RH850コンパイラ CC-RH
スタートアップ・ルーチン

ルネサス システムデザイン株式会社
ツールビジネス事業部 ツール技術部

2014/10/7 Rev. 1.00

R20UT3214JJ0100

スタートアップ・ルーチンの役割

スタートアップ・ルーチンとは、RH850をリセットしたあとmain 関数を実行する前までに実行するルーチンのことです。CC-RHではスタートアップ・ルーチンはアセンブリ言語で記述して頂く必要があります。

なお、CS+で新規プロジェクトを作成した場合には、スタートアップ・ルーチンのサンプルとして”cstart.asm”がプロジェクトに登録されています。本ファイルでは主に以下の処理を行っています。必要に応じてカスタマイズしてご使用ください。

- スタック領域の確保
- セクション初期化テーブルの確保
- 各種ポインタの設定
- ハードウェアの初期化(ECC機能向け)
- RAM領域の初期化
- main関数への分岐

cstart.asm の説明(1/11)

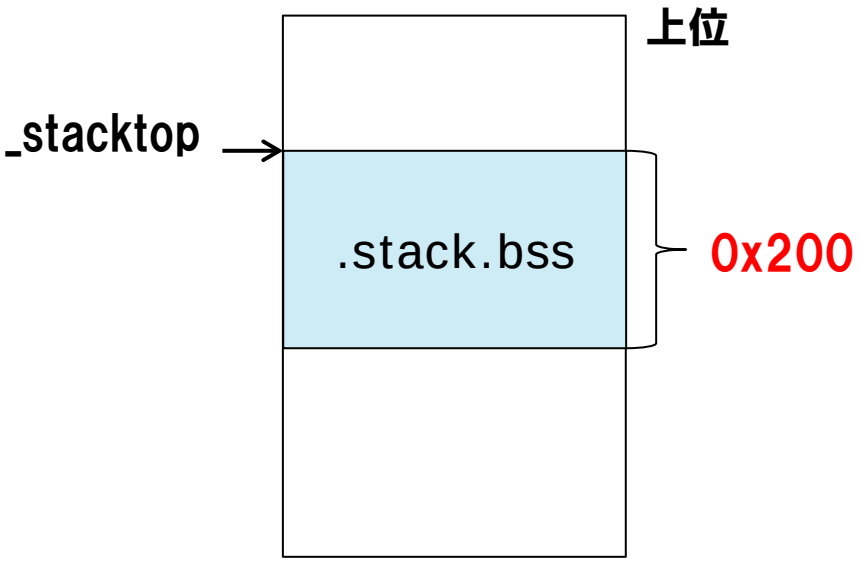
■ スタック領域の確保

```
;------------------------------------------------------------
;       system stack
;------------------------------------------------------------
STACKSIZE         .set    0x200
          .section ".stack.bss", bss
          .align   4
          .ds      (STACKSIZE)
          .align   4
_stacktop:
```

スタック領域は.stack.bss セクションに配置

デフォルトでは0x200バイト分のスタックが確保、必要に応じてサイズを変更

この後、ラベル“_stacktop”のアドレスをsp（スタック・ポインタ）に格納します。



cstart.asm の説明(2/11)

■ セクション初期化テーブルの確保1

セクション初期化テーブルとは、_INITSCT_RH関数に渡す引数用のテーブル

特殊シンボル：
アセンブラにおいて
- セクション名に"#_s"を付けると、そのセクションの先頭アドレス
- セクション名に"#_e"を付けると、そのセクションの終端アドレス

例：0x1000～0x2000番地に.dataセクションが配置されている場合

```
#__s.data は0x1000
#__e.data は0x2000 としてアセンブル
```

cstart.asm の説明(3/11)

■ セクション初期化テーブルの確保2

```
;------------------------------------------------------
;       section initialize table
;------------------------------------------------------
        .section        ".INIT_DSEC.const", const
        .align  4
        .dw     #__s.data,      #__e.data,      #__s.data.R

        .section        ".INIT_BSEC.const", const
        .align  4
        .dw     #__s.bss,       #__e.bss
```

data属性セクションの初期化テーブル“.INIT_DSEC.const”セクションには

ROMセクションの先頭アドレス，ROMセクションの終端アドレス，RAMセクションの先頭アドレス

bss属性セクションの初期化テーブル“.INIT_BSEC.const”セクションには

先頭アドレス，終端アドレス

がそれぞれ4バイト分確保

cstart.asm の説明(4/11)

■ セクション初期化テーブルの確保3

デフォルトでは.dataセクションと.bssセクションのテーブルが確保されていますが、お客様自身でセクションを追加した場合、本テーブルにも同様のフォーマットで追記してください。

