T-Kernel入門

TRON Forum
トロンフォーラム

第一章
T-Kernelとは？

T-Kernel

- T-Kernelは2002年に公開された、T-Engine Forumが開発し公開している組込みリアルタイムOS
- 2011年5月17日にバージョンアップ版の T-Kernel 2.0 を公開
- T-Engineアーキテクチャの心臓部
  - µITRONの技術を継承し…
  - より大規模なシステムにも対応できるように強化された…
  - 次世代の応用に適したリアルタイムOS
- T-Kernel Extensionが利用するマイクロカーネル
  - T-Kernelはコアであり、ファイルシステムやネットワークプロトコルなどの上位機能はT-Kernel Extensionが提供
  - T-Kernel Extensionでは、T-Kernelの機能を拡張したより高度な機能をアプリケーションに提供

特徴

- T-Kernelが持つITRONにない特徴
  - 強い標準化
  - Single One Source
  - リファレンスハードウェアの標準化と仕様の公開
  - シリーズ化（μT-Kernel、MP T-Kernel、…）
  - Extension（T-Kernel Standard Extension）
  - ダイナミックロード
- T-Kernelシリーズ間の高い互換性
- μITRONとの相互流通
  - T-KernelはTRONで培った技術を継承した新時代のRTOS
  - μITRON 用ソフトウェアとの相性もよく、ミドルウェアの相互流通を図れる
  - T-Kernel上でITRON用アプリを実行できるラッパーも

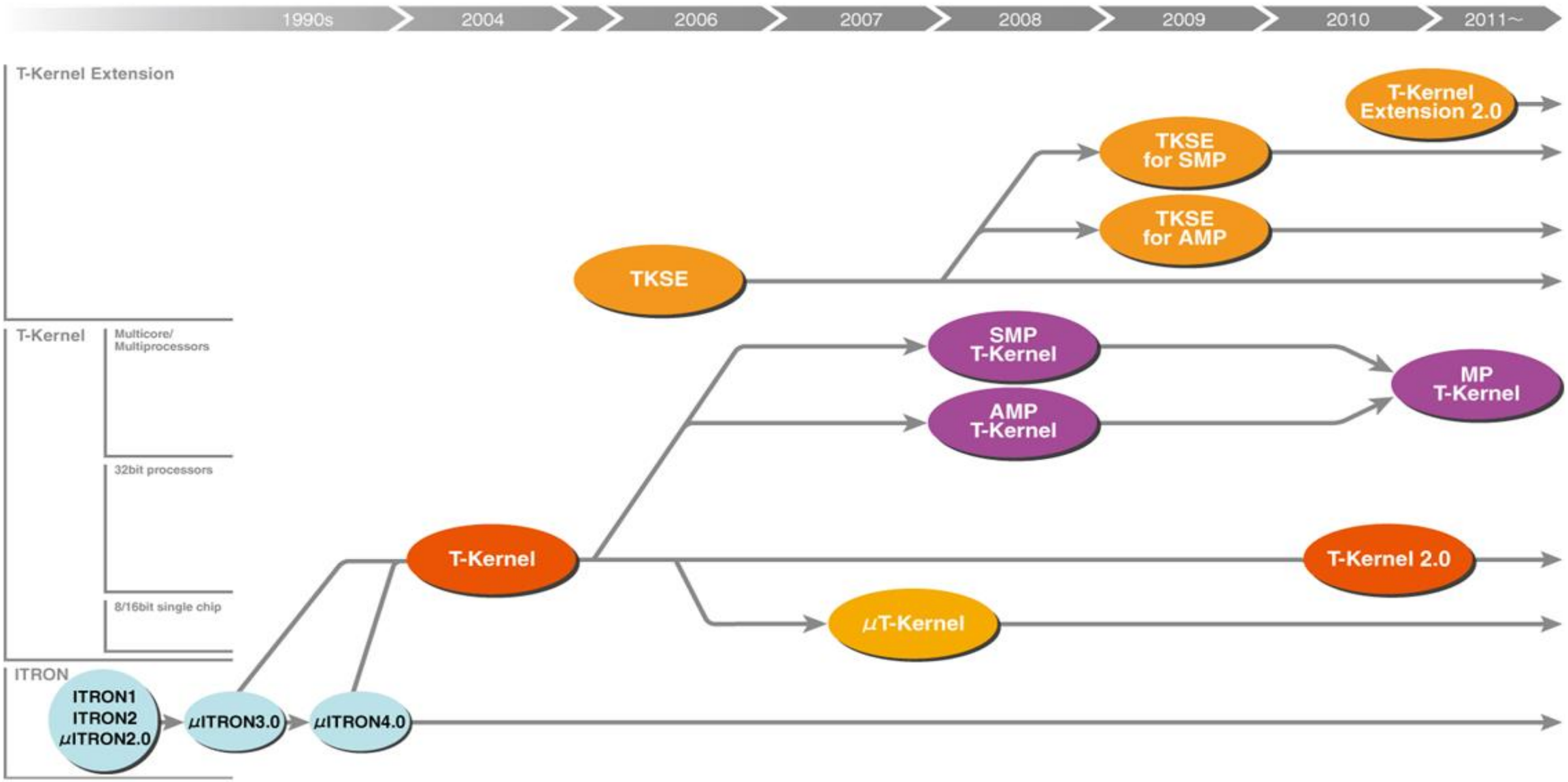
「強い標準化」

- ITRONは弱い標準化に特徴
  - ハードは規定しない。仕様のみ公開。
  - 1987年当時はPCでさえ16ビットCPUが主流
  - ハードウェアに併せてチューニングできる余地が重要（20年前）

- 21世紀になり、組込みシステムのハードウェア性能の向上に伴い、適応化よりも標準化が重要な要素に

- 強い標準化へ変化
  - リファレンスコードを無償で公開
    - Single One Source
  - 大規模システムを前提としたミドルウェアの流通促進を重視

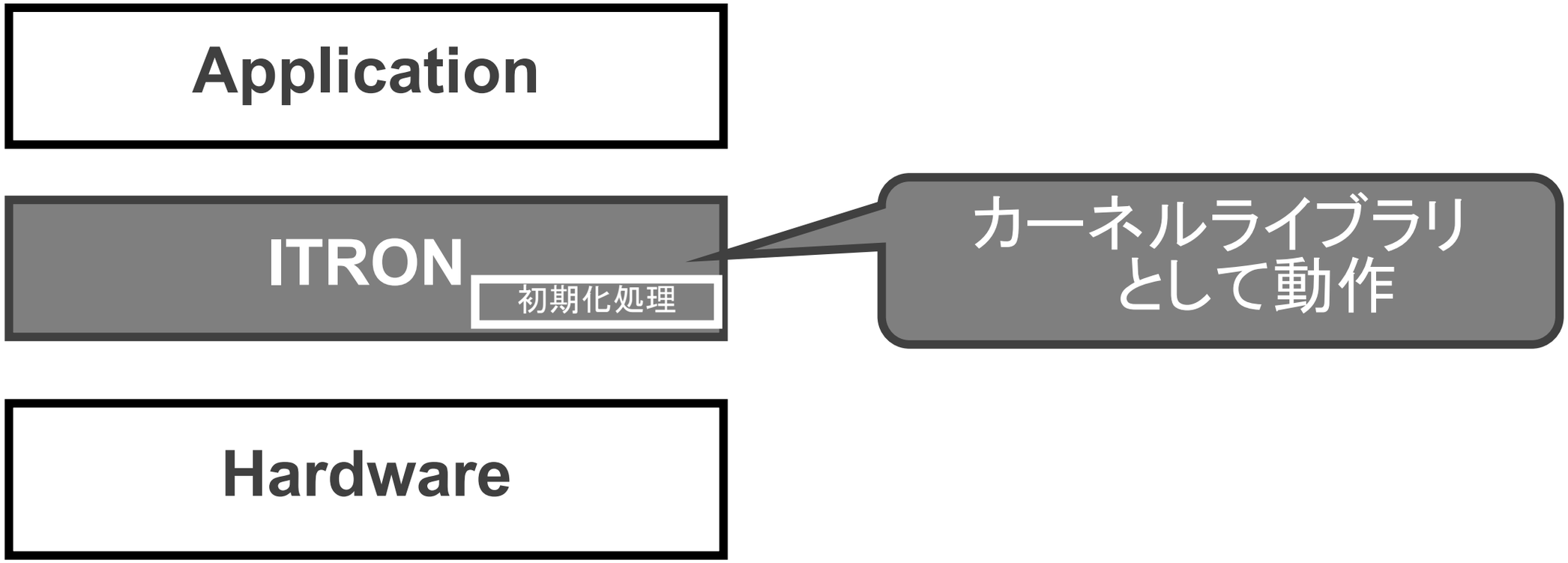
T-KernelとITRONの開発ロードマップ



第二章
T-Kernelを利用した組込みシステムの基本構成

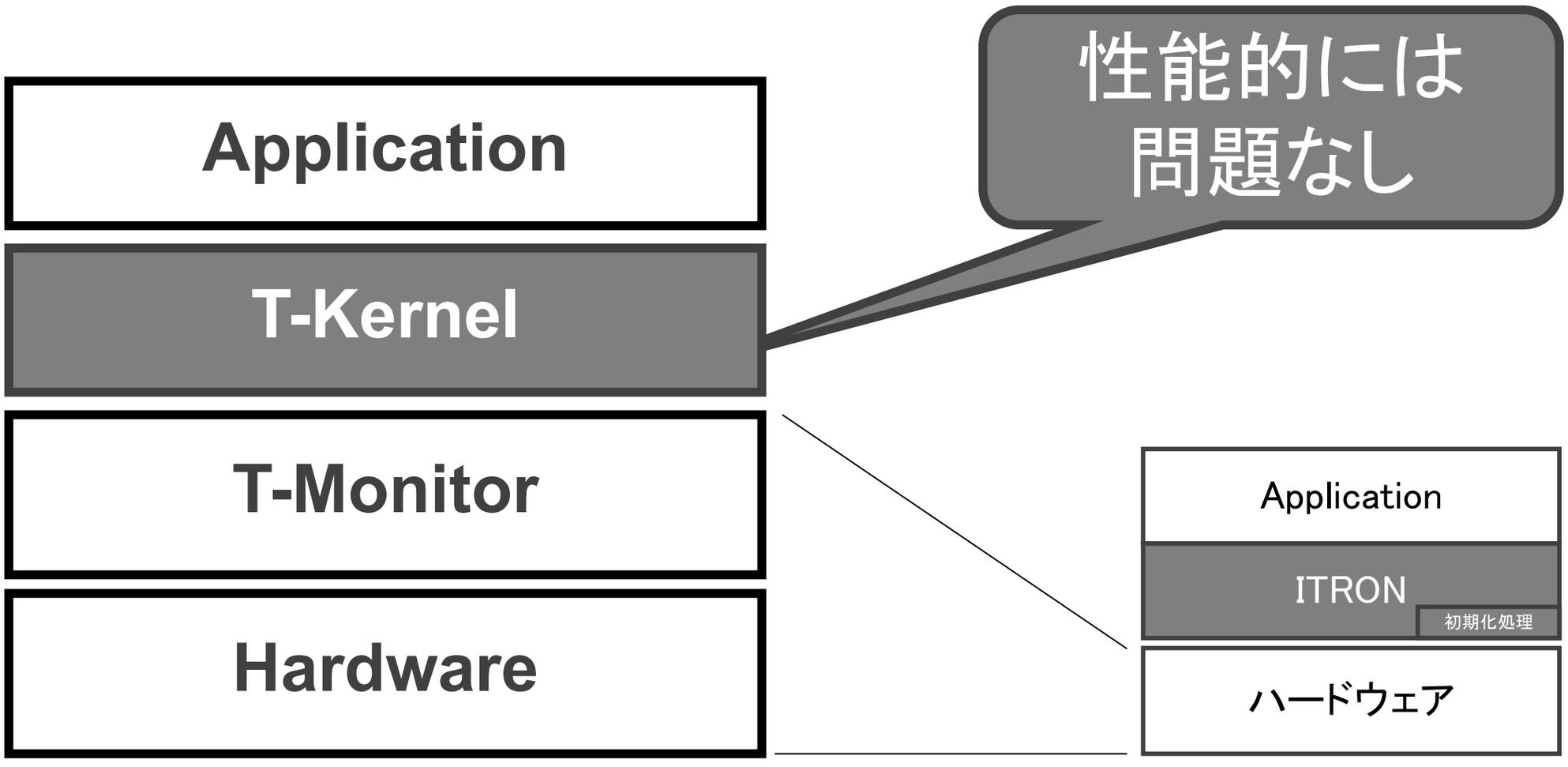
ITRONでの構成

- シンプルだが、開発にはとても有用
  - 基本機能は全てITRONが提供



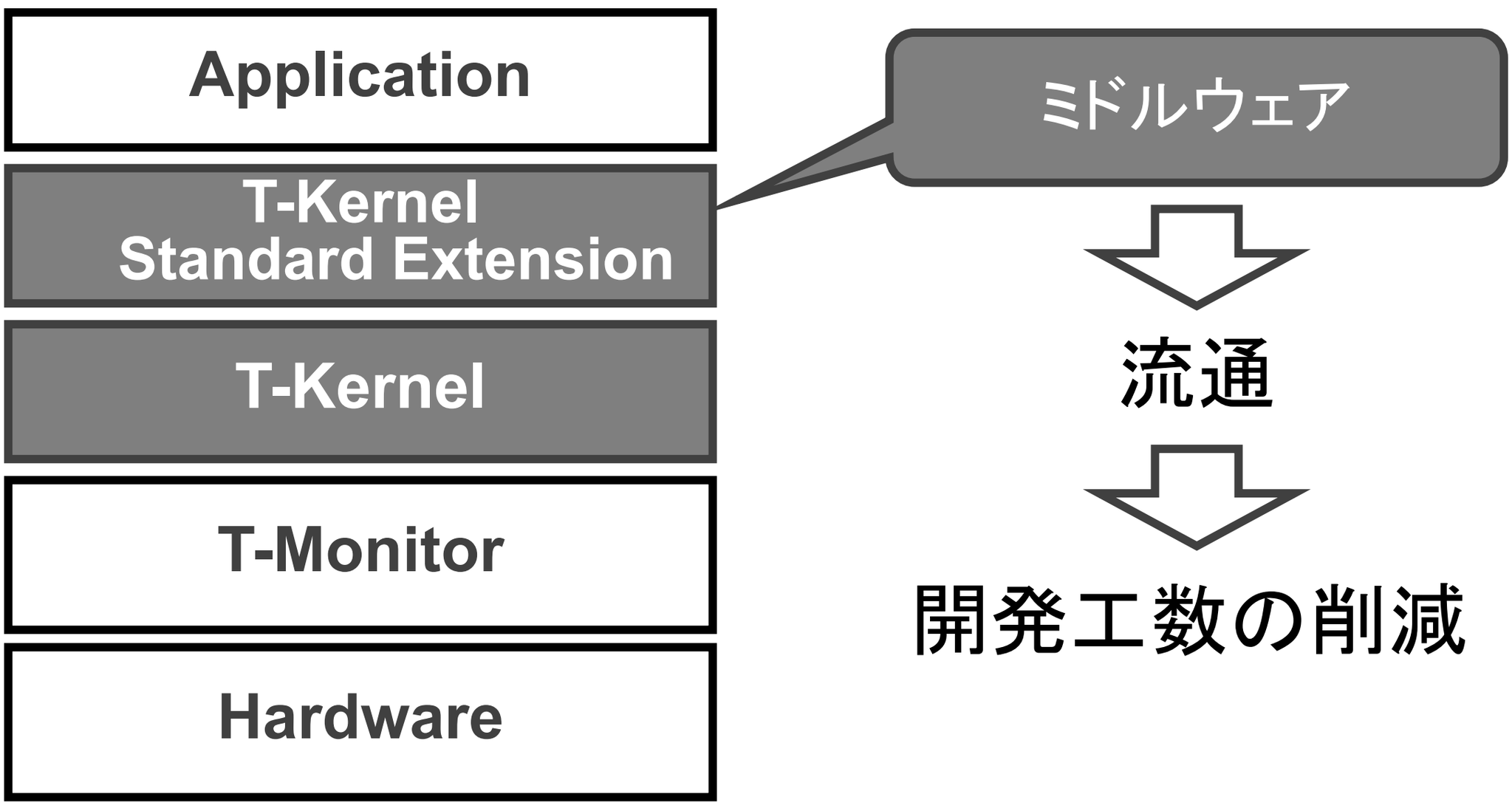
T-Kernelを単なるRTOSとして用いた場合

- ITRONと同じ使用方法



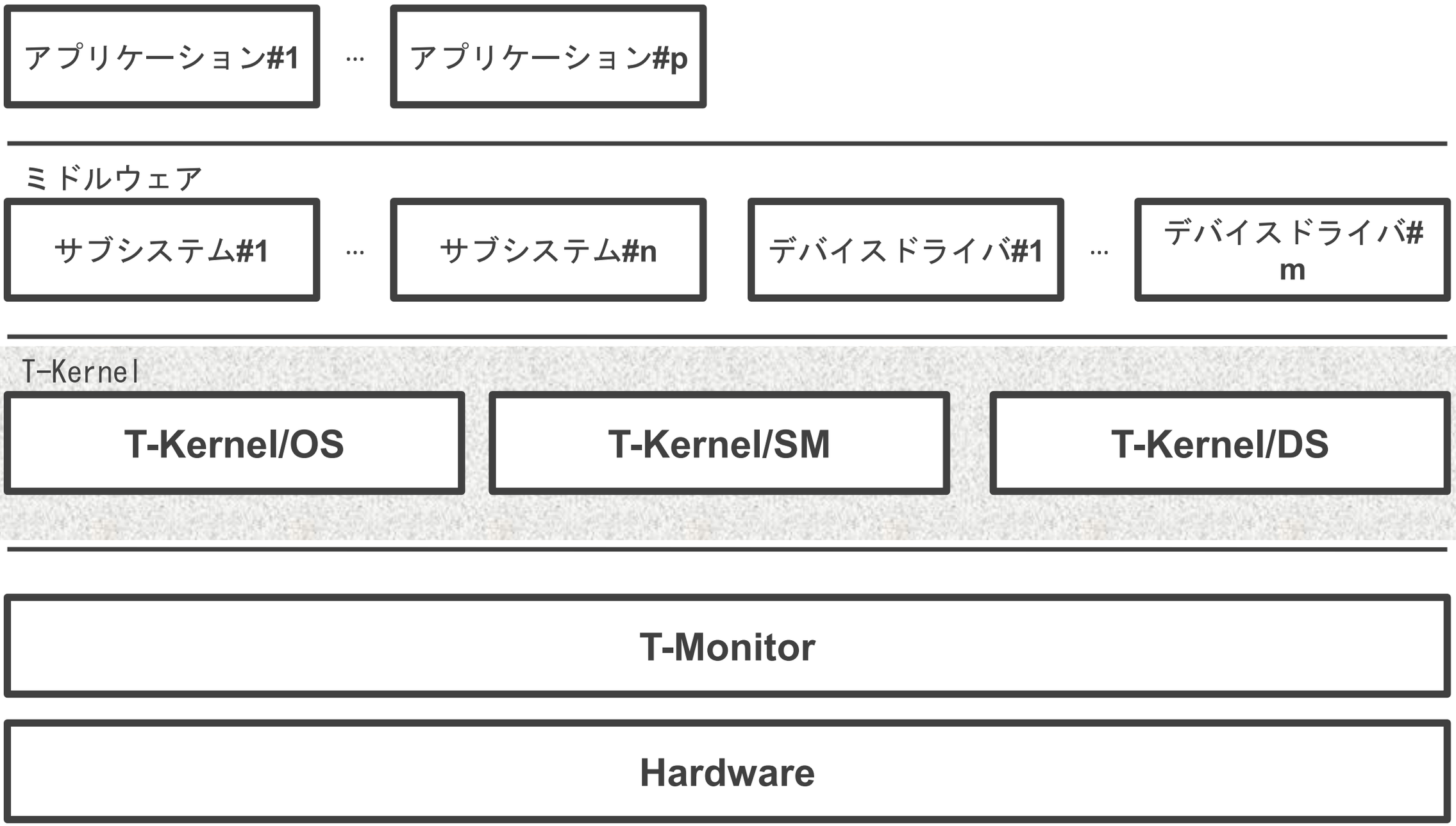
本来はT-Kernel Standard Extensionを入れて使う

- 大規模システム対応



第三章
T-Kernelソフトウェア構成

T-Kernelソフトウェア構成



T-Monitor

ハードウェアの初期化や割込みのハンドリングを行う

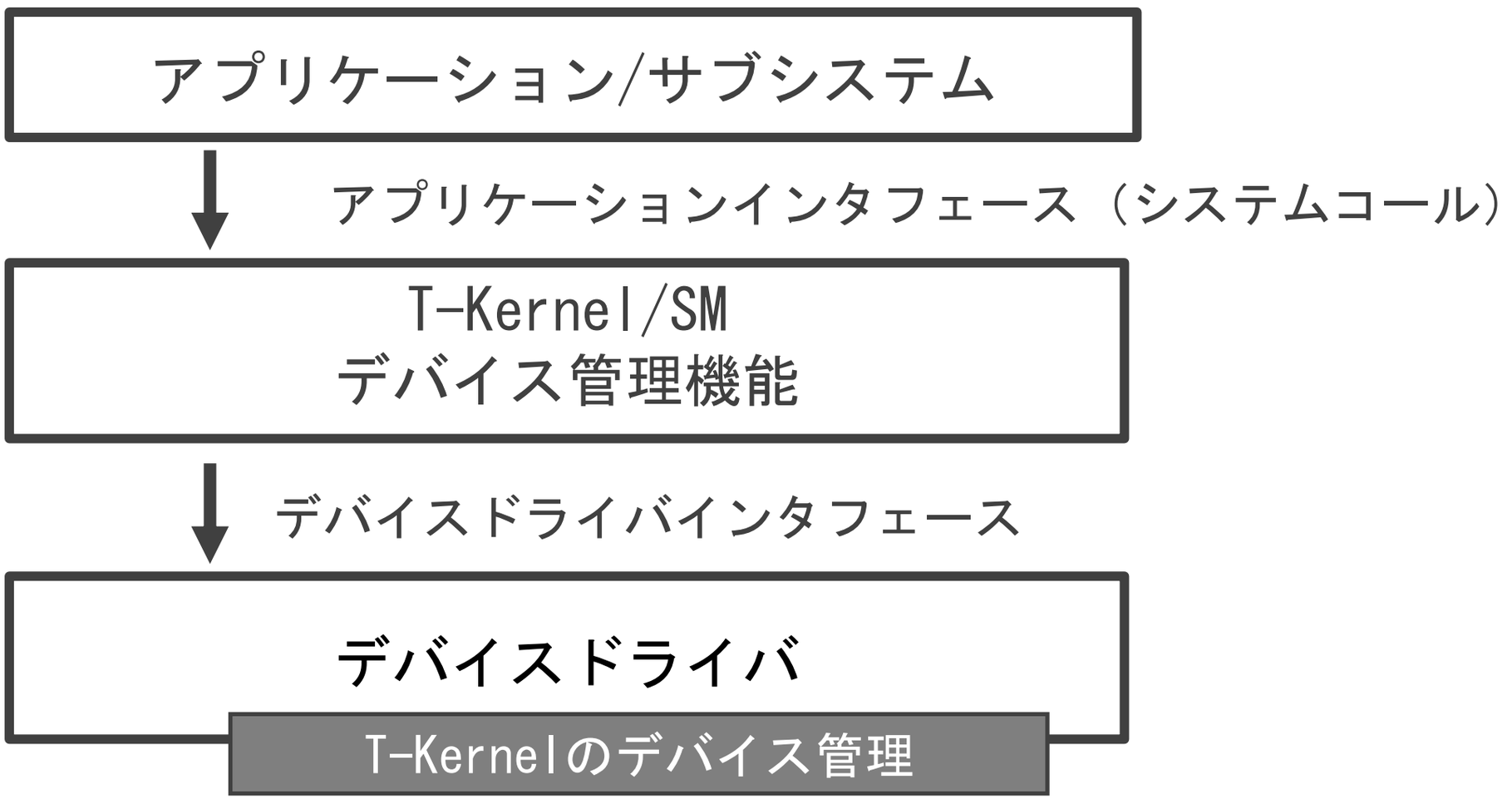
- システム機能
  - ハードウェアの初期化
  - システムの起動
- デバッグ機能
  - プログラムのロード、実行
  - ブレークポイントおよびトレース
  - メモリ、レジスタ、I/O操作
- プログラムサポート機能
  - モニタサービス関数の提供

T-Kernel

- T-Kernel/OS(Operating System)
  - リアルタイムOSとしての基本的な機能を提供。従来のITRONの機能に相当（狭義のT-Kernel）
- T-Kernel/SM(System Manager)
  - システムメモリ管理機能やアドレス空間管理機能など、デバイスドライバやサブシステムなどのミドルウェアを管理するための機能を提供
- T-Kernel/DS(Debugger Support)
  - デバッガなどの開発ツールのための機能を提供

