefines" "\${TC_INSTALL}/arm-none-eabi/optlibinc"
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}/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" 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"

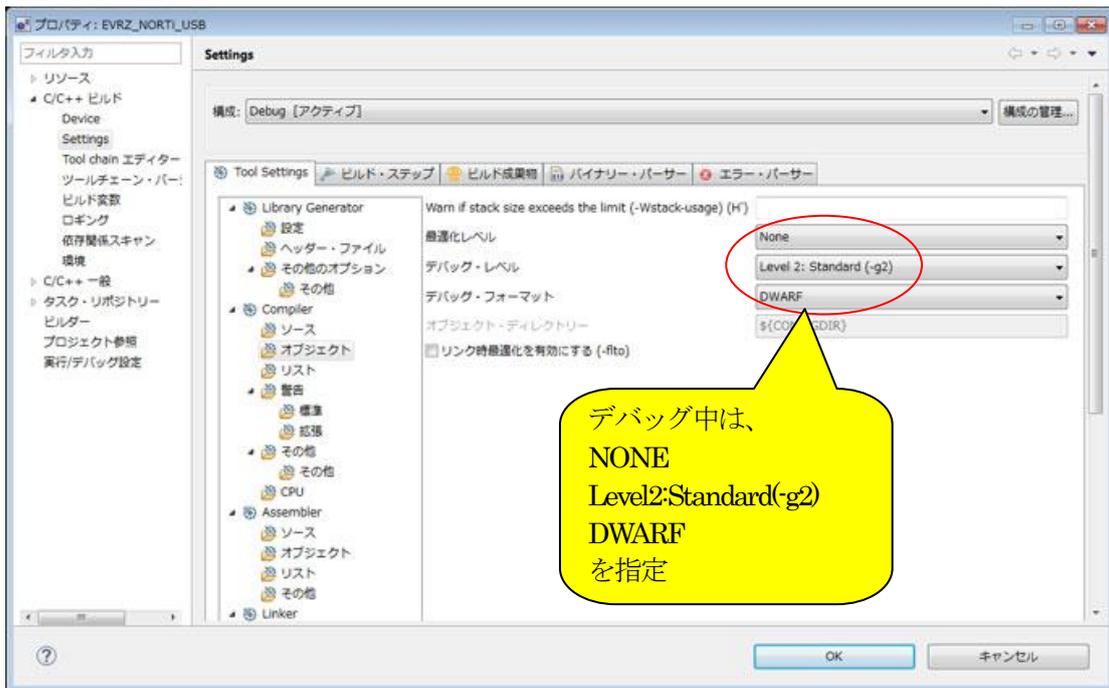
C ソースに #ifdef 等のマクロ定義している場合に使用します。	
注* 1	
<u>USED_DEFnano</u> =x	x = DEFnano を使用[1]する・[0]しない。
GNU	GNU 版の場合に定義
RTOS	NORTi 使用時に定義
CH2	NORTi 使用時に定義
ITF_LIB	USB-Function 使用時に定義

サンプルプロジェクト別に必要なマクロ定義例						
USER_Debug	<u>USED_DEFnano</u> =1	GNU				
EVrxRZ_Sample	<u>USED_DEFnano</u> =1	GNU				
EVrxRZ_Sample_USB	<u>USED_DEFnano</u> =1	GNU				ITF_LIB
EVrxRZ_NORTi	<u>USED_DEFnano</u> =1	GNU	RTOS	CH2		
EVrxRZ_NORTi_USB	<u>USED_DEFnano</u> =1	GNU	RTOS	CH2		ITF_LIB
EVRZ_Sample	<u>USED_DEFnano</u> =1	GNU				
EVRZ_Sample_USB	<u>USED_DEFnano</u> =1	GNU				ITF_LIB
EVRZ_NORTi	<u>USED_DEFnano</u> =1	GNU	RTOS	CH2		
EVRZ_NORTie_USB	<u>USED_DEFnano</u> =1	GNU	RTOS	CH2		ITF_LIB