例：.sdata/.sbssセクションと.zdata23/.zbss23セクションを追加した場合

-rom=.sdata=.sdata.R -rom=.zdata23=.zdata23.R 指定時

```
.section        ".INIT_DSEC.const", const
.align  4
.dw     #__s.data,      #__e.data,      #__s.data.R
.dw     #__s.sdata,     #__e.sdata,     #__s.sdata.R
.dw     #__s.zdata23,   #__e.zdata23,   #__s.zdata23.R

.section        ".INIT_BSEC.const", const
.align  4
.dw     #__s.bss,       #__e.bss
.dw     #__s.sbss,      #__e.sbss
.dw     #__s.zbss23,    #__e.zbss23
```

cstart.asm の説明(5/11)

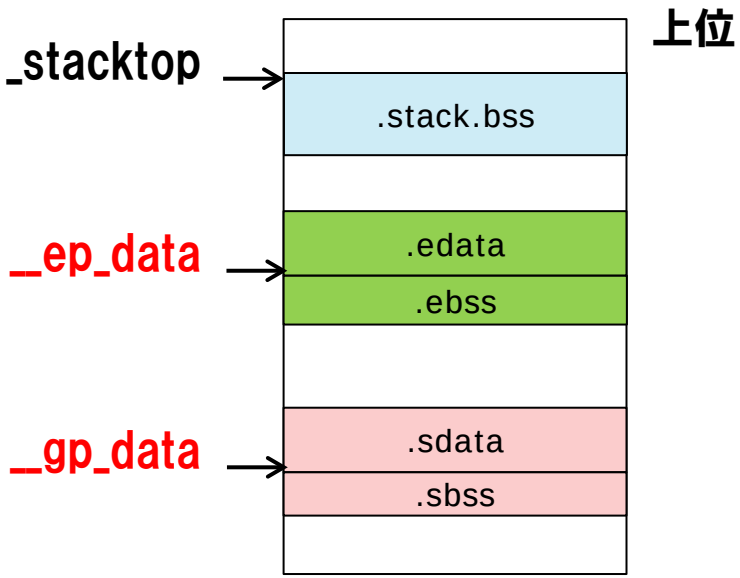
■ 各種ポインタの設定1

```
;------------------------------------------------------------
;       startup
;------------------------------------------------------------
        .section ".text", text
        .public  __start
        .align   2
__start:
        mov     #_stacktop, sp          ;  set sp register
        mov     #__gp_data, gp          ;  set gp register
        mov     #__ep_data, ep          ;  set ep register
```

“__gp_data”、”__ep_data”はリンカが自動的に生成

これらのシンボルをそれぞれsp（スタック・ポインタ）、gp（グローバル・ポインタ）、ep（エレメント・ポインタ）に設定

gp/epをベースレジスタとするセクションが存在する場合、gp/epの設定が必要です。



cstart.asm の説明(6/11)

■ 各種ポインタの設定2

“__gp_data”、”__ep_data”はリンカが自動的に生成しますが、ユーザーが明示的に指定することも可能です。

例：“__gp_data”を0xfede5000に指定してgpに設定

```
        .public __gp_data
        __gp_data .equ 0xfede5000


...

mov     __gp_data, gp
```

cstart.asm の説明(7/11)

■ ハードウェアの初期化(ECC機能向け)1

```
;-----------------------------------------------------------
;       hdwinit
;-----------------------------------------------------------
        .section ".text", text
        .align   2
_hdwinit:
        mov     lp, r14                     ; save return address

        mov     PRIMARY_LOCAL_RAM_ADDR, r6
        mov     PRIMARY_LOCAL_RAM_END, r7
        jarl    _zeroclr4, lp               ; clear Primary local RAM

        省略:SECONDARY_LOCAL_RAM_ADDR から
            SECONDARY_LOCAL_RAM_END のゼロクリア
        省略:RETENTION_RAM_ADDR から
            RETENTION_RAM_END のゼロクリア

        mov     r14, lp
        jmp     [lp]
```

ECC機能向けにRAM領域をゼロ初期化

⇒ ゼロ初期化されていないRAMをアクセスするとECCエラーが発生

cstart.asm の説明(8/11)

■ ハードウェアの初期化(ECC機能向け)2

ECC機能向けゼロ初期化に使用しているRAMの先頭アドレス、終端アドレスは
cstart.asm 内で定義しています。**対象デバイス向けにアドレスを変更**してください。

