RI600/PX メモリ保護機能の概説

アプリケーションノート

R20AN0166JJ0100
Rev.1.00
3/14/2012

本アプリケーションノートは、リアルタイム OS RI600/PX のメモリ保護機能について解説します。

目次

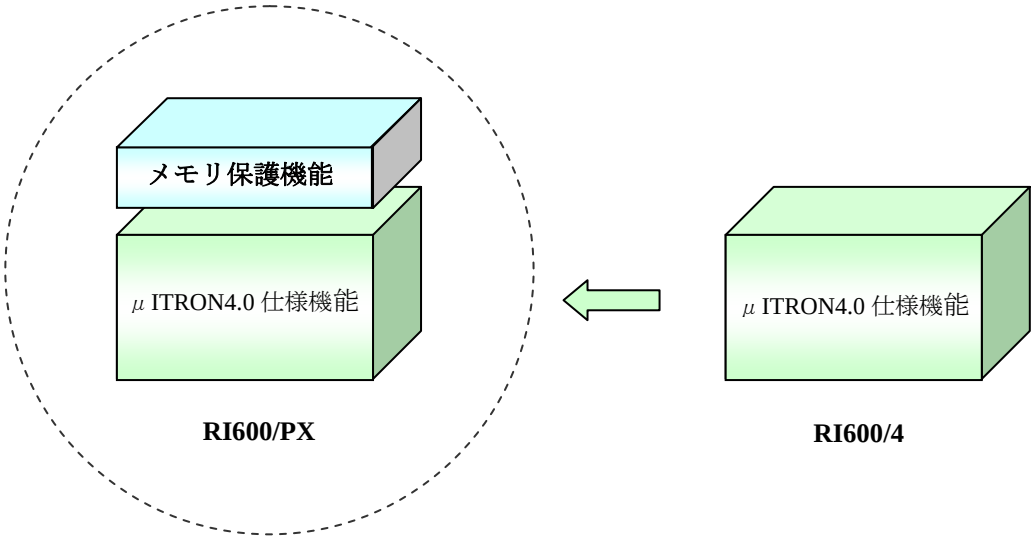
1. 概要

本アプリケーションノートは、RI600/PX が持つメモリ保護機能についての特長や機能をよりご理解いただき、ユーザのアプリケーションにご使用いただくための説明を記載しています。

1.1 機能

RI600/PX は、μITRON4.0 仕様に基づいて開発されたリアルタイム OS の RI600/4 をベースとし、μITRON4.0 仕様保護機能拡張であるメモリ保護機能を追加した製品です。

このため、RI600/4 やμITRON 仕様に準拠した他のリアルタイム OS を用いて開発したアプリケーションプログラムを、容易に RI600/PX に移植することが可能です。



図－1　RI600/PXとRI600/4

1.1.1 MCU

メモリ保護機能は、MCU が持つ MPU（Memory Protection Unit）をカーネルが制御することで実現しています。よって、MPU 内蔵の RX ファイミリ RX600 シリーズの環境であれば、本カーネルを使用することができます。

1.1.2 メモリ保護機能の特長

メモリ保護機能は、各アプリケーションが他のアプリケーションからメモリ領域破壊を防ぐための機能です。メモリ保護機能には以下の特長があります。

(1) 信頼性向上

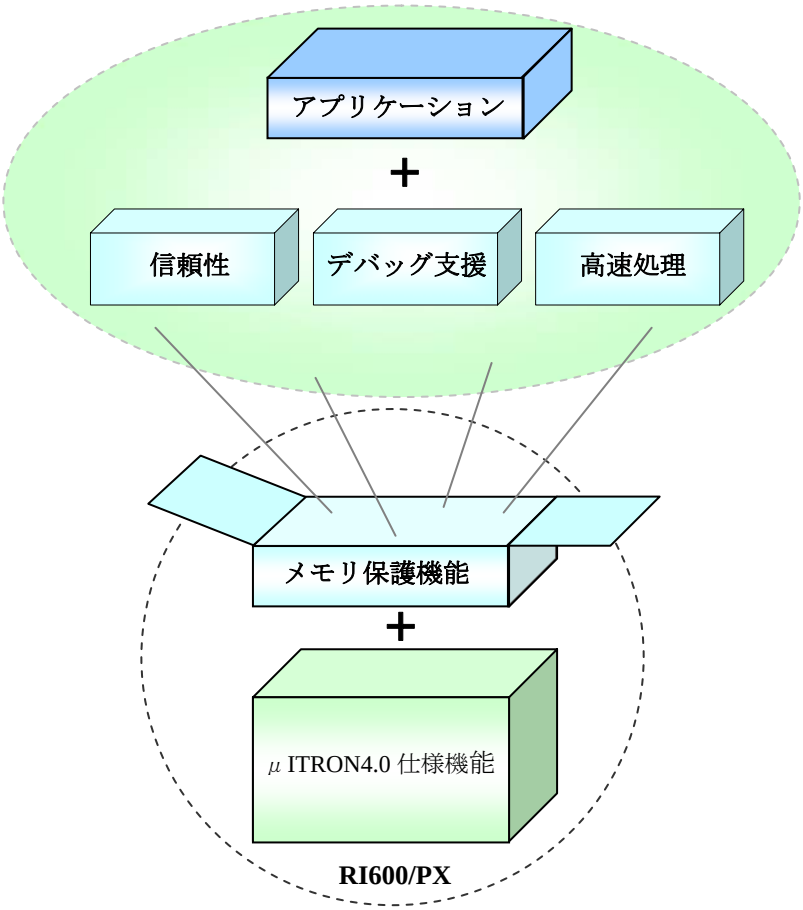
バグによる不正なアクセスや、悪意を持ったソフトウェアによる破壊からシステムを守ります。
また、システムの一部に問題があった場合にもシステム全体の動作を継続することができます。

(2) デバッグ支援

ソフトウェアモジュール間の障害の切り分けが容易になり、デバッグ作業の効率化を図ります。
また、不正なメモリ領域をアクセスした場合に、それを即時に検出することができます。

(3) 高速処理の実現

メモリ保護機能によりシステムで懸念されるのが、処理に対するオーバヘッドです。
カーネルは MCU が持つ MPU（Memory Protection Unit）を制御することで実現していますので、システムへの影響も最小限とし、高速処理に対応しています。



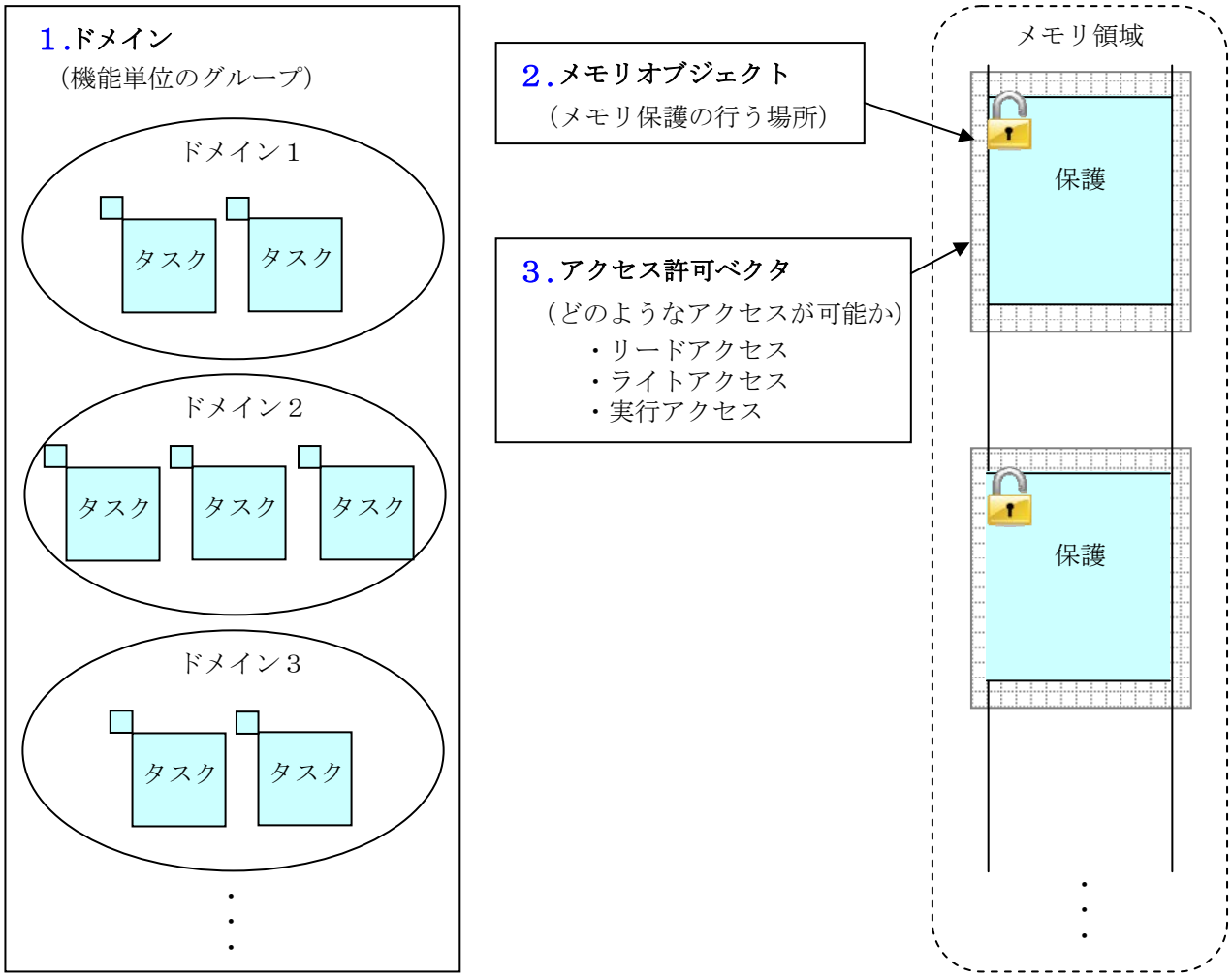
図－2 メモリ保護機能の特長

1.1.3 メモリ保護機能の制御

メモリ保護機能は、次の 3 つの情報より実現しています。

1．だれが ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ ドメイン
2．どこに対して ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ メモリオブジェクト
3．どのようなアクセスが可能か ・・・・・・・・・・・・・・・・・・ アクセス許可ベクタ

メモリ保護機能の制御を図1−3 に示します。



図−3 メモリ保護機能の制御

(1) ドメイン

ドメインは、ある特定の処理を行う機能単位のグループを示します。ドメインは 1～15 のドメイン ID として識別されます。

タスクとそのタスク例外処理ルーチンは、いずれかのドメインに所属します。

タスク以外のカーネル、割込みなどのハンドラはドメインには所属しません。

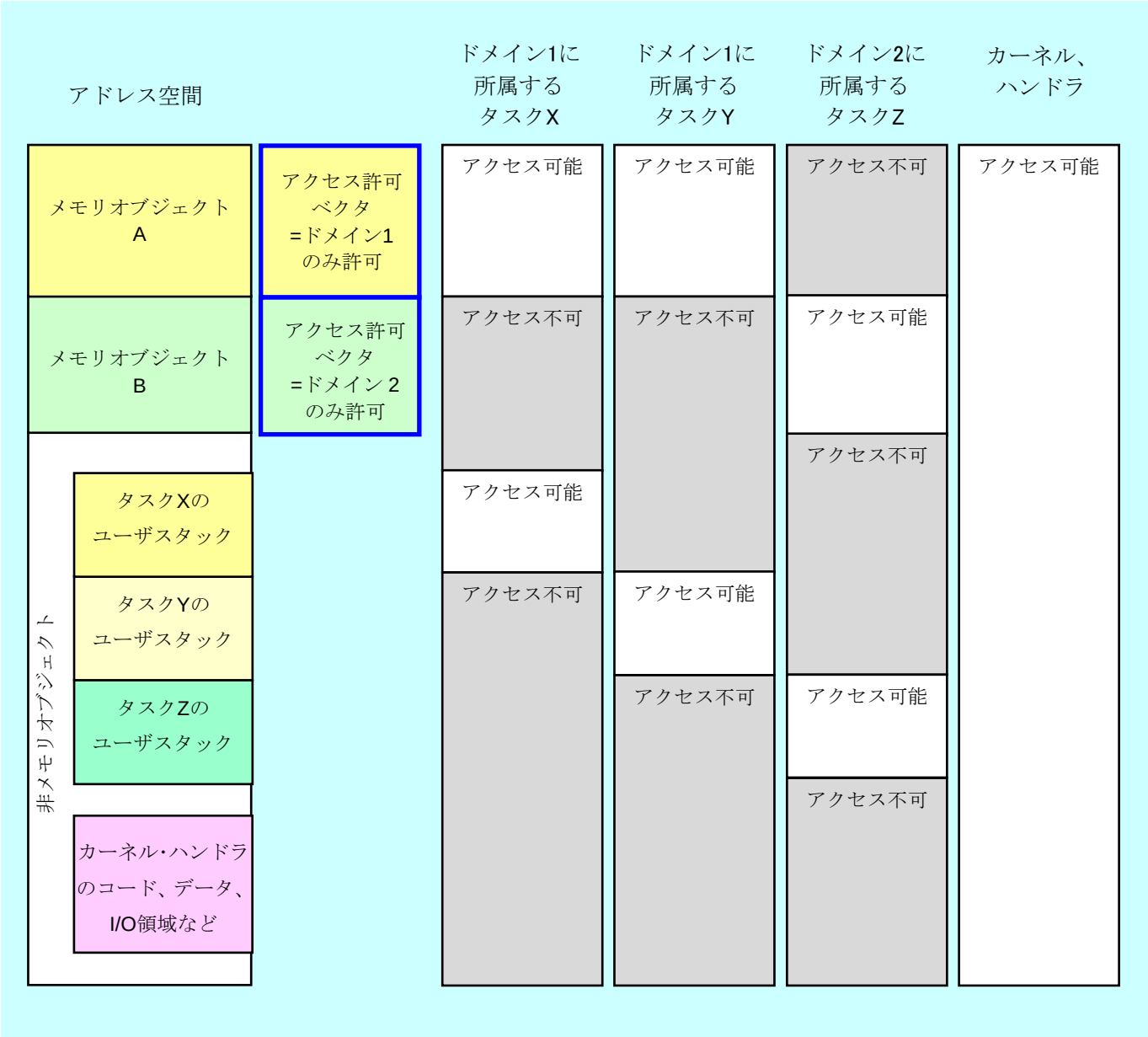
(2) メモリオブジェクト

メモリオブジェクトとはメモリ保護を行う場所を示します。
ドメインに所属するタスクからアクセスされるメモリ空間を指します。
タスクから、メモリオブジェクトに対して許可されていない場合は、エラーを検出します。

