

(12) 特許協力条約に基づいて公開された国際出願

(19) 世界知的所有権機関
国際事務局



(43) 国際公開日
2011 年 8 月 25 日(25.08.2011)

PCT

(10) 国際公開番号
WO 2011/101972 A1

- (51) 国際特許分類:
G06F 21/22 (2006.01) G06F 21/24 (2006.01)
- (21) 国際出願番号: PCT/JP2010/052465
- (22) 国際出願日: 2010 年 2 月 18 日(18.02.2010)
- (25) 国際出願の言語: 日本語
- (26) 国際公開の言語: 日本語
- (71) 出願人 (米国を除く全ての指定国について): 株式会社東芝 (Kabushiki Kaisha Toshiba) [JP/JP]; 〒1058001 東京都港区芝浦一丁目 1 番 1 号 Tokyo (JP).
- (72) 発明者; および
- (75) 発明者/出願人 (米国についてのみ): 春木 洋美 (HARUKI, Hiroyoshi) [JP/JP]; 〒1058001 東京都港区芝浦一丁目 1 番 1 号 株式会社東芝 知的財産部内 Tokyo (JP). 橋本 幹生 (HASHIMOTO, Mikio) [JP/JP]; 〒1058001 東京都港区芝浦一丁目 1 番 1 号 株式会社東芝 知的財産部内 Tokyo (JP). 中西 福友 (NAKANISHI, Fukutomo) [JP/JP]; 〒1058001 東京都港区芝浦一丁目 1 番 1 号 株式会社東芝 知的財産部内 Tokyo (JP). 林 良太郎 (HAYASHI, Ryotaro) [JP/JP]; 〒1058001 東京都港区芝浦一丁目 1 番 1 号 株式会社東芝

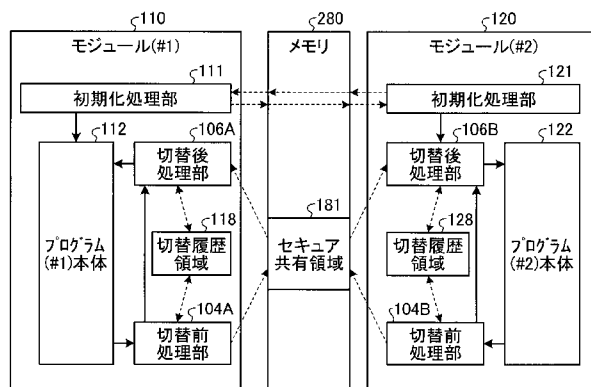
- 知的財産部内 Tokyo (JP). 藤松 由里恵 (FUJIMATSU, Yurie) [JP/JP]; 〒1058001 東京都港区芝浦一丁目 1 番 1 号 株式会社東芝 知的財産部内 Tokyo (JP). 城勘 友秀 (JOKAN, Tomohide) [JP/JP]; 〒1058001 東京都港区芝浦一丁目 1 番 1 号 株式会社東芝 知的財産部内 Tokyo (JP). 川端 健 (KAWABATA, Takeshi) [JP/JP]; 〒1058001 東京都港区芝浦一丁目 1 番 1 号 株式会社東芝 知的財産部内 Tokyo (JP).
- (74) 代理人: 酒井 宏明, 外 (SAKAI, Hiroaki et al.); 〒1006020 東京都千代田区霞が関三丁目 2 番 5 号 霞が関ビルディング 酒井国際特許事務所 Tokyo (JP).
- (81) 指定国 (表示のない限り、全ての種類の国内保護が可能): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

[続葉有]

(54) Title: PROGRAM

(54) 発明の名称: プログラム

[図4]



104A, 104B PRE-SWITCHING PROCESSING UNIT
106A, 106B POST-SWITCHING PROCESSING UNIT
110 MODULE (#1)
111, 121 INITIALIZATION PROCESSING UNIT
112 PROGRAM (#1) MAIN BODY
118, 128 SWITCHING HISTORY AREA
120 MODULE (#2)
122 PROGRAM (#2) MAIN BODY
181 SECURE SHARED AREA
280 MEMORY

(57) Abstract: At the time of start of execution, a secure shared area for sharing between modules (#1) and (#2) is reserved upon memory, and key exchange is performed. By performing data exchange between modules (#1) and (#2) that have used the secure shared area through encryption/decryption processing using the key, access to the secure shared area from other than the modules (#1) and (#2) is disabled. When module (#1) calls module (#2), immediately before switching of the operation, an identifier indicating the module of the call destination is written to the secure shared area by a pre-switching processing unit. In the called module (#2), a post-switching processing unit compares the identifier that has been read out from the secure shared area with the identifier of the module itself, and if the two identifiers match, starts program execution. If the identifiers do not match, the module (#2) is made to enter into a sleep state.

(57) 要約: 実行開始時に、モジュール (#1) および (#2) 間で共有するセキュア共有領域がメモリ上に確保されると共に、鍵交換が行われる。セキュア共有領域を利用したモジュール (#1) および (#2) 間でのデータのやりとりを、この鍵を用いた暗号化/復号処理を介して行うことで、モジュール (#1) および (#2) 外からのセキュア共有領域に対するアクセスが不能となる。モジュール (#1) がモジュール (#2) を呼び出すと、動作の切替直前に、切替前処理部により呼び出し先のモジュールを示す識別子がセキュア共有領域に書き込まれる。呼び出されたモジュール (#2) において、切替後処理部は、セキュア共有領域から読み出した識別子と自身のモジュールの識別子とを比較し、両者が一致したらプログラムの実行を開始する。識別子が一致しない場合には、モジュール (#2) をスリープ状態とする。

ジュール (#2) を呼び出すと、動作の切替直前に、切替前処理部により呼び出し先のモジュールを示す識別子がセキュア共有領域に書き込まれる。呼び出されたモジュール (#2) において、切替後処理部は、セキュア共有領域から読み出した識別子と自身のモジュールの識別子とを比較し、両者が一致したらプログラムの実行を開始する。識別子が一致しない場合には、モジュール (#2) をスリープ状態とする。

WO 2011/101972 A1



(84) 指定国 (表示のない限り、全ての種類の広域保護が可能): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), ユーラシア (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), ヨーロッパ (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, SM, TR), OAPI (BF, BJ,

CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

添付公開書類:

— 国際調査報告 (条約第 21 条(3))

明 細 書

発明の名称：プログラム

技術分野

[0001] 本発明は、複数のモジュールが連携動作する際の実行順序の変更を防止する技術に関する。

背景技術

[0002] オープンシステムでは、コンピュータのハードウェアやオペレーティングシステム（OS）のソースコードが開示されているので、ユーザがオペレーティングプログラムを改変して所望のプログラムを得ることができる。そのため、オープンシステムでは、OSのプログラムを改変してアプリケーションプログラムを攻撃することが想定される。OSに対してアプリケーションプログラムへの攻撃を防ぐための処理を施すだけでは、このような第三者による攻撃を防ぐことが困難である。

[0003] 一方、ハードウェアは、ユーザによる改変が困難である。OSなどの改変によるプログラムへの攻撃を防ぐように構成したセキュアプロセッサが既に提案されている（特許文献1、非特許文献1）。このようなセキュアプロセッサは、マルチタスク環境でプログラムと当該プログラムで利用される情報とを暗号化することで、プログラムおよび情報の第三者への漏洩やプログラムの改変を防止し、プログラムから生成されたプロセス単体が正しい順序で実行可能となる。

[0004] また、複数モジュールが連携して動作するように構成されたアプリケーションが多く存在する。セキュアプロセッサでは、各モジュールは別のモジュールを一部分しか信用しない状況が考えられる。例えば、各モジュールは別々の鍵で暗号化保護され、別々のコンテキストで動作し、各コンテキストはOSや他のモジュールから隔離される。モジュール間で交換したデータは、潜在的に敵対的なOSや連携しないモジュールに渡さない。このモデルでは、他のモジュールからコンテキストを隔離してモジュール内のプライベート

データを保護する一方で、モジュール間で連携動作するためのデータのやりとりには共有領域を利用する。

[0005] このようなモジュール構成によるアプリケーション形態の一つとして、共有ライブラリを用いる方法が挙げられる。マルチプロセスで実現する場合、各プロセスは互いに独立して動作することから、各プロセス間において動作の同期を取るための記述が必要となる。一方、共有ライブラリは、単体のアプリケーションを作成する場合と同じように、通常の呼び出し規約に従って記述するだけでよい。また、動作も逐次的になるため、簡単に記述できるといった利点がある。

[0006] セキュアプロセッサで共有ライブラリを用いる場合、連携しようとするモジュールが適切かどうかを検証する必要がある。特許文献2では、共有ライブラリを呼び出すプログラムが共有ライブラリの初期化時に認証鍵交換を行って共有ライブラリの正当性を検証している。そして、共有ライブラリを呼び出す際に、共有ライブラリ内の特定のエントリポイントを必ず実行する。また、特許文献3では、呼び出し元となるモジュールと、呼び出し先となるモジュールがそれぞれ正当であるか否かを、そのモジュールを復号するための鍵によって検証するようにしている。

先行技術文献

特許文献

[0007] 特許文献1：特許第4226816号公報

特許文献2：特許第4115759号公報

特許文献3：特開2005-196257号公報

非特許文献

[0008] 非特許文献1：リー他著 “コピー及び耐タンパソフトウェアに対するアーキテクチャ支援”、コンピュータアーキテクチャーニュース、28(5)、p. 168

発明の概要

発明が解決しようとする課題

[0009] ところで、各モジュールが別々にコンテキストを持ち、実行制御はOSが管理することから、モジュールは、OSの実行制御により動作が開始可能となる。このとき、他のモジュールからの呼び出し待ち状態のモジュールでも、当該モジュールは動作を開始してしまう。このように、呼び出し待ち状態のモジュールであっても、OSの実行制御によって動作開始する。

[0010] 上述した特許文献2および3は、呼び出し先のモジュールの正当性は検証できるが、呼び出されたモジュールが動作すべきタイミングであるか否かを判断することができない。したがって、複数モジュールが連携しながら動作する場合に、その実行順序を変更することが可能となり、共有ライブラリのような決められた順番で逐次的に動作することを保証できないという問題点があった。

[0011] 本発明は、上記に鑑みてなされたものであって、第三者による実行順序の変更を、より確実に防止するプログラムを提供することを目的とする。

課題を解決するための手段

[0012] 上述した課題を解決し、目的を達成するために、本発明は、コンピュータに実行させるための複数のモジュールを含むプログラムであって、コンピュータは、連係して動作するモジュールのみがアクセス可能な領域であって、連係して動作するモジュールのうち、OS上で動作するモジュールの識別子を示す実行モジュール識別子を格納する共有領域を含むメモリを備え、モジュールのそれぞれは、連携する他のモジュールへの動作の切替を行う直前に、他のモジュールの識別子を実行モジュール識別子として共有領域に格納する第1処理ステップと、他のモジュールからの動作の切替が行われた直後に、共有領域に格納されている実行モジュール識別子と自モジュールの識別子とが一致した場合に、自モジュール内の機能を実行させる第2処理ステップとを有することを特徴とする。

発明の効果

[0013] 本発明によれば、第三者による実行順序の変更を、効率的にかつ確実に防

止することができる効果がある。

図面の簡単な説明

- [0014] [図1] 第 1 の実施形態に適用可能なシステムの構成の図。
- [図2] 鍵テーブルの一例のデータ構成の図。
- [図3] 複数のモジュールがセキュアプロセッサ上で動作する図。
- [図4] 第 1 の実施形態によるモジュール構成の機能ブロック図。
- [図5] セキュア共有領域が構築されたメモリのメモリマップの図。
- [図6] モジュール間で切り替えを行う処理の全体の流れの図。
- [図7] アプリケーションモジュールが行う初期化処理の図。
- [図8] ライブラリモジュールが行う初期化処理の図。
- [図9] 切り替え前処理を示すフローチャート。
- [図10] 切り替え後処理を示すフローチャート。
- [図11] セキュア共有領域および切替履歴領域の状態の変化の図。
- [図12] 複数モジュールが連携する正規の実行順序の例の図。
- [図13] 実行順序攻撃の第 1 の例の図。
- [図14] 実行順序攻撃の第 2 の例の図。
- [図15] 実行順序攻撃の第 3 の例の図。
- [図16] 実行順序攻撃の第 1 の例に第 1 の実施形態を適用させた例の図。
- [図17] 実行順序攻撃の第 2 の例に第 1 の実施形態を適用させた例の図。
- [図18] 実行順序攻撃の第 3 の例に第 1 の実施形態を適用させた例の図。
- [図19] 発明の第 2 の実施形態に適用可能なシステムの構成の図。
- [図20] モジュール切替管理テーブルのデータ構成の図。
- [図21] 第 2 の実施形態によるモジュール構成の機能ブロック図。
- [図22] モジュール切替管理部の処理を示すフローチャート。
- [図23] アプリケーションモジュールが行う初期化処理を示すフローチャート。
- 。
- [図24] ライブラリモジュールが行う初期化処理を示すフローチャート。
- [図25] 切り替え前処理を示すフローチャート。

- [図26] 切り替え後処理を示す一例のフローチャート。
- [図27] セキュア共有領域および切替履歴領域の状態の変化の図。
- [図28] 仮想マシンモニタを用いた構成の例の図。
- [図29] 言語例外処理機構による動作の図。
- [図30] 発明の第3の実施形態に適用可能なシステムの構成の図。
- [図31] 第3の実施形態によるモジュール構成の機能ブロック図。
- [図32] 第3の実施形態による言語例外に対する処理の図。
- [図33] 第3の実施形態による例外通知処理を示すフローチャート。
- [図34] 第3の実施形態による例外受付処理を示すフローチャート。
- [図35] セキュア共有領域および切替履歴領域の状態の変化の図。
- [図36] 関数 `set jmp / long jmp` による一例の動作の図。
- [図37] 第4の実施形態によるモジュール構成の機能ブロック図。
- [図38] コンテキスト管理テーブルのデータ構成の図。
- [図39] セキュア共有領域および切替履歴領域の状態の変化の図。
- [図40] 第4の実施形態によるコンテキスト登録処理を示すフローチャート。
- [図41] 切り替え後処理を示す一例のフローチャート。
- [図42] コンテキスト設定処理を示すフローチャート。
- [図43] コンテキスト変更通知処理を示すフローチャート。
- [図44] コンテキスト変更受付処理を示すフローチャート。
- [図45] 第5の実施形態によるモジュール構成の機能ブロック図。
- [図46] コンテキスト管理テーブルの一例のデータ構成の図。
- [図47] セキュア共有領域および切替履歴領域の状態の変化の図。
- [図48] コンテキスト設定処理を示すフローチャート。

発明を実施するための形態

[0015] <第1の実施形態>

以下、図面を参照しながら本発明の第1の実施形態について説明する。先ず、本発明の第1の実施形態を適用可能な一例のシステムについて説明する。図1は、本発明の第1の実施形態に適用可能なシステムの一例の構成を模

式的に示す。図 1 では、概略的なハードウェア構成と、システムビューとを同時に表している。

[0016] ターゲットシステムは、ハードウェア的には、セキュアプロセッサ 200 と、メモリ 280 とを備えており、各部はバス 281 で接続されている。セキュアプロセッサ 200 は、コア 210 と暗号復号管理部 220 とを備える。コア 210 は、このセキュアプロセッサ 200 の中核をなし各種プログラムの実行を司る部分であって、現タスク識別子レジスタ 212 を備える。現タスク識別子レジスタ 212 は、コア 210 で現在実行しているモジュールのタスク識別子（現タスク識別子と呼ぶ）を格納する。また、暗号復号管理部 220 は、鍵テーブル 222、セクタ 224 および暗号／復号器 226 を備える、暗号復号管理部 220 は、セキュアプロセッサ 200 外へのアクセスに同期して動作する。セクタ 224 は、コア 210 の制御により、鍵テーブル 222 に格納される各値の、暗号・復号器 226 に対する供給を制御する。

[0017] ターゲットシステムにおいて、セキュアプロセッサ 200 のコア 210 上で単一の OS（オペレーティングシステム）180 が動作し、OS 180 上で 1 または複数のモジュールが動作する。図 1 の例では、複数のモジュール（#1）110、モジュール（#2）120、モジュール（#3）130 およびモジュール（#4）140 が動作している。OS 180 上で動作する各モジュールは、そのモジュールの由来となるプログラムがある。また、各モジュールは、セキュアプロセッサ 200 内のコア 210 において各モジュールを一意に識別するための値である、タスク識別子を持つ。

[0018] 図 1 の例では、モジュール（#1）110 は、タスク識別子が「#1」であって、プログラム A（Prg A）から生成されていることが示されている。モジュール（#2）120 は、タスク識別子が「#2」であって、プログラム B（Prg B）から生成されている。モジュール（#3）130 は、タスク識別子が「#3」であって、プログラム C（Prg C）から生成されている。また、モジュール（#4）140 は、タスク識別子が「#4」で

あって、プログラムBから生成されている。モジュール（＃２）１２０とモジュール（＃４）１４０のように、１つのプログラムから複数のモジュールを生成することができ、由来を同一とする複数のモジュールが異なるタスク識別子を持つことがある。

[0019] 図２は、鍵テーブル２２２の一例のデータ構成を示す。鍵テーブル２２２は、タスク識別子をインデクスとして、インデクス（ＩＤ）が「＃１」から「＃ｎ」までのｎ個のエントリを有する。各エントリは、鍵値フィールド、先頭アドレスフィールドおよび末尾アドレスフィールドを有する。鍵値フィールドは、暗号／復号器２２６で暗号化や復号を行う際に用いる鍵を格納する。先頭アドレスフィールドおよび末尾アドレスフィールドは、鍵値フィールドに格納される鍵で暗号化や復号を行う対象となるデータの、メモリ２８０上での先頭アドレスおよび末尾アドレスをそれぞれ格納する。

[0020] 各モジュール（＃１）１１０～モジュール（＃４）１４０のこの鍵テーブル２２２へのアクセスは、アクセスを行うモジュールのタスク識別子に対応するエントリに限定して行われ、他のタスク識別子に対応するエントリにはアクセスできないように制御される。例えば、モジュール（＃１）１１０は、鍵テーブルのＩＤ「＃１」にのみアクセス可能であって、それ以外のＩＤ「＃２」～「＃ｎ」にはアクセスできないように制御される。

[0021] あるモジュール（＃ｍ）がコア２１０上で実行されているときに、鍵テーブル２２２に対して先頭アドレスＰ、末尾アドレスＱおよび鍵値Ｋの書き込みを行う場合について考える。この場合、セキュアプロセッサ２００は、コア２１０にある現タスク識別子レジスタ２１２を参照してコア２１０上で実行されているモジュール（＃ｍ）のタスク識別子を特定する。そして、特定されたタスク識別子に対応する鍵テーブル２２２内のエントリ＃ｍに対して、鍵値Ｋと先頭アドレスＰおよび末尾アドレスＱを書く。

[0022] 次に、そのモジュール（＃ｍ）が、メモリ２８０上のデータにアクセスする方法について説明する。以下では、鍵テーブル２２２のエントリのうちＩＤ「＃ｍ」に対して、既に先頭アドレスＰ、末尾アドレスＱ、鍵Ｋが登録さ

れており、モジュール（#m）がメモリ280上のアドレスXにあるデータへアクセスする場合を例に説明する。

[0023] モジュール（#m）がメモリ280のアドレスXからデータを読み出すときには、セキュリティプロセッサ200は、メモリ280からアドレスXに対応するデータD₁を取得する。次に、暗号復号管理部220は、現タスク識別子レジスタ212から取り出したタスク識別子（#m）に対応する鍵値などを鍵テーブル222から取得する。暗号復号管理部220は、読み出し先のアドレスXが鍵テーブル222に書かれた先頭アドレスPと末尾アドレスQの間であれば、暗号／復号器226でデータD₁を復号した値をコア210に返す。一方、読み出し先のアドレスXが鍵テーブル222に書かれた先頭アドレスPと末尾アドレスQの間でなければ、データD₁をコア210に返す。

[0024] 同様に、モジュール（#m）がメモリ280にデータD₂を書くとき、セキュアプロセッサ200は、書き出し先アドレスYがアドレスPからアドレスQの間であれば、データD₂を鍵値Kで暗号化したデータをメモリ280上に書く。一方、書き出し先アドレスYがアドレスPからアドレスQの間でなければ、データD₂をメモリ280上に書く。

[0025] ここで、セキュアプロセッサ200上で複数のモジュールが連係動作する場合のモジュール構成について、概略的に説明する。セキュアプロセッサ200上で動作する複数のモジュールは、それぞれ他のモジュールを一部分しか信用しないように構成される。各モジュールは、別々の鍵で暗号化されて保護されて、別々のコンテキストで動作する。各モジュールのコンテキストは、OSや他のモジュールから隔離されると共に、モジュール間で交換したデータは、潜在的に敵対的なOSや連携しないモジュールに渡さない。

[0026] 図3は、セキュアプロセッサ200上で複数のモジュールが動作する場合の、一例のモジュール構成を概念的に示す。図3の例では、モジュール（#1）110およびモジュール（#2）120は、それぞれ別々のスタック（#1）119およびスタック（#2）129などのコンテキストを保持し、

それらのコンテキストは、他のモジュールから直接的に見ることができないようにされる。

[0027] モジュール（＃１）１１０とモジュール（＃２）１２０との間でデータをやり取りするために、共有領域１９０が設けられる。すなわち、セキュアプロセッサ２００上で複数のモジュールが動作する場合には、他のモジュールからコンテキストを隔離してモジュール内のプライベートデータを保護すると共に、モジュール間で連携動作するためのデータのやり取りには共有領域１９０を利用する。

[0028] 図４は、本発明の第１の実施形態によるモジュール構成の一例の機能ブロック図を示す。図４の構成に対し、図１を用いて説明したキュアプロセッサ２００が共通して適用される。なお、図４では、セキュアプロセッサ２００は省略されている。

[0029] 図４において、モジュール（＃１）１１０およびモジュール（＃２）１２０は、それぞれ本第１の実施形態によるモジュール切り替え機構を備える。ここでは、モジュール（＃１）１１０がアプリケーションモジュールであり、モジュール（＃２）１２０がライブラリモジュールであるものとする。モジュール（＃１）１１０およびモジュール（＃２）１２０は、協働して動作する。なお、アプリケーションモジュールは、初期化処理後にプログラム本体を実行するモジュールを指し、ライブラリモジュールは、初期化処理後に他のモジュールからの呼び出しを待つモジュールを指す。

[0030] モジュール（＃１）１１０は、初期化処理部１１１、プログラム（＃１）本体１１２、切替前処理部１０４Ａ、切替後処理部１０６Ａおよび切替履歴領域１１８を備える。同様に、モジュール（＃２）１２０は、初期化処理部１２１、プログラム（＃２）本体１２２および切替履歴領域１２８を備えると共に、それぞれモジュール（＃１）１１０の切替前処理部１０４Ａおよび切替後処理部１０６Ａと共通の処理を行う切替前処理部１０４Ｂおよび切替後処理部１０６Ｂを備える。また、連携するモジュール（＃１）１１０およびモジュール（＃２）１２０間では、メモリ２８０上にセキュア共有領域１

８１を共有する。セキュア共有領域１８１は、モジュール（＃１）１１０とモジュール（＃２）１２０とで共有する共有領域１９０の一部である。

[0031] 初期化処理部１１１および１２１は、切替前処理部１０４Ａおよび１０４Ｂ、ならびに、切替後処理部１０６Ａおよび１０６Ｂで利用するセキュア共有領域１８１を構築するための処理を行う。切替前処理部１０４Ａおよび１０４Ｂは、モジュールが他のモジュールへ切り替える直前の処理を行う。

[0032] 切替後処理部１０６Ａおよび１０６Ｂは、モジュールが他のモジュールから切り替えられた直後の処理を行う。切替履歴領域１１８は、切替前処理１０４Ａおよび切替後処理１０６Ａが切替による復帰に必要とする情報を格納するための領域である。同様に、切替履歴領域１２８は、切替前処理１０４Ｂおよび切替後処理１０６Ｂが切替による復帰に必要とする情報を格納するための領域である。

[0033] セキュア共有領域１８１は、連携するモジュール（＃１）１１０およびモジュール（＃２）１２０の間でのみ共有すべき情報を格納するための領域である。セキュア共有領域１８１は、連携するモジュール以外からの正常な読み書きができないように構成される。

[0034] 図５は、本第１の実施形態によるモジュール切り替え機構において利用する、セキュア共有領域１８１が構築されたメモリ２８０の一例のメモリマップを示す。セキュア共有領域１８１は、連携するモジュール毎にメモリ２８０上に構築され共有される。

[0035] 図５の例では、連携するモジュール（＃１）１１０およびモジュール（＃２）１２０がセキュア共有領域１８１－１（`Sh__mem1`）を共有し、連携するモジュール（＃３）１３０およびモジュール（＃４）１４０がセキュア共有領域１８１－２（`Sh__mem2`）を共有している。セキュア共有領域１８１－１は、メモリ２８０上の開始アドレス`Sh__start1`から終了アドレス`Sh__end1`の範囲とされる。また、セキュア共有領域１８１－２は、メモリ２８０上の開始アドレス`Sh__start2`から終了アドレス`Sh__end2`の範囲とされる。

- [0036] 各セキュア共有領域 181-1、181-2、…は、実行モジュール識別子フィールド 182、切替フラグフィールド 183、切替パラメータフィールド 184 および切替元モジュール識別子フィールド 185 を含み、それぞれ実行モジュール識別子、切替元モジュール識別子、切替フラグおよび切替パラメータが格納される。実行モジュール識別子は、OS 上で動作するモジュールのモジュール識別子である。切替元モジュール識別子は、OS 上で動作するモジュールを実行モジュール識別子が示すモジュールに切り替えた場合に、切り替える前に OS 上で動作していたモジュールのモジュール識別子である。なお、モジュール識別子とは、連携するモジュール間で互いのモジュールを一意に識別するための値である。
- [0037] モジュール識別子として、セキュアプロセッサ 200 で利用するタスク識別子を用いることができる。これに限らず、モジュール識別子として、連携するモジュール間で互いを識別可能に決定した値を用いることも可能である。切替フラグは、モジュールの切り替えの要因がモジュールの呼び出しによるものなのか、モジュールへの復帰によるものなのかを示す値を示す。切替パラメータは、モジュールの呼び出しの場合には呼び出し先の関数名と引数を示し、モジュールへの復帰の場合には実行結果を示す。
- [0038] 次に、本第 1 の実施形態によるモジュール切り替え機能を用いた各モジュールの一例の動作について説明する。ここでは、モジュール識別子としてタスク識別子を用いるものとする。また、モジュール（#1）110 は、モジュール識別子が「#1」で、プログラム A から生成されたアプリケーションモジュールであり、モジュール（#2）120 は、モジュール識別子が「#2」で、プログラム B から生成されたライブラリモジュールであるとする。
- [0039] 以下では、モジュール（#1）110 とモジュール（#2）120 とが連携動作して、モジュール（#1）110 からモジュール（#2）120 を呼び出し、その後、モジュール（#2）120 からモジュール（#1）110 に復帰する例を用いて説明する。
- [0040] 図 6 は、この、モジュール（#1）110 とモジュール（#2）120 と

の間で切り替えを行う処理の全体の流れを示す。OS 180がモジュール（#1）110およびモジュール（#2）120の実行を開始すると、アプリケーションモジュールであるモジュール（#1）110において、初期化処理部111が初期化処理を行う（ステップS10）と共に、ライブラリモジュールであるモジュール（#2）120において、初期化処理部121が初期化処理を行う（ステップS20）。

[0041] 初期化処理が終了すると、モジュール（#2）120は、呼び出しを待機するスリープ状態となる。また、モジュール（#1）110では、プログラム（#1）本体112が実行される（ステップS11）。

[0042] モジュール（#1）110のプログラム（#1）本体112において、モジュール（#2）120の呼び出しを行うときには、切替前処理部104Aでモジュール（#1）110の切り替え前処理を実行し（ステップS12）、モジュール（#1）110からOS 180に対して切り替えを要求する。OS 180は、この切り替え要求を受けて、スリープ状態にあるモジュール（#2）120を呼び出して実行を再開させ、モジュール（#2）120をスリープ状態から復帰させる。一方、モジュール（#1）110は、呼び出しを待機するスリープ状態となる。

[0043] OS 180からの実行再開によりモジュール（#2）120が復帰すると、切替後処理部106Bでモジュール（#2）120の切り替え後処理を実行する（ステップS21）。そして、モジュール（#2）120において、次のステップS22で、呼び出し先すなわちプログラム（#2）本体122が実行される。

[0044] モジュール（#2）120のプログラム（#2）本体122において、モジュール（#1）110の呼び出しを行うときには、切替前処理部104Bでモジュール（#2）120の切り替え前処理を実行し（ステップS23）、モジュール（#2）120からOS 180に対して切り替えを要求する。OS 180は、この切り替え要求を受けて、スリープ状態にあるモジュール（#1）110を呼び出して実行を再開させ、モジュール（#1）110を

スリープ状態から復帰させる。

[0045] OS 180からの呼び出しによりモジュール(#1) 110が復帰されると、切替処理部106Aでモジュール(#1) 110の切り替え後処理を実行する(ステップS13)。そして、モジュール(#1) 110において、プログラム(#1) 本体112の実行が再開される(ステップS14)。

[0046] このように、本第1の実施形態においては、例えばモジュール(#1) 110は、初期化処理部111による初期化処理の後に、プログラム(#1) 本体112の実行が開始される。また、モジュール(#1) 110は、他のモジュールを呼び出す際に、切替前処理部104Aで切り替え前処理を実行する。また、他のモジュールから呼び出された場合は、切替後処理部106Aで切り替え後処理を実行してから、プログラム(#1) 本体112が実行される。

[0047] 以下、図6における各処理について、詳細に説明する。

[0048] 先ず、モジュールの初期化処理について説明する。各モジュール(#1) 110およびモジュール(#2) 120は、OS 180によって実行が開始されると、モジュール間での連携を行うためにそれぞれ初期化処理を実行する。図7は、アプリケーションモジュールが行う初期化処理の一例のフローチャートである。また、図8は、ライブラリモジュールが行う初期化処理の一例のフローチャートである。これら図7に示すアプリケーションモジュールの処理と、図8に示すライブラリモジュールの処理とは、互いに連携して実行される。以下、図7のフローチャートを中心に、必要に応じて図8のフローチャートを参照して説明を行う。

[0049] OS 180により実行が開始されると、図7のステップS111-1および図8のステップS121-1で、連携しようとするモジュール(#1) およびモジュール(#2) の間でAKE (Authentication and Key Exchange) を実施する。AKEの実施により、モジュール(#1) 110およびモジュール(#2) 120は、互いの正当性を検証すると共に、一時鍵Key_ABを共有する。このように、AKE実施

により、連携する各モジュールは、適切な連携対象のプログラムのみが知り、他のモジュールが知ることができない鍵を共有する。

[0050] モジュール（＃１）１１０は、次のステップＳ１１１－２で、メモリ２８０上にセキュア共有領域１８１とするための領域を確保すると共に、モジュール共有鍵`Key__shared`を生成する。モジュール共有鍵は、連携するモジュール間で共有する鍵であり、セキュア共有領域１８１を暗号化するために用いる。このモジュール共有鍵は、連携対象ではないモジュールが知ることができない値であればよく、その値はアプリケーションモジュール（この例ではモジュール（＃１）１１０）が決定してもよいし、乱数から生成してもよい。

[0051] 次のステップＳ１１１－３および図８のステップＳ１２１－２で、モジュール（＃１）１１０およびモジュール（＃２）１２０は、ステップＳ１１１－２でモジュール（＃１）１１０がメモリ２８０上に確保した領域をセキュア共有領域１８１－１とするためのセキュア共有領域情報を交換する。より具体的には、先ず、モジュール（＃１）１１０がセキュア共有領域情報とモジュール（＃１）のモジュール識別子とを一時鍵`Key__AB`で暗号化して、メモリ２８０を経由してモジュール（＃２）１２０に送付する。

[0052] ここで、セキュア共有領域情報は、ステップＳ１１１－２でモジュール（＃１）１１０がメモリ２８０上に確保した領域の先頭アドレス、末尾アドレスおよびモジュール共有鍵を含む。この例では、モジュール（＃１）１１０は、セキュア共有領域情報として、ステップＳ１１１－２で確保したメモリ２８０上の領域の先頭アドレス`Sh__start1`および末尾アドレス`Sh__end1`、ならびに、モジュール共有鍵`Key__shared`をモジュール（＃２）１２０に送付する。それと共に、モジュール（＃１）１１０は、モジュール（＃１）１１０のモジュール識別子（この例ではタスク識別子「＃１」）をモジュール（＃２）１２０に送付する。

[0053] モジュール（＃２）１２０は、モジュール（＃１）１１０から送られてきたデータを一時鍵`Key__AB`で復号し、セキュア共有領域情報（先頭アド

レス `Sh__start 1`、末尾アドレス `Sh__end 1` およびモジュール共有鍵 `Key__shared`）と、モジュール（＃１）１１０のモジュール識別子とを取得する。

[0054] 一方、モジュール（＃２）１２０は、モジュール（＃２）１２０のモジュール識別子を、モジュール（＃２）１２０のタスク識別子から「＃２」とし、このモジュール識別子を一時鍵 `Key__AB` で暗号化する。そして、モジュール識別子が暗号化されたこのデータを、モジュール（＃１）１１０にメモリ ２８０を介して送付する。モジュール（＃１）１１０は、モジュール（＃２）１２０から送られてきたデータを一時鍵 `Key__AB` で復号することで、モジュール（＃２）１２０のモジュール識別子（この場合はタスク識別子の「＃２」）を得ることができる。

[0055] 図 7 のステップ `S 1 1 1-3` および図 8 のステップ `S 1 2 1-2` の処理により、連携するモジュール間で特定のメモリ領域とその特定領域を暗号化するための鍵を共有すると共に、各モジュールに対して連携するモジュール間でそれぞれのモジュールを一意に識別するための値であるモジュール識別子を決することができる。

[0056] モジュール（＃１）１１０は、次のステップ `S 1 1 1-4` で、セキュア共有領域情報を用いてセキュア共有領域 １８１-１に対する暗号化設定を行う。これと並行的に、モジュール（＃２）１２０は、図 8 のステップ `S 1 2 1-3` で、セキュア共有領域情報を用いてセキュア共有領域 １８１-１に対する暗号化設定を行う。具体的には、モジュール（＃１）１１０およびモジュール（＃２）１２０は、セキュアプロセッサ ２００に対して、セキュア共有領域情報に含まれるセキュア共有領域 １８１の先頭アドレス `Sh__start 1`、末尾アドレス `Sh__end 1` および鍵値 `Key__shared` を与えて暗号化設定を行う。

[0057] 各モジュール（＃１）１１０およびモジュール（＃２）１２０それぞれがセキュアプロセッサ ２００上で暗号化設定を行うと、セキュアプロセッサ ２００は、鍵テーブル ２２２を参照し、現タスク識別子に対応するエントリに

対して、先頭アドレス、末尾アドレスおよび鍵値を書く。

[0058] 具体的には、モジュール（＃１）１１０の暗号化設定によって、セキュアプロセッサ２００は、鍵テーブル２２２のモジュール（＃１）１１０に対応するインデックス「＃１」に対して、先頭アドレス`Sh_start1`、末尾アドレス`Sh_end1`および鍵値`Key_shared`を書く。同様に、モジュール（＃２）１２０の暗号化設定によって、セキュアプロセッサ２００は、鍵テーブル２００のモジュール（＃２）１２０に対応するインデックス「＃２」に対して、先頭アドレス`Sh_start1`、末尾アドレス`Sh_end1`および鍵値`Key_shared`を書く。

[0059] これらの設定により、モジュール（＃１）１１０やモジュール（＃２）１２０がメモリ２８０上の領域`Sh_mem1`へ値を書く場合には、セキュアプロセッサ２００は、鍵`Key_shared`で暗号化した値をメモリ領域１８１に書く。また、モジュール（＃１）やモジュール（＃２）がメモリ２８０上の領域`Sh_mem1`から値を読む場合には、セキュアプロセッサ２００は、領域`Sh_mem1`から読み出したデータを鍵`Key_shared`で復号した値をコア２１０に与える。

[0060] モジュール（＃２）１２０は、ステップ`S121-3`でセキュア共有領域情報を鍵テーブル２２２に書き込んだ後、モジュール（＃１）１１０からの呼び出しを待つために、後述する呼び出される側の処理の直前で処理を停止し、スリープ状態となる（図８のステップ`S121-4`）。

[0061] 一方、モジュール（＃１）１１０は、ステップ`S111-4`でセキュア共有領域情報を鍵テーブル２２２に書き込んだ後、ステップ`S111-5`で、自身のモジュール識別子（この場合は、タスク識別子の「＃１」）を、セキュア共有領域１８１-１の実行モジュール識別子フィールド１８２に対して実行モジュール識別子として書く。そして、次のステップ`S111-6`で、モジュール（＃１）１１０におけるプログラム（＃１）本体１１２の実行を開始する。

[0062] このように、本第１の実施形態では、モジュール（＃１）１１０およびモ

ジュール（＃２）１２０が、メモリ２８０上の同一の領域に対して同一の鍵を用いて暗号化設定を行う。これにより、モジュール（＃１）１１０およびモジュール（＃２）１２０は、当該領域から平文の値を読むことができ、ＯＳ１８０や、モジュール（＃１）１１０およびモジュール（＃２）１２０と連携していない他のモジュールからは、当該領域から暗号化された値しか読むことができない。つまり、連携するモジュール（この場合は、モジュール（＃１）１１０およびモジュール（＃２）１２０）は、メモリ２８０上の領域Ｓｈ＿ｍｅｍ１に対して、連携するモジュールだけにその平文の値がアクセス可能な、セキュアな共有領域を設けることができる。

[0063] 次に、モジュールから他のモジュールを呼び出す際の処理について、図９～図１１を用いて説明する。図６を用いて説明したように、モジュール（呼出元モジュール）から他のモジュール（呼び出し先モジュール）を呼び出す際には、切替前処理部１０４Ａまたは１０４Ｂにより呼出元モジュールに対して切り替え前処理が実行され、その後、切替後処理部１０６Ａまたは１０６Ｂにより呼び出し先モジュールに対して切り替え後処理が実行される。

[0064] 以下では、モジュール（＃１）１１０からモジュール（＃２）１２０内のプログラム（＃２）本体１２２における関数ｓｕｂ（５）を呼び出す場合を例に説明する。なお、関数ｓｕｂ（５）において、「ｓｕｂ」は関数名であり、括弧内の数値「５」は、当該関数ｓｕｂ（）に渡す引数である。