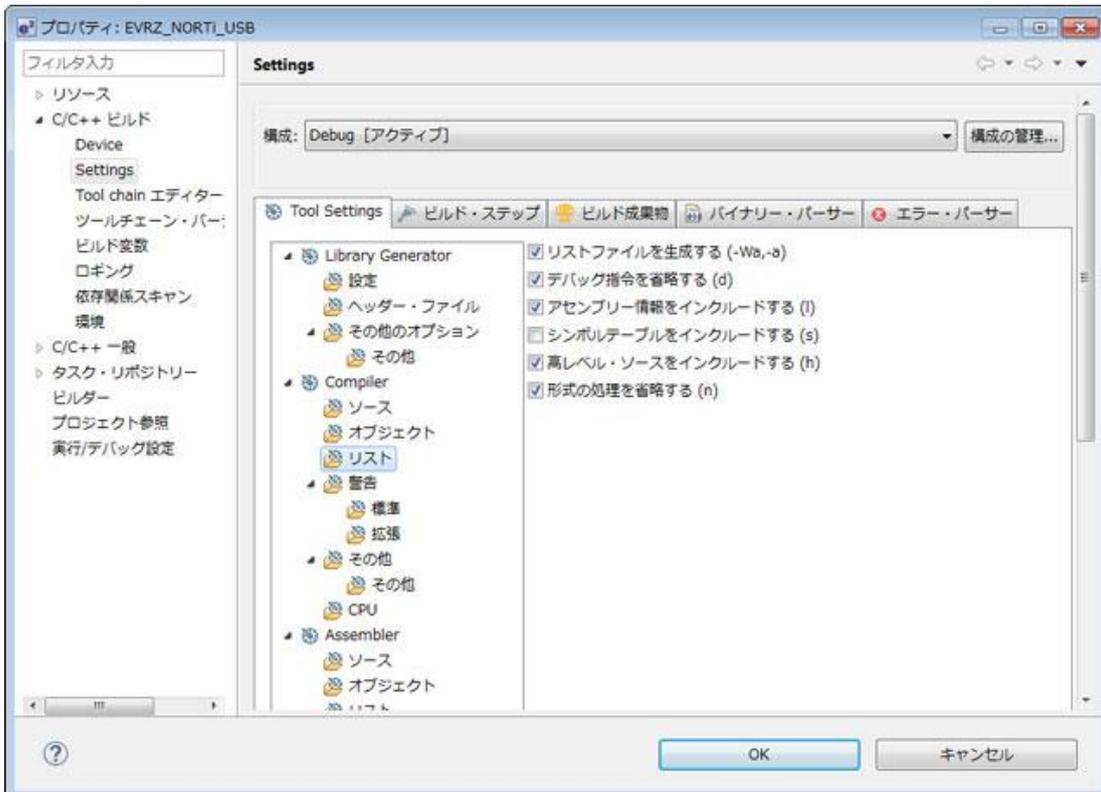
注* 1

「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。

3) 「オブジェクト」設定画面

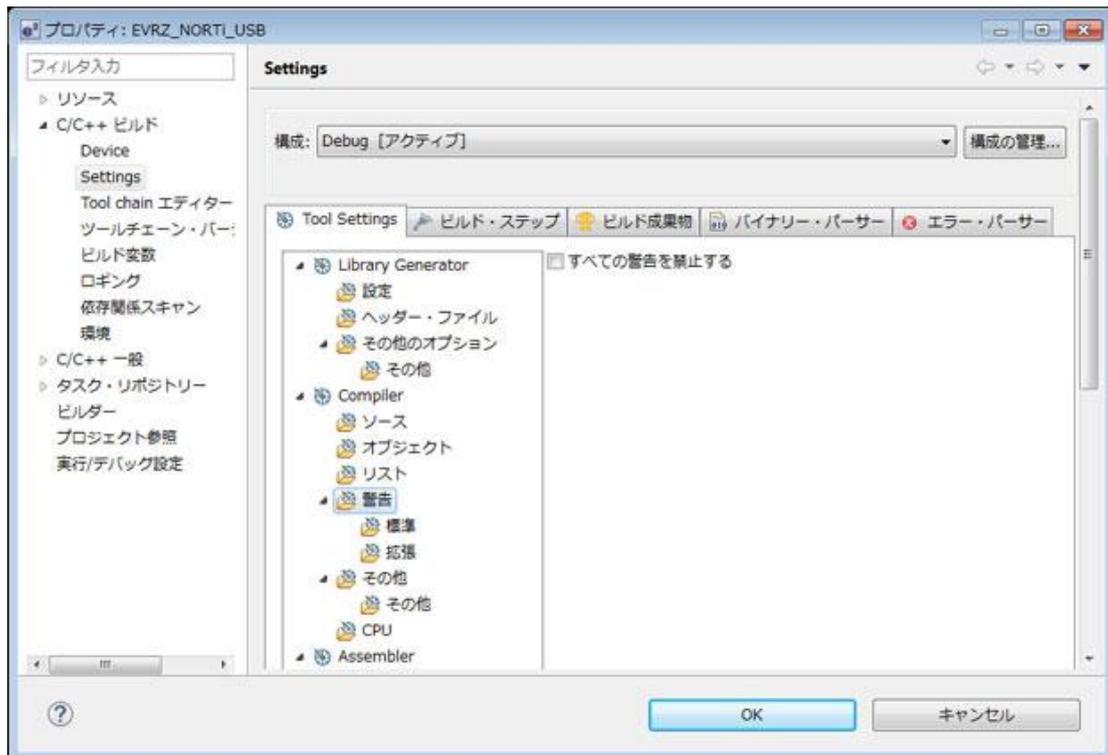


4) 「リスト」設定画面

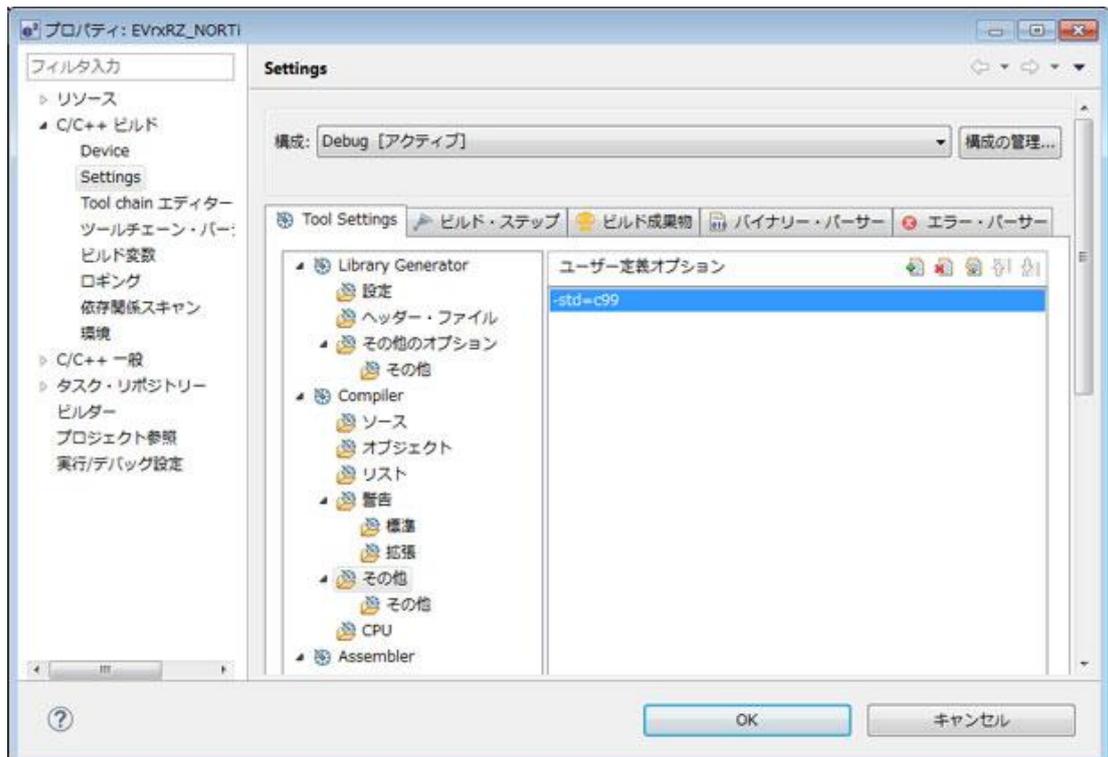


必要に応じて設定して下さい。

5) 「警告」 設定画面 (デフォルト)

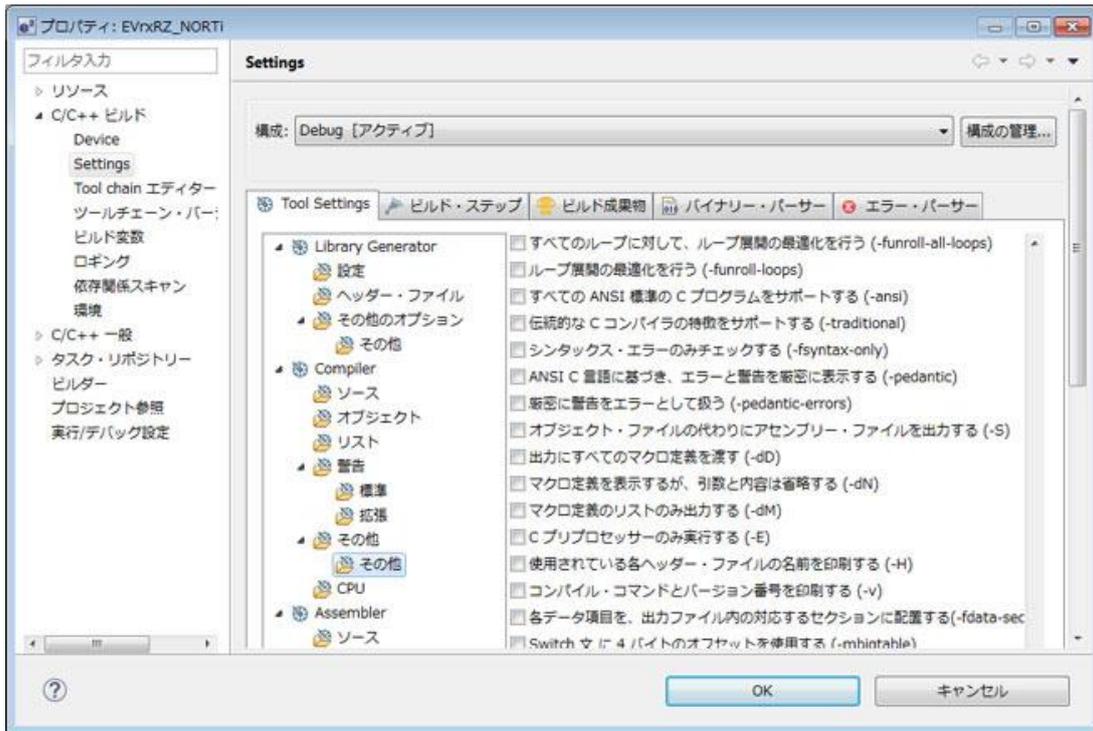


6) 「その他」 設定画面

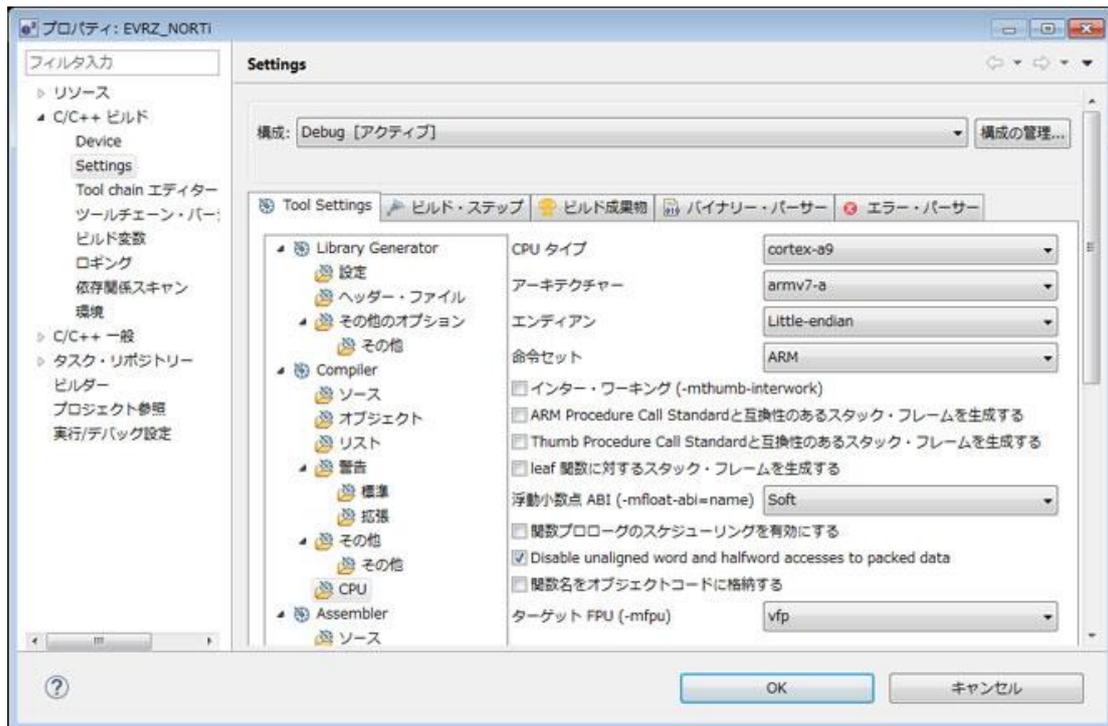


-std=c99	C 言語仕様を C99 に指定する。
----------	--------------------

7) 「その他ーその他」 設定画面 (デフォルト)



