NCES

ATK2
外部仕様書
(Nios2 依存部)

Ver.1.2.0

2014/03/12

名古屋大学大学院情報科学研究科附属 組込みシステム研究センター

＜目次＞

1. 概要

1.1　本文書の概要

本文書は「次世代車載システム向け RTOS 外部仕様書」で ATK2 でターゲット定義となっている仕様に関して，ATK2 の Nios2 依存部における仕様を規定するものである．

1.2　関連文書

以下の表は，本文書から参照している文書，または本文書を理解するために必要な文書である．

文書名	バージョン
Nios II Processor Reference Handbook	Ver.11.0
Embedded Peripherals IP User Guide	Ver.11.0
次世代車載システム向け RTOS 要求仕様書	Ver.3.0.0
次世代車載システム向け RTOS ハードウェア要求仕様書	Ver.3.0.0
次世代車載システム向け RTOS 用語集	Ver.3.0.0
次世代車載システム向け RTOS 外部仕様書	Ver.3.2.0

1.3　凡例

本文書では，以下に示す仕様番号を用いて仕様を区別して管理を行う．仕様番号は，要求事項にのみ付与することを基本とする．

仕様番号	内容
【COSxxx】	関連文書 OSEK/VDX Operating System で規定された仕様. AUTOSAR 仕様では CoreOS として扱われるため, COS と表記を行う.
【OSxxx】	関連文書 AUTOSAR Specification of Operating System で規定された仕様．AUTOSAR 仕様で記述されている OS 仕様番号を用いる．本仕様に採用しなかった AUTOSAR 仕様についても，【OSxxx】で記載している.
【OSaxxx】	関連文書 AUTOSAR Specification of Operating System に記述されているが，仕様番号表記がないもの．AUTOSAR 仕様の要求事項であるため，本仕様では，仕様番号を付与する.
【NOSxxxx】	NCES で新規に規定した仕様.
【IOSxxx】	ATK2 の実装に依存した仕様.
【NiosIOSxxx】	ATK2 の Nios2 依存部の実装に依存した仕様.
〔COSxxx〕 〔OSxxx〕 〔NOSxxxx〕	本文中で上記の仕様番号を参照する際に使用する. 仕様定義である【COSxxx】【OSxxx】【NOSxxxx】と区別するため，〔COSxxx〕〔OSxxx〕〔NOSxxxx〕とする.

2. Nios2 ハードウェア構成

ATK2 を Nios2 で動作させるためのハードウェア構成について記述する．

2.1　対応ボード

Nios2 DE2-115

Nios2 2s180

2.2　ATK2 を動作させるための Nios2 の最小構成

- Nios2/f プロセッサコア
  - <MC>コア数は 2 以上
- シャドウレジスタセットは使用しない
- exception レジスタ有効
- ベクトル割込みコントローラ 1 セット
- メモリ
- メモリ保護ユニット(ATK2-SC3 を動作させる場合)
  - 使用したいメモリ保護領域数に応じた MPU リージョン数
  - MPU リージョン設定は，アドレスの上限下限で設定

また，サンプルプログラムを動作させるため，以下の構成も必要となる．

- Avalon タイマ
- JTAG UART
- <MC> Mutex 回路（コア数 × 2 + 2 個）
- <MC>コア ID の提供機構
- <MC>コア間割込み回路

2.3　メモリマップ

2.3.1　DE2-115 のメモリマップ

表 2-1　DE2-115 シングルコアのメモリマップ一覧

メモリ名	アドレス及び範囲
SDRAM	0x00000000 から 0x01ffffff
FLASH	0x04000000 から 0x04ffffff
SRAM	0x05000000 から 0x050fffff
リセットアドレス	0x05000000
例外アドレス	0x00000020
VIC ベースアドレス	0x0f000000
バージョンレジスタ	0x080001c0

メモリ名	アドレス及び範囲
タイマベースアドレス	0x08000160
JTAGUART ベースアドレス	0x080001a0
UART ベースアドレス	0x02000d00

表 2-2　DE2-115 マルチコアのメモリマップ一覧

メモリ名	コア 0 のアドレス及び範囲	コア 1 のアドレス及び範囲
SDRAM	0x00000000 から 0x01ffffff	
FLASH	0x04000000 から 0x04ffffff	
SRAM	0x05000000 から 0x051fffff	
リセットアドレス	0x08000000	
例外アドレス	0x08000020	
バージョンレジスタ	0x08100000	
スピンロッククロック用 Mutex アドレス	0x08200000	
IOC ロック用 Mutex アドレス	0x08200010	
VIC ベースアドレス	0x02001000	0x02101000
タイマベースアドレス	0x02000800	0x02100800
JTAGUART ベースアドレス	0x02000820	0x02100820
オンチップメモリ	0x02080000 から 0x020a0000	0x02180000 から 0x021a0000
コア間割込み機構のアドレス	0x02080c00	0x02180c00
タスクロック用 Mutex アドレス	0x02080c10	0x02180c10
カウンロック用 Mutex アドレス	0x02080c20	0x02180c20
UART ベースアドレス	0x02000d00	0x02100d00

2.3.2　2s180 のメモリマップ

表 2-3　2s180 のメモリマップ一覧

メモリ名	アドレス及び範囲
FLASH	0x00000000 から 0x00ffffff
ONCHIP(64KB)	0x01100000 から 0x0110ffff
ONCHIP(512KB)	0x05000000 から 0x0507ffff
SRAM	0x09000000 から 0x090fffff
SDRAM	0x0a000000 から 0x0affffff
リセットアドレス	0x05000000
例外アドレス	0x05000020

メモリ名	アドレス及び範囲
VIC ベースアドレス	0x01000c00
バージョンレジスタ	0x0f000000
タイマベースアドレス	0x01001000
JTAGUART ベースアドレス	0x010008a0

2.3.3　メモリリージョンの性質について

メモリリージョンが上記メモリマップに従って, 任意に定義でき, 全てのメモリは RAM であるため, 読み書き及び命令フェッチが許可し，それ以外特別な性質は持たない【NiosIOS107】〔IOS136〕.

3. 基本のデータ型

Nios2 では基本とするデータ型（基本のデータ型）は，以下に示すように typedef で定義する．

表 3-1　基本のデータ型一覧

基本のデータ型名	定義値
boolean	unsigned char
uint	unsigned int
sint	signed int
uint8	unsigned char
sint8	signed char
uint16	unsigned short
sint16	signed short
uint32	unsigned long
sint32	signed long
uint8_least	unsigned long
uint16_least	unsigned long
uint32_least	unsigned long
sint8_least	signed long
sint16_least	signed long
sint32_least	signed long
float32	float
float64	double

4. 機能セット

機能セットのうち，ターゲット定義となっている項目について規定する．

4.1　同時に実行可能状態，実行状態になることができるタスクの最大数

同時に実行可能状態，実行状態になることができるタスクの最大数は 4294967295 個である【NiosIOS001】〔IOS017〕．

4.2　使用できるリソース数と使用できる内部リソース数

使用できるリソース数と使用できる内部リソース数の最大数は，リソース数と内部リソース数の合計 4294967295 個である【NiosIOS002】〔IOS021〕〔IOS022〕．

4.3　使用できるアラーム数

使用できるアラーム数の最大数は 4294967295 個である【NiosIOS003】〔IOS023〕．

4.4　スケジュールテーブル数

使用できるスケジュールテーブル数の最大数は 4294967295 個である【NiosIOS004】〔IOS089〕．

4.5　ソフトウェアカウンタ数

使用できるソフトウェアカウンタ数の最大数はハードウェアカウンタとソフトウェアカウンタの合計 4294967295 個である【NiosIOS005】〔IOS090〕．

4.6　OSAP 数

使用できる OSAP 数の最大数は信頼 OSAP と非信頼 OSAP を合わせて 4294967295 個である【NiosIOS006】〔IOS091〕．但し，使用できる非信頼 OSAP の最大数は 32 個である【NiosIOS069】〔IOS091〕．

4.7　カウンタのティックの最大値

カウンタのティックとして設定できる最大値は 2147483647 である【NiosIOS060】〔IOS152〕．

4.8　カウンタの最小周期値の最大値

カウンタに設定できる最小周期値の最大値は 2147483647 である【NiosIOS061】〔IOS153〕．

4.9　アラーム自動起動時の初回満了値の最大値

アラーム自動起動時の初回満了時間に設定できる最大値は 2147483647 である【NiosIOS062】〔IOS154〕．

4.10 アラーム自動起動時の周期最大値

アラーム自動起動時の周期時間に設定できる最大値は2147483647である【NiosIOS063】〔IOS155〕.