[0065] 図９は、切り替え前処理を示す一例のフローチャートである。図１０は、切り替え後処理を示す一例のフローチャートである。切り替え前処理および切り替え後処理は、呼出元モジュールから呼び出し先モジュールを呼び出す呼び出し処理と、呼び出し先モジュールから呼出元モジュールを呼び出して呼出元モジュールを復帰させる復帰処理とで処理内容が異なる。

[0066] 図１１は、呼び出し処理および後述する復帰処理に伴う、セキュア共有領域１８１、ならびに、切替履歴領域１１８および１２８の状態の変化を概略的に示す。なお、図１１において、セキュア共有領域１８１内の「ＩＤ」、「フラグ」および「パラメータ」は、それぞれ実行モジュール識別子、切替

フラグおよび切替パラメータを示す。上述した初期化処理の直後では、セキュア共有領域 181 には、図 7 のステップ S 111-5 の処理で書き込まれた実行モジュール識別子「#1」のみが保存されている。また、切替履歴領域 118 および 128 には、何も保存されていない。

[0067] まず、呼び出し処理について説明する。モジュール（#1）110 において、切替前処理部 104A は、モジュール（#2）120 を呼び出す直前に、モジュール切り替えの種別を判定する（ステップ S 104-1）。ここでは、切り替えの要因が呼び出しによると判定され、処理がステップ S 104-2 に移行される。ステップ S 104-2 では呼び出し先から復帰したときに正しい呼び出し元のアドレスに戻るために、呼び出し元のアドレス *Addr 1* を、モジュール（#1）110 の切替履歴領域 118 に保存する。

[0068] 次のステップ S 104-3 で、切替前処理部 104A は、セキュア共有領域 181 の切替フラグフィールド 183 に対して、切り替えの要因が「呼び出し」であることを示す切替フラグ「*call*」を書く。

[0069] 次のステップ S 104-4 で、切替前処理部 104A は、切替パラメータフィールド 184 に対して、モジュール切り替えに必要な呼び出し情報として、呼び出し先の関数名「*sub*」とその引数「5」を書く。さらに、次のステップ S 104-5 で、切替前処理部 104A は、切替元モジュール識別子フィールド 185 に対して、自身のモジュール識別子「#1」を書く。

[0070] 処理は呼び出し処理および復帰処理で共通のステップ S 104-9 に移行され、切替前処理部 104A は、セキュア共有領域 181 にある実行モジュール識別子フィールド 182 に対して、呼び出し先のモジュール（この例ではモジュール（#2）120）のモジュール識別子「#2」を書く。そして、処理はステップ S 104-10 に移行され、スリープ状態となる。これは、呼び出し先から復帰する場合や、他のモジュールなどからの呼び出しに備えるためであり、モジュール（#1）110 は、自身をスリープ状態として、切り替え後処理の直前で処理を停止する。

[0071] なお、実行モジュール識別子フィールド 182 に対して実行モジュール識

別子を書くことによって、モジュールの実行権が実行モジュール識別子で示されるモジュールに移ったと考えることができる。そのため、ステップS 104-9による実行モジュール識別子の書き込みから、ステップS 104-10によるモジュールの処理停止までの期間は、可能な限り短くするのが好ましいと共に、その期間で当該モジュールの情報保護資産に関わる処理を行うことは、避けるのが望ましい。

[0072] ステップS 104-10によってモジュール（# 1）110がスリープ状態になると、実行権がOS 180に切り替わる。その後、OS 180のスケジューラなどの制御によって実行権がモジュール（# 2）120に移り、モジュール（# 2）120が実行を再開する。

[0073] モジュールの実行権が切り替わると、図10に示す手順に従い切り替え後処理が実行される。実行されるモジュールがモジュール（# 1）110からモジュール（# 2）120に切り替わると、モジュール（# 2）120の切替後処理部106Bは、セキュア共有領域181から実行モジュール識別子を得る（ステップS 106-1）。そして、獲得した実行モジュール識別子の値が、モジュール（# 2）120のモジュール識別子の値と一致するか否かを判定する（ステップS 106-2）。

[0074] 若し、ステップS 106-2で、実行モジュール識別子の値とモジュール（# 2）120のモジュール識別子の値とが一致しないと判定されたら、処理はステップS 106-9に移行されてスリープ状態とされ、モジュール（# 2）120の状態が呼び出し処理の直前の状態に戻る。一方、ステップS 106-2で、実行モジュール識別子の値とモジュール（# 2）120のモジュール識別子の値とが一致すると判定されたら、処理はステップS 106-3に移行される。

[0075] この例では、セキュア共有領域181の実行モジュール識別子フィールド182に、モジュール識別子「# 2」が書き込まれているため、モジュール（# 2）120のモジュール識別子と値が一致すると判定される。したがって、切替後処理部106Bは、モジュール（# 2）120を実行すべきであ

ることが分かるので、ステップS 106-3に処理を移行させて、切り替え後処理を継続する。

[0076] ステップS 106-3で、切替後処理部106Bは、セキュア共有領域181の切替フラグフィールド183から切替フラグを得る。次のステップS 106-4で、獲得した切替フラグの値が「呼び出し」および「復帰」の何れを示すかが判定される。若し、切替フラグの値が「呼び出し」を示すと判定されたら、処理はステップS 106-5に移行される。一方、切替フラグの値が「復帰」を示すと判定されたら、処理はステップS 106-7に移行される。

[0077] この例では、上述したように図9のステップS 104-3で「呼び出し」を示す切替フラグ「c a l l」が切替フラグフィールド183に書き込まれているので、切替フラグの値が「呼び出し」を示すと判定され、関数呼び出しが要求されていることが分かる。したがって、処理がステップS 106-5に移行される。

[0078] ステップS 106-5で、切替後処理部106Bは、セキュア共有領域181の切替元モジュール識別子フィールド185から切替元モジュール識別子を取り出して、モジュール(#2)120の切替履歴領域128に保存する。このように、切替元のモジュールを示すモジュール識別子を予め保存しておくことで、モジュール(#2)120は、関数実行後に呼び出し元のモジュールに復帰することができる。

[0079] この例では、上述したように図9のステップS 104-5でモジュール識別子「#1」が切替元モジュール識別子フィールド185に書き込まれているので、このモジュール識別子「#1」が切替元モジュール識別子として取り出され、切替履歴領域185に保存される。

[0080] 次のステップS 106-6で、切替後処理部106Bは、セキュア共有領域181の切替パラメータフィールド184から、呼び出し情報として書き込まれた関数名「s u b」と、引数「5」とを得る。そして、これら関数名「s u b」および引数「5」に従い、プログラム(#2)本体122にお

る関数 `sub` (5) の実行を開始する。

[0081] 図 11 に例示されるように、呼び出し処理が終了した時点では、セキュア共有領域 181 には、切り替え前処理すなわち図 9 のステップ S104-3、ステップ S104-4 およびステップ S104-9 で書き込まれた実行モジュール識別子「#2」、切替フラグ「`call`」、ならびに、切替パラメータ「`sub`」および「5」が保存されている。また、モジュール (#1) 110 の切替履歴領域 118 には、ステップ S104-2 で書き込まれた呼び出し元のアドレス `Addr1` が保存されている。モジュール (#2) 120 の切替履歴領域 128 には、切り替え後処理すなわち図 10 のステップ S106-5 で書き込まれたモジュール識別子「#1」が保存されている。

[0082] 次に、復帰処理について説明する。この例では、実行が呼び出し先のモジュール (#2) 120 から、当該モジュール (#2) 120 を呼び出した呼出元のモジュール (#1) 110 に復帰する場合の処理となる。

[0083] モジュール (#2) 120 の切替前処理部 104B は、モジュール (#2) 120 から呼び出し元のモジュールに復帰する直前に、ステップ S104-1 で、モジュール切り替えの種別を判定する。ここでは、切り替えの要因が復帰によるものであると判定され、処理がステップ S104-6 に移行される。

[0084] ステップ S104-6 では、切替前処理部 104B は、切替履歴領域 128 から、呼び出された際 (図 9 のステップ S106-5) に保存しておいた切替元のモジュールのモジュール識別子「#1」を得る。次のステップ S104-7 で、切替前処理部 104B は、セキュア共有領域 181 の切替フラグフィールド 183 に対して、切り替え要因が「復帰」であることを示す切替フラグ「`ret`」を書く。さらに、次のステップ S104-8 で、切替前処理部 104B は、セキュア共有領域 181 の切替パラメータフィールド 184 に対して、復帰処理に必要な呼び出し復帰情報として、切替前処理部 104B の実行の結果である実行結果 `Result_1` を書く。

[0085] 処理はステップ S104-9 に移行され、切替前処理部 104B は、セキ

セキュア共有領域 181にある実行モジュール識別子フィールド 182に対して、呼出元のモジュール（この例ではモジュール（#1）110）のモジュール識別子「#1」を書く。そして、処理はステップ S104-10に移行され、スリープ状態となり、切り替え後処理の直前で、処理を停止する。

[0086] ステップ S104-10によってモジュール（#2）120がスリープ状態になると、実行権が OS 180に切り替わる。その後、OS 180のスケジューラなどによって実行権がモジュール（#1）110に移り、モジュール（#1）110が実行を再開する。

[0087] モジュールの実行権が切り替わると、図 10に示す手順に従い切り替え後処理が実行される。実行されるモジュールがモジュール（#2）120からモジュール（#1）110に切り替わると、モジュール（#1）110の切替後処理部 106Aは、セキュア共有領域 181の実行モジュール識別子フィールド 182からモジュール識別子「#1」を取り出し（ステップ S106-1）、取り出した値がモジュール（#1）110のモジュール識別子と一致するため、切り替え後処理を継続する（ステップ S106-2）。

[0088] ステップ S106-3で、切替後処理部 106Aは、セキュア共有領域 181の切替フラグフィールド 183から切替フラグを得る。この例では、上述した図 9のステップ S104-7で切替フラグフィールド 183に書かれた切替フラグ「ret」が得られるため、切替フラグの値が「復帰」を示すと判定し（ステップ S106-4）、関数呼び出し復帰が要求されていることが分かる。したがって、処理がステップ S106-7に移行する。

[0089] ステップ S106-7で、切替後処理部 106Aは、切替履歴領域 118から、上述のステップ S104-2で保存しておいた呼び出し元のアドレス Addr1を得ると共に、切替履歴領域 118に保存されている当該アドレス Addr1を削除する。次のステップ S106-8で、切替後処理部 106Aは、セキュア共有領域 181の切替パラメータフィールド 184から呼び出し復帰情報を得る。そして、モジュール（#1）110は、獲得した呼び出し復帰情報に含まれる実行結果 Result_1を用いて、呼び出し元

アドレス `Addr1` に戻って実行を再開する。

[0090] 図 11 に例示されるように、復帰処理が終了した時点では、セキュア共有領域 181 には、切り替え前処理すなわち図 9 のステップ S104-7、ステップ S104-8 およびステップ S104-9 で書き込まれた切替フラグ「`ret`」、切替パラメータ「`Result_1`」および実行モジュール識別子「`#1`」が保存されている。また、モジュール（`#1`）110 の切替履歴領域 118 は、切り替え後処理すなわち図 10 のステップ S106-7 で呼び出し元のアドレス `Addr1` が削除され、何も保存されていない。同様に、モジュール（`#2`）120 の切替履歴領域 128 は、切り替え前処理すなわちステップ S104-6 で呼び出し元のモジュール識別子が削除され、何も保存されていない。

[0091] このように、連携するモジュール間でのみアクセス可能な領域を設け、実行が呼び出し元のモジュールが他のモジュールに切り替わる直前に、当該領域に呼び出し先のモジュールのモジュール識別子を書く。そして、呼び出し先のモジュールは、実行が呼出元モジュールから呼び出し先モジュールに切り替えられた直後に、当該領域に書かれた呼び出し先モジュールのモジュール識別子の値と自身のモジュール識別子とを比較することにより、実行順序の妥当性を保証することができる。この実行順序の妥当性が保証できることを示す具体的な例については、後述する。

[0092] 図 12～図 18 を用いて、本第 1 の実施形態がモジュールの実行順序の妥当性を保証できることについて説明する。ここでは、プログラム X（`Prog X`）によるアプリケーションモジュールであるモジュール `#1` と、それぞれライブラリモジュールであるプログラム Y（`Prog Y`）によるモジュール `#2` およびプログラム Z（`Prog Z`）によるモジュール `#3` の 3 のモジュールが連携して実行されるものとする。このとき、モジュール `#1` は、モジュール `#2` を呼び出し、モジュール `#2` は、モジュール `#3` を呼び出すように構成されているものとする。

[0093] 図 12 は、正規の実行順序の例を示す。モジュール `#1`～`#3` が実行開始

されると、モジュール# 2 および# 3 がスリープ状態とされ、タイミング A で実行するモジュールが切り替えられ、モジュール# 1 からモジュール# 2 が呼び出されると共に、モジュール# 1 がスリープ状態とされる。タイミング B で実行するモジュールが切り替えられ、モジュール# 2 からモジュール# 3 が呼び出されると共にモジュール# 2 がスリープ状態とされる。次に、タイミング C で実行するモジュールが切り替えられ、モジュール# 3 からモジュール# 2 が復帰されると共に、モジュール# 3 がスリープ状態とされる。

[0094] 以下、図 1 2 に示した実行順序に対してなされる実行順序攻撃の例について、図 1 3 ~ 図 1 5 を用いて説明する。これらの実行順序攻撃は、通常、OS のプログラムを利用や改変してモジュールの実行順序などを変更することで行われる。

[0095] 図 1 3 は、実行順序攻撃の第 1 の例を示す。第 1 の例では、スリープ状態のモジュールを、当該モジュールが正規の呼び出し元のモジュールから呼び出される前に、実行開始させる。図 1 3 では、OS のスケジューラなどを利用し、タイミング A でモジュール# 1 から呼び出されるべきモジュール# 2 が、タイミング A より前の、タイミング D の時点で実行開始させる攻撃を示している。この場合、モジュール# 1 がタイミング A でモジュール# 2 を呼び出した時点で、既に、モジュール# 2 の先頭部分の実行が終了していることになる。一例として、モジュール# 2 の先頭部分に認証処理を含む場合、この認証処理を介さずにモジュール# 2 の残りの部分を実行できる。この第 1 の例を、タイミング攻撃と呼ぶ。

[0096] 図 1 4 は、実行順序攻撃の第 2 の例を示す。第 2 の例では、モジュールの呼び出し順序を正規の順序に対して変更する。正規には、図 1 2 に例示するようにモジュール# 1 からモジュール# 2 が呼び出され、さらにモジュール# 2 からモジュール# 3 が呼び出される。これに対し、図 1 4 では、OS のスケジューラなどを利用し、モジュール# 1 から呼び出すタイミング A で、モジュール# 2 ではなく、モジュール# 3 を実行開始させる攻撃を示してい

る。一例として、モジュール# 2での認証処理で認証された後に、モジュール# 3が実行可能となるような場合に、モジュール# 2での認証処理を介さずにモジュール# 3が実行されてしまうことになる。この第2の例を、呼び出し順序攻撃と呼ぶ。

[0097] 図15は、実行順序攻撃の第3の例を示す。第3の例では、OSが予めモジュール# 1とは別のモジュール# 5とモジュール# 2とで実行を開始する。そして、モジュール# 5からモジュール# 2を呼び出して、例えば先頭部分のみを実行させて途中状態でモジュール# 5に復帰させる。モジュール# 2は、スリープ状態となる。このとき、OSは、モジュール# 2の途中状態を保存しておき、その後、タイミングAでモジュール# 1から途中状態のモジュール# 2を呼び出すようにすることができる。この場合においても、モジュール# 2の先頭部分に認証処理を含む場合、この認証処理を介さずにモジュール# 2の残り部分を実行できる。この第3の例を、途中状態モジュール挿入攻撃と呼ぶ。

[0098] 図16～図18を用いて、本第1の実施形態による構成で上述した実行順序攻撃の第1～第3の例を回避できるか否かを検証する。なお、図16～図18において、共有領域内CurIDは、セキュア共有領域181に保存される実行モジュール識別子を示す。

[0099] 図16は、上述の図13で示した第1の例すなわちタイミング攻撃に本第1の実施形態を適用させた例である。この場合、タイミングAでモジュール# 1からモジュール# 2が呼び出されるまで、セキュア共有領域181には、実行モジュール識別子としてモジュール# 1を示す「# 1」が保存されている。一方、モジュール# 2をスリープ状態から実行再開させると、モジュール# 2は、上述した切り替え後処理により、図10のステップS106-2で自身のモジュール識別子とセキュア共有領域181に保存されている実行モジュール識別子とが一致するか否かを判定する。

[0100] 図13に例示したタイミングDでモジュール# 2を再開させた場合、セキュア共有領域181に保存される実行モジュール識別子は、値が「# 1」で

あり、モジュール# 2 自身のモジュール識別子は値が「# 2」であり、両者が一致しない。そのため、処理がステップS 1 0 6－9に移行されて、モジュール# 3 がスリープ状態とされ、タイミング攻撃が回避される。

[0101] 図 1 7 は、上述の図 1 4 で示した第 2 の例すなわち呼び出し順序攻撃に本第 1 の実施形態を適用させた例である。この場合、モジュール# 1 から呼び出されたモジュール# 3 のモジュール識別子の値が「# 3」である。これに対し、セキュア共有領域 1 8 1 には、切り替え前処理により、図 9 のステップS 1 0 4－9 で次に実行すべき実行モジュール識別子として値「# 2」が保存されている。したがって、図 1 0 を用いて上述した切り替え後処理のステップS 1 0 6－2 で、自身のモジュール識別子と、セキュア共有領域 1 8 1 に保存される実行モジュール識別子とが一致しないと判定される。したがって、処理がステップS 1 0 6－9に移行されて、モジュール# 2 がスリープ状態とされ、呼び出し順序攻撃が回避される。

[0102] 図 1 8 は、上述の図 1 5 で示した第 3 の例すなわち途中状態モジュール挿入攻撃に本第 1 の実施形態を適用させた例である。この場合、モジュール# 2 およびモジュール# 5 は、モジュール# 1 の実行とは別系統で初期化されており、モジュール共有鍵Key__Shared__1を共有する。一方、モジュール# 1 は、初期化によってモジュール共有鍵Key__Shared__2を持つ。そのため、モジュール# 2 およびモジュール# 5 の間で共有されるモジュール共有鍵Key__Shared__1は、モジュール# 1 とは共有されないことになる。したがって、モジュール# 1 がモジュール# 2 を呼び出す際に、モジュール# 2 は、セキュア共有領域 1 8 1 に保存された情報を、平文ではない状態で読むことになる。

[0103] このように、モジュール# 1 とモジュール# 2 は、セキュア共有領域 1 8 1 に書かれた情報を共有することができないため、モジュール間で連携することができない。すなわち、途中状態のモジュールを挿入することを回避できる。

[0104] このように、本第 1 の実施形態では、モジュールが実行再開された際に、

切り替え後処理により、自身のモジュール識別子と、セキュア共有領域 181 に保存される、実行されるべきモジュールを示す実行モジュール識別子とが一致しているか否かを判定することにより、実行順序攻撃を回避できる。

[0105] なお、本第 1 の実施形態は、本発明の主旨を逸脱しない範囲で、上述以外にも様々な変形が可能である。例えば、上述では、セキュア共有領域 181 として、セキュアプロセッサ 200 とバス 281 で接続されるメモリ 280 を利用したが、これは適用可能な記憶装置の一例を示したに過ぎない。例えば、セキュア共有領域 181 として、セキュアプロセッサ 200 が内蔵するキャッシュメモリを用いてもよいし、フラッシュメモリといった不揮発性のメモリを用いてもよい。さらに、セキュア共有領域 181 は、半導体メモリに限られず、例えばハードディスクを用いてもよい。

[0106] 同様に、上述では、初期化処理における AKE やセキュア共有領域情報などの交換をメモリ 280 を用いて行うように説明したが、これは適用可能な記憶装置の一例を示したに過ぎない。すなわち、AKE やセキュア共有領域情報などの交換を、キャッシュメモリ、フラッシュメモリ、ハードディスクなどを介して行ってもよい。

[0107] 上述では、セキュアプロセッサ 200 がコア 210 と暗号復号管理部 220 のみを含むように説明したが、これはこの例に限定されない。例えば、セキュアプロセッサ 200 は、内部メモリ、DMA (Direct Memory Access) コントローラなどを含んでもよい。さらに、上述では、暗号復号管理部 220 内の鍵テーブル 222 を動的に設定していたが、これはこの例に限定されず、予め作成され、セキュアプロセッサ 200 内の不揮発性の記憶領域に埋め込まれた鍵であってもよい。

[0108] 上述の第 1 の実施形態では、AKE をアプリケーションモジュール側から開始したが、AKE をライブラリモジュール側から開始してもよい。

[0109] 上述の第 1 の実施形態では、各モジュールの実行開始時に初期化処理を実行するように説明したが、これはこの例に限定されない。すなわち、本発明では、初期化処理をプログラムの実行開始時に限定するものではない。例え

ば、アプリケーションモジュールとライブラリモジュールとが連携する直前に初期化処理を実行してもよい。また、初期化処理は、モジュールが連携する直前までであれば、どの時点で実行してもよい。

[0110] 上述の第1の実施形態では、アプリケーションモジュールと連携するライブラリモジュールの個数を2乃至3に限定して説明したが、これはこの例に限られない。すなわち、4以上のライブラリモジュールがアプリケーションモジュールと連携する場合にも、第1の実施形態を適用することができる。

[0111] すなわち、本題1の実施形態は、アプリケーションモジュールが複数のライブラリモジュールと連携する場合、ライブラリモジュールから他のライブラリモジュールを呼び出す場合、ならびに、ライブラリモジュールからアプリケーションモジュールを呼び出す場合などにも適用可能である。これらの何れの場合でも、全てのモジュールで同じ鍵を共有し、その鍵を用いてセキュア共有領域を構築することにより、4以上のモジュールがセキュア共有領域を共有することが可能になる。

[0112] 上述の第1の実施形態では、1のアプリケーションモジュールに対して1つのセキュア共有領域を設けたが、1のアプリケーションモジュールが複数のセキュア共有領域を利用してもよい。例えば、アプリケーションモジュールと連携するあるモジュールとはセキュア共有領域Sh__mem3を共有し、連携する別のモジュールとはセキュア共有領域Sh__mem4を共有する。また、モジュール#1、#2および#3が連携する場合に、モジュール#1および#2間、モジュール#2および#3間、ならびに、モジュール#1および#3間でそれぞれ1つずつセキュア共有領域を確保してもよい。

[0113] 上述の第1の実施形態では、暗号復号管理部220が独立したユニットとして構成されているように説明したが、これはこの例に限定されない。例えば、BIU(Bus Interface Unit)やDMAコントローラなどの、プロセッサ外とのアクセスを行うユニットに内包してもよい。

[0114] 上述の第1の実施形態では、セキュアプロセッサ200のデータ暗号化機構を用いて、連携するモジュール間でのみアクセス可能なセキュア共有領域

１８１を構築したが、これはこの例に限定されない。すなわち、データ暗号化機構に限らず、セキュアプロセッサ２００のサポートによって連携するモジュール間でのみアクセス可能な領域が構築可能であれば、他の方法を用いてセキュア共有領域１８１を構築してもよい。

[0115] 上述の第１の実施形態では、ＯＳ１８０のサポートを得ることによってモジュールの切り替えを行っているが、これはこの例に限定されず、例えばハードウェア機構を用いてモジュールを直接的に切り替えるようにした場合（特許文献１参照）にも、本発明を適用することができる。

[0116] 上述の第１の実施形態では、切替履歴領域１１８および１２８を、独立したデータ領域として管理しているが、これはこの例に限定されない。すなわち、切替履歴領域１１８および１２８は、必ずしも独立した構成である必要はなく、各モジュールに含まれるスタックに包含するように構成してもよい。

[0117] 上述の第１の実施形態では、モジュール識別子としてタスク識別子を利用したが、これはこの例に限られず、モジュール識別子は、連携するモジュール間でユニークな値であれば、他の値を用いてもよい。例えば、各モジュールは、それぞれモジュール識別子を乱数によって生成してもよい。また、セキュアプロセッサ２００が複数のプロセッサコアを持ち、各プロセッサコアに対してユニークな識別子が付与されている場合も考えられる。この場合、各モジュールは、プロセッサコアを識別するコア識別子とタスク識別子とを組み合わせた値を、モジュール識別子として用いることができる。これに限らず、連携するモジュール間での取り決めによって、何方か一方のモジュールだけがモジュール識別子としてタスク識別子を用い、他方のモジュールは、モジュール識別子として乱数から得た値を用いる方法でもよい。

[0118] <第２の実施形態>

次に、本発明の第２の実施形態について説明する。上述の第１の実施形態では、連携するモジュールでのみ読み書き可能にするセキュア共有領域１８１を、メモリ２８０上の一部の領域を用いて構成している。これに対し、本

第2の実施形態では、セキュア共有領域を、セキュアプロセッサ上にハードウェアとして持つようにする。

[0119] 図19は、本第2の実施形態に係るシステムの一例の構成を模式的に示す。図19では、概略的なハードウェア構成と、システムビューとが同時に表されている。なお、図19において、上述した図1と共通する部分には同一の符号を付し、詳細な説明を省略する。

[0120] 図19において、本第2の実施形態によるセキュアプロセッサ400は、コア410、BIU430、モジュール切替管理部420を備える。モジュール切替管理部420は、モジュール切替管理テーブル422とテーブルアクセス制御部424とを備える。

[0121] テーブルアクセス制御部424は、コア410からモジュール切替管理テーブル422に対するアクセスを許可するか否かを判定し、許可しない場合は、例外などを発行する。BIU430は、外部のバス281に接続され、セキュアプロセッサ400の内部からバス281に接続するためのバスインターフェイスユニットである。コア410は、BIU430を介してバス281に接続され、例えばメモリ280にアクセスすることができる。

[0122] なお、この第2の実施形態によるセキュアプロセッサ400は、上述した第1の実施形態によるセキュアプロセッサ200と異なり、鍵テーブルや暗号／復号器を持つ暗号／復号管理部を含まない。

[0123] 図20は、モジュール切替管理テーブル422の一例のデータ構成を示す。モジュール切替管理テーブル422は、共有領域番号を示すインデクス（Idx）が「#1」から「#n」までのn個のエントリを有する。各エントリは、モジュール共有鍵フィールド、実行モジュール識別子フィールド、切替元モジュール識別子フィールド、切替フラグフィールド、切替パラメータフィールドおよび有効ビットを有する。なお、有効ビットは、図20においては「V」として表されている。これらのうち、実行モジュール識別子フィールド、切替元モジュール識別子フィールド、切替フラグフィールドおよび切替パラメータフィールドは、図5を用いて説明したセキュア共有領域18

1における名称が対応するフィールドと共通である。

[0124] モジュール共有鍵フィールドは、当該フィールドにアクセス可能なモジュールのみが知り得る値を格納する。有効ビットは、そのエントリが有効か無効かを示すフラグである。

[0125] 図21は、本第2の実施形態によるモジュール構成の一例の機能ブロック図を示す。なお、図21において、上述した図4と共通する部分には同一の符号を付し、詳細な説明を省略する。

[0126] 図21において、アプリケーションモジュールであるモジュール(#1)310は、初期化処理部311、プログラム(#1)本体112、切替前処理部304A、切替後処理部306Aおよび切替履歴領域118を備える。また、ライブラリモジュールであるモジュール(#2)320は、初期化処理部321、プログラム(#2)本体122および切替履歴領域128を備えると共に、それぞれモジュール(#1)310の切替前処理部304Aおよび切替後処理部306Aと共通の処理を行う切替前処理部304Bおよび切替後処理部306Bを備える。

[0127] なお、本第2の実施形態においては、上述した第1の実施形態とは異なり、メモリ280上には、モジュール(#1)310およびモジュール(#2)320とで共有するセキュア共有領域を設けない。

[0128] 次に、本第2の実施形態によるモジュール切り替え機能を用いた各モジュールの一例の動作について説明する。ここでは、モジュール識別子としてタスク識別子を用いるものとする。また、モジュール(#1)310は、モジュール識別子が「#1」で、プログラムAから生成されたアプリケーションモジュールであり、モジュール(#2)320は、モジュール識別子が「#2」で、プログラムBから生成されたライブラリモジュールであるとする。

[0129] 以下では、モジュール(#1)110とモジュール(#2)120とが連携動作して、モジュール(#1)110からモジュール(#2)120を呼び出し、その後、モジュール(#2)120からモジュール(#1)110に復帰する例を用いて説明する。

- [0130] ここで、例えばモジュール（＃１）３１０において、初期化処理部３１１、切替前処理部３０４Ａおよび切替後処理部３０６Ａによる処理は、上述した第１の実施形態による初期化処理部１１１、切替前処理部１０４Ａおよび切替後処理部１０６Ａによる処理に対して、初期化処理で共有した共有領域番号とモジュール共有鍵とを用いてモジュール切替管理テーブルへのアクセスを行う点が異なる。これは、モジュール（＃２）３２０でも同様である。
- [0131] 図２２は、モジュール切替管理部４２４の一例の処理を示すフローチャートである。モジュール切替管理部４２４は、モジュール切替管理テーブル４２２に対してアクセスされたときに、この図２２のフローチャートに従い処理を行う。
- [0132] 各モジュール（＃１）３１０およびモジュール（＃２）３２０がモジュール切替管理テーブル４２２上のフィールドにアクセスする場合には、コア４１０がアクセス先の共有領域番号とモジュール共有鍵とを指定する。例えばモジュール（＃１）３１０からモジュール切替管理テーブル４２２に対してアクセスがなされた場合、まず、コア４１０からモジュール切替管理部４２０内のテーブルアクセス制御部４２４に対して、アクセス先の共有領域番号とモジュール共有鍵とが指定される。
- [0133] ステップＳ４２４－１で、テーブルアクセス制御部４２４は、モジュール切替管理テーブル４２２を参照して、指定された共有領域番号に対応する有効ビットの値を取得する。次のステップＳ４２４－２で、テーブルアクセス制御部４２４は、取得した有効ビットが「無効を示す値」であるか否かを判定する（ステップＳ４２４－２）。
- [0134] 若し、取得した有効ビットが「無効を示す値」だと判定されたら、処理はステップＳ４２４－３に移行され、テーブルアクセス制御部４２４は、コア４１０から指定されたアクセスの要求が書き込み要求か否かを判定する。判定の結果、若し、アクセス要求が書き込み要求以外の要求だと判定されたら、処理はステップＳ４２４－６に移行され、テーブルアクセス制御部４２４は、例外を発行する。

- [0135] 一方、ステップS 4 2 4－3で、アクセス要求が書き込み要求であると判定されたら、処理はステップS 4 2 4－4に移行される。ステップS 4 2 4－4で、テーブルアクセス制御部4 2 4は、指定された共有領域番号に対応する有効ビットを「有効を示す値」に書き換える。そして、次のステップS 4 2 4－5で、指定されたモジュール切替管理テーブル4 2 2の指定された共有領域番号に対して、指定された書き込みを実行する。
- [0136] 上述したステップS 4 2 4－2により、有効ビットが「無効を示す値」以外であると判定された場合、処理はステップS 4 2 4－7に移行される。ステップS 4 2 4－7では、テーブルアクセス制御部4 2 4は、モジュール切替管理テーブル4 2 2から、指定された共有領域番号に対応するモジュール共有鍵を得る。
- [0137] 次のステップS 4 2 4－8で、テーブルアクセス制御部4 2 4は、獲得したモジュール共有鍵の値が、各モジュール（＃１）３１０またはモジュール（＃２）３２０から与えられたモジュール共有鍵と一致するか否かを判定する。若し、一致すると判定されたら、処理はステップS 4 2 4－9に移行され、テーブルアクセス制御部4 2 4は、モジュール切替管理テーブル4 2 2に対して指定された読み書きを実行する（S 4 2 4－9）。一方、一致しないと判定されたら、処理はステップS 4 2 4－6に移行され、テーブルアクセス制御部4 2 4は例外を発行する。
- [0138] 次に、本第２の実施形態によるモジュールの初期化処理について説明する。各モジュール（＃１）３１０およびモジュール（＃２）３２０は、OS 180によって実行が開始されると、モジュール間での連携を行うためにそれぞれ初期化処理を実行する。図２３は、アプリケーションモジュールであるモジュール（＃１）３１０が行う初期化処理を示す一例のフローチャートである。また、図２４は、ライブラリモジュールであるモジュール（＃２）３２０が行う初期化処理を示す一例のフローチャートである。これら図２３に示すアプリケーションモジュールの処理と、図２４に示すライブラリモジュールの処理とは、互いに連携して実行される。

- [0139] 以下、図 23 のフローチャートを中心に、必要に応じて図 24 のフローチャートを参照して説明を行う。なお、図 23 および図 24 のフローチャートにおいて、上述した図 7 および図 8 のフローチャートと共通する処理には同一の符号を付し、詳細な説明を省略する。
- [0140] OS 180 により実行が開始されると、図 23 のステップ S 111-1 および図 24 のステップ S 121-1 で、モジュール（#1）310 およびモジュール（#2）320 のそれぞれにおいて、互いの間で AKE を実施する。これにより、モジュール（#1）110 およびモジュール（#2）120 は、互いのプログラムの正当性を検証し、一時鍵 Key_AB を共有する。
- [0141] 図 23 において、次のステップ S 311-2 で、モジュール（#1）310 の初期化処理部 311 は、モジュール切替管理テーブル 422 内の特定のエントリ（以下、共有領域と呼ぶ）へのアクセスを許可するためのモジュール共有鍵 Key_shared を、乱数などから生成する。
- [0142] 次のステップ S 311-3 で、初期化処理部 311 は、特定の共有領域番号（例えば「#1」）に対して、モジュール共有鍵 Key_shared と自身のモジュールを示すモジュール識別子（値が「#1」とする）とを指定する。ここで、共有領域番号とは、モジュール間で共有する、モジュール切替管理テーブル 422 内のエントリのインデックスを指す。これにより、モジュール切替管理テーブル 422 の空き領域を確保され、モジュール（#2）320 との間で共有する共有領域がモジュール切替管理テーブル 422 上に確保される。
- [0143] このステップ S 311-3 において、モジュール切替管理テーブル 422 に対するアクセスがなされるため、モジュール切替管理部 420 は、上述した図 22 のフローチャートに従い当該アクセスの制御を行う。この場合、指定された共有領域番号に対する有効ビットが「無効を示す値」であり、かつ、書き込み要求である。したがって、モジュール切替管理部 420 は、指定されたモジュール共有鍵とモジュール識別子とを、それぞれモジュール切替管理テーブル 422 におけるモジュール共有鍵フィールドと実行モジュール

識別子フィールドに書くと共に、有効ビットを「有効を示す値」にセットする。

[0144] 次のステップS 3 1 1－4で、モジュール（＃ 1） 3 1 0は、セキュア共有領域情報とモジュール識別子とを一時鍵K e y__A Bで暗号化して、メモリ2 8 0を経由してモジュール（＃ 2） 3 2 0に送付する。ここで、セキュア共有領域情報は、ステップS 3 1 1－3で確保した共有領域の共有領域番号およびモジュール共有鍵K e y__s h a r e dを指す。

[0145] この例では、モジュール（＃ 1） 3 1 0は、セキュア共有領域情報として、ステップS 3 1 1－3で確保した共有領域の共有領域番号「＃ 1」と共有鍵K e y__s h a r e dとを暗号化してモジュール（＃ 2） 3 2 0に送付する。それと共に、モジュール（＃ 1） 3 1 0は、自身を示すモジュール識別子として、タスク識別子「＃ 1」を暗号化してモジュール（＃ 2） 3 2 0に送付する。

[0146] 図2 4のステップS 3 2 1－2で、モジュール（＃ 2） 3 2 0は、モジュール（＃ 1） 3 1 0から送付されたデータを一時鍵K e y__A Bで復号し、セキュア共有領域に関する情報（共有領域番号「＃ 1」および共有鍵K e y__s h a r e d）と、モジュール識別子「＃ 1」とを取得する。これにより、モジュール（＃ 2） 3 2 0側で、モジュール（＃ 1） 3 1 0の情報を共有できる。

[0147] また、モジュール（＃ 2） 3 2 0は、自身を示すモジュール識別子として、タスク識別子「＃ 2」を鍵K e y__A Bで暗号化して、メモリ2 8 0を介してモジュール（＃ 1） 3 1 0に送付する。モジュール（＃ 1） 3 1 0は、モジュール（＃ 2） 3 2 0から送付されたデータを一時鍵K e y__A Bで復号することで、モジュール（＃ 2） 3 2 0を示すモジュール識別子「＃ 2」を得ることができる。これにより、モジュール（＃ 1） 3 1 0側で、モジュール（＃ 2） 3 2 0の情報を共有できる。

[0148] 図2 4において、ステップS 1 2 1－4で、モジュール（＃ 2） 3 2 0は、後述の呼び出される側の処理の直前で処理を停止してスリープ状態となり

、モジュール（＃１）３１０からの呼び出しを待機する。また、図２３において、ステップＳ１１１－６で、モジュール（＃１）は、プログラム（＃１）本体１１２の実行を開始する。

[0149] 次に、モジュールから他のモジュールを呼び出すときの処理を、図２５～図２７を用いて説明する。上述の第１の実施形態と同様に、モジュール（呼出元モジュール）から他のモジュール（呼び出し先モジュール）を呼び出す際には、切替前処理部３０４Ａまたは３０４Ｂにより呼出元モジュールに対して切り替え前処理が実行され、その後、切替後処理部３０６Ａまたは３０６Ｂにより呼び出し先モジュールに対して切り替え後処理が実行される。

[0150] 以下では、モジュール（＃１）３１０からモジュール（＃２）３２０内のプログラム（＃２）本体１２２における関数ｓｕｂ（５）を呼び出す場合を例に説明する。なお、関数ｓｕｂ（５）において、「ｓｕｂ」は関数名であり、括弧内の数値「５」は、当該関数ｓｕｂ（）に渡す引数である。

[0151] 図２５は、本第２の実施形態による切り替え前処理を示す一例のフローチャートである。図２６は、本第２の実施形態による切り替え後処理を示す一例のフローチャートである。切り替え前処理および切り替え後処理は、呼出元モジュールから呼び出し先モジュールを呼び出す呼び出し処理と、呼び出し先モジュールから呼出元モジュールを呼び出して呼出元モジュールを復帰させる復帰処理とで処理内容が異なる。

[0152] 図２７は、呼び出し処理および後述する復帰処理に伴う、モジュール切替管理テーブル４２２、ならびに、切替履歴領域１１８および１２８の状態の変化を概略的に示す。なお、図２７において、モジュール切替管理テーブル４２２内の「Ｋｅｙ」、「ＣＭＩＤ」、「フラグ」、「パラメータ」および「Ｖ」は、それぞれモジュール共有鍵、実行モジュール識別子、切替フラグ、切替パラメータおよび有効フラグを示す。また、モジュール切替管理テーブル４２２は、共有領域番号「＃１」に対応するエントリの状態を示す。