(3) アクセス許可ベクタ

メモリオブジェクトに対するリードアクセス、ライトアクセス、実行アクセスの許可を示します。
許可のされていないアクセスが行われた場合は、エラーを検出します。

メモリ保護の概要を図1－4 に示します。



図－4　メモリ保護の概要

2. メモリ保護機能の効果

RI600/PX のメモリ保護機能は、タスクのドメイン定義やメモリオブジェクトを使用することにより、システムに対する信頼性やデバッグ作業時において、高い効果を発揮します。

2.1 信頼性の高いシステムの実現

タスクやメモリオブジェクトをドメインで識別することで、所属するドメインからのみにアクセスが制限されるため、他のドメインのアクセスから保護されます。

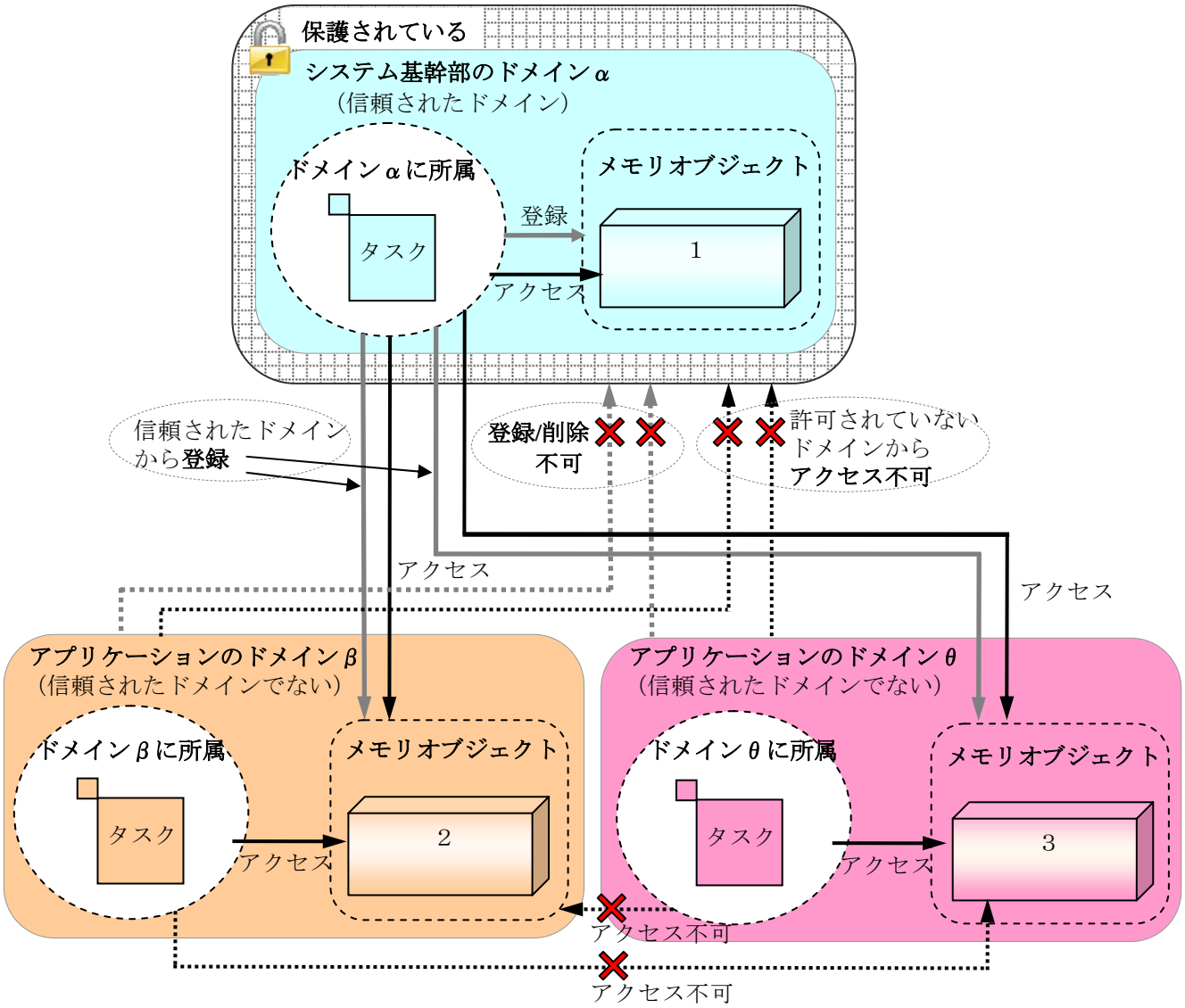
例えば、システム設計が完了したシステム基幹部に対して、品質レベルの異なるアプリケーションが追加された場合、システム基幹のタスクやオブジェクトは保護され、他のアプリケーションから破壊されるのを防ぐことができます。

他のアプリケーションが変更された場合や、追加された場合に発生する品質レベルの相違があっても、アプリケーションを混在させた環境でのシステムの評価が可能になります。

さらに、不正なアクセスを行ったタスクなどを局所的に停止する仕掛けを作ることが可能となり、システム停止や暴走を防ぐことができます。

これにより、システム機器の動作を継続することが可能となり、信頼性の高いシステムを実現します。

信頼性の高いシステムの実現例を図2－1 に示します。



メモリ保護機能では、オブジェクト生成・削除は『信頼されたドメイン』またはハンドラからのみ可能であるため、システム基幹部による資源以外を生成・定義することができません。

よって、システム基幹のドメインαに所属するタスクやオブジェクトは保護され、他のアプリケーションの変化や、追加されたアプリケーションによって影響や破壊されるのを防ぐことができます。

図－1　信頼性の高いシステムの実現例

2.2 デバッグ効率の向上

不正ポインタへのアクセスなどは、メモリ内容が破壊されてから動作不正が現象として表面化するまで時差があり、不正アクセスが実際に発生した原因を追究するのが困難であるケースがよくあります。

メモリ保護機能を使用することで直接的な原因を事前に検出し、動作不正の現象が発生する前に検知することができます。

その結果、システム開発中のデバッグ効率が飛躍的に向上します。

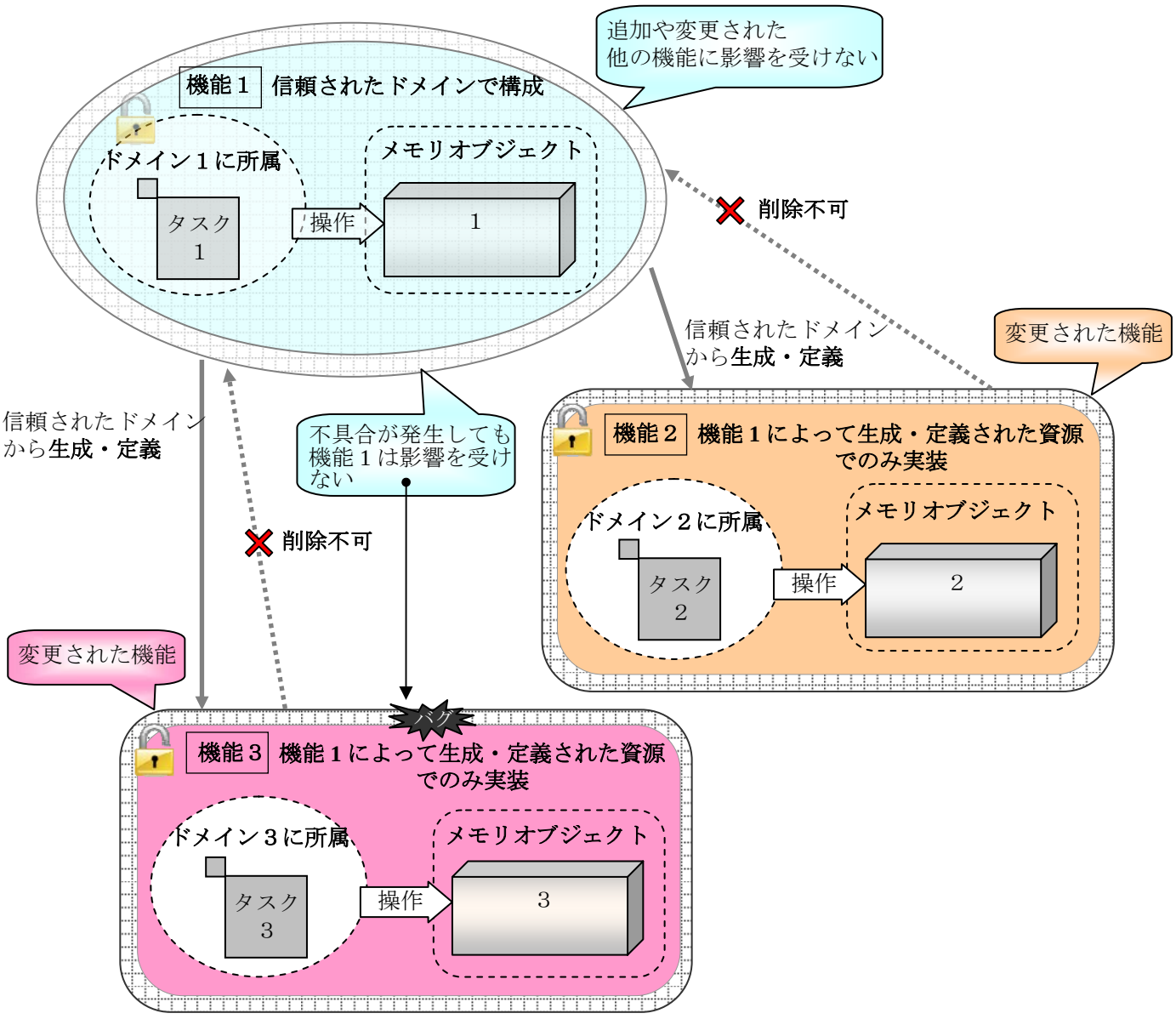
メモリ保護機能を持たないカーネルでは、これらの事前検出は行われず、原因追求までに多くの時間を費やしてしまいます。

2.2.1 機能単位の保護

機能単位に割り当てたカーネルオブジェクト（メモリ領域）に対するアクセスは、所属ドメインで制限されるため、その他のドメインのアクセスから保護されます。

機能毎のメモリ空間を保護することで、機能単位の品質確保に繋がります。

機能単位の保護を図2－2 に示します。



機能 1、2、3 はそれぞれ独立したアプリケーションです。

メモリ保護機能では、オブジェクト生成・削除は『信頼されたドメイン』またはハンドラからのみ可能であるため、機能 2、3 は機能 1 による資源以外を生成・定義することができません。