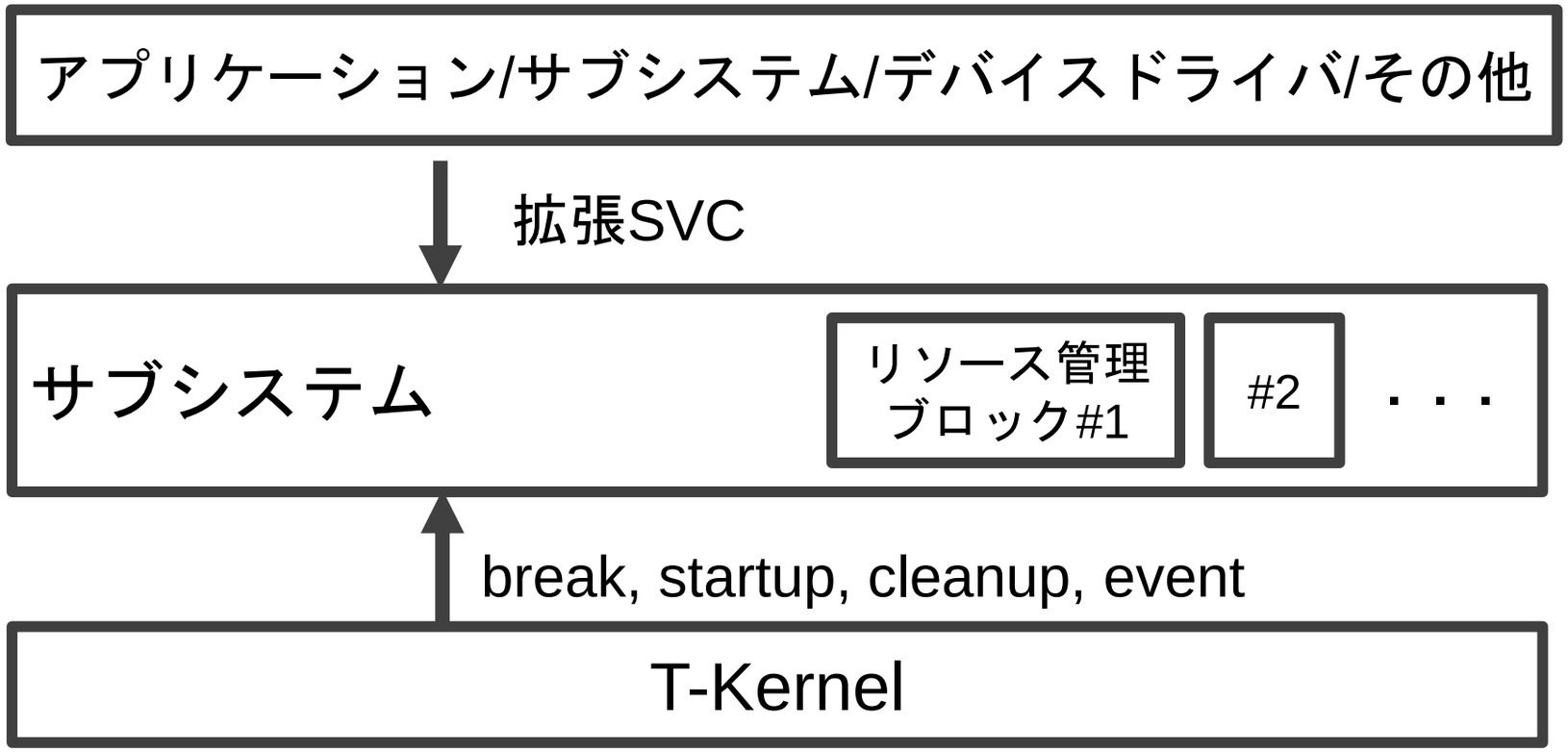
デバイスドライバ

- 標準デバイスのドライバは提供
  - シリアル、LCD、タッチパネル、等



T-Kernelサブシステム

- サブシステムの構成
  - アプリケーション等からの要求を受け付ける拡張ＳＶＣハンドラ
  - OSからの要求を処理する関数群
    - ブレーク関数、スタートアップ関数、クリーンアップ関数、イベント処理関数
  - リソース管理ブロック



T-Kernel Standard Extension

- T-Kernel の機能を増強するExtension（拡張プログラム）
  - メモリ管理
  - プロセス／タスク管理
  - プロセス間メッセージ
  - グローバル名
  - タスク間同期・通信
  - 標準入出力
  - 標準ファイル管理
  - イベント管理
  - デバイス管理
  - 時間管理
  - システム管理
  - 共有ライブラリ

第四章
T-Kernel 2.0

T-Kernel 2.0の位置付け

- 従来のT-Kernel 1.0の設計方針や特長は不変
- CPUの高性能化、デバイスの大容量化を活かす追加機能
  - 64ビットデータ、マイクロ秒単位の時間指定など
- T-Kernel 1.0に対しては上位互換
  - ソース互換およびバイナリ互換

T-Kernel 2.0の追加機能

- 64ビットデータの導入
- マイクロ秒単位の時間管理機能
- 大容量デバイスへの対応
- 物理タイマ機能
- その他の追加機能

64ビットデータの導入

- C言語の規格(C99)で64ビットのlong long型が正式に仕様化
- T-Kernelで使う大部分のコンパイラ(gccなど)でサポート済
- マイクロ秒単位の時間管理や64ビットデバイスに利用

```
typedef         signed long long        D;      /* 符号付き64 ビット整数 */
typedef         unsigned long long      UD;     /* 符号無し64 ビット整数 */
```

マイクロ秒単位の時間管理機能

- ITRONやT-Kernel 1.0の時間管理の指定はミリ秒単位
- より細かい時間分解能への要求
  - CPUの高性能化
    - システムコール実行時間が1マイクロ秒に迫る
    - タスクの開始から終了までミリ秒未満のケースも
  - FA用の制御装置、PLCなどでの要求

マイクロ秒単位の時間管理機能

- 指定可能な時間の範囲
  - **32**ビットでミリ秒単位では約**24**日間
  - **32**ビットマイクロ秒単位では、この**1000**分の**1**で**35**分間
  - **64**ビットデータの採用により指定範囲を拡大
- マイクロ秒指定のシステムコールを追加
  - 互換性やマイクロ秒が不要な用途のため、ミリ秒単位のシステムコールも残る。混在利用も可能。

マイクロ秒単位の時間管理機能

- **64** ビットマイクロ秒単位を扱うデータタイプは ´**_U**´ を付加

```
typedef     D     TMO_U;      /* タイムアウト(μsec) */
typedef     UD    RELTIM_U;   /* 相対時間(μsec) */
typedef     D     SYSTIM_U;   /* システム時刻(μsec) */
```

マイクロ秒単位の時間管理機能

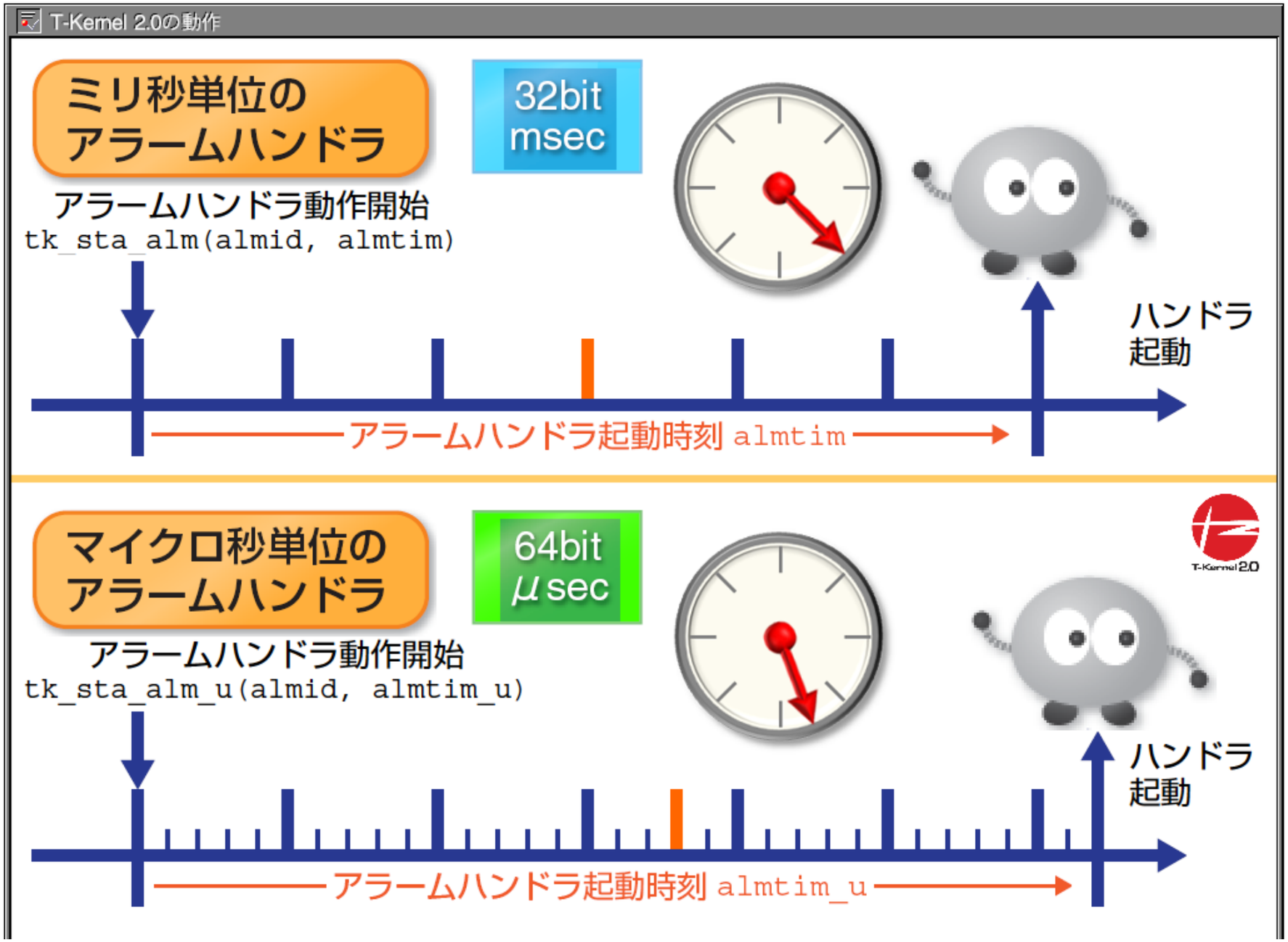
- マイクロ秒単位のシステムコールは ´_u´ を付加
  例:「アラームハンドラの動作開始」を行うAPI
  - **tk_sta_alm(ID almid, RELTIM almtim);**
    起動時刻almtimは32ビットミリ秒単位で指定
  - **tk_sta_alm_u(ID almid, RELTIM_U almtim_u);**
    起動時刻almtim_uは64ビットマイクロ秒単位で指定

※ T-Kernel 2.0ではtk_sta_alm_u()が追加

マイクロ秒単位の時間管理機能

- マイクロ秒単位を扱う追加システムコール
  - 時刻の設定や取得
    tk_get_tim_u, tk_set_tim_uなど
  - 周期ハンドラやアラームハンドラの生成
    tk_cre_cyc_u, tk_sta_alm_u
  - 待ちに入るシステムコールのタイムアウト指定
    tk_slp_tsk_u, tk_wai_sem_u, tk_wai_flg_uなど
  - タスク遅延(**tk_dly_tsk()**)での待ち時間指定
  - 時間情報を含むタスクやハンドラなどの状態参照
    tk_ref_tsk_u, tk_inf_tsk_u, tk_ref_cyc_uなど

マイクロ秒単位の時間管理機能



大容量デバイスへの対応

- デバイス管理のAPIの開始位置(start)を64ビット対応に
- 指定可能な範囲
  - 512バイト/セクタのHDDでは、32ビット(約2G)で約1TB
  - 開始位置(start)を64ビットにしてこの制限を解消
- 64ビット指定のシステムコールは純粋に追加
  - 32ビット指定のAPIはそのまま残し、混在利用も可能

大容量デバイスへの対応

- **64**ビットを扱うシステムコールは ´**_d**´ を付加

  例:「デバイスへの書き込み」を行う**API**

  - **tk_wri_dev(ID dd、W start、VP buf、W size、TMO tmout)**

    書込み開始位置**start**は**32**ビットで指定

    タイムアウト時間**tmout**は**32**ビットミリ秒単位で指定

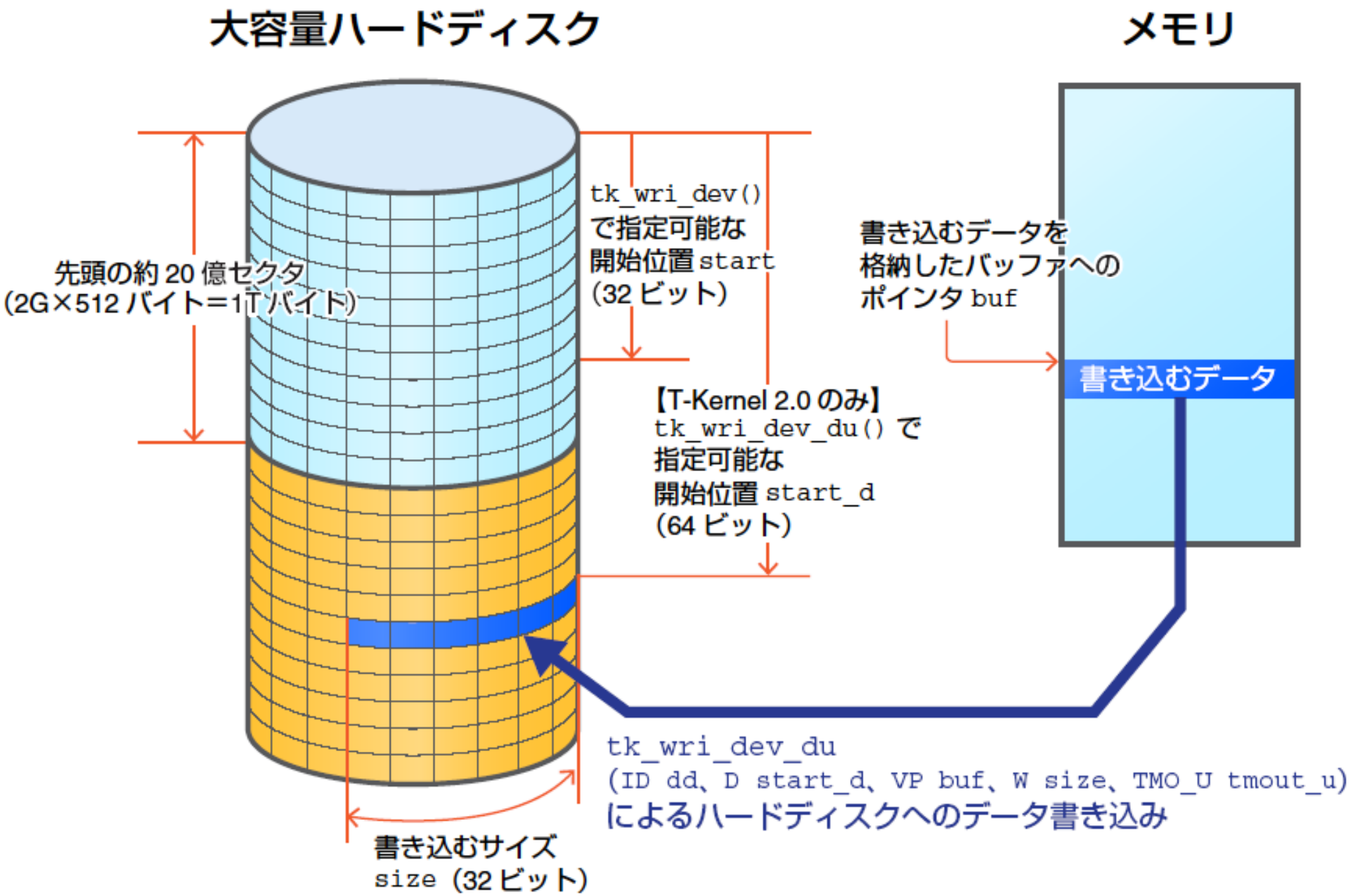
  - **tk_wri_dev_du(ID dd、D start_d、VP buf、W size、TMO_U tmout_u)**

    書込み開始位置**start_d**は**64**ビットで指定

    タイムアウト時間**tmout_u**は**64** ビットマイクロ秒で指定

※ T-Kernel 2.0では**tk_wri_dev_du()**が追加

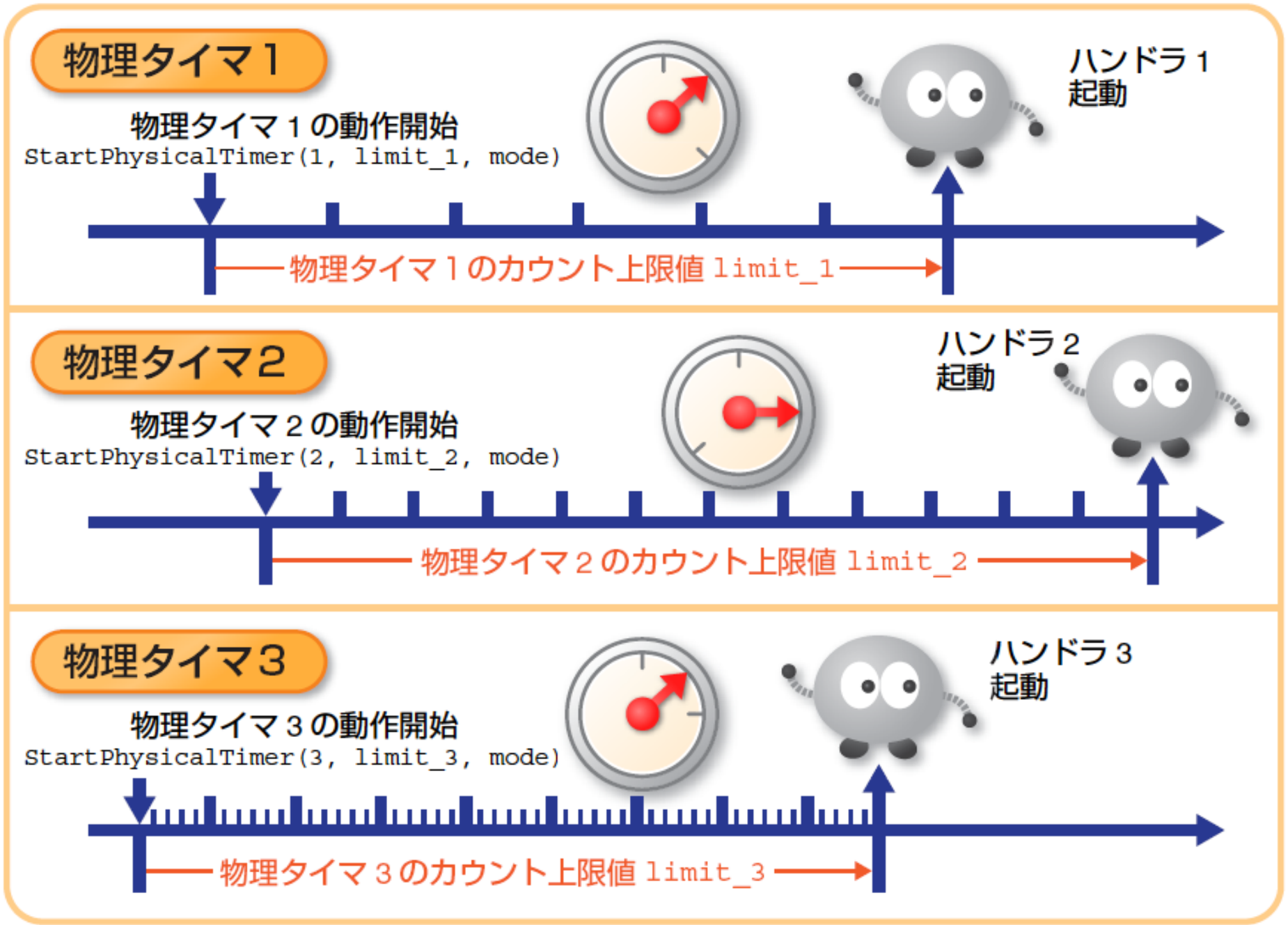
大容量デバイスへの対応



物理タイマ機能

- **SoC(System on a Chip)**上の豊富なハードウェアタイマを活用
- **1**つのハードウェアタイマが**1**つの物理タイマに対応
- 複数の物理タイマが独立して動作
- 機能はプリミティブ
  - 一定周期で単純にインクリメントするカウンタ
  - カウント値が上限値に達すると**0**に戻る
  - カウント値が**0**に戻る際にハンドラを実行
  - ワンショットの動作と周期的な動作を選択

物理タイマ機能



※物理タイマ１～物理タイマ３はすべて独立して動作する。

第五章
T-Kernelの機能

T-Kernelの機能（T-Kernel/OS）

- T-Kernel/OS (Operating System)
  - タスク管理機能
  - タスク付属同期機能
  - タスク例外処理機能
  - 同期・通信機能
  - 拡張同期・通信機能
  - メモリプール管理機能
  - 時間管理機能
  - 割込み管理機能
  - システム状態管理機能
  - サブシステム管理機能

T-Kernelの機能（T-Kernel/SM）

- T-Kernel/SM (System Manager)
  - システムメモリ管理機能
  - アドレス空間管理機能
  - デバイス管理機能
  - 割込み管理機能
  - I/Oポートアクセスサポート機能
  - 省電力機能
  - システム構成情報管理機能
  - メモリキャッシュ制御機能
  - 物理タイマ機能
  - ユーティリティ機能

T-Kernelの機能（ T-Kernel/DS）

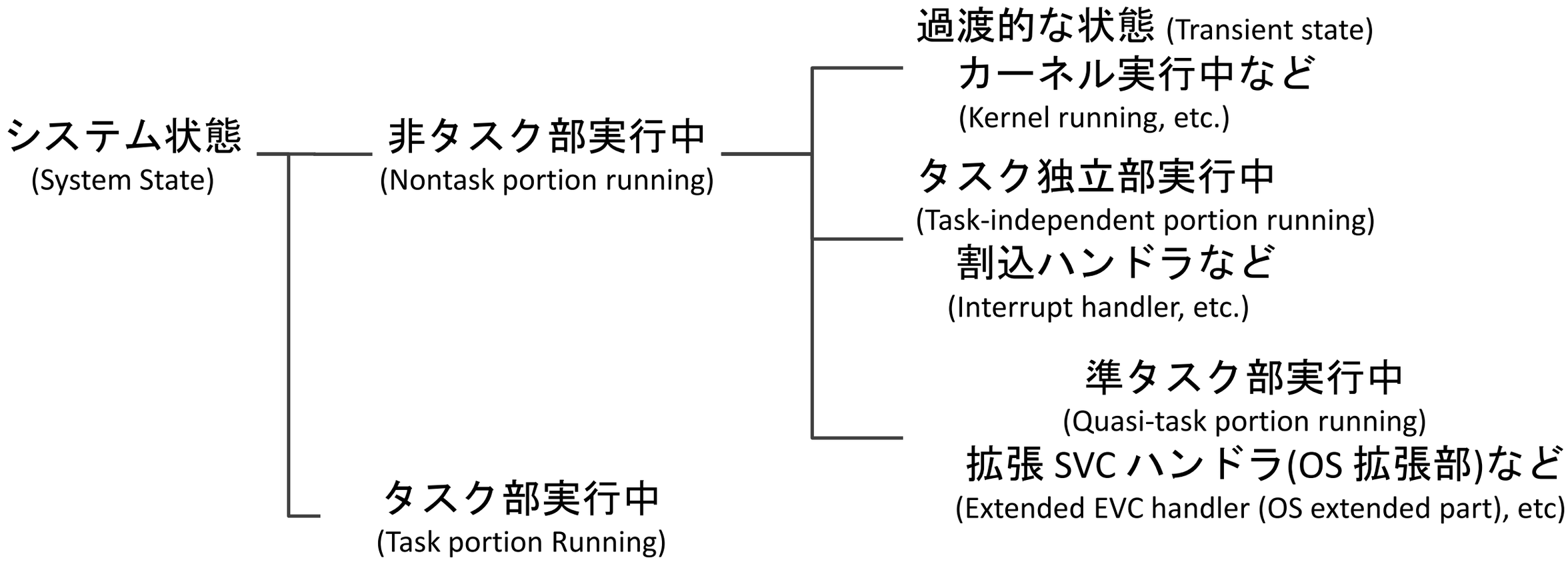
- T-Kernel/DS (Debugger Support)
  - カーネル内部状態取得機能
  - 実行トレース機能