4.11 カウンタ固有の最大値

カウンタ固有の値(OS は不使用)に設定できる最大値は 2147483647 である【NiosIOS064】〔IOS156〕.

4.12 スケジュールテーブルの周期最大値

スケジュールテーブル周期に設定できる最大値は 2147483647 である【NiosIOS065】〔IOS157〕.

4.13 スケジュールテーブルの自動起動時間の最大値

スケジュールテーブル自動起動時間に設定できる最大値は 2147483647 である【NiosIOS066】〔IOS158〕.

4.14 使用できるスピンロック数<<SC1-MC>><<SC3-MC>>

使用できるスピンロック数の最大数は 4294967295 個である.

但し，OS 割込み禁止スピンロックがネスト獲得される場合，OS 割込み禁止のネスト数を合わせて，合計最大数は 255 である【NiosIOS089】〔IOS202〕.

また，全割込み禁止スピンロックがネスト獲得される場合，全割込み禁止のネスト数を合わせて，合計最大数は 255 である【NiosIOS090】〔IOS203〕.

4.15 使用できる IOC 数<<SC1-MC>><<SC3-MC>>

使用できる IOC 数の最大数は 4294967296 個である.

4.16 使用できるコア間割込み数<<SC1-MC>><<SC3-MC>>

ディスパッチとシャットダウンのためにシステム定義のコア間割込みは，ユーザ定義のコア間割込みと別々で管理するため，ユーザ定義できるコア間割込み数の最大数は 31 個である.

5. スタック使用量

C2ISR の割込み入口処理と，CPU 例外入口処理で使用するスタック使用量について示す.

5.1　SC1

SC1 でのスタック使用量について示す.

5.1.1　C2ISR の割込み入口処理でのスタック使用量

C2ISR の割込みが発生したときのスタック使用量について示す.

5.1.1.1　タスク実行中の割込み

タスク実行中に割込みが発生した場合は，レジスタ退避のためにタスクスタックを 72 バイト使用する．その後，C2ISR 用スタックにスタックを切り替え，12 バイトを使用する【NiosIOS007】.

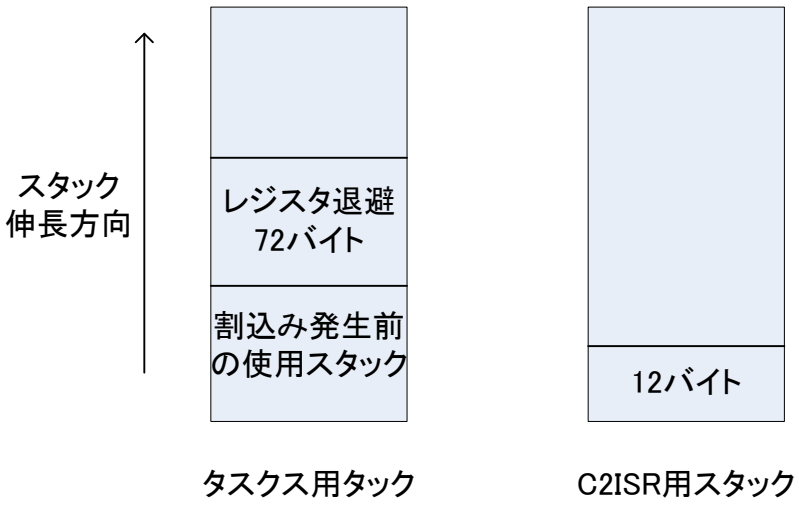

図 5-1 タスク実行中の割込みのスタック使用量(SC1)

5.1.1.2　多重割込み

C2ISR 実行中に割込みが発生した場合は，割込みスタックを 80 バイト使用する【NiosIOS008】.

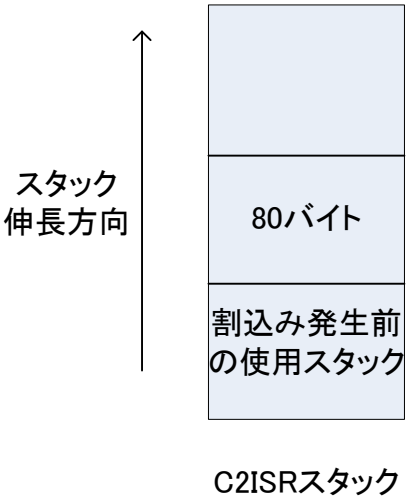

図 5-2 多重割込みのスタック使用量(SC1)

5.1.2　CPU 例外の入口処理でのスタック使用量

CPU 例外が発生したときのスタック使用量について示す.

5.1.2.1　タスク実行中の CPU 例外

タスク実行中に CPU 例外が発生した場合は, レジスタ退避のためにタスクスタックを 72 バイト使用する．その後，C2ISR 用スタックにスタックを切り替え，40 バイトを使用する【NiosIOS009】．ただし，40 バイトのうち，36 バイトはコンパイラ依存であり，他のコンパイラを使用するとスタック使用量が変わる可能性がある．また，プロテクションフック有効の場合，CPU 例外情報を退避するため，更に 16 バイトを使用するので，合わせて 56 バイトを使用する【NiosIOS073】.

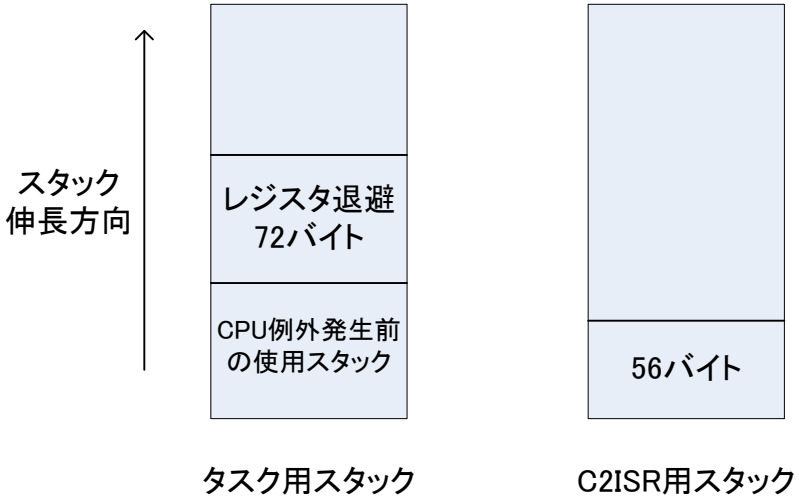

図 5-3 タスク実行中の CPU 例外のスタック使用量(SC1)

5.1.2.2　スタートアップフック，C2ISR，C1ISR 実行中の CPU 例外

スタートアップフック，C2ISR，C1ISR 実行中の CPU 例外は使用中のスタックを 128 バイト使用する【NiosIOS010】.