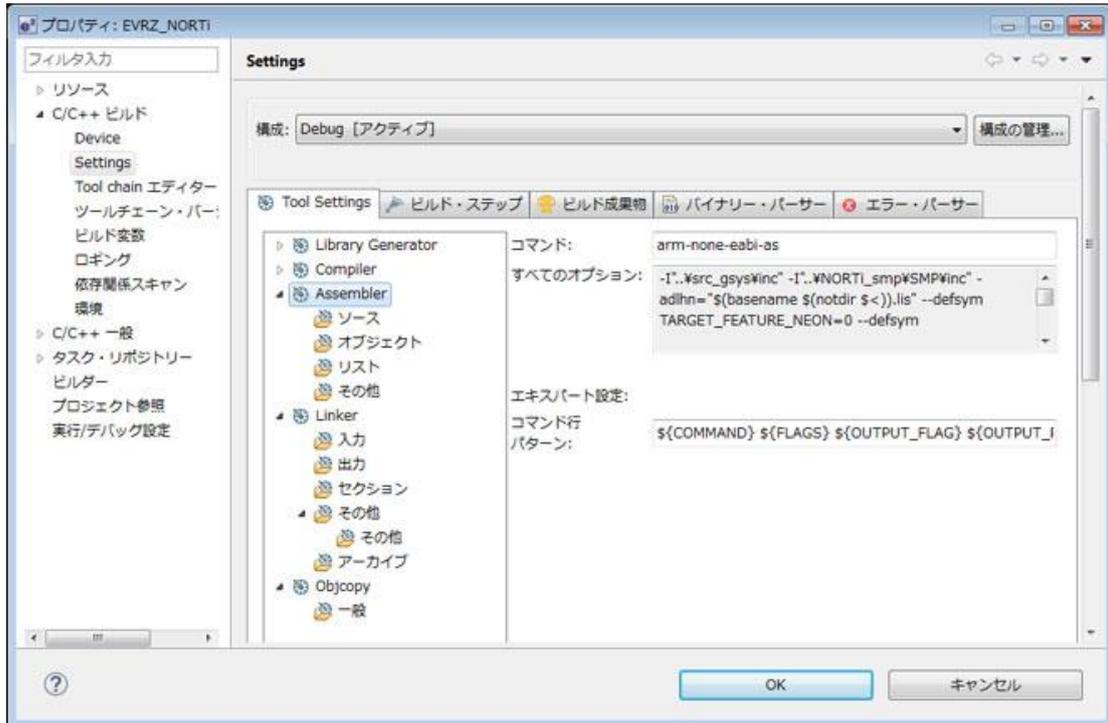
8) 「CPU」 設定画面



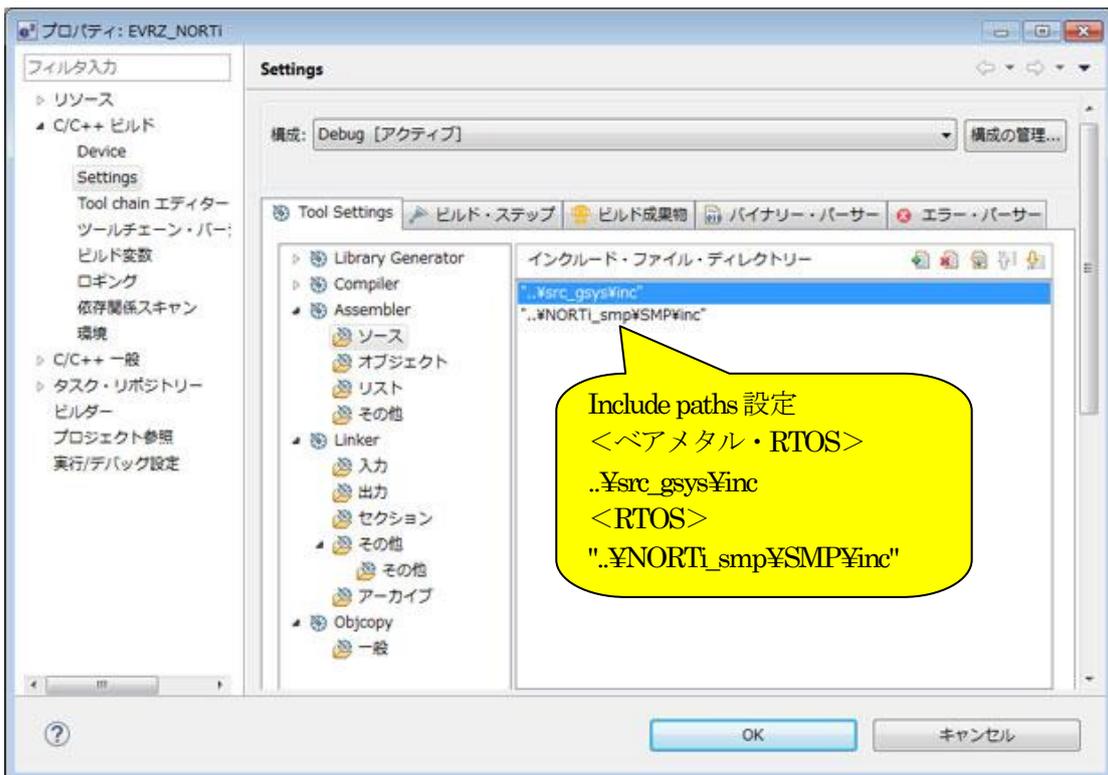
CPU タイプ	cortex-a9
アーキテクチャー	armv7-a
エンディアン	Little-endian
命令セット	ARM を選択
浮動小数点ABI(-mfloat-abi=name)	Soft
	Softfp (VFP/NEON 使用時)
ターゲット FPU(-mfpu)	vfp
<input checked="" type="checkbox"/> Disable unaligned word and halfword accesses to packed data	