[0153] 上述した初期化処理の直前では、モジュール切替管理テーブル４２２のモジュール共有鍵、実行モジュール識別子、切替フラグおよび切替パラメータ

の各フィールドには何も保存されていない。また、有効ビットフィールドは、「無効を示す値」である値「0」が保存されている。初期化処理を行うことで、モジュール切替管理テーブル422のモジュール共有鍵フィールドおよび実行モジュール識別子フィールドに対し、それぞれモジュール共有鍵Key__sharedおよび実行モジュール識別子「#1」が保存される。また、有効ビットフィールドに「有効を示す値」である値「1」が保存される。

[0154] まず、呼び出し処理について説明する。モジュール（#1）310において、切替前処理部304Aは、モジュール（#2）320を呼び出す直前に、モジュール切り替えの種別を判定する（ステップS304-1）。ここでは、切り替えの要因が呼び出しによると判定され、処理がステップS304-2に移行される。ステップS304-2では呼び出し先から復帰したときに正しい呼び出し元のアドレスに戻るために、呼び出し元のアドレスAddr1を、モジュール（#1）310の切替履歴領域118に保存する。

[0155] 次のステップS304-3で、切替前処理部304Aは、モジュール切替管理テーブル422の切替フラグフィールドに対して、切り替えの要因が「呼び出し」であることを示す切替フラグ「call」を書く処理を行う。このとき、上述した図22のフローチャートに従い、モジュール切替管理部420が有効ビットおよびモジュール共有鍵による判定を行い、モジュール切替管理テーブル422に対する書き込みを制御する。

[0156] すなわち、まず、モジュール（#1）310は、共有領域番号「#1」、モジュール共有鍵Key__sharedおよび切替フラグ「call」を指定する。セキュアプロセッサ400（モジュール切替管理部420）は、上述した図22のフローチャートに従い、モジュール切替管理テーブル422から、モジュール（#1）310から指定された共有領域番号「#1」で示されるエントリに保存されている有効ビットを取得する（図22のステップS424-1）。有効ビットは、図23で説明した初期化処理において「有効を示す値」に書き換えられている（ステップS424-4）。したがって

、モジュール切替管理部 420 は、図 22 のステップ S 424-7 により、モジュール切替管理テーブル 422 から当該共有領域番号「#1」で示されるエントリに保存されているモジュール共有鍵を取得する。

[0157] モジュール切替管理部 420 は、このモジュール切替管理テーブル 422 から取得したモジュール共有鍵と、モジュール（#1）310 から与えられたモジュール共有鍵 `Key_shared` とが一致するか否かを判定する。この例では、これらモジュール共有鍵とモジュール共有鍵 `Key_shared` とが一致するため、モジュール切替管理部 420 は、モジュール切替管理テーブル 422 における共有領域番号「#1」に対応するエントリの切替フラグフィールドに対して切替フラグ「call」を書く。

[0158] 次のステップ S 304-4 で、切替前処理部 304A は、モジュール切替管理テーブル 422 の切替パラメータフィールドに対して、モジュール切り替えに必要な呼び出し情報として、呼び出し先の関数名「sub」とその引数「5」を書く。この場合にも、上述したようにして、モジュール切替管理部 420 が、有効ビットおよびモジュール共有鍵による判定結果に基づきモジュール切替管理テーブル 422 に対する書き込みを制御する。

[0159] 処理は呼び出し処理および復帰処理で共通のステップ S 304-8 に移行され、切替前処理部 304A は、モジュール切替管理テーブル 422 の切替元モジュール識別子フィールドに対して、自身のモジュール識別子「#1」を書く。さらに、次のステップ S 304-9 で、切替前処理部 304A は、モジュール切替管理テーブル 422 の実行モジュール識別子フィールドに対して、呼び出し先のモジュール（この例ではモジュール（#2）320）のモジュール識別子「#2」を書く。

[0160] これらステップ S 304-8 およびステップ S 304-9 の処理においても、上述したようにして、モジュール切替管理部 420 が、有効ビットおよびモジュール共有鍵による判定結果に基づきモジュール切替管理テーブル 422 に対する書き込みを制御する。

[0161] そして、処理はステップ S 304-10 に移行され、モジュール（#1）

310がスリープ状態となる。これは、呼び出し先から復帰する場合や、他のモジュールなどからの呼び出しに備えるためであり、モジュール（#1）310は、自身をスリープ状態として、切り替え後処理の直前で処理を停止する。

[0162] ステップS304-10によってモジュール（#1）310がスリープ状態になると、実行権がOS180に切り替わる。その後、OS180のスケジューラなどの制御によって実行権がモジュール（#2）320に移り、モジュール（#2）320が実行を再開する。

[0163] モジュールの実行権が切り替わると、図26に示す手順に従い切り替え後処理が実行される。実行されるモジュールがモジュール（#1）310からモジュール（#2）320に切り替わると、ステップS306-1で、モジュール（#2）320の切替後処理部306Bは、共有領域情報「#1」およびモジュール共有鍵Key__sharedを指定して、セキュアプロセッサ400に対して実行モジュール識別子の取得を要求する。

[0164] すなわち、まず、モジュール（#2）320は、共有領域番号「#1」およびモジュール共有鍵Key__sharedを指定する。セキュアプロセッサ400（モジュール切替管理部420）は、上述した図22のフローチャートに従い、モジュール切替管理テーブル422から、モジュール（#1）310から指定された共有領域番号「#1」で示されるエントリに保存されている有効ビットを取得する（図22のステップS424-1）。有効ビットは、図23で説明した初期化処理において「有効を示す値」に書き換えられている（ステップS424-4）。したがって、モジュール切替管理部420は、図22のステップS424-7により、モジュール切替管理テーブル422から当該共有領域番号「#1」で示されるエントリに保存されているモジュール共有鍵を取得する。

[0165] モジュール切替管理部420は、このモジュール切替管理テーブル422から取得したモジュール共有鍵と、モジュール（#1）310から与えられたモジュール共有鍵Key__sharedとが一致するか否かを判定する。

この例では、これらモジュール共有鍵とモジュール共有鍵 `Key__shared` とが一致するため、モジュール切替管理部 420 は、モジュール切替管理テーブル 422 における共有領域番号「#1」に対応するエントリの実行モジュール識別子フィールドから実行モジュール識別子「#2」を取得する。取得された実行モジュール識別子「#2」は、モジュール（#2）320 に送信される。

[0166] モジュール（#2）320 の切替後処理部 306B は、セキュアプロセッサ 400 から送信された実行モジュール識別子の値「#2」と、モジュール（#2）320 のモジュール識別子の値とが一致するか否かを判定する（ステップ S306-2）。

[0167] 若し、ステップ S306-2 で、実行モジュール識別子の値とモジュール（#2）320 のモジュール識別子の値とが一致しないと判定されたら、処理はステップ S306-9 に移行されてスリープ状態とされ、モジュール（#2）320 の状態が呼び出し処理の直前の状態に戻る。一方、ステップ S306-2 で、実行モジュール識別子の値とモジュール（#2）320 のモジュール識別子の値とが一致すると判定されたら、処理はステップ S306-3 に移行される。

[0168] この例では、セキュア共有領域 181 の実行モジュール識別子フィールド 182 に、モジュール識別子「#2」が書き込まれているため、モジュール（#2）320 のモジュール識別子と値が一致すると判定される。したがって、切替後処理部 306B は、モジュール（#2）320 を実行すべきであることが分かるので、ステップ S306-3 に処理を移行させて、切り替え後処理を継続する。

[0169] ステップ S306-3 で、切替後処理部 306B は、共有領域番号「#1」およびモジュール共有鍵 `Key__shared` を指定して、モジュール切替管理テーブル 422 から切替フラグを得る。この場合にも、上述したようにして、モジュール切替管理部 420 が、共有領域番号「#1」が示すエントリに対する有効ビットおよびモジュール共有鍵 `Key__shared` によ

る判定結果に基づき、モジュール切替管理テーブル422からの読み出しを制御する。

[0170] 次のステップS306-4で、獲得した切替フラグの値が「呼び出し」および「復帰」の何れを示すかが判定される。若し、切替フラグの値が「呼び出し」を示すと判定されたら、処理はステップS306-5に移行される。一方、切替フラグの値が「復帰」を示すと判定されたら、処理はステップS306-7に移行される。

[0171] この例では、上述したように図25のステップS304-3で「呼び出し」を示す切替フラグ「call」がモジュール切替管理テーブル422の切替フラグフィールドに書き込まれている。そのため、切替フラグの値が「呼び出し」を示すと判定され、関数呼び出しが要求されていることが分かる。したがって、処理がステップS306-5に移行される。

[0172] ステップS306-5で、切替後処理部306Bは、呼び出し元に復帰するために、共有領域番号「#1」およびモジュール共有鍵Key_sharedを指定して、モジュール切替管理テーブル422の切替元モジュール識別子フィールドから切替元モジュール識別子を得る。この場合にも、上述したようにして、モジュール切替管理部420が、共有領域番号「#1」が示すエントリに対する有効ビットおよびモジュール共有鍵Key_sharedによる判定結果に基づき、モジュール切替管理テーブル422からの読み出しを制御する。取り出された切替元モジュール識別子は、モジュール（#2）320の切替履歴領域128に保存される。

[0173] 次のステップS306-6で、切替後処理部306Bは、共有領域番号「#1」およびモジュール共有鍵Key_sharedを指定して、モジュール切替管理テーブル422の切替パラメータフィールドから、呼び出し情報として書き込まれた関数名「sub」と、引数「5」とを得る。この場合にも、上述したようにして、モジュール切替管理部420が、共有領域番号「#1」が示すエントリに対する有効ビットおよびモジュール共有鍵Key_sharedによる判定結果に基づき、モジュール切替管理テーブル422

からの読み出しを制御する。取り出された関数名「s u b」および引数「5」に従い、プログラム（# 2）本体 1 2 2 における関数 s u b（5）の実行を開始する。

[0174] 図 2 7 に例示されるように、呼び出し処理が終了した時点では、モジュール切替管理テーブル 4 2 2 には、切り替え前処理すなわち図 2 5 のステップ S 3 0 4 - 3、ステップ S 3 0 4 - 4 およびステップ S 3 0 4 - 9 で書き込まれた実行モジュール識別子「# 2」、切替フラグ「c a l l」、ならびに、切替パラメータ「s u b」および「5」が保存されている。有効ビットフィールドの値は、「1」のままである。また、モジュール（# 1）3 1 0 の切替履歴領域 1 1 8 には、ステップ S 3 0 4 - 2 で書き込まれた呼び出し元のアドレス A d d r 1 が保存されている。モジュール（# 2）1 2 0 の切替履歴領域 1 2 8 には、切り替え後処理すなわち図 2 6 のステップ S 3 0 6 - 5 で書き込まれたモジュール識別子「# 1」が保存されている。

[0175] 次に、復帰処理について説明する。この例では、実行が呼び出し先のモジュール（# 2）3 2 0 から、当該モジュール（# 2）3 2 0 を呼び出した呼出元のモジュール（# 1）3 1 0 に復帰する場合の処理となる。

[0176] モジュール（# 2）3 2 0 の切替前処理部 3 0 4 B は、モジュール（# 2）3 2 0 から呼び出し元のモジュールに復帰する直前に、ステップ S 3 0 4 - 1 で、モジュール切り替えの種別を判定する。ここでは、切り替えの要因が復帰によるものであると判定され、処理がステップ S 3 0 4 - 5 に移行される。ステップ S 3 0 4 - 5 では、切替前処理部 3 0 4 B は、切替履歴領域 1 2 8 から、呼び出された際（図 2 6 のステップ S 3 0 6 - 5）に保存しておいた切替元のモジュールのモジュール識別子「# 1」を得る。

[0177] 次のステップ S 3 0 4 - 6 で、切替前処理部 3 0 4 B は、共有領域番号「# 1」およびモジュール共有鍵 K e y _ s h a r e d を指定して、復帰を示すフラグ「r e t」を切替フラグとして、モジュール切替管理テーブル 4 2 2 の切替フラグフィールドに書く。この場合にも、上述したようにして、モジュール切替管理部 4 2 0 が、共有領域番号「# 1」が示すエントリに対す

る有効ビットおよびモジュール共有鍵Key__sharedによる判定結果に基づき、モジュール切替管理テーブル422に対する書き込みを制御する。

[0178] 次のステップS304-7で、切替前処理部304Bは、共有領域番号「#1」およびモジュール共有鍵Key__sharedを指定して、モジュール切替管理テーブル422の切替パラメータフィールドに対して、復帰処理に必要な呼び出し復帰情報として、モジュール（#2）320の実行の結果である実行結果「Result__1」を書く。

[0179] 処理は、呼び出し処理および復帰処理で共通のステップS304-8に移行され、切替前処理部304Aは、モジュール切替管理テーブル422の切替元モジュール識別子フィールドに対して、自身のモジュール識別子「#2」を書く。さらに、次のステップS304-9で、切替前処理部304Aは、モジュール切替管理テーブル422の実行モジュール識別子フィールドに対して、呼び出し先のモジュール（この例ではモジュール（#2）310）のモジュール識別子「#1」）を書く。

[0180] これらステップS304-8およびステップS304-9の処理においても、上述したようにして、モジュール切替管理部420が、有効ビットおよびモジュール共有鍵による判定結果に基づきモジュール切替管理テーブル422に対する書き込みを制御する。

[0181] そして、処理はステップS304-10に移行され、モジュール（#1）310がスリープ状態となる。これは、呼び出し先から復帰する場合や、他のモジュールなどからの呼び出しに備えるためであり、モジュール（#1）310は、自身をスリープ状態として、切り替え後処理の直前で処理を停止する。

[0182] ステップS304-10によってモジュール（#2）320がスリープ状態になると、実行権がOS180に切り替わる。その後、OS180のスケジューラなどによって実行権がモジュール（#1）310に移り、モジュール（#1）310が実行を再開する。

[0183] モジュールの実行権が切り替わると、図 26 に示す手順に従い切り替え後処理が実行される。実行されるモジュールがモジュール（# 2）320 からモジュール（# 1）310 に切り替わると、モジュール（# 1）310 の切替後処理部 306A は、共有領域番号「# 1」およびモジュール共有鍵 `Key__shared` を指定して、セキュアプロセッサ 400 に対して実行モジュール識別子の取得を要求する。セキュアプロセッサ 400 は、この要求に応じてモジュール切替管理テーブル 422 の実行モジュール識別子フィールドに書き込まれた実行モジュール識別子「# 1」を取り出し、モジュール（# 1）310 に返す。

[0184] すなわち、先ず、モジュール（# 1）310 は、共有領域番号「# 1」およびモジュール共有鍵 `Key__shared` を指定する。セキュアプロセッサ 400（モジュール切替管理部 420）は、上述した図 22 のフローチャートに従い、モジュール切替管理テーブル 422 から、モジュール（# 1）310 から指定された共有領域番号「# 1」で示されるエントリに保存されている有効ビットを取得する（図 22 のステップ S424-1）。有効ビットは、図 23 で説明した初期化処理において「有効を示す値」に書き換えられている（ステップ S424-4）。したがって、モジュール切替管理部 420 は、図 22 のステップ S424-7 により、モジュール切替管理テーブル 422 から当該共有領域番号「# 1」で示されるエントリに保存されているモジュール共有鍵を取得する。

[0185] モジュール切替管理部 420 は、このモジュール切替管理テーブル 422 から取得したモジュール共有鍵と、モジュール（# 1）310 から与えられたモジュール共有鍵 `Key__shared` とが一致するか否かを判定する。この例では、これらモジュール共有鍵とモジュール共有鍵 `Key__shared` とが一致するため、モジュール切替管理部 420 は、モジュール切替管理テーブル 422 における共有領域番号「# 1」に対応するエントリの実行モジュール識別子フィールドから実行モジュール識別子「# 1」を取得する。

- [0186] 切替後処理部 306A は、セキュアプロセッサ 400 から実行モジュール識別子「#1」が返されると、次のステップ S306-2 で、このセキュアプロセッサ 400 から返された実行モジュール識別子「#1」と、モジュール（#1）310 のモジュール識別子の値とが一致するか否かを判定する。
- [0187] 若し、ステップ S306-2 で、実行モジュール識別子の値とモジュール（#1）310 のモジュール識別子の値とが一致しないと判定されたら、処理はステップ S306-9 に移行されてスリープ状態とされ、モジュール（#1）310 の状態が呼び出し処理の直前の状態に戻る。一方、ステップ S306-2 で、実行モジュール識別子の値とモジュール（#1）310 のモジュール識別子の値とが一致すると判定されたら、処理はステップ S306-3 に移行される。
- [0188] この例では、セキュア共有領域 181 の実行モジュール識別子フィールド 182 に、モジュール識別子「#1」が書き込まれているため、モジュール（#1）310 のモジュール識別子と値が一致すると判定される。したがって、切替後処理部 306 は、モジュール（#1）310 を実行すべきであることが分かるので、ステップ S306-3 に処理を移行させて、切り替え後処理を継続する。
- [0189] ステップ S306-3 で、切替後処理部 306A は、共有領域番号「#1」およびモジュール共有鍵 `Key_shared` を指定して、モジュール切替管理テーブル 422 から切替フラグを得る。この場合にも、上述したようにして、モジュール切替管理部 420 が、共有領域番号「#1」が示すエントリに対する有効ビットおよびモジュール共有鍵 `Key_shared` による判定結果に基づき、モジュール切替管理テーブル 422 からの読み出しを制御する。
- [0190] 次のステップ S306-4 で、獲得した切替フラグの値が「呼び出し」および「復帰」の何れを示すかが判定される。若し、切替フラグの値が「呼び出し」を示すと判定されたら、処理はステップ S306-5 に移行される。一方、切替フラグの値が「復帰」を示すと判定されたら、処理はステップ S

306-7に移行される。

[0191] この例では、上述したように図25のステップS304-6で「復帰」を示す切替フラグ「ret」が切替フラグフィールド183に書き込まれているので、切替フラグの値が「復帰」を示すと判定され、関数呼び出しが要求されていることが分かる。したがって、処理がステップS306-7に移行される。

[0192] ステップS306-7で、切替後処理部306Aは、切替履歴領域118から、上述のステップS304-2で保存しておいた呼び出し元のアドレスAddr1を得ると共に、切替履歴領域118に保存されている当該アドレスAddr1を削除する。次のステップS306-8で、切替後処理部306Aは、共有領域番号「#1」およびモジュール共有鍵Key_sharedを指定して、モジュール切替管理テーブル422の切替パラメータフィールド184から、呼び出し復帰情報を得る。そして、モジュール（#1）310は、獲得した呼び出し復帰情報に含まれる実行結果「Result_1」を用いて、呼び出し元アドレスAddr1に戻って実行を再開する。

[0193] このように、本第2の実施形態では、モジュール（#1）310およびモジュール（#2）320がモジュール共有鍵を共有し、モジュール切替管理部420がそのモジュール共有鍵によってモジュール切替管理テーブル422のエントリへのアクセス制御を行っている。これにより、モジュール切替管理テーブル422のエントリに書かれた値の読み書きを、モジュール（#1）およびモジュール（#2）320からは可能とし、OS180や他のモジュールからは不可能とすることができる。このように、悪意により改変されたOS180や他のモジュールからでは、モジュール切替管理テーブル422に書かれた値を書き換えることができない。そのため、モジュール切替管理部420は、第1の実施形態のセキュア共有領域181と同等の役割を果たしているといえる。

[0194] 図27に例示されるように、復帰処理が終了した時点では、モジュール切替管理テーブル422には、切り替え前処理すなわち図25のステップS3

04-6、ステップS304-7およびステップS304-9で書き込まれた切替フラグ「ret」、切替パラメータ「Result_1」および実行モジュール識別子「#1」が保存されている。また、モジュール（#1）310の切替履歴領域118は、切り替え後処理すなわち図26のステップS306-7で呼び出し元のアドレスAddr1が削除され、何も保存されていない。同様に、モジュール（#2）120の切替履歴領域128は、切り替え前処理すなわちステップS304-5で呼び出し元のモジュール識別子が削除され、何も保存されていない。

[0195] なお、本第2の実施形態では、セキュアプロセッサ400上にモジュール切替管理部420を設けたが、これはこの例に限定されない。すなわち、図28に例示されるように、信頼できる仮想マシンモニタ（Trusted VMM）440が同様の機構を設けるように構成することができる。

[0196] 例えば、プロセッサ450は、本第2の実施形態に特有のモジュール切替管理部420などを含まない一般的なプロセッサとし、仮想マシンモニタ440内部にモジュール切替管理部420を備える方法が考えられる。各モジュール（#1）310、モジュール（#2）320、モジュール（#3）330およびモジュール（#4）340は、仮想マシンモニタ440にあるモジュール切替管理部420を利用して、本第2の実施形態と同様の方法でモジュール切替を行うことが可能である。図28の例では、仮想マシンモニタ440上で複数のOS（OS180、OS_y181およびOS_z182）が動作し、各モジュールは、OS180上で動作している。また、モジュール切替管理部420をハードウェアではなく、OS上に構成することも可能である。

[0197] また、本第2の実施形態では、モジュール切替管理テーブル422における全ての値をセキュアプロセッサ400内に格納するように説明したが、これはこの例に限定されない。すなわち、モジュール切替管理テーブル422の情報の一部をセキュアプロセッサ400内に格納し、残りの部分を、第1の実施形態で示したメモリ280上のセキュア共有領域181へ格納するよ

うに構成することもできる。

[0198] さらに、本第2の実施形態では、モジュール切替管理テーブル422へのアクセスにおいて、1フィールド単位で読み書きを行うように説明したが、これはこの例に限定されず、例えば一度に複数フィールドを読み書きするようにしてもよい。

[0199] <第3の実施形態>

次に、本発明の第3の実施形態について説明する。上述した第1および第2の実施形態は、モジュールの呼び出しを一般的な呼び出し処理により行っていた。これに対して、本第3の実施形態は、モジュールの呼び出しを、言語例外により行う場合の例である。

[0200] ここで、言語例外について説明する。コンピュータの分野において、例外という用語は、ハードウェア例外と言語例外の2種類がある。ハードウェア例外とは、ハードウェア割り込みや不正命令実行などの事象を指す。ハードウェア例外に対しては、ハードウェア例外に対応する割り込みハンドラへ制御を移すといった、プロセッサの機構により処理する。

[0201] 一方、言語例外とは、ファイルが見つからないといった例外的な事象を指し、言語例外はC++やJava（登録商標）などのプログラミング言語環境で提供される。ライブラリなどの、言語例外が発生した関数からその呼び出し元関数を辿って言語例外に対応する例外ハンドラへ制御を移すといった機構により、言語例外を処理する。本第3の実施形態では、後者の言語例外を対象とする。

[0202] また、このような言語例外を扱うための機構を、言語例外処理機構と呼ぶ。言語例外処理機構とは、呼び出し先の関数ではそれらの処理を記述せずに、呼び出し元の関数で言語例外が起きた場合の処理を記述して、呼び出し元の関数に合わせたエラー処理を実行可能とするための仕組みである。このような仕組みは、例えば「Bjarne Stroustrup著 “C++の設計と進化”」などに記載されている。C++やC#などのプログラミング言語環境では、try～catchの形式で記述され、多くのアプリケー

ションで利用されており、その仕組みはよく知られている。

[0203] 図29は、言語例外処理機構800による動作を模式的に示す。モジュール内のある関数で言語例外が発生すると、言語例外処理機構800が例外処理ライブラリを呼び出す。例外処理ライブラリは、まず、言語例外が発生した関数内で、発生した言語例外に対応する例外ハンドラを探索する。例外ハンドラの探索は、例えば発生した言語例外の型に基づき行うことができる。若し、対応する例外ハンドラが見つければ当該例外ハンドラに制御を移し、例外ハンドラが見つからなければ、関数の巻き戻しを行う。

[0204] 関数の巻き戻しとは、呼び出し元関数に戻るための処理を指し、スタックなどに含まれるフレーム情報に基づいて、スタックを呼び出し前の状態に戻し、呼び出し元のアドレスを特定して、そのアドレスに制御を移す。呼び出し元関数に戻ると、例外処理ライブラリは、言語例外に対応する例外ハンドラを探索する。このように、例外処理ライブラリは、言語例外に対応する例外ハンドラが見つかるまで関数の巻き戻しを行って、呼び出し元関数に遡る。

[0205] 図29の例では、モジュール#1～#3の3のモジュールが動作し、モジュール#1からモジュール#2、モジュール#3の順にモジュールを呼び出している。モジュール#3の実行中に言語例外が発生し、その言語例外をキャッチする例外ハンドラがモジュール#1にある場合を示している。図中の実線は、プログラムの実行の流れを示し、波線は、言語例外処理機構800が関数の巻き戻しを行う際に探索される関数の順番を示す。

[0206] モジュール#3で言語例外が発生すると、処理が言語例外処理機構800に移行する。言語例外処理機構800は、先ず、言語例外が発生したモジュール#3内の関数で、言語例外に対応する例外ハンドラを探索する。この例では、言語例外に対応する例外ハンドラが見つからないため、関数の巻き戻しを行ってモジュール#3内の呼び出し元関数に戻る。同様に、呼び出し元関数でも言語例外に対応する例外ハンドラを探索する。言語例外処理機構800は、言語例外に対応する例外ハンドラが見つかるまで、これを繰り返す。

行う。

[0207] 図29の例では、モジュール#3からモジュール#2に戻り、モジュール#2で例外ハンドラを探索する。モジュール#2で例外ハンドラが見つからなければ、さらにモジュール#1に戻って、例外ハンドラを探索する。この例では、モジュール#1の関数まで戻ることによって、発生した言語例外に対応する例外ハンドラが探索されている。例外ハンドラが探索されると、言語例外処理機構800は、探索された例外ハンドラに実行制御を移す。

[0208] このような言語例外処理機構800による言語例外に対する処理を上述した第1または第2の実施形態に適用する場合について考える。セキュアプロセッサでは、図3を用いて説明したように、各モジュールが別々にスタックを管理するため、モジュール内で関数の巻き戻しを行う場合と、モジュールを跨って関数の巻き戻しを行う場合とでは処理が異なる。モジュール内で関数の巻き戻しを行う場合には、通常通り、そのモジュールのスタックなどを調査することで、その関数の呼び出し元の関数を特定することができる。一方、モジュールを跨って関数の巻き戻しを行う場合には、別のモジュールがそのスタックを直接的に見ることができないため、呼び出し元の関数を特定することができない。したがって、本発明の前提となるセキュアプロセッサでは、複数モジュールで構成されると、従来の言語例外処理機構800の関数の巻き戻しが正しく動作しない場合がある。

[0209] 図30は、本発明の第3の実施形態に適用可能なシステムの一例の構成を模式的に示す。図30では、概略的なハードウェア構成と、システムビューとを同時に表している。なお、図30において、上述した図1の構成と共通する部分には同一の符号を付し、詳細な説明を省略する。

[0210] ターゲットシステムは、ハードウェア的には、図1で説明した構成をそのまま適用できるので、ここでの説明を省略する。ターゲットシステムにおいて、セキュアプロセッサ200のコア210上で単一のOS（オペレーティングシステム）180が動作し、OS180上で1または複数のモジュールが動作する。図30の例では、複数のモジュール（#1）510、モジュール

ル（＃２）５２０、モジュール（＃３）５３０およびモジュール（＃４）５４０が動作している。

[0211] 図３０の例では、モジュール（＃１）５１０は、タスク識別子が「＃１」であって、プログラムＡ（Ｐｒｇ Ａ）から生成されていることが示されている。モジュール（＃２）５２０は、タスク識別子が「＃２」であって、プログラムＢ（Ｐｒｇ Ｂ）から生成されている。モジュール（＃３）５３０は、タスク識別子が「＃３」であって、プログラムＣ（Ｐｒｇ Ｃ）から生成されている。また、モジュール（＃４）５４０は、タスク識別子が「＃４」であって、プログラムＢから生成されている。

[0212] 図３１は、本第３の実施形態によるモジュール構成の一例の機能ブロック図を示す。図３１において、モジュール（＃１）５１０、モジュール（＃２）５２０およびモジュール（＃３）５３０は、協働して動作し、それぞれ本第３の実施形態によるモジュール切り替え機構を備える。ここでは、モジュール（＃１）５１０がアプリケーションモジュールであり、モジュール（＃２）５２０およびモジュール（＃３）５３０がライブラリモジュールであるものとする。

[0213] モジュール（＃１）５１０は、上述の図４の構成と同様に初期化処理部１１１、プログラム（＃１）本体１１２、切替前処理部１０４Ａ、切替後処理部１０６Ａおよび切替履歴領域１１８を備えると共に、例外通知部５０５Ａおよび例外受付部５０７Ａをさらに備える。同様に、モジュール（＃２）５２０は、初期化処理部１２１、プログラム（＃２）本体１２２、切替前処理部１０４Ｂ、切替後処理部１０６Ｂおよび切替履歴領域１２８を備えると共に、それぞれモジュール（＃１）５１０の例外通知部５０５Ａおよび例外受付部５０７Ａと共通の処理を行う例外通知部５０５Ｂおよび例外受付部５０７Ｂをさらに備える。また、連携するモジュール（＃１）５１０およびモジュール（＃２）５２０間では、メモリ２８０上にセキュア共有領域１８１を共有する。

[0214] また、図３１では詳細を省略しているが、モジュール（＃３）５３０も、

モジュール（＃１）５１０およびモジュール（＃２）５２０と同様の構成を備える。また、モジュール（＃３）５３０も、モジュール（＃１）５１０およびモジュール（＃２）５２０とメモリ２８０上のセキュア共有領域１８１を共有する。

[0215] 図３２は、本第３の実施形態を適用したモジュール間で言語例外が発生した場合の一例の処理を、模式的に示す。図３２において、長破線は、本第３の実施形態を適用したことによって発生するプログラム実行の流れの例を示す。

[0216] 以下では、図３２に例示されるように、モジュール（＃１）５１０、モジュール（＃２）５２０およびモジュール（＃３）５３０が連係動作し、モジュール（＃１）５１０からモジュール（＃２）５２０を経由してモジュール（＃３）５３０を呼び出す。このとき、モジュール（＃２）５２０から呼び出されたモジュール（＃３）において言語例外が発生し、モジュール（＃３）５３０からモジュール（＃１）５１０にある例外ハンドラまで関数を巻き戻す場合を例に説明する。ここで、各モジュール（＃１）５１０、モジュール（＃２）５２０およびモジュール（＃３）５３０は、それぞれ言語例外処理機構８００Ａ、８００Ｂおよび８００Ｃを有する。

[0217] 本第３の実施形態では、第１の実施形態で示したモジュール間の切り替え処理に多少の変更を加えた処理を行うことにより、モジュールを跨いだ関数の巻き戻し処理を実現する。初期化処理は、第１の実施形態で図７および図８を用いて説明した処理と同一の処理を適用可能であるため、ここでの説明を省略する。初期化処理によって、モジュール（＃１）５１０、モジュール（＃２）５２０およびモジュール（＃３）５３０は、メモリ２８０上に設けられたセキュア共有領域１８１を共有できるようになる。

[0218] また、本第３の実施形態において、言語例外が発生していない通常の状態における、モジュールから他のモジュールを呼び出す際の処理や、他のモジュールを呼び出したモジュールが復帰する際の処理は、上述した第１の実施形態において図６～図１０を用いて説明した処理をそのまま適用可能である

ため、ここでの説明を省略する。

[0219] 図 3 3 ~ 図 3 5 を用いて、本第 3 の実施形態による、モジュールを跨いで関数の巻き戻しを行う際の処理について説明する。図 3 3 は、本第 3 の実施形態による一例の例外通知処理を示すフローチャートである。図 3 4 は、本第 3 の実施形態による一例の例外受付処理を示すフローチャートである。

[0220] また、図 3 5 は、上述した図 1 1 と対応し、言語例外発生の場合の、呼び出し処理および復帰処理、ならびに、言語例外通知処理および言語例外受付処理に伴うセキュア共有領域 1 8 1、ならびに、各モジュール（# 1） 5 1 0、モジュール（# 2） 5 2 0 およびモジュール（# 3） 5 3 0 の切替履歴領域の状態の変化を概略的に示す。

[0221] モジュール（# 1） 5 1 0 から順次モジュールが呼び出され、モジュール（# 3） 5 3 0 に実行権が移るまでは、第 1 の実施形態で図 1 1 を用いて説明したのと同様にして、セキュア共有領域 1 8 1 および各切替履歴領域の状態が変化する。図 3 5 の例では、初期化処理の直後では、セキュア共有領域 1 8 1 には、実行モジュール識別子「# 1」のみが保存されている。また、各切替履歴領域には、何も保存されていない。

[0222] この状態からモジュール（# 2） 5 2 0 が呼び出されると、セキュア共有領域 1 8 1 の実行モジュール識別子フィールド 1 8 2 および切替フラグフィールド 1 8 3 に対して、それぞれ実行モジュール識別子「# 2」、切替フラグ「c a l l」が保存されると共に、切替パラメータフィールド 1 8 4 に対して切替パラメータとして関数名「s u b」および引数「5」が保存される。また、切替履歴領域 1 1 8 には呼び出し元アドレス A d d r 1 が、切替履歴領域 1 2 8 にはモジュール識別子「# 1」がそれぞれ保存される。モジュール（# 3） 5 3 0 の切替履歴領域には何も保存されない。

[0223] さらに、モジュール（# 3） 5 3 0 が呼び出されると、セキュア共有領域 1 8 1 の実行モジュール識別子フィールド 1 8 2 および切替フラグフィールド 1 8 3 に対して、それぞれ実行モジュール識別子「# 3」、切替フラグ「c a l l」が保存されると共に、切替パラメータフィールド 1 8 4 に対して

切替パラメータとして関数名「h e l l o」が保存される。また、切替履歴領域 1 1 8 には呼び出し元アドレス A d d r 1 が、切替履歴領域 1 2 8 にはモジュール識別子「# 1」および呼び出し元アドレス A d d r 2 が、モジュール（# 3） 5 3 0 の切替履歴領域にはモジュール識別子「# 2」が、それぞれ保存される。

[0224] 図 3 3 を参照し、モジュール（# 3） 5 3 0 において言語例外が発生すると、通常の言語例外と同様に、モジュール（# 3） 5 3 0 に含まれる言語例外処理機構 8 0 0 C に制御を移す。言語例外処理機構 8 0 0 C は、言語例外が発生した関数から当該言語例外に対応する例外ハンドラを探索する。一例として、言語例外処理機構 8 0 0 C は、発生した言語例外の型に基づき例外ハンドラを探索することができる。

[0225] 図 3 2 の例では、言語例外処理機構 8 0 0 C は、当該関数から対応する例外ハンドラを見つけることができない。そのため、言語例外処理機構 8 0 0 C は、関数の巻き戻しを実行してモジュール（# 3）内の呼び出し元関数に戻り、当該呼び出し元関数で例外ハンドラを探索する。この例では、この呼び出し元関数からも例外ハンドラが見つからないため、言語例外処理機構 8 0 0 C は、モジュール（# 3） 5 3 0 の例外通知部に制御を移す。例外通知部 5 0 5 は、モジュール（# 3） 5 3 0 の呼び出し元であるモジュール（# 2） 5 2 0 に戻るための処理を行う。

[0226] 図 3 3 において、ステップ S 5 0 5 - 1 で、モジュール（# 3） 5 3 0 の例外通知部は、モジュール（# 3） 5 3 0 の切替履歴領域から、呼び出された際に保存しておいた切り替え元（すなわち、巻き戻し先）のモジュールを示すモジュール識別子「# 2」を得る。そして、モジュール（# 3） 5 3 0 の切替履歴領域から当該モジュール識別子「# 2」を削除する。

[0227] 次のステップ S 5 0 5 - 2 で、モジュール（# 3） 5 3 0 は、セキュア共有領域 1 8 1 にある切替フラグフィールド 1 8 3 に対して、モジュール（# 2） 5 2 0 への切り替えの要因が「言語例外」であることを示すフラグ e x c e p t i o n を書く。このフラグ e x c e p t i o n は、換言すれば、通

常の呼び出しまたは復帰以外を示す値であるといえる。

[0228] 次のステップS505-3で、モジュール(#3)530は、セキュア共有領域181の切替パラメータフィールド184に対して、言語例外情報`excData`を書く。この言語例外情報`excData`には、言語例外の型や言語例外の内容やメッセージなどが含まれる。すなわち、この言語例外情報`excData`は、切り替えの要因の内容を示す情報である。

[0229] 次のステップS505-4で、モジュール(#3)530は、セキュア共有領域181の実行モジュール識別子フィールド182に対して、巻き戻し先のモジュールを示すモジュール識別子「#2」を書く。そして、次のステップS505-5で、モジュール(#3)530は、再度の呼び出しに備え、切り替え後処理の直前で処理を停止し、スリープ状態となる。

[0230] ステップS505-5でモジュール(#3)530がスリープ状態になると、実行権がOS180に切り替わる。その後、OS180のスケジューラなどによって実行権がモジュール(#2)520に移り、モジュール(#2)520が実行を再開する。

[0231] この時点では、図35に例示されるように、セキュア共有領域181には、例外通知処理で書き込まれた実行モジュール識別子「#2」、切替フラグ「`exception`」(図35では「`exc`」と略記)、ならびに、切替パラメータ「`excData`」が保存されている。また、切替履歴領域118には呼び出し元アドレス`Addr1`が、モジュール(#2)520の切替履歴領域128には呼び出し元アドレス`Addr2`とモジュール識別子「#1」とが、それぞれ保存されている。一方、モジュール(#3)530の切替履歴領域には何も保存されない。

[0232] モジュール(#2)520は、割り込みなどによりモジュール(#3)530の例外通知部からの例外通知を受け取ると、モジュール(#2)520内の例外受付部507Bに制御を移す。

[0233] 図34において、例外受付部507は、セキュア共有領域181の実行モジュール識別子フィールド182から、実行モジュール識別子を得る(ステ

ップS 5 0 7 - 1)。そして、次ステップS 5 0 7 - 2で、獲得した実行モジュール識別子の値がモジュール（# 2）5 2 0のモジュール識別子「# 2」と一致するか否かを判定する。若し、一致しないと判定されたら、処理はステップS 5 0 7 - 9に移行され、モジュール（# 2）5 2 0がスリープ状態とされて、次の呼び出しが待機される。呼び出しがあると、処理がステップS 5 0 7 - 1に戻され、処理が再開される。

[0234] 一方、ステップS 5 0 7 - 2で、獲得した実行モジュール識別子の値とモジュール（# 2）5 2 0のモジュール識別子「# 2」とが一致すると判定されたら、処理はステップS 5 0 7 - 3に移行される。この例では、セキュア共有領域1 8 1のモジュール識別子フィールド1 8 2に対して実行モジュール識別子「# 2」が保存されており、これは、モジュール（# 2）5 2 0のモジュール識別子「# 2」と一致するため、モジュール（# 2）5 2 0は実行すべきであると判断でき、処理がステップS 5 0 7 - 3に移行される。