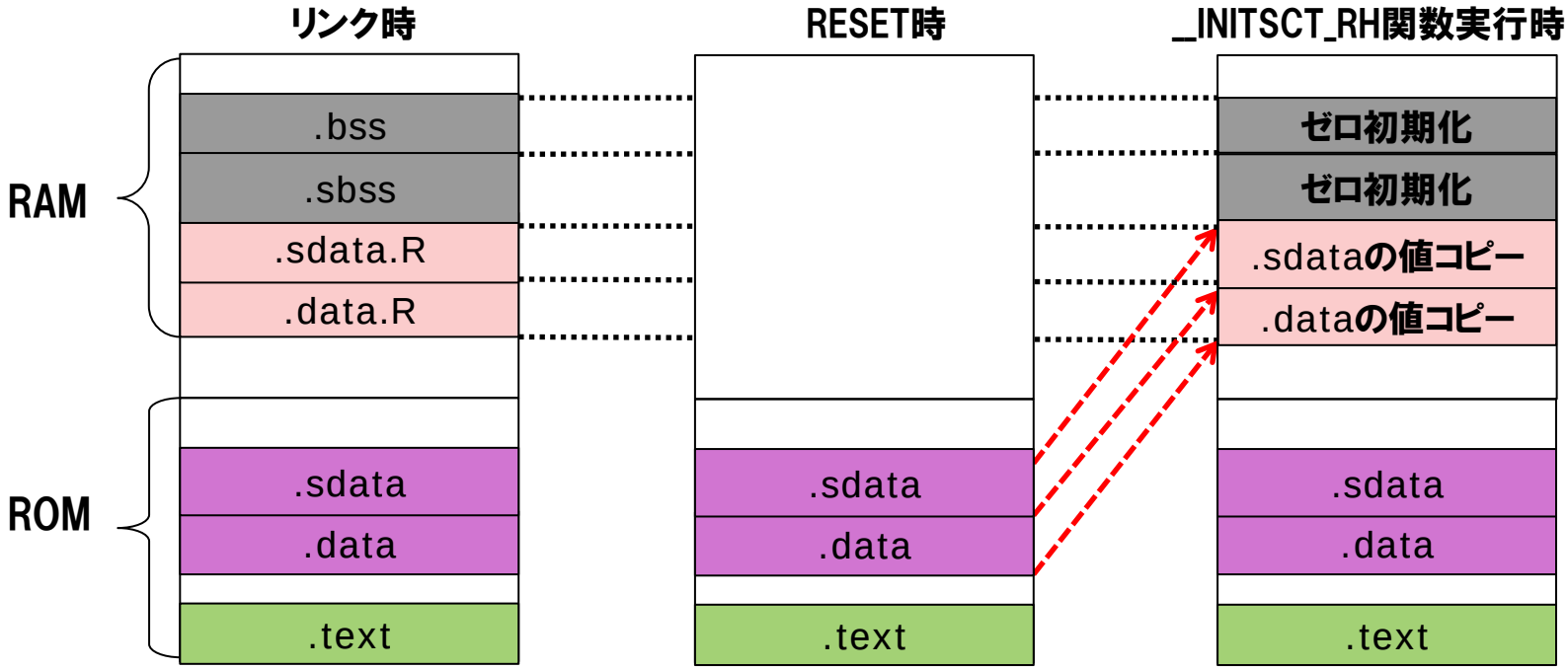
```
;-------------------------------------------------------
;target dependence informations (specify values suitable to your system)
;-------------------------------------------------------
$if 1
        ; RAM address
        PRIMARY_LOCAL_RAM_ADDR          .set    0xfede0000
        PRIMARY_LOCAL_RAM_END           .set    0xfedfffff

        SECONDARY_LOCAL_RAM_ADDR        .set    0xfedd8000
        SECONDARY_LOCAL_RAM_END         .set    0xfeddffff

        RETENTION_RAM_ADDR              .set    0xfee00000
        RETENTION_RAM_END               .set    0xfee07fff
```

cstart.asm の説明(9/11)

■ RAM領域の初期化1

```
        mov     #__s.INIT_DSEC.const, r6
        mov     #__e.INIT_DSEC.const, r7
        mov     #__s.INIT_BSEC.const, r8
        mov     #__e.INIT_BSEC.const, r9
        jarl32  __INITSCT_RH, lp        ;  initialize RAM area
```

_INITSCT_RH関数を使用して、data属性セクションの値のコピー（ROM化）、bss属性セクション領域のゼロ初期化を行います。



cstart.asm の説明(10/11)

RAM領域の初期化2

data属性セクションの値のコピー、あるいはbss属性セクションのゼロ初期化のどちらかが不要の場合は、_INITSCT_RH関数の引数（r6～r9）に0を指定してください。

例：data属性セクションのゼロ初期化が不要の場合

```
        mov     0, r6
        mov     0, r7
        mov     #__s.INIT_BSEC.const, r8
        mov     #__e.INIT_BSEC.const, r9
        jarl32  __INITSCT_RH, lp        ;  initialize RAM area
```

例：bss属性セクションのゼロ初期化が不要の場合

```
        mov     #__s.INIT_DSEC.const, r6
        mov     #__e.INIT_DSEC.const, r7
        mov     0, r8
        mov     0, r9
        jarl32  __INITSCT_RH, lp        ;  initialize RAM area
```

cstart.asm の説明(11/11)

main 関数への分岐

```
        stsr    5, r10, 0               ; r10 <- PSW
        ldsr    r10, 3, 0               ; FEPSW <- r10

        mov     #_exit, lp              ; lp <- #_exit
        mov     #_main, r10
        ldsr    r10, 2, 0               ; FEPC <- #_main

        ; apply PSW and PC to start user mode
        feret
```

main 関数に分岐します。これでスタートアップ・ルーチンの処理は終了です。

ルネサス システムデザイン株式会社