4-1. T-Kernel/OS

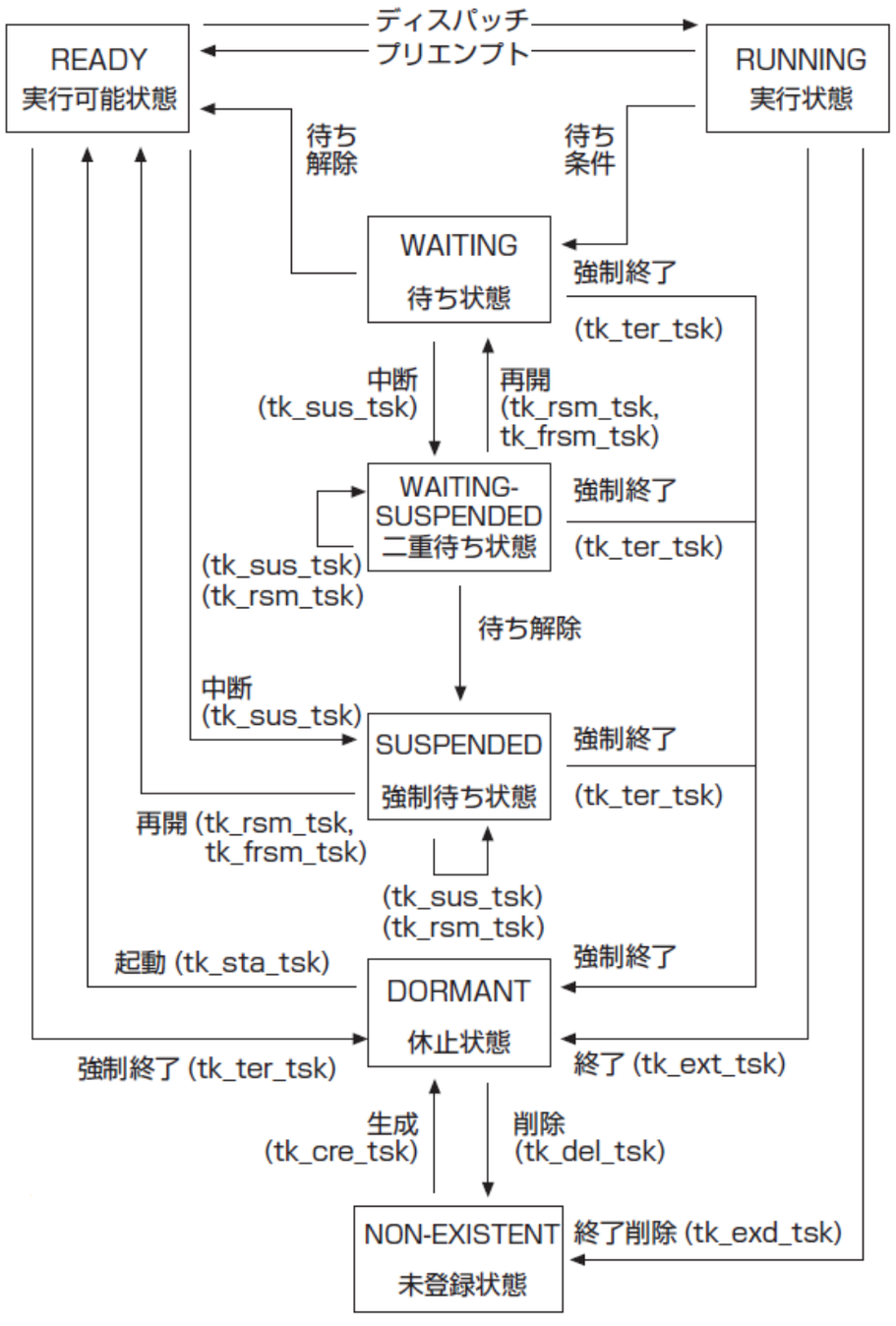
T-Kernel/OSの機能

- タスク管理機能
- タスク付属同期機能
- タスク例外処理機能
- 同期・通信機能
  - セマフォ
  - イベントフラグ
  - メールボックス
- 拡張同期・通信機能
  - ミューテックス
  - メッセージバッファ
  - ランデブ
- メモリプール管理機能
  - 固定長メモリプール
  - 可変長メモリプール
- 時間管理機能
  - システム時刻管理
  - 周期ハンドラ
  - アラームハンドラ
- 割込み管理機能
- システム状態管理機能
- サブシステム管理機能

T-Kernelにおけるシステム状態



タスクの状態遷移

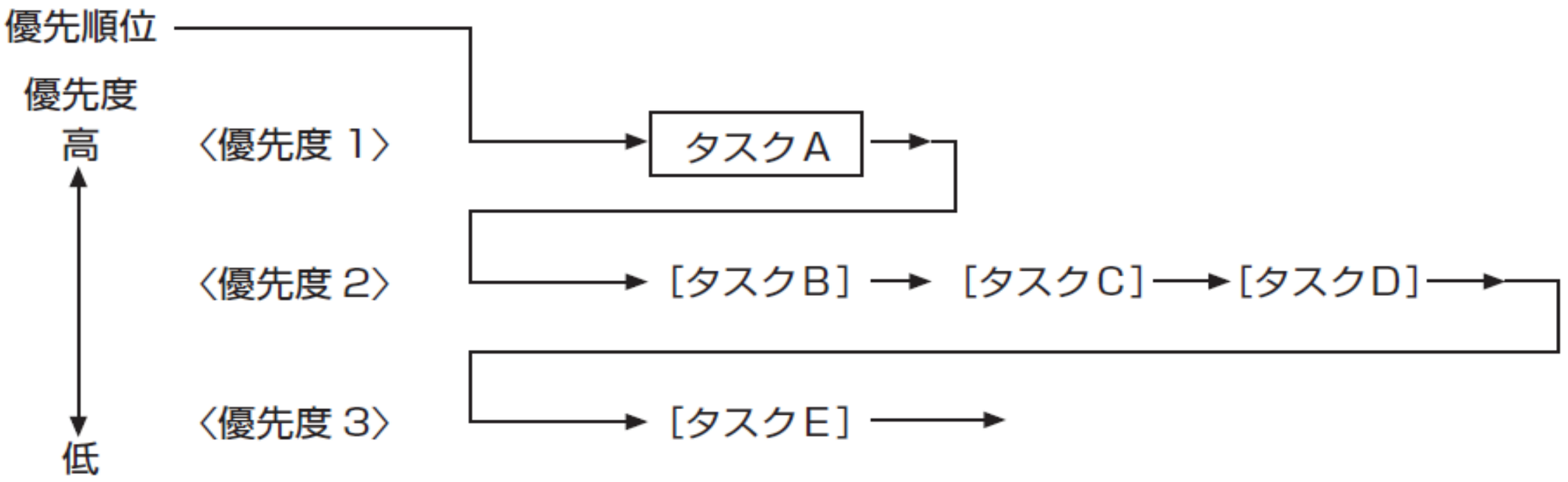


- マルチタスク動作を実現する為に、外部や内部の事象により、タスクは様々な状態をとる
  - 実行可能状態（READY）
    - 動く準備は整っているが、他に優先度の高いタスクが実行しているため、CPU割付を待っている状態
  - 実行状態（RUNNING）
    - CPUが割付けられ実行している状態
  - 待ち状態（WAITING）
    - 何らかの事象を待っている状態
    - 自らの要求でのみ、この状態に遷移

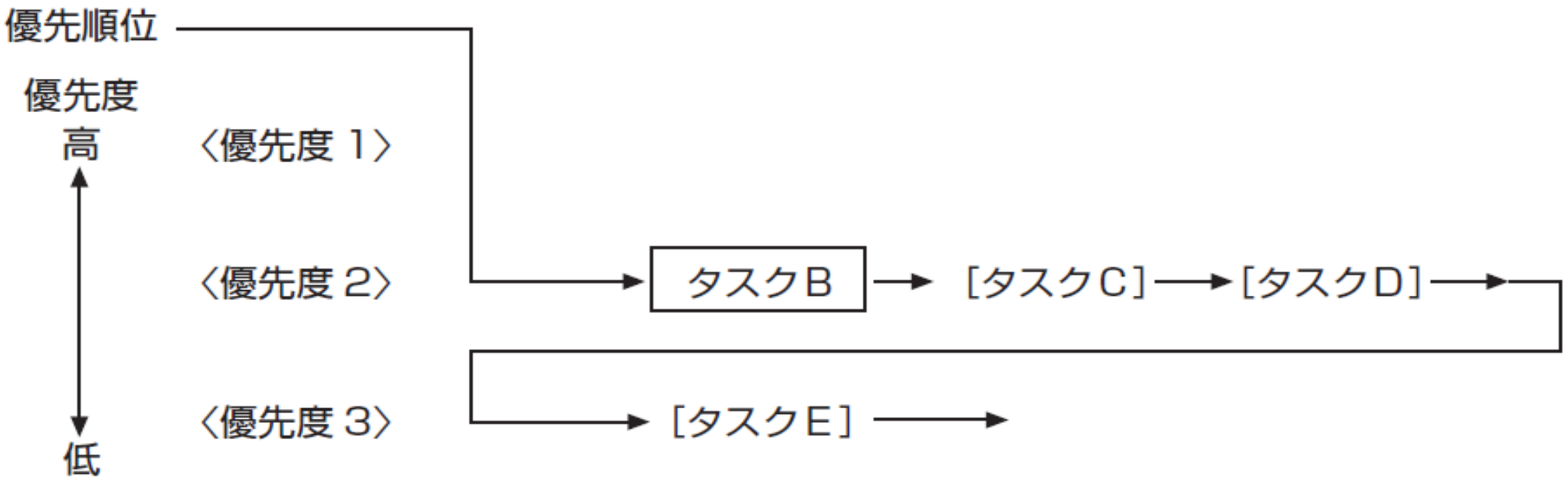
スケジューリング

- スケジューリングルール：優先度ベース
  - カーネルは、READY状態の中で最も高い優先度を持つタスクを実行させる。
  - 同一優先度のタスクが複数存在する場合は、FCFS（First Come First Service）で実行させる。
  - 優先度は小さな値ほど高い。
  - 優先度は、tk_chg_priシステムコールで変更可能。
- スケジューリングタイミング：イベントドリブン
  - イベントドリブン、つまり「何らかの事象」が発生したタイミングで、カーネルは実行すべきタスクを再検索する。
  - 「何らかの事象」とは、具体的には「システムコール」を意味する。
  - 逆に言うと、カーネルに対して何らシステムコールを要求しなければ、同じタスクが実行し続けることになる。
- ラウンドロビンスケジューリング
  - ラウンドロビンスケジューリングとは、一定時間毎にレディキューを回転させ同一優先度を持つタスクのCPU割付け時間を平均化するスケジューリング。
  - tk_chg_sltシステムコールでラウンドロビンスケジューリングを実現できる。

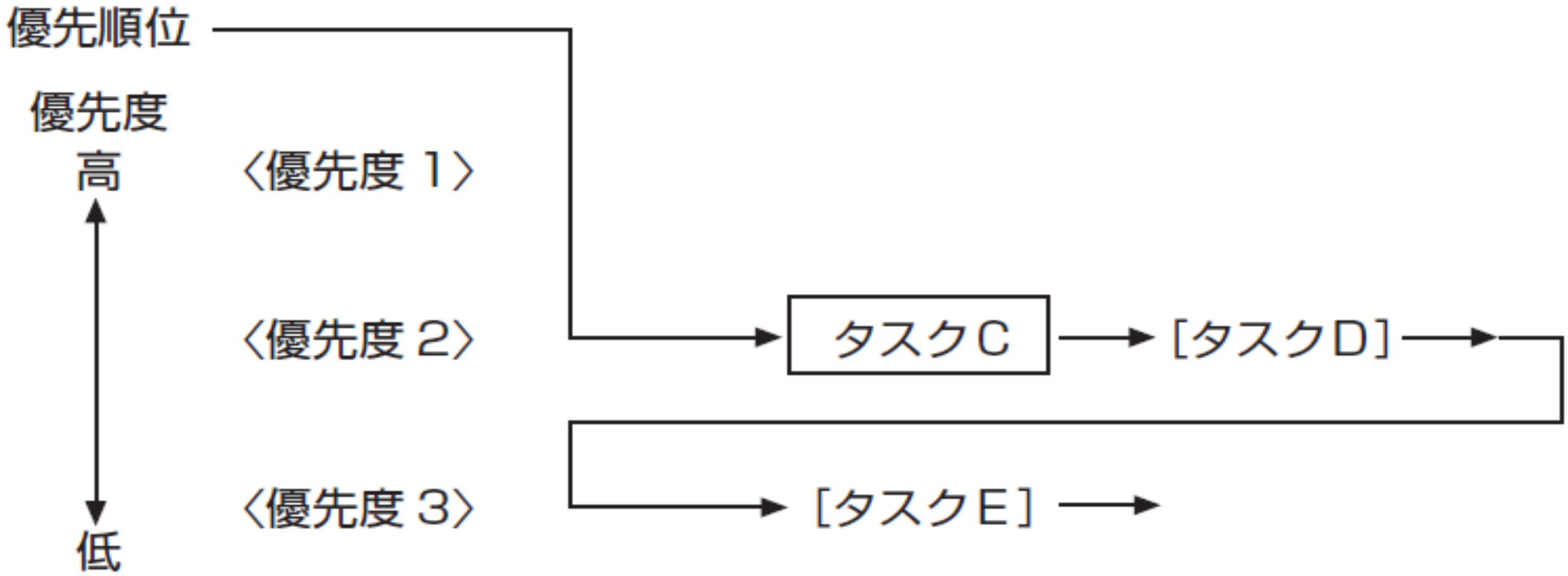
【スケジューリングの例】
タスクレディキュー（初期状態）
タスクAが実行状態(RUNNING)



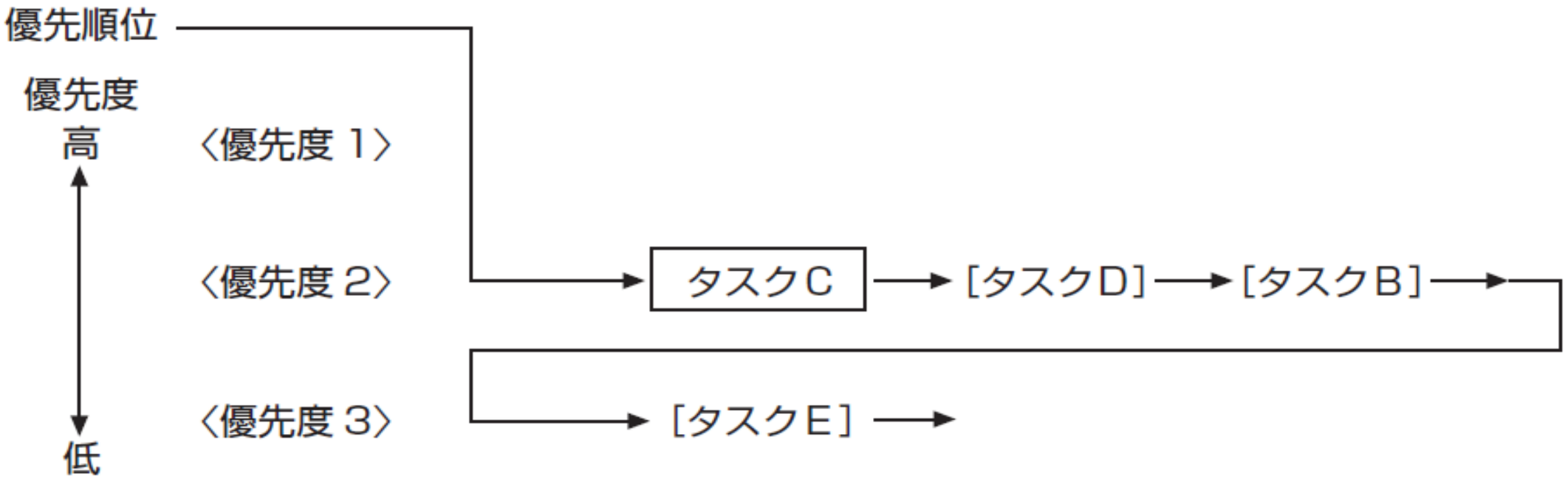
【スケジューリングの例】
タスクAが終了
タスクBが実行状態(RUNNING)になる



【スケジューリングの例】
タスクBが待ち状態(WAITING)になった後、
タスクC が実行状態(RUNNING) になる

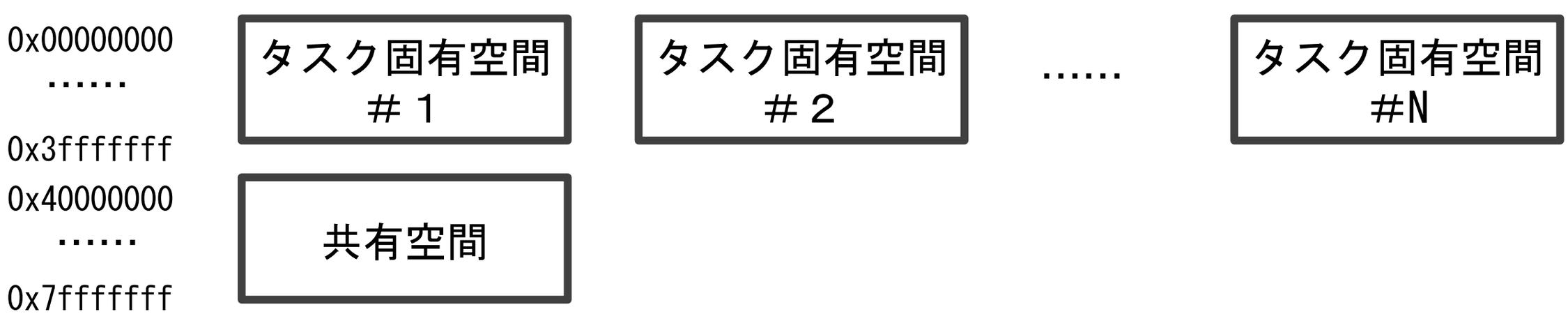


【スケジューリングの例】
タスクBが実行可能状態(READY)状態に戻る



メモリのアドレス空間

論理アドレス（例）



メモリ保護レベル

- T-Kernelでは0～3の保護レベルがある。
  - メモリには保護レベルがある。
  - 実行部も保護レベルを持つ。
- 保護レベルによるメモリ保護
  - 実行部の保護レベルと同じか、またはそれよりも低いレベルのメモリにのみアクセス可能。
- 各保護レベルの用途
  - “0”＝カーネル、サブシステム、デバイスドライバなど
    - ユーザ作成の処理もカーネルと同じ保護レベルで実行しうる
  - “1”＝システムアプリケーションタスク
  - “2”＝予約
  - “3”＝ユーザアプリケーションタスク
- 実行部の保護レベルの遷移は、システムコールや拡張SVC呼び出し、割込み、CPU例外によって行われる。

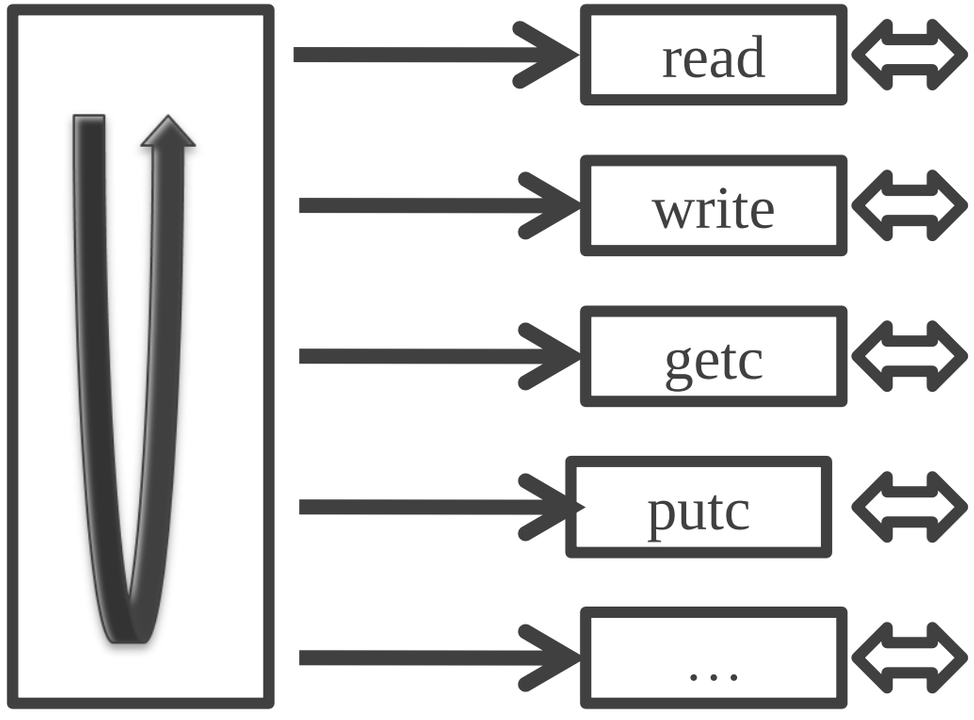
(A)　タスク管理機能
タスク付属同期機能
タスク例外処理機能

組込みシステムの典型的なコード構成

Internal Control

情報処理の
ロジック
（内部）

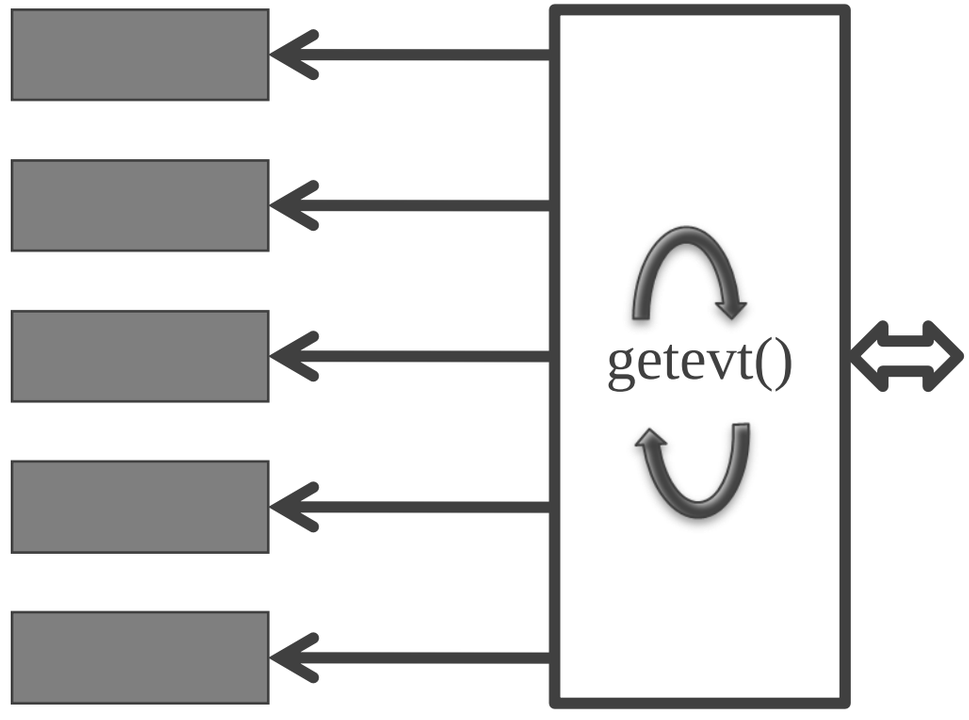
I/O 処理
（外部）



External Control

情報処理の
ロジック
（内部）

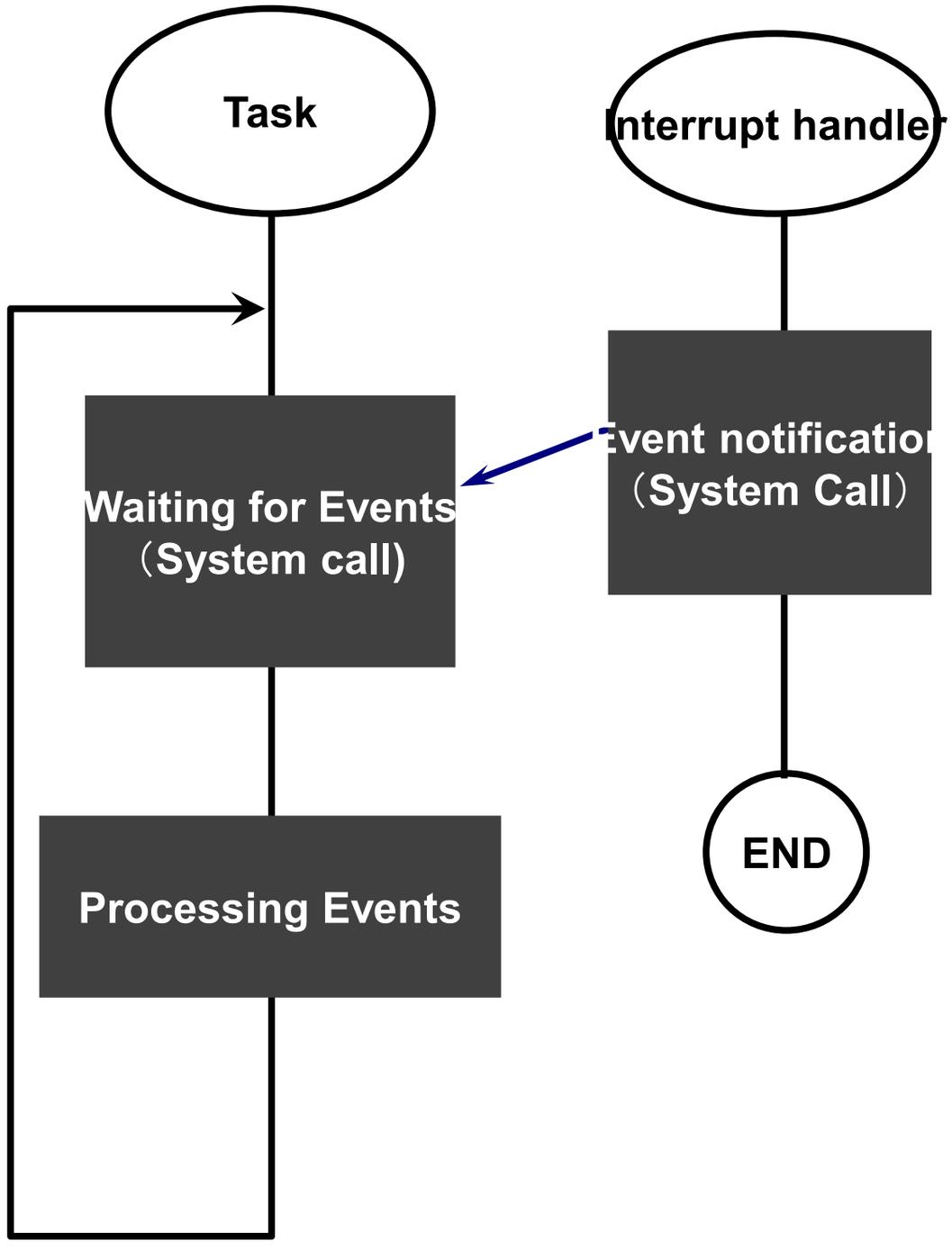
I/O 処理
（外部）



Interrupt Handler + Backend Task　　Kernel

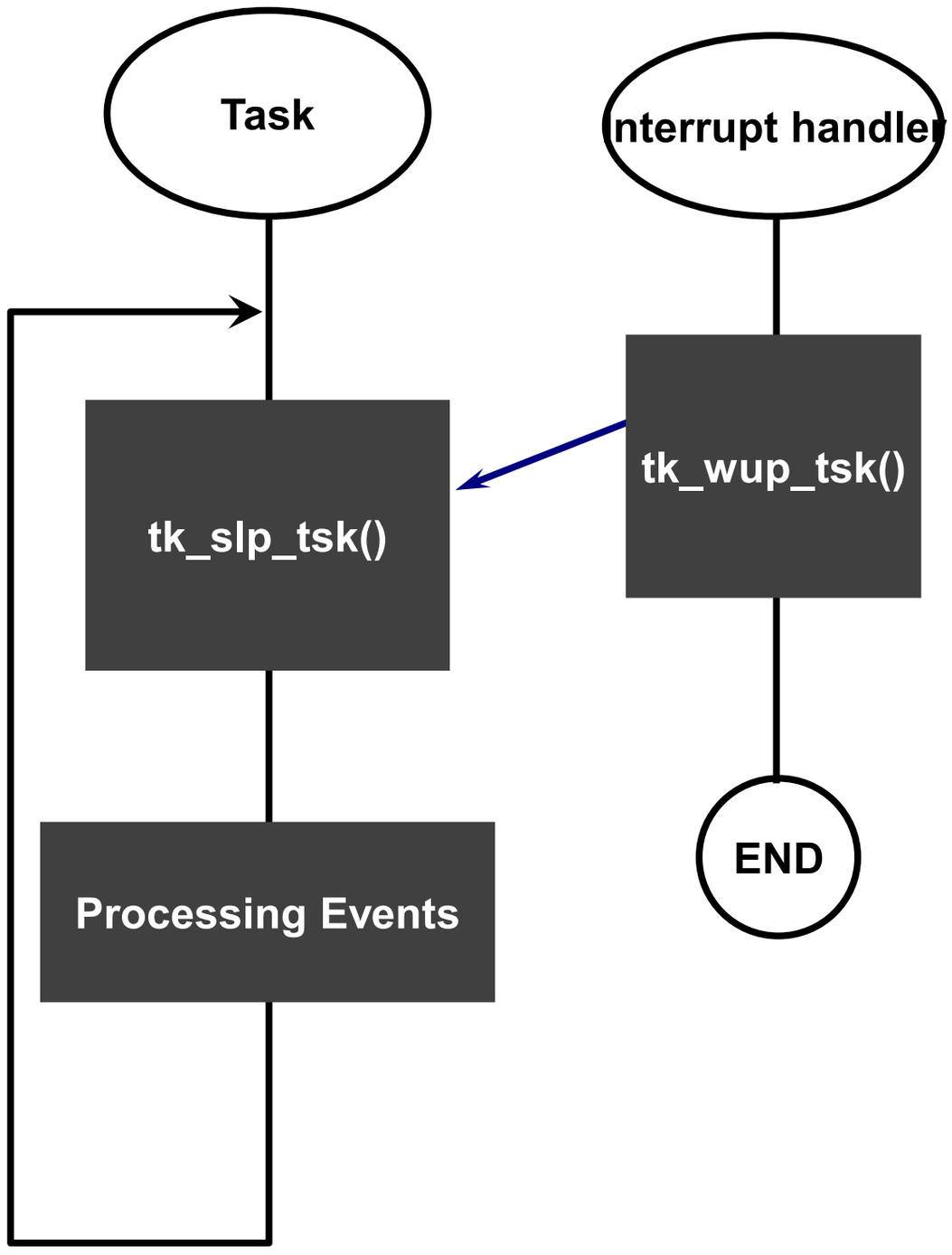
例１：基本的な処理のパターンの実現(1)