[0235] ステップS 5 0 7 - 3で、モジュール（# 2）5 2 0は、セキュア共有領域1 8 1の切替フラグフィールド1 8 3から、切替フラグを取り出し、次のステップS 5 0 7 - 4で、獲得した切替フラグの値が例外を示す値「e x c e p t i o n」であるか否かを確認する。

[0236] 切替フラグの値が例外を示す値「e x c e p t i o n」であることが確認されたら、次のステップS 5 0 7 - 5で、モジュール（# 2）5 2 0は、切替履歴領域1 2 8から、保存しておいた呼び出し元アドレスA d d r 2を獲得し、獲得した呼び出し元アドレスA d d r 2を切替履歴領域1 2 8から削除する。

[0237] そして、次のステップS 5 0 7 - 6で、モジュール（# 2）5 2 0は、セキュア共有領域1 8 1の切替パラメータフィールド1 8 4から言語例外情報e x c D a t aを獲得し、獲得した言語例外情報e x c D a t aの値を用いて、呼び出し元アドレスA d d r 2に対して擬似的に言語例外を発行する。

[0238] 言語例外が発行されると、モジュール（# 2）5 2 0は、モジュール（# 2）5 2 0に含まれる言語例外処理機構8 0 0 Bに制御を移す。言語例外処

処理機構 800B は、モジュール（#2）520 内で、発生した言語例外に対応する例外ハンドラの探索を行う。若し、当該例外ハンドラが見つかったら、この例外ハンドラによる処理を実行する。一方、当該例外ハンドラが見つからなかった場合には、関数の巻き戻しを実行する。この例では、言語例外処理機構 800B は、モジュール（#2）520 内で当該言語例外に対応する例外ハンドラを見つけることができないため、例外通知部 505B に制御を移す。

[0239] そして、例外通知部 505B は、上述した、モジュール（#3）530 からモジュール（#2）520 に言語例外の発生を通知する場合と同様にして、モジュール（#2）520 からモジュール（#1）510 に対して言語例外の発生を通知する。言語例外の発生がモジュール（#1）510 に通知されると、実行制御がモジュール（#1）510 に移され、モジュール（#1）510 の例外受付部 507A において、擬似的に言語例外が発行される。

[0240] この時点では、図 35 に例示されるように、セキュア共有領域 181 には、例外通知処理で書き込まれた実行モジュール識別子「#1」、切替フラグ「exception」、ならびに、切替パラメータ「excData」が保存されている。また、各モジュールの切替履歴領域には、何も保存されていない。

[0241] 言語例外が発行されると、制御がモジュール（#1）510 に含まれる言語例外処理機構 800A に移される。言語例外処理機構 800A は、上述したようにして、言語例外に対応する例外ハンドラの探索を行う。この探索によって該当する例外ハンドラを見つけると、当該例外ハンドラに制御を移す。

[0242] このように、モジュールの切り替え前後で言語例外用の処理を追加することにより、別々のコンテキストで保護されたモジュール間であって、他のモジュールが直接コンテキストを操作することができなくても、言語例外における関数の巻き戻し処理を実現することができる。

[0243] なお、本第 3 の実施形態では、例外通知部と切替前処理部とを別々に扱っ

てきたが、これはこの例に限られず、切替前処理部が例外通知部の機能を含むように構成することもできる。すなわち、切替前処理の内部で、通常呼び出しの場合には通常の切り替え前処理を行い、言語例外による呼び出しの場合には例外通知部の機能による処理を行う。

[0244] 同様に、本第3の実施形態では、例外受付部と切替後処理部とを別々に扱ってきたが、これはこの例に限られず、切替後処理部が例外受付部の機能を含むように構成することもできる。すなわち、切替後処理部の内部で、通常呼び出しの場合には通常の切り替え後処理を行い、言語例外による呼び出しの場合には例外受付部の処理を行う。

[0245] また、本第3の実施形態では、各モジュールそれぞれが言語例外処理機構を保持するように説明したが、これはこの例に限定されない。例えば、言語例外処理機構を各モジュール内に保持せずに、モジュールが実行するプラットフォーム内に言語例外処理機構を含んで、各モジュールがそのインスタンスを生成し、実行する形式であってもよい。

[0246] さらに、本第3の実施形態では、例外ハンドラを探索するために各モジュールが擬似的に言語例外を発行して言語例外処理機構を呼び出すように説明したが、これはこの例に限定されない。例えば、各モジュールの例外受付部が `call` 命令といった命令を用いて明示的に言語例外処理機構を呼び出して実行してもよい。

[0247] <第4の実施形態>

次に、本発明の第4の実施形態について説明する。本第4の実施形態では、ライブラリとして提供されている関数 `setjmp` および関数 `longjmp` に対する処理について説明する。これら関数 `setjmp` および関数 `longjmp` は、関数の外のジャンプする機構を実現するために設けられたものである。

[0248] 図36は、関数 `setjmp` / `longjmp` による一例の動作を模式的に示す。図36の例では、3のモジュール#1～#3が動作し、モジュール#1からモジュール#2、モジュール#3の順にモジュールを呼び出してい

る。モジュール# 1の実行途中で関数 `set jmp` を行い、モジュール# 3の実行途中で関数 `long jmp` を行っている。

[0249] 関数 `set jmp` は、関数 `set jmp` が実行された場所でのプログラムカウンタやスタックポインタなどのコンテキスト情報を保持する。関数 `long jmp` は、関数 `set jmp` で保存しておいたコンテキスト情報を用いて、関数 `set jmp` を実行した直前の状態に戻る。図36の例では、関数 `set jmp` でモジュール# 1における当該関数 `set jmp` が実行される直前の状態を保存し、関数 `long jmp` でモジュール# 3からモジュール# 1に戻り、モジュール# 1の状態を復元する。このように、関数の外にジャンプする機構は、戻り先のコンテキスト情報を獲得し、現在のコンテキストをそのコンテキスト情報で上書きすることにより実現される。

[0250] 本発明の前提であるセキュアプロセッサにおいては、図3を用いて説明したように、各モジュールが別々にスタック（コンテキスト）を管理しているため、このような、モジュールを跨って関数外へのジャンプを行う機構が正しく動作しない。

[0251] この点についてより具体的に説明する。モジュール内で関数 `set jmp` / `long jmp` を行う場合であれば、従来通り、関数 `set jmp` でコンテキスト情報を獲得し、関数 `long jmp` で当該コンテキストに切り替えることができる。一方、複数モジュールを跨いで関数 `set jmp` / `long jmp` を行う場合には、モジュール間で別々にコンテキストが管理されていると、あるモジュールとは別のモジュールのコンテキストを、直接的に上書きすることができない。したがって、モジュールで別々にコンテキスト管理することを前提とした場合、関数 `set jmp` で得たコンテキストに切り替えることができない場合がある。

[0252] また、モジュールが個別にコンテキストを管理するため、関数 `long jmp` のようにコンテキストを切り替える場合には、関数 `set jmp` を実行したモジュールから関数 `long jmp` を実行するまでに呼び出したモジュールの全てに対して、コンテキストを関数 `set jmp` を実行する前の状態

に戻す必要がある。これは、若し、関数 `set jmp` を実行したモジュールのコンテキストのみを関数 `long jmp` によって変更した場合、それ以外のモジュールは、関数 `long jmp` したときの状態のコンテキストままとなるからである。

[0253] 本第4の実施形態は、コンテキストを別々に管理するモジュール間で関数 `set jmp` / `long jmp` を実現する。本第4の実施形態では、モジュール識別子と、コンテキスト毎にユニークなコンテキスト番号とによってコンテキスト情報を管理し、モジュール間でのコンテキスト情報の直接的なやり取りを行わない。コンテキスト情報には、他のモジュールには秘匿すべき情報が含まれている可能性が多いため、その代わりとなるモジュール識別子とコンテキスト番号とを利用する。

[0254] セキュアプロセッサを用いる本第4の実施形態では、関数 `set jmp` に相当するコンテキスト登録処理により、現在のコンテキスト情報を登録してコンテキスト情報を示すコンテキスト番号を取得する。関数 `long jmp` に相当するコンテキスト設定処理により、モジュール識別子およびコンテキスト番号を指定して、登録されたコンテキスト情報におけるコンテキストを設定する。

[0255] 図37は、本発明の第4の実施形態によるモジュール構成の一例の機能ブロック図を示す。なお、図37において、上述した図1の構成と共通する部分には同一の符号を付し、詳細な説明を省略する。なお、セキュアプロセッサを含むハードウェアの構成は、図1で説明した構成をそのまま適用できるので、ここでの説明を省略する。

[0256] 図37において、モジュール(#1) 910およびモジュール(#2) 920は、それぞれ本第4の実施形態によるモジュール切り替え機構を備える。ここでは、モジュール(#1) 910がアプリケーションモジュールであり、モジュール(#2) 920がライブラリモジュールであるものとする。モジュール(#1) 910およびモジュール(#2) 920は、後述するモジュール(#3) 930と共に、互いに協働して動作する。

- [0257] モジュール（＃１）９１０は、上述の図４の構成と同様に初期化処理部１１１、プログラム（＃１）本体９１２、切替前処理部１０４Ａおよび切替履歴領域１１８を備えると共に、図４の切替後処理部１０６Ａとは部分的に異なる処理を行う切替後処理部９０６Ａを備える。モジュール（＃１）９１０は、さらに、コンテキスト変更通知部９０５Ａ、コンテキスト変更受付部９０７Ａおよびコンテキスト管理テーブル９１９を備える。
- [0258] 同様に、モジュール（＃２）９２０は、初期化処理部１２１、プログラム（＃１）本体９２２、切替前処理部１０４Ｂおよび切替履歴領域１２８を備えると共に、図４の切替後処理部１０６Ｂとは部分的に異なる処理を行う切替後処理部９０６Ｂを備える。モジュール（＃１）９１０は、さらに、コンテキスト管理テーブル９２９を備えると共に、モジュール（＃１）９１０のコンテキスト変更通知部９０５Ａおよびコンテキスト変更受付部９０７Ａと共通の処理を行うコンテキスト変更通知部９０５Ｂおよびコンテキスト変更受付部９０７Ｂを備える。
- [0259] また、アプリケーションモジュールであるプログラム（＃１）本体９１２は、コンテキスト登録部９０８およびコンテキスト設定部９０９Ａを含む。一方、ライブラリモジュールであるモジュール（＃２）９２０は、コンテキスト設定部９０９Ｂを含み、コンテキスト登録部９０８を含まない。これは、ライブラリモジュールであるモジュール（＃３）９３０においても同様である。コンテキスト登録部９０８は、関数 `set jmp` に相当する。コンテキスト設定部９０９Ａは、関数 `long jmp` に相当する。
- [0260] また、図３７では詳細を省略しているが、モジュール（＃３）９３０も、モジュール（＃１）９１０およびモジュール（＃２）９２０と同様の構成を備える。モジュール（＃１）９１０、モジュール（＃２）９２０およびモジュール（＃３）９３０は、メモリ２８０上にセキュア共有領域１８１を共有する。
- [0261] 図３８は、コンテキスト管理テーブル９１９および９２９、ならびに、モジュール（＃３）９３０の図示されないコンテキスト管理テーブルの一例の

データ構成を示す。以下では、特に記載のない限り、コンテキスト管理テーブル 919 および 929、ならびに、モジュール（#3）930 の図示されないコンテキスト管理テーブルを、コンテキスト管理テーブルで代表させて説明する。

[0262] コンテキスト管理テーブルは、コンテキスト番号を示すインデクスが「#1」から「#n」までの n 個のエントリを有する。各エントリは、モジュール識別子フィールドおよびコンテキスト情報フィールドを有する。コンテキスト情報フィールドは、コンテキスト情報としてプログラムカウンタやスタックポインタなどが格納される。

[0263] 図 39～図 44 を用いて、本第 4 の実施形態による、コンテキストの登録／設定を行う際の処理について説明する。ここでは、モジュール（#1）910 でコンテキスト登録を実行し、モジュール（#1）910 からモジュール（#2）920 を呼び出し、さらにモジュール（#2）920 からモジュール（#3）930 を呼び出す。そして、モジュール（#3）930 でコンテキスト設定を実行する例について説明する。なお、各モジュールの初期化処理については、図 7 および図 8 を用いて説明した第 1 の実施形態による処理と同様なので、ここでの詳細な説明を省略する。

[0264] 図 39 は、上述した図 11 に対応し、コンテキスト登録／設定の実行に伴うセキュア共有領域 181、ならびに、各モジュール（#1）910、モジュール（#2）920 およびモジュール（#3）930 の切替履歴領域の状態の変化を概略的に示す。初期化処理が終了した時点では、セキュア共有領域 181 において実行モジュール識別子フィールド 182 に対して実行モジュール識別子「#1」が保存され、切替フラグフィールド 183 および切替パラメータフィールド 184 には、何も保存されていない。また、各モジュール（#1）910、モジュール（#2）920 およびモジュール（#3）930 それぞれの切替履歴領域にも、何も保存されていない。

[0265] コンテキスト登録処理について説明する。図 40 は、本第 4 の実施形態による一例のコンテキスト登録処理を示すフローチャートである。モジュール

(#1) 910においてプログラム(#1) 本体912が実行開始され、関数 `set jmp` 相当の処理を行うため、モジュール(#1) 910は、コンテキスト登録部908に制御を移す。コンテキスト登録部908は、ステップS908-1で、一般的な関数 `set jmp` と同様に、現在のコンテキスト情報Cを取得する。

[0266] 次のステップS908-2で、コンテキスト登録部908は、コンテキスト管理テーブル919内の空いているエントリのコンテキスト番号「#m」を決定する。決定されたコンテキスト番号「#m」は、モジュール(#1) 910に返される。そして、コンテキスト登録部908は、ステップS908-1で取得したコンテキスト情報Cを、コンテキスト管理テーブル919の当該コンテキスト番号「#m」のエントリにおけるコンテキスト情報フィールドに保存する。

[0267] なお、一般的な関数 `set jmp / long jmp` では、他のモジュールに対して直接的にコンテキストを渡す。これに対して、本第4の実施形態では、コンテキスト管理テーブルによってコンテキストを管理し、他のモジュールには、コンテキスト設定情報を渡す。ここでいうコンテキスト設定情報とは、コンテキスト情報をコンテキスト管理テーブルに登録した際の、当該コンテキストが由来するモジュールを識別するモジュール識別子と、コンテキスト登録部908が決定したコンテスト番号とを指す。

[0268] 本第4の実施形態では、各モジュールがコンテキストを設定するときは、コンテキスト情報を直接的に指定するのではなく、コンテキスト設定情報を指定する。

[0269] 次に、モジュールの呼び出し処理について説明する。モジュール(#1) 910においてコンテキストの登録が終了された後、モジュール(#1) 910からモジュール(#2) 920を呼び出し、さらに、モジュール(#2) 920からモジュール(#3) 930を呼び出す。切替前処理部104Aによる処理は、図9を用いて説明した処理と同様であるので、ここでの詳細な説明を省略する。

- [0270] 切り替え前処理により、呼び出し元のアドレス `Addr1` と、呼び出し処理を示す切替フラグ「`call`」と、実行モジュールを示す実行モジュール識別子「`#2`」とが、セキュア共有領域 `181` も呼び出し元アドレスフィールド `Addr1`、切替フラグフィールド `183` および実行モジュール識別子フィールド `182` に対してそれぞれ書き込まれる。
- [0271] 切替パラメータとして、呼び出し先の関数の関数名「`sub`」と引数「`5`」とがセキュア共有領域 `181` の切替パラメータフィールド `183` に対して書き込まれる。本第4の実施形態においては、さらに、コンテキスト設定情報、すなわち、コンテキスト登録が実行されたモジュールを示すモジュール識別子「`#1`」と、コンテキスト番号「`#m`」とが、セキュア共有領域 `181` の切替パラメータフィールド `183` に対して書き込まれる。これにより、呼び出し先のモジュールにおいてコンテキスト設定が実行可能となる。
- [0272] これらモジュール識別子「`#1`」およびコンテキスト番号「`#m`」は、切り替え前処理および切り替え後処理により、呼び出し先のモジュールに対して順次、渡される。
- [0273] 図41は、本第4の実施形態による切り替え後処理を示す一例のフローチャートである。なお、図41において、上述した図10と共通する処理については同一の符号を付し、詳細な説明を省略する。モジュール（`#1`）910からモジュール（`#2`）920が呼び出されると、モジュール（`#2`）920の切替後処理部906Bは、セキュア共有領域 `181` の実行モジュール識別子フィールド `182` の値が「`#2`」であることを確認する（ステップ `S106-1`、ステップ `S106-2`）。
- [0274] 次に、切替後処理部 `106B` は、セキュア共有領域 `181` の切替フラグフィールド `183` から切替フラグを得る（ステップ `S106-3`）。切替後処理部 `106B` は、獲得した値「`call`」から、関数呼び出しが要求されていることが分かる（ステップ `S106-4`）。そして、切替後処理部 `106B` は、ステップ `S106-5` で切替元モジュール識別子フィールド `185` からモジュール識別子「`#1`」を取り出し、切替履歴領域 `118` に保存する。

- [0275] 次のステップS 1 0 6－6で、切替後処理部 1 0 6 Bは、セキュア共有領域 1 8 1の切替パラメータフィールド 1 8 4から、呼び出し情報として書き込まれた関数名「s u b」と、引数「5」とを得ると共に、コンテキスト登録処理が実行されたモジュールを示すモジュール識別子「# 1」とコンテキスト番号「# m」とを得る。
- [0276] 次のステップS 9 0 6－7で、切替後処理部 9 0 6 Bは、取得した切替パラメータがコンテキスト設定情報を含むか否かを判定する。若し、含まないと判定されたら、処理はステップS 9 0 6－9に移行される。一方、含むと判定されたら、処理はステップS 9 0 6－8に移行される。ステップS 9 0 6－8で、切替後処理部 9 0 6 Bは、図 4 1のステップS 1 0 6－6で取得された切替パラメータのうち、コンテキスト設定情報に含まれる各情報を、コンテキスト管理テーブル 9 2 9に保存する。
- [0277] より具体的には、ステップS 9 0 6－8において、コンテキスト設定情報に含まれるモジュール識別子およびコンテキスト番号がコンテキスト管理テーブル 9 2 9に保存される。それと共に、後述するコンテキスト設定処理により呼び出し元のモジュールに戻るためのコンテキスト情報を、コンテキスト管理テーブル 9 2 9のコンテキスト情報フィールドに保存する。このコンテキスト情報は、後述するコンテキスト変更通知処理により用いられる。
- [0278] この時点では、図 3 9に例示されるように、セキュア共有領域 1 8 1には、切り替え前処理で書き込まれた実行モジュール識別子「# 2」および切替フラグ「c a l l」が保存されている。それと共に、セキュア共有領域 1 8 1には、切替パラメータとして、関数名「s u b」および引数「5」、ならびに、モジュール識別子「# 1」およびコンテキスト番号「# m」が保存されている。また、モジュール（# 1） 9 1 0の切替履歴領域 1 1 8には呼び出し元アドレス A d d r 1が保存され、モジュール（# 2） 9 2 0の切替履歴領域 1 2 8にはモジュール識別子「# 1」が保存されている。
- [0279] 次のステップS 9 0 6－9で、切替後処理部 9 0 6 Bは、上述のステップS 1 0 6－6で取得した切替パラメータのうち、呼び出し先の関数名「s u

b」と引数「5」に基づき、プログラム（#2）本体122における関数「sub」の実行を開始させる。

[0280] モジュール（#2）920からモジュール（#3）930を呼び出す際にも、同様の処理を行う。モジュール（#3）930の呼び出しまで完了すると、モジュール（#1）910、モジュール（#2）920およびモジュール（#3）930それぞれのコンテキスト管理テーブルは、モジュール識別子「#1」、コンテキスト番号「#m」のエントリが含まれる状態になる。

[0281] また、コンテキスト管理テーブル919には、図40に示した処理に従いコンテキスト登録を行った時点のコンテキスト情報が保存される。さらに、コンテキスト管理テーブル929およびモジュール（#3）930のコンテキスト管理テーブルには、それぞれが呼び出し元のモジュールに戻るためのコンテキスト情報が含まれる。

[0282] この時点では、図39に例示されるように、セキュア共有領域181には、切り替え前処理で書き込まれた実行モジュール識別子「#3」および切替フラグ「call」が保存されている。それと共に、セキュア共有領域181には、切替パラメータとして、関数名「hello」、ならびに、モジュール識別子「#1」およびコンテキスト番号「#m」が保存されている。また、モジュール（#2）920の切替履歴領域128には、モジュール識別子「#1」および呼び出し元アドレスAddr2が保存される。モジュール（#3）930の切替履歴領域には、モジュール識別子「#2」が保存されている。モジュール（#1）910の切替履歴領域118には、呼び出し元アドレスAddr1が保存される。

[0283] 次に、コンテキスト設定処理について説明する。この例では、図36に例示されるように、モジュール（#3）930においてコンテキスト設定処理が実行される。

[0284] 図42は、第4の実施形態による一例のコンテキスト設定処理を示すフローチャートである。ステップS909-1で、モジュール（#3）930のコンテキスト設定部は、図41のステップS106-5でセキュア共有領域

181から取得した切替パラメータ中の、コンテキスト設定情報に含まれるモジュール識別子「#1」およびコンテキスト番号「#m」に基づき、モジュール（#3）930のコンテキスト管理テーブルを参照する。そして、これらモジュール識別子「#1」およびコンテキスト番号「#m」に対応するコンテキスト情報を取得する。それと共に、当該コンテキスト情報を、コンテキスト管理テーブルから削除する。

[0285] 次のステップS909-2で、モジュール（#3）930のコンテキスト設定部は、ステップS909-1で取得したコンテキスト情報を、モジュール（#3）930における現在のコンテキスト情報に対して上書きする。この場合、ステップS909-1で取得したコンテキスト情報は、モジュール（#3）930の呼び出し元であるモジュール（#2）920に戻るためのものである。そのため、モジュール（#3）930は、モジュール（#3）930のコンテキスト変更通知部に制御を移す。

[0286] 図43は、第4の実施形態による一例のコンテキスト変更通知処理を示すフローチャートである。ステップS905-1で、コンテキスト変更通知部は、モジュール（#3）930の切替履歴領域から、モジュール（#2）920から呼び出された際に保存しておいた呼び出し元のモジュールを示すモジュール識別子「#2」を取り出し、当該切替履歴領域から、獲得したモジュール識別子を削除する。

[0287] 次のステップS905-2で、モジュール（#3）930のコンテキスト変更通知部は、セキュア共有領域181の切替フラグフィールド183に対し、モジュール切替の要因が「コンテキスト設定」であることを示す切替フラグ「L」を書く。この切替フラグ「L」は、換言すれば、通常の呼び出し、呼び出し復帰または言語例外以外を示す値といえる。さらに、コンテキスト変更通知部は、ステップS905-3で、切替パラメータフィールド184に対して、モジュール識別子「#1」とコンテキスト番号「#m」とを、コンテキスト設定情報として書く。このコンテキスト設定情報は、換言すれば、モジュール切替の要因を示す情報といえる。

- [0288] 次のステップS 9 0 5 - 4で、モジュール（# 3） 9 3 0のコンテキスト変更通知部は、セキュア共有領域 1 8 1の実行モジュール識別子フィールド 1 8 2に対し、呼び出し元のモジュールを示すモジュール識別子「# 2」を書く。さらに、モジュール（# 3） 9 3 0のコンテキスト変更通知部は、ステップS 9 0 5 - 6で、モジュール（# 3） 9 3 0をスリープ状態とする。切り替え後処理の直前でスリープ状態として処理を停止させることで、再度の呼び出しに備える。
- [0289] この時点では、図 3 9に例示されるように、セキュア共有領域 1 8 1には、コンテキスト変更通知処理で書き込まれた実行モジュール識別子「# 2」、切替フラグ「L」が保存されていると共に、切替パラメータとしてモジュール識別子「# 1」とコンテキスト番号「# m」とが保存されている。また、モジュール（# 1） 9 1 0の切替履歴領域 1 1 8には呼び出し元アドレス A d d r 1が、モジュール（# 2） 9 2 0の切替履歴領域 1 2 8には呼び出し元アドレス A d d r 2が、それぞれ保存されている。一方、モジュール（# 3） 9 3 0の切替履歴領域には、何も保存されていない。
- [0290] ステップS 9 0 5 - 5でモジュール（# 3） 9 3 0がスリープ状態となると、実行権がOSに切り替わる。その後、OSのスケジューラなどによって実行権がモジュール（# 2） 9 2 0に移り、モジュール（# 2） 9 2 0が実行を再開する。モジュール（# 2） 9 2 0は、実行が再開され呼び出し元のモジュール（# 3） 9 3 0のコンテキスト変更通知部から変更通知を受けると、制御をコンテキスト変更受付部 9 0 7 Bに移す。
- [0291] 図 4 4は、第 4の実施形態による一例のコンテキスト変更受付処理を示すフローチャートである。コンテキスト変更受付部 9 0 7 Bは、ステップS 9 0 7 - 1で、セキュア共有領域 1 8 1の実行モジュール識別子フィールド 1 8 2から実行モジュール識別子を得る。次のステップS 9 0 7 - 2で、コンテキスト変更受付部 9 0 7 Bは、獲得した実行モジュール識別子の値が、モジュール（# 2） 9 2 0のモジュール識別子「# 2」と一致するか否かを判定する。もし、一致しないと判定されたら、処理はステップS 9 0 7 - 9に

移行され、モジュール（＃２）９２０がスリープ状態とされて、次の呼び出しが待機される。呼び出しがあると、処理がステップＳ９０７－１に戻され、処理が再開される。

[0292] 一方、ステップＳ９０７－２で、獲得した実行モジュール識別子の値とモジュール（＃２）９２０のモジュール識別子「＃２」とが一致すると判定されたら、処理はステップＳ９０７－３に移行される。この例では、セキュア共有領域１８１のモジュール識別子フィールド１８２に対して実行モジュール識別子「＃２」が保存されており、これは、モジュール（＃２）９２０のモジュール識別子「＃２」と一致する。したがって、モジュール（＃２）９２０を実行すべきであると判断でき、処理がステップＳ９０７－３に移行される。

[0293] ステップＳ９０７－３で、コンテキスト変更受付部９０７Ｂは、セキュア共有領域１８１の切替フラグフィールド１８３から切替フラグを取り出し、次のステップＳ９０７－４で、獲得した切替フラグの値がモジュール切替の要因が「コンテキスト設定」であることを示す値「Ｌ」であるか否かを確認する。切替フラグの値が値「Ｌ」であることが確認されたら、処理はステップＳ９０７－５に移行される。

[0294] ステップＳ９０７－５で、コンテキスト変更受付部９０７Ｂは、セキュア共有領域１８１の切替パラメータフィールド１８４からコンテキスト設定情報を取得する。その後、コンテキスト設定部９０９Ｂに制御を移す。

[0295] 図４２を参照し、ステップＳ９０９－１で、コンテキスト設定部９０９Ｂは、セキュア共有領域１８１に保存される切替パラメータ中の、コンテキスト設定情報に含まれるモジュール識別子「＃１」およびコンテキスト番号「＃ｍ」に基づき、コンテキスト管理テーブル９２９を参照する。そして、これらモジュール識別子「＃１」およびコンテキスト番号「＃ｍ」に対応するコンテキスト情報を取得する。それと共に、当該コンテキスト情報を、コンテキスト管理テーブルから削除する。

[0296] 次のステップＳ９０９－２で、モジュール（＃２）９２０は、ステップＳ

909-1で取得したコンテキスト情報がモジュール（#2）920の呼び出し元であるモジュール（#1）910に戻るための情報であるため、コンテキスト設定部909Bからコンテキスト変更通知部905Bに制御を移す。

[0297] 図43を参照し、コンテキスト変更通知部905Bは、上述したモジュール（#3）930のコンテキスト変更通知部による処理と同様の処理を行い、セキュア共有領域181の切替フラグフィールド183に対して切替フラグ「L」を、切替パラメータフィールド184に対してモジュール識別子「#1」とコンテキスト番号「#m」とを、実行モジュール識別子フィールド182に対して、呼び出し元のモジュールを示すモジュール識別子「#1」を、それぞれ書く。そして、コンテキスト変更通知部905Bは、モジュール（#2）920をスリープ状態とする（図43のステップS905-1～ステップS905-6参照）。

[0298] この時点では、図39に例示されるように、セキュア共有領域181には、コンテキスト変更通知処理で書き込まれた実行モジュール識別子「#1」、切替フラグ「L」が保存されていると共に、切替パラメータとしてモジュール識別子「#1」とコンテキスト番号「#m」とが保存されている。また、モジュール（#1）910の切替履歴領域118には呼び出し元アドレスAddr1が保存されている。一方、他のモジュールの切替履歴領域には、何も保存されていない。

[0299] モジュール（#2）920がスリープ状態となると、実行権がOSに切り替わる。その後、OSのスケジューラなどによって実行権がモジュール（#1）910に移り、モジュール（#1）910が実行を再開する。モジュール（#1）910は、実行が再開され呼び出し元のモジュール（#2）920のコンテキスト変更通知部から変更通知を受けると、制御をコンテキスト変更受付部907Aに移す。

[0300] コンテキスト変更受付部907Aは、上述したモジュール（#2）920のコンテキスト変更受付部907Bと同様の処理を行う。図44を参照し、

コンテキスト変更受付部 907A は、セキュア共有領域 181 の実行モジュール識別子フィールド 182 の値が「#1」であることを確認する（ステップ S907-1、ステップ S907-2）。

[0301] 次に、コンテキスト変更受付部 907B は、セキュア共有領域 181 の切替フラグフィールド 183 から切替フラグを獲得し、獲得した切替フラグの値がモジュール切替の要因が「コンテキスト設定」であることを示す値「L」であることを確認する（ステップ S907-3、ステップ S907-4）。

[0302] 次のステップ S907-5 で、コンテキスト変更受付部 907A は、セキュア共有領域 181 の切替パラメータフィールド 184 から切替パラメータに含まれるコンテキスト設定情報を得て、コンテキスト設定情報からモジュール識別子とコンテキスト番号とを獲得する。そして、コンテキスト設定情報から獲得したモジュール識別子とコンテキスト番号とを指定して、コンテキスト設定部 909A に制御を移す。

[0303] 図 42 を参照し、ステップ S909-1 で、コンテキスト設定部 909A は、セキュア共有領域 181 に保存される切替パラメータ中の、コンテキスト設定情報に含まれるモジュール識別子「#1」およびコンテキスト番号「#m」に基づきコンテキスト管理テーブル 919 を参照する。そして、コンテキスト管理テーブル 919 から、これらモジュール識別子「#1」およびコンテキスト番号「#m」に対応するコンテキスト情報を取得し、当該コンテキスト情報を、コンテキスト管理テーブルから削除する。

[0304] ステップ S909-1 で取得したコンテキスト情報は、モジュール(#1) 910 内でコンテキスト登録処理によりコンテキストを登録した場所のものである。そのため、次のステップ S909-2 で、モジュール(#1) 910 は、実行位置がコンテキスト登録処理で登録された場所まで戻り、実行を再開する。

[0305] このように、コンテキスト登録を行ったモジュールでは、コンテキスト登録した時点でのコンテキスト情報を保存しておく。また、コンテキスト登録

を行った以外のモジュールでは、コンテキスト設定のための値を書いて、呼び出し元に実行を切り替える際の切り替え前処理におけるコンテキスト情報を保存する。これにより、コンテキスト設定で戻るときには、コンテキスト管理テーブルに保存されるコンテキスト情報で上書きし続けるだけで、コンテキストを登録した場所に戻れることが可能とされる。

[0306] なお、本第4の実施形態では、コンテキスト変更通知部と切替前処理部とを別々に扱ってきたが、これはこの例に限られず、切替前処理部がコンテキスト変更通知部の機能を含むように構成することもできる。すなわち、切替処理部の内部で、通常呼び出しの場合には通常の切り替え前処理を行い、コンテキスト設定であればコンテキスト変更通知部の機能による処理を行う。

[0307] 同様に、本第4の実施形態では、コンテキスト変更受付部と切替後処理部とを別々に扱ってきたが、これはこの例に限定されず、切替後処理部がコンテキスト変更受付部の機能を含むように構成することもできる。すなわち、切替後処理部の内部で、通常呼び出しの場合には通常の切り替え後処理を行い、コンテキスト設定であればコンテキスト変更受付部の機能による処理を行う。

[0308] また、本第4の実施形態では、コンテキスト管理テーブルをモジュール単位で保持するように説明したが、これはこの例に限定されない。例えば、セキュア共有領域と同じように、連携するモジュール間で共有する領域にコンテキスト管理テーブルを設けることも可能である。さらに、本第4の実施形態では、モジュール識別子とコンテキスト番号によりコンテキスト情報を管理しているが、これはこの例に限定されない。すなわち、モジュール識別子とコンテキスト番号は、モジュール間でユニークな値を1つ持つ方法であってもよい。

[0309] <第5の実施形態>

次に、本発明の第5の実施形態について説明する。本第5の実施形態は、上述した第4の実施形態の、Cライブラリとして提供されている関数 `setjmp` および関数 `longjmp` に対する処理についての変形例である。こ

ここでは、上述と同様に、アプリケーションモジュールであるモジュール# 1でコンテキストを登録し、それぞれライブラリモジュールであるモジュール# 2を経由して、モジュール# 3でコンテキストを設定する場合を例に説明する。

[0310] 本第5の実施形態では、上述の第4の実施形態と同様に、モジュール識別子とコンテキスト番号とによりコンテキスト情報を管理する。また、本第5の実施形態では、関数 `set jmp` 相当のコンテキスト登録処理により、コンテキスト情報を示すコンテキスト番号を取得し、関数 `long jmp` 相当のコンテキスト設定処理により、モジュール識別子およびコンテキスト番号を指定して、コンテキストを切り替える。

[0311] 図45は、本発明の第5の実施形態によるモジュール構成の一例の機能ブロック図を示す。なお、図45において、上述した図37の構成と共通する部分には同一の符号を付し、詳細な説明を省略する。なお、セキュアプロセッサを含むハードウェアの構成は、図1で説明した構成をそのまま適用できるので、ここでの説明を省略する。

[0312] 図45において、モジュール(#1) 710およびモジュール(#2) 720は、それぞれ本第5の実施形態によるモジュール切り替え機構を備える。ここでは、モジュール(#1) 710がアプリケーションモジュールであり、モジュール(#2) 720がライブラリモジュールであるものとする。モジュール(#1) 710およびモジュール(#2) 720は、後述するモジュール(#3) 730と共に、互いに協働して動作する。

[0313] モジュール(#1) 710は、初期化処理部111、プログラム(#1) 本体712、切替前処理部104A、切替後処理部706A、切替履歴領域118、コンテキスト変更通知部905A、コンテキスト変更受付部 907Aおよびコンテキスト管理テーブル719を備える。また、プログラム(#1) 本体712は、コンテキスト登録部908およびコンテキスト設定部702Aを備える。コンテキスト登録部908は、関数 `set jmp` に相当し、コンテキスト設定部702Aは、関数 `long jmp` に相当する。

- [0314] これらのうち、コンテキスト設定部 702A およびコンテキスト管理テーブル 719 は、図 37 を用いて説明した、第 4 の実施形態におけるコンテキスト設定部 902A およびコンテキスト管理テーブル 919 にそれぞれ対応し、一部の機能が異なる。
- [0315] 同様に、モジュール（#2）720 は、初期化処理部 121、プログラム（#1）本体 722、切替前処理部 104B、切替後処理部 106B、切替履歴領域 128、コンテキスト変更通知部 905B、コンテキスト変更受付部 907B およびコンテキスト管理テーブル 729 を備える。また、プログラム（#1）本体 722 は、コンテキスト設定部 702B を備える。コンテキスト設定部 702B は、関数 `long jmp` に相当する。
- [0316] 上述のモジュール（#1）710 と同様に、これらのうち、コンテキスト設定部 702B およびコンテキスト管理テーブル 729 は、図 37 を用いて説明した、第 4 の実施形態におけるコンテキスト設定部 902B およびコンテキスト管理テーブル 929 にそれぞれ対応し、一部の機能が異なる。
- [0317] また、図 45 では詳細を省略しているが、モジュール（#3）730 も、モジュール（#2）720 と同様の構成を備え、プログラム（#3）本体がコンテキスト設定部を有する（図示しない）。モジュール（#1）710、モジュール（#2）720 およびモジュール（#3）730 は、メモリ 280 上に例えば領域 `Sh_mem1` としてセキュア共有領域 181 を共有する。
- [0318] 図 46 は、コンテキスト管理テーブル 719 および 729、ならびに、モジュール（#3）730 の図示されないコンテキスト管理テーブルの一例のデータ構成を示す。以下では、特に記載のない限り、コンテキスト管理テーブル 719 および 729、ならびに、モジュール（#3）730 の図示されないコンテキスト管理テーブルを、コンテキスト管理テーブルで代表させて説明する。
- [0319] コンテキスト管理テーブルは、コンテキスト番号を示すインデクスが「#1」から「#n」までの n 個のエントリを有する。各エントリは、コンテキ

スト情報フィールドを有する。コンテキスト情報フィールドは、コンテキスト情報としてプログラムカウンタやスタックポインタなどが格納される。

[0320] 図 4 3、図 4 4、図 4 7 および図 4 8 を用いて、本第 5 の実施形態による、コンテキスト登録／設定を行う際の処理について説明する。ここでは、モジュール（# 1） 7 1 0 でコンテキスト登録を実行し、モジュール（# 1） 7 1 0 からモジュール（# 2） 7 2 0 を呼び出し、さらにモジュール（# 2） 7 2 0 からモジュール（# 3） 7 3 0 を呼び出す。そして、モジュール（# 3） 7 3 0 でコンテキスト設定を実行する例について説明する。なお、各モジュールの初期化処理については、図 7 および図 8 を用いて説明した第 1 の実施形態による処理と同様なので、ここでの詳細な説明を省略する。

[0321] 図 4 7 は、上述した図 1 1 に対応し、コンテキスト登録／設定の実行に伴うセキュア共有領域 1 8 1、ならびに、各モジュール（# 1） 7 1 0、モジュール（# 2） 7 2 0 およびモジュール（# 3） 7 3 0 の切替履歴領域の状態の変化を概略的に示す。初期化処理が終了した時点では、セキュア共有領域 1 8 1 において実行モジュール識別子フィールド 1 8 2 に対して実行モジュール識別子「# 1」が保存され、切替フラグフィールド 1 8 3 および切替パラメータフィールド 1 8 4 には、何も保存されていない。また、各モジュール（# 1） 7 1 0、モジュール（# 2） 7 2 0 およびモジュール（# 3） 7 3 0 それぞれの切替履歴領域には、何も保存されていない。