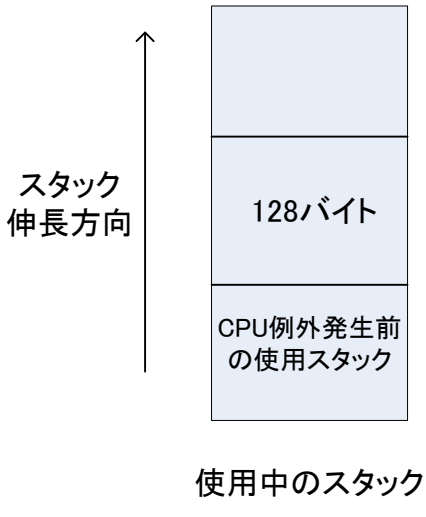

図 5-4 スタートアップフック，ISR 実行中の CPU 例外のスタック使用量(SC1)

5.2　SC1-MC

SC1-MC でのスタック使用量について示す.

5.2.1　C2ISR の割込み入口処理でのスタック使用量

C2ISR の割込みが発生したときのスタック使用量について示す.

5.2.1.1　タスク実行中の割込み

タスク実行中に割込みが発生した場合は，レジスタ退避のためにタスクスタックを 72 バイト使用する．その後，C2ISR 用スタックにスタックを切り替え，16 バイトを使用する【NiosIOS070】.

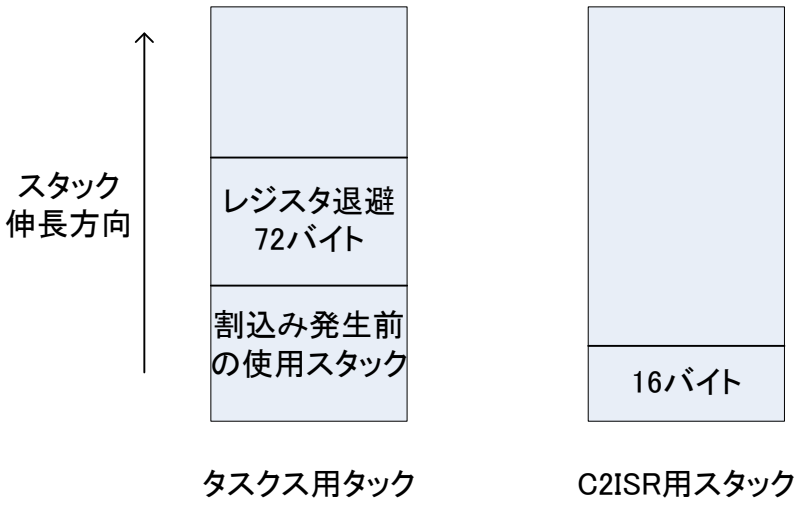

図 5-5 タスク実行中の割込みのスタック使用量(SC1-MC)

5.2.1.2　多重割込み

C2ISR 実行中に割込みが発生した場合は，割込みスタックを 84 バイト使用する【NiosIOS071】.

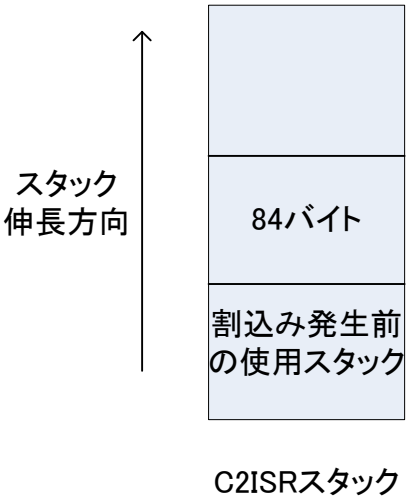

図 5-6 多重割込みのスタック使用量(SC1-MC)

5.2.2　CPU 例外の入口処理でのスタック使用量

CPU 例外が発生したときのスタック使用量について示す.

5.2.2.1　タスク実行中の CPU 例外

タスク実行中に CPU 例外が発生した場合は, レジスタ退避のためにタスクスタックを 72 バイト使用する．その後，C2ISR 用スタックにスタックを切り替え，32 バイトを使用する【NiosIOS072】．ただし，32 バイトのうち，28 バイトはコンパイラ依存であり，他のコンパイラを使用するとスタック使用量が変わる可能性がある．また，プロテクションフック有効の場合，CPU 例外情報を退避するため，

更に 20 バイトを使用するので，合わせて 52 バイトを使用する【NiosIOS074】.

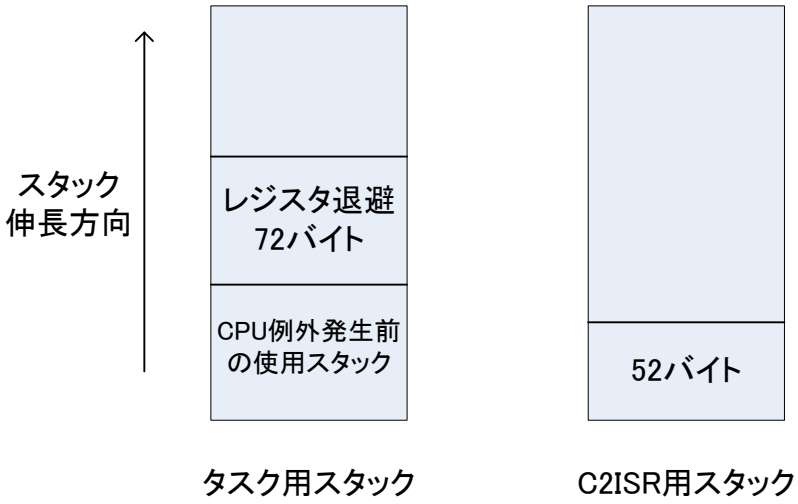

図 5-7 タスク実行中の CPU 例外のスタック使用量(SC1-MC)

5.2.2.2　スタートアップフック，C2ISR，C1ISR 実行中の CPU 例外

スタートアップフック，C2ISR，C1ISR 実行中の CPU 例外は使用中のスタックを 124 バイト使用する【NiosIOS075】.

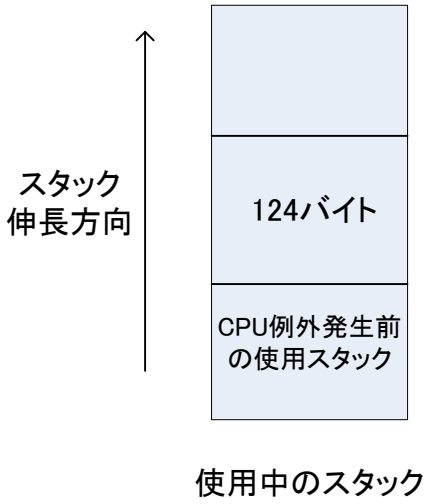

図 5-8 スタートアップフック，ISR 実行中の CPU 例外のスタック使用量(SC1-MC)

5.3　SC3

SC3 でのスタック使用量について示す.

5.3.1　C2ISR の割込み入口処理でのスタック使用量

C2ISR の割込みが発生したときのスタック使用量について示す.

5.3.1.1　タスク実行中の割込み

非信頼タスク実行中に割込みが発生した場合は，レジスタ退避のために，非信頼タスク用システムスタック（信頼タスク実行中の場合は，信頼タスク用スタック）を 76 バイト使用する．非信頼タスク用スタックを使用している場合は，非信頼タスク用システムスタックに切り替えてから 76 バイトを使用する．

その後，C2ISR 用スタックにスタックを切り替え，20 バイトを使用する【NiosIOS011】.