- 右図のようにイベントとの同期により処理することが一般的。
- 割込みハンドラや他のタスクからのシステムコールにより待ち状態(WAITING)が解除。
- このシステムコールは「イベント通知」という意味を持つ。
- イベントの発生時だけタスクが動く。必要な時だけ動くのでCPUの無駄使いがない。
- T-Kernelは、様々な同期機構を用意している。



例１：基本的な処理のパターンの実現(2)

- tk_slp_tsk(tmout)：単に待つ。（RUNNING→WAITING)
  - tmout経過後はエラーコードE_TMOUTで復帰。（tmout=TMO_FEVRにより、タイムアウトまでの時間は無限大を示す：タイムアウト指定無しの意味）
  - tk_wup_tskによってWAITINGが解除された場合は、エラーコードE_OKで復帰。
- tk_wup_tsk(tskid)：tk_slp_tskにより待っているタスクのWAITINGを解除。
- イベント待ちにtk_slp_tskを使い、イベントの通知にtk_wup_tskを使う。
- 最も単純な同期方法だが、イベントを通知する方は相手タスクIDが既知の必要がある。
- イベント待ち側・通知側の関係が密なため、拡張性は優れていない。



タスク管理機能

タスクの状態を直接的に操作／参照する機能

tk_cre_tsk()	タスク生成
tk_del_tsk()	タスク削除
tk_sta_tsk()	タスク起動
tk_ext_tsk()	自タスク終了
tk_exd_tsk()	自タスクの終了と削除
tk_ter_tsk()	他タスク強制終了
tk_ref_tsk()	タスク状態参照
tk_ref_tsk_u()	タスク状態参照(マイクロ秒単位)

初期化タスクでの処理例

```
{
  ER       ercd;
  T_CTSK   ctsk_A, ctsk_B;             /* タスク生成情報 */
  ID       tskid_A, tskid_B;           /* タスクID       */
     ここでタスク構造体 ctsk_A, ctsk_B を設定
  tskid_A = tk_cre_tsk(&ctsk_A);       /* タスク生成     */
  tskid_B = tk_cre_tsk(&ctsk_B);
  ercd  = tk_sta_tsk(tskid_A, mbxId); /* タスク起動     */
  ercd  = tk_sta_tsk(tskid_B, mbxId);
  ercd  = tk_slp_tsk(TMO_FEVR);        /* タスクスリープ */
}
```

※エラー処理は入っていません。

...前頁のシステムコールの説明詳細

ID tskid = tk_cre_tsk (CONST T_CTSK *pk_ctsk);

- pk_ctskの内容に従ってタスクを生成する。
- 生成されたタスクは休止状態(DORMANT)になる。

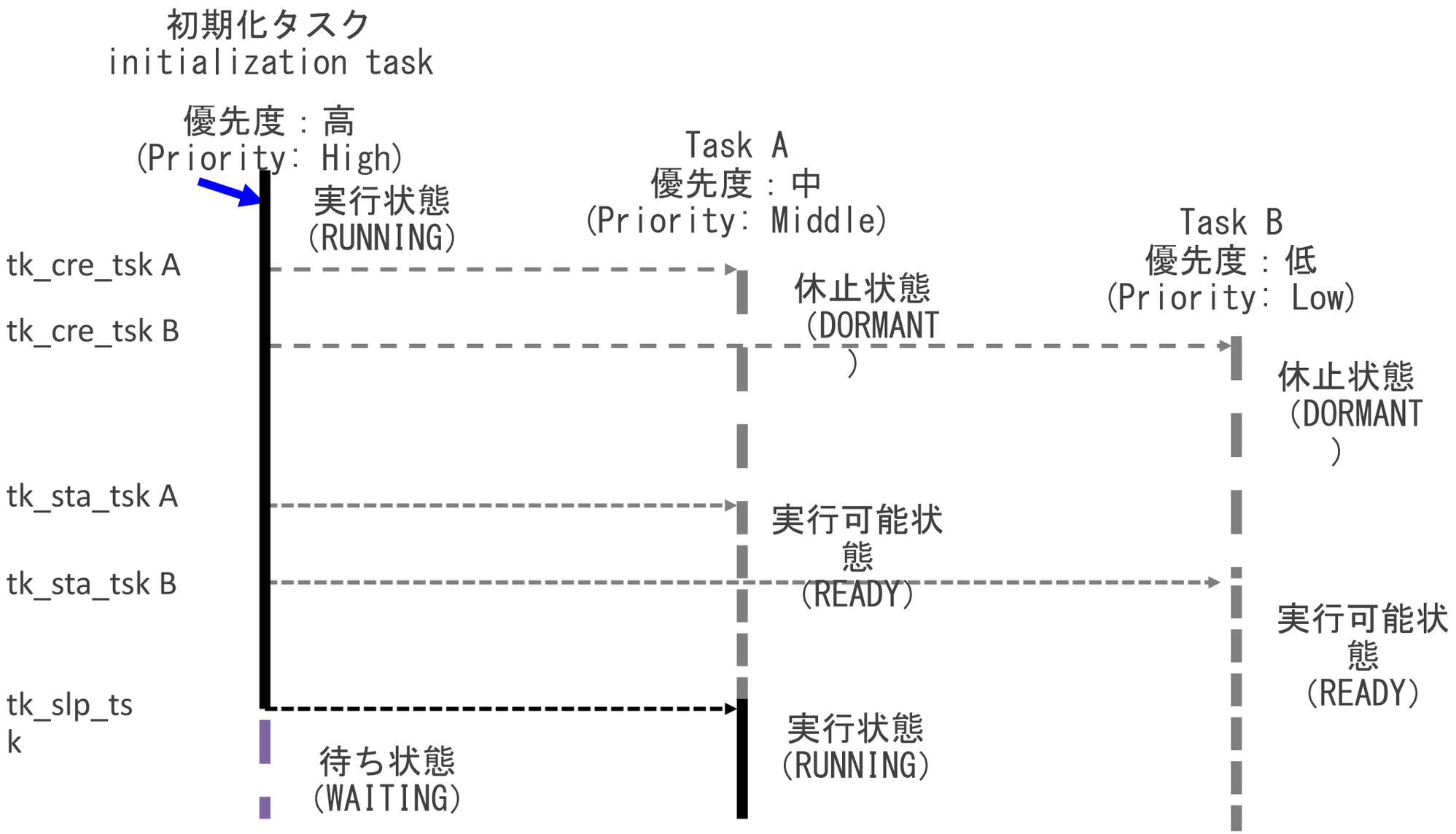
ER ercd = tk_sta_tsk (ID tskid , INT stacd);

- tskidで指定したタスクを起動する。実行可能状態(READY)になる。

ER ercd = tk_slp_tsk (TMO tmout);

- 起床待ち状態になる。tmoutで待ち時間(ミリ秒)を指定できる。

例２：初期化タスクの流れ

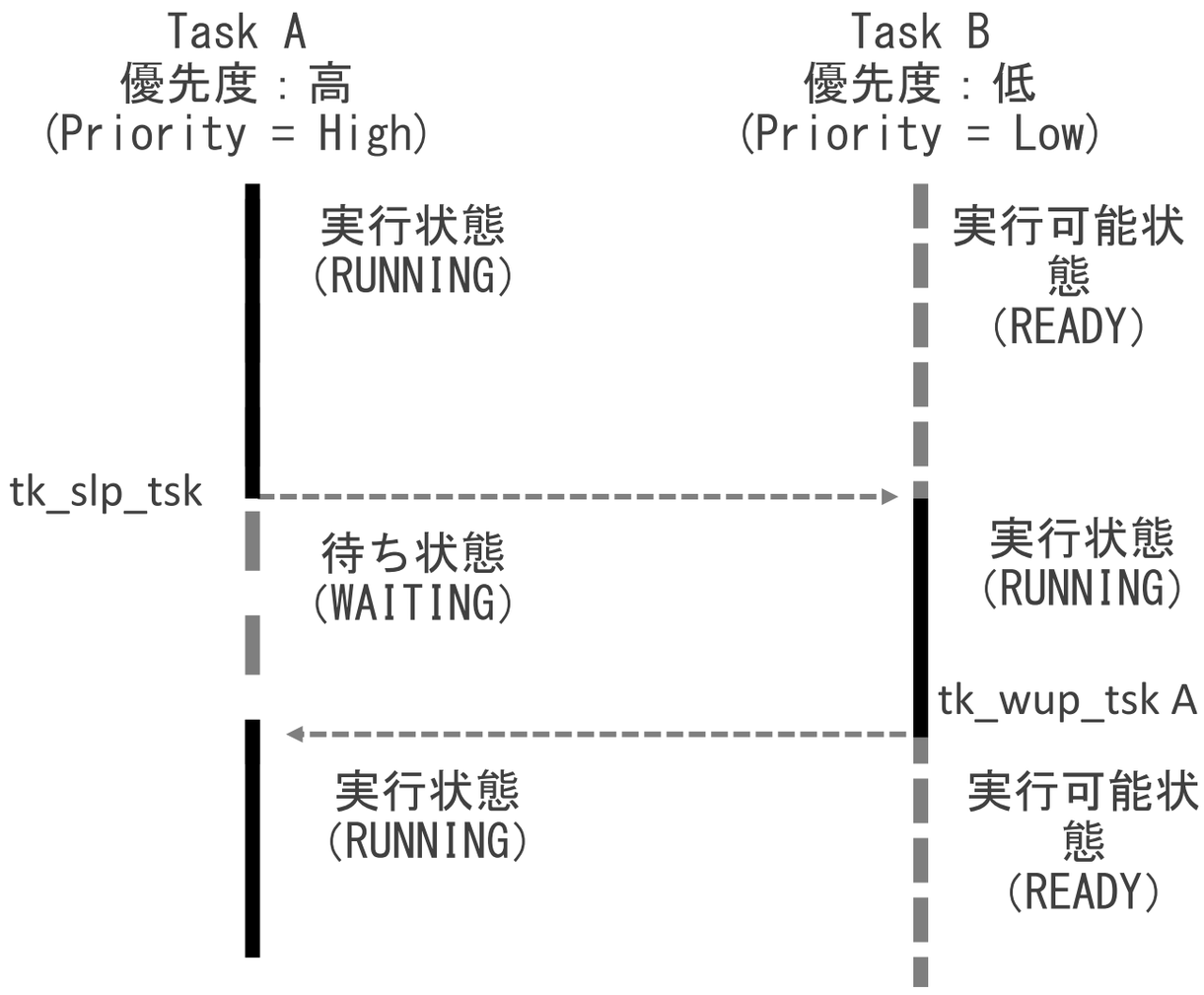


タスク付属同期機能

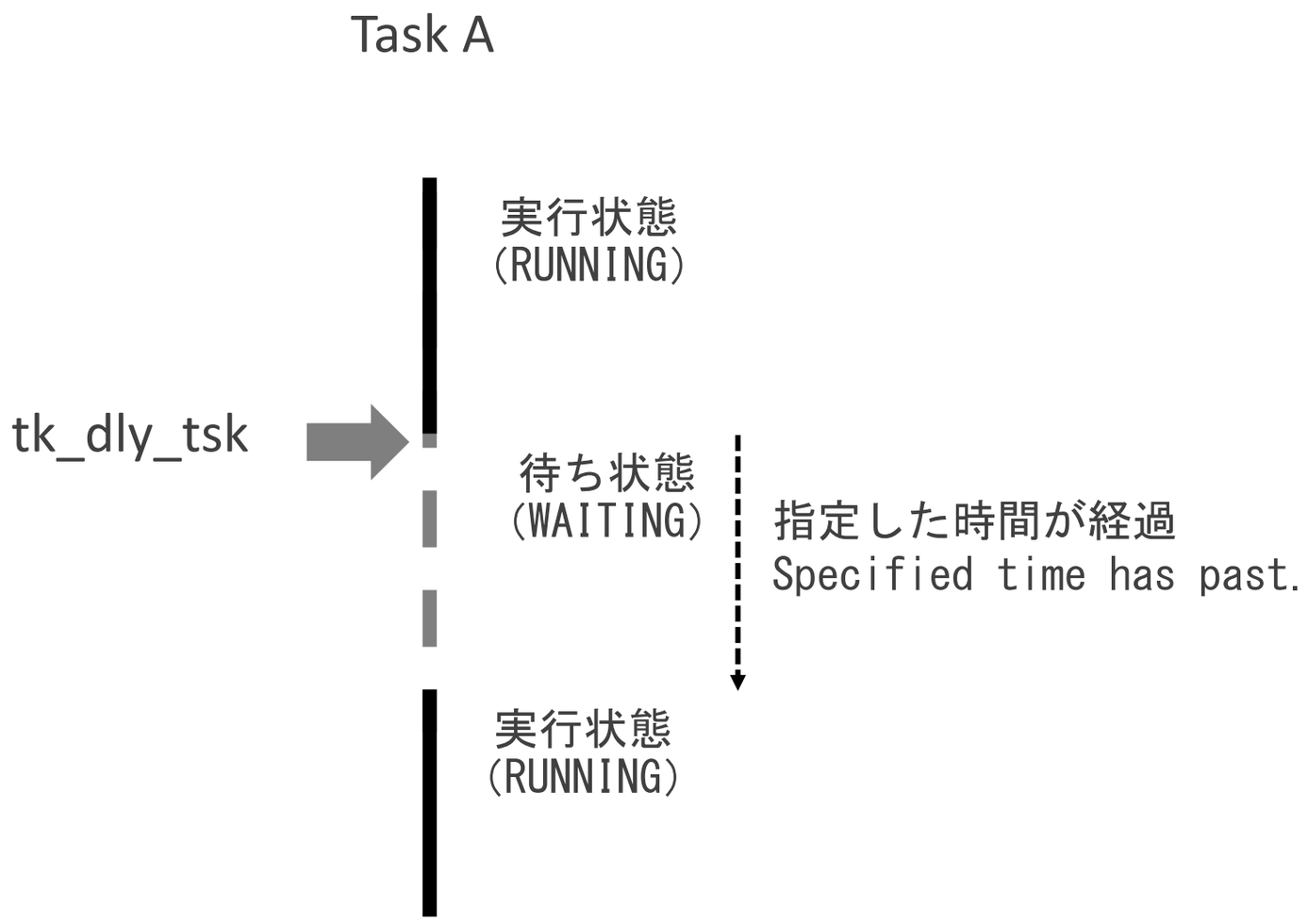
タスク状態を直接操作して同期を行う機能

tk_slp_tsk()	自タスクを起床待ち状態へ移行
tk_slp_tsk_u()	自タスクを起床待ち状態へ移行(マイクロ秒単位)
tk_wup_tsk()	他タスクの起床
tk_dly_tsk()	タスク遅延
tk_dly_tsk_u()	タスク遅延(マイクロ秒単位)

例３：起床待ちと起床(Sleep and wake-up)



例４：タスク遅延(Delay task)



タスク例外処理機能

タスクに発生した例外事象に対する処理を、タスクのコンテキストで行うための機能

tk_def_tex ()	タスク例外ハンドラの定義
tk_ena_tex()	タスク例外の許可
tk_dis_tex()	タスク例外の禁止
tk_ras_tex()	タスク例外を発生
tk_end_tex()	タスク例外ハンドラの終了
tk_ref_tex()	タスク例外の状態参照

(B)　同期・通信機能
拡張同期・通信機能

同期・通信機能（セマフォ）

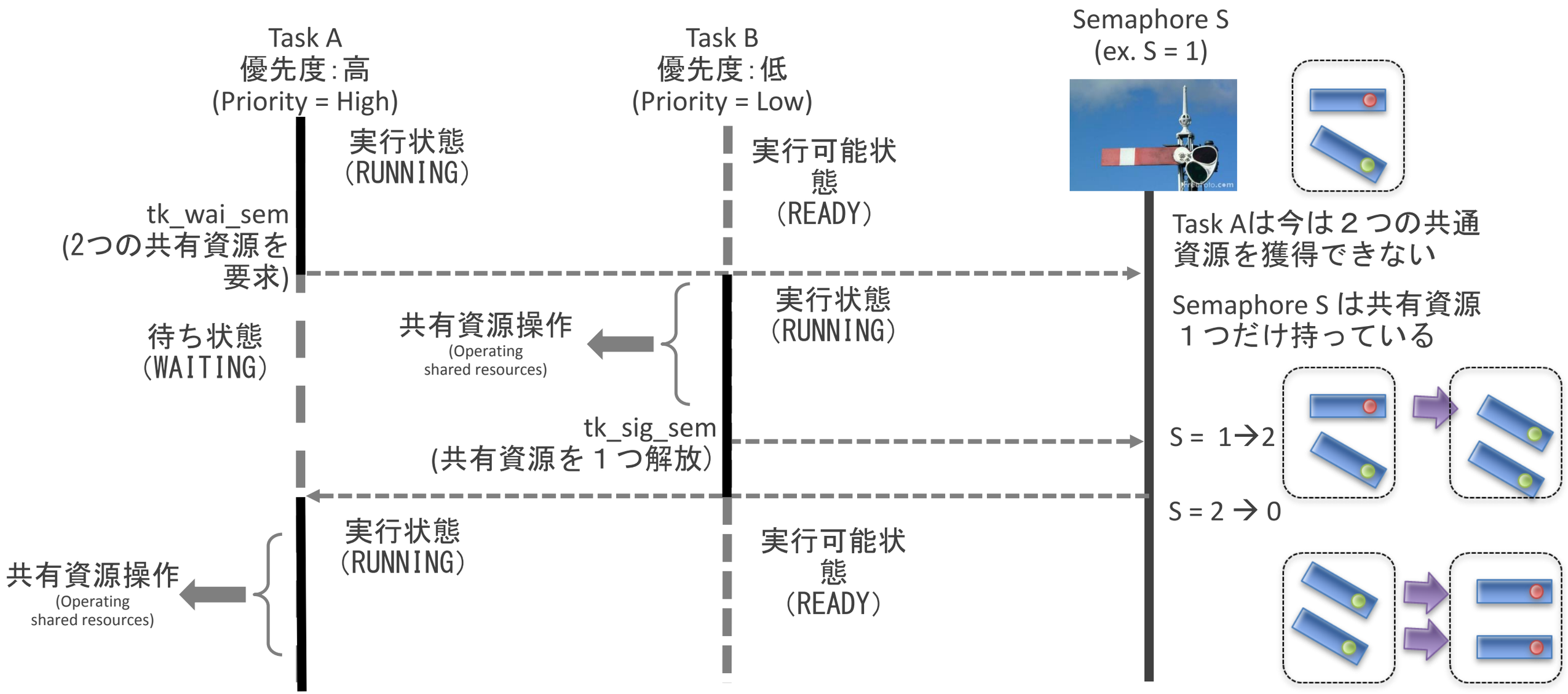
セマフォは、利用可能な共有資源の数をカウンタで表し、共有資源の排他制御を実現する機能

tk_cre_sem()	セマフォ生成
tk_del_sem()	セマフォ削除
tk_sig_sem()	セマフォ資源返却
tk_wai_sem()	セマフォ資源獲得
tk_wai_sem_u()	セマフォ資源獲得(マイクロ秒単位)

T-Kernel 2.0

（使い方）資源を利用する前に、利用する資源の数だけセマフォのカウンタから獲得し、終わると返却する。カウンタが必要な数を持ってないと、他タスクが返却するのを待つことで、共有資源の排他制御を実現

セマフォ（Semaphore）の実行例

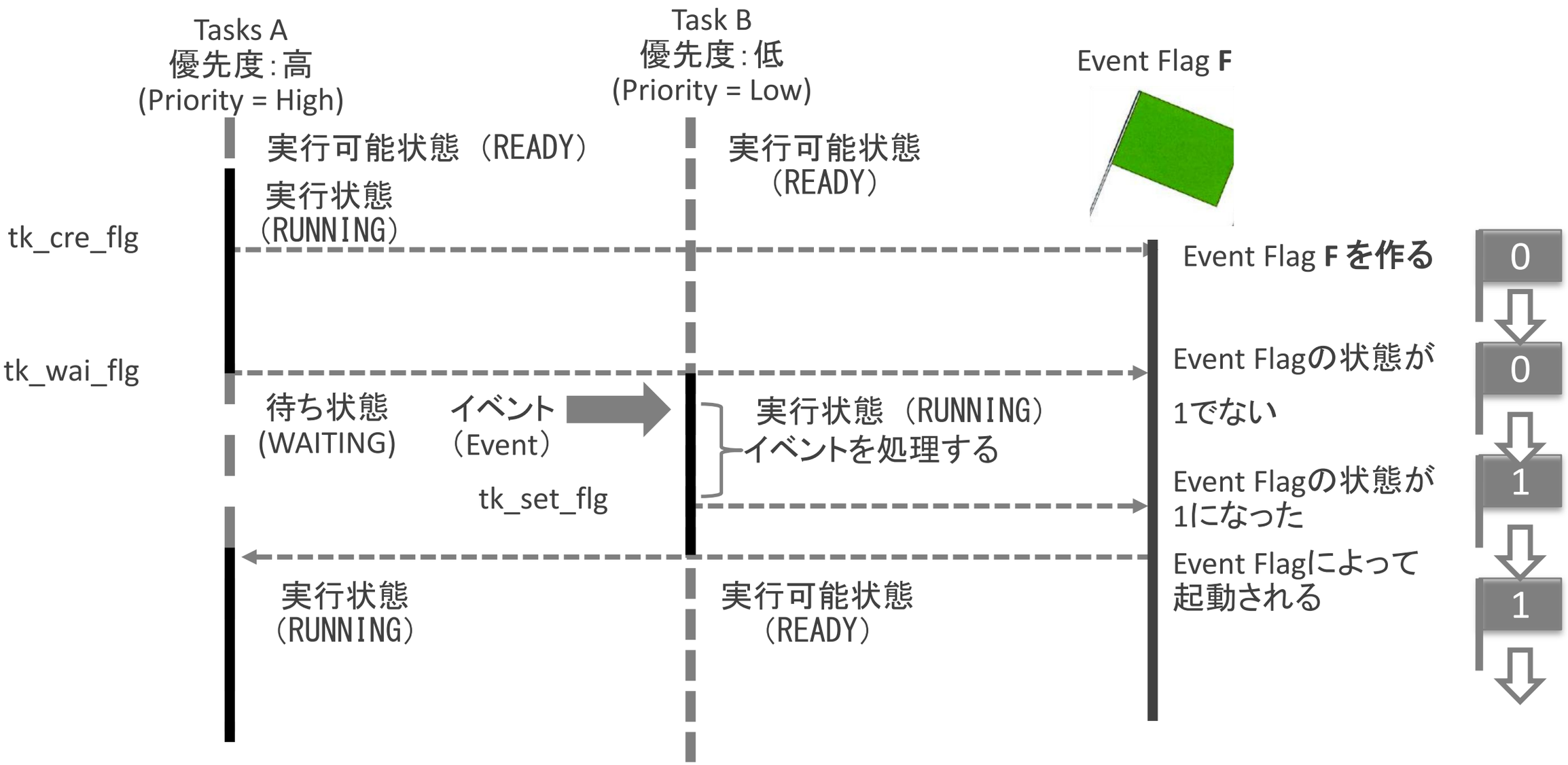


同期・通信機能（イベントフラグ）

イベントフラグは、イベントの有無をビット毎のフラグで表現して、同期する機能

tk_cre_flg()	イベントフラグ生成
tk_del_flg()	イベントフラグ削除
tk_set_flg()	イベントフラグのセット
tk_clr_flg()	イベントフラグのクリア
tk_wai_flg()	イベントフラグ待ち
tk_wai_flg_u()	イベントフラグ待ち(マイクロ秒単位)