よって、機能 1 の動作が保護され、他の機能異によって影響や破壊されるのを防ぐことができます。

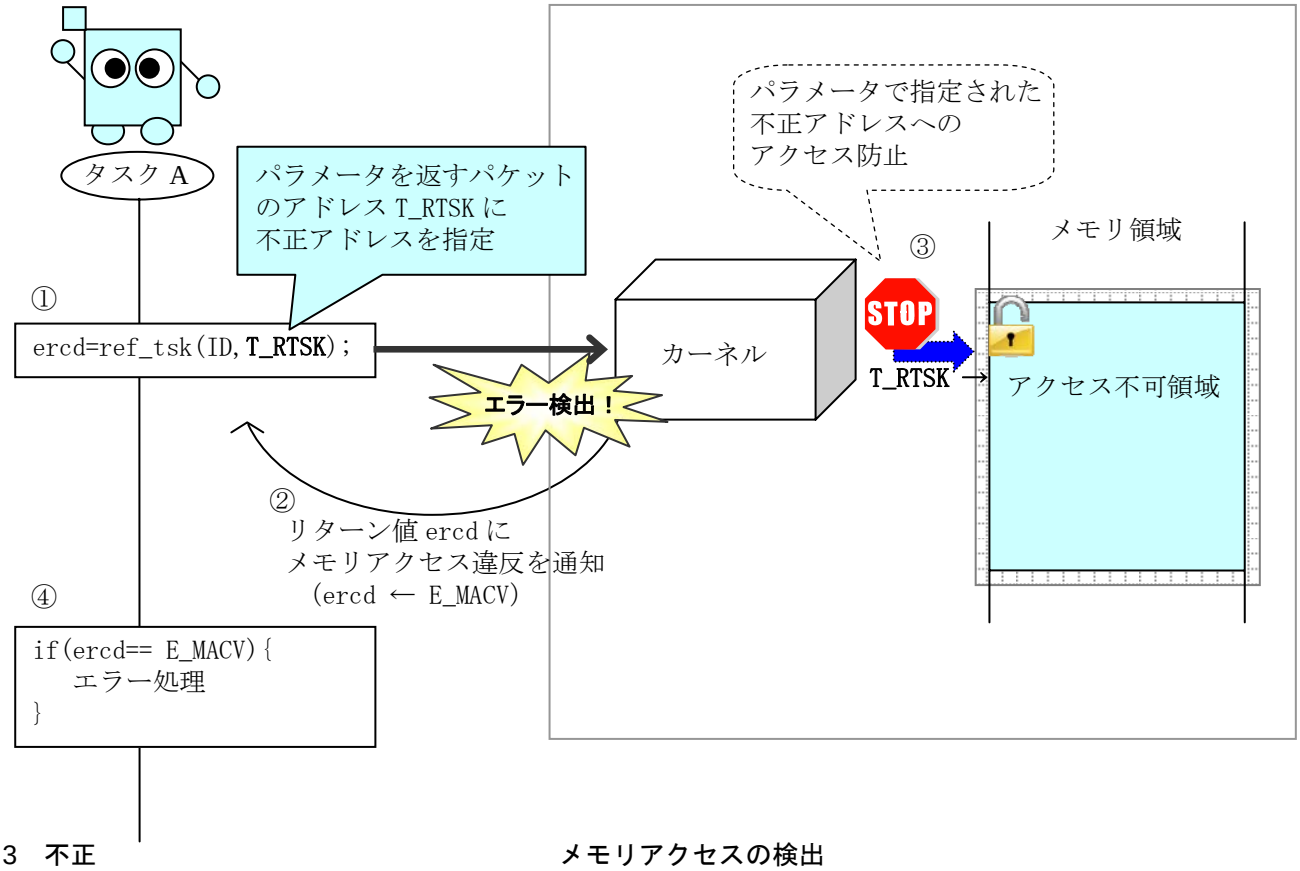
図－2　機能単位の保護

2.2.2　不正ポインタアクセスの事前検出

カーネルは、サービスコールを呼び出したタスクがポインタで指定された領域をアクセス可能であるかチェックし、不正を検出します。

アクセス許可がない場合には、サービスコールはエラーとしてメモリアクセス違反（E_MACV）を返します。

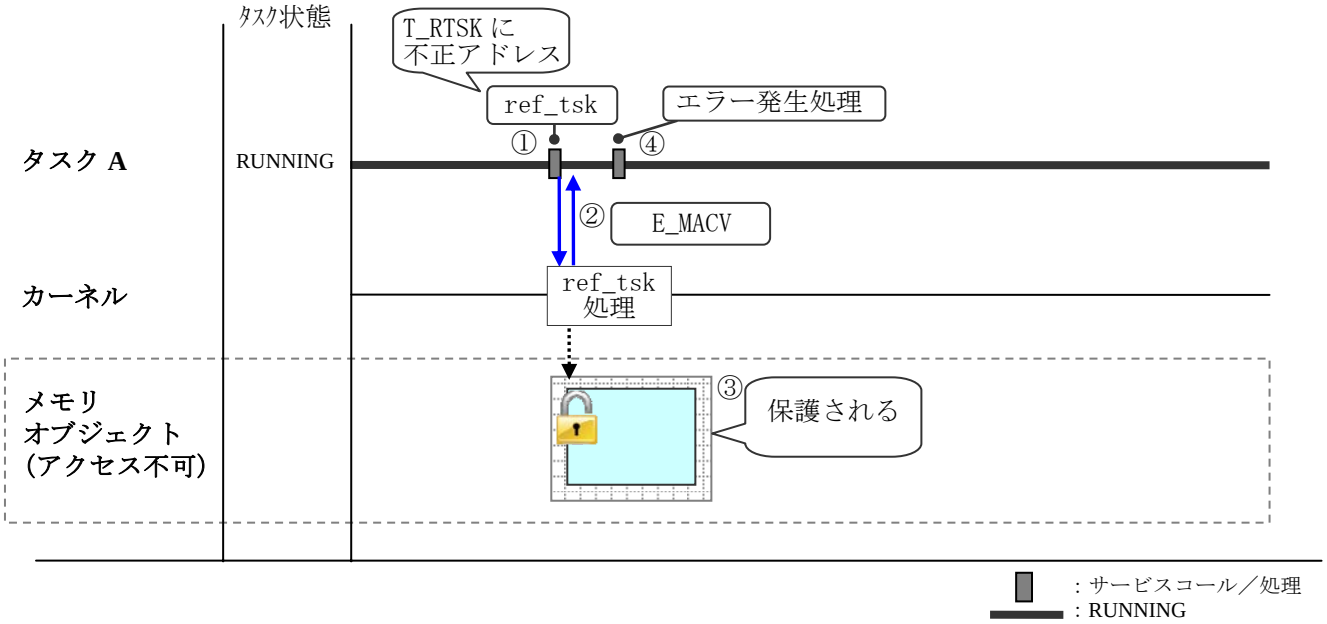
この機能は、タスクコンテキストから呼び出されたサービスコールでのみ行われます。



図－3　不正メモリアクセスの検出

① タスク A が、ref_tsk サービスコールでタスク状態を返すポインタ T_RTSK に不正アドレスを指定。
② カーネルはメモリアクセス違反としてエラーE_MACV を返す。
③ メモリオブジェクトとして定義していた領域は保護される。
④ タスク A は、ref_tsk のリターン値 E_MACV により対応処理を実施することができる。

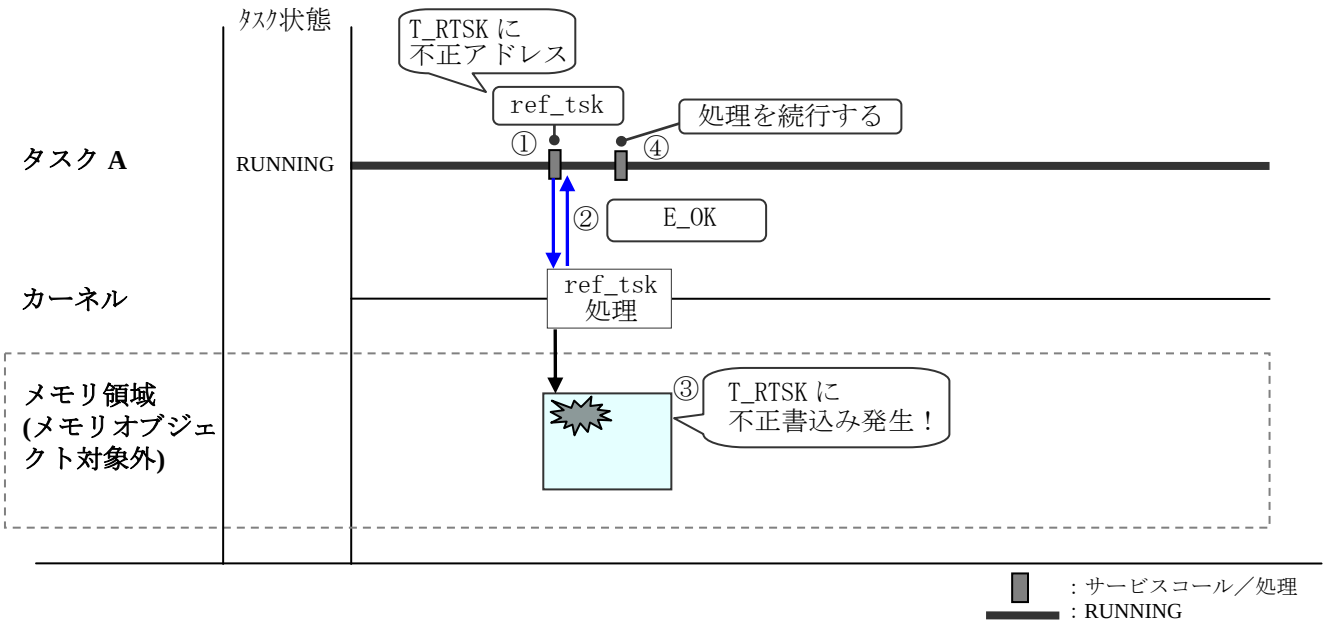
(1) 不正メモリアクセスの検出（メモリ保護機能あり）



図－4 不正メモリアクセスの検出（メモリ保護機能あり）

① タスク A が、ref_tsk サービスコールでタスク状態を返すポインタ T_RTSK に不正アドレスを指定。
② カーネルはメモリアクセス違反としてエラーE_MACV を返す。
③ メモリオブジェクトとして定義していた領域は保護される。
④ タスク A は、ref_tsk のリターン値 E_MACV により対応処理を実施することができる。

(2) 不正メモリアクセスの検出（メモリ保護機能なし）



図－5 不正メモリアクセスの検出（メモリ保護機能なし）

① タスク A が、ref_tsk サービスコールでタスク状態を返すポインタ T_RTSK に正アドレスを指定。
② カーネルは T_RTSK にタスク状態を設定し、リターン値に E_OK を返す。
③ アドレスとして指定された T_RTSK のメモリ領域は破壊される。
④ タスク A は、ref_tsk のリターン値 E_OK によりそのまま処理を続ける。後に不正書込みによる動作の不正が顕在化する。