[0322] 先ず、アプリケーションモジュールであるモジュール（# 1） 7 1 0 においてコンテキストの登録処理が行われる。このコンテキスト登録処理は、図 4 0 を用いて説明した、第 4 の実施形態によるコンテキスト登録処理と同一の手順により行われる。このとき、コンテキスト管理テーブル 7 1 9 のエントリ（# m）に対して、コンテキスト登録処理を実行した際のコンテキストが登録される。

[0323] モジュール（# 1） 7 1 0 においてコンテキストの登録が終了された後、モジュール（# 1） 7 1 0 からモジュール（# 2） 7 2 0 を呼び出し、さらに、モジュール（# 2） 7 2 0 からモジュール（# 3） 7 3 0 を呼び出す。

モジュール（＃１）７１０からモジュール（＃２）７２０を呼び出す際の切替前処理部１０４Ａによる処理、ならびに、モジュール（＃２）７２０からモジュール（＃３）７３０を呼び出す際の切替前処理部１０４Ｂによる処理は、図７を用いて説明した処理と同様であるので、ここでの詳細な説明を省略する。

[0324] 切り替え前処理により、呼び出し元のアドレス `Addr1` と、呼び出し処理を示す切替フラグ「`call`」と、実行モジュールを示す実行モジュール識別子「＃２」とが、セキュア共有領域１８１も呼び出し元アドレスフィールド `Addr1`、切替フラグフィールド１８３および実行モジュール識別子フィールド１８２に対してそれぞれ書き込まれる。

[0325] 切替パラメータとして、呼び出し先の関数の関数名「`sub`」と引数「５」とがセキュア共有領域１８１も切替パラメータフィールド１８３に対して書く。本第５の実施形態においては、さらに、コンテキスト設定情報、すなわち、コンテキスト登録が実行されたモジュールを示すモジュール識別子「＃１」と、コンテキスト番号「＃*m*」とが、セキュア共有領域１８１の切替パラメータフィールド１８３に対して書き込まれる。これにより、呼び出し先のモジュールにおいてコンテキスト設定が実行可能となる。

[0326] これらモジュール識別子「＃１」およびコンテキスト番号「＃*m*」は、切り替え前処理および切り替え後処理により、呼び出し先のモジュールに対して順次、渡される。

[0327] 切替前処理部１０４Ａまたは切替前処理部１０４Ｂにおける切り替え前処理が終了すると、呼び出し先のモジュールが呼び出され、実行するモジュールが切り替えられる。呼び出されたモジュールに制御が移ると、当該モジュールにおいて切り替え後処理が行われる。切り替え後処理は、上述の第４の実施形態において、図４１を用いて第１の実施形態による処理と同様であるとして説明した処理と異なるところがないため、ここでの詳細な説明を省略する。

[0328] モジュールの呼び出し処理に伴うセキュア共有領域１８１および各モジュ

ールの切替履歴領域の状態の変化について、図47を参照しながら説明する。モジュール(#1)710からモジュール(#2)720に制御が切り替わり、モジュール(#2)720において切り替え後処理が終了した時点では、セキュア共有領域181に対して、実行モジュール識別子「#2」および切替フラグ「c a l l」が保存されている。それと共に、セキュア共有領域181に対して、切替パラメータとして、関数名「s u b」および引数「5」、ならびに、モジュール識別子「#1」およびコンテキスト番号「#m」が保存されている。また、モジュール(#1)710の切替履歴領域118には呼び出し元アドレスA d d r 1が保存され、モジュール(#2)720の切替履歴領域128にはモジュール識別子「#1」が保存されている。

[0329] また、モジュール(#2)720からモジュール(#3)730に制御が切り替わり、モジュール(#3)730において切り替え後処理が終了した時点では、セキュア共有領域181に、実行モジュール識別子「#3」および切替フラグ「c a l l」が保存される。それと共に、セキュア共有領域181の切替パラメータに、関数名「h e l l o」、ならびに、モジュール識別子「#1」およびコンテキスト番号「#m」が保存される。また、モジュール(#2)720の切替履歴領域128には、モジュール識別子「#1」および呼び出し元アドレスA d d r 2が保存される。モジュール(#3)730の切替履歴領域には、モジュール識別子「#2」が保存されている。モジュール(#1)710の切替履歴領域118には、呼び出し元アドレスA d d r 1が保存される。

[0330] このとき、モジュール(#1)710のコンテキスト管理テーブル719は、コンテキスト番号「#m」のエントリが含まれる状態となる。一方、モジュール(#2)720およびモジュール(#3)730それぞれのコンテキスト管理テーブルは、何も含まない状態である。本第5の実施形態は、この点で、上述した第4の実施形態と異なる。

[0331] 次に、本第5の実施形態によるコンテキスト設定処理について説明する。この例では、図36に例示されるように、モジュール(#3)730におい

てコンテキスト設定処理が実行される。

[0332] 図 4 8 は、第 5 の実施形態によるコンテキスト設定処理を示す一例のフローチャートである。まず、モジュール（# 3） 7 3 0 のコンテキスト設定部は、呼び出し元のモジュールから、セキュア共有領域 1 8 1 を介して渡されたコンテキスト設定情報（モジュール識別子およびコンテキスト番号）を用いて、コンテキスト設定処理を実行する。まず、ステップ S 7 0 7 - 1 で、モジュール識別子の値が自身のモジュールを示すモジュール識別子の値と一致しているか否かを判定する。若し、モジュール識別子の値と自身のモジュールを示すモジュール識別子の値とが一致していると判定されたら、処理はステップ S 7 0 7 - 2 に移行される。

[0333] ステップ S 7 0 7 - 2 では、コンテキスト番号に対応するコンテキスト情報を取得すると共に、当該コンテキスト情報を、コンテキスト管理テーブルから削除する。そして、次のステップ S 7 0 7 - 3 で、モジュール（# 3） 7 3 0 のコンテキスト設定部は、ステップ S 7 0 7 - 1 で取得したコンテキスト情報を、モジュール（# 3） 7 3 0 における現在のコンテキスト情報に対して上書きする。

[0334] 一方、ステップ S 7 0 7 - 1 で、モジュール識別子が自身のモジュールを示していないと判定されたら、処理はステップ S 7 0 7 - 4 に移行される。この場合、呼び出し元のモジュールに戻る必要があると判断できる。そこで、ステップ S 7 0 7 - 4 で、モジュール（# 3） 7 3 0 のコンテキスト設定部は、モジュール（# 3） 7 3 0 のスタックなどのコンテキストを、切り替え直後の状態に戻す。そして、制御をモジュール（# 3） 7 3 0 のコンテキスト変更通知部に移す。

[0335] この例では、モジュール識別子「# 1」が自身のモジュールを示すモジュール識別子「# 3」と異なっているため、処理がステップ S 7 0 7 - 4 に移行され、モジュール（# 3） 7 3 0 のスタックなどのコンテキストを、切り替え前の状態に戻す。そして、制御をモジュール（# 3）のコンテキスト変更通知部に移す。モジュール（# 3） 7 3 0 のコンテキスト変更通知部によ

る処理は、図 4 3 を用いて説明した、第 4 の実施形態によるコンテキスト変更通知部による処理と同一であるので、ここでの説明を省略する。

[0336] この時点では、図 4 7 に例示されるように、セキュア共有領域 1 8 1 には、コンテキスト変更通知処理で書き込まれた実行モジュール識別子「# 2」、切替フラグ「L」が保存されていると共に、切替パラメータとしてモジュール識別子「# 1」とコンテキスト番号「#m」とが保存されている。また、モジュール（# 1）7 1 0 の切替履歴領域 1 1 8 には呼び出し元アドレス A d d r 1 が、モジュール（# 2）7 2 0 の切替履歴領域 1 2 8 には呼び出し元のモジュール識別子「# 1」が、それぞれ保存されている。一方、モジュール（# 3）7 3 0 の切替履歴領域には、何も保存されていない。

[0337] モジュール（# 3）7 3 0 がスリープ状態となると（図 4 3 のステップ S 9 0 5 - 5 参照）、実行権が O S に切り替わる。その後、O S のスケジューラなどによって実行権がモジュール（# 2）7 2 0 に移り、モジュール（# 2）7 2 0 が実行を再開する。モジュール（# 2）7 2 0 は、実行が再開され呼び出し元のモジュール（# 3）7 3 0 のコンテキスト変更通知部から変更通知を受けると、制御をコンテキスト変更受付部 9 0 7 B に移す。

[0338] コンテキスト変更受付部 9 0 7 B による処理は、図 4 4 を用いて説明した、第 4 の実施形態によるコンテキスト変更受付部での処理と同一であるので、ここでの説明を省略する。最後に、コンテキスト変更通知部 9 0 7 B は、取得した切替パラメータに含まれるコンテキスト設定情報を指定して、コンテキスト設定部 7 0 2 B に制御を移す（図 4 4 のステップ S 9 0 7 - 5 参照）。

[0339] 図 4 8 を参照し、コンテキスト設定部 7 0 2 B は、ステップ S 7 0 7 - 1 で、コンテキスト変更受付部 9 0 7 B で取得された切替パラメータから獲得したモジュール識別子が値「# 1」であり、自身のモジュールを示すモジュール識別子が値「# 2」であって、両者は一致しないため、処理がステップ S 7 0 7 - 4 に移行されることになる。

[0340] ステップ S 7 0 7 - 4 で、コンテキスト設定部 7 0 2 B は、モジュール（

#2) 720のスタックなどのコンテキストを、切り替え直後の状態に戻す。そして、制御をモジュール(#2) 720のコンテキスト変更通知部905Bに移す。コンテキスト変更通知部905Bによる処理は、図43を用いて説明した、第4の実施形態によるコンテキスト変更通知部での処理と同一なので、ここでの説明を省略する。

[0341] この時点では、図47に例示されるように、セキュア共有領域181には、コンテキスト変更通知処理で書き込まれた実行モジュール識別子「#1」、切替フラグ「L」が保存されていると共に、切替パラメータとしてモジュール識別子「#1」とコンテキスト番号「#m」とが保存されている。また、モジュール(#1) 710の切替履歴領域118には、呼び出し元アドレスAddr1が保存されている。一方、モジュール(#2) 720およびモジュール(#3) 730の各切替履歴領域には、何も保存されていない。

[0342] モジュール(#2) 720がスリープ状態となると(図43のステップS905-5参照)、実行権がOSに切り替わる。その後、OSのスケジューラなどによって実行権がモジュール(#1) 710に移り、モジュール(#1) 710が実行を再開する。モジュール(#1) 710は、実行が再開され呼び出し元のモジュール(#2) 720のコンテキスト変更通知部から変更通知を受けると、制御をコンテキスト変更受付部907Aに移す。

[0343] コンテキスト変更受付部907Aによる処理は、図44を用いて説明した、第4の実施形態によるコンテキスト変更受付部での処理と同一であるので、ここでの説明を省略する。最後に、コンテキスト変更通知部907Aは、取得した切替パラメータに含まれるコンテキスト設定情報を指定して、コンテキスト設定部702Aに制御を移す(図44のステップS907-5参照)。

[0344] 図48を参照し、コンテキスト設定部702Aは、呼び出し元のモジュールから、セキュア共有領域181を介して渡されたコンテキスト設定情報に含まれるモジュール識別子の値が、自身のモジュールを示すモジュール識別子の値と一致しているか否かを判定する。この場合には、コンテキスト設定

情報に含まれるモジュール識別子が値「# 1」であり、モジュール（# 1）710を示すモジュール識別子が値「# 1」であるので、両者が一致する。そのため、コンテキスト設定部702Aは、自身のモジュール内で発行されたコンテキスト登録の時点まで戻るべきと判断できる。

[0345] 処理はステップS707-2に移行され、コンテキスト設定部702Aは、コンテキスト設定情報からコンテキスト番号を取り出し、コンテキスト管理テーブル919から、獲得したコンテキスト番号に対応するコンテキスト情報を取得する。そして、コンテキスト設定部702Aは、コンテキスト管理テーブル919から、獲得したコンテキスト情報を削除する。そして、次のステップS707-3で、コンテキスト設定部702Aは、ステップS707-2で取得したコンテキスト情報を、モジュール（# 1）710における現在のコンテキスト情報に対して上書きし、コンテキスト登録を行った場所まで戻る。

[0346] このように、本第5の実施形態では、コンテキスト登録において、コンテキストの代わりにコンテキスト設定情報としてモジュール識別子とコンテキスト番号を提供する。そして、コンテキスト設定において、コンテキスト登録で提供されたコンテキスト設定情報に含まれるモジュール識別子とコンテキスト番号とを指定し、呼び出し元に順番に戻っていく。これにより、個別にコンテキストを持ち保護されたモジュール間であっても、関数 `set jmp` / `long jmp` に相当する機能を実現できる。

[0347] なお、本第5の実施形態では、コンテキスト変更通知部と切替前処理部とを別々に扱ってきたが、これはこの例に限られず、切替前処理部がコンテキスト変更通知部の機能を含むように構成することもできる。すなわち、切替処理部の内部で、通常呼び出しの場合には通常の切り替え前処理を行い、コンテキスト設定であればコンテキスト変更通知部の処理を行う。

[0348] また、本第5の実施形態では、コンテキスト変更受付部と切替後処理部とを別々に扱ってきたが、切替後処理部がコンテキスト変更受付部の機能を含むように構成することもできる。すなわち、切替後処理部の内部で、通常呼

び出しの場合には通常の切り替え切替後を行い、コンテキスト設定であればコンテキスト変更受付部の機能による処理を行う。

[0349] また、本第5の実施形態では、コンテキスト管理テーブルをモジュール単位で保持するように説明したが、これはこの例に限定されない。例えば、セキュア共有領域と同様に、連携するモジュール間で共有する領域にコンテキスト管理テーブルを設けるようにすることも可能である。

[0350] さらに、本第5の実施形態では、モジュール識別子とコンテキスト番号とによりコンテキスト情報を管理するように説明したが、これはこの例に限定されない。すなわち、このモジュール識別子とコンテキスト番号は、モジュール間でユニークな値を持つためのものであればよく、2つの番号を持たない方法が考えられる。例えば、初期化処理において、モジュール間で利用するコンテキスト番号の範囲を決定し、モジュール#1はコンテキスト番号#1～#100を、モジュール#2はコンテキスト番号#101～#200を利用して、コンテキスト番号のみを利用する方法が考えられる。

[0351] また、各モジュールが利用するコンテキスト番号は、初期化処理で決定するだけでなく、各モジュールが利用しているコンテキスト番号から衝突しないように調整して、コンテキスト番号を決定する方法であってもよい。

[0352] <各実施形態の他の対応>

上述の各実施形態の他の対応によれば、プログラムは、第1処理ステップが自モジュールから他のモジュールへの動作の切替を行う直前に、切替の要因が呼び出しによるものである場合に、呼び出し元のアドレスを切替履歴領域に保存し、第2処理ステップが他のモジュールからの動作の切替が行われた直後に、切替フラグが呼び出し復帰を示す場合に、切替履歴領域から呼び出し元のアドレスを取得し、そのアドレスから実行を再開することを特徴とする。

[0353] また、プログラムは、初期化ステップがモジュールの識別子をマイクロプロセッサがモジュールを識別するためのタスク識別子を用いて決定することを特徴とする。

- [0354] また、プログラムは、初期化ステップがモジュールの識別子を乱数を用いて決定すると共に、連携するモジュール間で識別子が重複しないことを確認することを特徴とする。
- [0355] また、プログラムは、第 1 処理ステップが自モジュールから他のモジュールへの切替を行う直前に、共有鍵を指定して切替フラグおよび切替パラメータをそれぞれ暗号化して共有領域に格納し、第 2 処理ステップが他のモジュールから自モジュールへの切替が行われた直後に、共有鍵を指定して共有領域に格納される切替フラグおよび切替パラメータを復号し、復号された切替フラグおよび切替パラメータとに従い自モジュール内の機能を実行させることを特徴とする。
- [0356] また、共有領域と共有領域へのアクセスを制御するアクセス制御部がマイクロプロセッサ上に設けられ、プログラムは、アクセス制御部を介して共有領域にアクセスすることを特徴とする。
- [0357] また、共有領域と共有領域へのアクセスを制御するアクセス制御部が、マイクロプロセッサ上で動作し、連携するモジュールを複数動作させる仮想マシンモニタ上に設けられ、プログラムは、仮想マシンモニタからアクセス制御部を介して共有領域にアクセスすることを特徴とする。

符号の説明

- [0358] 104A, 104B, 104C 切替前処理部
106A, 106B, 106C, 906A, 906B 切替後処理部
111, 121 初期化処理部
112, 912 プログラム（#1）本体
110, 310, 510, 710, 910 モジュール（#1）
118, 128 切替履歴領域
120, 320, 520, 720, 920 モジュール（#2）
122, 922 プログラム（#2）本体
130, 330, 530, 730, 930 モジュール（#3）
140, 340, 540, 740, 940 モジュール（#4）

- 1 8 0 O S
- 1 8 1 セキュア共有領域
- 1 8 2 実行モジュール識別子フィールド
- 1 8 3 切替フラグフィールド
- 1 8 4 切替パラメータフィールド
- 1 8 5 切替元モジュール識別子
- 1 9 0 共有領域
- 2 0 0, 4 0 0 セキュアプロセッサ
- 2 1 0, 4 1 0 コア
- 2 1 2, 4 1 2 現タスク識別子レジスタ
- 2 2 0 暗号復号管理部
- 2 2 2 鍵テーブル
- 2 2 4 セレクタ
- 2 2 6 暗号／復号器
- 2 8 0 メモリ
- 4 2 0 モジュール切替管理部
- 4 2 2 モジュール切替管理テーブル
- 4 2 4 テーブルアクセス制御部
- 5 0 5 A, 5 0 5 B 例外通知部
- 5 0 7 A, 5 0 7 B 例外受付部
- 7 0 2 A, 7 0 2 B, 9 0 9 A, 9 0 9 B コンテキスト設定部
- 9 0 5 A, 9 0 5 B コンテキスト変更通知部
- 9 0 7 A, 9 0 7 B コンテキスト変更受付部
- 7 1 9, 7 2 9, 9 1 9, 9 2 9 コンテキスト管理テーブル
- 9 0 8 コンテキスト登録部

請求の範囲

- [請求項1] コンピュータに実行させるための複数のモジュールを含むプログラムであって、
- 前記コンピュータは、
- 連係して動作するモジュールのみがアクセス可能な領域であって、該連係して動作するモジュールのうち、OS上で動作するモジュールの識別子を示す実行モジュール識別子を格納する共有領域を含むメモリを備え、
- 前記モジュールのそれぞれは、
- 連携する他のモジュールへの動作の切替を行う直前に、前記他のモジュールの識別子を前記実行モジュール識別子として前記共有領域に格納する第1処理ステップと、
- 前記他のモジュールからの動作の切替が行われた直後に、前記共有領域に格納されている前記実行モジュール識別子と自モジュールの識別子とが一致した場合に、該自モジュール内の機能を実行させる第2処理ステップと
- を有する
- ことを特徴とするプログラム。
- [請求項2] 前記第1処理ステップは、
- 少なくとも、前記切替の要因を示す切替フラグと、該切替に必要な情報を含む切替パラメータとを前記共有領域にさらに格納し、
- 前記第2処理ステップは、
- 前記共有領域に格納される前記切替フラグと前記切替パラメータとに従い前記自モジュール内の機能を実行させる
- ことを特徴とする請求項1に記載のプログラム。
- [請求項3] 前記第1処理ステップは、
- 前記切替の要因が呼び出しである場合に、該呼び出しを示す前記切替フラグと、該呼び出しにより前記切替が行われる前記他のモジュール

ル内の呼び出し位置を示す呼び出し情報を少なくとも含む前記切替パラメータとを該共有領域に格納し、

前記第 2 処理ステップは、

前記切替フラグが前記呼び出しを示す場合に、前記切替パラメータに含まれる前記呼び出し情報に従い前記自モジュール内で呼び出しを行う

ことを特徴とする請求項 2 に記載のプログラム。

[請求項 4]

前記第 1 処理ステップは、

前記切替の要因が呼び出しからの復帰である場合に、該復帰を示す前記切替フラグと、該呼び出しにより前記切替が行われた前記自モジュールの実行結果を示す呼び出し復帰情報を少なくとも含む前記切替パラメータとを前記共有領域に格納し、

前記第 2 処理ステップは、

前記切替フラグが前記復帰を示す場合に、前記切替パラメータに含まれる前記呼び出し復帰情報を用いて前記自モジュールが前記呼び出しを行われた位置に戻って実行を再開させる

ことを特徴とする請求項 3 に記載のプログラム。

[請求項 5]

連携する他のモジュールへの動作の切り替えを行う直前に、前記切替の要因が前記呼び出しまたは前記復帰以外の場合に、該呼び出しまたは復帰以外を示す前記切替フラグと、該切替の要因の内容を示す切替要因情報を少なくとも含む前記切替パラメータとを前記共有領域に格納する第 3 処理ステップと、

前記他のモジュールからの動作の切り替えが行われた直後に、前記切替フラグが前記呼び出しまたは復帰以外を示す場合に、前記切替パラメータに含まれる前記切替要因情報が示す前記要因に対応する処理を実行する第 4 処理ステップと

をさらに有する

ことを特徴とする請求項 4 に記載のプログラム。

[請求項6]

前記第3処理ステップは、

前記切替の要因が言語例外である場合に、該言語例外を示す前記切替フラグと、該言語例外を通知するために必要な言語例外情報を少なくとも含む前記切替パラメータとを前記共有領域に格納し、

前記第4処理ステップは、

前記切替フラグが前記言語例外を示す場合に、前記切替パラメータに含まれる前記言語例外情報を用いて前記自モジュールに対して言語例外を発生させる

ことを特徴とする請求項5に記載のプログラム。

[請求項7]

前記識別子と、コンテキスト毎にユニークなコンテキスト番号と、該コンテキストを示すコンテキスト情報とを1対1に対応付けて管理する管理テーブルから、前記他のモジュールにより指定される該コンテキスト番号に対応する該コンテキスト情報を取得し、取得した該コンテキスト情報で前記自モジュールの現在のコンテキストを置き換える設定ステップをさらに有し、

前記第3処理ステップは、

前記切替の要因がコンテキスト設定である場合に、前記切替が行われる直前に、前記管理テーブルから、前記識別子と前記コンテキスト番号とに対応する前記コンテキスト情報を削除すると共に、コンテキスト設定を示す前記切替フラグと、前記設定ステップにより設定されるコンテキストを示すコンテキスト設定情報を少なくとも含む前記切替パラメータとを前記共有領域に格納し、

前記第4処理ステップは、

前記切替が行われた直後に、前記切替フラグが前記呼び出しを示し、且つ、前記切替パラメータが前記コンテキスト設定情報を含む場合に、前記他のモジュールを示す前記識別子と、該他のモジュールにより指定される前記コンテキスト番号と、前記第1処理ステップに戻るためのコンテキスト情報とを前記管理テーブルに登録し、

前記切替フラグがコンテキスト設定を示す場合には、前記切替パラメータに含まれる前記コンテキスト設定情報を用いて設定ステップに制御を移す

ことを特徴とする請求項 6 に記載のプログラム。

[請求項 8]

登録部が、前記識別子と、コンテキスト毎にユニークなコンテキスト番号と、該コンテキストを示すコンテキスト情報とを 1 対 1 に対応付けて管理する管理テーブルに対してコンテキスト情報を登録し、該コンテキスト情報に対応する該コンテキスト番号を返す登録ステップと、

前記自モジュールから指定される前記識別子と、該自モジュールの前記識別子とが一致する場合には、該自モジュールにより指定される前記コンテキスト番号に対応する前記コンテキスト情報を前記管理テーブルから取得し、取得した該コンテキスト情報で該自モジュールのコンテキストを置き換え、

前記自モジュールから指定される前記識別子と、該自モジュールの前記識別子とが一致しない場合には、該自モジュールのコンテキストを前記第 1 処理ステップの実行直前の時点のコンテキストに置き換える設定ステップと

をさらに備え、

前記第 3 処理ステップは、

前記切替の要因がコンテキスト設定である場合に、前記切替が行われる直前に、コンテキスト設定を示す前記切替フラグと、前記設定ステップにより設定されるコンテキストを示すコンテキスト設定情報を少なくとも含む前記切替パラメータとを前記共有領域に格納し、

前記第 4 処理ステップは、

前記切替フラグが前記コンテキスト設定を示す場合に、前記切替が行われた直後に、前記切替パラメータに格納された前記コンテキスト設定情報を用いて前記設定ステップに制御を移す

ことを特徴とする請求項 6 記載のプログラム。

[請求項 9]

前記共有領域は、切替元のモジュールを示す切替元識別子がさらに格納され、

前記第 1 処理ステップは、

前記切替の要因が呼び出しである場合に、前記切替が行われる直前に、前記自モジュールを示す前記識別子を前記切替元識別子として前記共有領域に格納し、

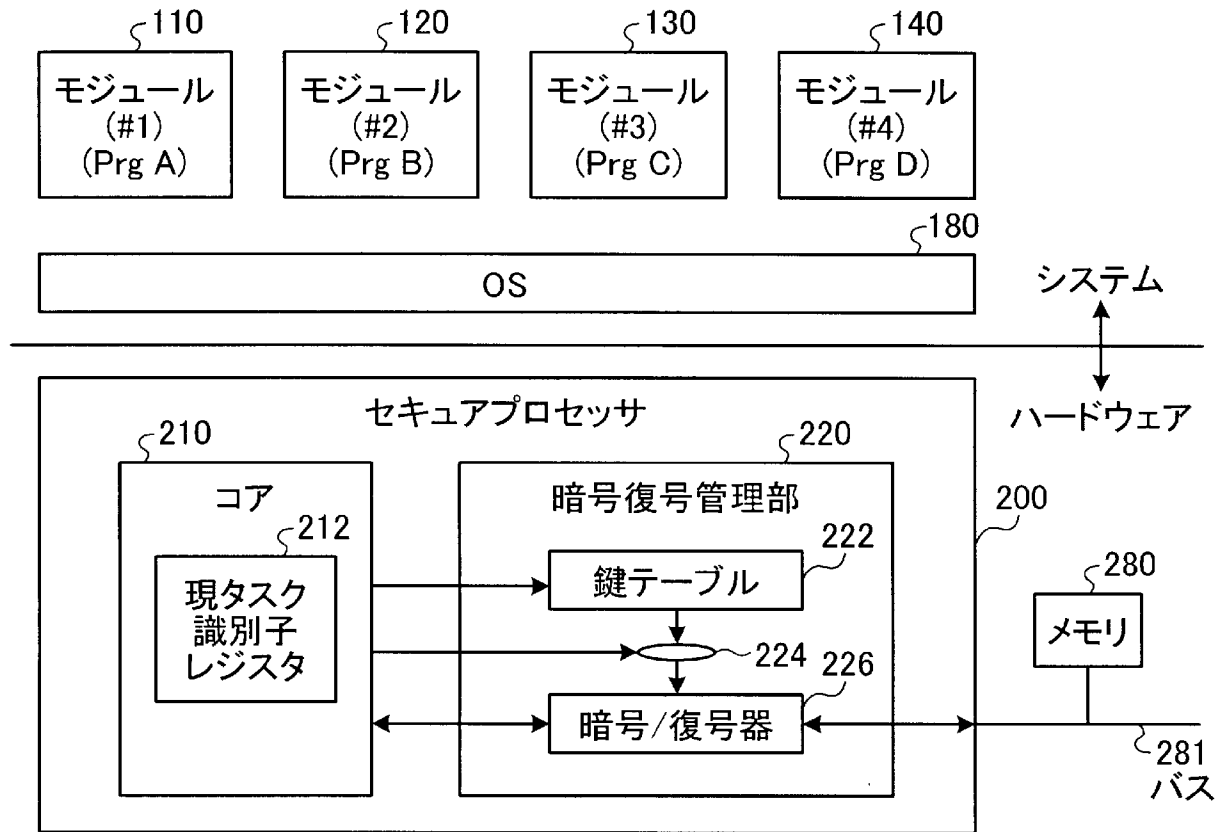
前記切替の要因が呼び出しからの復帰である場合に、前記切替による情報を保持する切替履歴領域から前記切替元のモジュールを示す前記識別子を取得して前記切替元識別子として前記共有領域に格納し、

前記第 2 処理ステップは、

前記切替フラグが前記呼び出しを示す値である場合に、前記切替が行われた直後に、前記共有領域から前記切替元識別子を取得して前記切替履歴領域に格納する

ことを特徴とする請求項 2 に記載のプログラム。

[図1]

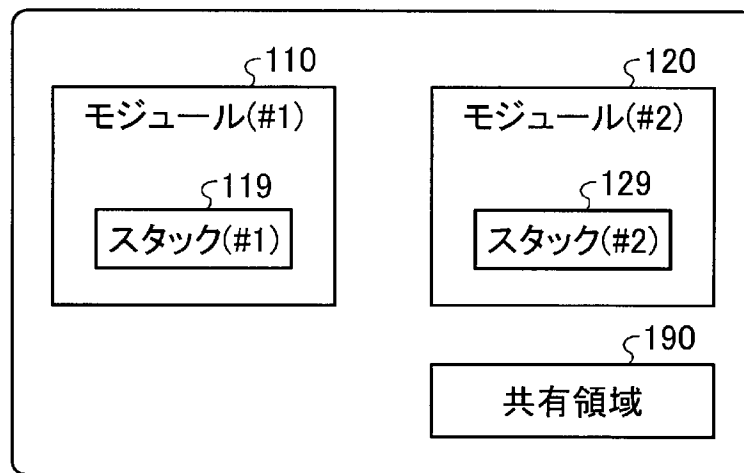


[図2]

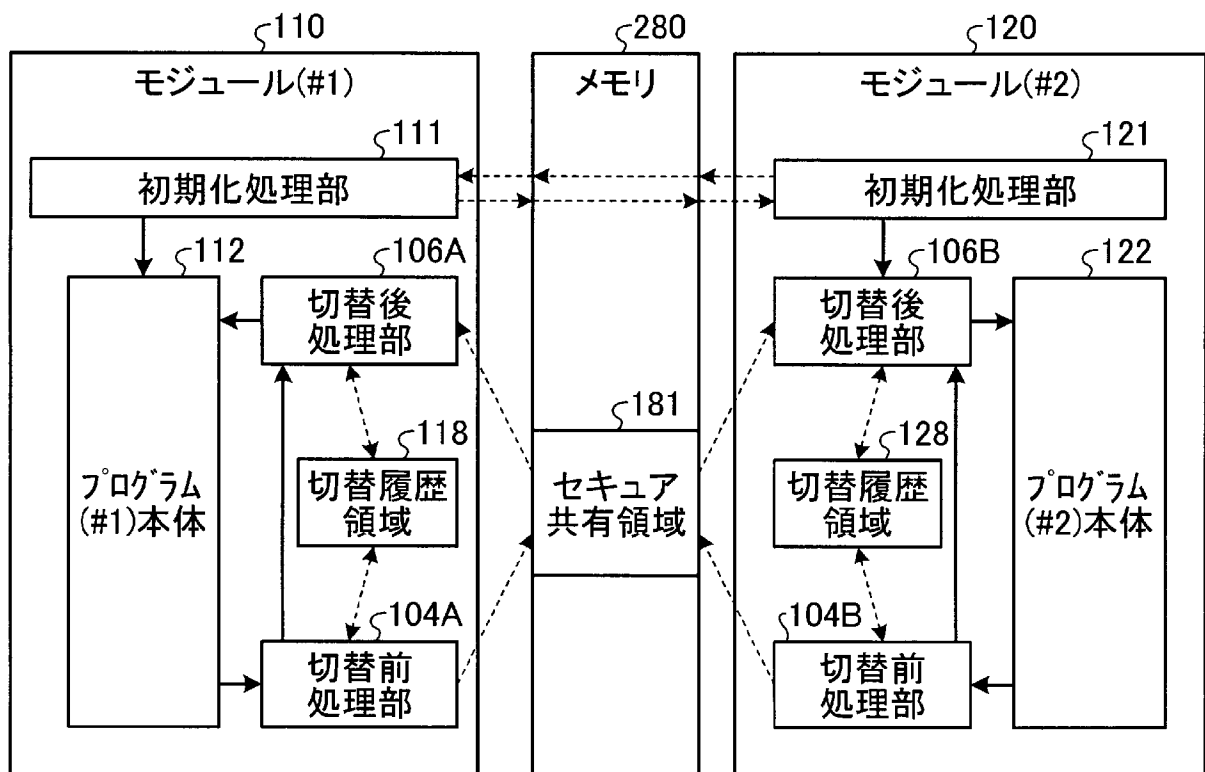
ID	鍵値	先頭アドレス	末尾アドレス
#1			
#2			
#3			
...			
#n			

222

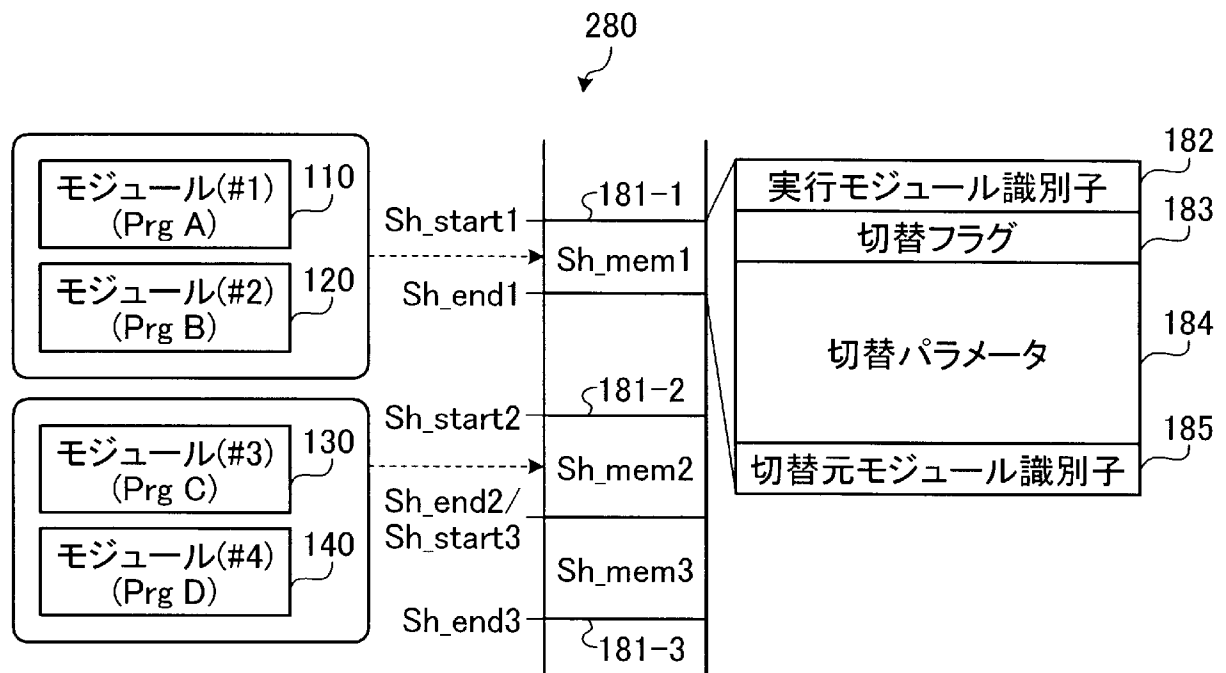
[図3]



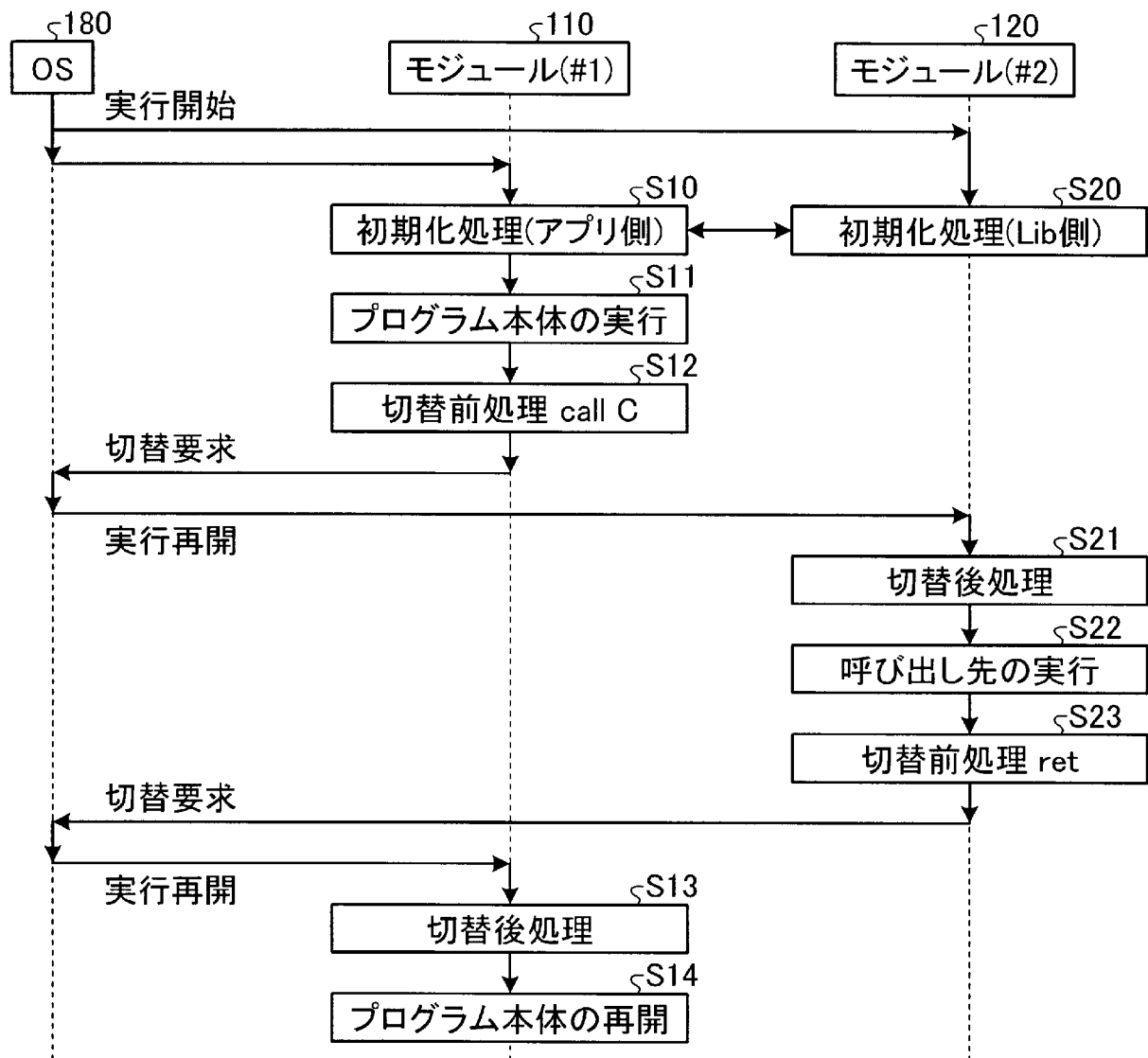
[図4]