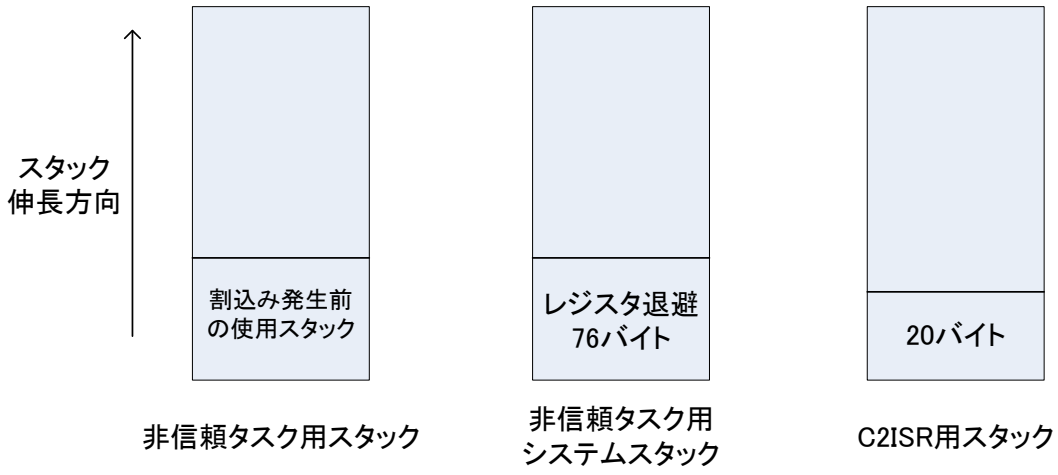

図 5-9 タスク実行中(非特権モード)の割込みのスタック使用量(SC3)

5.3.1.2　多重割込み

C2ISR 実行中に割込みが発生した場合は，割込みスタックを 92 バイト使用する【NiosIOS012】.

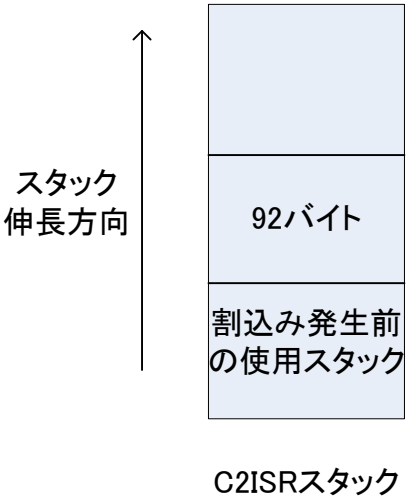

図 5-10 多重割込みのスタック使用量(SC3)

5.3.2　CPU 例外の入口処理でのスタック使用量

CPU 例外が発生した場合は，124 バイトを使用する．ただし，124 バイトのうち，48 バイトはコンパイラ依存であり，他のコンパイラを使用するとスタック使用量が変わる可能性がある．スタックの切り替えを行う場合は，切り替えた後に 124 バイトを使用し，切り替えない場合は，使用中のスタックの 124 バイトを使用する【NiosIOS013】．また，プロテクションフック有効の場合，CPU 例外情報を退避するため，更に 16 バイトを使用するので，合わせて 140 バイトを使用する【NiosIOS076】.

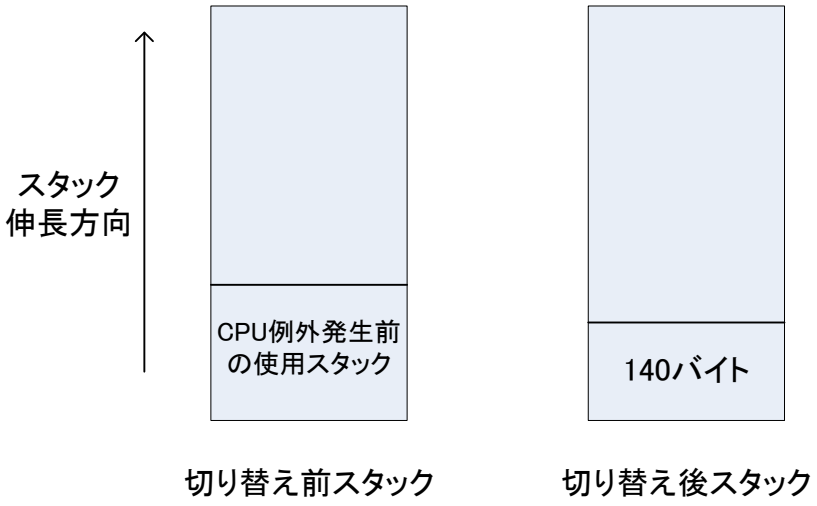

図 5-11 CPU 例外のスタック使用量(SC3，スタック切り替えあり)

5.4　SC3-MC

SC3-MC でのスタック使用量について示す.

5.4.1　C2ISR の割込み入口処理でのスタック使用量

C2ISR の割込みが発生したときのスタック使用量について示す.

5.4.1.1　タスク実行中の割込み

非信頼タスク実行中に割込みが発生した場合は，レジスタ退避のために，非信頼タスク用システムスタック（信頼タスク実行中の場合は，信頼タスク用スタック）を 76 バイト使用する．非信頼タスク用スタックを使用している場合は，非信頼タスク用システムスタックに切り替えてから 76 バイトを使用する.

その後，C2ISR 用スタックにスタックを切り替え，20 バイトを使用する【NiosIOS011】.

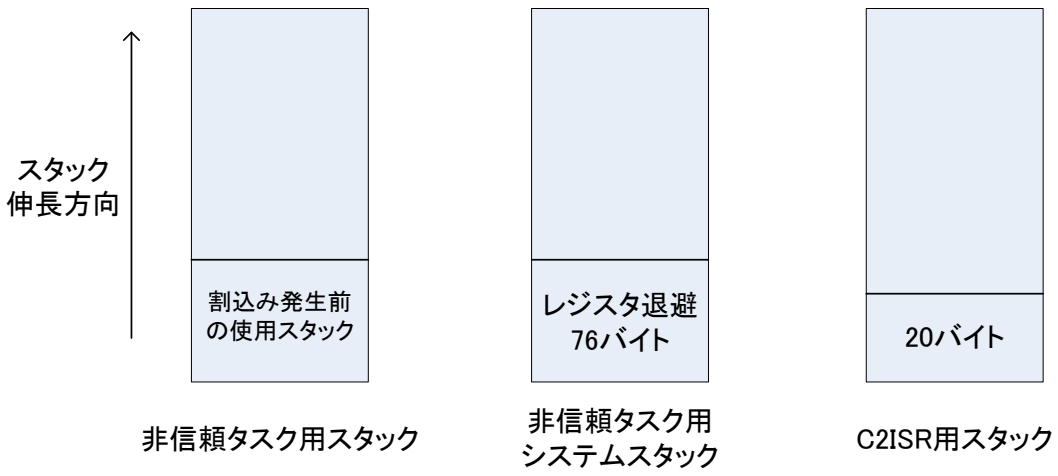

図 5-12 タスク実行中(非特権モード)の割込みのスタック使用量(SC3-MC)

5.4.1.2　多重割込み

C2ISR 実行中に割込みが発生した場合は，割込みスタックを 92 バイト使用する【NiosIOS012】.