4. Assembler の設定

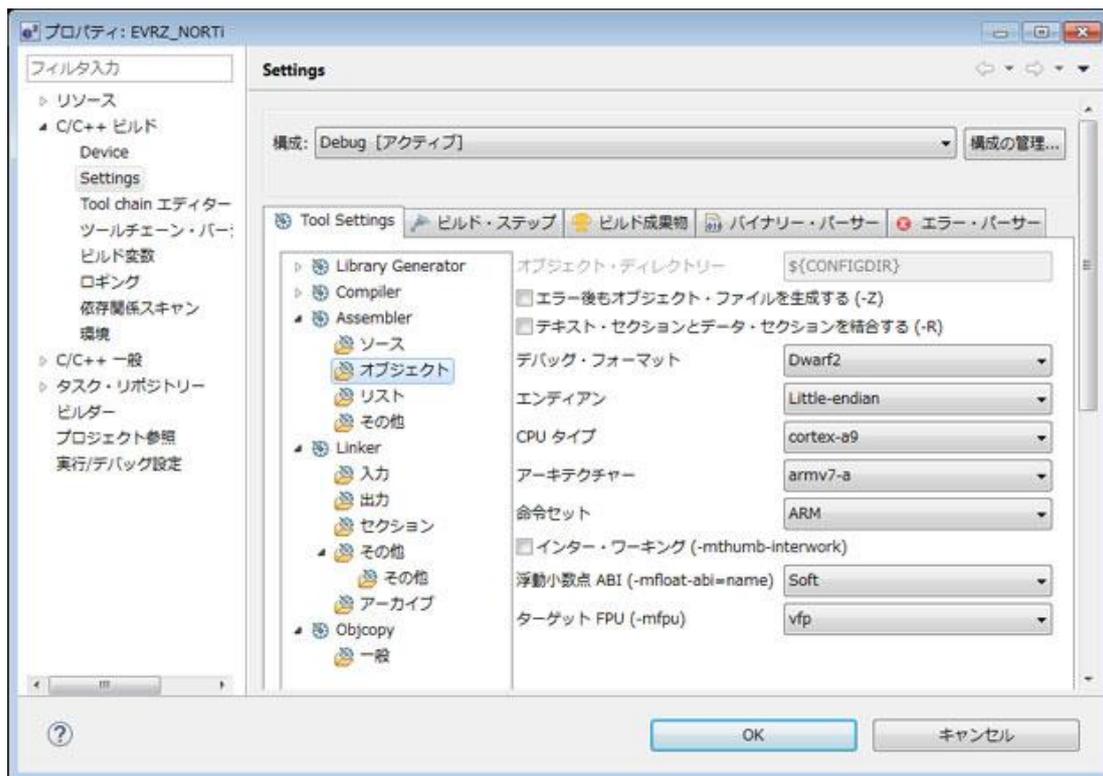
1) Assembler のすべてのオプションを表示



2) 「ソース」設定画面



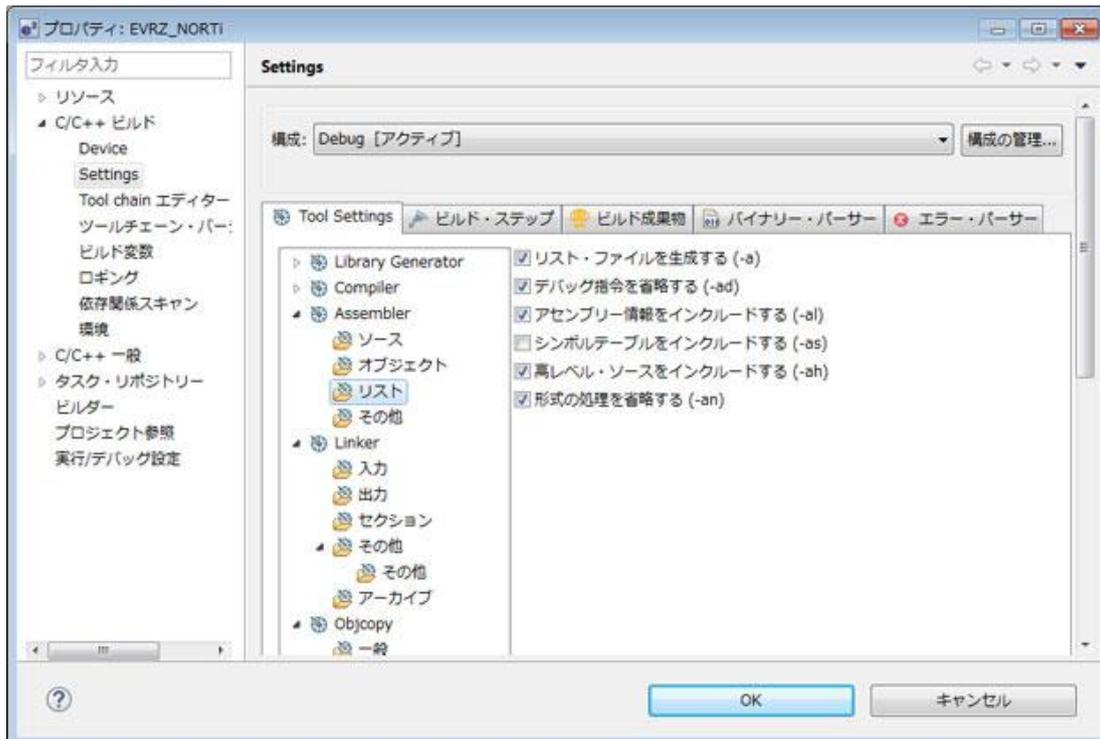
3) 「オブジェクト」設定画面



GCC C コンパイラの CPU 設定と同じにする。

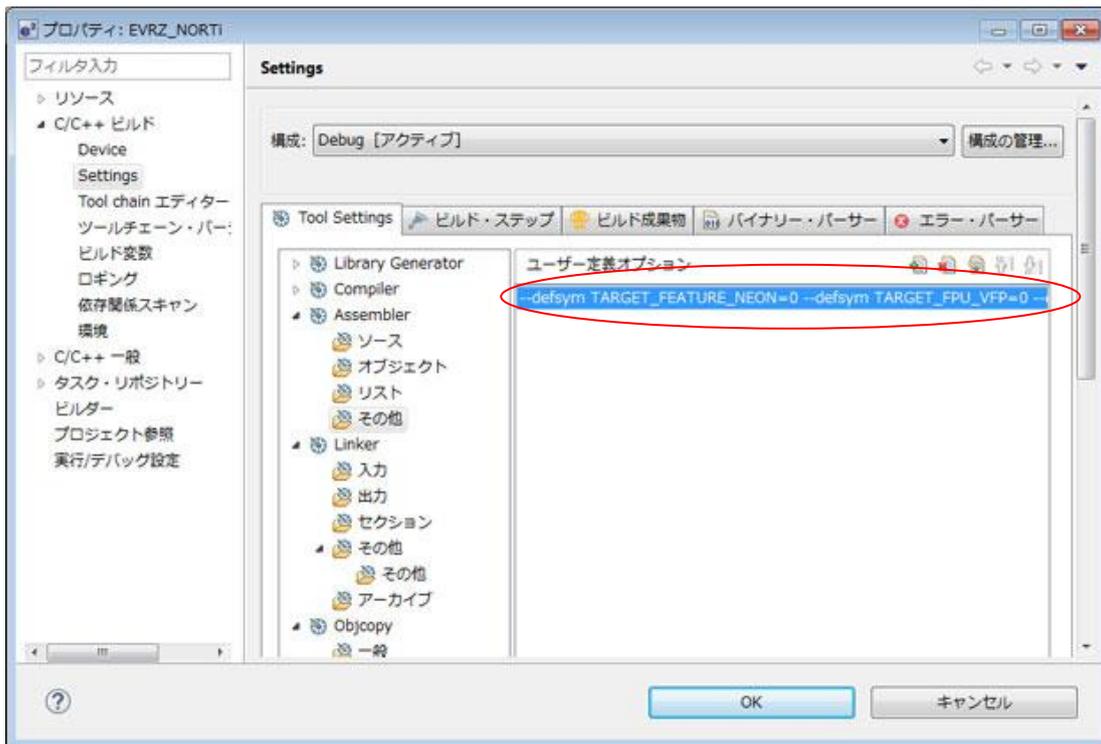
デバッグ・フォーマット	Dwarf2
エンディアン	Little-endian
CPU タイプ	cortex-a9
アーキテクチャー	armv7-a
命令セット	ARM
浮動小数点 ABI(-mfloat-abi=name)	Soft Softfp (VFP/NEON 使用時)
ターゲット FPU(-mfpu)	vfp

4) 「リスト」設定画面



必要に応じて設定して下さい。

5) 「その他」 設定画面



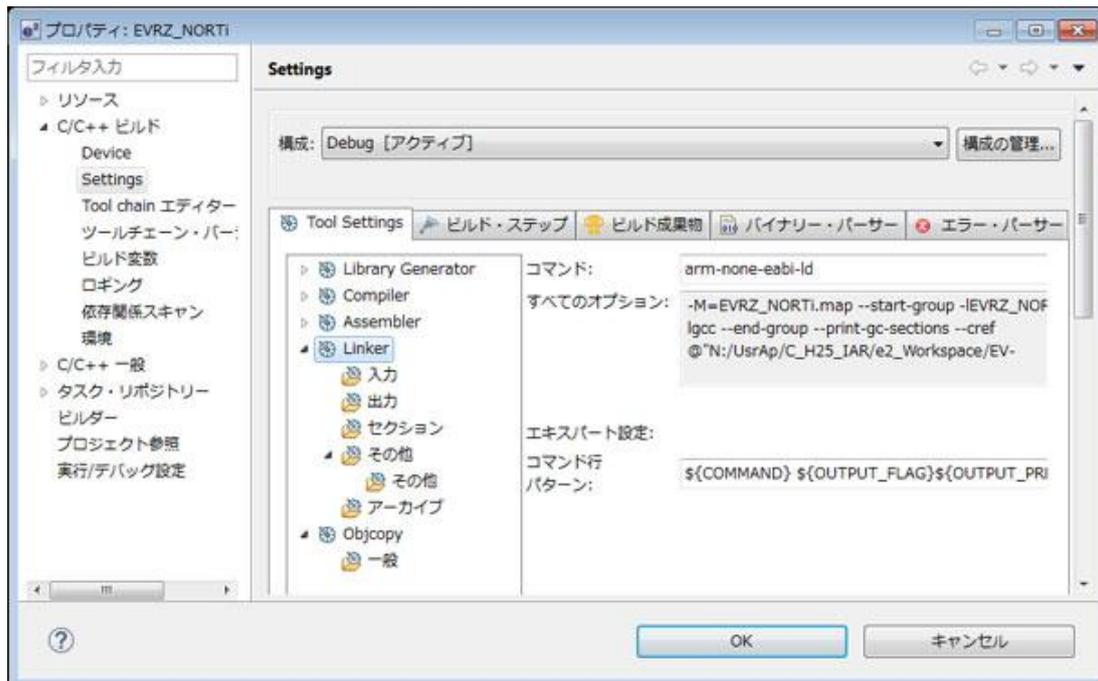
サンプルプロジェクトの追加オプション	
USER_Debug	-a=%\${OUTPUT_PREFIX}%\${OUTPUT}.lst
EVrxRZ_Sample	IF (浮動小数点ABI==Soft) {
EVrxRZ_Sample_USB	--defsym TARGET_FEATURE_NEON=0
EVRZ_Sample	--defsym TARGET_FPU_VFP=0
EVRZ_Sample_USB	}
EVrxRZ_Norti	ELSE IF (浮動小数点ABI==Softfp) {
EVrxRZ_Norti_USB	--defsym TARGET_FEATURE_NEON=1
EVRZ_Norti	--defsym TARGET_FPU_VFP=1
EVRZ_Norti_USB	}
	注* 1
	--defsym __USED_DEFnano_=1

注* 1

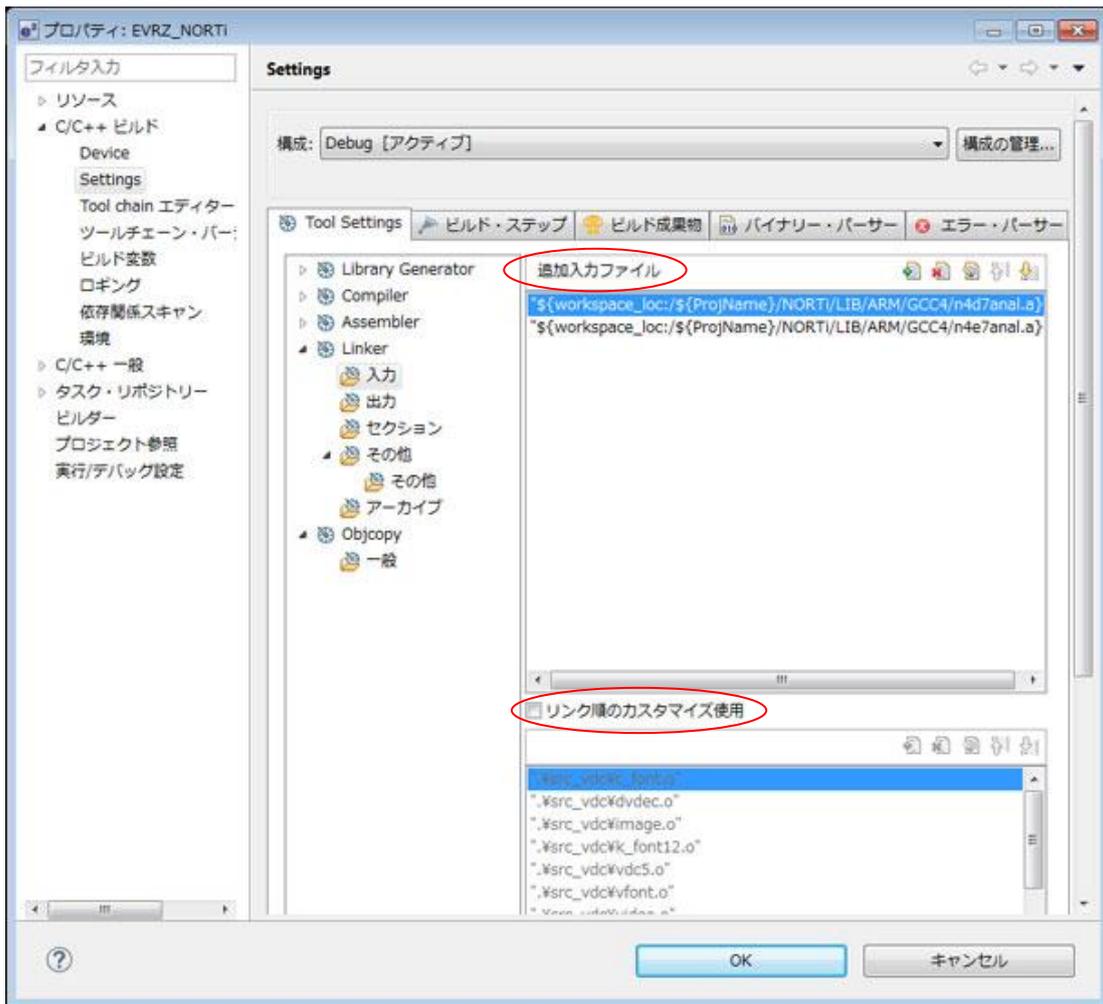
「__USED_DEFnano_=0」と使用しない側に定義しても内蔵 RAM へのダウンロードとシリアルフラッシュ ROM への書き込み操作は可能です。ただし、再操作する場合はターゲット側のリセット操作が必要になります。