イベントフラグ（Event Flag）の実行例

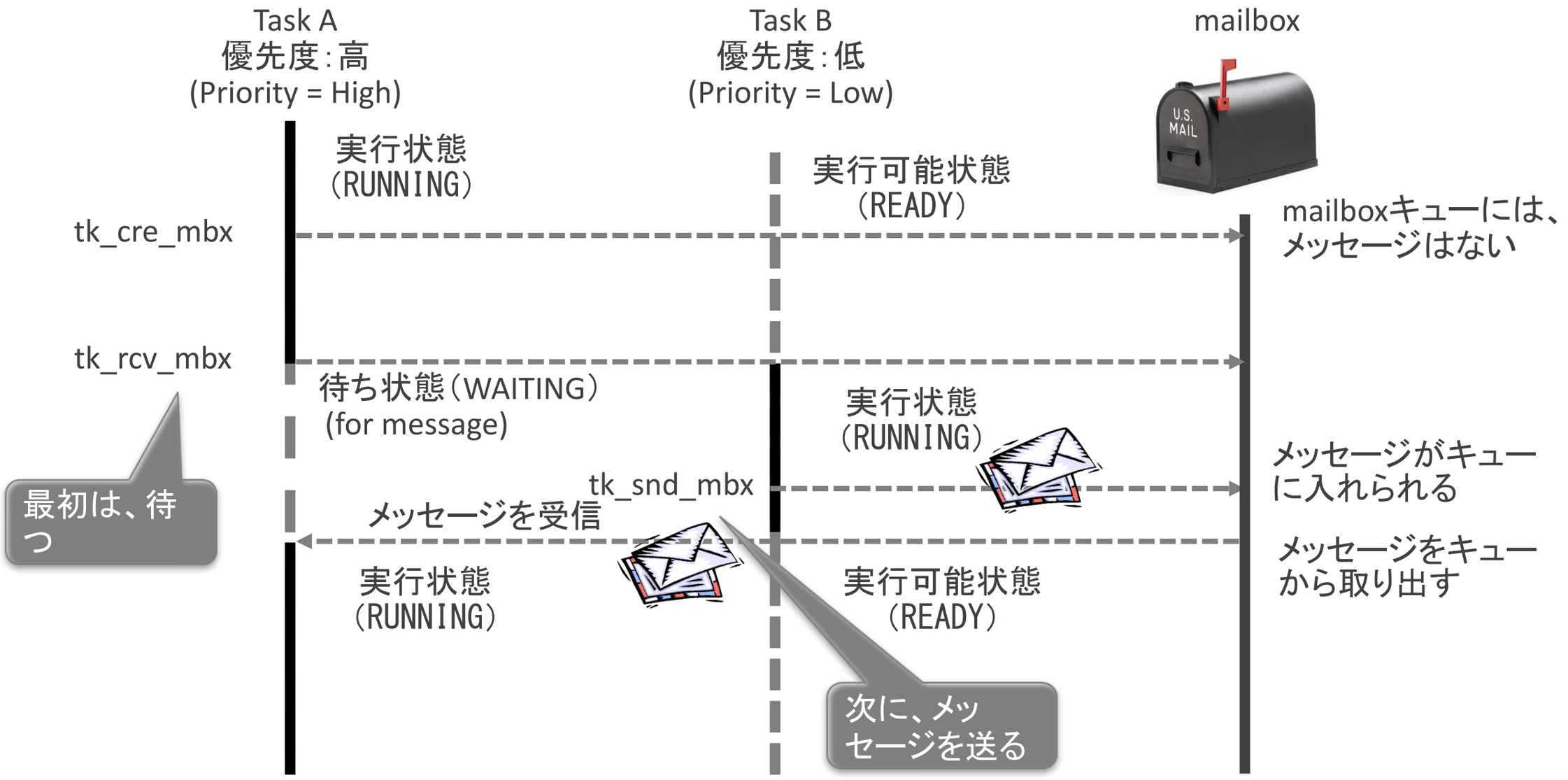


同期・通信機能（メールボックス）

メールボックスは、共有メモリ上に置かれたメッセージを受け渡しして、同期通信を行う機能

tk_cre_mbx()	メールボックス生成
tk_del_mbx()	メールボックス削除
tk_snd_mbx()	メールボックスへ送信
tk_rcv_mbx()	メールボックスから受信
tk_rcv_mbx_u()	メールボックスから受信(マイクロ秒単位)

メールボックス（Mailbox）の実行例



拡張同期・通信機能（ミューテックス）

ミューテックスは、共有資源に関するタスク間の排他制御を実現
優先度逆転を防ぐために、優先度継承プロトコル、優先度上限
プロトコルをサポートしている。

tk_cre_mtx()	ミューテックス生成
tk_del_mtx()	ミューテックス削除
tk_loc_mtx()	ミューテックスのロック
tk_loc_mtx_u()	ミューテックスのロック(マイクロ秒単位)
tk_unl_mtx()	ミューテックスのアンロック
tk_ref_mtx()	ミューテックス状態参照

拡張同期・通信機能（メッセージバッファ）

メッセージバッファは、可変長のメッセージを受渡しすることにより、
同期と通信を行う機能
メッセージバッファの領域サイズを調整することで、
同期メッセージ、非同期メッセージの両方を実現可能

tk_cre_mbf()	メッセージバッファ生成
tk_del_mbf()	メッセージバッファ削除
tk_snd_mbf()	メッセージバッファへ送信
tk_snd_mbf_u()	メッセージバッファへ送信(マイクロ秒単位)
tk_rcv_mbf()	メッセージバッファから受信
tk_rcv_mbf_u()	メッセージバッファから受信(マイクロ秒単位)
tk_ref_mbf()	メッセージバッファ状態参照

拡張同期・通信機能（ランデブ）

ランデブ機能は、複数のタスクがサーバとクライアントの関係にある場合に、それらのタスク間での同期通信を行う機能
ビットパターンによるランデブ条件によって、通常のクライアントサーバーモデルよりも柔軟な同期通信を実現できる。

tk_cre_por()	ランデブポート生成
tk_del_por()	ランデブポート削除
tk_cal_por()	ランデブポートに対するランデブの呼出
tk_cal_por_u()	ランデブポートに対するランデブの呼出 (マイクロ秒単位)
tk_acp_por()	ランデブポートに対するランデブ受付
tk_acp_por_u()	ランデブポートに対するランデブ受付 (マイクロ秒単位)

拡張同期・通信機能（ランデブ）

tk_fwd_por()	ランデブポートに対するランデブ回送
tk_rpl_rdv()	ランデブ返答
tk_ref_por()	ランデブポート状態参照

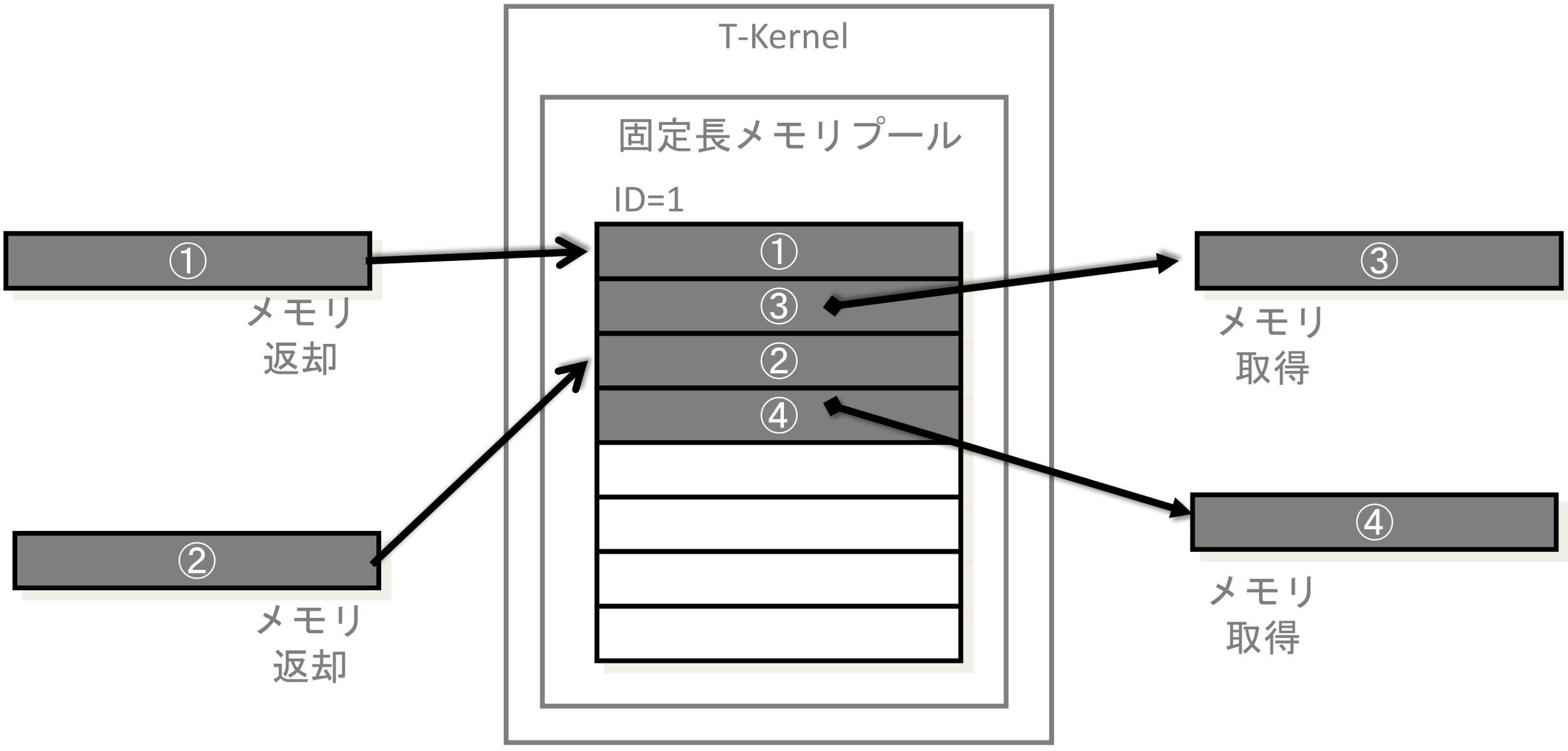
(C) メモリプール管理機能

メモリプール管理機能(固定長メモリプール)

固定長メモリプールは、固定サイズのメモリブロックを動的に管理する機能

tk_cre_mpf()	固定長メモリプール生成
tk_del_mpf()	固定長メモリプール削除
tk_get_mpf()	固定長メモリブロック獲得
tk_get_mpf_u()	固定長メモリブロック獲得(マイクロ秒単位)
tk_rel_mpf()	固定長メモリブロック返却

固定長メモリプール機能の動作



取得時のメモリブロックサイズが固定サイズ

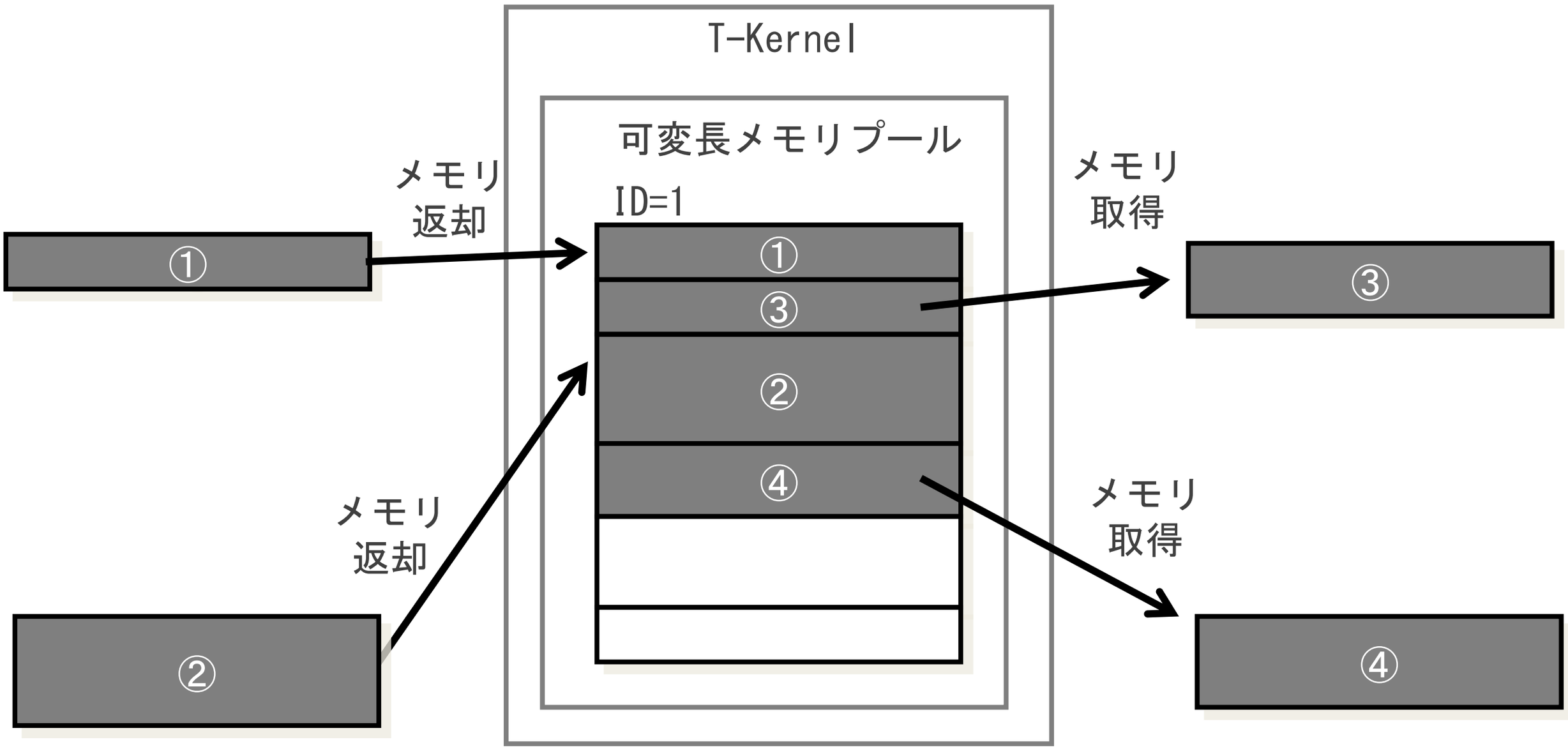
可変長メモリプールよりは柔軟性に欠けるが、分断の心配はなく、かつ高速。

メモリプール管理機能(可変長メモリプール)

可変長メモリプールは、任意サイズのメモリブロックを動的に管理する機能

tk_cre_mpl()	可変長メモリプール生成
tk_del_mpl()	可変長メモリプール削除
tk_get_mpl()	可変長メモリブロック獲得
tk_get_mpl_u()	可変長メモリブロック獲得(マイクロ秒単位)
tk_rel_mpl()	可変長メモリブロック返却

可変長メモリプール機能の動作



取得時のメモリブロックサイズが任意サイズ

手軽だが、獲得・解放を繰り返すうちにプール内部が分断され、大きなサイズが獲得できなくなる可能性がある。
(T-Kernelは、分断されたメモリ領域を統合する機能までは持っていない）。

(D) 時間管理機能

時間管理機能(システム時刻管理)

システム時刻を操作する機能

tk_set_tim()	システム時刻設定
tk_set_tim_u()	システム時刻設定(マイクロ秒単位)
tk_get_tim()	システム時刻参照
tk_get_tim_u()	システム時刻参照(マイクロ秒単位)

時間管理機能（周期ハンドラ）

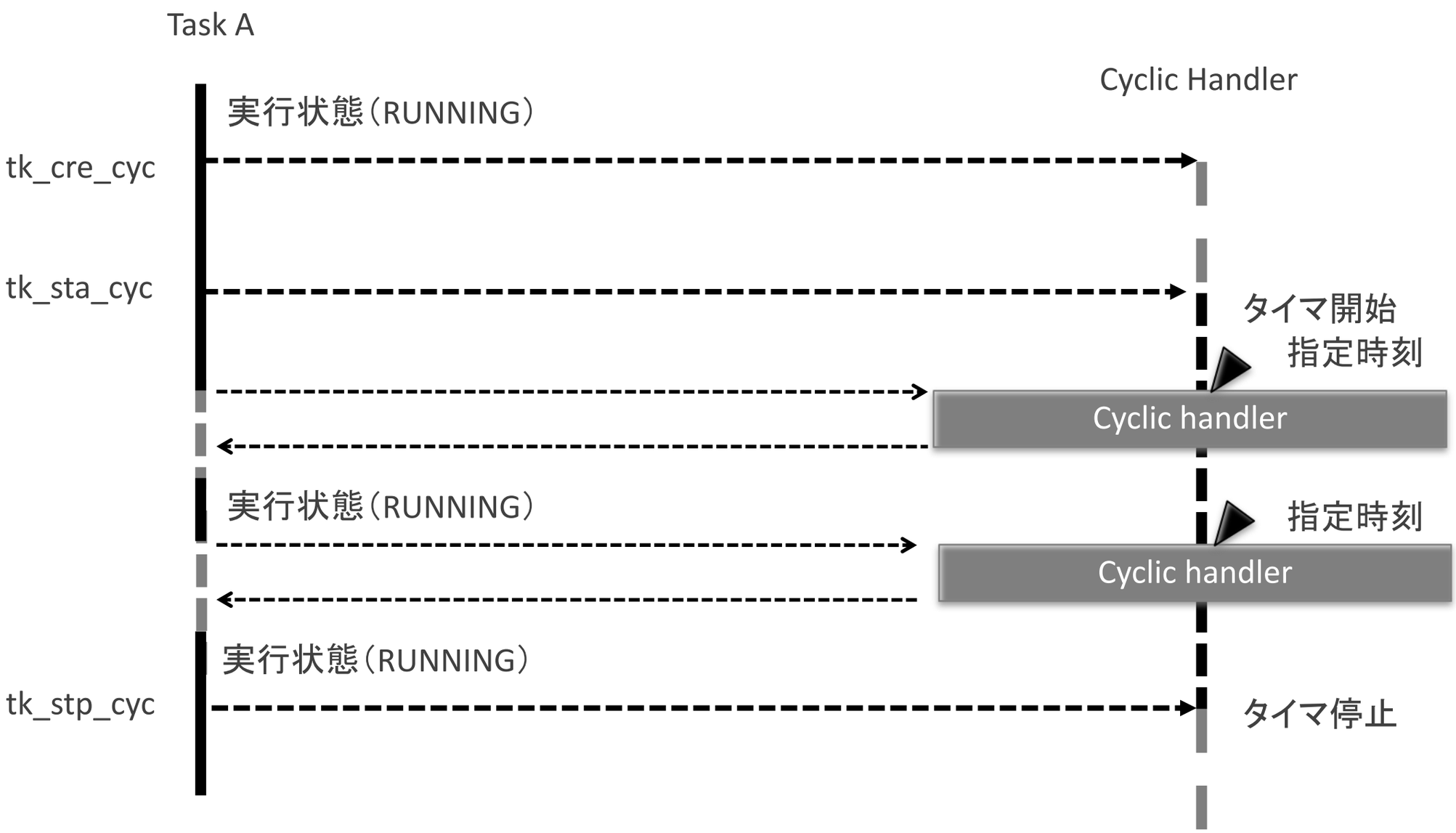
周期ハンドラは、一定周期で起動されるタイムイベントハンドラ

tk_cre_cyc()	周期ハンドラの生成
tk_cre_cyc_u()	周期ハンドラの生成(マイクロ秒単位)
tk_del_cyc()	周期ハンドラの削除
tk_sta_cyc()	周期ハンドラの動作開始
tk_stp_cyc()	周期ハンドラの動作停止

T-Kernel 2.0

！周期ハンドラは、タスク独立部で動作

周期起動ハンドラ（Cyclic Handler）の実行例



tk_cre_cyc() :　　周期ハンドラの生成

tk_sta_cyc() :　　周期ハンドラの動作開始

tk_stp_cyc() :　　周期ハンドラの動作停止

時間管理機能（アラームハンドラ）

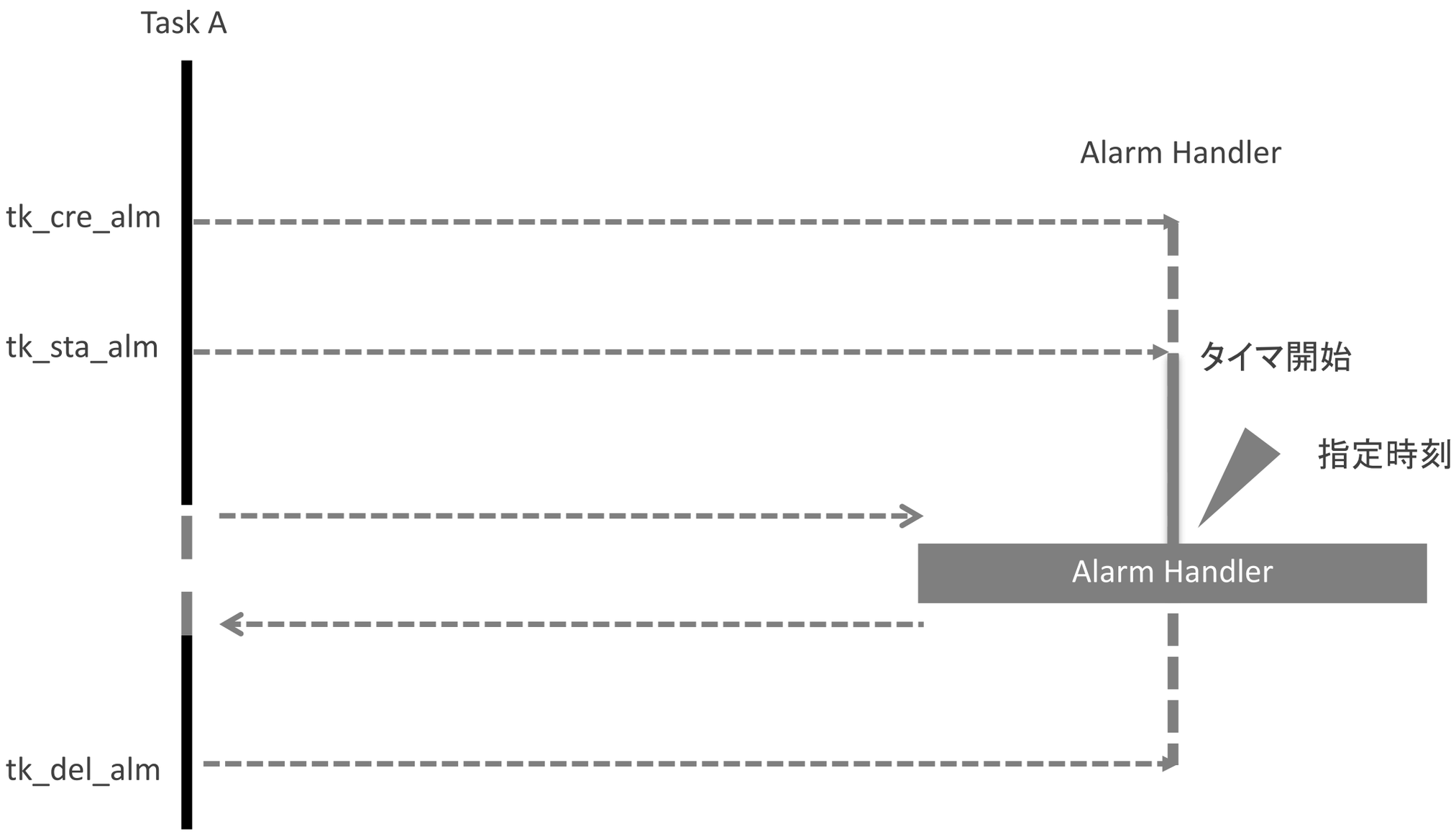
アラームハンドラは、指定した時間に起動されるタイムイベントハンドラ

tk_cre_alm()	アラームハンドラの生成
tk_del_alm()	アラームハンドラの削除
tk_sta_alm()	アラームハンドラの動作開始
tk_sta_alm_u()	アラームハンドラの動作開始(マイクロ秒単位)
tk_stp_alm()	アラームハンドラの動作停止