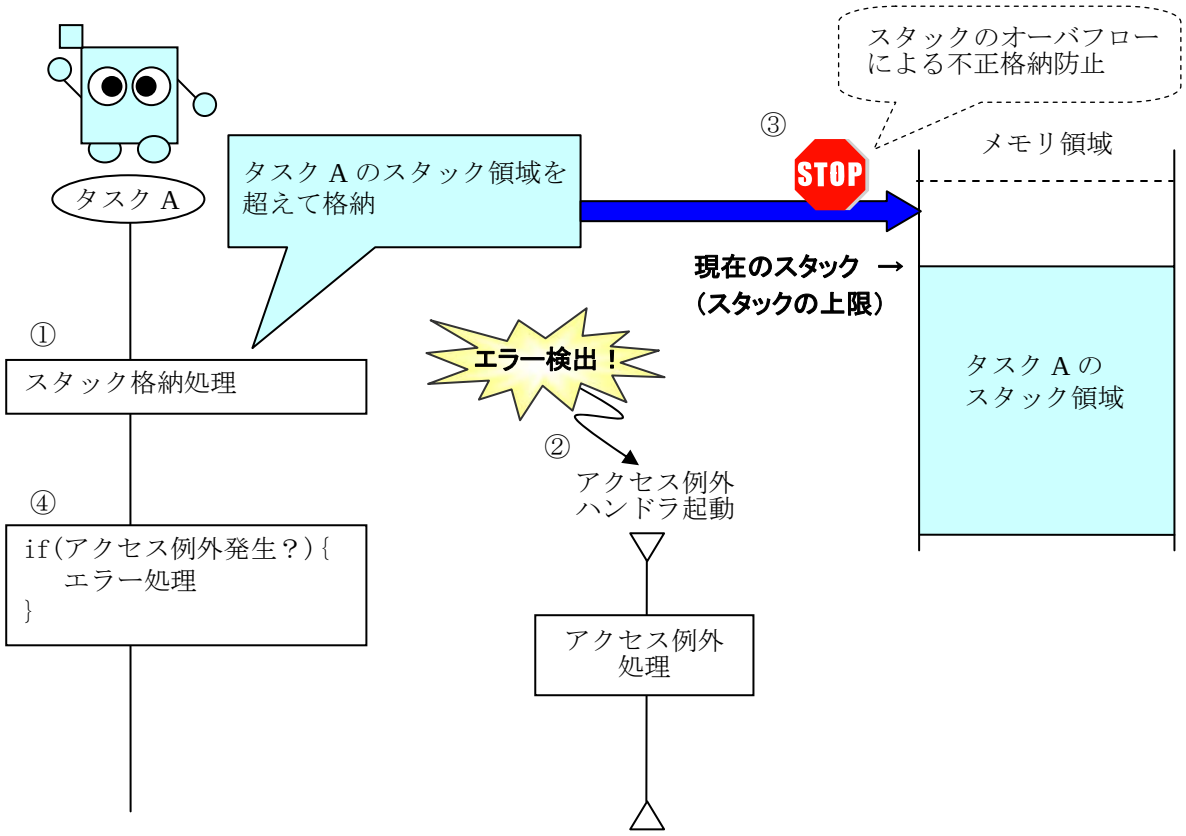
2.2.3 タスクスタック（ユーザスタック）保護

タスクスタックのオーバフローは、破壊されたデータを参照して初めて表面化するため、タスクスタックに対する事前の不正検出は高い効果があります。

タスクスタックに対しては、アクセス例外処理の起動による検出と、カーネルがサービスコールで検出する２つのエラー検出方法があります。

(1) 他のタスクからのスタックアクセス検出

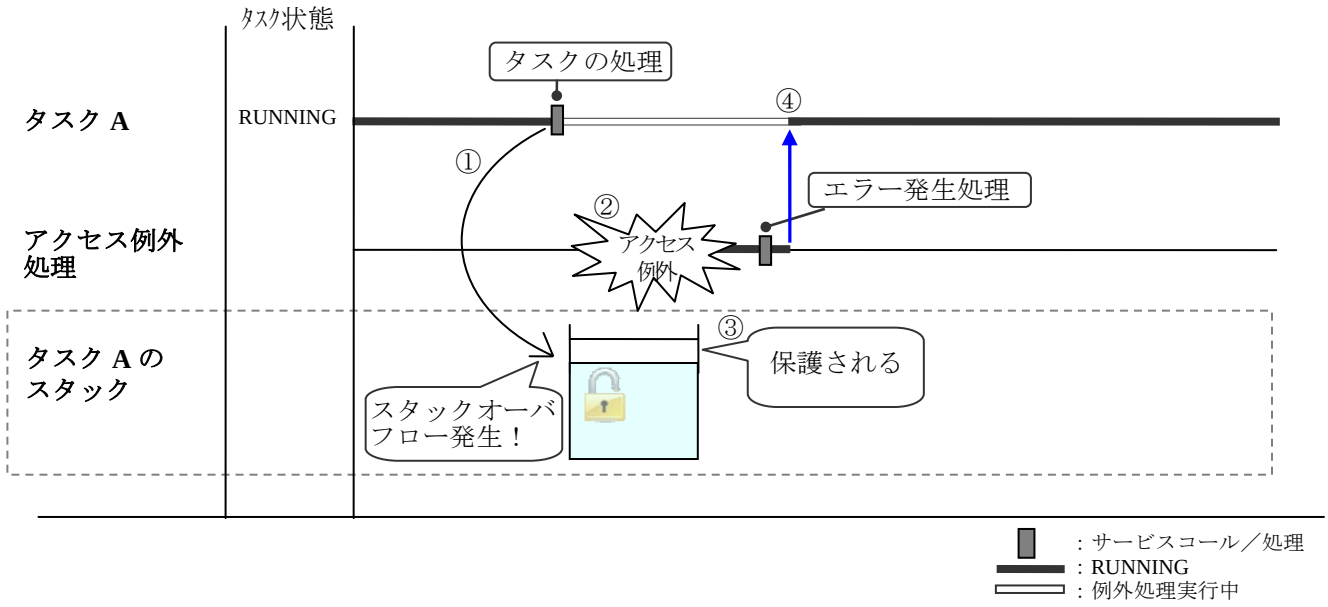
各タスクのユーザスタックは､他のタスクからはアクセスできません｡ユーザスタックのオーバフローや､他のタスクのユーザスタックをアクセスしようとすると、アクセス例外ハンドラが起動されます。



図－6 ユーザスタックの不正アクセス検出

① タスク A で、スタックオーバフローが発生した。
② アクセス例外ハンドラが起動され、アクセスエラーの対応処理を実施することができる。
③ アクセス例外処理によりスタックオーバフローを検出し、不正アクセスから保護される。
④ タスク A は、アクセス例外処理からの復帰で処理を再開することができる。

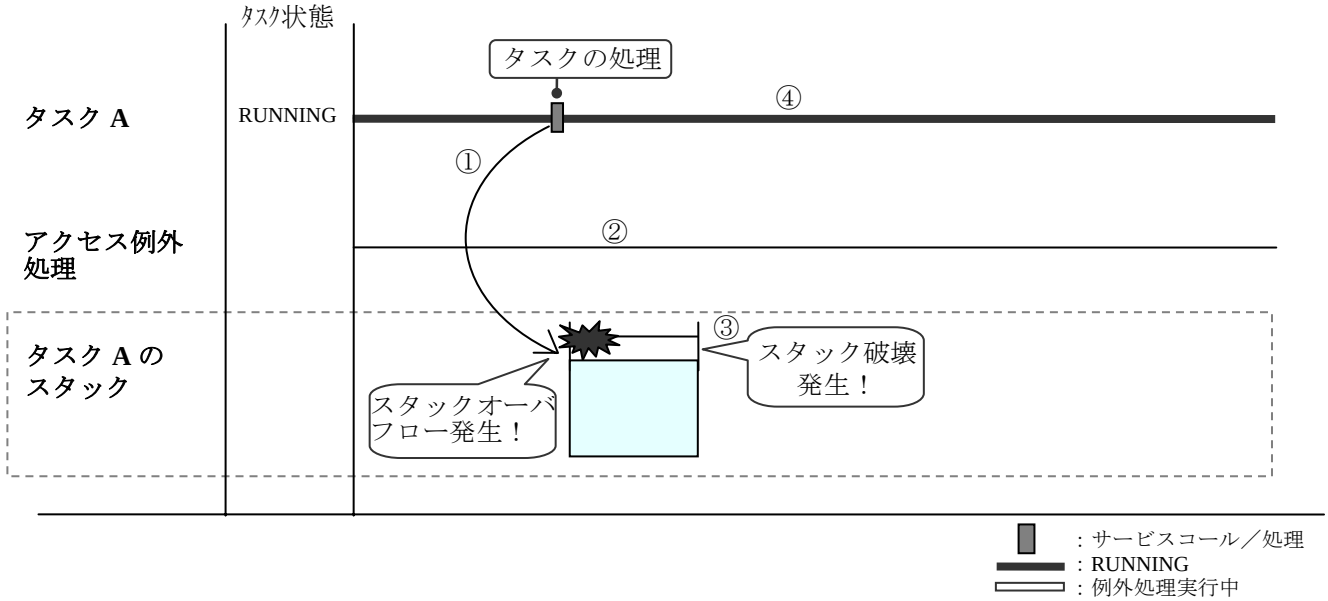
(a)　ユーザスタックの不正メモリアクセスの検出（メモリ保護機能あり）



図－7　ユーザスタックの不正メモリアクセスの検出（メモリ保護機能あり）

① タスク A で、スタックオーバフローが発生した。
② アクセス例外ハンドラが起動され、アクセスエラーの対応処理を実施することができる。
③ アクセス例外処理によりスタックオーバフローを検出し、不正アクセスから保護される。
④ タスク A は、アクセス例外処理からの復帰で処理を再開することができる。

(b)　ユーザスタックの不正メモリアクセスの検出（メモリ保護機能なし）



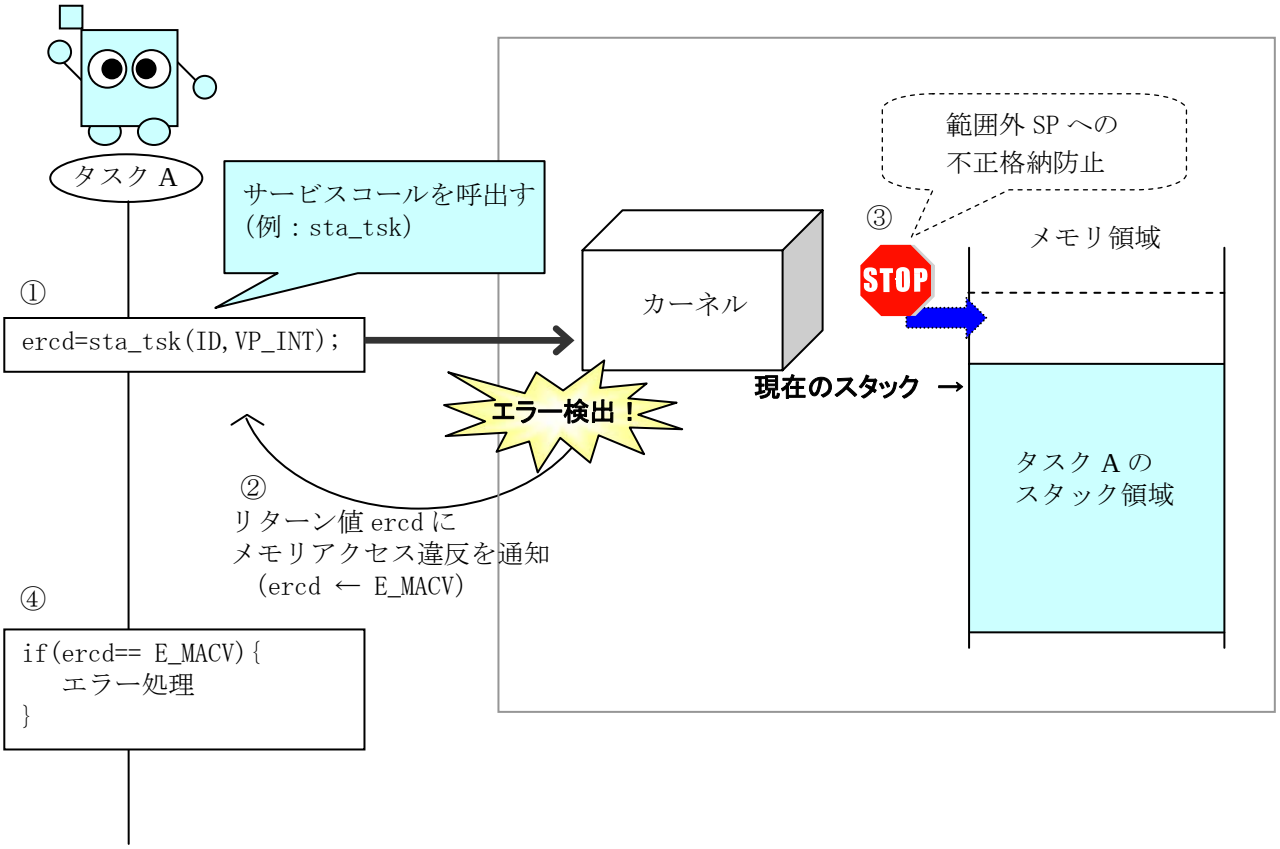
図－8　ユーザスタックの不正メモリアクセスの検出（メモリ保護機能なし）

① タスク A で、スタックオーバフローが発生した。
② アクセス例外ハンドラは起動されない。
③ スタック領域が不正にアクセスされ、破壊される。
④ タスク A は、不正アクセスを検知できないため処理を続行する。後に不正書込みによる動作の不正が顕在化する。

(2) カーネルによるユーザスタック範囲外の検出

カーネルは、タスクスタックのオーバフローをその時点で検出します。

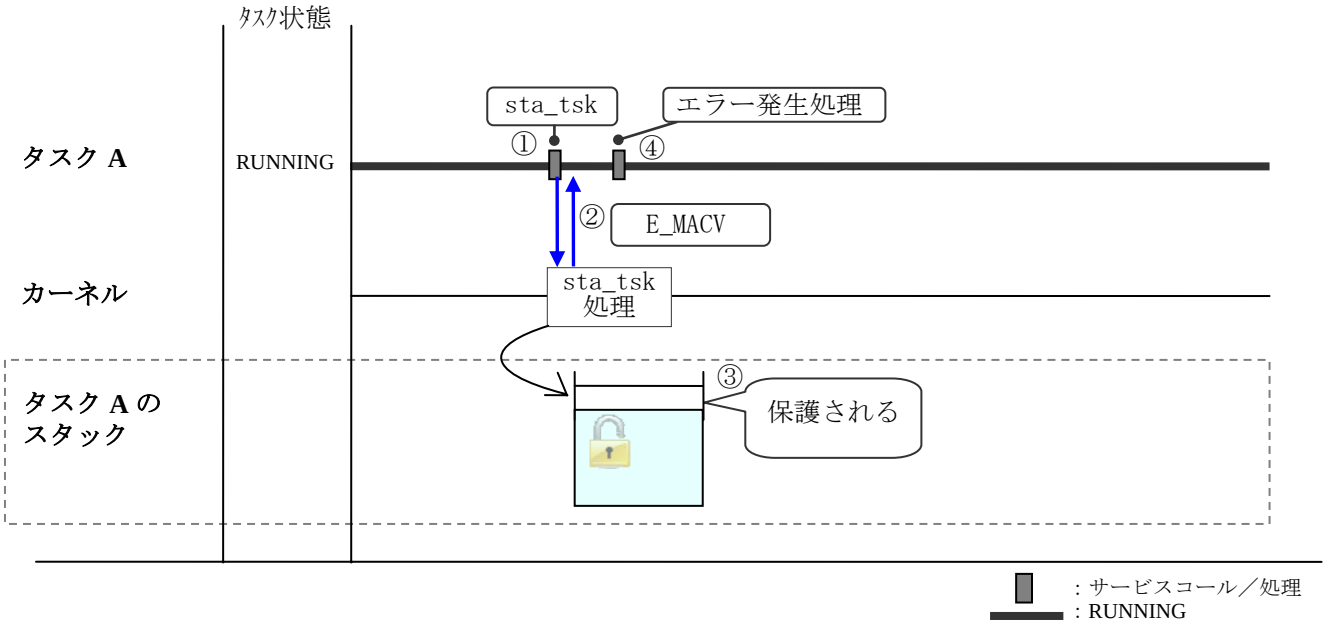
タスクから呼び出されたサービスコールでカーネルがユーザスタックを使用する場合、カーネルはスタックポインタがそのタスクのユーザスタック領域内にあるかどうかを検査します。範囲外の場合にはエラーとしてメモリアクセス違反（E_MACV）を返します。