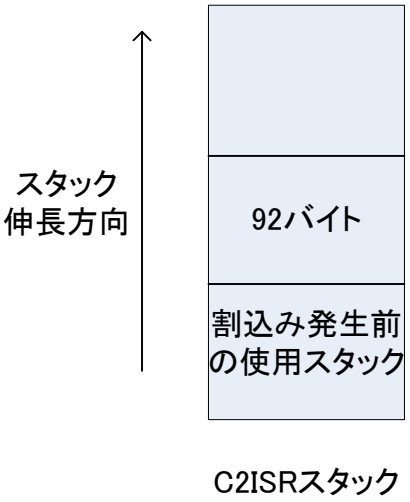

図 5-13 多重割込みのスタック使用量(SC3-MC)

5.4.2　CPU 例外の入口処理でのスタック使用量

CPU 例外が発生した場合は，112 バイトを使用する．ただし，112 バイトのうち，36 バイトはコンパイラ依存であり，他のコンパイラを使用するとスタック使用量が変わる可能性がある．スタックの切り替えを行う場合は，切り替えた後に 112 バイトを使用し，切り替えない場合は，使用中のスタックの 112 バイトを使用する【NiosIOS013】．また，プロテクションフック有効の場合，CPU 例外情報を退避するため，更に 20 バイトを使用するので，合わせて 132 バイトを使用する【NiosIOS076】.

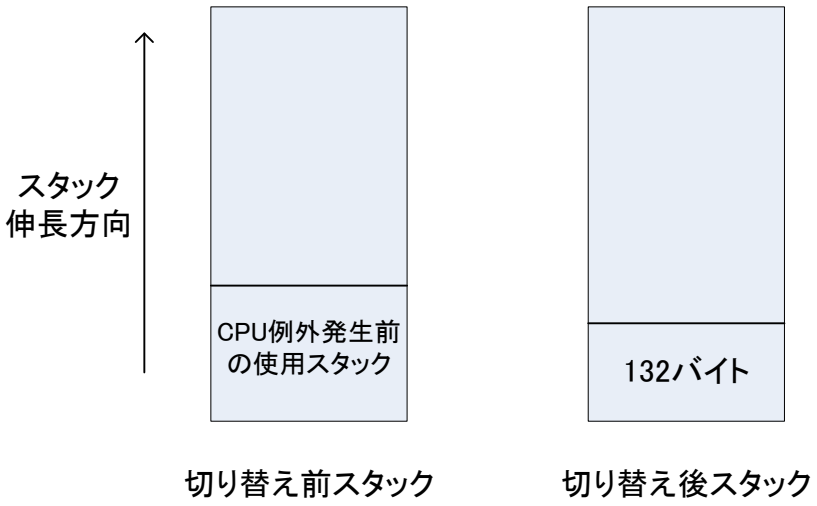

図 5-14 CPU 例外のスタック使用量(SC3-MC, スタック切り替えあり)

6. 割込み

割込みに関する Nios2 依存の仕様を規定する.

6.1　割込み番号

/AUTOSAR/EcucDefs/Os/OsIsr/OsIsrInterruptNumber で定義する割込み番号は 0～31 が使用可能である【NiosIOS014】〔IOS146〕.

また，マルチコアの場合，コア毎に/AUTOSAR/EcucDefs/Os/OsIsr/OsIsrInterruptNumber で定義する割込み番号は以下の様に使用可能である【NiosIOS094】〔IOS146〕.

- コア 0：0x10000，0x10001，……，0x1001f
- コア 1：0x20000，0x20001，……，0x2001f

※コア毎の有効な割込み番号を求める式：(((コア ID + 1) << 16) & (0～31 の何れか))

6.2　割込み優先度

/AUTOSAR/EcucDefs/Os/OsIsr/OsIsrInterruptPriority で定義する割込み優先度は 1～64 が使用可能である【NiosIOS015】〔IOS147〕.

6.3　C1ISR の本体記述

C1ISR は C 言語とアセンブリ言語のどちらでも定義可能である【NiosIOS096】〔IOS032〕. C 言語で実装する場合でも出入り口処理をアセンブリ言語で実装する必要がある. C1ISR の出入り口処理で行うべき処理を以下に示す.

- ea に ea – 4 を代入
- 使用する caller saved レジスタの退避
- ra, ea, estatus レジスタの退避
- 割込み禁止の解除
- callevel_stat をスタックに退避
- callevel_stat に 0x800 を論理和した値を callevel_stat に代入する
- 必要に応じてスタックを切り替える
- C1ISR の処理を実行する
- スタックを元に戻す
- callevel_stat をスタックから復帰する
- 割込みを禁止する
- レジスタを復帰する
- eret で割込み元にリターンする

これらの処理をアセンブリ言語で記述すると以下のようになる．なお，以下の例ではシングルコアを

想定し，スタックの切り替えは行なっていない．マルチコアの場合は，callevel_stat の管理方法に応じて，アセンブリコードを変更する必要がある．

```
C1ISRMainISRName:
        /* 戻り番地をデクリメント */
        addi    ea,     ea,     -4
        /* レジスタの保存 */
        addi    sp,     sp,     -76
        stw             at,     0(sp)
        stw             r2,     4(sp)
        stw             r3,     8(sp)
        stw             r4,     12(sp)
        stw             r5,     16(sp)
        stw             r6,     20(sp)
        stw             r7,     24(sp)
        stw             r8,     28(sp)
        stw             r9,     32(sp)
        stw             r10,    36(sp)
        stw             r11,    40(sp)
        stw             r12,    44(sp)
        stw             r13,    48(sp)
        stw             r14,    52(sp)
        stw             r15,    56(sp)
        stw             ra,     60(sp)
        stw             ea,     64(sp)
        rdctl           et,     estatus
        stw             et,     68(sp)
        /* 割込み禁止を解除 */
        rdctl           r2,     status
        ori             r2,     r2,     1
        wrctl           status, r2

        /* 割込み発生前の callevel_stat をスタックに保存 */
        ldh             r2,     %gprel(callevel_stat)(gp)
        stw             r2,     72(sp)
```

```
/* callevel_stat に 0x800 をセット */
ldhu        et,     %gprel(callevel_stat)(gp)
ori         et,     et,     0x800
sth         et,     %gprel(callevel_stat)(gp)

call    c1isr_func

/* callevel_stat を復帰 */
ldw         r2      72(sp)
sth         r2,     %gprel(callevel_stat)(gp)

/* 割込み禁止 */
rdctl       r2,     status
movi        r3,     ~1
and         r2,     r2,     r3
wrctl       status, r2

/* レジスタの復帰 */
ldw         at,     0(sp)
ldw         r2,     4(sp)
ldw         r3,     8(sp)
ldw         r4,     12(sp)
ldw         r5,     16(sp)
ldw         r6,     20(sp)
ldw         r7,     24(sp)
ldw         r8,     28(sp)
ldw         r9,     32(sp)
ldw         r10,    36(sp)
ldw         r11,    40(sp)
ldw         r12,    44(sp)
ldw         r13,    48(sp)
ldw         r14,    52(sp)
ldw         r15,    56(sp)
```

```
        ldw             ra,     60(sp)
        ldw             ea,     64(sp)
        ldw             et,     68(sp)
        wrctl           estatus, et
        addi            sp,     sp,     76

        eret
```

6.4　コア間割込み<<SC1-MC>><<SC3-MC>>

マルチコアの場合は, コアが協調動作するため, コア間通信にコア間割込みを使用する【NiosIOS081】. ディスパッチャ起動要求・OS シャットダウン要求・ユーザ定義のコア間割込み要求を他のコアに通知するため，同じ間割込みを使用する【NiosIOS082】. 各コアに使用するコア間割込みの，割込み番号と割込み優先度を，コア毎に定義し，ユーザによる変更は認めない【NiosIOS083】.

Nios2 では，コア毎のコア間割込み番号は，コア数分の配列で 1 つずつ定義する【NiosIOS084】. 各コアのコア間割込みの優先度も同様に，コア数分の配列で 1 つずつ定義する【NiosIOS085】.