T-Kernel 2.0

！アラームハンドラは、タスク独立部で動作

アラームハンドラ（Alarm Handler）の実現例



(E) 割込み管理機能

割込み管理機能

外部割込みおよびCPU例外に対するハンドラ定義などをの操作を行う機能

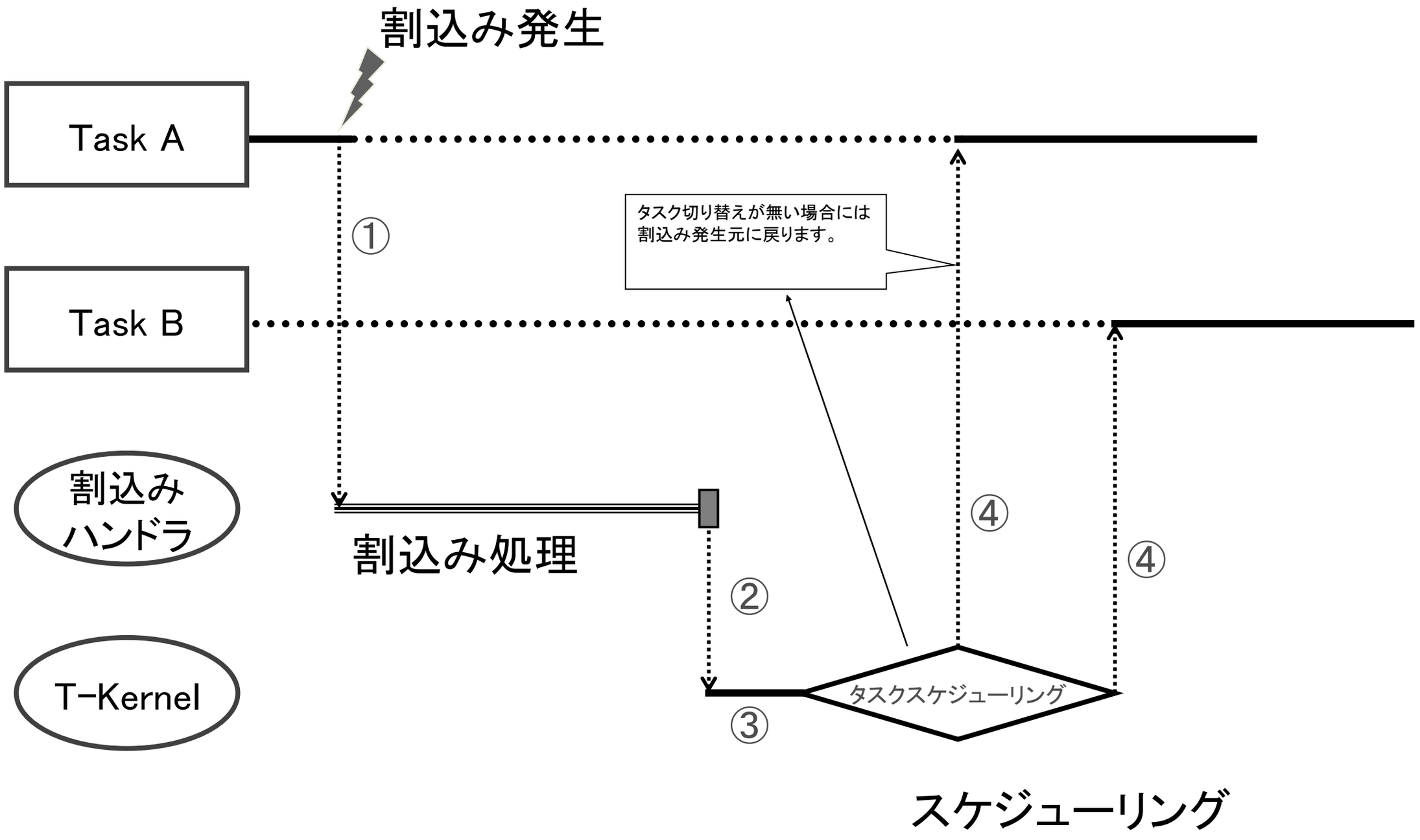
tk_def_int()	割込みハンドラ定義
tk_ret_int()	割込みハンドラから復帰

！割込みハンドラは、タスク独立部で動作

！TA_HLNG属性を指定して定義された高級言語で書かれた割込みハンドラからは、 tk_ret_int()を呼び出してはいけない。

- 高級言語内のルーチンから暗黙的に復帰相当の機能が実行されるため

割込み処理動作（例）


割込み発生
Task A
Task B
割込み
ハンドラ
T-Kernel
①
タスク切り替えが無い場合には
割込み発生元に戻ります。
割込み処理
②
③
④
④
タスクスケジューリング
スケジューリング

(F) システム状態管理機能

システム状態管理機能

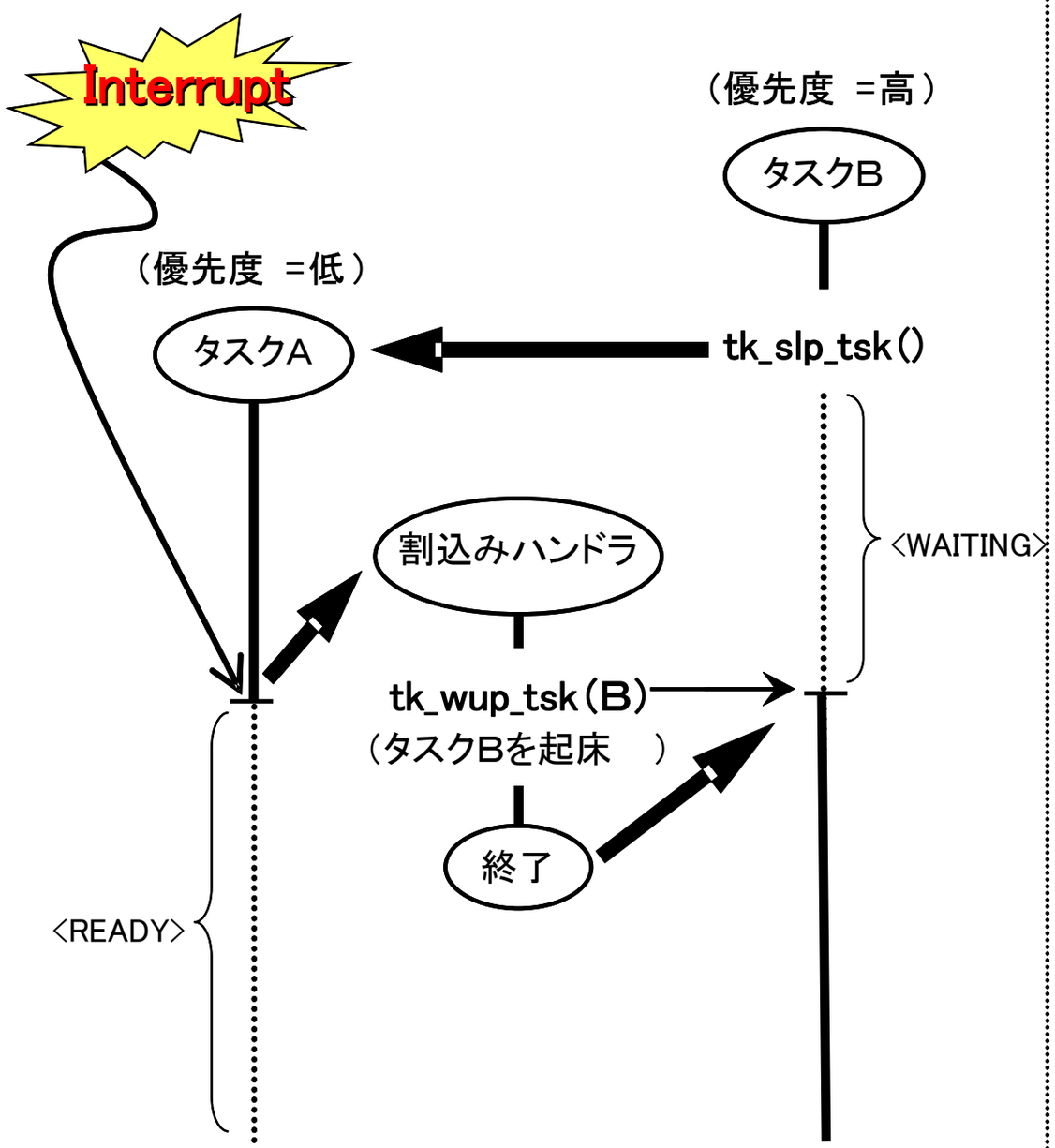
システムの状態を変更／参照する機能

tk_dis_dsp()	ディスパッチ禁止
tk_ena_dsp()	ディスパッチ許可
tk_set_pow()	省電力モード設定

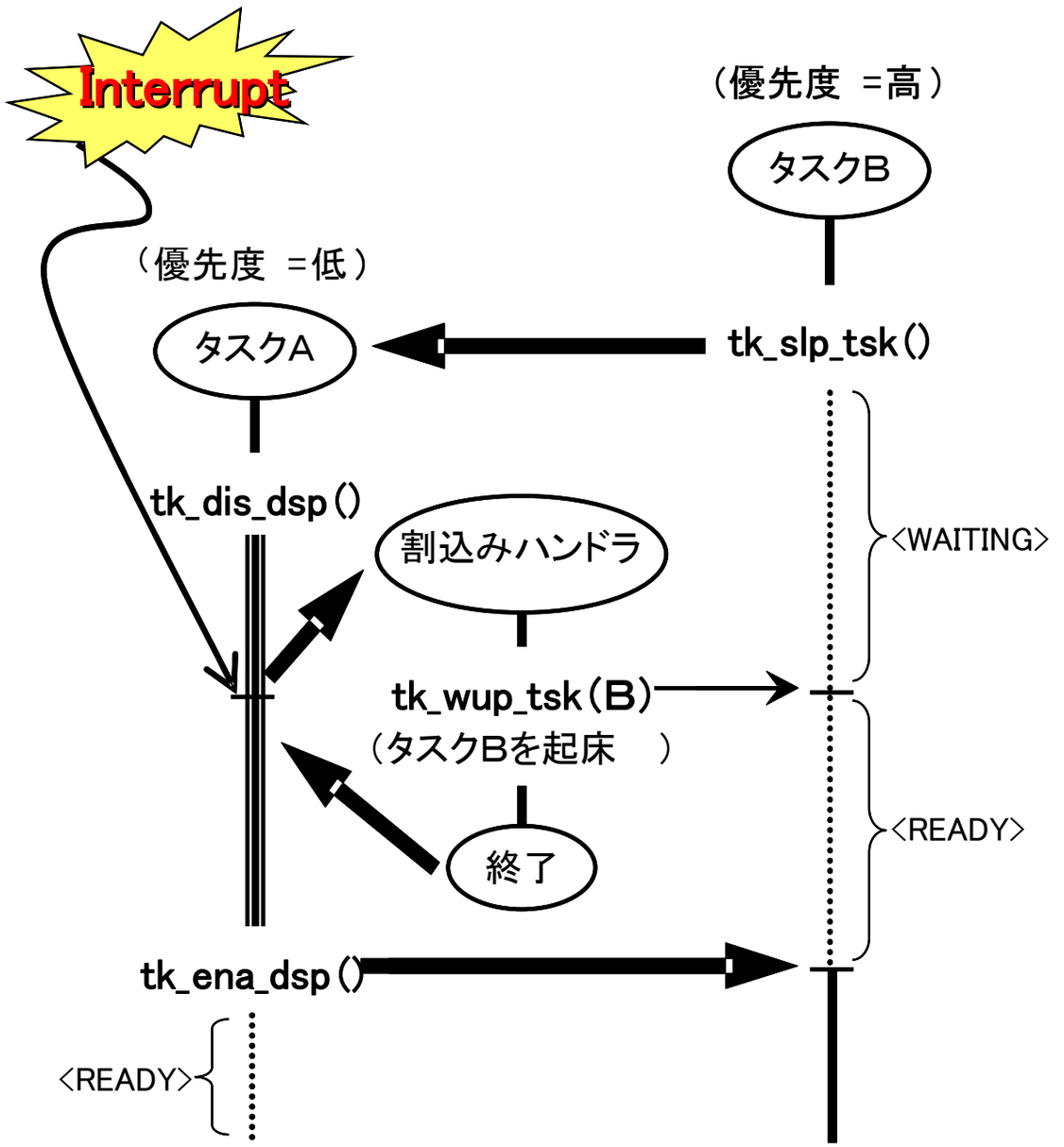
（解説）プロセッサが実行するタスクを切り替えることをディスパッチと呼ぶ。tk_dis_dsp()で自タスクをディスパッチ禁止すると、他のタスクに切り替わることはなくなるが、割込みハンドラは起動される。

ディスパッチ禁止によるタスク切替タイミング操作

通常の場合



ディスパッチ禁止状態の場合



tk_ena_dsp()によってタスク切替のタイミングを操作できる。

(G) サブシステム管理機能

サブシステム管理機能

T-Kernel上で動作するミドルウェア等を実装するために、「サブシステム」と呼ばれるユーザ定義の機能をカーネルに追加し、T-Kernel本体の機能を拡張するための機能

ユーザ定義のシステムコール(「拡張SVC」と呼ぶ)を実行するための拡張SVCハンドラのほか、例外発生時の処理を行うブレーク関数、デバイス等からのイベント発生時の処理を行うイベント処理関数、タスクのリソースグループ毎に起動時や終了時の処理を行うためのスタートアップ関数とクリーンアップ関数、およびリソース管理ブロックから構成

サブシステム管理機能

tk_def_ssy()	サブシステム定義
tk_sta_ssy()	スタートアップ関数呼出
tk_cln_ssy()	クリーンアップ関数呼出
tk_evt_ssy()	イベント処理関数呼出

4-2. T-Kernel/SM

T-Kernel/SMの機能

- システムメモリ管理機能
- アドレス空間管理機能
- デバイス管理機能
- 割込み管理機能
- I/Oポートアクセスサポート機能
- 省電力機能
- システム構成情報管理機能
- メモリキャッシュ制御機能
- 物理タイマ機能
- ユーティリティ機能

(A) システムメモリ管理機能

システムメモリ管理機能

T-Kernelが動的に割り当てる全てのメモリ(システムメモリ)を管理する機能。T-Kernel内部で使用しているメモリやタスクのスタック、メッセージバッファ、メモリプールなどもここから割り当てる。

システムメモリ割当て

tk_get_smb()	システムメモリの割当て
tk_rel_smb()	システムメモリの解放

システムメモリ管理機能

メモリ割当てライブラリ

Vmalloc()	非常駐メモリの割当て
Vfree()	非常駐メモリの解放
Kmalloc()	常駐メモリの割当て
Kfree()	常駐メモリの解放

(B) アドレス空間管理機能

アドレス空間管理機能

論理アドレス空間に対して各種の操作や管理を行うための機能

MMUやページテーブルを操作することによって実現

MMUを使用しないシステムであっても、アドレス空間管理機能のAPIは提供

ChkSpaceRW()	メモリ読込み書込みアクセス権の検査
LockSpace()	メモリ領域のロック
UnlockSpace()	メモリ領域のアンロック
CnvPhysicalAddr()	物理アドレスの取得
MapMemory()	メモリのマップ
UnmapMemory()	メモリのアンマップ

(C) デバイス管理機能

デバイス管理機能

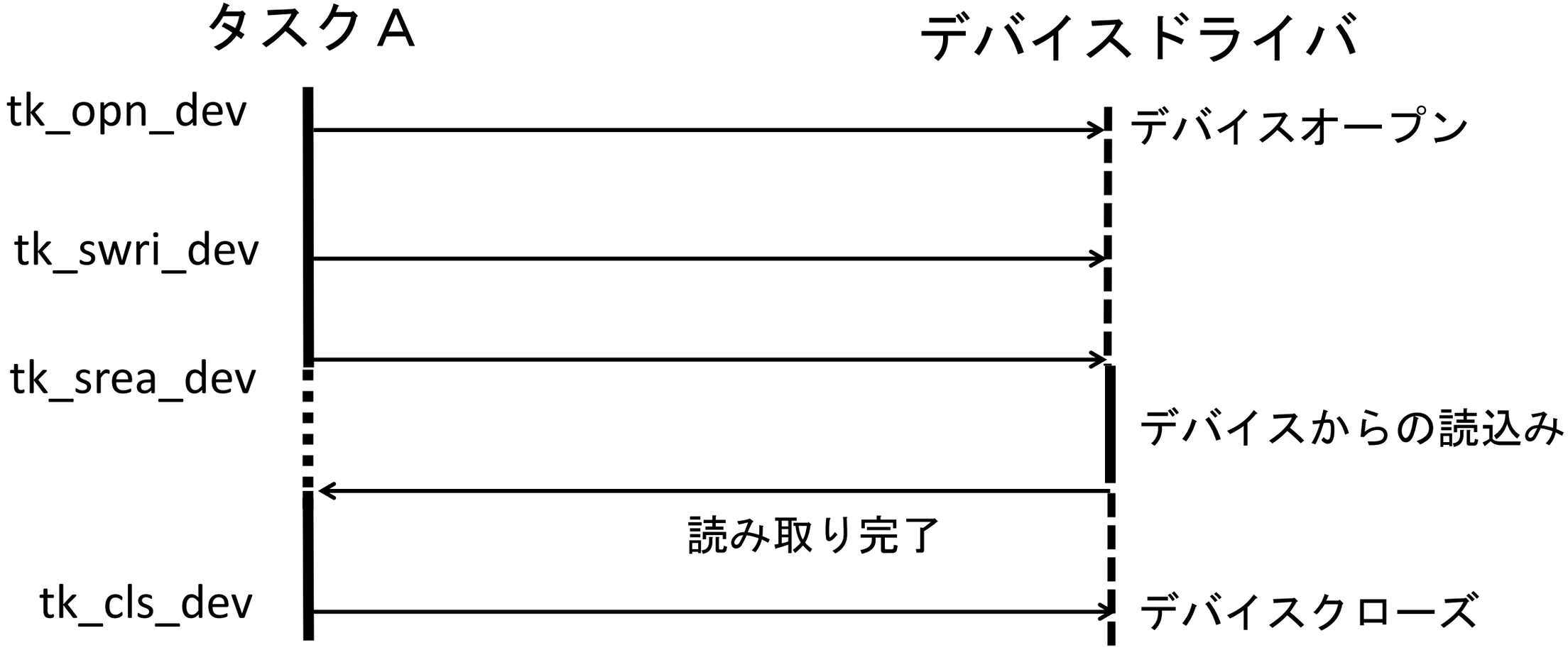
T-Kernel上で動作するデバイスドライバを管理するための機能

デバイス管理では、デバイスドライバのアプリケーションインタフェースとして、下記関数を提供する

tk_opn_dev()	デバイスのオープン
tk_cls_dev()	デバイスのクローズ
tk_rea_dev()	デバイスの読込み開始
tk_rea_dev_du()	デバイスの読込み開始(64ビットマイクロ秒単位)
tk_srea_dev()	デバイスの同期読込み
tk_srea_dev_d()	デバイスの同期読込み(64ビット)
tk_wri_dev()	デバイスの書込み開始
tk_wri_dev_du()	デバイスの書込み開始(64ビットマイクロ秒単位)
tk_swri_dev()	デバイスの同期書込み
tk_swri_dev_d()	デバイスの同期書込み(64ビット)
tk_def_dev()	デバイスの登録

デバイス制御の例

tk_opn_dev	デバイスのオープン
tk_cls_dev	デバイスのクローズ
tk_srea_dev	デバイスの同期読込み
tk_swri_dev	デバイスの同期書込み



(D) 割込み管理機能

割込み管理機能

外部割込みの禁止や許可、割込み禁止状態の取得、割込みコントローラの制御などを行うための機能

割込み管理機能は、ライブラリ関数またはC言語のマクロとして提供

CPU割込み制御

DI()	外部割込み禁止（C言語のマクロ）
EI()	外部割込み許可（C言語のマクロ）
isDI()	外部割込み禁止状態の取得（C言語のマクロ）

割込み管理機能

割込みコントローラ制御

DINTNO()	割込みベクタから割込みハンドラ番号へ変換
EnableInt()	割込み許可
DisableInt()	割込み禁止
ClearInt()	割込み発生のクリア
EndOfInt()	割込みコントローラにEOI発行
CheckInt()	割込み発生の検査
SetIntMode()	割込みモード設定

T-Kernel 2.0

! EOI(End Of Interrupt)

(E) I/Oポートアクセスサポート機能

I/Oポートアクセスサポート機能

入出力デバイスへのアクセスや操作をサポートするための機能

I/Oポートアクセス機能は、ライブラリ関数またはC言語のマクロで提供

out_b()	I/Oポート書込み(バイト)
out_h()	I/Oポート書込み(ハーフワード)
out_w()	I/Oポート書込み(ワード)
out_d()	I/Oポート書込み(ダブルワード)
in_b()	I/Oポート読込み(バイト)
in_h()	I/Oポート読込み(ハーフワード)
in_w()	I/Oポート読込み(ワード)
in_d()	I/Oポート読込み(ダブルワード)

(F) 省電力機能

省電力機能

システムの省電力を実現するための機能

T-Kernel/OSの中からコールバック型の関数として呼び出される

(G) システム構成情報管理機能

システム構成情報管理機能

システム構成に関する情報（最大タスク数）およびその他の任意の情報を保持・管理するための機能

tk_get_cfn()	システム構成情報から数値列取得
tk_get_cfs()	システム構成情報から文字列取得

メモリキャッシュ制御機能

キャッシュの制御やモード設定を行うための機能

SetCacheMode()	キャッシュモードの設定
ControlCache()	キャッシュの制御

物理タイマ機能

複数のハードウェアタイマが使えるシステムにおいて、タイマ割込み間隔(TTimPeriod)よりも細かい単位の時間経過を条件とした処理を行う場合に有効な機能

StartPhysicalTimer()	物理タイマの動作開始
StopPhysicalTimer()	物理タイマの動作停止
GetPhysicalTimerCount()	物理タイマのカウント値取得
DefinePhysicalTimerHandler()	物理タイマハンドラ定義
GetPhysicalTimerConfig()	物理タイマのコンフィグレーション情報取得

ユーティリティ機能

T-Kernel上のアプリケーション、ミドルウェア、デバイスドライバなどプログラム全般から利用される共通性の高い機能

ユーティリティ機能は、ライブラリ関数またはC言語のマクロで提供

オブジェクト名設定

SetOBJNAME()　オブジェクト名設定

ユーティリティ機能

高速ロック・マルチロックライブラリ

デバイスドライバやサブシステムの中において、複数タスク間の排他制御をより高速に行うためのライブラリ

CreateLock()	高速ロックの生成
DeleteLock()	高速ロックの削除
Lock()	高速ロックのロック操作
Unlock()	高速ロックのロック解除操作

ユーティリティ機能

高速ロック・マルチロックライブラリ

CreateMLock()	高速マルチロックの生成
DeleteMLock()	高速マルチロックの削除
MLock()	高速マルチロックのロック操作
MLockTmo()	高速マルチロックのロック操作 (タイムアウト指定付き)
MLockTmo_u()	高速マルチロックのロック操作 (タイムアウト指定付き、マイクロ秒単位)
MUnlock()	高速マルチロックのロック解除操作

4-3. T-Kernel/DS

T-Kernel/DSの機能

- カーネル内部状態取得機能
- 実行トレース機能

(A) カーネル内部状態取得機能

カーネル内部状態取得機能

デバッガがカーネルの内部状態を取得するための機能

td_lst_tsk()	タスクIDのリスト参照
td_rdy_que()	タスクの優先順位の参照
td_ref_tsk()	タスク状態参照
td_ref_tsk_u()	タスク状態参照(マイクロ秒単位)
td_inf_tsk()	タスク統計情報参照
td_inf_tsk_u()	タスク統計情報参照(マイクロ秒単位)
td_get_reg()	タスクレジスタの参照
td_ref_dsname()	DSオブジェクト名称の参照
td_set_dsname()	DSオブジェクト名称の設定

(B) 実行トレース機能

実行トレース機能

デバッガがプログラムの実行をトレースするための機能

td_hok_svc()	システムコール・拡張SVCのフックルーチン定義
td_hok_dsp()	タスクディスパッチのフックルーチン定義
td_hok_int()	割込みハンドラのフックルーチン定義

第六章
T-Kernelを動かしてみる

ワンストップサービス

- **T-Kernel 2.0** はオープンソース
  - **T-Engine**リファレンスボードで動作するソースを公開
  - 組込み向けに利用しやすい**T-License 2.0**
- **T-Kernel 1.0** と比べてソースの提供範囲を拡大、ワンパッケージ化
  - **T-Monitor**、一部のデバイスドライバ、開発環境、**PC**上のシミュレータも含めて一括公開
- **ucode**を用いたソースコードのトレーサビリティシステム
- **T-Kernel 2.0** は**2011**年**5**月**17**日から公開開始

提供されるソフトウェア

- T-Kernel 2.0
  - 動作対象機種の tef_em1d に合わせてARM11コア依存部を追加
- T-Monitor
  - ハードウェアの初期設定
  - T-Kernelのブート処理
  - 割込みや例外のハンドリング
  - ハードウェア階層の対話型デバッグ機能
    - メモリやレジスタの参照
- デバイスドライバ
  - 時計(RTC)
  - シリアルコンソール
  - タッチパネル
  - スクリーン(LCD)
  - システムディスク