5. Linker の設定

1) Linker のすべてのオプションを表示



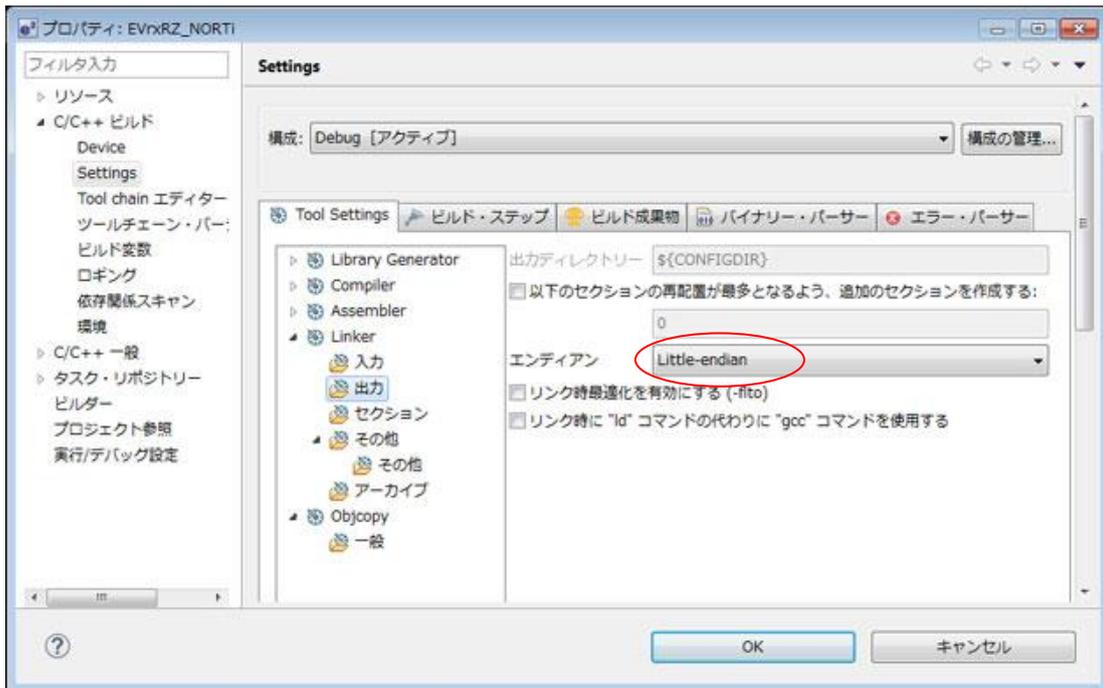
2) 「入力」 設定画面



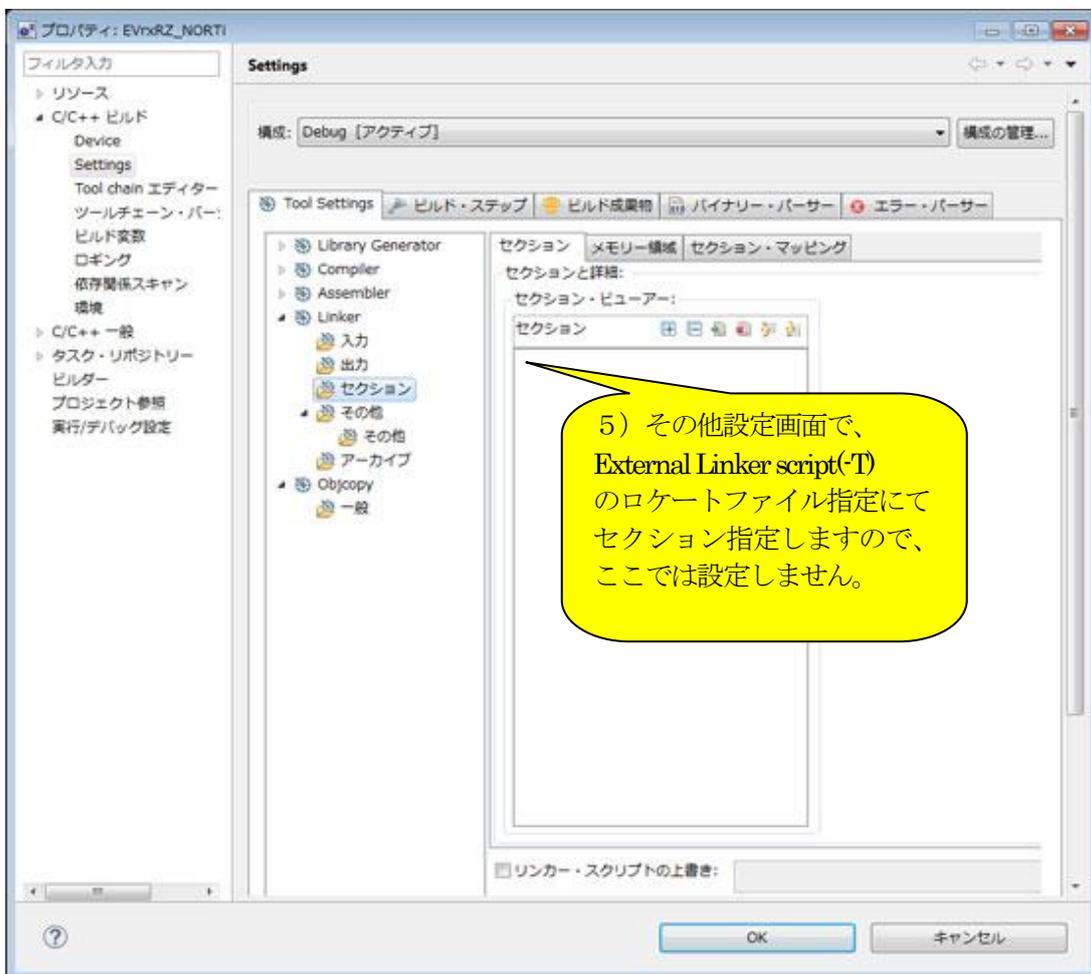
追加入力ファイル		
USER_Debug EVrxRZ_Sample EVRZ_Sample	なし	
EVrxRZ_Sample_USB EVRZ_Sample_USB	ITF_LIB/ITFUSBLib_RZA1H_A0_GNU.a (別売り)	
EVrxRZ_Norti EVRZ_Norti	n4d7anal.a (別売り)	
	Floating point mode:Soft 指定	n4e7anal.a (別売り)
	Floating point mode:Softfp 指定	n4e7aval.a (別売り)
EVrxRZ_Norti_USB EVRZ_Norti_USB	ITF_LIB/ITFUSBLib_RZA1H_A0_GNU.a (別売り)	
	n4d7anal.a (別売り)	
	Floating point mode:Soft 指定	n4e7anal.a (別売り)
	Floating point mode:Softfp 指定	n4e7aval.a (別売り)

リンク順のカスタマイズ使用	
<input checked="" type="checkbox"/> チェック	各モジュールのリンク順番をカスタマイズ可能になります。
<input type="checkbox"/> ノーチェック	IDE が設定した順番になります。

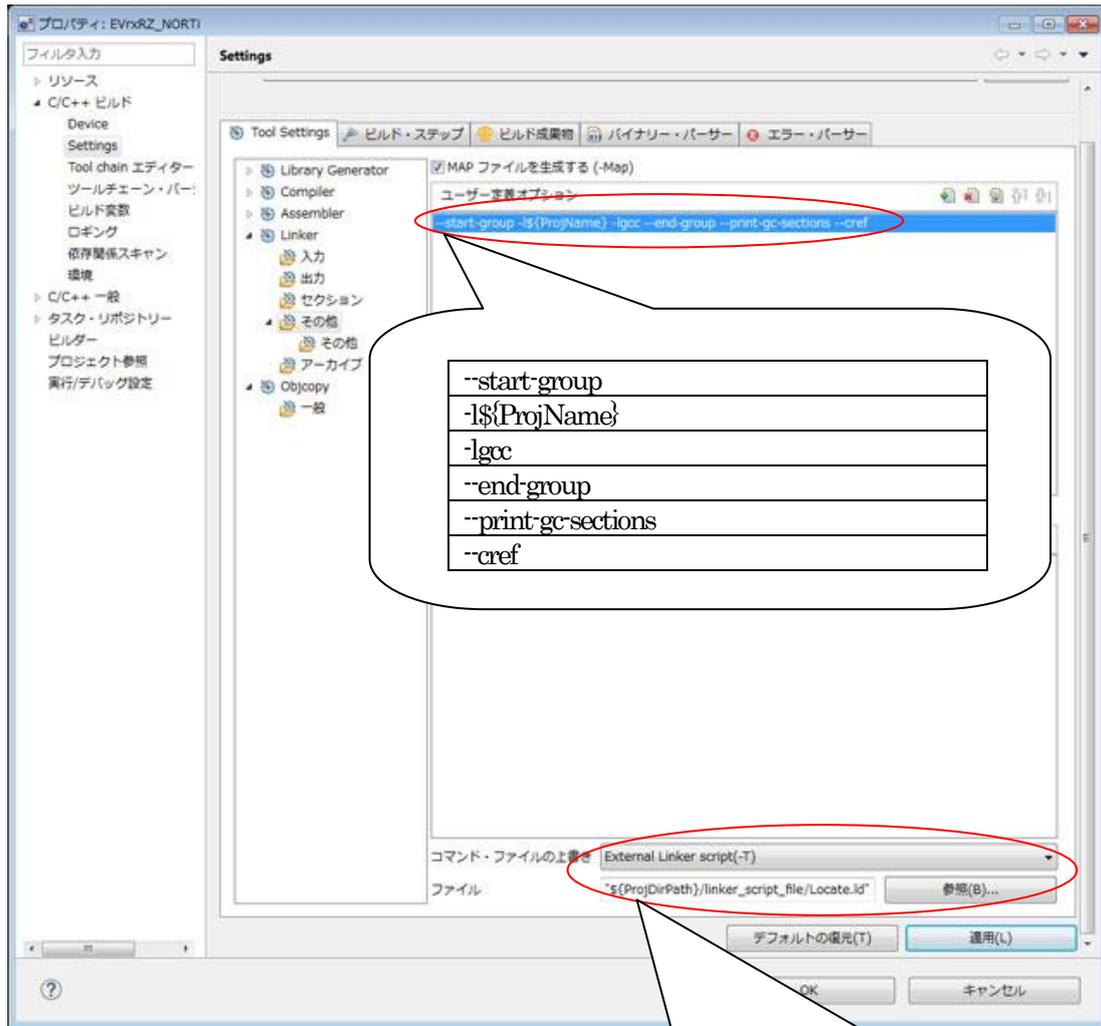
3) 「出力」 設定画面



4) 「セクション」 設定画面 (設定なし)