Nios2 で定義しているコア間割込み番号：

- コア 0：0x10002
- コア 1：0x20002

Nios2 で定義しているコア間割込み番号の優先度：

- コア 0：INTPRI_ICI0（2）
- コア 1：INTPRI_ICI1（2）

7. CPU 例外発生時の情報取得

プロテクションフック内で取得できる CPU 例外の情報について示す【NiosIOS097】〔IOS131〕.

7.1　CPU 例外番号<<SC1>><<SC3>>

変数名	nios2_cpu_exp_no【NiosIOS050】
データ型	uint32
概要	CPU 例外番号が格納される.

7.2　CPU 例外番号<<SC1-MC>><<SC3-MC>>

変数名	nios2_cpu_exp_no[TotalNumberOfCores]【NiosIOS077】
データ型	uint32
概要	コア毎に CPU 例外番号が格納される.

7.3　CPU 例外発生時の pc<<SC1>><<SC3>>

変数名	nios2_cpu_exp_pc【NiosIOS051】
データ型	uint32
概要	CPU 例外が発生した時の pc レジスタの値が格納される.

7.4　CPU 例外発生時の pc<<SC1-MC>><<SC3-MC>>

変数名	nios2_cpu_exp_pc[TotalNumberOfCores]【NiosIOS078】
データ型	uint32
概要	コア毎に CPU 例外が発生した時の pc レジスタの値が格納される.

7.5　CPU 例外発生時の badaddr<<SC1>><<SC3>>

変数名	nios2_cpu_exp_badaddr【NiosIOS052】
データ型	uint32
概要	CPU 例外が発生した時の badaddr レジスタの値が格納される.

7.6　CPU 例外発生時の badaddr<<SC1-MC>><<SC3-MC>>

変数名	nios2_cpu_exp_badaddr[TotalNumberOfCores]【NiosIOS079】
データ型	uint32
概要	コア毎に CPU 例外が発生した時の badaddr レジスタの値が格納される.

7.7　CPU 例外発生時の sp<<SC1>><<SC3>>

変数名	nios2_cpu_exp_sp【NiosIOS053】
データ型	uint32
概要	CPU 例外が発生しレジスタを退避した後の sp レジスタの値が格納される.

7.8　CPU 例外発生時の sp<<SC1-MC>><<SC3-MC>>

変数名	nios2_cpu_exp_sp[TotalNumberOfCores]【NiosIOS080】
データ型	uint32
概要	コア毎に CPU 例外が発生しレジスタを退避した後の sp レジスタの値が格納される.

7.9　CPU 例外発生時のレジスタ情報

CPU 例外発生時に各レジスタをスタックに保存する．nios2_cpu_exp_sp(マルチコアの場合は, nios2_cpu_exp_sp[coreid])を使って保存してあるレジスタの値を取得することができる．保存してあるレジスタとアドレスの対応を以下に規定する．【NiosIOS081】

アドレス	レジスタ
nios2_cpu_exp_sp	r1(at)
nios2_cpu_exp_sp + 4	r2
nios2_cpu_exp_sp + 8	r3
nios2_cpu_exp_sp + 12	r4
nios2_cpu_exp_sp + 16	r5
nios2_cpu_exp_sp + 20	r6
nios2_cpu_exp_sp + 24	r7
nios2_cpu_exp_sp + 28	r8
nios2_cpu_exp_sp + 32	r9
nios2_cpu_exp_sp + 36	r10
nios2_cpu_exp_sp + 40	r11
nios2_cpu_exp_sp + 44	r12
nios2_cpu_exp_sp + 48	r13
nios2_cpu_exp_sp + 52	r14
nios2_cpu_exp_sp + 56	r15
nios2_cpu_exp_sp + 60	ra
nios2_cpu_exp_sp + 64	ea
nios2_cpu_exp_sp + 68	estatus
nios2_cpu_exp_sp + 72	sp

8. アライメント制約

ターゲット依存のアライメント制約について示す.

8.1　タスクスタックサイズ

SC1, SC2 のタスク／SC3, SC4 の信頼タスクの場合，OsTaskStackStartAddress を指定した場合の OsTaskStackSize で指定するスタックサイズは, 4 の倍数でなければならない【NiosIOS016】〔IOS074〕.

SC3, SC4 で非信頼タスクの場合，OsTaskStackStartAddress を指定した場合の OsTaskStackSize で指定するスタックサイズは，64 の倍数でなければならない【NiosIOS017】〔IOS074〕.

8.2　タスクスタックの先頭番地

SC1, SC2 のタスク／SC3, SC4 で信頼タスクの場合，OsTaskStackStartAddress で指定するスタックの先頭番地は，4 バイトアライメントでなければならない【NiosIOS018】〔IOS072〕.

SC3, SC4 で非信頼タスクの場合，OsTaskStackStartAddress で指定するスタックの先頭番地は，64 バイトアライメントでなければならない【NiosIOS019】〔IOS072〕.

8.3　非信頼タスク用システムスタックサイズ

SC3, SC4 の非信頼タスクの場合，OsTaskSystemStackStartAddress を指定した場合の OsTaskSystemStackSize で指定するスタックサイズは，4 の倍数でなければならない【NiosIOS020】〔IOS075〕.

8.4　非信頼タスク用システムスタックの先頭番地

SC3, SC4 で非信頼タスクの場合，OsTaskSystemStackStartAddress で指定するスタックの先頭番地は，4 バイトアライメントでなければならない【NiosIOS021】〔IOS073〕.

8.5　非信頼フック用スタックサイズ

OsNonTrustedHookStackStartAddress を指定した場合の OsNonTrustedHookStackSize で指定するスタックサイズは，64 の倍数でなければならない【NiosIOS022】〔IOS076〕.

8.6　非信頼フック用スタックの先頭番地

OsNonTrustedHookStackStartAddress で指定するスタックの先頭番地は，64 バイトアライメントでなければならない【NiosIOS023】〔IOS077〕.

8.7　C2ISR 用スタックサイズ

OsIsrStackStartAddress を指定した場合の OsIsrStackSize で指定するスタックサイズは，4 の倍数でなければならない【NiosIOS024】.

8.8　C2ISR 用スタックの先頭番地

OsIsrStackStartAddress で指定するスタックの先頭番地は, 4 バイトアライメントでなければならない【NiosIOS025】〔IOS211〕.

8.9　非信頼 C2ISR 用システムスタックの先頭番地

OsIsrSystemStartAddress で指定するスタックの先頭番地は，64 バイトアライメントでなければならない【NiosIOS105】〔IOS212〕.

8.10　ICISR 用スタックの先頭番地

OsInterCoreInterruptStackStartAddress で指定するスタックの先頭番地は，4 バイトアライメントでなければならない【NiosIOS106】〔IOS214〕.

8.11　非信頼 ICISR 用システムスタックの先頭番地

OsInterCoreInterruptSystemStackStartAddress で指定するスタックの先頭番地は，64 バイトアライメントでなければならない【NiosIOS107】〔IOS218〕.

8.12　メモリ領域のサイズ

OsMemoryAreaSize で指定するメモリ領域のサイズは, 64 の倍数でなければならない【NiosIOS026】〔IOS141〕.

8.13　メモリ領域の先頭番地

OsMemoryAreaStartAddress で指定するメモリ領域の先頭番地は，64 バイトアライメントでなければならない【NiosIOS027】〔IOS141〕.

8.14　メモリリージョンの先頭番地

OsMemoryRegionStartAddress で指定するメモリリージョンの先頭番地のアライメント制約はない【NiosIOS028】〔IOS141〕.

9. スタックの定義方法

スタック領域をユーザが確保する方法について以下に規定する【NiosIOS095】〔IOS133〕.
マルチコアにおいても，スタック領域をユーザが確保する方法は，この規定に従う.