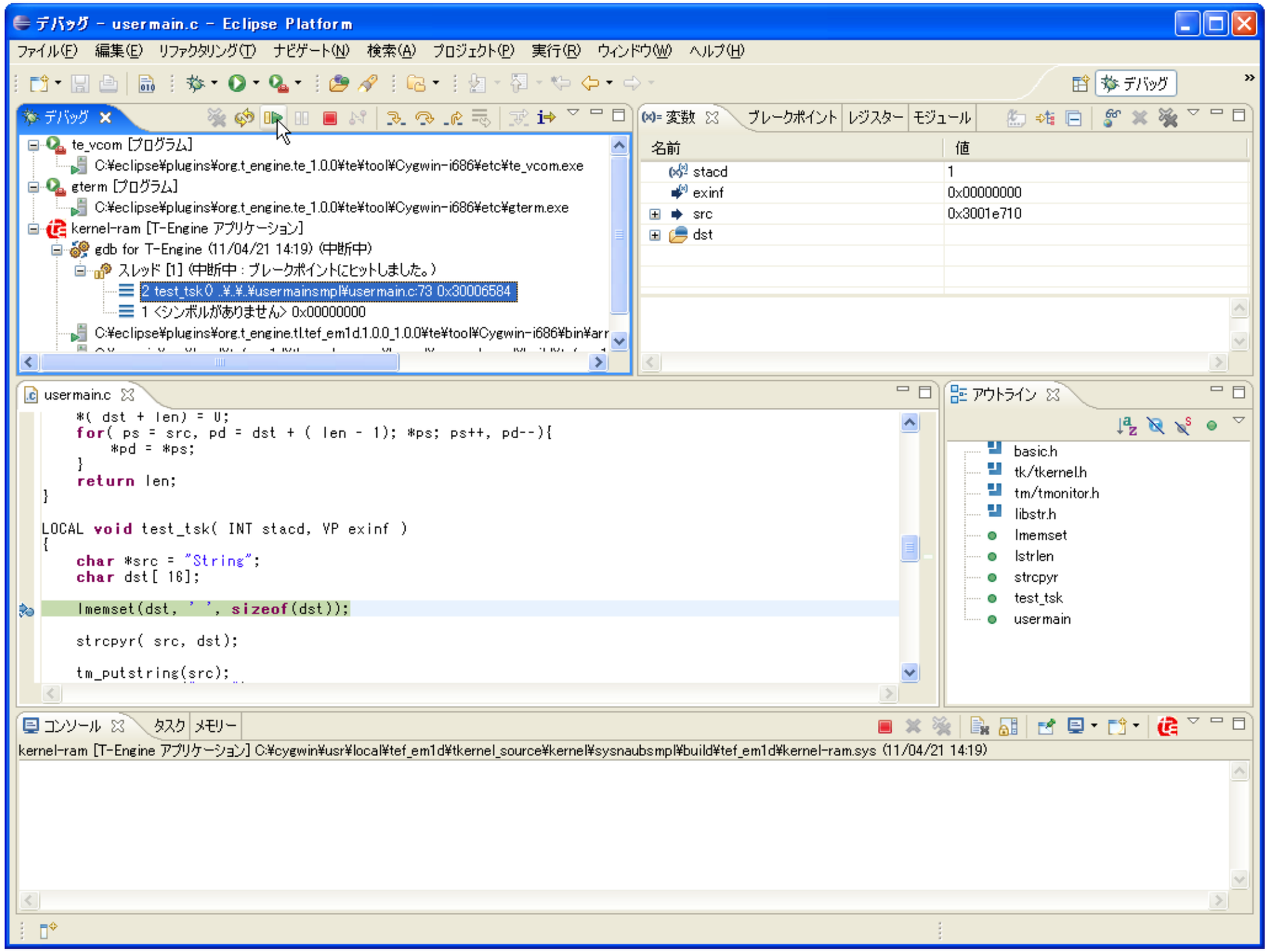
提供されるソフトウェア

- Eclipse開発環境
  - Windows PC上で動作
  - コンパイルやビルド
  - 実機へのプログラム転送と実行
  - デバッグ機能
    - ブレークポイントの設定
    - 変数値の参照や変更など

※ このほか、Linuxのコマンドベース(非GUI)による開発も可能

- QEMUによるエミュレータ
  - Windows PC上で動作
    - ハードウェア(実機)がなくても開発できる
  - CPUおよびボード搭載の各デバイスに対応
    - タイマ、microSDカード、I2C、UART(シリアル)、USB、RTC、LCD画面、タッチパネル、LANなど

Eclipse開発環境



- ソースパッケージの構成

```
tkernel_source
    |----config                設定情報
    |----drv                   デバイスドライバ
    |----include               インクルードファイル
    |----kernel                T-Kernel 2.0本体
    |----lib                   ライブラリ
    |----monitor               T-Monitor
```

- T-Kernel 2.0本体のソース

```
tkernel_source
  |-kernel
      |--sysdepend          ハードウェア依存部
      |    |--cpu
      |    |    |--em1d     CPU依存部
      |    |--device
      |         |--tef_em1d  デバイス依存部
      |--sysinit            初期化
      |--sysmain            システムメイン
      |--sysmgr             T-Kernel/SM
      |--tkernel            T-Kernel/OS,/DS
      |    |--build         ビルド(make)用
      |    |    |--tef_em1d tef_em1dでのビルド(make)用
      |    |--src           ソース
      |--usermain           アプリ利用時のメイン
      |--usermain_drv       ドライバ利用時のメイン
```

- T-Monitorのソース

```
tkernel_source
  |--monitor
      |--cmdsvc              コマンド,SVC処理
      |   |--src
      |       |--armv6               ARMv6依存部
      |--driver              T-Monitor用ドライバ
      |   |--flash           FlashROM
      |   |--memdisk         メモリディスク
      |   |--sio             シリアルI/O
      |--hwdepend            ハードウェア依存部
      |   |--tef_em1d              tef_em1d依存部
      |--tmmain              T-Monitorメイン
          |--build           ビルド(make)用
          |--src             ソース
```

- デバイスドライバのソース

```
tkernel_source
  |--drv
    |--tef_em1d     機種名を表わすディレクトリ
      |--clk        時計(RTC)
      |--console    シリアルコンソール
      |--kbpd       KB/PD(タッチパネル)
      |--lowkbpd    KB/PD実IO
      |--screen     スクリーン(LCD)
      |--sysdsk     システムディスク
        |--build    ビルド(make)用
        |--src      ソース
```

T-Kernelのアプリケーションを動かしてみる

T-Kernelのアプリケーションを動かしてみる

- usermain() から開始
  - kernel/usermain/usermain.c
  - usermain()　　タスク本体
    - 初期タスクから呼び出される関数
    - 他のタスクを生成・起動してアプリケーションにする。

usermain() のコード

- オリジナルの usermain()
  - メッセージを表示して、
  - コンソール(SIO)でキーが入力されたら
  - 電源をOFFにする。

```
EXPORT   INT     usermain( void )
{
        tm_putstring((UB*)"Push any key to shutdown the T-Kernel.¥n");
        tm_getchar(-1);

        return 0;
}
```

改造した usermain() のサンプル

- usermain()からタスクを起動

```
EXPORT INT  usermain( void )
{
	T_CTSK	ctsk;
	ID	tskid;
	ER	ercd;

	tm_putstring((UB*)"Start User Application.¥n");

	memset(&ctsk, 0, sizeof(T_CTSK));
	ctsk.tskatr  = TA_HLNG | TA_RNG0;
	ctsk.task    = task1;
	ctsk.itskpri = 1;
	ctsk.stksz   = 1024;

	tskid = tk_cre_tsk(&ctsk);
	if ( tskid < 0 ) {
		return 0;
	}
	ercd = tk_sta_tsk(tskid, 0);
	if ( ercd < 0 ) {
		return 0;
	}
	tk_slp_tsk(TMO_FEVR);
	return 0;
}
```

- タスク task1() の生成
- タスク task1() の起動
  - 正常終了時はtk_slp_tsk()で無限待ち
  - エラー発生時はusermain()を終了

```
void task1(INT stacd, VP exinf)
		{
		while( 1 ){
		  /* do! */
		  }

		tk_exd_tsk();
		}
```

タスクに関する注意事項

- タスクの終了時には、必ず以下のいずれかのSVCを呼び出すこと
  - tk_ext_tsk()　自タスク終了
  - tk_exd_tsk()　自タスクの終了と削除
    - force_dispatch()を呼出して、他タスクに処理を移行する。

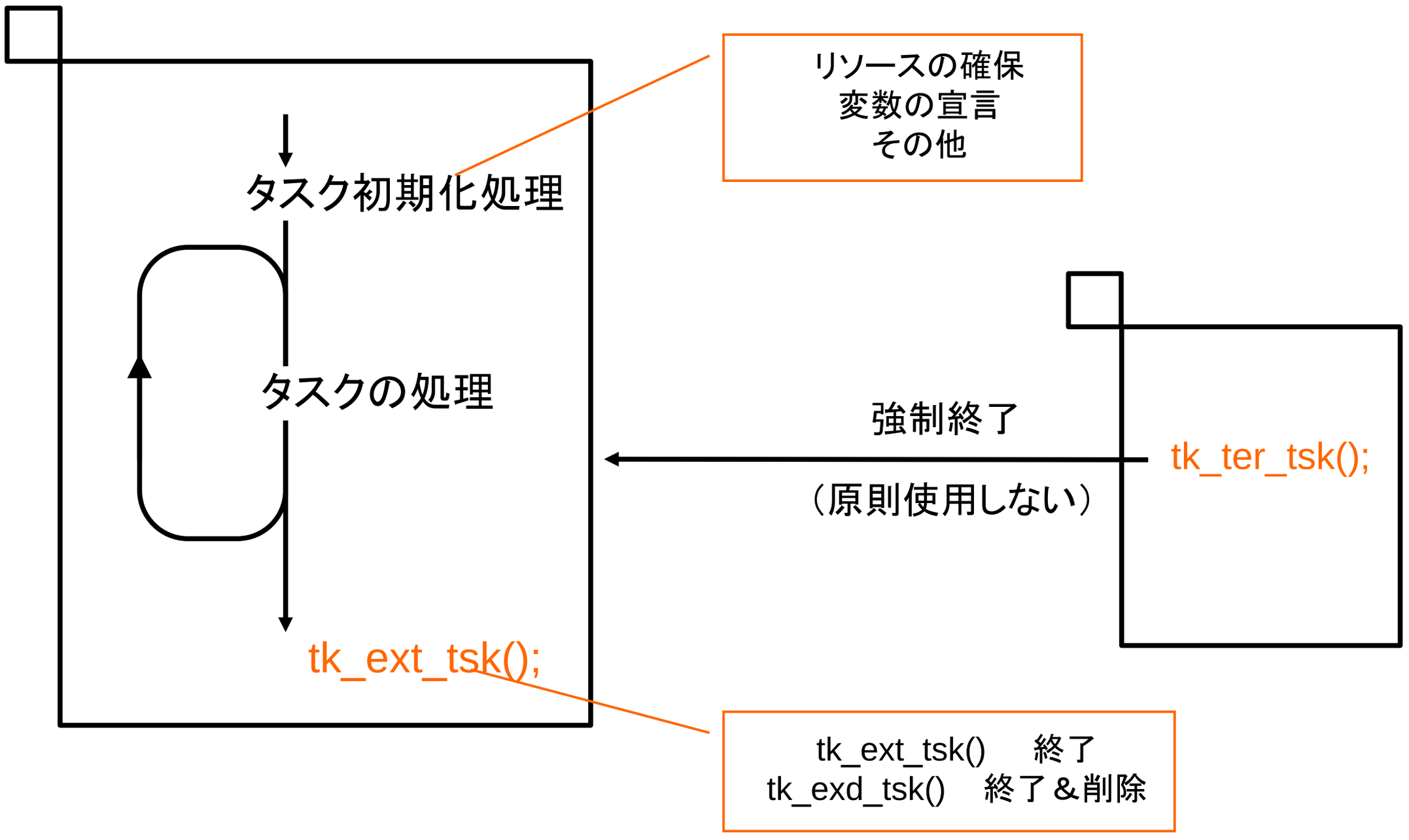
- 上記を呼び出さない場合の動作は不定
  - タスクも関数の形で記述しているので…
  - 戻り先が不定な状態で関数を終了してしまう。

タスクに関する注意事項

- 他のタスクから終了する。
  - tk_ter_tsk()　タスク強制終了

  - タスクの状態が不明なまま強制的に終了させるので
    推奨されない。
    - 「他タスクの強制終了は、デバッガなどのOSに密接に関連した
      ごく一部でのみ使用することを原則とする。」(T-Kernel仕様書)

タスクの基本構造


リソースの確保
変数の宣言
その他
タスク初期化処理
タスクの処理
tk_ext_tsk();
強制終了
（原則使用しない）
tk_ter_tsk();
tk_ext_tsk()　終了
tk_exd_tsk()　終了＆削除

その他のタスク終了方法

- 待ち状態を禁止 or 解除
  - 他タスクの待ち状態解除　　tk_rel_wai()
  - タスク待ち状態の禁止　　tk_dis_wai()

  - 自タスクの終了/終了と削除　　tk_exd_tsk()

- タスク例外　　k_xxx_tex()
  - タスク例外を登録しておき、
    タスク例外コード=0でタスク例外を発生させる。

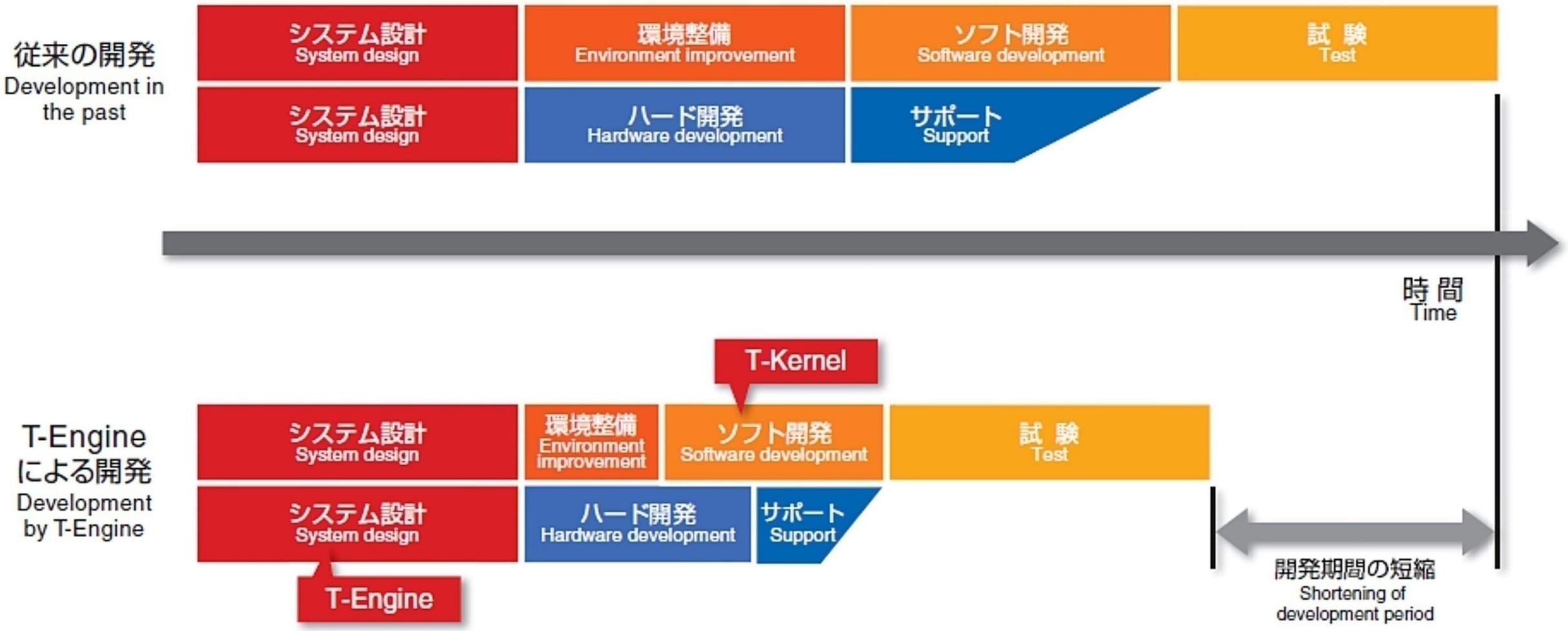
第七章
T-Kernelを用いた製品開発

組込み機器の製品開発手順

組込み機器の製品開発手順

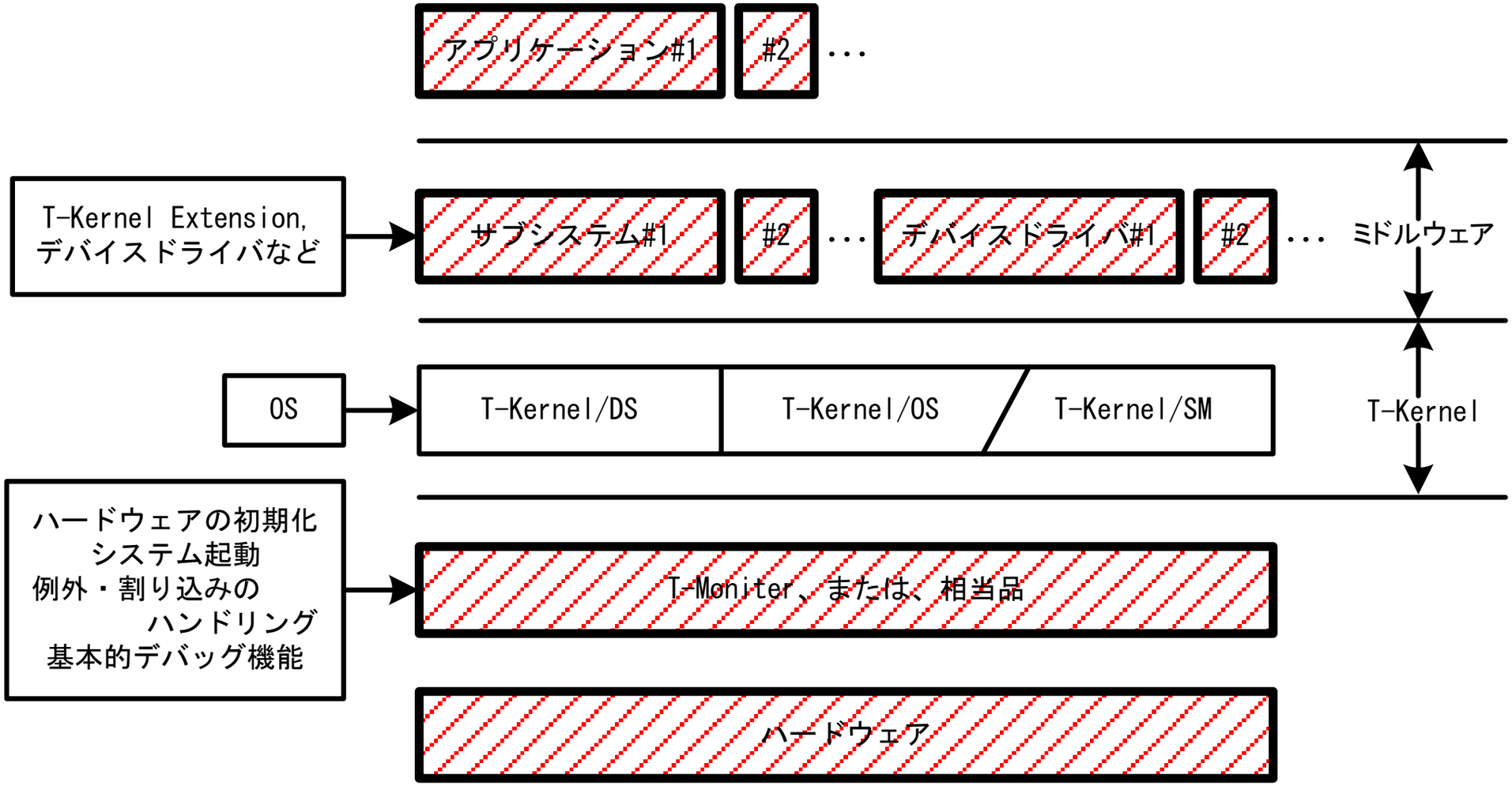
- システム全体の設計
  - 機能、性能の決定
  - 開発(デバッグ)方法の設計
  - コストなどの営業的な側面とのすり合わせ
- ハードウェアの設計
  - CPU、周辺装置などのコアとなるパーツの選択
  - 選択したパーツを組み合わせて効率の良いハードウェアを設計
  - 筐体などのデザイン
  - コストを最小限に抑えた状態で、機能や性能を極大化させる
- ソフトウェアの設計
  - OS、開発環境、デバッガなどの要素技術の選択
  - モニタやデバイスドライバなどの基本機能の設計
  - ミドルウェアの選択(購入、流用)、または、設計(自社開発)
  - アプリケーションの設計

T-Engine, T-Kernelを利用して開発期間を短縮



開発するソフトウェア

- OS以外は新規開発が必要
  - 他のベンダーから購入する場合、既存製品から移植する場合、開発や移植の作業を他のベンダーに委託する場合もある。



T-Engine, T-Kernelを利用することで...

- T-Engine , T-Kernel は標準開発プラットフォーム

- 比較的大規模なシステムを効率的に開発可能
- T-Kernel上で動作するアプリケーションやデバイスドライバは、既存のT-Kernel 応用製品の上で先行開発を進めることが可能

新しいボードへの移植、
新機種の追加

新しいボードへの移植、新機種の追加

- 機種依存部を追加
  - T-Kernel 2.0ソースコードの tef_em1d あるいは [TARGET] となっていた箇所の並びに追加
  - ＣＰＵ依存部分のプログラム開発
    - 同一、同系列、類似のCPUのファイルをコピーして改変
  - ボード依存部分のプログラム開発
    - 類似のボードやデバイスのファイルをコピーして改変
  - T-Kernel 1.0のソースの機種依存部も参考に
    - T-Engineリファレンスボード 例)　tef_em1d
    - 標準T-Engine　　std_xxx　　例)　std_sh7760
    - μT-Engine　　mic_xxx　　例)　mic_vr4131
    - Appliance　　app_xxx　　例)　app_mb91403

付録A
T-Kernel/OSのシステムコール

T-Kernel/OSの機能

[1] タスク管理機能
[2] タスク付属同期機能
[3] タスク例外処理機能
[4] 同期・通信機能
[5] 拡張同期・通信機能
[6] メモリプール管理機能
[7] 時間管理機能
[8] 割込み管理機能
[9] システム状態管理機能
[10] サブシステム管理機能

[1] タスク管理機能

- tk_cre_tsk　　タスク生成
- tk_del_tsk　　タスク削除
- tk_sta_tsk　　タスク起動
- tk_ext_tsk　　自タスク終了
- tk_exd_tsk　　自タスクの終了と削除
- tk_ter_tsk　　他タスク強制終了
- tk_chg_pri　　タスク優先度変更
- tk_chg_slt　　タスクスライスタイム変更
- tk_chg_slt_u　タスクスライスタイム変更(マイクロ秒単位)

[1] タスク管理機能

- tk_get_tsp　　タスク固有空間の参照
- tk_set_tsp　　タスク固有空間の設定
- tk_get_rid　　タスクの所属リソースグループの参照
- tk_set_rid　　タスクの所属リソースグループの設定
- tk_get_reg　　タスクレジスタの取得
- tk_set_reg　　タスクレジスタの設定
- tk_get_cpr　　コプロセッサのレジスタの取得
- tk_set_cpr　　コプロセッサのレジスタの設定
- tk_inf_tsk　　タスク統計情報参照
- tk_inf_tsk_u　　タスク統計情報参照(マイクロ秒単位)
- tk_ref_tsk　　タスク状態参照
- tk_ref_tsk_u　　タスク状態参照(マイクロ秒単位)

[2] タスク付属同期機能

- tk_slp_tsk　　自タスクを起床待ち状態へ移行
- tk_slp_tsk_u　自タスクを起床待ち状態へ移行(マイクロ秒単位)
- tk_wup_tsk　　他タスクの起床
- tk_can_wup　　タスクの起床要求を無効化
- tk_rel_wai　　他タスクの待ち状態解除
- tk_sus_tsk　　他タスクを強制待ち状態へ移行
- tk_rsm_tsk　　強制待ち状態のタスクを再開
- tk_frsm_tsk　　強制待ち状態のタスクを強制再開
- tk_dly_tsk　　タスク遅延
- tk_dly_tsk_u　タスク遅延(マイクロ秒単位)

[2] タスク付属同期機能

- tk_sig_tev　　タスクイベントの送信
- tk_wai_tev　　タスクイベント待ち
- tk_wai_tev_u　タスクイベント待ち(マイクロ秒単位)
- tk_dis_wai　　タスク待ち状態の禁止
- tk_ena_wai　　タスク待ち禁止の解除

[3] タスク例外処理機能

- tk_def_tex　　タスク例外ハンドラの定義
- tk_ena_tex　　タスク例外の許可
- tk_dis_tex　　タスク例外の禁止
- tk_ras_tex　　タスク例外を発生
- tk_end_tex　　タスク例外ハンドラの終了
- tk_ref_tex　　タスク例外の状態参照

[4] 同期・通信機能（セマフォ）

- tk_cre_sem　　セマフォ生成
- tk_del_sem　　セマフォ削除
- tk_sig_sem　　セマフォ資源返却
- tk_wai_sem　　セマフォ資源獲得
- tk_wai_sem_u　セマフォ資源獲得(マイクロ秒単位)
- tk_ref_sem　　セマフォ状態参照

[4] 同期・通信機能（イベントフラグ）

- tk_cre_flg　　イベントフラグ生成
- tk_del_flg　　イベントフラグ削除
- tk_set_flg　　イベントフラグのセット
- tk_clr_flg　　イベントフラグのクリア
- tk_wai_flg　　イベントフラグ待ち
- tk_wai_flg_u　イベントフラグ待ち(マイクロ秒単位)
- tk_ref_flg　　イベントフラグ状態参照

[4] 同期・通信機能（メールボックス）

- tk_cre_mbx　　メールボックス生成
- tk_del_mbx　　メールボックス削除
- tk_snd_mbx　　メールボックスへ送信
- tk_rcv_mbx　　メールボックスから受信
- tk_rcv_mbx_u　メールボックスから受信(マイクロ秒単位)
- tk_ref_mbx　　メールボックス状態参照

[5] 拡張同期・通信機能（ミューテックス）

- tk_cre_mtx　　ミューテックス生成
- tk_del_mtx　　ミューテックス削除
- tk_loc_mtx　　ミューテックスのロック
- tk_loc_mtx_u　ミューテックスのロック(マイクロ秒単位)
- tk_unl_mtx　　ミューテックスのアンロック
- tk_ref_mtx　　ミューテックス状態参照

[5] 拡張同期・通信機能（メッセージバッファ）

- tk_cre_mbf　　メッセージバッファ生成
- tk_del_mbf　　メッセージバッファ削除
- tk_snd_mbf　　メッセージバッファへ送信
- tk_snd_mbf_u　メッセージバッファへ送信(マイクロ秒単位)
- tk_rcv_mbf　　メッセージバッファから受信
- tk_rcv_mbf_u　メッセージバッファから受信(マイクロ秒単位)
- tk_ref_mbf　　メッセージバッファ状態参照