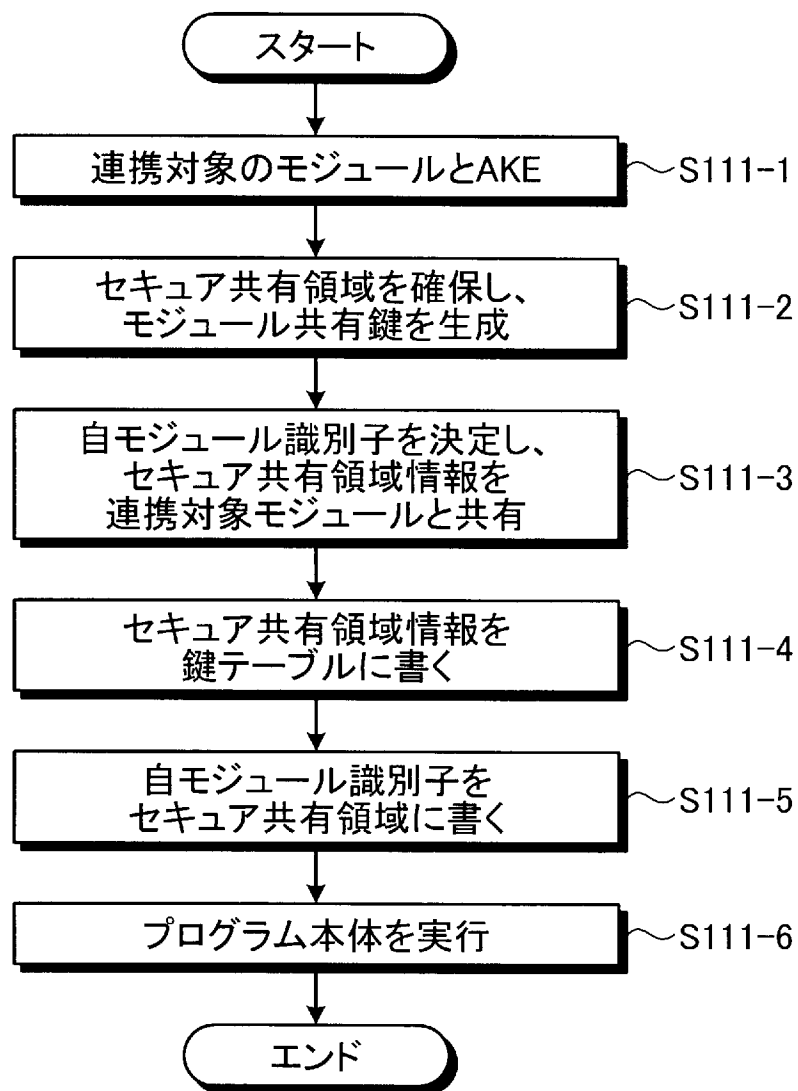
[図5]



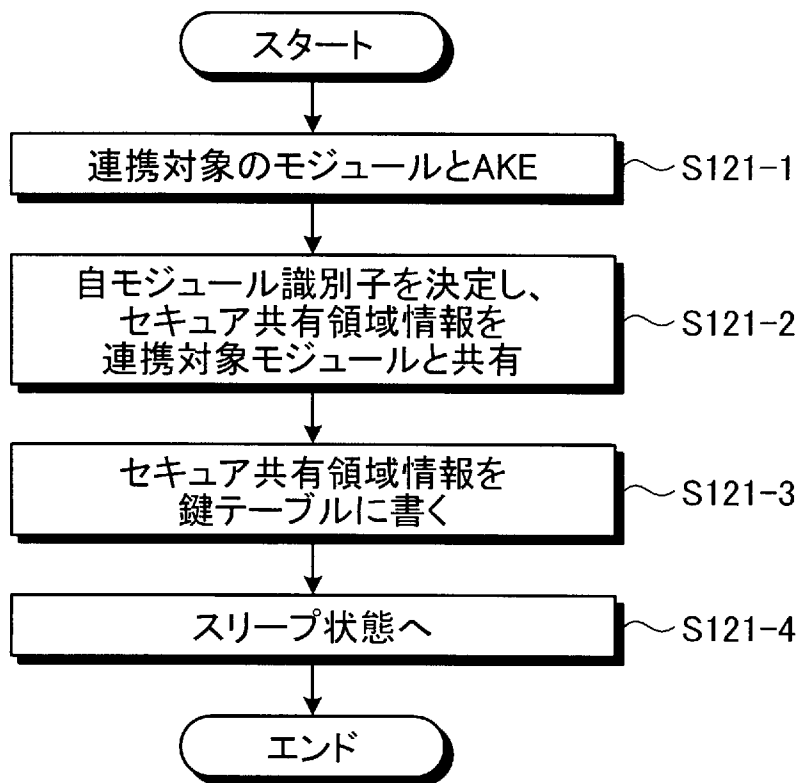
[図6]



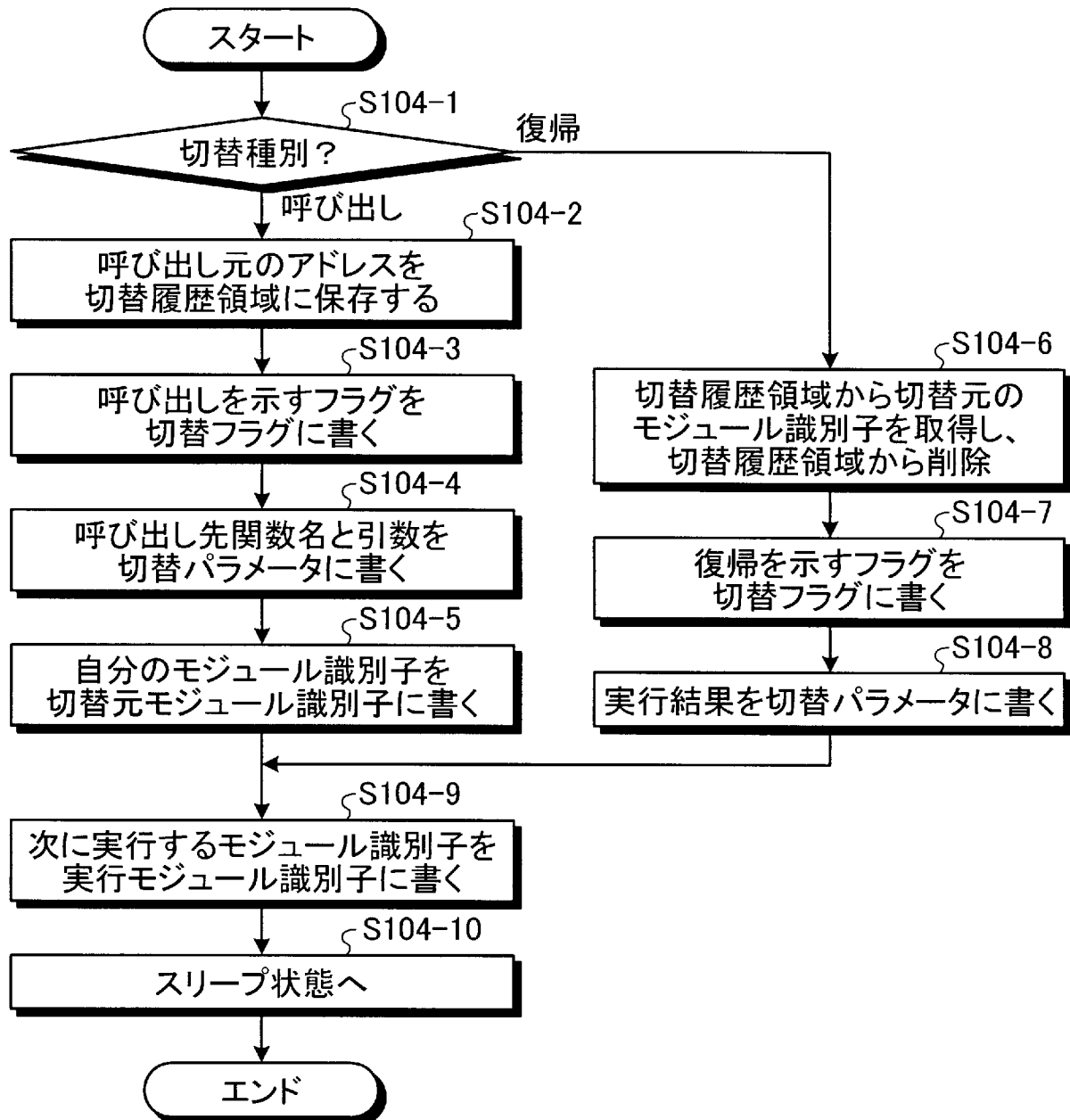
[図7]



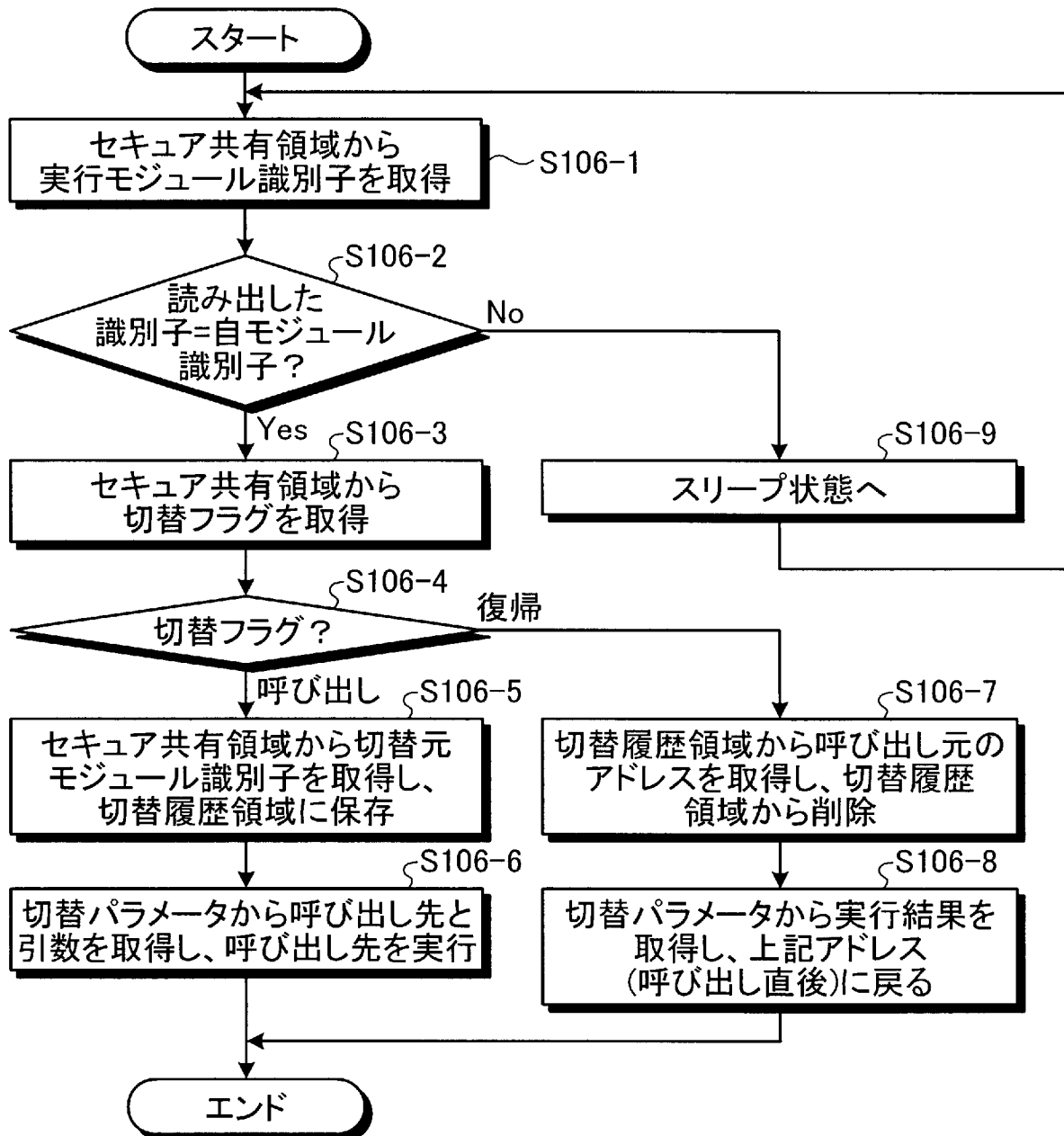
[図8]



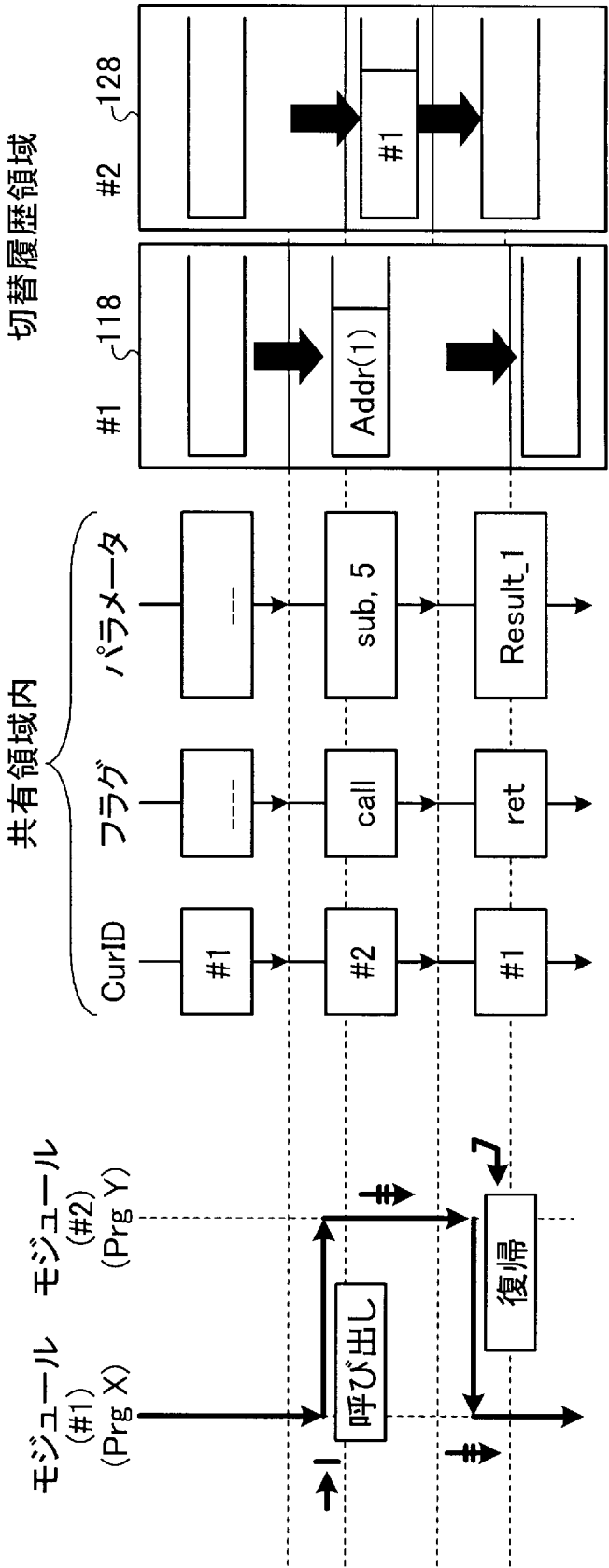
[図9]



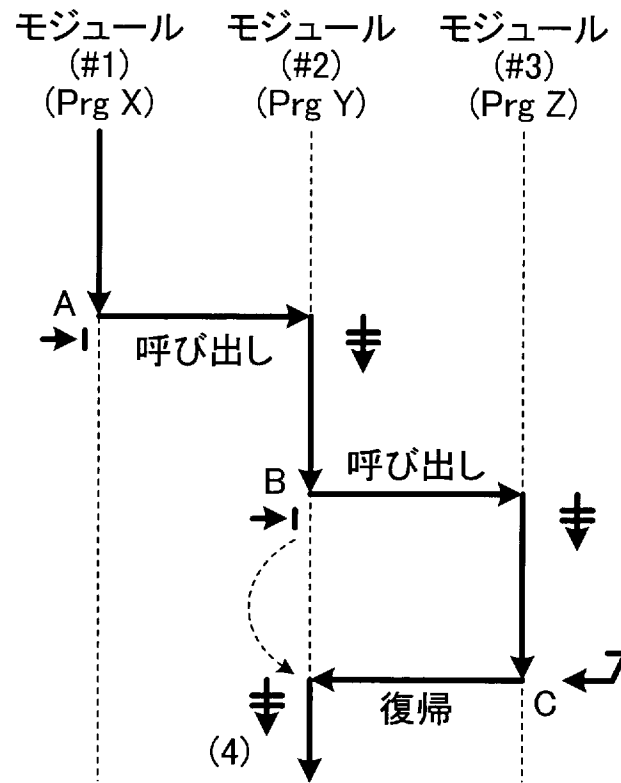
[図10]



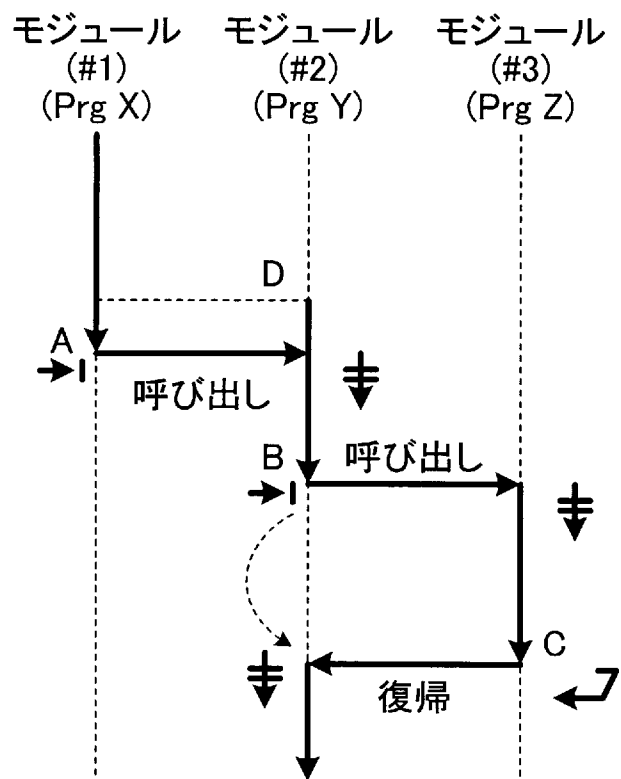
[図11]



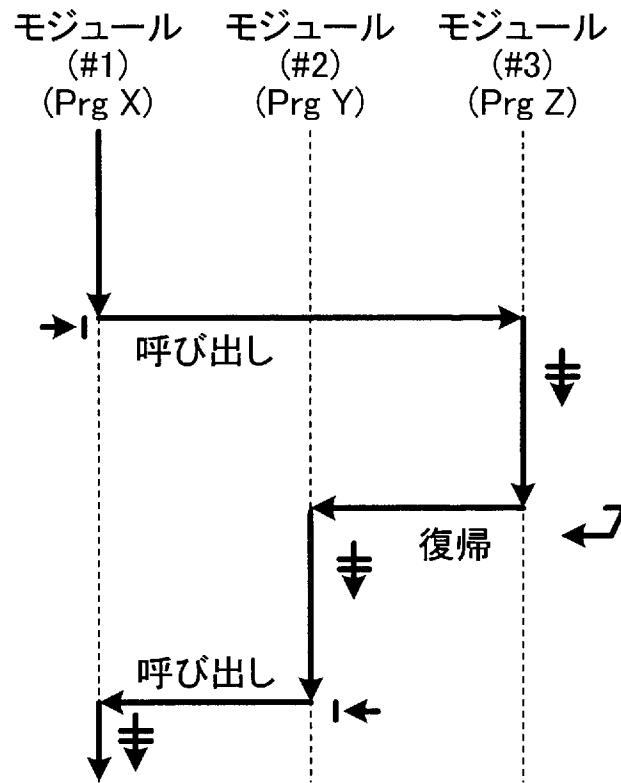
[図12]



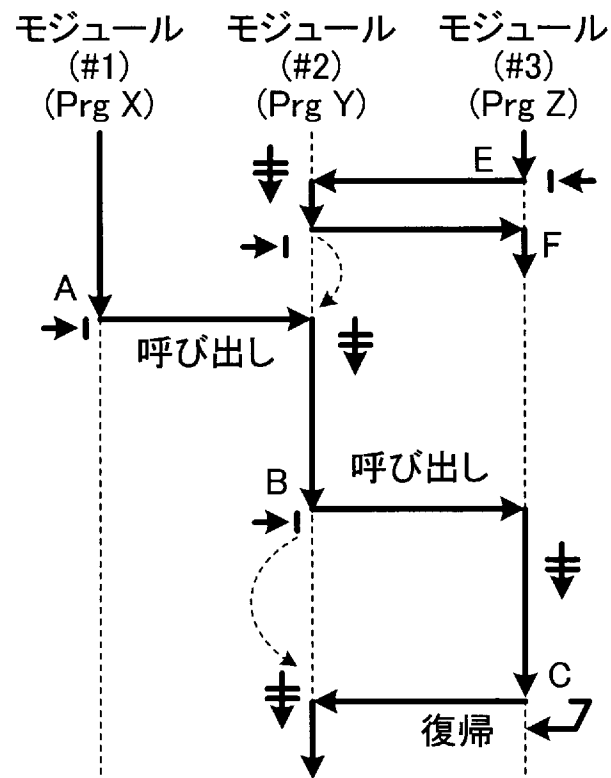
[図13]



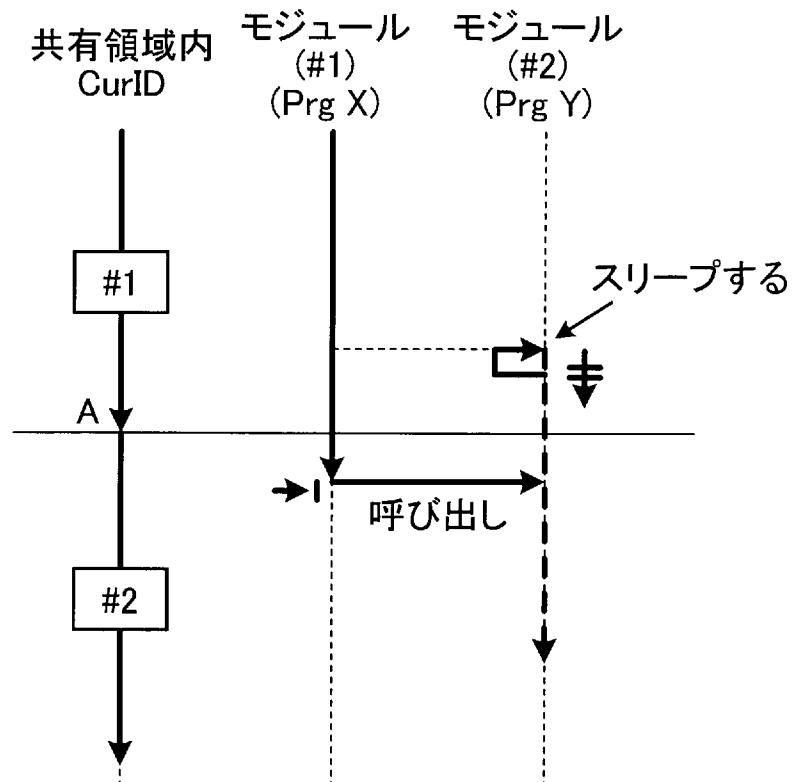
[図14]



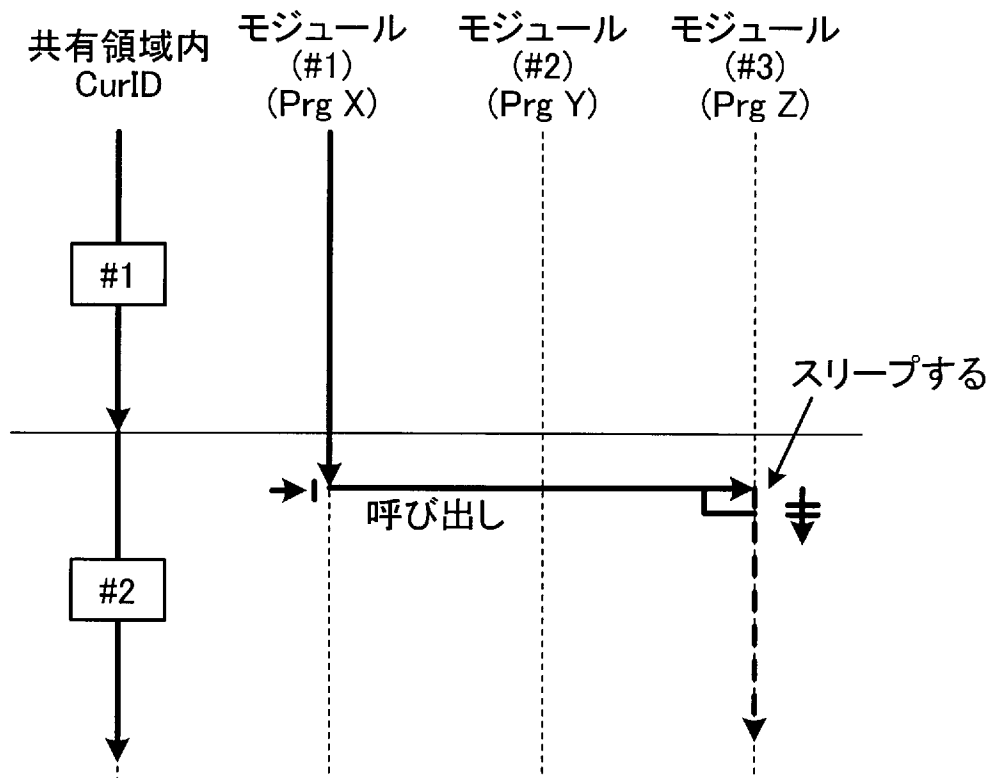
[図15]



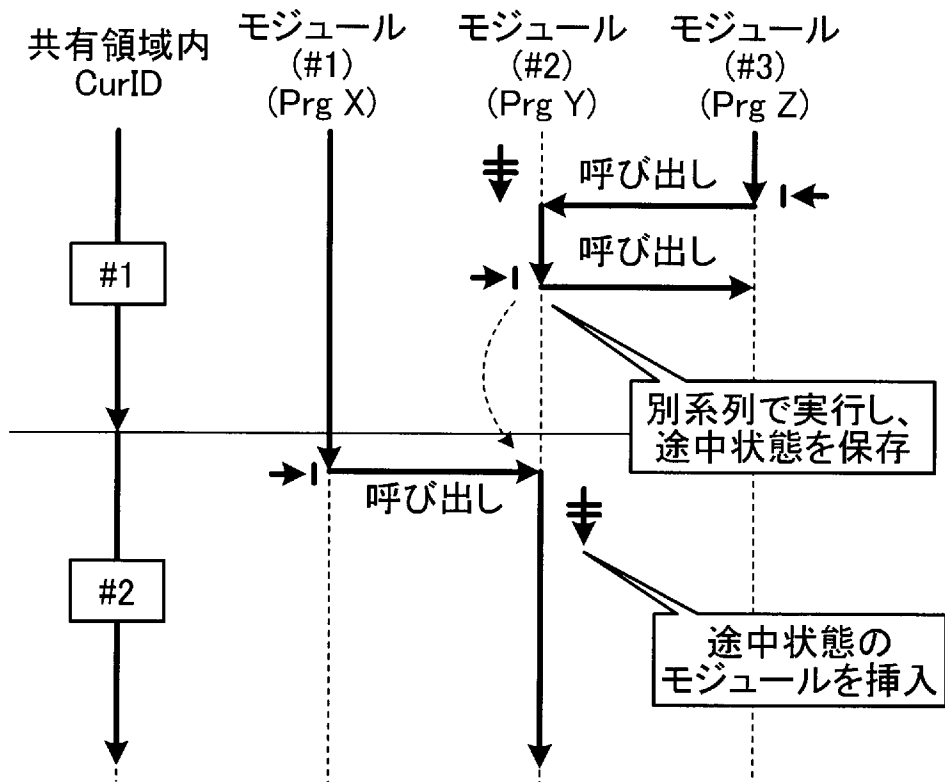
[図16]



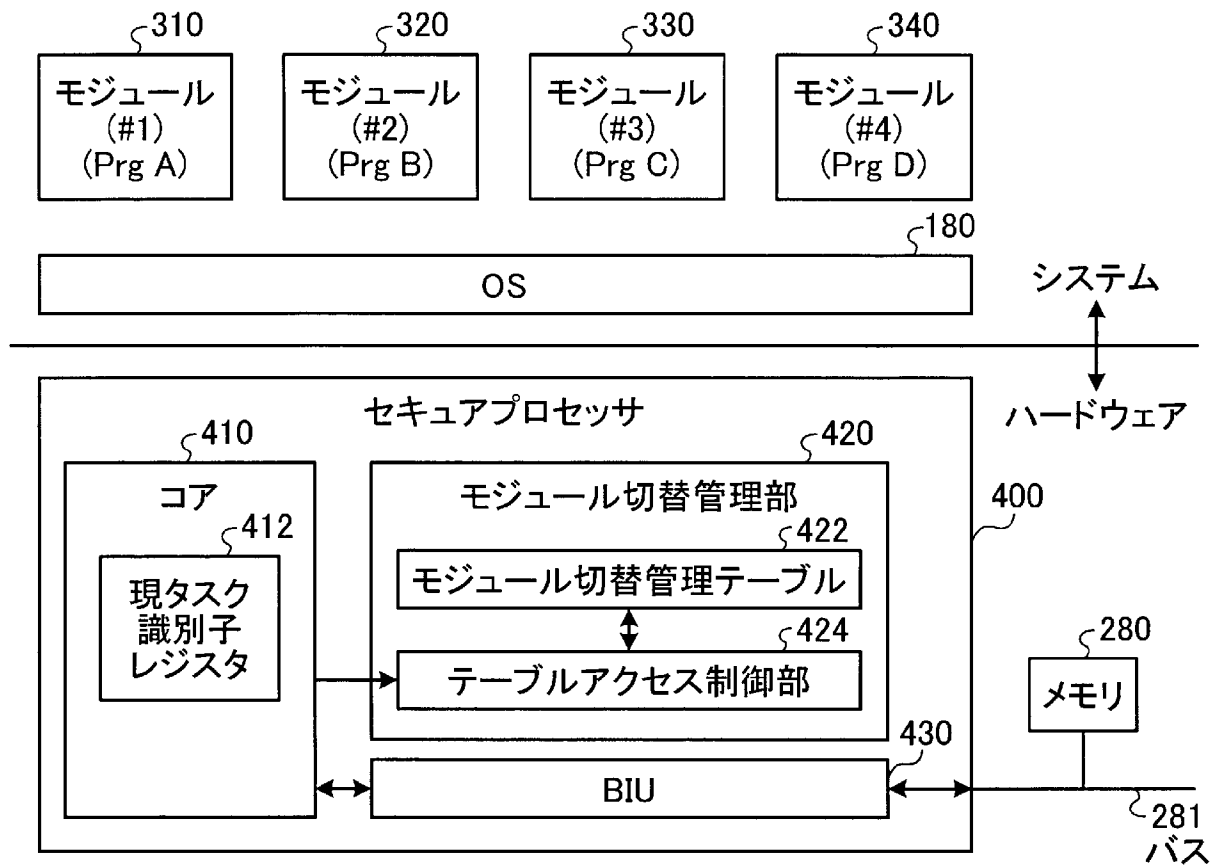
[図17]



[図18]



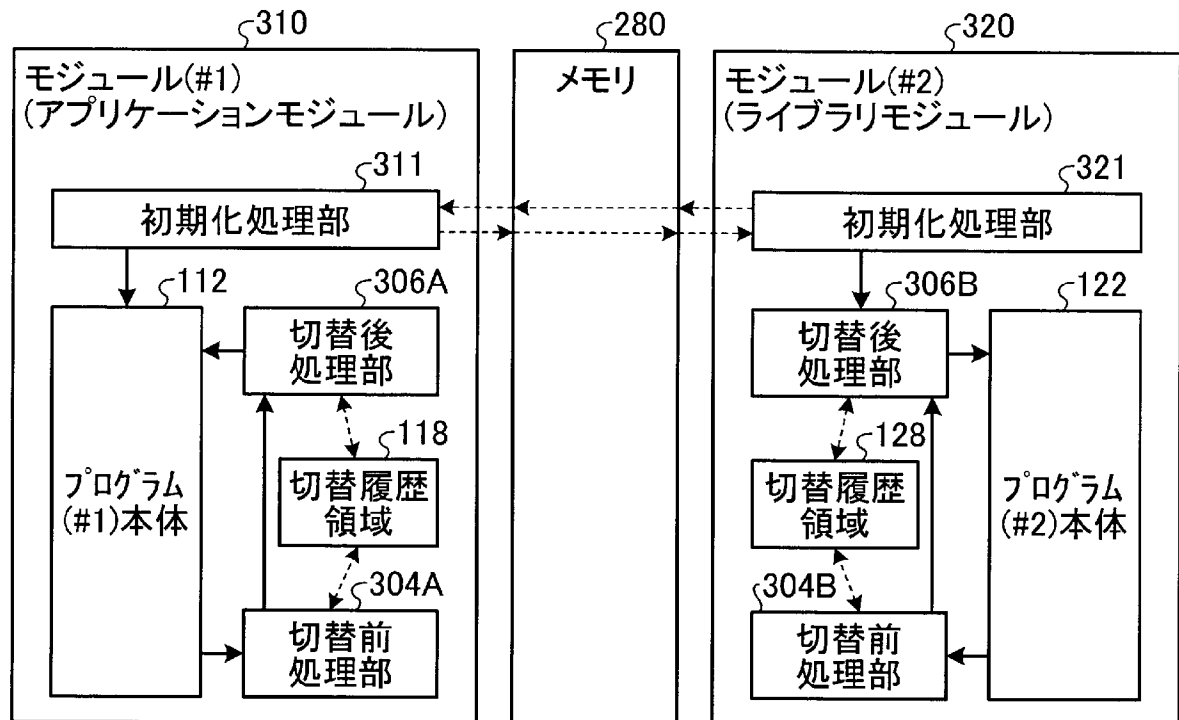
[図19]



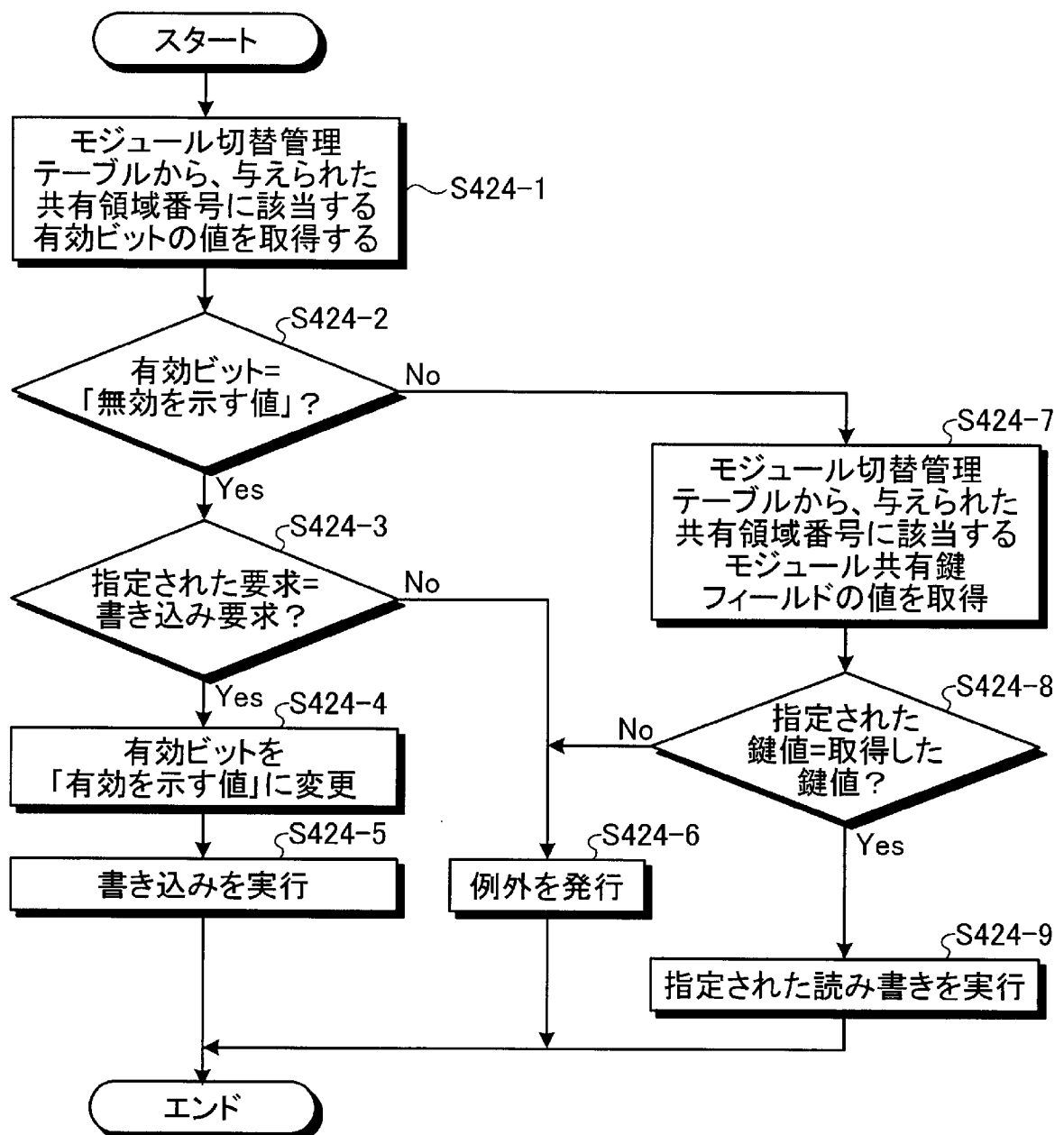
[図20]