9.1　SC1, SC2 のスタック

SC1, SC2 では以下のようにスタックを確保する．データ型は StackType を使う．確保したいサイズを COUNT_STK_T の引数にして配列のサイズとして指定する.

```
StackType stack[COUNT_STK_T(0x100)];
```

9.2　SC3, SC4 の信頼タスク用スタック

SC3, SC4 の信頼タスク用スタック，非信頼タスク用システムスタック，C2ISR 用スタックを確保するには以下のように行う．データ型は StackType を使う．確保したいサイズを COUNT_STK_T の引数にして配列のサイズとして指定する.

```
DEFINE_VAR_SSTACK(StackType, stack[COUNT_STK_T(0x100)]);
```

スタックの extern 宣言は以下のように行う．サイズを指定してもよい.

```
extern StackType stack[];
```

9.3　SC3, SC4 の非信頼タスク用スタック

SC3, SC4 の非信頼タスク用スタックを確保するには以下のように行う．データ型は StackType を使う．確保したいサイズを COUNT_STK_T の引数にして配列のサイズとして指定する．第 3 引数はスタックを配置するセクションであり，このセクションを OsLinkSection で標準 RAM に配置する必要がある.

```
DEFINE_VAR_USTACK(StackType, stack[COUNT_STK_T(0x100)], ".stack_section");
```

スタックの extern 宣言は以下のように行う．サイズを指定してもよい.

```
extern StackType stack[];
```

9.4　SC3, SC4 の非信頼フック用スタック

SC3, SC4 の非信頼フック用スタックを確保するには以下のように行う．データ型は StackType を使う．確保したいサイズを COUNT_STK_T の引数にして配列のサイズとして指定する.

DEFINE_VAR_HSTACK(StackType, stack[COUNT_STK_T(0x100)]);

スタックの extern 宣言は以下のように行う．サイズを指定してもよい．

extern StackType stack[];

10. メモリ保護

10.1 非信頼 OSAP の動作モード

非信頼 OSAP は非特権モードで動作する【NiosIOS029】〔NOS0237〕.

10.2 標準のセクション

Nios2 では以下の標準のセクションが存在し, メモリ保護属性も以下とする【NiosIOS101】〔IOS178〕.

10.2.1 text セクション

text セクションは関数が入るセクションである．text セクションは読込み，実行が可能であり，標準 ROM に配置される.

10.2.2 rodata セクション

rodata セクションは定数が入るセクションである．rodata セクションは読込みが可能であり，標準 ROM に配置される.

rodata セクションには，以下のように 9 バイト以上の const が付いた変数が入る．8 バイト以下の場合は sdata セクションになる.

```
const char const_var[9] = {1,2,3,4,5,6,7,8,9};
```

10.2.3 data セクション

data セクションは初期値ありデータが入るセクションである．data セクションは読込み，書込みが可能であり，標準 RAM に配置される.

data セクションには，以下のように 9 バイト以上の初期値ありの変数が入る．8 バイト以下の場合は sdata セクションになる.

```
char const_var[9] = {1,2,3,4,5,6,7,8,9};
```

10.2.4 bss セクション

bss セクションはゼロ初期化されるデータが入るセクションである．bss セクションは読込み，書込みが可能であり，標準 RAM に配置される.

bss セクションには, 以下のように 9 バイト以上の初期値がない変数か 0 に初期化された変数が入る．8 バイト以下の場合は sbss セクションになる.

```
char const_var[9];
```

10.2.5 prsv セクション

prsv セクションはコンパイラが生成するセクションではないが，便宜上，標準のセクションにしてい

る．prsv セクションは読込み，書込みが可能であり，標準 RAM に配置される．prsv セクションは初期化が行われないセクションであるため，スタック領域等に使う．prsv セクションに変数を配置するには以下のように行う．

DEFINE_VAR_SEC(uint32, prsv_var[0x100], ".prsv");

10.2.6 sdata セクション

sdata セクションは初期値ありのショートデータが入るセクションである．sdata セクションは読込み，書込みが可能であり，標準 RAM に配置される．

sdata セクションには，以下のように 8 バイト以下の初期値ありの変数が入る．

char const_var[8] = {1,2,3,4,5,6,7,8};

10.2.7 sbss セクション

sbss セクションはゼロ初期化されるショートデータが入るセクションである．sbss セクションは読込み，書込みが可能であり，標準 RAM に配置される．

sbss セクションには，以下のように 8 バイト以下の初期値がない変数か 0 に初期化された変数が入る．

char const_var[8];

10.2.8　rodata.str1.4 セクション

rodata.str1.4 セクションは文字列が入るセクションである．rodata.str1.4 セクションは読込みが可能であり，標準 ROM に配置される．

10.3 OS が提供するセクション

OS は以下のセクション名を提供する．

<セクション種別><標準のセクション名>

<セクション種別>

kernel：カーネル及び信頼 OSAP しかアクセスできないセクション

非信頼 OSAP 名：カーネル，信頼 OSAP 及び非信頼 OSAP 名で指定した OSAP しかアクセスできないセクション

shared：共有セクション

<標準のセクション名>

text：10.2.1 text セクションを参照する

rodata：10.2.2 rodata セクションを参照する

data：10.2.3 data セクションを参照する
bss：10.2.4 bss セクションを参照する
prsv：10.2.5 prsv セクションを参照する
sdata：10.2.6 sdata セクションを参照する
sbss：10.2.7 sbss セクションを参照する

10.4 メモリ保護領域のアクセス権

10.4.1 タスクと C2ISR のスタック領域の保護

タスク，C2ISR のスタック領域に対して，実行アクセスすることを禁止する【NiosIOS030】〔NOS0213〕〔IOS201〕.

10.4.2 OSAP の専有リードオンリーデータ領域の保護

非信頼 OSAP の専有リードオンリーデータ領域に対して，実行アクセスすることを禁止する【NiosIOS031】〔IOS199〕〔NOS0307〕.

10.4.3 OSAP の専有リードライトデータ領域の保護

非信頼 OSAP の専有リードライトデータ領域に対して，実行アクセスすることを禁止する【NiosIOS032】〔NOS0214〕.

10.4.4 OSAP の共有リード専有ライトデータ領域の保護

非信頼 OSAP の共有リード専有ライトデータ領域に対して，実行アクセスすることを禁止する【NiosIOS033】〔NOS0216〕.

10.4.5 共有リードオンリーデータ領域の保護

共有リードオンリーデータ領域に対して，実行アクセスすることを禁止する【NiosIOS034】〔IOS200〕〔NOS0309〕.

10.4.6 共有リードライトデータ領域の保護

共有リードライトデータ領域に対して，実行アクセスすることを禁止する【NiosIOS035】〔NOS0215〕.

10.4.7 OSAP の専有コード領域の保護

非信頼 OSAP の専有コード領域に対して，非信頼 OSAP 自身が読出しアクセスすることを許可する【NiosIOS036】〔IOS099〕〔NOS0265〕.

10.4.8 共有コード領域の保護

共有コード領域に対して，すべての非信頼 OSAP が，読出しアクセスすることを許可する【NiosIOS037】〔IOS100〕〔NOS0267〕.

10.5 配置されていない領域の保護

Nios2 ではバックグラウンド領域を全メモリ領域（0～0xffffffff）設定することにより，周辺デバイス等が配置さていない領域は，信頼 OSAP が読み出し，書込み，実行アクセスすることができる【NiosIOS067】〔NOS0265〕〔IOS159〕.