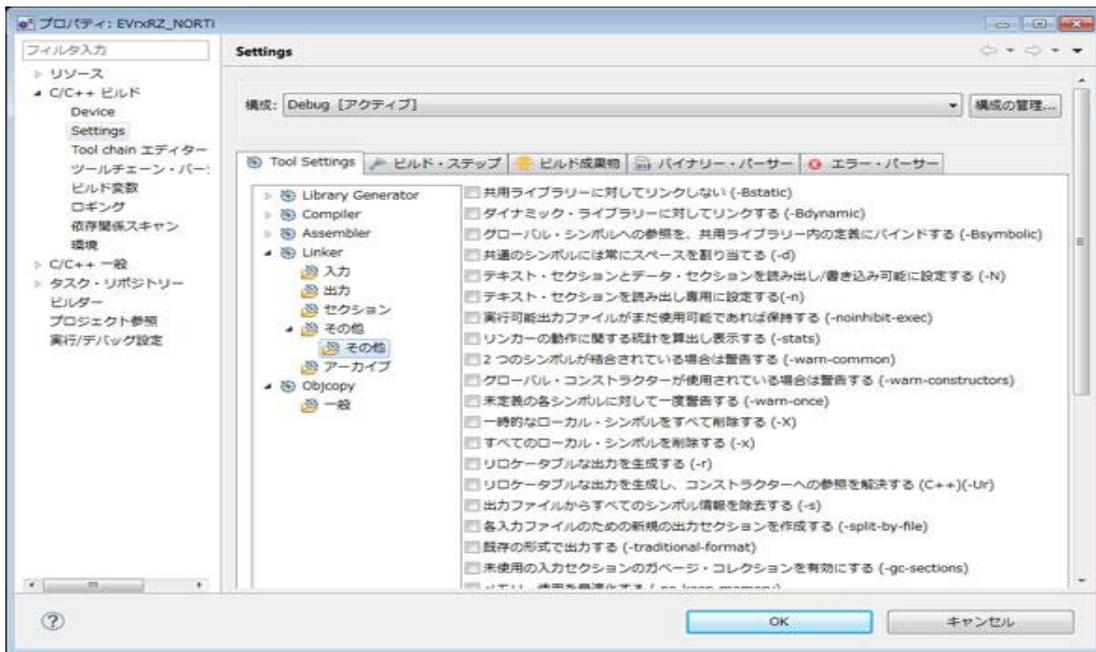


5) 「その他」 設定画面

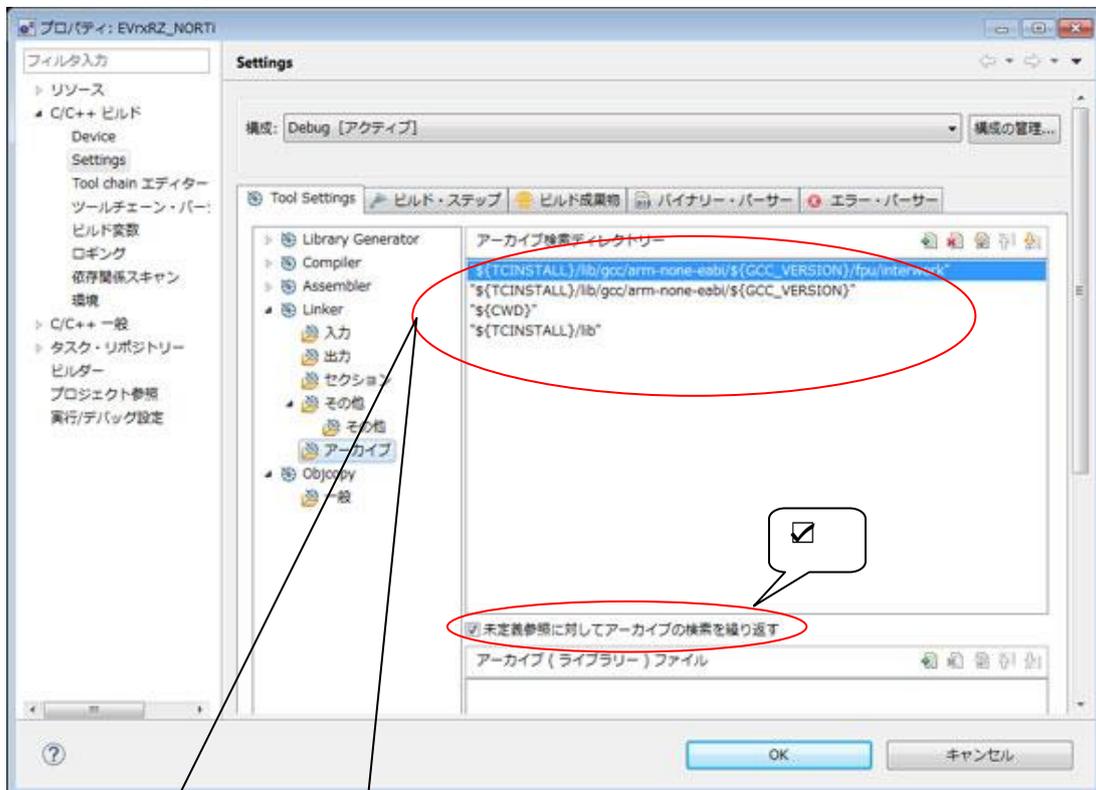


ロケートファイルにてセクションのアドレス指定をここで定義する。
External Linker script(-T)
"\${ProjDirPath}/linker_script_file/Locate.ld"

5) 「その他」設定画面 (デフォルト)