422						
Idx	モジュール共有鍵	実行モジュール識別子	切替元モジュール識別子	切替フラグ	切替パラメータ	V
#1						
#2						
#3						
...						
#n						

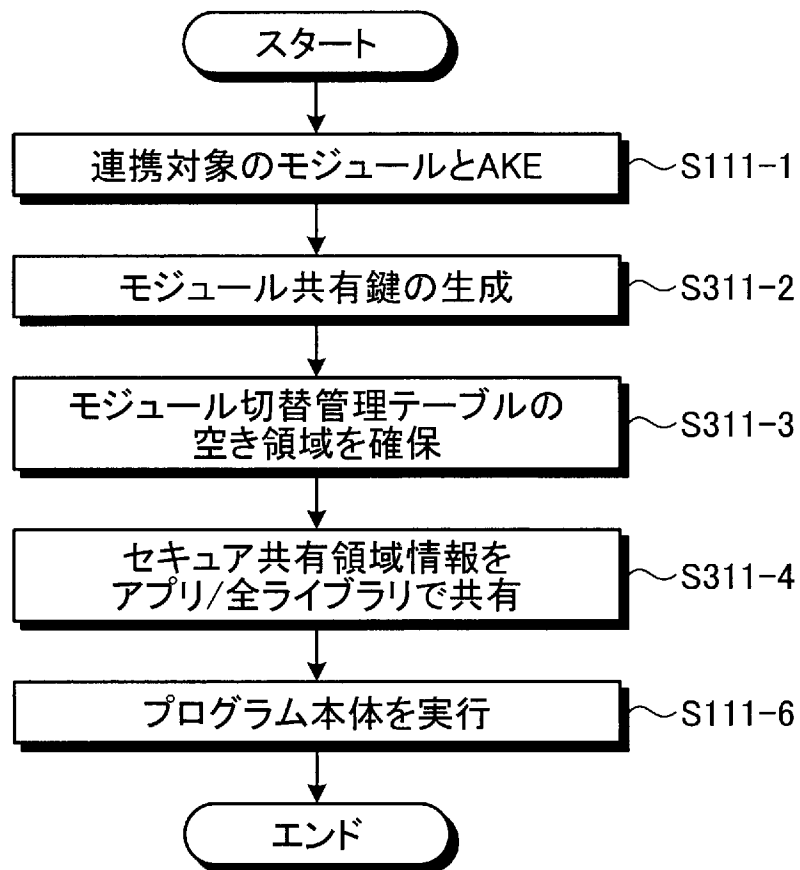
[図21]



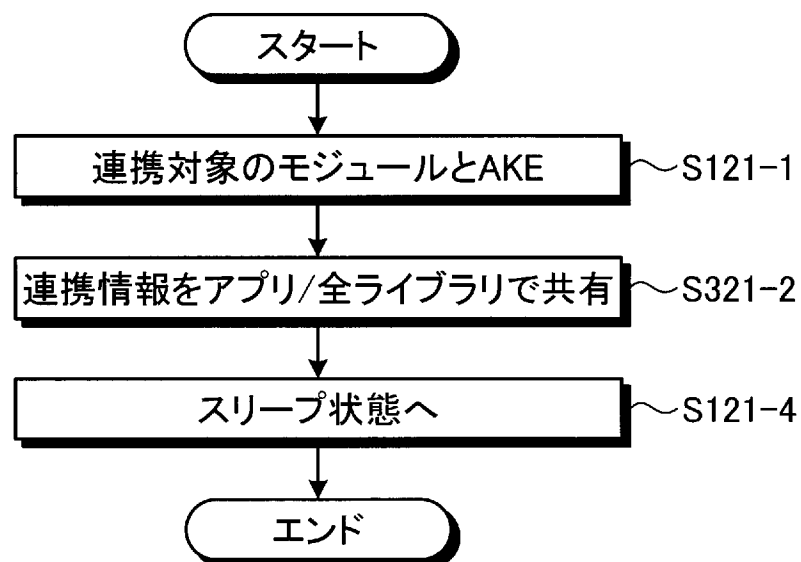
[図22]



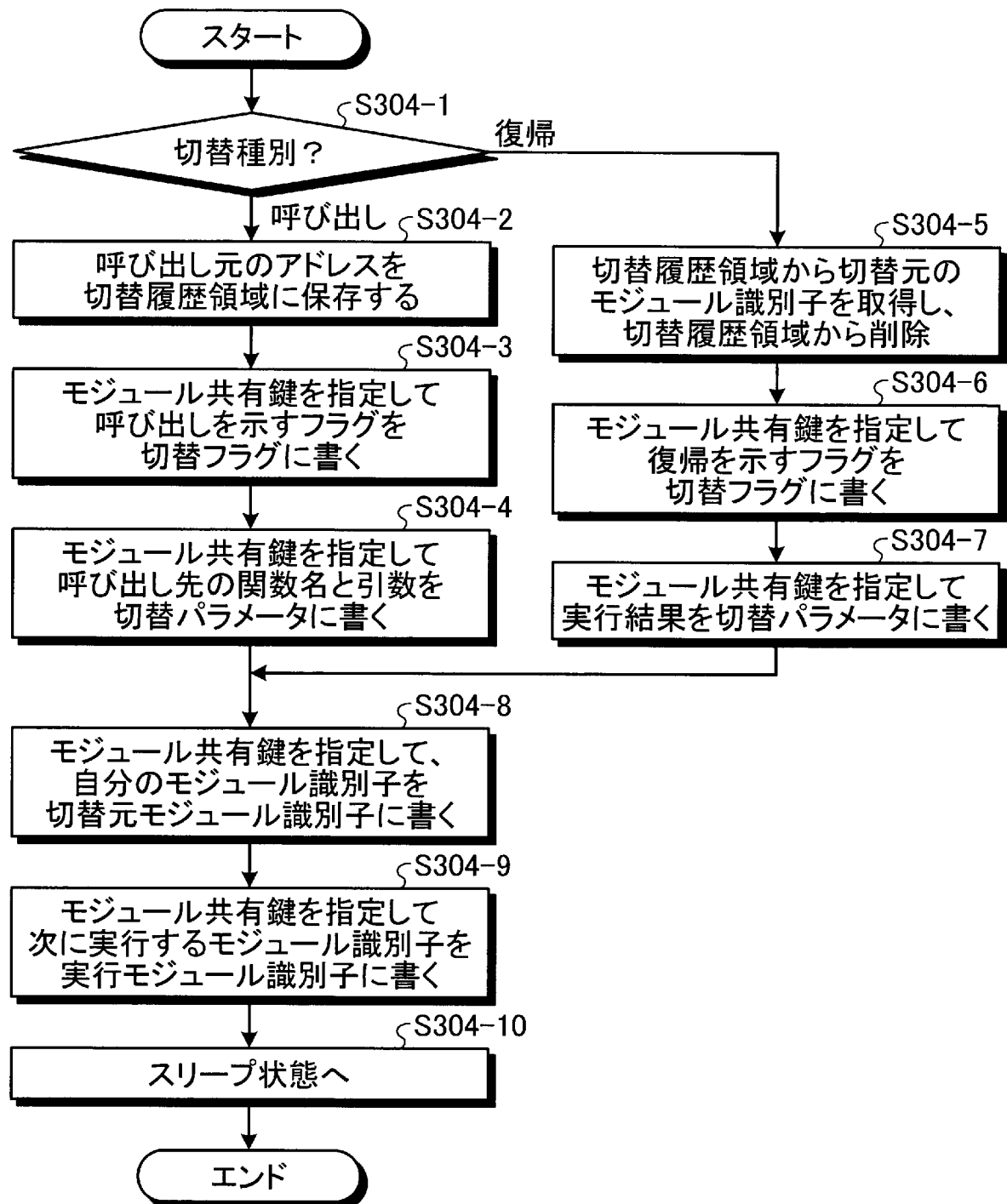
[図23]



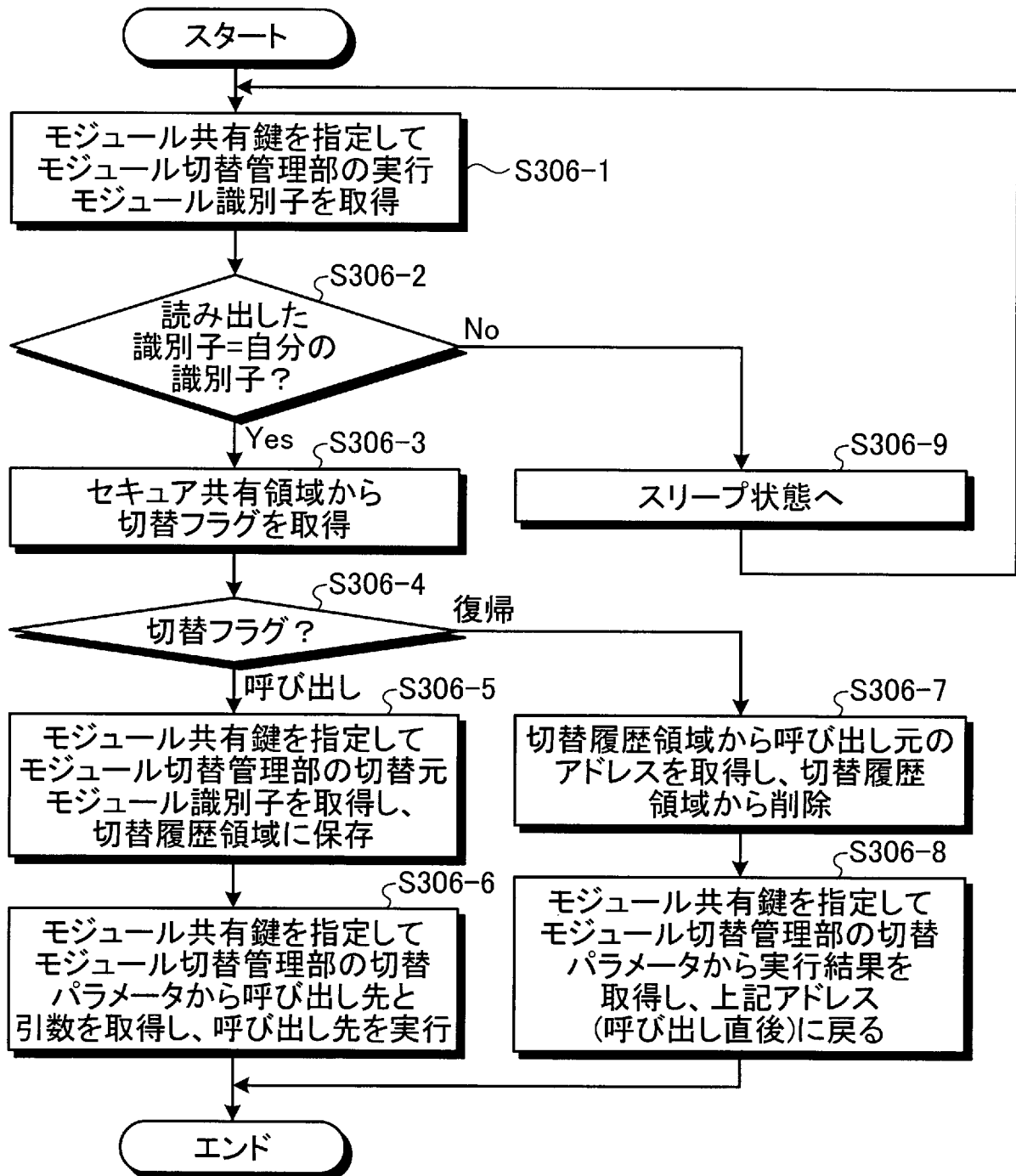
[図24]



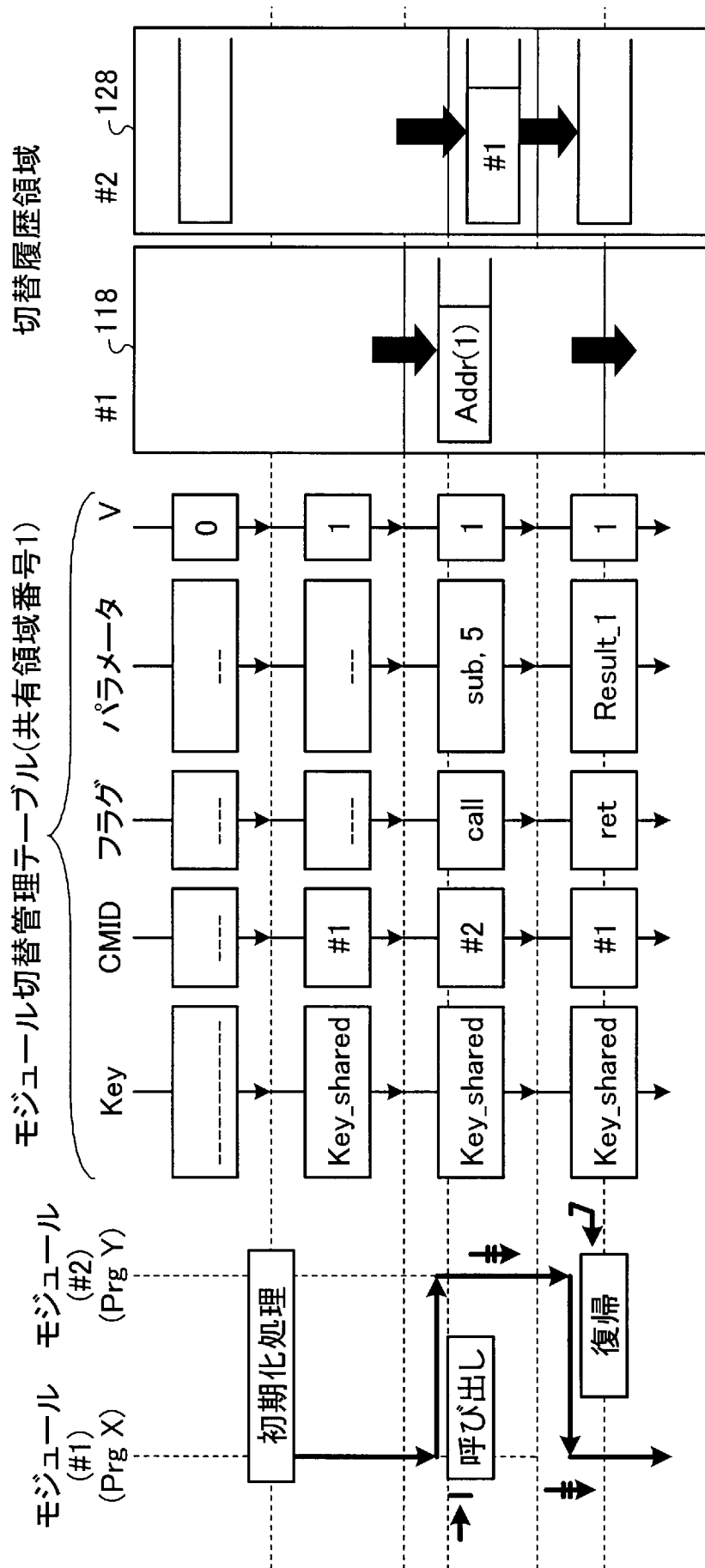
[図25]



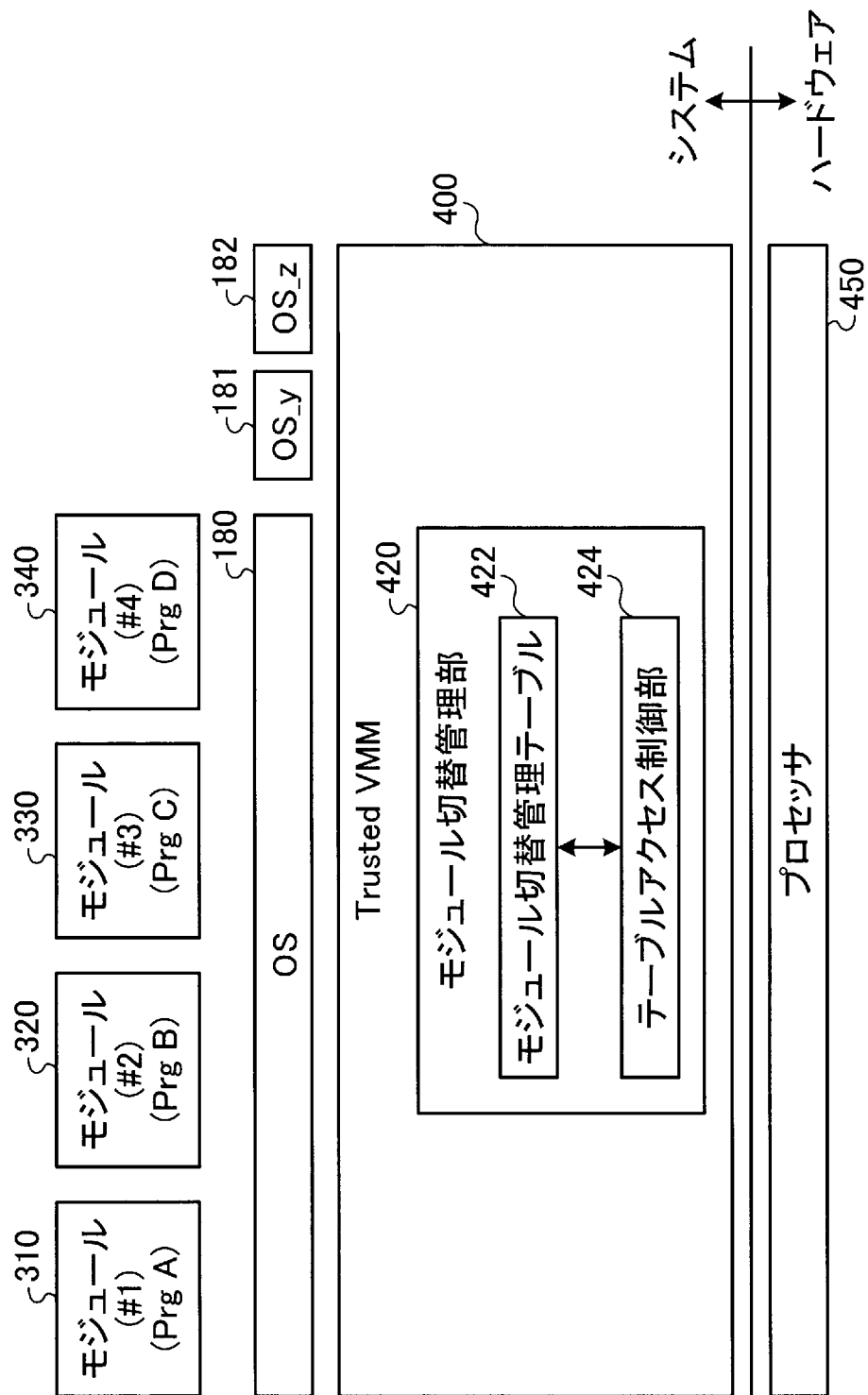
[図26]



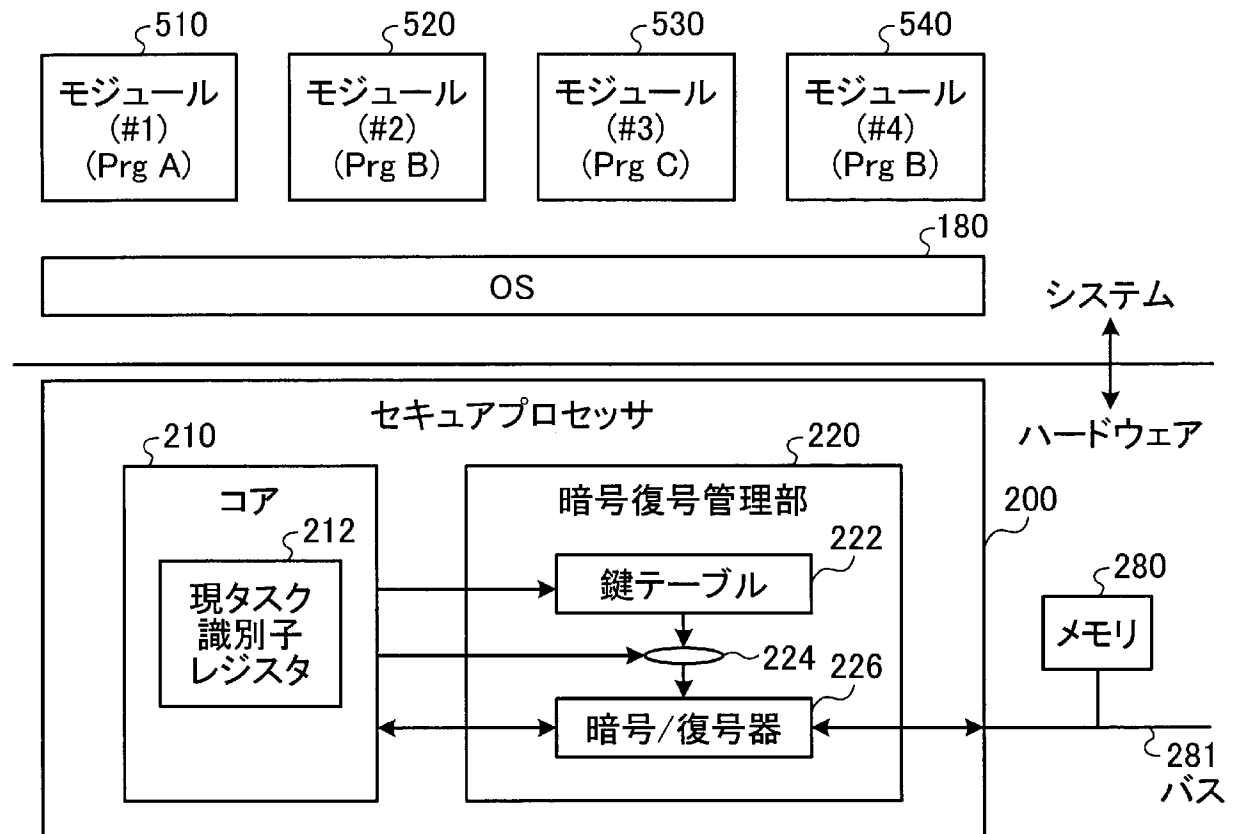
[図27]



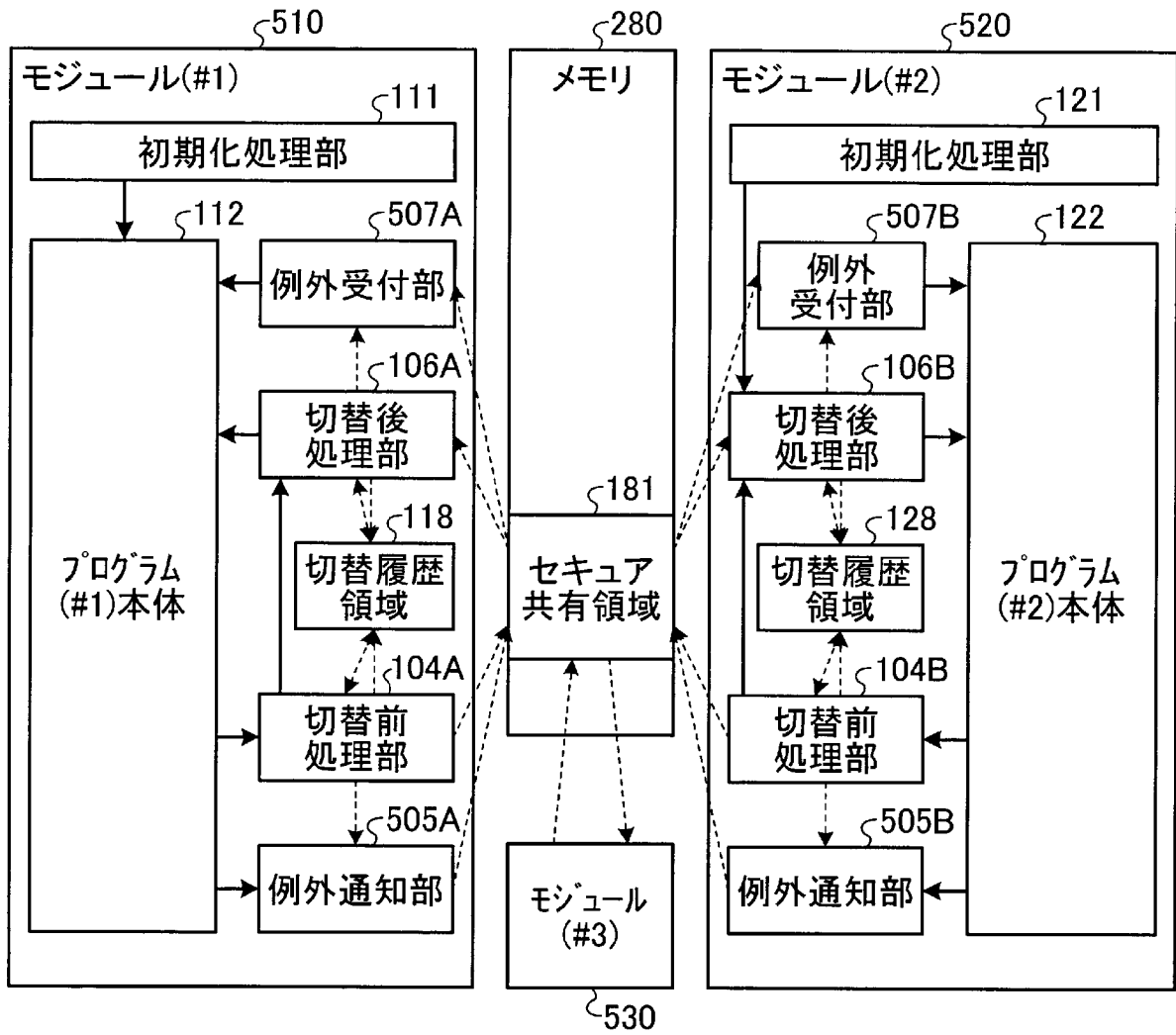
[図28]



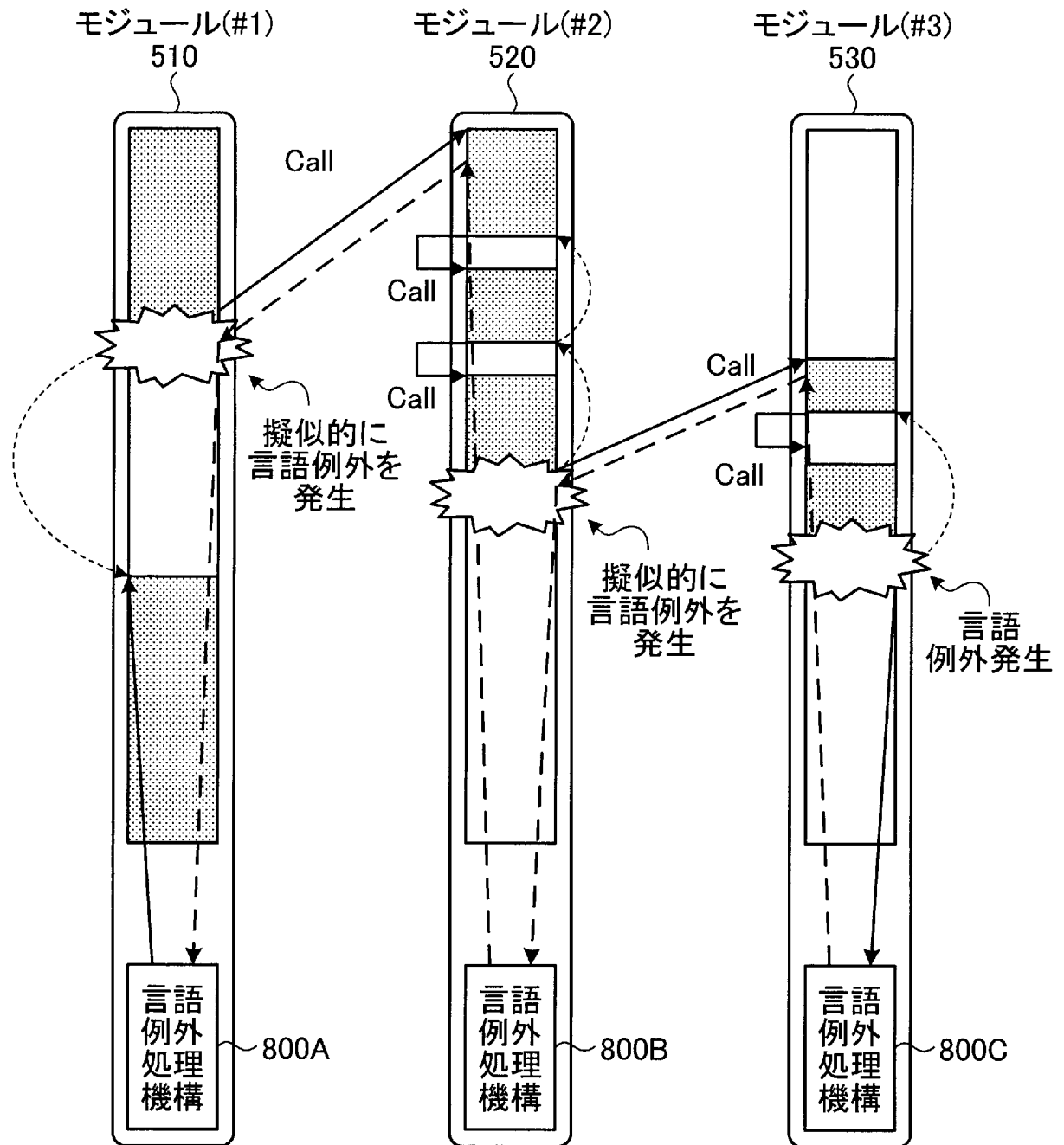
[図30]



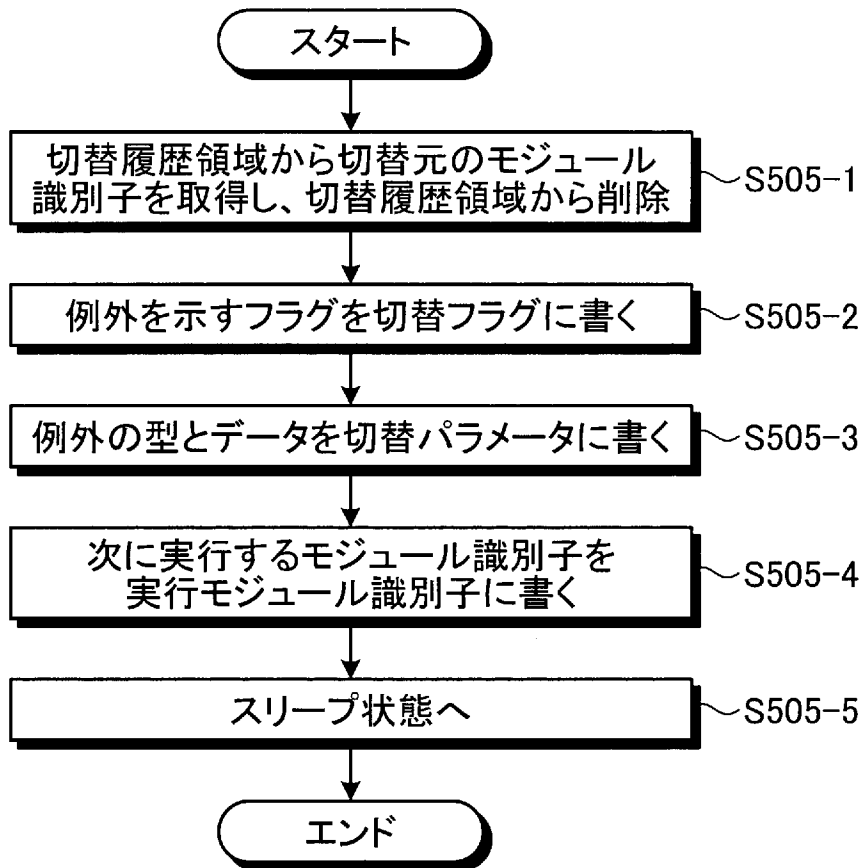
[図31]



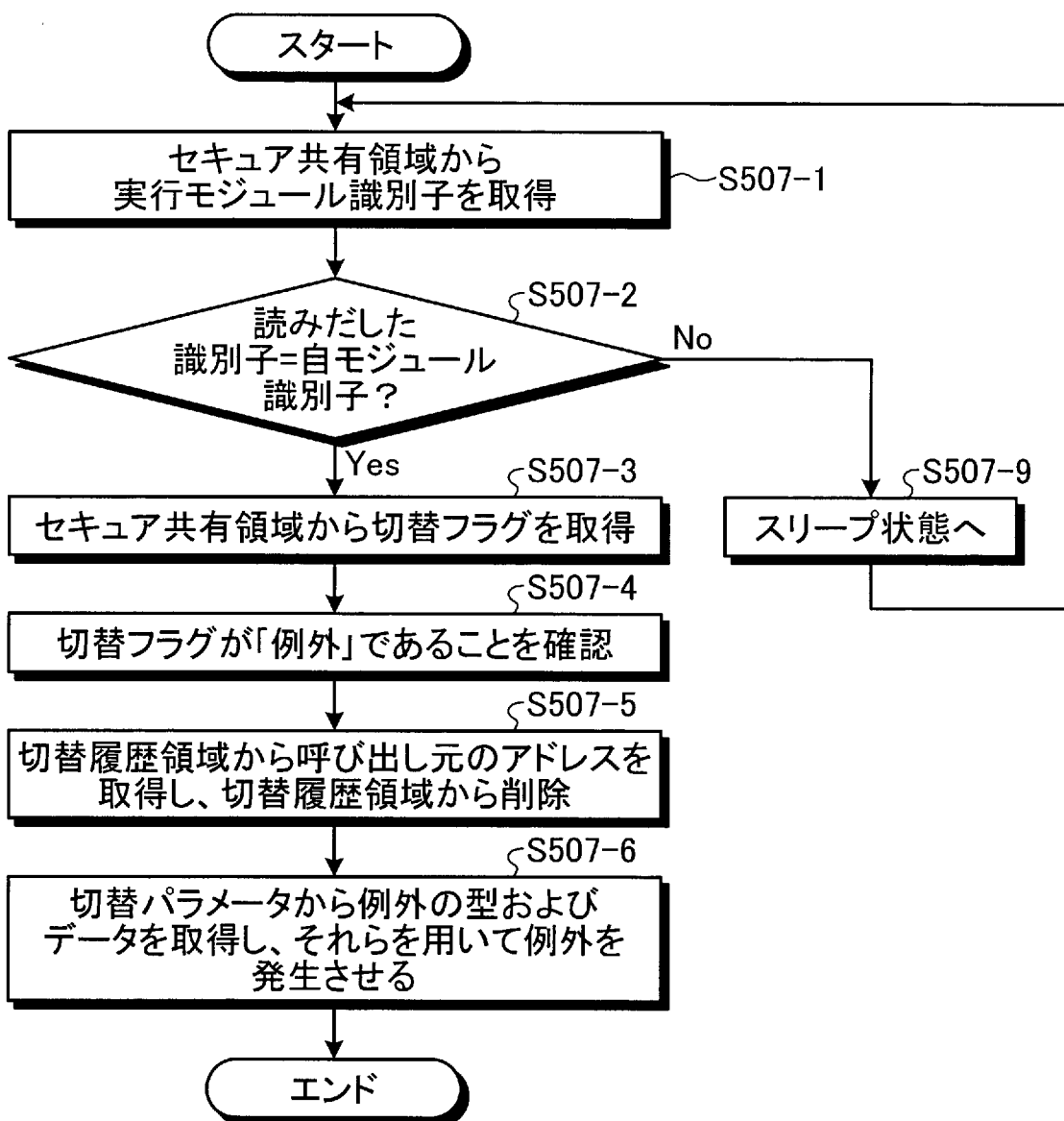
[図32]



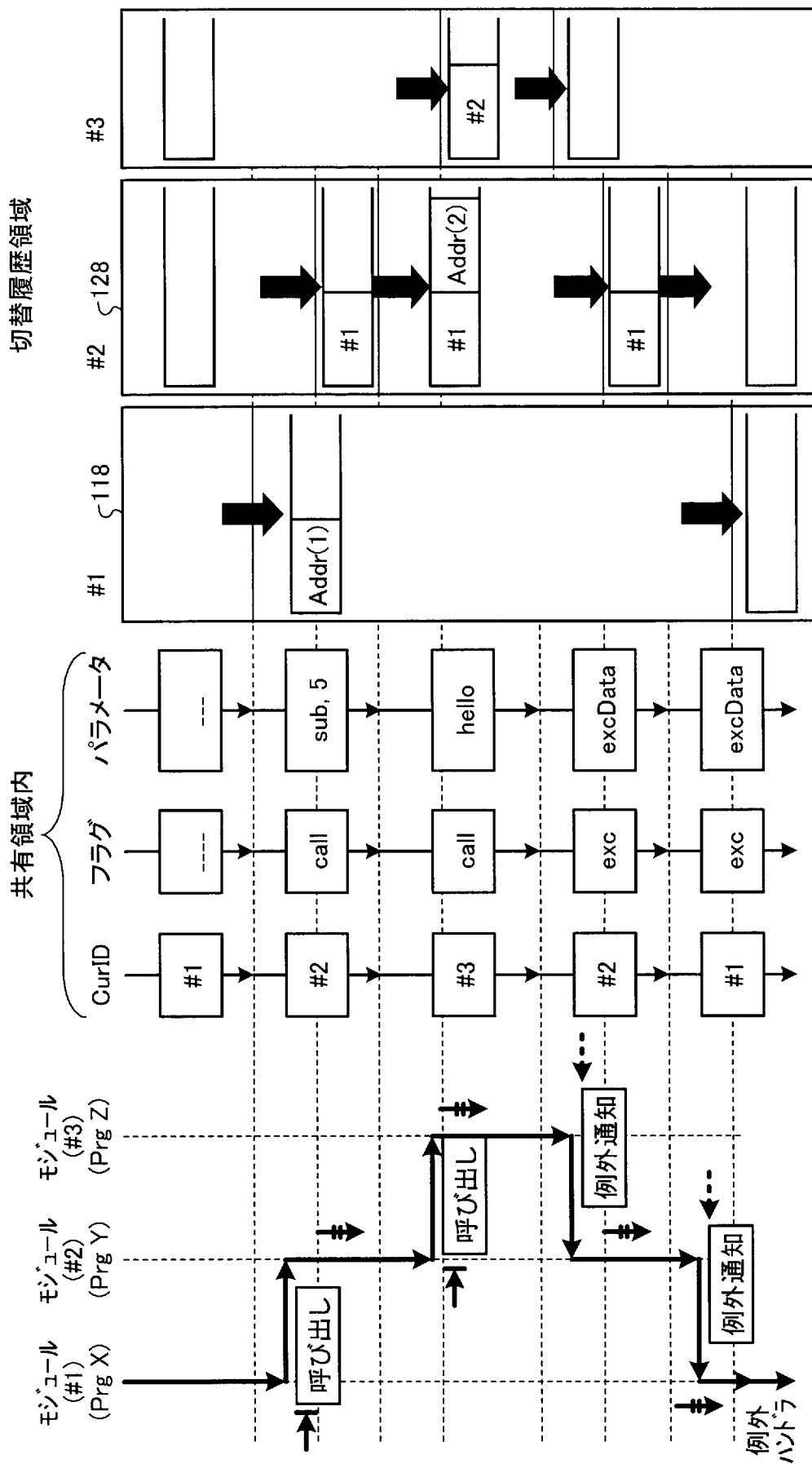
[図33]