図－9　サービスコールによるユーザスタックのエラー検出

① タスク A が、サービスコール（例えば sta_tsk）を呼び出す。
② カーネルはスタックポインタが呼出しタスクのユーザスタックポインタ領域の範囲外を指していることを検出して、メモリアクセス違反としてエラーE_MACV を返す。
③ スタック領域に隣接したメモリは、不正アクセスから保護される。
④ タスク A は、サービスコールのリターン値 E_MACV により対応処理を実施することができる。

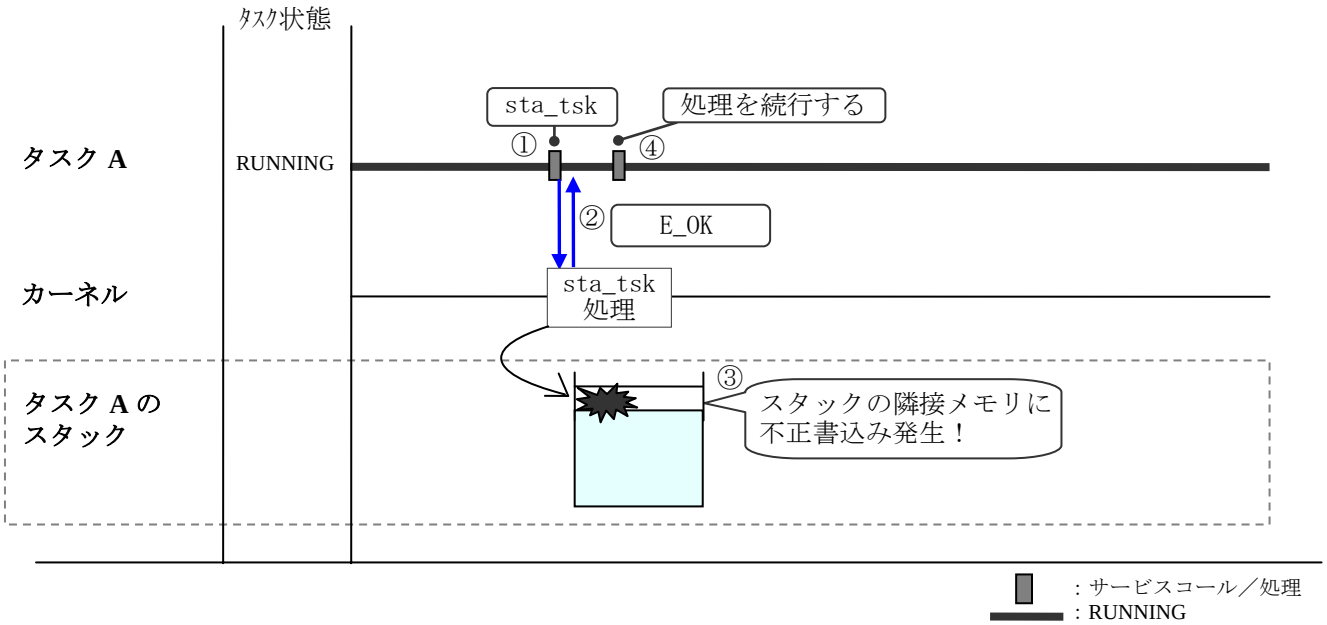
(a) カーネルによるユーザスタック範囲外の検出（メモリ保護機能あり）



図－10　カーネルによるユーザスタック範囲外の検出（メモリ保護機能あり）

① タスク A が、サービスコール（例えば sta_tsk）を呼び出す。
② カーネルはスタックポインタが呼出しタスクのユーザスタックポインタ領域の範囲外を指していることを検出して、メモリアクセス違反としてエラーE_MACV を返す。
③ スタック領域に隣接したメモリは、不正アクセスから保護される。
④ タスク A は、サービスコールのリターン値 E_MACV により対応処理を実施することができる。

(b) カーネルによるユーザスタック範囲外の検出（メモリ保護機能なし）



図－11　カーネルによるユーザスタック範囲外の検出（メモリ保護機能なし）

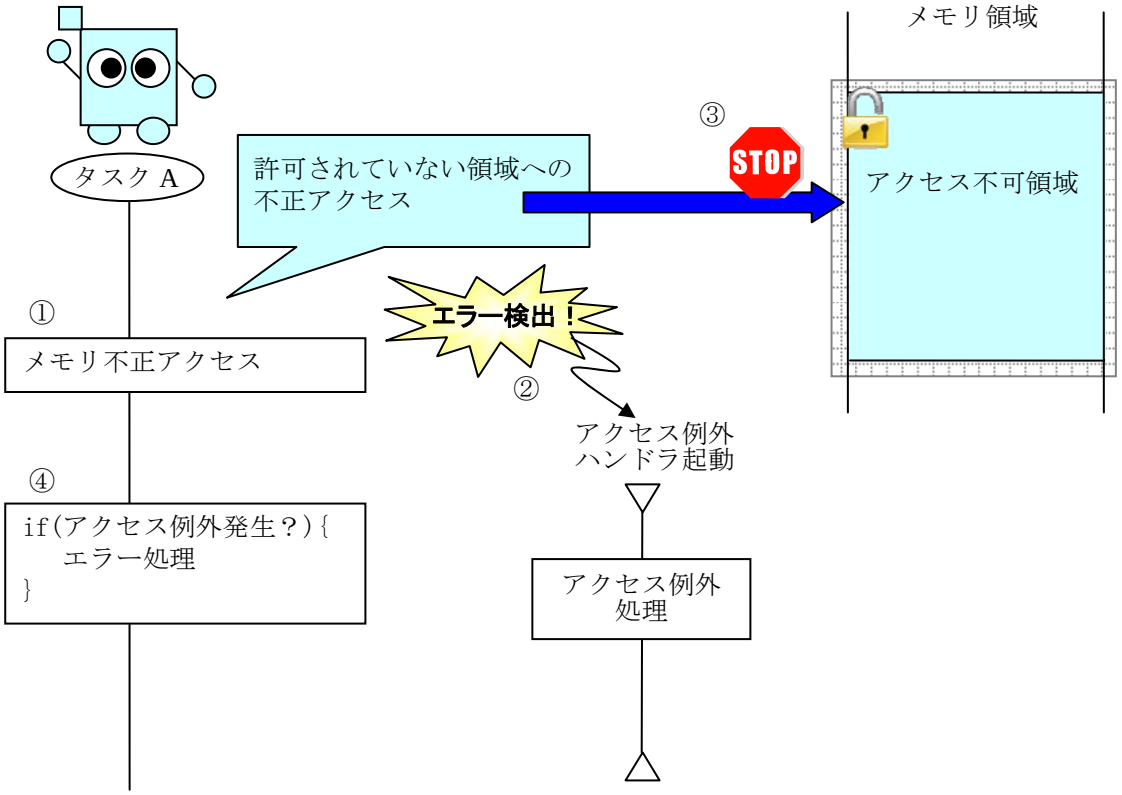
① タスク A が、サービスコール（例えば sta_tsk）を呼び出す。
② カーネルはスタックポインタが呼出しタスクのユーザスタックポインタ領域の範囲外かの検出はしない。このため、リターン値に E_OK を返す。
③ スタック領域に隣接したメモリは不正書込みにより破壊される。
④ タスク A は、サービスコールのリターン値 E_OK によりそのまま処理を続ける。後に不正書込みによる動作の不正が顕在化する。

2.2.4 メモリ属性不正の検出

ROM 空間への書き込みアクセスを検出します。

また、RAM 空間での実行アクセスは、許可ベクタに従って不正を検出します。

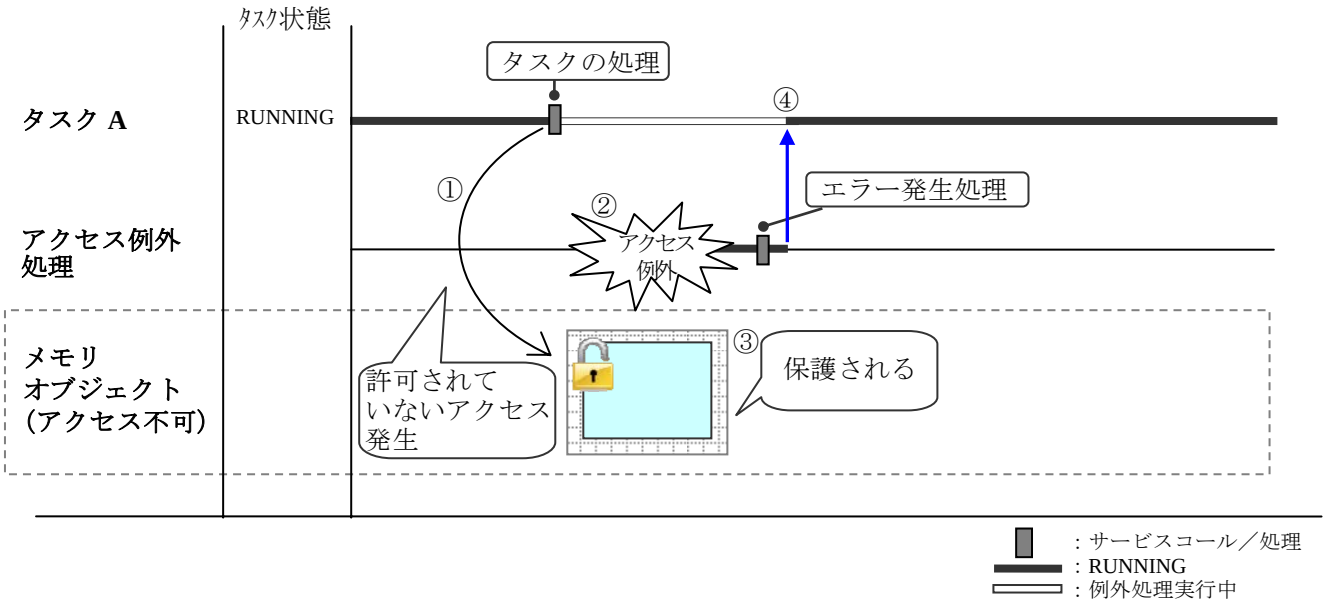
タスク・タスク例外処理ルーチンは、アクセス許可された領域(メモリオブジェクト)のみにアクセスできます。許可されていない領域をアクセスすると、アクセス例外ハンドラが起動されます。



図－12 メモリ属性不正の検出

① タスク A が、許可されていないメモリオブジェクトの領域に対してアクセスした。
② アクセス例外ハンドラが起動され、アクセスエラーの対応処理を実施することができる。
③ アクセス例外発生により不正アクセスを検出し、不正アクセスから保護される。
④ タスク A は、アクセス例外処理からの復帰で処理を再開することができる。

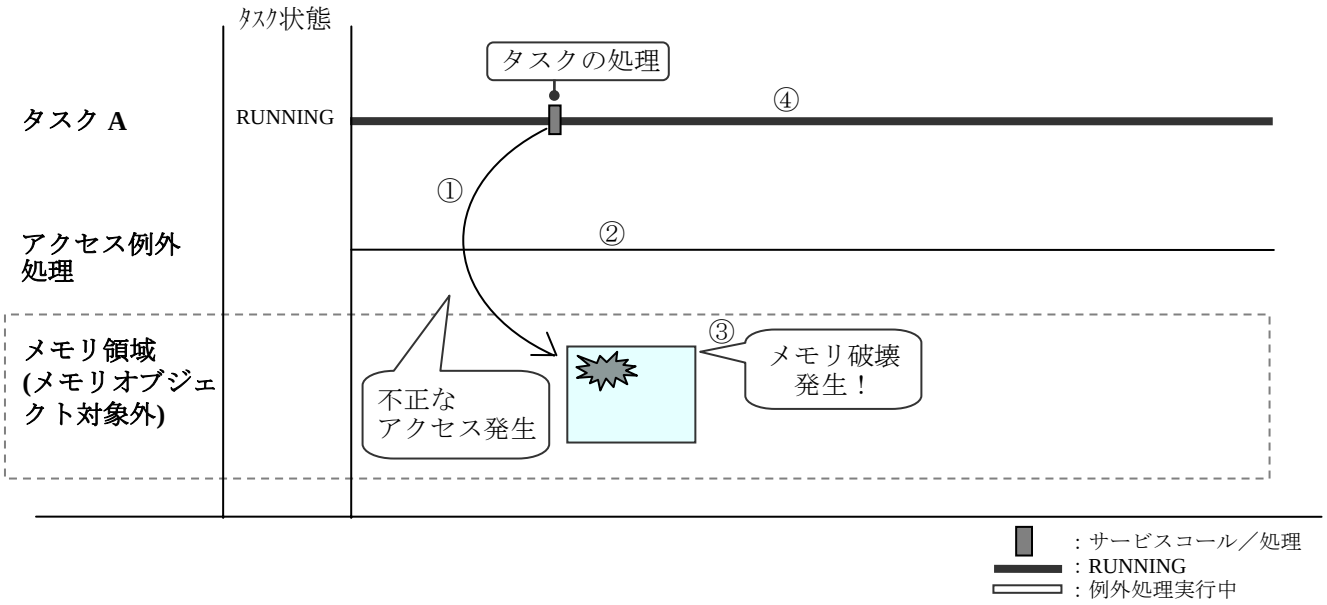
(1) メモリ属性不正の検出（メモリ保護機能あり）



図－13　メモリ属性不正の検出（メモリ保護機能あり）

① タスク A が、許可されていないメモリオブジェクトの領域に対してアクセスした。
② アクセス例外ハンドラが起動され、アクセスエラーの対応処理を実施することができる。
③ アクセス例外発生により不正アクセスを検出し、不正アクセスから保護される。
④ タスク A は、アクセス例外処理からの復帰で処理を再開することができる。