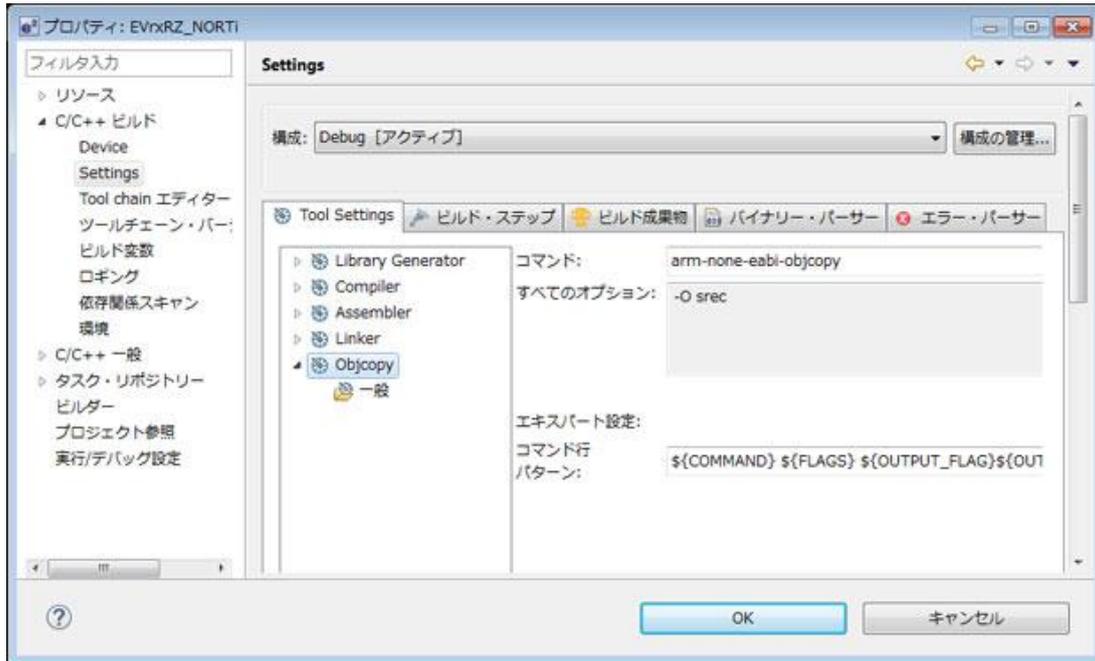
6) 「アーカイブ」設定画面



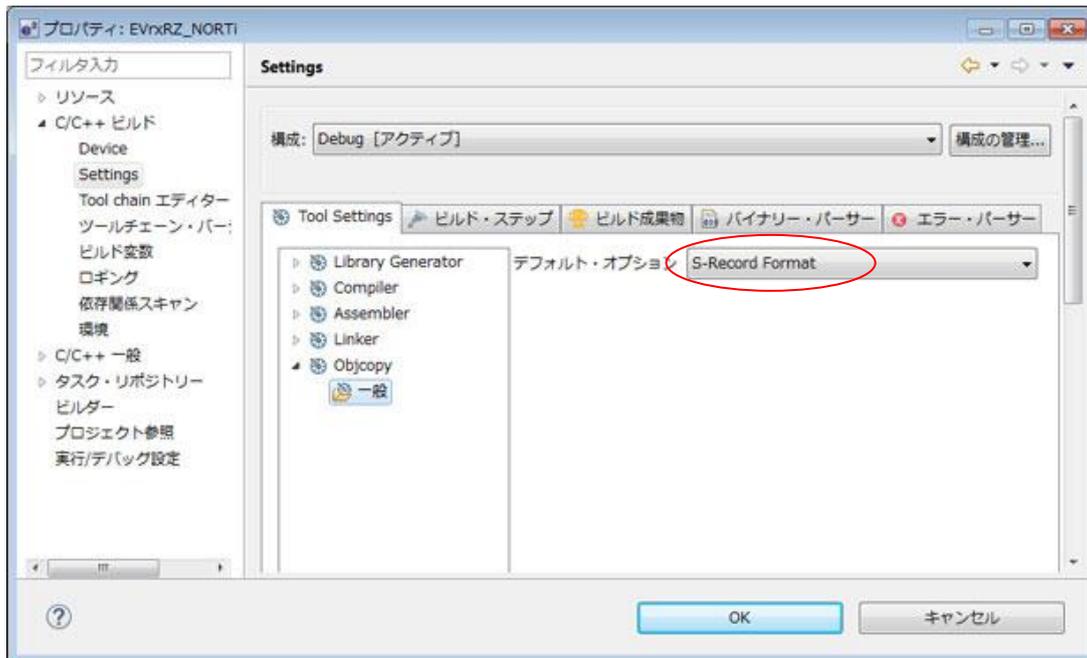
"\${TC_INSTALL}/lib/gcc/arm-none-eabi/\${GCC_STRING}/fpu/interwork"
"\${TCINSTALL}/lib/gcc/arm-none-eabi/\${GCC_VERSION}"
"\${CWD}"
"\${TCINSTALL}/lib"

6. Objcopy の設定

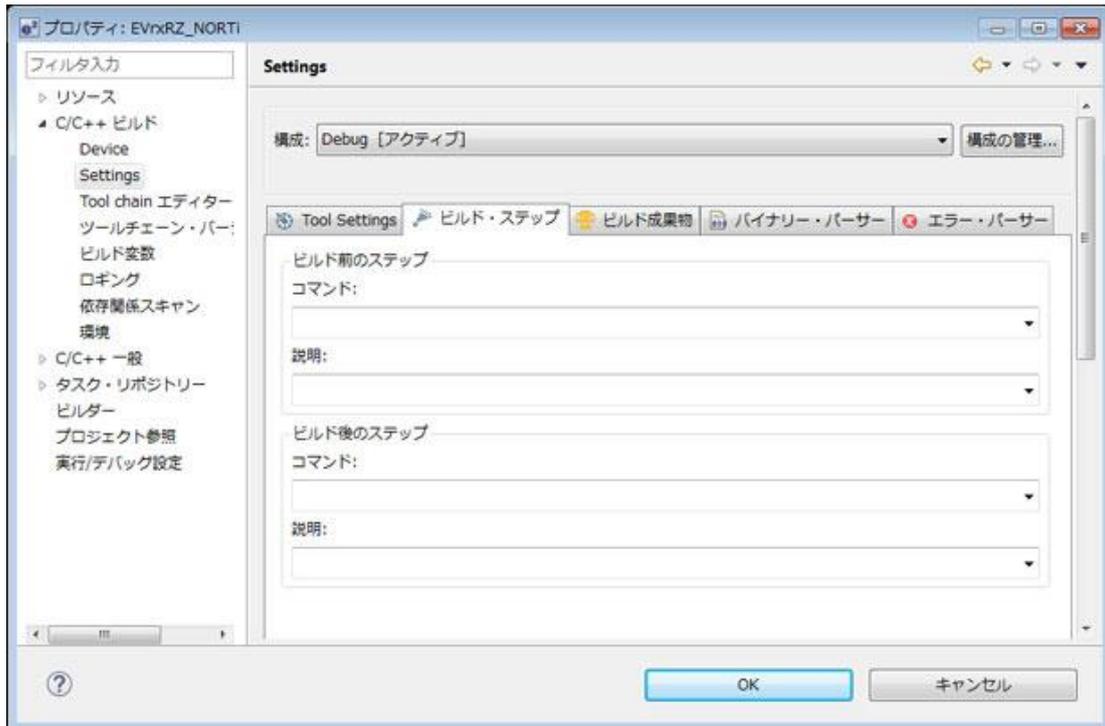
1) Objcopy のすべてのオプションを表示



2) 「一般」設定画面



7. 「ビルド・ステップ」設定画面（デフォルト）



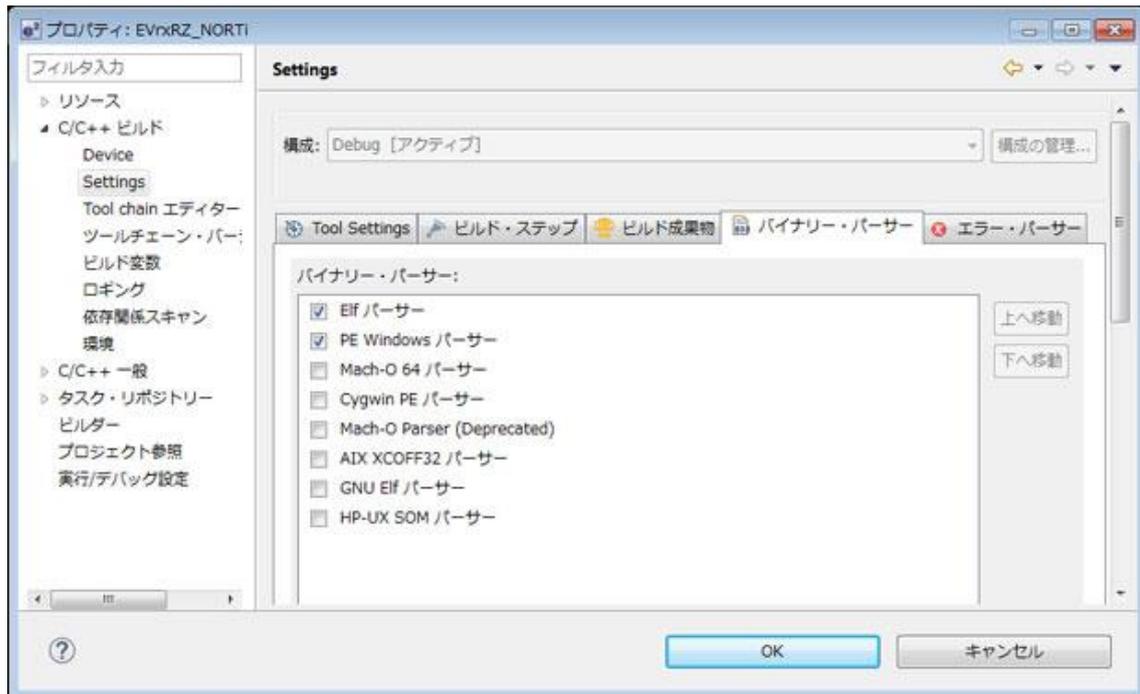
ビルド実行前と実行後に追加したいコマンドがある場合に設定します。

8. 「ビルド成果物」設定画面（デフォルト）

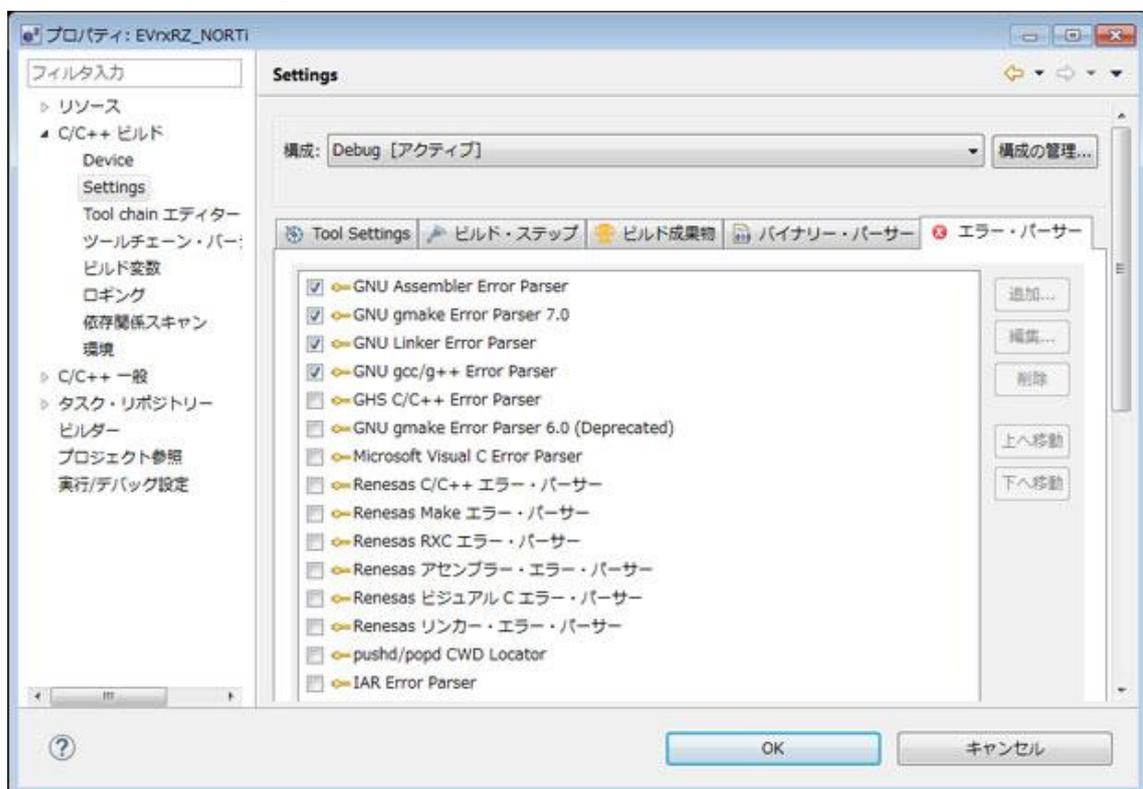


サンプルプロジェクトの設定	
成果物タイプ	
成果物名	プロジェクト名
成果物拡張子	mot

9. 「バイナリー・パーサー」 設定画面 (デフォルト)