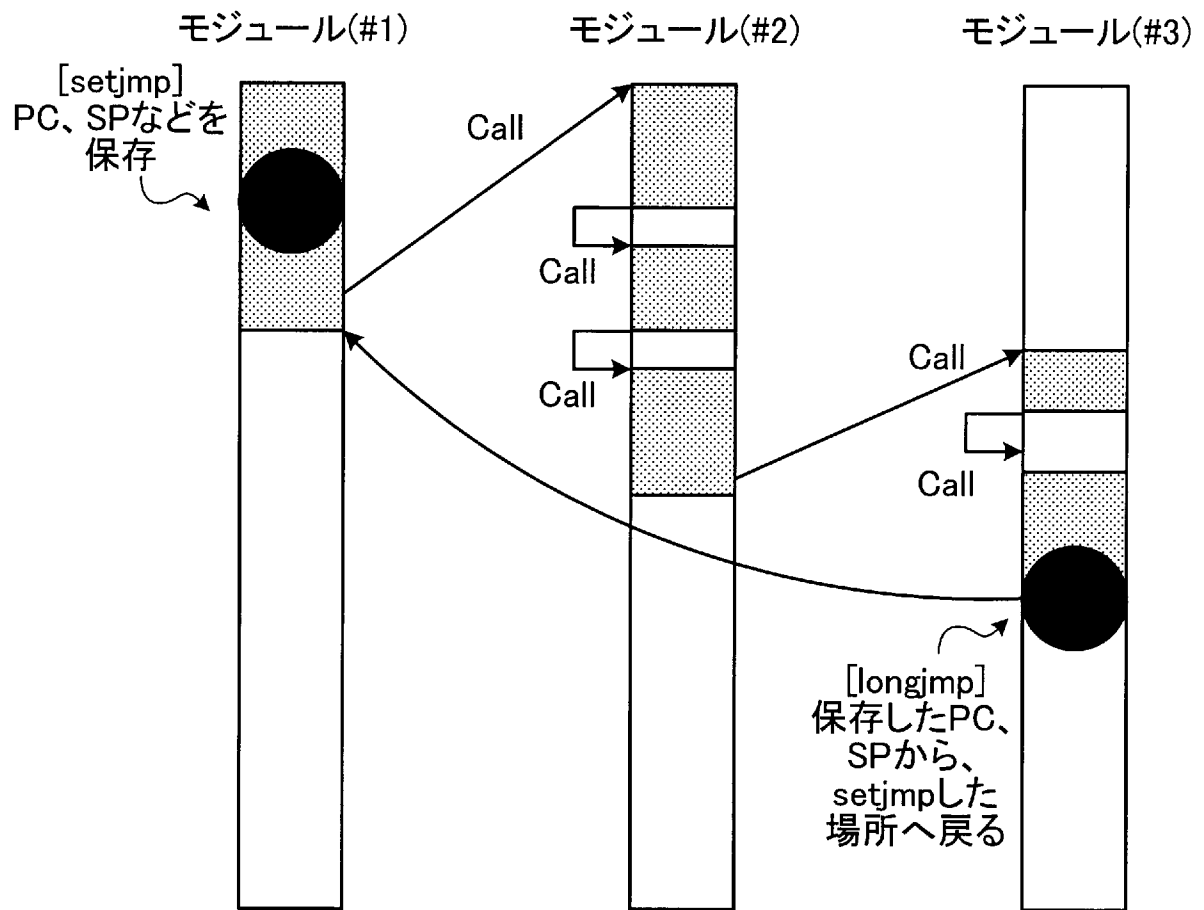
[図34]



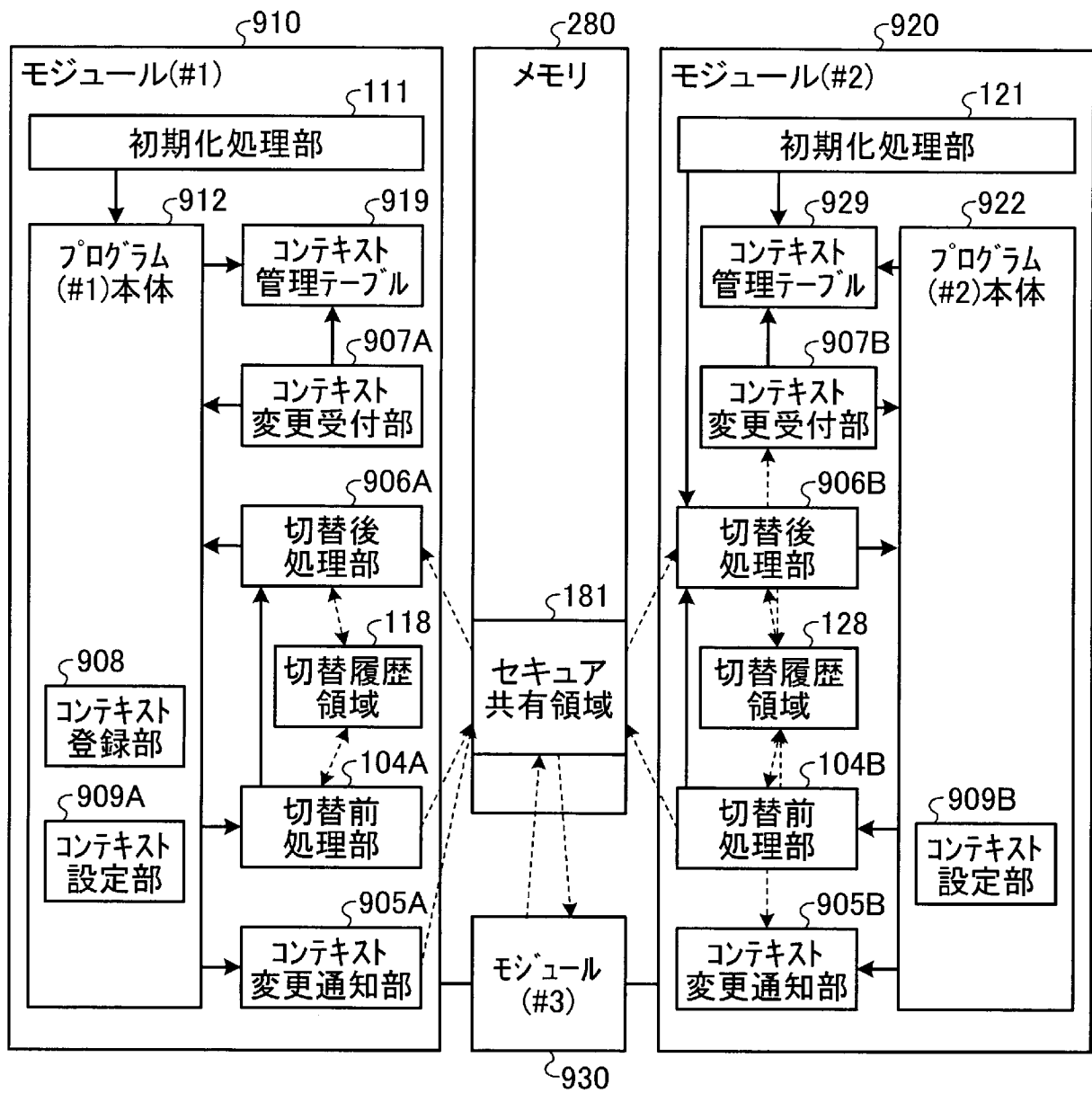
[図35]



[図36]



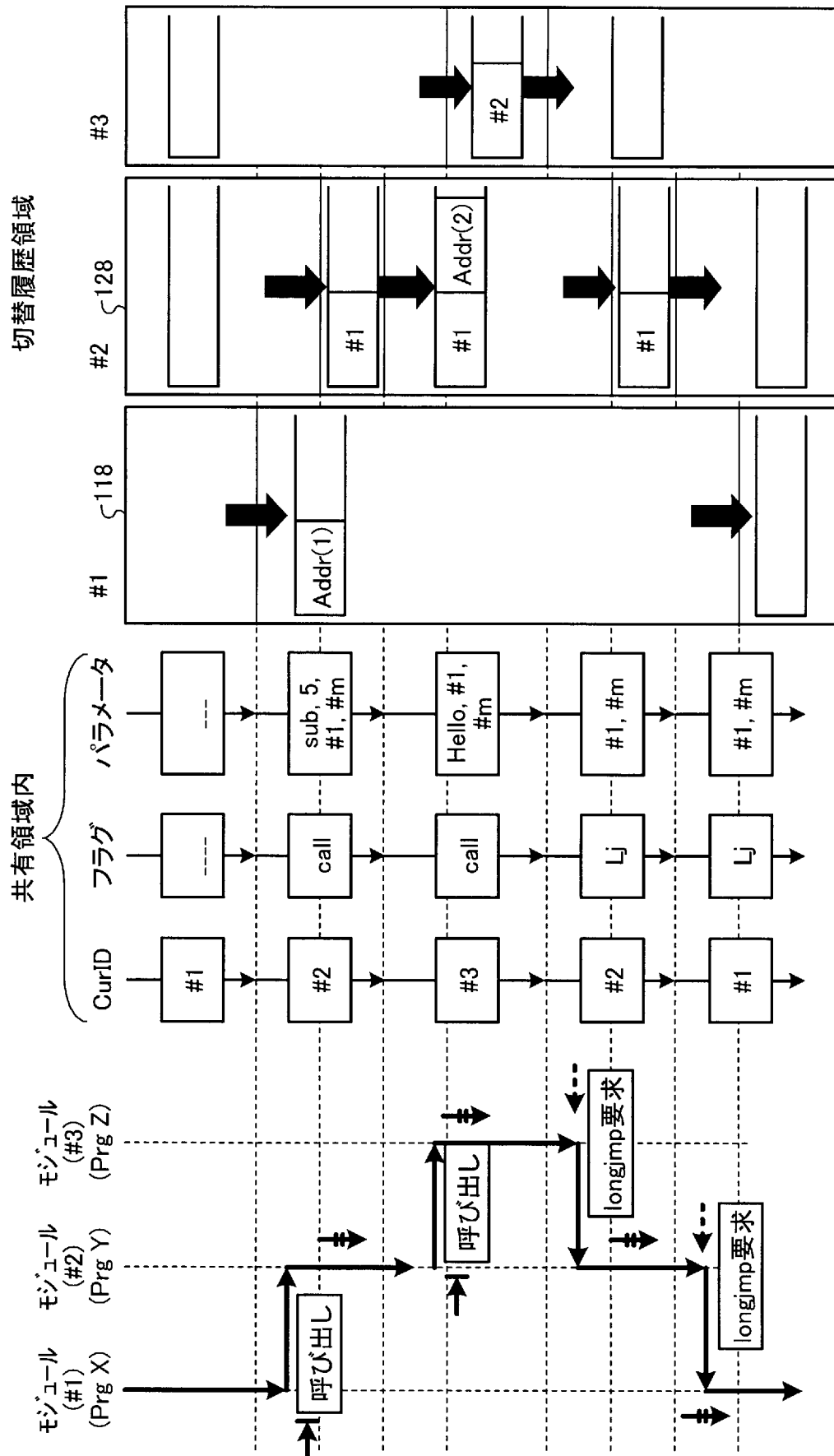
[図37]



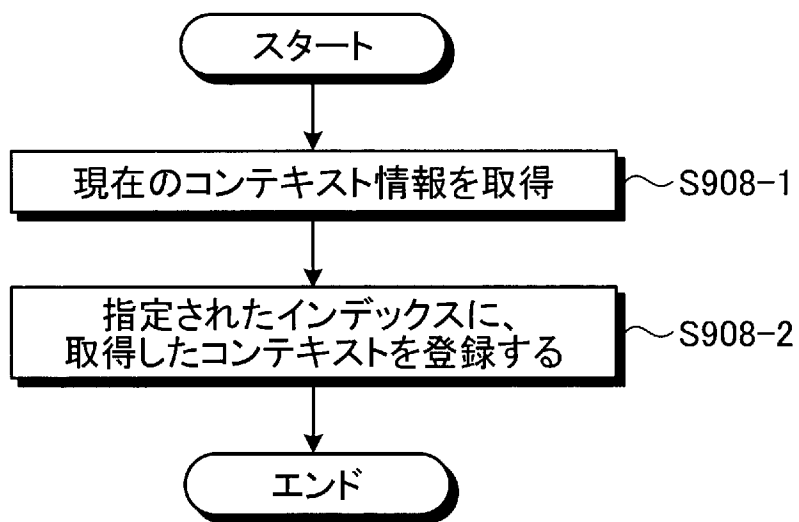
[図38]

モジュール識別子	コンテキスト番号	コンテキスト情報 (PC, SPなど)
...	...	...

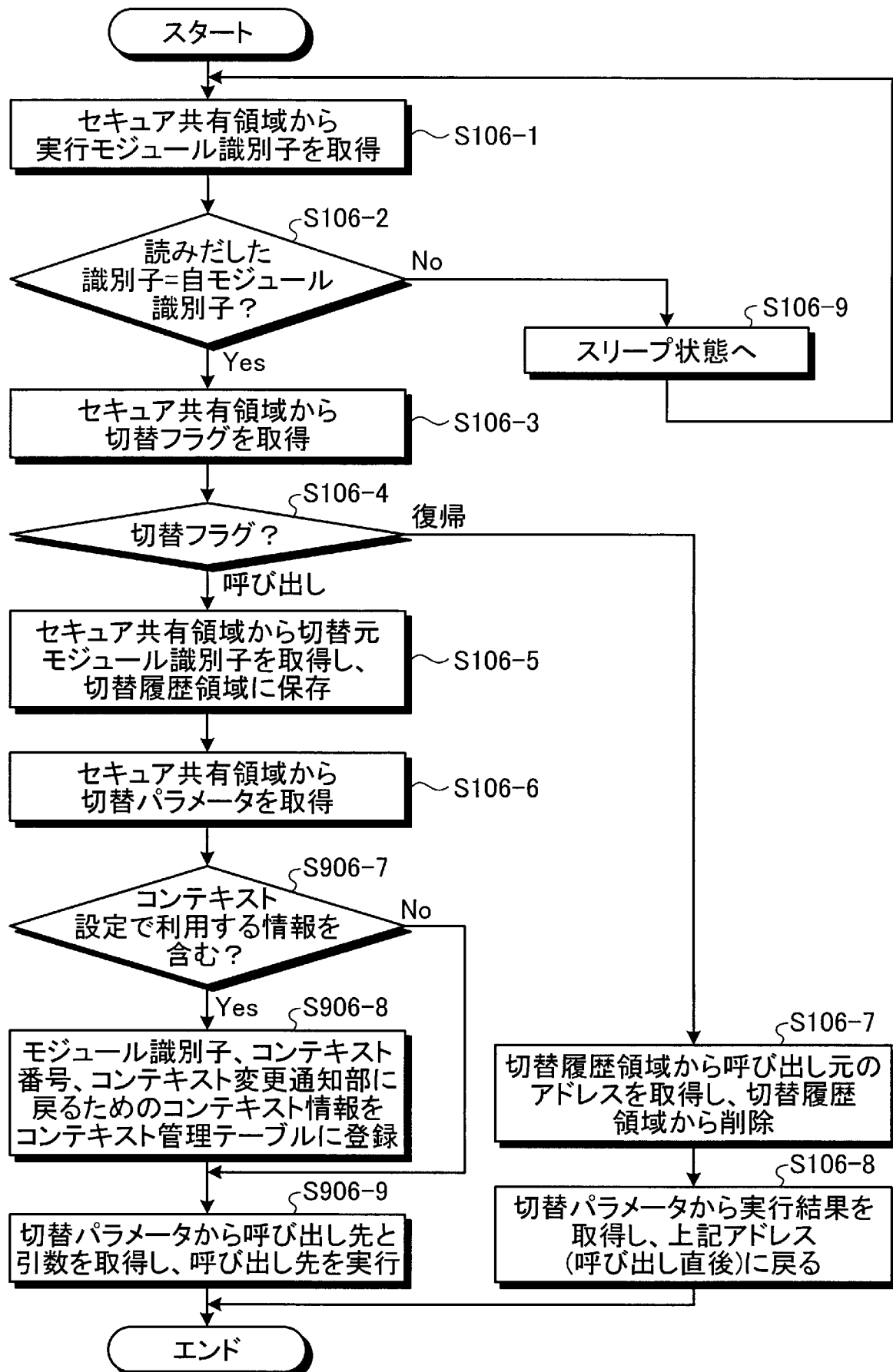
[図39]



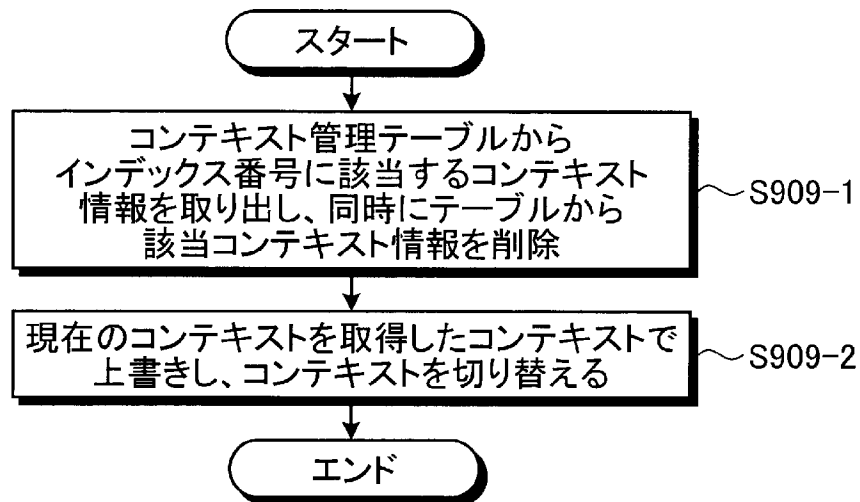
[図40]



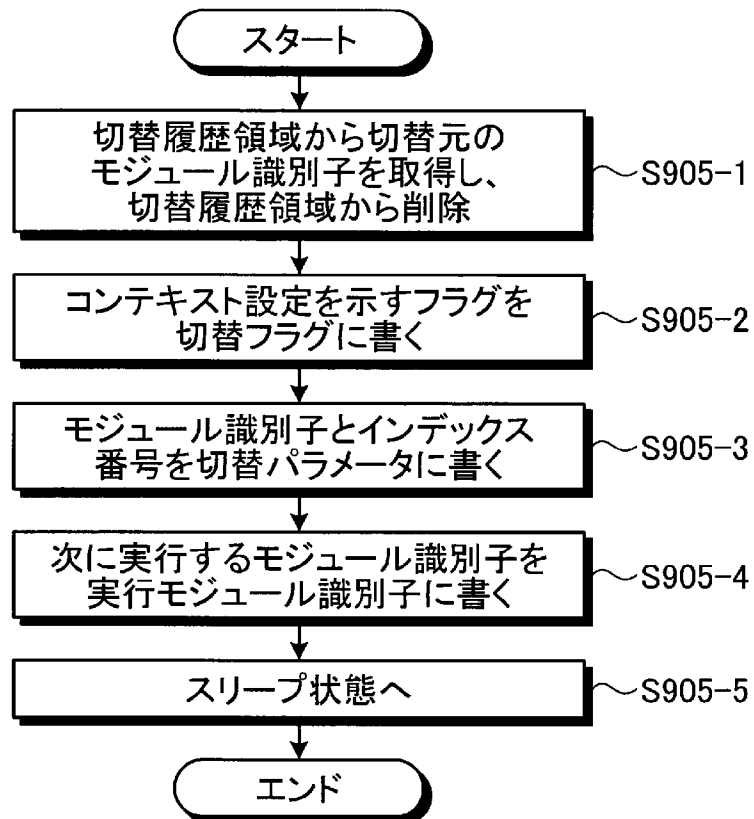
[図41]



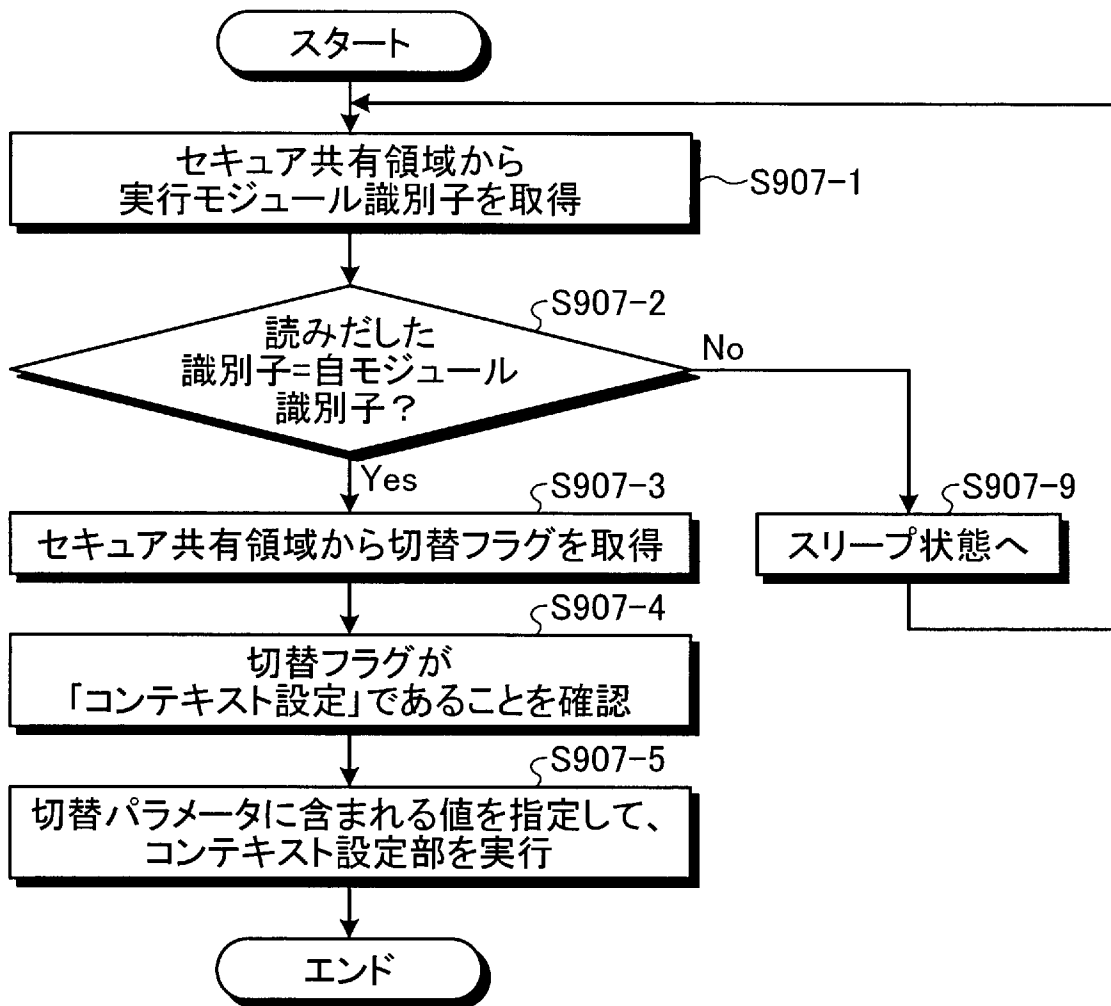
[図42]



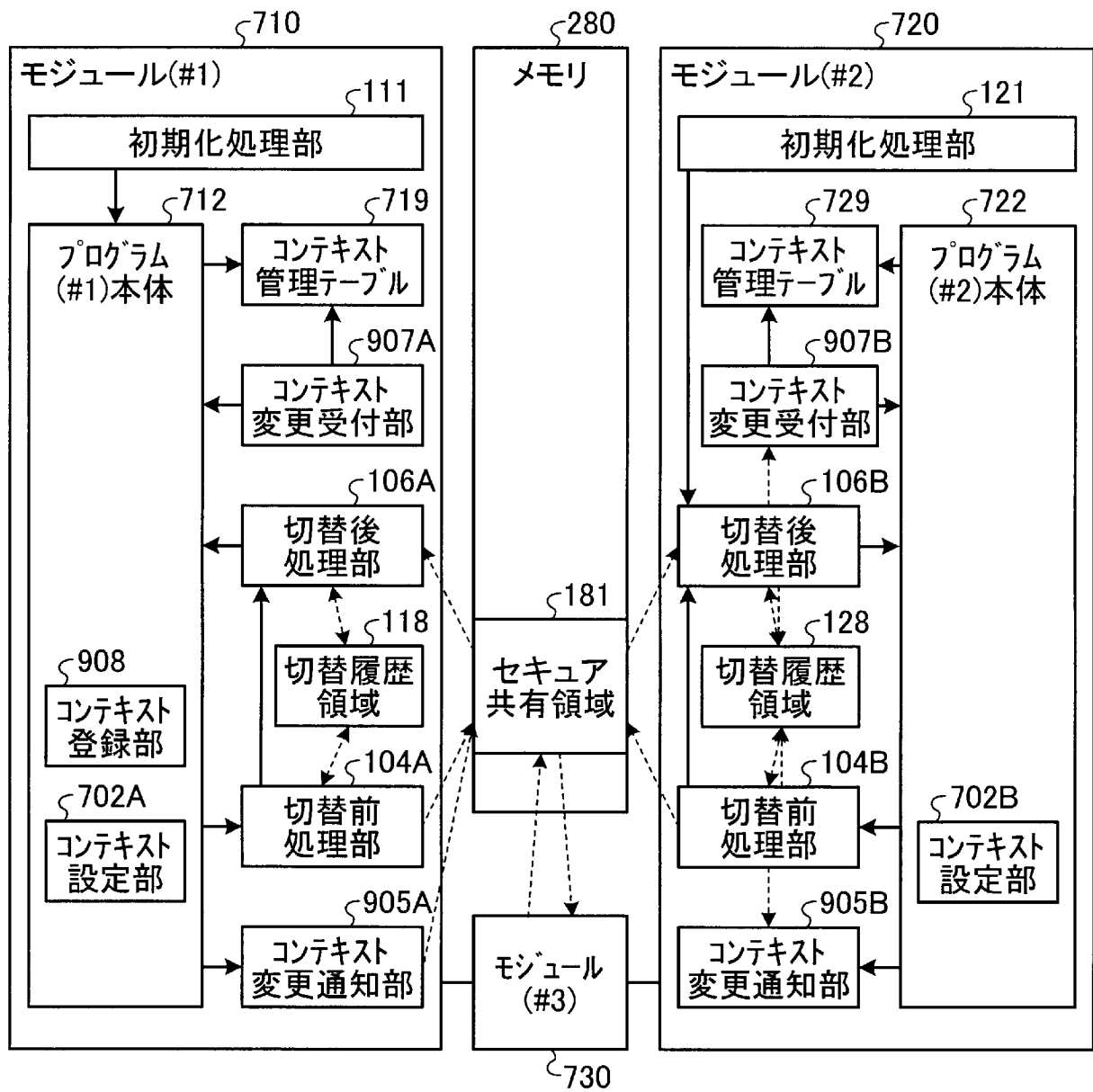
[図43]



[図44]



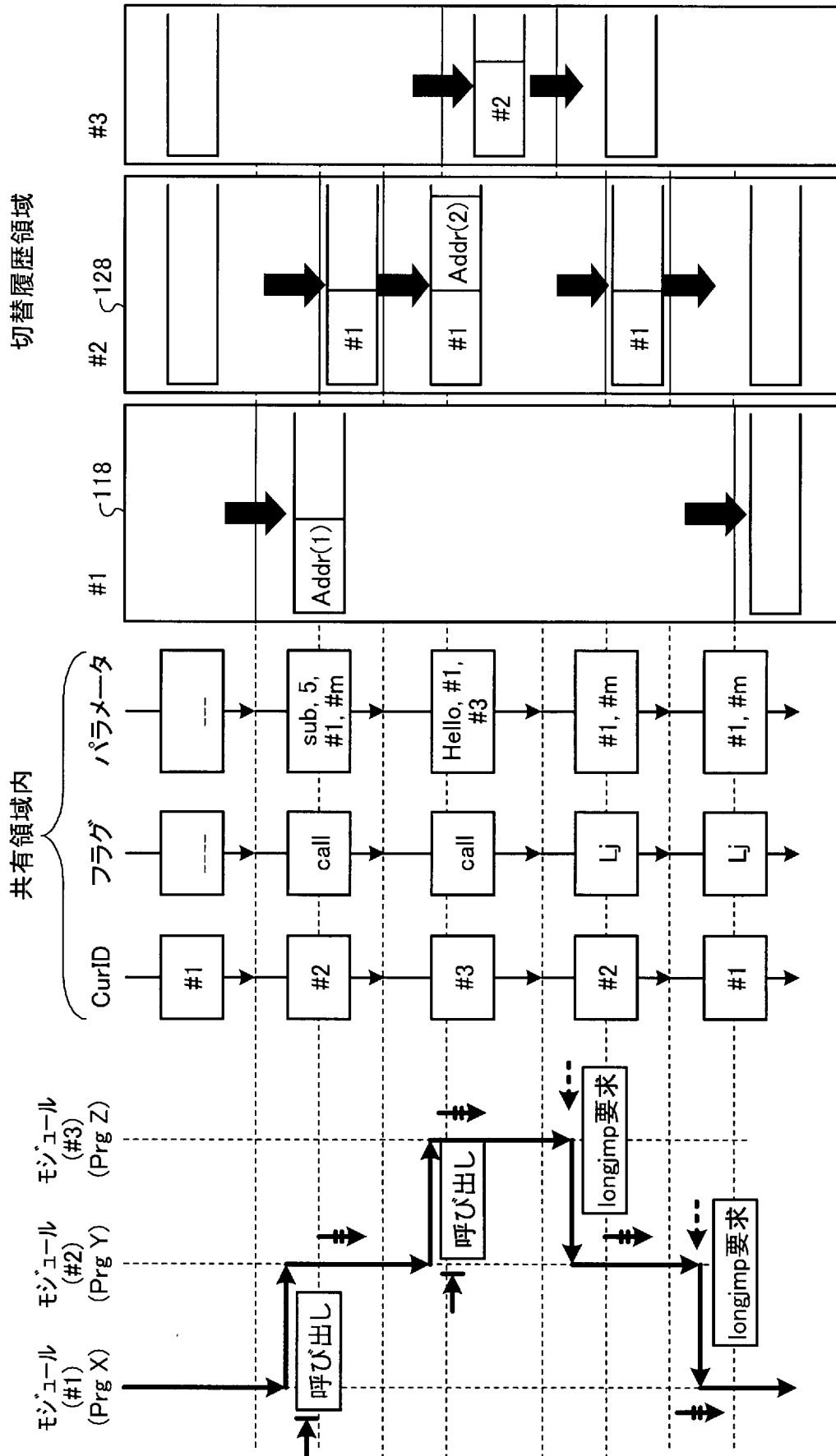
[図45]



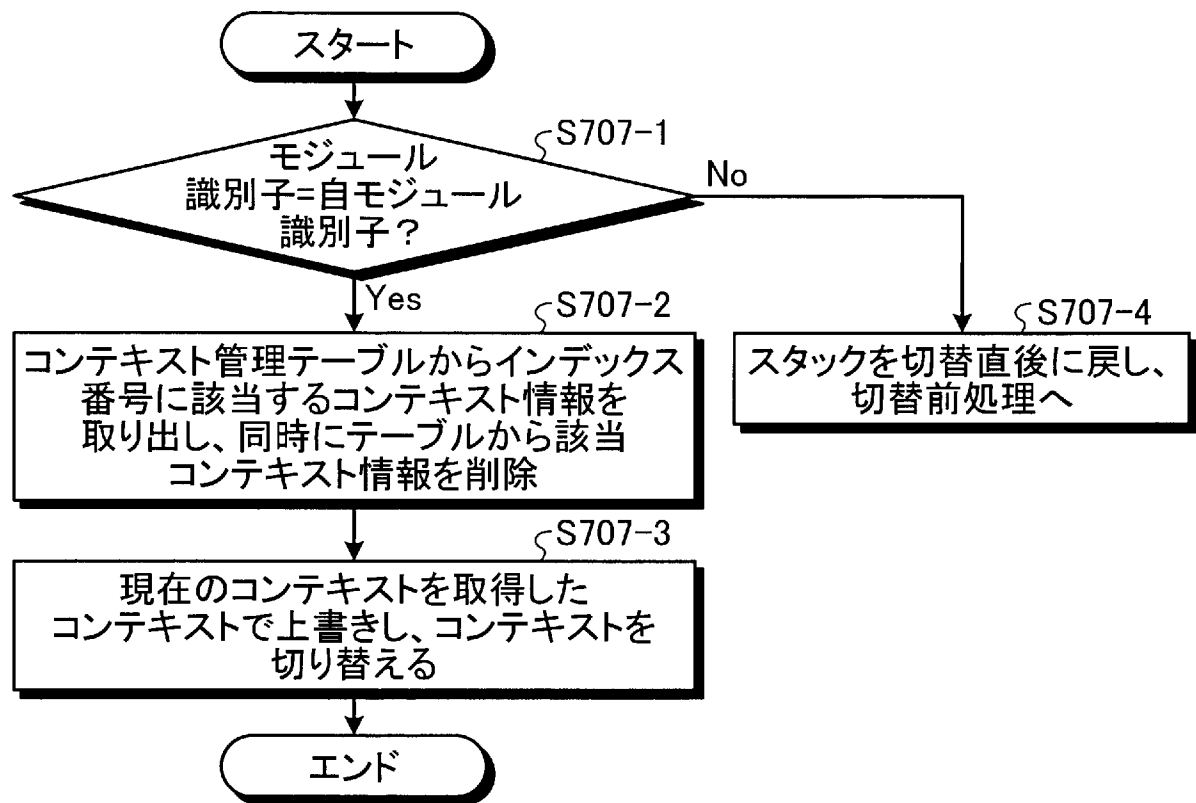
[図46]

コンテキスト番号	コンテキスト情報 (PC, SPなど)
#1	
#2	
#3	
...	...
#n	

[図47]



[図48]



INTERNATIONAL SEARCH REPORT

International application No.

PCT/JP2010/052465

A. CLASSIFICATION OF SUBJECT MATTER

G06F21/22 (2006.01) i, G06F21/24 (2006.01) i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F21/22, G06F21/24

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Jitsuyo Shinan Koho 1922-1996 Jitsuyo Shinan Toroku Koho 1996-2010

Kokai Jitsuyo Shinan Koho 1971-2010 Toroku Jitsuyo Shinan Koho 1994-2010

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JP 4282472 B2 (Toshiba Corp.), 24 June 2009 (24.06.2009), entire text; all drawings & US 2005/0144438 A1	1-9
A	JP 2007-148962 A (Fuji Xerox Co., Ltd.), 14 June 2007 (14.06.2007), entire text; all drawings & US 2007/0136728 A1 & CN 1975678 A & KR 10-2007-0056940 A	1-9
A	JP 2007-164595 A (Toshiba Corp.), 28 June 2007 (28.06.2007), entire text; all drawings & US 2007/0169068 A1	1-9

☒ Further documents are listed in the continuation of Box C.☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search
08 October, 2010 (08.10.10)Date of mailing of the international search report
19 October, 2010 (19.10.10)Name and mailing address of the ISA/
Japanese Patent Office

Authorized officer

Facsimile No.

Telephone No.

INTERNATIONAL SEARCH REPORT

International application No.

PCT/JP2010/052465

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JP 2005-166051 A (Kabushiki Kaisha Inka Internet), 23 June 2005 (23.06.2005), entire text; all drawings & KR 10-0483700 B1 & CN 1624617 A	1-9
A	JP 63-132334 A (NEC Corp.), 04 June 1988 (04.06.1988), entire text; all drawings (Family: none)	1-9

A. 発明の属する分野の分類（国際特許分類（I P C））
Int.Cl. G06F21/22 (2006.01)i, G06F21/24 (2006.01)i

B. 調査を行った分野

調査を行った最小限資料（国際特許分類（I P C））

Int.Cl. G06F21/22, G06F21/24

最小限資料以外の資料で調査を行った分野に含まれるもの

日本国実用新案公報	1 9 2 2－1 9 9 6 年
日本国公開実用新案公報	1 9 7 1－2 0 1 0 年
日本国実用新案登録公報	1 9 9 6－2 0 1 0 年
日本国登録実用新案公報	1 9 9 4－2 0 1 0 年

国際調査で使用した電子データベース（データベースの名称、調査に使用した用語）

C. 関連すると認められる文献

引用文献の カテゴリー*	引用文献名 及び一部の箇所が関連するときは、その関連する箇所の表示	関連する 請求項の番号
A	JP 4282472 B2（株式会社東芝） 2009.06.24, 全文, 全図 & US 2005/0144438 A1	1-9
A	JP 2007-148962 A（富士ゼロックス株式会社） 2007.06.14, 全文, 全図 & US 2007/0136728 A1 & CN 1975678 A & KR 10-2007-0056940 A	1-9

☒ C 欄の続きにも文献が列挙されている。

☐ パテントファミリーに関する別紙を参照。

* 引用文献のカテゴリー

「A」特に関連のある文献ではなく、一般的技術水準を示すもの
「E」国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの
「L」優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献（理由を付す）
「O」口頭による開示、使用、展示等に言及する文献
「P」国際出願日前で、かつ優先権の主張の基礎となる出願

の日の後に公表された文献

「T」国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの
「X」特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの
「Y」特に関連のある文献であって、当該文献と他の1以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの
「&」同一パテントファミリー文献

国際調査を完了した日

0 8 . 1 0 . 2 0 1 0

国際調査報告の発送日

1 9 . 1 0 . 2 0 1 0

国際調査機関の名称及びあて先

日本国特許庁（I S A / J P）
郵便番号 1 0 0－8 9 1 5
東京都千代田区霞が関三丁目4番3号

特許庁審査官（権限のある職員）

市川 武宜

5 S

4 0 5 6

電話番号 0 3－3 5 8 1－1 1 0 1 内線 3 5 4 6

C (続き) . 関連すると認められる文献		
引用文献の カテゴリー*	引用文献名 及び一部の箇所が関連するときは、その関連する箇所の表示	関連する 請求項の番号
A	JP 2007-164595 A (株式会社東芝) 2007.06.28, 全文, 全図 & US 2007/0169068 A1	1-9
A	JP 2005-166051 A (株式会社 インカインターネット) 2005.06.23, 全文, 全図 & KR 10-0483700 B1 & CN 1624617 A	1-9
A	JP 63-132334 A (日本電気株式会社) 1988.06.04, 全文, 全図 (ファミリーなし)	1-9