(2) メモリ属性不正の検出（メモリ保護機能なし）



図－14　メモリ属性不正の検出（メモリ保護機能なし）

① タスク A が、メモリオブジェクトの対象外の領域に対して不正にアクセスした。
② アクセス例外ハンドラは起動されない。
③ メモリ領域は不正にアクセスされ破壊される。
④ タスク A は、不正アクセスを検知できないまま処理を続行する。後に不正書込みによる動作の不正が顕在化する。

2.2.5 信頼されたドメイン

RI600/PX では、メモリ以外の保護機能をサポートしていません。このため、悪意による不正メモリアクセスを防ぐために、RI600/PX は「信頼されたドメイン」と呼ぶ機能をサポートしています。

ソフトウェア構成に変更を与えるサービスコールは、「信頼されたドメイン」に所属するタスクからのみ呼出し可能となっています。これらのサービスコールを、信頼されていないドメインからの呼び出した場合は、エラーとしてオブジェクトアクセス違反（E_OACV）が検出されます。

信頼されたドメインのタスクから呼出し可能なサービスコールを表2－1 に示します。

表－1　信頼されたドメインのタスクから呼出し可能なサービスコール

No.	呼出し可能なサービスコール	機能内容
1	cre_???	???の生成（???はカーネルオブジェクトを示します。）
2	del_???	???の削除（???はカーネルオブジェクトを示します。）
3	def_tex	タスク例外処理ルーチンの定義
4	ata_mem	メモリオブジェクトの登録
5	det_mem	メモリオブジェクトの登録解除
6	sac_mem	メモリオブジェクトのアクセス許可ベクタの変更

2.2.6 保護の対象とならないもの

RI600/PX では、以下に示すものは保護の対象にならないので注意が必要です。

(1) オブジェクトに対する保護機能

μITRON4.0/PX 仕様には、メモリに対する保護機能のほかに、オブジェクトに対する保護機能があります。RI600/PX の保護機能では、オブジェクトに対するアクセス保護はサポートしておりません。

(2) 割込みなどのハンドラからのアクセスに対する保護機能

ドメインに所属しないハンドラからは、カーネル管理領域を含め、すべてのメモリ空間のアクセスが許可されています。

ハンドラからのアクセスは保護の対象になりません。

(3) ユーザ管理領域のカーネル管理領域に対する保護機能

ユーザ管理領域内部に確保されるカーネル管理領域は保護されません。

保護されない領域を以下に示します。

・固定長／可変長メモリプール本体内でのカーネル管理領域
・メールボックスのメッセージに含まれるカーネル管理領域（メッセージヘッダ部）

3. RI600/4 からの移行

アプリケーションを RI600/4 から移行した場合、メモリ保護機能を組み込むための情報として、次の情報が必要となります。

- ドメインの定義
- タスクの所属ドメインの定義
- メモリオブジェクトの定義
- アクセス例外ハンドラ

3.1 ドメインの定義

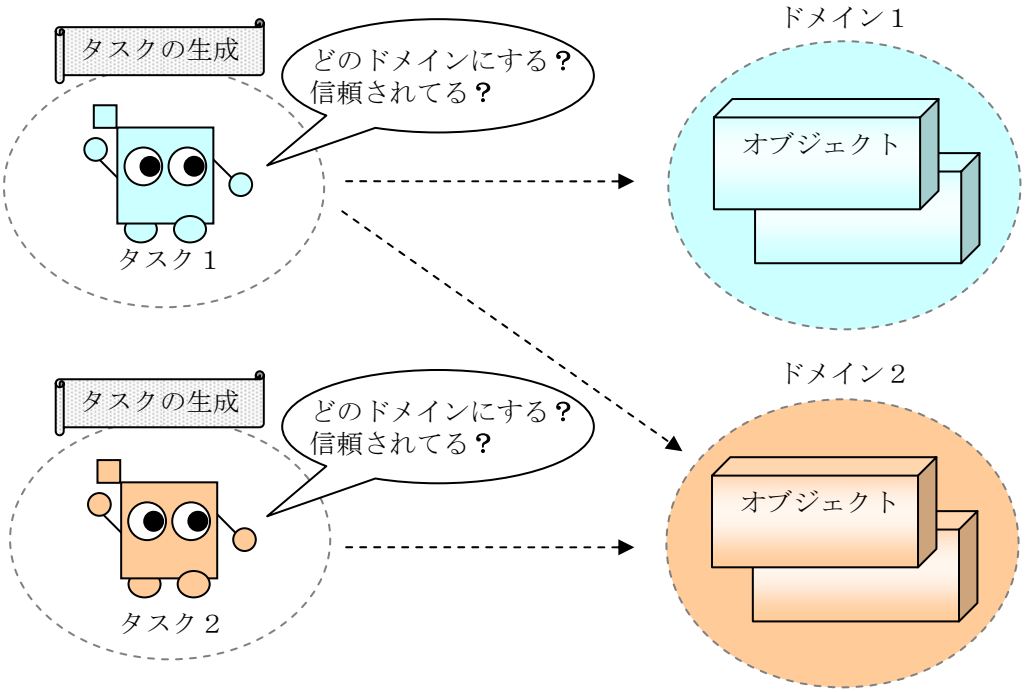
ドメインの定義として、cfg ファイルの domain[ ]に『信頼されたドメイン』の指定を行います。
cfgファイルによるドメインの定義の記述例を図3－1 に示します。

```
// Domain Definition
domain[1]{
        trust   = YES;          // ドメインID=1を信頼されたドメインとする
};
```

図－1　cfgファイルによる静的な生成のドメイン指定の記述例

3.2　タスクの所属ドメインの定義

タスクの所属ドメインの定義は、タスクの生成時に指定します。



図－2　タスクの所属ドメインの定義

(1)　静的生成

cfg ファイルの task[ ]の domain_num に所属するドメインを指定します。
cfgファイルによる生成時のドメイン指定の記述例を図3－3 に示します。

```
//Task Definition
task[1]{
    name            = TASK_ID1;
    entry_address   = task1();
    stack_size      = 0x200;
//  stack_section   = SURI_STACK;
    priority        = 2;
    initial_start   = ON;
    exinf           = 0x1234;
    texrtn          = Texrtn1;
    domain_num      = 1;            // 所属するドメインを1に指定
};
```

図－3　cfgファイルによるタスク生成のドメイン指定の記述例

(2) **動的生成**

サービスコール cre_tsk（acre_tsk）のパラメータであるタスク属性として指定します。
タスク属性の tskatr には以下を指定できます。

tskatr := (TA_HLNG | [TA_ACT] | [TA_DOM(domid)])

以下に、tskatr のビット位置を示します。

b15～b8	b7	b6	b5	b4	b3	b2	b1	b0
0	ドメイン ID (TA_DOM(domid))				0	0	TA_ACT (=2)	TA_HLNG (=0)

TA_DOM(domid) (ビット 4～7)には生成するタスクが所属するドメイン ID を指定します。
0 の場合は、本サービスコールを呼び出したタスクと同じドメインとします。
domid には、所属するドメイン ID を指定します。domid に TDOM_SELF を指定した場合は、本サービスコールを呼び出したタスクと同じドメインとします。

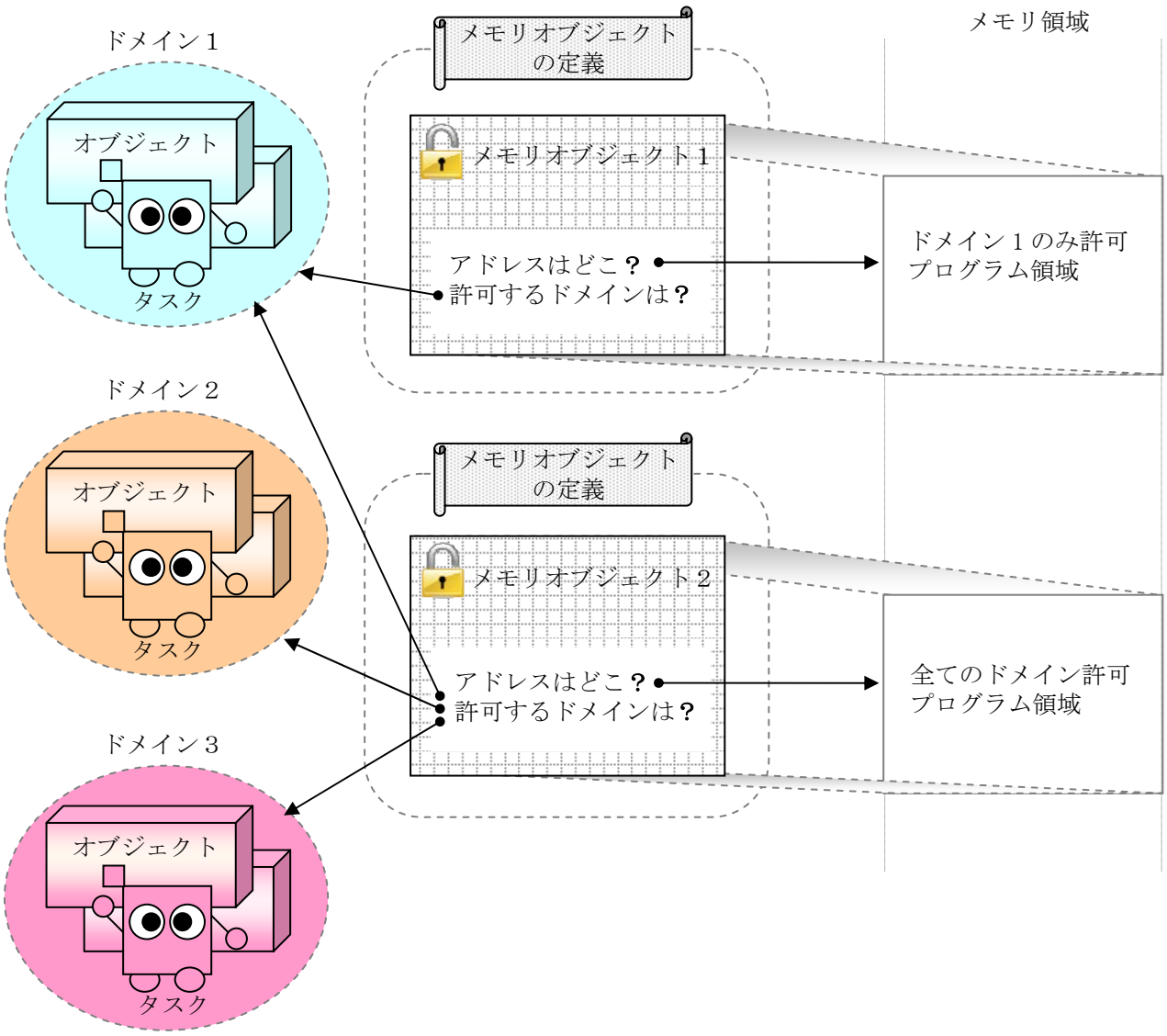
サービスコールcre_tskによるドメイン定義の記述例を図3－4 に示します。

```
cretsk.tskatr  = TA_HLNG|TA_ACT|TA_DOM(1);
cretsk.exinf   = 0x1234;
cretsk.task    = (FP)task1;
cretsk.itskpri = 2;
cretsk.stksz   = 0x200;
cretsk.stk     = STK001;
ercd = cre_tsk(TASK_ID1,&cretsk);
if (ercd != E_OK)
{
    error();
}
```

図－4　サービスコールcre_tskによるドメイン指定の記述例

3.3　メモリオブジェクトの定義

メモリオブジェクトの定義は、対象となるメモリ領域と、どのドメインに対してアクセスを許可するかを指定して、メモリオブジェクトを登録します。



図－5　メモリオブジェクトの定義

(1) 静的定義

cfg ファイルの memory_object[ ]にメモリオブジェクトを定義します。
cfgファイルによるメモリオブジェクト定義の記述例を図3－6 に示します。

```
// Memory Object Definition
memory_object[1]{
    start_address = BU_MASTERDOM;
    end_address   = RU_MASTERDOM_2;
    acptn1        = 0x0001;          // ドメインID=1のみドリードアクセス可能
    acptn2        = 0x0001;          // ドメインID=1のみライトアクセス許可
    acptn3        = 0;               // 全ドメインの実行禁止
};
```