10. 「エラー・パーサー」 設定画面 (デフォルト)



- 「GNU Assembler Error Parser」
- 「GNU gmake Error Parser 6.0(Deprecated)」
- 「GNU Linker Error Parser」
- 「GNU gcc/g++ Error Parser」

1 1. ロケートファイル(Locate.LD)について

ロケートファイルとは、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。ロケートファイル独自の予約語がありますので各自で調査して下さい。

サンプルプロジェクト[Locate.LD]の定義例1 「ベアメタル版」

```

MEMORY {
    ROM          (rx)  : ORIGIN = 0x20080000, LENGTH = 0x00300000
    STACK        (rw)  : ORIGIN = 0x20380000, LENGTH = 0x00080000
    CACHED_RAM   (rw)  : ORIGIN = 0x203A0000, LENGTH = 0x00380000
    UNCACHED_RAM (rw)  : ORIGIN = 0x60800000, LENGTH = 0x00200000
}

SECTIONS
{
    .text : {

        . = 0x00000000;
        Image$$VECTOR_TABLE$$Base = .;
        * (VECTOR_TABLE)          /* asm */

        . = 0x00000200;
        * (RESET_INIT_HANDLER)    /* asm */

        * (InRoot$$Sections)

        * (.text .text.*)
        * (.rodata .rodata.*)

    } > ROM
    .data : {
        __data_load = .;
        __data_start = LOADADDR(.data) + ( __data_load - ADDR(.data) );

        * (.data .data.*)

        __data_end = LOADADDR(.data) + ( . - ADDR(.data) );
        Image$$DATA$$Limit = .;
    } > ROM
    .bss : {
        __bss_start = .;
        * (.bss .bss.*)
        * (COMMON)
        __bss_end = .;
        * (_ebss _end)
        end = .;
    } > CACHED_RAM

```

```
.stack : {
    . = ALIGN( 0x10 );
    Image$$ARM_LIB_STACK$$ZI$$Base = .;
    . += 0x00008000;
    Image$$ARM_LIB_STACK$$ZI$$Limit = .;

    . = ALIGN( 0x10 );
    Image$$IRQ_STACK$$ZI$$Base = .;
    . += 0x00004000;
    Image$$IRQ_STACK$$ZI$$Limit = .;

    . = ALIGN( 0x10 );
    Image$$FIQ_STACK$$ZI$$Base = .;
    . += 0x00004000;
    Image$$FIQ_STACK$$ZI$$Limit = .;

    . = ALIGN( 0x10 );
    Image$$SVC_STACK$$ZI$$Base = .;
    . += 0x00004000;
    Image$$SVC_STACK$$ZI$$Limit = .;

    . = ALIGN( 0x10 );
    Image$$ABT_STACK$$ZI$$Base = .;
    . += 0x00004000;
    Image$$ABT_STACK$$ZI$$Limit = .;

    . = ALIGN( 0x4000 );
    Image$$TTB$$ZI$$Base = .;
    . += 0x00004000;
    Image$$TTB$$ZI$$Limit = .;
} > STACK

.uncached_RAM (NOLOAD) : {
    * (VIDEO_BSS)
} > UNCACHED_RAM
}
```

サンプルプロジェクト[Locate.LD]の定義例2 「NORTi版」

```

MEMORY {
  SYSTEM_RAM   (rwx) : ORIGIN = 0x20080000, LENGTH = 0x00680000
  STACK_MEM    (rw)  : ORIGIN = 0x20700000, LENGTH = 0x00100000
  STACK_ABT    (rw)  : ORIGIN = 0x20800000, LENGTH = 0x00004000
  MMU_TTB      (rw)  : ORIGIN = 0x20804000, LENGTH = 0x00004000
  UNCACHED_RAM (rw)  : ORIGIN = 0x60808000, LENGTH = 0x001F8000
}

SECTIONS
{
  .text : {
    PROVIDE(__ro_start__ = .);
    PROVIDE(__copy_start__ = .);

    . = 0x00000000;
    Image$$VECTOR_TABLE$$Base = .;
    * (VECTOR_TABLE)          /* asm */

    . = 0x00000200;
    * (RESET_INIT_HANDLER)   /* asm */

    * (InRoot$$Sections)

    * (.text .text.*)
    * (.rodata .rodata.*)
    PROVIDE(__ro_end__ = .);
  } > SYSTEM_RAM

  .data : {
    . = ALIGN(8);
    PROVIDE(__data_start__ = .);
    * (.data .data.*)
    PROVIDE(__data_end__ = .);
    . = ALIGN(8);
    PROVIDE(__copy_end__ = .);
  } > SYSTEM_RAM

  .bss : {
    . = ALIGN(8);
    PROVIDE(__bss_start__ = .);
    * (.bss .bss.*)
    * (COMMON)
    PROVIDE(__bss_end__ = .);
  } > SYSTEM_RAM

```

```
.MPLMEM ALIGN(8) : {
    * (.MPLMEM)
} > SYSTEM_RAM

.STKMEM ALIGN(8) : {
    * (.STKMEM)
} > SYSTEM_RAM

.stack (NOLOAD) : {
    PROVIDE(STARTOF_STACK = .);
    . += LENGTH(STACK_MEM);
    . = ALIGN(8);
    PROVIDE(STACK = .);

.stack_abt (NOLOAD) : {
    Image$$ABT_STACK$$ZI$$Base = .;
    . += LENGTH(STACK_ABT);
    Image$$ABT_STACK$$ZI$$Limit = .;
} > STACK_ABT

.ttb (NOLOAD) : {
    Image$$TTB$$ZI$$Base = .;
    . += LENGTH(MMU_TTB);
    Image$$TTB$$ZI$$Limit = .;
} > MMU_TTB

.uncached_RAM (NOLOAD) : {
    * (NOCACHE_AREA)
} > UNCACHED_RAM
}
```

1 2. 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。

1) 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」

<ベアメタル版>

```

Start:
@
@ Entry point for the Reset handler
@
vector_table:
LDR pc, =Reset_handler      @; Start+0x0000: リセット
LDR pc, =Undefined_handler  @; Start+0x0004: 未定義命令
LDR pc, =Svc_handler        @; Start+0x0008: ソフトウェア割り込み
LDR pc, =Prefetch_handler   @; Start+0x000c: プリフェッチアポート
LDR pc, =Abort_handler      @; Start+0x0010: データアポート
LDR pc, =Reserved_handler   @; Start+0x0014: Reserved
LDR pc, =Irq_handler        @; Start+0x0018: IRQ
LDR pc, =Fiq_handler        @; Start+0x001c: FIQ(NMI)
@
@ SFROM に登録したローダーに渡す情報
@
Info_table:
.long Image$$VECTOR_TABLE$$Base @; Start+0x0020 ①内蔵 RAM 転送先の開始アドレス
.long Image$$DATA$$Limit        @; Start+0x0024: ②内蔵 RAM 転送先の終了アドレス(+1)
.long vector_table              @; Start+0x0028: ③初期 PC 値
.long 0                        @; Start+0x002C: ④デバッグモードフラグ
                                @; DEFnano を未使用にして、USB0 を開放する場合は、
                                @; 0xDEF0DEF0 を定義する。
Info_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。

【注意事項】
 ④で「DEFnano 未使用」コード「0xDEF0DEF0」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。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「0x2_002C」を未使用コード「0xDEF0DEF0」以外の数値を直接書き込んでください。

2) 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」
 <NORT i 版>

```

Start:
@
@ Entry point for the Reset handler
@
vector_table:
LDR pc, =Reset_Handler      @; Start+0x0000:リセット
LDR pc, =Undefined_Handler  @; Start+0x0004:未定義命令
LDR pc, =Svc_Handler        @; Start+0x0008:ソフトウェア割り込み
LDR pc, =Prefetch_Handler   @; Start+0x000c:プリフェッチアポート
LDR pc, =Abort_Handler      @; Start+0x0010:データアポート
LDR pc, =Reserved_Handler   @; Start+0x0014:Reserved
LDR pc, =IRQ_Handler        @; Start+0x0018:IRQ
LDR pc, =FIQ_Handler        @; Start+0x001c:FIQ(NMI)
@
@ SFROMに登録したローダーに渡す情報
@
Info_table:
.long _copy_start_          @; Start+0x0020 ①内蔵 RAM 転送先の開始アドレス
.long _copy_end_ + 1        @; Start+0x0024:②内蔵 RAM 転送先の終了アドレス(+1)
.long __RESET               @; Start+0x0028:③初期 PC 値
.long 0                     @; Start+0x002C:④デバッグモードフラグ
                             @; DEFnanoを未使用にして、USB0を開放する場合は、
                             @; 0xDEF0DEF0を定義する。
Info_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。

【注意事項】
 ④で「DEFnano 未使用」
 コード「0xDEF0DEF0」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。
 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「0x2_002C」を未使
 用コード「0xDEF0DEF0」以外の数値を直接書
 き込んでください。

以上です。

1 3. 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を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。

1 4. 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。

1 5. 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
- ・その他

〒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>