10.6 メモリ保護領域数の上限

Nios2 の MPU のメモリ保護領域は，（マルチコアの場合は，コア毎に）データ領域 16 個，命令領域 16 個である．OsMemorySection, OsMemoryArea コンテナによりメモリ保護領域を追加した場合に，ある OSAP に対して以下の制限を超える場合はエラーとなる【NiosIOS038】〔IOS160〕〔IOS161〕.

1 つの OSAP の専有データ領域数 ＋ 共有データ領域数 ＞13 領域

1 つの OSAP の専有コード領域数 ＋ 共有コード領域数 ＞15 領域

10.7 メモリ保護違反となる CPU 例外番号

CPU 例外が発生した際に, プロテクションフックに引数として渡される保護違反の種類として, CPU 例外番号の 16 と 17 がメモリ保護違反(E_OS_PROTECTION_MEMORY)となり，それ以外は CPU 例外(E_OS_PROTECTION_EXCEPTION)とする【NiosIOS059】.

11. コンテナに対する仕様

11.1 OsTaskStackSize の最小値

SC1, SC2 の場合，OsTaskStackSize に 196 より小さい値を指定した場合，ジェネレータはエラーを検出する【NiosIOS039】〔IOS140〕.

SC3, SC4 における信頼タスクの場合，OsTaskStackSize と OsTaskSystemStackSize に指定されたサイズとの合計値が 204 より小さい場合, ジェネレータはエラーを検出する【NiosIOS040】〔NOS0820〕

11.2 OsTaskSystemStackSize の最小値

SC3, SC4 における非信頼タスクの場合で，OsTaskSystemStackSize が 204 より小さい場合，ジェネレータはエラーを検出する【NiosIOS041】〔IOS140〕.

補足

それぞれのスタックの最小値はシステムサービスでどれだけスタックを消費するかを測定して，決定した．IncrementCounter が最もスタックを消費するシステムサービスであるが，IncrementCounter を発行し，満了処理も IncrementCounter だった場合に，スタックの消費量の上限を決めることができない．満了処理で行われる IncrementCounter は最大 1 回までとし，スタック量を見積もった．そのため，満了処理で IncrementCounter が 2 回以上呼ばれる場合は，より多くのスタックを消費する．満了処理で実行される IncrementCounter が 1 回の場合で，最もスタックを消費するケースは，次の通りである．IncrementCounter を発行し，満了処理で IncrementCounter を実行する．さらに満了処理が発生し，その間に割込みが発生すると，レジスタをスタックに退避する．この場合が最もスタック消費量が大きい.

SC1, SC2 と SC3, SC4 での信頼タスクの場合は，システムサービスとタスクの実行時のスタックが同じであるため，タスク実行時のスタック消費量も考慮する必要がある.

11.3 OsOsStack のサイズ

OsOsStack コンテナが省略された場合，C2ISR に対して OsIsrStackSize で指定された値の，割込み優先度毎の最大値を合計した値と，OsHookStackSize で指定された値(省略された場合は 1024)の合計値に，256 を足した値（※）のスタックを確保する【NiosIOS042】〔IOS052〕.

本コンテナに，※の合計値より，小さい値を指定した場合，ジェネレータはエラーを検出するため，依存部で，DEFAULT_OSSTKSZ マクロを【NiosIOS102】〔IOS054〕.

11.3.1 OsOsStackStartAddress のチェック

OsOsStackStartAddress で指定した値が，ターゲット定義のアライメント制約を満たしていない場合，ジェネレータはエラーを検出する【NiosIOS103】〔【IOS071〕.

11.3.2 OsOsStackSize のチェック

OsOsStackStartAddress を指定した場合の OsOsStackSize で指定するスタックサイズは，4 の倍数

でなければならない【NiosIOS104】〔IOS070〕.

11.4 OsMemoryRegion のエラーチェック

Nios2 のメモリ空間は 32 ビットであるため，OsMemoryRegionAddress と OsMemoryRegionSize を足した値が 0xffffffff を超える場合，ジェネレータはエラーを検出する【NiosIOS044】.

11.5 OsMemorySectionShort の使用制限

OsMemorySection の OsMemorySectionShort パラメータは対象セクションが標準 RAM に配置される場合のみ true にしてよい．それ以外の場合で true にした場合，ジェネレータはエラーを検出する【NiosIOS045】.

11.6 OsIsrInterruptTrigger の使用可否

OsIsr の OsIsrInterruptTrigger パラメータは使用できない【NiosIOS054】〔IOS148〕.

11.7 OsMemorySectionCacheable

OsMemorySection の OsMemorySectionCacheable パラメータは無視する【NiosIOS055】.

11.8 OsMemorySectionDevice

OsMemorySection の OsMemorySectionDevice パラメータは無視する【NiosIOS056】.

11.9 OsMemoryAreaCacheable

OsMemoryArea の OsMemoryAreaCacheable パラメータは無視する【NiosIOS057】.

11.10 OsMemoryAreaDevice

OsMemoryArea の OsMemoryAreaDevice パラメータは無視する【NiosIOS058】.

11.11 OsMemoryModule

SC3, SC4 において, OsMemoryModule によるオブジェクトファイル単位のメモリセクション情報指定をサポートする【NiosIOS068】〔IOS180〕. オブジェクトファイル内の標準のセクションは標準 ROM と標準 RAM に配置される.

12. その他のターゲット定義仕様

12.1 マジックナンバーの値

スタックモニタリングのマジックナンバー方式で使うマジックナンバーの値は 0x4E434553 である【NiosIOS046】〔IOS130〕.

12.2 バージョンチェック

OS 起動時に Nios2 のターゲットのバージョンチェックを行い，想定しているバージョンと一致しているか確認する．想定しているバージョンと異なっている場合は，エラーメッセージを出力して無限ループにする【NiosIOS047】.

12.3 trap 命令の使用禁止

Nios2 の trap 命令は OS 内部で使用するため，ユーザが使用してはならない【NiosIOS049】.

OS 機能として，ユーザが不正な機能コードを指定して，システムサービス発行のための特権呼び出しを行った場合，E_OS_SERVICEID を返すように，ターゲット依存部で実装する【NiosIOS060】．エラーフックが有効の場合，エラーコードに E_OS_SERVICEID を，OSErrorGetServiceId にユーザ指定したシステムサービス ID を設定して，エラーフックを呼び出す【NiosIOS061】.

12.4 cpuid レジスタ

各コアの ID を提供するコア ID レジスタには，各コアのペリフェラルのベースアドレスを保持する【NiosIOS085】． ソフトウェアで，コア ID 仕様に従って変換して使用する【NiosIOS086】.

12.5 ターゲット依存の終了処理

OS サービスを終了するため，OS 管理する割込み（C2ISR）の禁止処理を依存部で実装する【NiosIOS098】〔IOS129〕〔IOS209〕．マルチコアにおいて， OS サービスを終了するため，コア停止判定用レジスタのクリア処理を依存部で実装する【NiosIOS099】〔IOS129〕〔IOS209〕． SC3 において，OS サービスを終了するため，MPU 機能を無効にする処理をターゲット依存部で実装する【NiosIOS100】〔IOS129〕〔IOS209〕.

変更履歴

Version	Date	Detail	Editor
0.0.1	2012/3/30	ATK2 コンソーシアム向けリリース	NCES
0.0.2	2012/06/01	メモリ保護関連情報の追記	NCES
1.0.0	2013/01/22	・誤字，説明不足等を修正 ・一般向けリリース	NCES
1.1.0	2013/06/28	・マルチコア拡張 ・誤字，説明不足等を修正	NCES
1.1.1	2013/07/10	・誤字を修正 ・文法的な文言を修正	NCES
1.2.0	2014/03/12	・誤字を修正 ・メモリ保護関係の情報の一部追記	NCES