図－6　cfgファイルによるメモリオブジェクト定義の記述例

(2) 動的定義

サービスコール ata_mem により、メモリオブジェクトを定義します。
サービスコールata_memによるメモリオブジェクト定義の記述例を図3－7 示します。

```
    pk_atamem.mematr = 0x0;
    pk_atamem.base   = __sectop("BU_RAM");
    pk_atamem.size   = (UW)__secend("BU_RAMEND") - (UW)__sectop("BU_RAM");
    p_acvct.acptn1 = TACP_SHARED;        // 全ドメインのリード許可
    p_acvct.acptn2 = TACP_SHARED;        // 全ドメインのライト許可
    p_acvct.acptn3 = TACP_SHARED;        // 全ドメインの実行許可
    ercd = ata_mem(&pk_atamem,&p_acvct);
    if (ercd != E_OK)
    {
        error();
    }
```

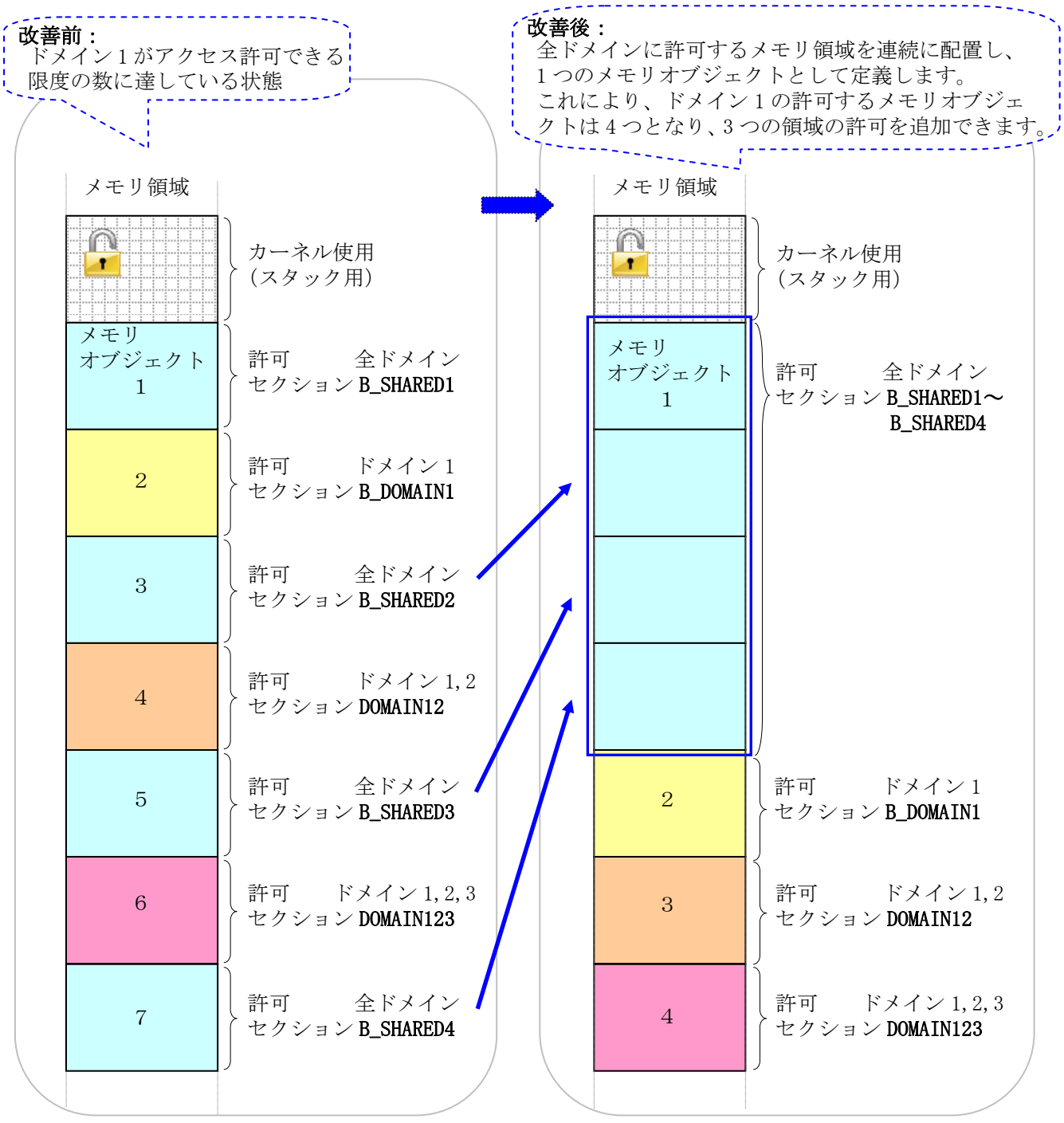
図－7　サービスコールata_memによるメモリオブジェクト定義の記述例

3.3.1 ドメインとメモリオブジェクトの定義数

ひとつのドメインにアクセス許可できる領域の数は、カーネルが内部で使用しているスタックも含め 8 つまで可能です。

たとえば、アクセス許可パターンに TACP_SHARED（全ドメインにアクセスを許可）を複数指定した場合には、定義できる数を容易に超えてしまいます。

アクセス許可パターン毎に連続したメモリに配置し、まとめて 1 つのメモリオブジェクトとして定義するなどして、メモリマップの設計を行ってください。



図－8 ドメインとオブジェクトの定義数

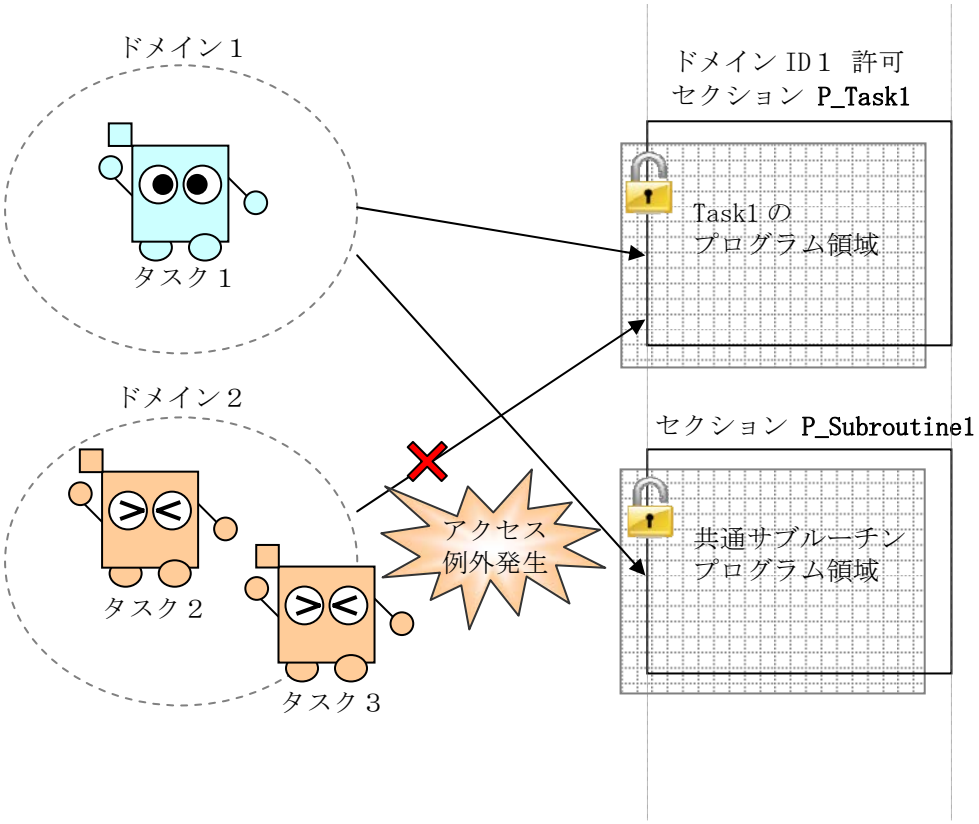
3.3.2 メモリオブジェクトの定義例

メモリオブジェクトとして定義する領域の例を、以下に示します。

(1) タスクのプログラム領域の保護

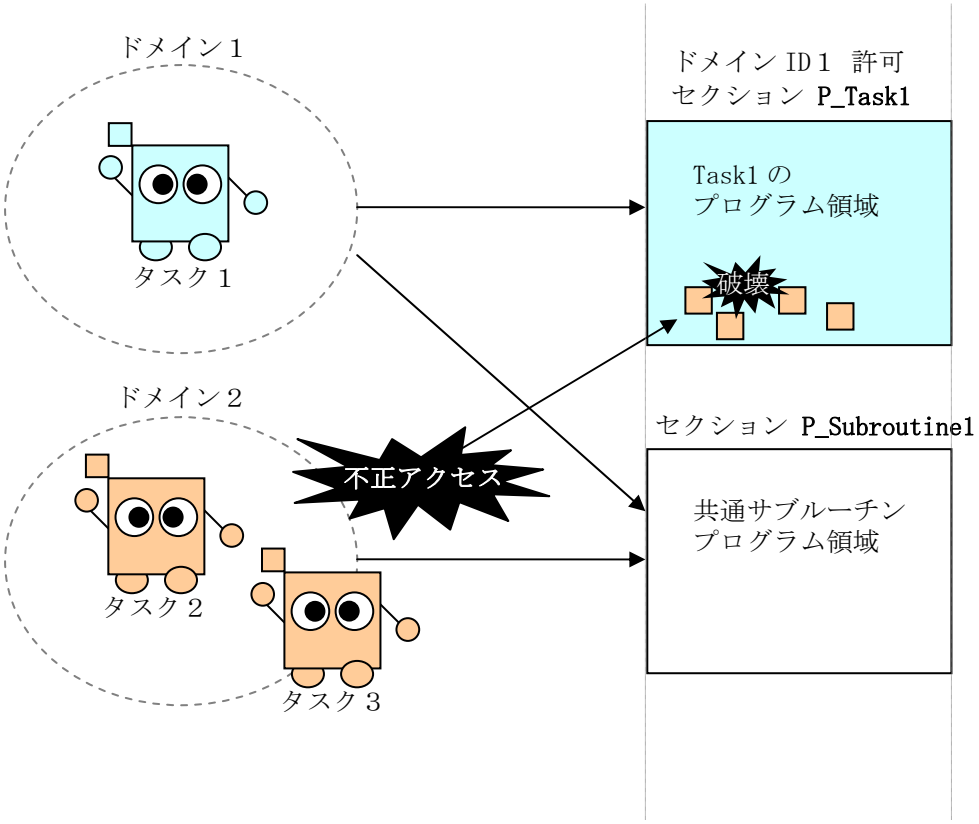
タスク、およびタスクを構成するサブルーチンのプログラム領域を保護対象に定義する場合には、プログラム領域対して、アクセス許可ベクタに実行アクセス許可を指定します。

複数のタスクで共通で使用するサブルーチンに対しては、実行されるすべてのドメイン ID を、実行アクセス許可に指定する必要があります。



図－9 タスクのプログラム領域保護の概要

メモリ保護機能がなければ、不正アクセスが表面化せずに、異なる保護ドメインのタスクによる不正アクセスを検出することができません。



図－10　タスクのプログラム領域の保護無しの場合

タスクのプログラム領域の保護するcfgファイルへの設定例を図3－11 に示します。

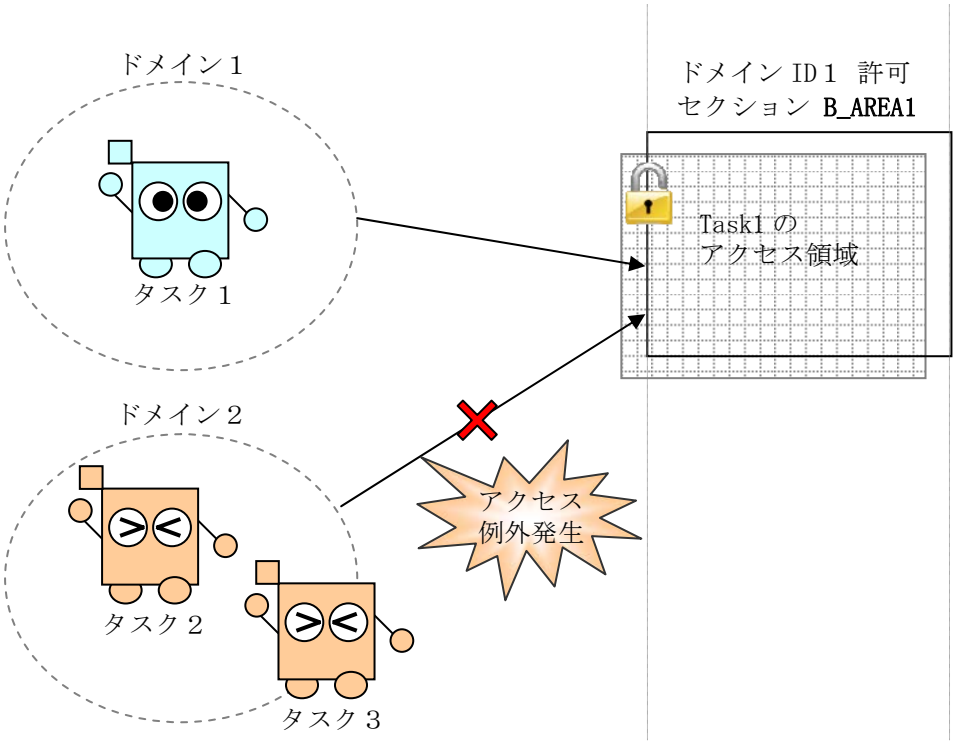
```
// Domain Definition
domain[1]{
    trust           = YES;          // 信頼されたドメインに指定
};

//Task Definition
task[1]{
    name            = TASK_ID1;
    initial_start   = ON;
    entry_address   = Task1();
    stack_size      = 0x200;
    priority        = 15;
    exinf           = 0x1234;
    domain_num      = 1;            // タスクが所属するドメインIDを1に指定
};
// Memory Object Definition
memory_object[1]{
    start_address   = P_Task1;      // 先頭アドレスにセクション名“P_Task1”を指定
    end_address     = P_Task1;      // 終端アドレスにセクション名“P_Task1”を指定
    acptn1          = 0;            // オペランドリードアクセス許可パターンに0（許可しない）を指定
    acptn2          = 0;            // オペランドライトアクセス許可パターンに0（許可しない）を指定
    acptn3          = 1;            // 実行アクセス許可パターンにドメインID1の許可を指定
}
memory_object[2]{
    start_address   = P_Subroutine1;  // 先頭アドレスにセクション名“P_Subroutine1”を指定
    end_address     = P_Subroutine1;  // 終端アドレスにセクション名“P_Subroutine1”を指定
    acptn1          = 0;              // オペランドリードアクセス許可パターンに0（許可しない）を指定
    acptn2          = 0;              // オペランドライトアクセス許可パターンに0（許可しない）を指定
    acptn3          = TACP_SHARED;    // 実行アクセス許可パターンに全ドメインIDの許可を指定
};
```

図－11　タスクのプログラム領域保護の記述例

(2) タスクがアクセスするメモリ領域の保護

タスクがアクセスするメモリ領域を保護対象に定義する場合、対象メモリ領域に対して、アクセス許可ベクタにリードアクセス許可とライトアクセス許可を指定します。



図－12　タスクがアクセスするメモリ領域保護の概要

タスクがアクセスするメモリ領域保護のcfgファイルへの設定例を図3－13 に示します。

```
// Domain Definition
domain[1]{
    trust           = YES;          // 信頼されたドメインに指定
};

//Task Definition
task[1]{
    name            = TASK_ID1;
    initial_start   = ON;
    entry_address   = Task1();
    stack_size      = 0x200;
    priority        = 15;
    exinf           = 0x1234;
    domain_num      = 1;            // タスクが所属するドメインIDを1に指定
};
// Memory Object Definition
memory_object[1]{
    start_address   = B_AREA1;      // 先頭アドレスにセクション名“B_AREA1”を指定
    end_address     = B_AREA1;      // 終端アドレスにセクション名“B_AREA1”を指定
    acptn1          = 1;            // オペランドリードアクセス許可パターンにドメインID1の許可を指定
    acptn2          = 1;            // オペランドライトアクセス許可パターンにドメインID1の許可を指定
    acptn3          = 0;            // 実行アクセス許可パターンに0（許可しない）を指定
};
```

図－13　タスクがアクセスするメモリ領域保護の記述例

(3)　タスクスタック領域の保護

タスクのスタック領域は、カーネルによって保護対象に登録されます。
カーネルが自動的にチェックするので、メモリオブジェクトとして登録する必要がありません。

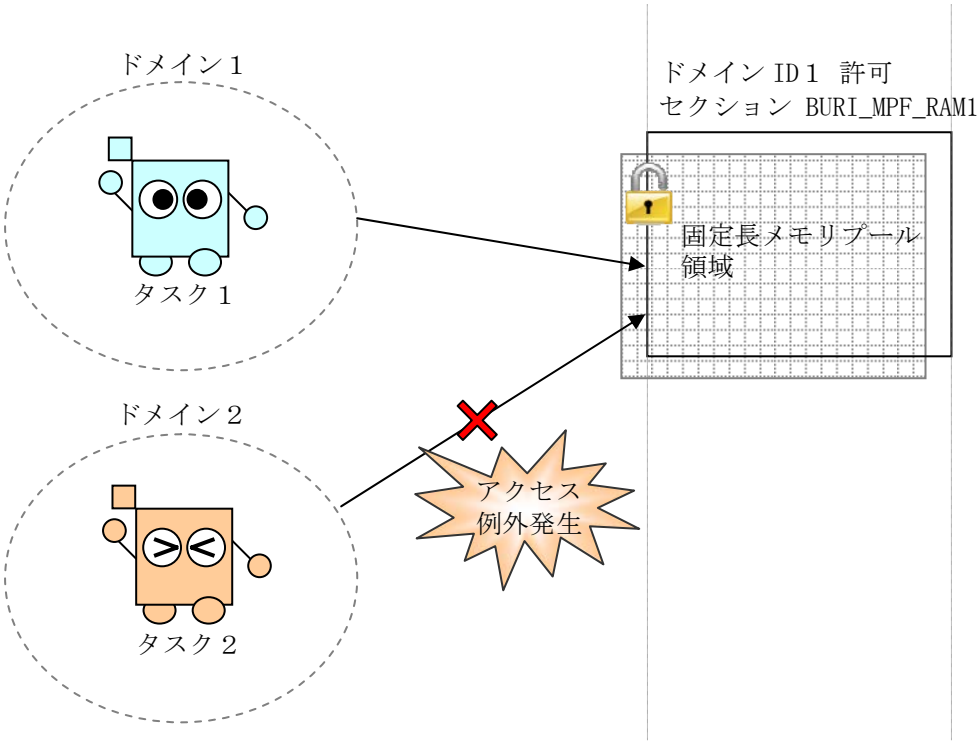
(4) 可変長/固定長メモリプール領域の保護

メモリプール領域を保護対象に定義する場合は、メモリプール領域を使用するタスクからアクセス可能なメモリオブジェクトにする必要があります。

メモリプールの生成時に指定する固定長/可変長メモリプールの領域を保護対象に指定します。

メモリプール領域をコード領域として使用する場合には、アクセス許可ベクタに、実行アクセス許可を指定します。

メモリプール領域をデータとして使用する場合には、アクセス許可ベクタに、リードアクセス許可とライトアクセス許可を指定します。



図－14　固定長メモリプール領域保護の概要

固定長メモリプール領域保護のcfgファイルへの設定例を図3－15 に示します。

```
// Domain Definition
domain[1]{
    trust           = YES;          // 信頼されたドメインに指定
};

//Task Definition
task[1]{
    name            = TASK_ID1;
    initial_start   = ON;
    entry_address   = Task1();
    stack_size      = 0x200;
    priority        = 15;
    exinf           = 0x1234;
    domain_num      = 1;            // タスクが所属するドメインIDを1に指定
};
// Fixed-size Memory Pool Definition
memorypool[1]{
      name          = MPF_ID1;
      section       = BURI_MPF_RAM1;
      num_block     = 1024;
      siz_block     = 8;
      wait_queue    = TA_TFIFO;
};
// Memory Object Definition
memory_object[1]{
    start_address   = BURI_MPF_RAM1; // 先頭アドレスにセクション名“BURI_MPF_RAM1”を指定
    end_address     = BURI_MPF_RAM1; // 終端アドレスにセクション名“BURI_MPF_RAM1”を指定
    acptn1          = 1;            // オペランドリードアクセス許可パターンにドメインID1の許可を指定
    acptn2          = 1;            // オペランドライトアクセス許可パターンにドメインID1の許可を指定
    acptn3          = 1;            // 実行アクセス許可パターンにドメインID1の許可を指定
};
```

図－15　固定長メモリプール領域保護の記述例

3.3.3 メモリオブジェクトから外すべき領域

(1) タスクのユーザスタック領域

タスクのユーザスタック領域は、メモリオブジェクト以外の領域にしてください。

ユーザスタック領域に対する保護は、カーネルが自動的に登録してチェックしますので、メモリオブジェクトから外してください。

(2) データキュー領域

データキュー領域は、カーネルからアクセスする領域ですので、メモリオブジェクト以外の領域にしてください。

メモリオブジェクト内にデータキュー領域を作成した場合は、そのメモリオブジェクトへのライトアクセスが許可されたタスクが、誤ってデータキュー領域を書き換えてしまう危険があります。

(3) メッセージバッファ領域

メッセージバッファ領域は、カーネルからアクセスする領域ですので、メモリオブジェクト以外の領域にしてください。

メモリオブジェクト内にメッセージバッファ領域を作成した場合は、そのメモリオブジェクトへのライトアクセスが許可されたタスクが、誤ってメッセージバッファ領域を書き換えてしまう危険があります。

(4) 固定長メモリプール管理領域

固定長メモリプール管理領域は、メモリオブジェクト以外の領域にしてください。

メモリオブジェクト内に固定長メモリプール管理領域を作成した場合は、そのメモリオブジェクトへのライトアクセスが許可されたタスクが、誤ってメモリプール管理領域を書き換えてしまう危険があります。

3.3.4 メモリオブジェクトの操作

カーネルではメモリオブジェクト管理機能のサービスコールをサポートしています。
このため、メモリオブジェクトに対する操作をサービスコールにより行うことができます。
メモリオブジェクト管理機能のサービスコールを表3−1 に示します。

表−1　メモリオブジェクト管理機能サービスコール一覧

No.	サービスコール	機能
1	ata_mem	メモリオブジェクトの登録
2	det_mem	メモリオブジェクトの登録解除
3	sac_mem	メモリオブジェクトのアクセス許可ベクタの変更
4	vprb_mem	メモリ領域に対するアクセス権のチェック
5	ref_mem	メモリオブジェクト状態の参照

3.4 アクセス例外ハンドラ

アクセス例外ハンドラは、タスクコンテキストで許可されていないメモリアクセスが発生した時に呼び出されます。アクセス違反の要因を取り除いて復帰するなどの処理を行うことができます。
アクセス例外ハンドラはシステムに 1 つだけ存在し、ユーザが作成します。

アクセス例外ハンドラの API を用意していますので、渡される引数を基に、ユーザのシステムに合った処理を実現することが可能です。

3.4.1 アクセス例外ハンドラの作成

アクセス例外ハンドラ開始関数のコーディング例を図3－16 に示します。

```
#include "kernel.h"
#include "kernel_id.h"
void _RI_sys_access_exception(UW pc ,UW psw, UW sts, UW addr);     // 関数名は固定
void _RI_sys_access_exception(UW pc ,UW psw, UW sts, UW addr)
{
    /* 処理 */
}
```

図－16　アクセス例外ハンドラ開始関数のコーディング例

アクセス例外ハンドラの開始関数のAPIは、図3－16 の通りです。
関数名は"_RI_sys_access_exception"に決められています。

渡される引数の仕様を表3－2 に示します。
アクセス例外ハンドラはスーパバイザモードで実行されます。
アクセス例外ハンドラでは、非タスクコンテキストから呼び出せるサービスコールを使用できます。

表－2　アクセス例外ハンドラに渡される引数

No.	引数	レジスタ	解説
1	pc	R1	アクセス例外を発生させた命令アドレス
2	psw	R2	アクセス例外発生時の PSW
3	sts	R3	アクセス違反要因 MPU(Memory Protection Unit)の MPESTS レジスタ値です。
4	addr	R4	オペランドアクセスエラーの場合は、そのアクセスアドレス (MPU の MPDEA レジスタの値) 実行アクセスエラーの場合は不定です。

4. 追加機能

RI600/PX では、RI600/4 に対してメモリ保護機能以外に、以下の機能を追加しています。

(1) オブジェクトの動的な生成・削除

オブジェクトを動的に生成・削除する機能を追加しました。これにより、システムの構造化設計と機能向上を図ります。

(2) タスク例外処理機能

タスクに対してタスク例外を要求すると、タスク例外処理ルーチンが起動される機能です。タスク毎に定義することが可能であるため、タスク毎の例外処理の機能として効果的です。

これらの追加機能については、『RI600/PX ユーザーズマニュアル』をご参照ください。

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/inquiry

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 未使用端子の処理
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】電源投入時は，製品の状態は不定です。
   電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。
   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. リザーブアドレス（予約領域）のアクセス禁止
   【注意】リザーブアドレス（予約領域）のアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. クロックについて
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。
   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
5. 製品間の相違について
   【注意】型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。
   同じグループのマイコンでも型名が違うと、内部 ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA機器、通信機器、計測機器、AV機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注2. 本資料において使用されている「当社製品」とは、注1において定義された当社の開発、製造製品をいいます。

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

※営業お問合せ窓口の住所・電話番号は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス販売株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル） (03)5201-5307

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/contact/

Colophon 1.1