[5] 拡張同期・通信機能（ランデブ）

- tk_cre_por　　ランデブポート生成
- tk_del_por　　ランデブポート削除
- tk_cal_por　　ランデブポートに対するランデブの呼出
- tk_cal_por_u　ランデブポートに対するランデブの呼出
  (マイクロ秒単位)
- tk_acp_por　　ランデブポートに対するランデブ受付
- tk_acp_por_u　ランデブポートに対するランデブ受付
  (マイクロ秒単位)
- tk_fwd_por　　ランデブポートに対するランデブ回送
- tk_rpl_rdv　　ランデブ返答
- tk_ref_por　　ランデブポート状態参照

[6] メモリプール管理機能（固定長メモリプール）

- tk_cre_mpf　　固定長メモリプール生成
- tk_del_mpf　　固定長メモリプール削除
- tk_get_mpf　　固定長メモリブロック獲得
- tk_get_mpf_u　固定長メモリブロック獲得(マイクロ秒単位)
- tk_rel_mpf　　固定長メモリブロック返却
- tk_ref_mpf　　固定長メモリプール状態参照

[6] メモリプール管理機能（可変長メモリプール）

- tk_cre_mpl　　可変長メモリプール生成
- tk_del_mpl　　可変長メモリプール削除
- tk_get_mpl　　可変長メモリブロック獲得
- tk_get_mpl_u　可変長メモリブロック獲得(マイクロ秒単位)
- tk_rel_mpl　　可変長メモリブロック返却
- tk_ref_mpl　　可変長メモリプール状態参照

[7] 時間管理機能（システム時刻管理）

- tk_set_tim　　システム時刻設定
- tk_set_tim_u　システム時刻設定(マイクロ秒単位)
- tk_get_tim　　システム時刻参照
- tk_get_tim_u　システム時刻参照(マイクロ秒単位)
- tk_get_otm　　システム稼働時間参照
- tk_get_otm_u　システム稼働時間参照(マイクロ秒単位)

[7] 時間管理機能（周期ハンドラ）

- tk_cre_cyc　　周期ハンドラの生成
- tk_cre_cyc_u　周期ハンドラの生成(マイクロ秒単位)
- tk_del_cyc　　周期ハンドラの削除
- tk_sta_cyc　　周期ハンドラの動作開始
- tk_stp_cyc　　周期ハンドラの動作停止
- tk_ref_cyc　　周期ハンドラ状態参照
- tk_ref_cyc_u　周期ハンドラ状態参照(マイクロ秒単位)

[7] 時間管理機能（アラームハンドラ）

- tk_cre_alm　　アラームハンドラの生成
- tk_del_alm　　アラームハンドラの削除
- tk_sta_alm　　アラームハンドラの動作開始
- tk_sta_alm_u　アラームハンドラの動作開始(マイクロ秒単位)
- tk_stp_alm　　アラームハンドラの動作停止
- tk_ref_alm　　アラームハンドラ状態参照
- tk_ref_alm_u　アラームハンドラ状態参照(マイクロ秒単位)

[8] 割込み管理機能

- tk_def_int　　割込みハンドラ定義
- tk_ret_int　　割込みハンドラから復帰

[9] システム状態管理機能

- tk_rot_rdq　　タスクの優先順位の回転
- tk_get_tid　　実行状態タスクのタスクID参照
- tk_dis_dsp　　ディスパッチ禁止
- tk_ena_dsp　　ディスパッチ許可
- tk_ref_sys　　システム状態参照
- tk_set_pow　　省電力モード設定
- tk_ref_ver　　バージョン参照

[10] サブシステム管理機能

- tk_def_ssy　　サブシステム定義
- tk_sta_ssy　　スタートアップ関数呼出
- tk_cln_ssy　　クリーンアップ関数呼出
- tk_evt_ssy　　イベント処理関数呼出
- tk_ref_ssy　　サブシステム定義情報の参照
- tk_cre_res　　リソースグループの生成
- tk_del_res　　リソースグループの削除
- tk_get_res　　リソース管理ブロックの取得

付録B

T-Kernel/SMの拡張SVC・ライブラリ

T-Kernel/SMの機能

[1] システムメモリ管理機能
[2] アドレス空間管理機能
[3] デバイス管理機能
[4] 割込み管理機能
[5] I/Oポートアクセスサポート機能
[6] 省電力機能
[7] システム構成情報管理機能
[8] メモリキャッシュ制御機能
[9] 物理タイマ機能
[10] ユーティリティ機能

[1] システムメモリ管理機能
（システムメモリ割当て）

- tk_get_smb　　システムメモリの割当て
- tk_rel_smb　　システムメモリの解放
- tk_ref_smb　　システムメモリ情報取得

[1] システムメモリ管理機能
（メモリ割当てライブラリ）

- Vmalloc　　非常駐メモリの割当て
- Vcalloc　　非常駐メモリの割当て
- Vrealloc　　非常駐メモリの再割当て
- Vfree　　非常駐メモリの解放
- Kmalloc　　常駐メモリの割当て
- Kcalloc　　常駐メモリの割当て
- Krealloc　　常駐メモリの再割当て
- Kfree　　常駐メモリの解放

[2] アドレス空間管理機能
（アドレス空間設定）

- SetTaskSpace　タスクのアドレス空間設定

[2] アドレス空間管理機能
（アドレス空間チェック）

- ChkSpaceR　　メモリ読込みアクセス権の検査
- ChkSpaceRW　　メモリ読込み書込みアクセス権の検査
- ChkSpaceRE　　メモリ読込みアクセス権および実行権の検査
- ChkSpaceBstrR　　文字列読込みアクセス権の検査
- ChkSpaceBstrRW　　文字列読込み書込みアクセス権の検査
- ChkSpaceTstrR　　TRONコード文字列読込みアクセス権の検査
- ChkSpaceTstrRW　　TRONコード文字列読込み書込みアクセス権の検査

[2] アドレス空間管理機能
（論理アドレス空間管理）

- LockSpace　　メモリ領域のロック
- UnlockSpace　　メモリ領域のアンロック
- CnvPhysicalAddr　　物理アドレスの取得
- MapMemory　　メモリのマップ
- UnmapMemory　　メモリのアンマップ
- GetSpaceInfo　　アドレス空間の各種情報の取得
- SetMemoryAccess　　メモリアクセス権の設定

[3] デバイス管理機能
（デバイスの入出力操作）

- tk_opn_dev　　デバイスのオープン
- tk_cls_dev　　デバイスのクローズ
- tk_rea_dev　　デバイスの読込み開始
- tk_rea_dev_du　　デバイスの読込み開始
  (64ビットマイクロ秒単位)
- tk_srea_dev　　デバイスの同期読込み
- tk_srea_dev_d　　デバイスの同期読込み(64ビット)
- tk_wri_dev　　デバイスの書込み開始
- tk_wri_dev_du　　デバイスの書込み開始
  (64ビットマイクロ秒単位)
- tk_swri_dev　　デバイスの同期書込み
- tk_swri_dev_d　　デバイスの同期書込み(64ビット)

[3] デバイス管理機能（デバイスの入出力操作）

- tk_wai_dev　　デバイスの要求完了待ち
- tk_wai_dev_u　　デバイスの要求完了待ち(マイクロ秒単位)
- tk_sus_dev　　デバイスのサスペンド
- tk_get_dev　　デバイスのデバイス名取得
- tk_ref_dev　　デバイスのデバイス情報取得
- tk_oref_dev　　デバイスのデバイス情報取得
- tk_lst_dev　　登録済みデバイス一覧の取得
- tk_evt_dev　　デバイスにドライバ要求イベントを送信

[3] デバイス管理機能
（デバイスドライバの登録）

- tk_def_dev　　デバイスの登録
- tk_ref_idv　　デバイス初期情報の取得

[3] デバイス管理機能
（デバイスドライバインタフェース）

- openfn　　オープン関数
- closefn　　クローズ関数
- execfn　　処理開始関数
- waitfn　　完了待ち関数
- abortfn　　中止処理関数
- eventfn　　イベント関数

[4] 割込み管理機能（CPU割込み制御）

- DI　　外部割込み禁止
- EI　　外部割込み許可
- isDI　外部割込み禁止状態の取得

[4] 割込み管理機能（割込みコントローラ制御）

- DINTNO　　割込みベクタから割込みハンドラ番号へ変換
- EnableInt　　割込み許可
- DisableInt　　割込み禁止
- ClearInt　　割込み発生のクリア
- EndOfInt　　割込みコントローラにEOI発行
- CheckInt　　割込み発生の検査
- SetIntMode　　割込みモード設定

[5] I/Oポートアクセスサポート機能（I/Oポートアクセス）

- out_b　　I/Oポート書込み(バイト)
- out_h　　I/Oポート書込み(ハーフワード)
- out_w　　I/Oポート書込み(ワード)
- out_d　　I/Oポート書込み(ダブルワード)
- in_b　　I/Oポート読込み(バイト)
- in_h　　I/Oポート読込み(ハーフワード)
- in_w　　I/Oポート読込み(ワード)
- in_d　　I/Oポート読込み(ダブルワード)

[5] I/0ポートアクセスサポート機能（微小待ち）

- WaitUsec　　微小待ち(マイクロ秒)
- WaitNsec　　微小待ち(ナノ秒)

[6] 省電力機能

- low_pow　　システムを低消費電力モードに移行
- off_pow　　システムをサスペンド状態に移行

[7] システム構成情報管理機能（システム構成情報の取得）

- tk_get_cfn　　システム構成情報から数値列取得
- tk_get_cfs　　システム構成情報から文字列取得

[8] メモリキャッシュ制御機能

- SetCacheMode　　キャッシュモードの設定
- ControlCache　　キャッシュの制御

[9] 物理タイマ機能

- StartPhysicalTimer　物理タイマの動作開始
- StopPhysicalTimer　物理タイマの動作停止
- GetPhysicalTimerCount　物理タイマのカウント値取得
- DefinePhysicalTimerHandler　物理タイマハンドラ定義
- GetPhysicalTimerConfig　物理タイマのコンフィグレーション情報取得

[10] ユーティリティ機能（オブジェクト名設定）

- SetOBJNAME　オブジェクト名設定

[10] ユーティリティ機能（高速ロック・マルチロックライブラリ）

- CreateLock　高速ロックの生成
- DeleteLock　高速ロックの削除
- Lock　高速ロックのロック操作
- Unlock　高速ロックのロック解除操作
- CreateMLock　高速マルチロックの生成
- DeleteMLock　高速マルチロックの削除
- MLock　高速マルチロックのロック操作
- MLockTmo　高速マルチロックのロック操作(タイムアウト指定付き)
- MLockTmo_u　高速マルチロックのロック操作(タイムアウト指定付き、マイクロ秒単位)
- MUnlock　高速マルチロックのロック解除操作

付録C
T-Kernel/DSのシステムコール

T-Kernel/DSの機能

[1] カーネル内部状態取得機能

[2] 実行トレース機能

[1] カーネル内部状態取得機能

- td_lst_tsk　　タスクIDのリスト参照
- td_lst_sem　　セマフォIDのリスト参照
- td_lst_flg　　イベントフラグIDのリスト参照
- td_lst_mbx　　メールボックスIDのリスト参照
- td_lst_mtx　　ミューテックスIDのリスト参照
- td_lst_mbf　　メッセージバッファIDのリスト参照
- td_lst_por　　ランデブポートIDのリスト参照
- td_lst_mpf　　固定長メモリプールIDのリスト参照
- td_lst_mpl　　可変長メモリプールIDのリスト参照
- td_lst_cyc　　周期ハンドラIDのリスト参照
- td_lst_alm　　アラームハンドラIDのリスト参照
- td_lst_ssy　　サブシステムIDのリスト参照

[1] カーネル内部状態取得機能

- td_rdy_que　　タスクの優先順位の参照
- td_sem_que　セマフォの待ち行列の参照
- td_flg_que　　イベントフラグの待ち行列の参照
- td_mbx_que　メールボックスの待ち行列の参照
- td_mtx_que　ミューテックスの待ち行列の参照
- td_smbf_que　メッセージバッファの送信待ち行列の参照
- td_rmbf_que　メッセージバッファの受信待ち行列の参照
- td_cal_que　　ランデブ呼出待ち行列の参照
- td_acp_que　　ランデブ受付待ち行列の参照
- td_mpf_que　固定長メモリプールの待ち行列の参照
- td_mpl_que　可変長メモリプールの待ち行列の参照

[1] カーネル内部状態取得機能

- td_ref_tsk　　タスク状態参照
- td_ref_tsk_u　　タスク状態参照(マイクロ秒単位)
- td_ref_tex　　タスク例外の状態参照
- td_ref_sem　　セマフォ状態参照
- td_ref_flg　　イベントフラグ状態参照
- td_ref_mbx　　メールボックス状態参照
- td_ref_mtx　　ミューテックス状態参照
- td_ref_mbf　　メッセージバッファ状態参照
- td_ref_por　　ランデブポート状態参照
- td_ref_mpf　　固定長メモリプール状態参照
- td_ref_mpl　　可変長メモリプール状態参照

[1] カーネル内部状態取得機能

- td_ref_cyc　　周期ハンドラ状態参照
- td_ref_cyc_u　周期ハンドラ状態参照(マイクロ秒単位)
- td_ref_alm　　アラームハンドラ状態参照
- td_ref_alm_u　アラームハンドラ状態参照(マイクロ秒単位)
- td_ref_sys　　システム状態参照
- td_ref_ssy　　サブシステム定義情報の参照
- td_inf_tsk　　タスク統計情報参照
- td_inf_tsk_u　タスク統計情報参照(マイクロ秒単位)
- td_get_reg　　タスクレジスタの参照
- td_set_reg　　タスクレジスタの設定
- td_get_tim　　システム時刻参照
- td_get_tim_u　システム時刻参照(マイクロ秒単位)

[1] カーネル内部状態取得機能

- td_get_otm　　システム稼働時間参照
- td_get_otm_u　システム稼働時間参照(マイクロ秒単位)
- td_ref_dsname　　DSオブジェクト名称の参照
- td_set_dsname　　DSオブジェクト名称の設定

[2] 実行トレース機能

- td_hok_svc　　システムコール・拡張SVCのフックルーチン定義
- td_hok_dsp　　タスクディスパッチのフックルーチン定義
- td_hok_int　　割込みハンドラのフックルーチン定義

付録D

μITRON3.0/μITRON4.0/ T-Kernelの比較

参考:各仕様の比較

- 本資料は、μITRON3.0/μITRON4.0/T-Kernelの各仕様のうち、代表的な機能とAPIの違いについて比較したものである
  - T-Kernel は 1.0 を対象とし、T-Kernel 2.0 の追加機能は記載していない。
- 各機能の分類等については、μITRON4.0仕様に基づいている
- 出典
  - 文書名：『μITRON仕様とT-Kernel仕様の違いについて』 第一版
  - 著者名：エルミック・ウェスコム株式会社(*1)を基に改訂

    (*1) 現・図研エルミック株式会社

用語

μITRON3.0仕様	μITRON4.0仕様	T-Kernel
仕様の準拠レベル レベルR (Required) レベルS (Standard) レベルE (Extended)	仕様の準拠レベル ベーシックプロファイル 自動車制御用プロファイル スタンダードプロファイル	
システムコール	サービスコール	システムコール
「タスク」を「タスク部」 「過渡的な状態」、「タスク独立部」、「準タスク部」を合わせて「非タスク部」	タスクのコンテキストをタスクコンテキスト、それ以外を非タスクコンテキスト 仕様上は過渡的な状態という用語は用いていない 準タスク部の概念は定義していない	「タスク」を「タスク部」 「過渡的な状態」、「タスク独立部」、「準タスク部」を合わせて「非タスク部」
システムクロック	システム時刻	システム時刻
周期起動ハンドラ	周期ハンドラ	周期ハンドラ
周期起動ハンドラ/アラームハンドラを総称して、タイマハンドラと呼ぶ	周期ハンドラ/アラームハンドラ/オーバーランハンドラを総称して、タイムイベントハンドラと呼ぶ	周期ハンドラ/アラームハンドラを総称して、タイムイベントハンドラと呼ぶ
メイルボックス	メールボックス	メールボックス

仕様

μITRON3.0仕様	μITRON4.0仕様	T-Kernel
オブジェクトの生成はシステムコールで要求	オブジェクトの生成は静的APIで記述する(スタンダードプロファイル) サービスコールで生成することも可能	オブジェクトの生成はシステムコールで要求
	静的APIの規定 コンフィギュレータに関する規定	
オブジェクトのID番号は利用者が指定する	オブジェクトのID番号はコンフィギュレータによる自動割付、もしくはサービスコールにより利用者が指定するか自動割付	オブジェクトのID番号は自動割付
カーネルが管理するオブジェクトには拡張情報を設定する	拡張情報を設定するのは、タスク/周期ハンドラ/アラームハンドラのみ	カーネルが管理するオブジェクトには拡張情報を設定する

タスク管理機能

μITRON3.0仕様	μITRON4.0仕様	T-Kernel

C言語記述形式
 	
void task(INT stacd)
{
　;
}
 	
	C言語記述形式
	
	void task(VP_INT exinf)
	{
	　;
	}
	
	exinf:
	　sta_tskで起動した場合stacd
	　act_tskで起動した場合exinf
		C言語記述形式
		
		void task(INT stacd, VP exinf)
		{
		　;
		}

タスクの起動方法 　システムコール:sta_tsk	タスクの起動方法 　タスク生成時の属性で起動指定 　サービスコール:act_tsk/sta_tsk	タスクの起動方法 　システムコール:tk_sta_tsk

タスク管理機能

μITRON3.0仕様	μITRON4.0仕様	T-Kernel
タスクのメインルーチンからリターンした場合は、動作は保障されない	タスクのメインルーチンからリターンした場合は、サービスコールext_tskを呼び出した場合と同じ振る舞いをする	関数からの単純なリターン(return)でタスクを終了することはできない(してはいけない)
		ラウンドロビンスケジューリングをサポート

タスク管理機能(API)

機能	μITRON3.0	μITRON4.0	T-Kernel
タスクの生成	cre_tsk	cre_tsk	
タスクの生成(ID番号自動割付)		acre_tsk	tk_cre_tsk
タスクの削除	del_tsk	del_tsk	tk_del_tsk
タスクの起動		act_tsk	
タスク起動要求のキャンセル		can_act	
タスクの起動(起動コード指定)	sta_tsk	sta_tsk	tk_sta_tsk
自タスクの終了	ext_tsk	ext_tsk	tk_ext_tsk
タスクの強制終了	ter_tsk	ter_tsk	tk_ter_tsk
タスク優先度の変更	chg_pri	chg_pri	tk_chg_pri
タスクスライスタイム変更			tk_chg_slt tk_chg_slt_u
タスク優先度の参照		get_pri	
タスクの状態参照	ref_tsk	ref_tsk	tk_ref_tsk tk_ref_tsk_u
タスクの状態参照(簡易版)		ref_tst	

タスク付属同期機能

μITRON3.0仕様	μITRON4.0仕様	T-Kernel
自タスクに対し起床要求はできない	自タスクに対し起床要求ができる	自タスクに対し起床要求はできない
自タスクを強制待ちにできない	自タスクを強制待ちにできる	自タスクを強制待ちにできない
自タスクを起床待ちにする要求は永久待ち、タイムアウトありの別々のシステムコールがある	自タスクを起床待ちにする要求は永久待ち、タイムアウトありの別々のサービスコールがある	自タスクを起床待ちにするシステムコールは一つで、永久待ちまたはタイムアウトの指定を行う
		待ち状態の許可/禁止を行う機能がある

タスク付属同期機能(API)

機能	μITRON3.0	μITRON4.0	T-Kernel
起床待ち	slp_tsk	slp_tsk	tk_slp_tsk (tmout==TMO_FEVR) tk_slp_tsk_u (tmout_u==TMO_FEVR)
起床待ち(タイムアウトあり)	tslp_tsk	tslp_tsk	tk_slp_tsk (tmout) tk_slp_tsk_u (tmout_u)
タスクの起床	wup_tsk	wup_tsk	tk_wup_tsk
タスク起床要求のキャンセル	can_wup	can_wup	tk_can_wup
強制待ち状態への移行	sus_tsk	sus_tsk	tk_sus_tsk
強制待ち状態からの再開	rsm_tsk	rsm_tsk	tk_rsm_tsk
強制待ち状態からの再開	frsm_tsk	frsm_tsk	tk_frsm_tsk

タスク付属同期機能(API)

機能	μITRON3.0	μITRON4.0	T-Kernel
自タスクの遅延	dly_tsk	dly_tsk	tk_dly_tsk tk_dly_tsk_u
タスクイベントの送信			tk_sig_tev
タスクイベント待ち			tk_wai_tev tk_wai_tev_u
タスク待ち状態の禁止			tk_dis_wai
タスク待ち状態の解除			tk_ena_wai

同期・通信機能

μITRON3.0仕様	μITRON4.0仕様	T-Kernel
セマフォの獲得/返却の資源数は1	セマフォの獲得/返却の資源数は1	セマフォの獲得/返却の資源数は要求時に指定
	スタンダードプロファイルでは、セマフォの最大資源数として65535以上の値が指定できなければならない	セマフォの最大値として少なくとも65535が指定できなければならない
セマフォの獲得待ちにする要求は永久待ち、タイムアウトありの別々のシステムコールがある	セマフォの獲得待ちにする要求は永久待ち、タイムアウトありの別々のシステムコールがある	セマフォの獲得待ちにするシステムコールは一つで、永久待ちまたはタイムアウトの指定を行う
イベントフラグ待ち時のクリア指定は待ち要求時に指定	イベントフラグ待ち時のクリア指定はイベントフラグの属性で指定	イベントフラグ待ち時のクリア指定は待ち要求時に指定
イベントフラグ待ち解除時のクリアは全ビット0	イベントフラグ待ち解除時のクリアは全ビット0	イベントフラグ待ち解除時のクリアは全ビット0か待ち条件クリアかを要求時に指定

同期・通信機能

μITRON3.0仕様	μITRON4.0仕様	T-Kernel
	スタンダードプロファイルではデータキューをサポートすることを規定	
メイルボックスのメッセージ管理がリングバッファ形式かリンク形式かは実装依存	メールボックスのメッセージ管理はリンク形式	メールボックスのメッセージ管理はリンク形式

同期・通信機能(API)：セマフォ

機能	μITRON3.0	μITRON4.0	T-Kernel
セマフォの生成	cre_sem	cre_sem	
セマフォの生成(ID番号自動割付)		acre_sem	tk_cre_sem
セマフォの削除	del_sem	del_sem	tk_del_sem
セマフォ資源の返却	sig_sem	sig_sem	tk_sig_sem
セマフォ資源の獲得	wai_sem	wai_sem	tk_wai_sem (tmout==TMO_FEVR) tk_wai_sem_u (tmout_u==TMO_FEVR)
セマフォ資源の獲得(ポーリング)	preq_sem	pol_sem	tk_wai_sem (tmout==TMO_POL) tk_wai_sem_u (tmout_u==TMO_POL)

同期・通信機能(API)：セマフォ

機能	μITRON3.0	μITRON4.0	T-Kernel
セマフォ資源の獲得(タイムアウトあり)	twai_sem	twai_sem	tk_wai_sem (tmout) tk_wai_sem_u (tmout_u)
セマフォの状態参照	ref_sem	ref_sem	tk_ref_sem

同期・通信機能(API)：イベントフラグ

機能	μITRON3.0	μITRON4.0	T-Kernel
イベントフラグの生成	cre_flg	cre_flg	
イベントフラグの生成(ID番号自動割付)		acre_flg	tk_cre_flg
イベントフラグの削除	del_flg	del_flg	tk_del_flg
イベントフラグのセット	set_flg	set_flg	tk_set_flg
イベントフラグのクリア	clr_flg	clr_flg	tk_clr_flg
イベントフラグ待ち	wai_flg	wai_flg	tk_wai_flg (tmout ==TMO_FEVR) tk_wai_flg_u (tmout_u ==TMO_FEVR)
イベントフラグ待ち(ポーリング)	pol_flg	pol_flg	tk_wai_flg (tmout ==TMO_POL) tk_wai_flg_u (tmout_u ==TMO_POL)

同期・通信機能(API)：イベントフラグ

機能	μITRON3.0	μITRON4.0	T-Kernel
イベントフラグ待ち(タイムアウトあり)	twai_flg	twai_flg	tk_wai_flg (tmout) tk_wai_flg_u (tmout_u)
イベントフラグの状態参照	ref_flg	ref_flg	tk_ref_flg

同期・通信機能(API)：データキュー

機能	μITRON3.0	μITRON4.0	T-Kernel
データキューの生成		cre_dtq	
データキューの生成(ID番号自動割付)		acre_dtq	
データキューの削除		del_dtq	
データキューへの送信		snd_dtq	
データキューへの送信(ポーリング)		psnd_dtq	
データキューへの送信(タイムアウトあり)		tsnd_dtq	
データキューへの強制送信		fsnd_dtq	
データキューからの受信		rcv_dtq	
データキューからの受信(ポーリング)		prcv_dtq	
データキューからの受信(タイムアウトあり)		trcv_dtq	
データキューの状態参照		ref_dtq	

同期・通信機能(API)：メールボックス

機能	μITRON3.0	μITRON4.0	T-Kernel
メールボックスの生成	cre_mbx	cre_mbx	
メールボックスの生成(ID番号自動割付)		acre_mbx	tk_cre_mbx
メールボックスの削除	del_mbx	del_mbx	tk_del_mbx
メールボックスへの送信	snd_msg	snd_mbx	tk_snd_mbx
メールボックスからの受信	rcv_msg	rcv_mbx	tk_rcv_mbx (tmout==TMO_FEVR) tk_rcv_mbx_u (tmout_u==TMO_FEVR)
メールボックスからの受信(ポーリング)	prcv_msg	prcv_mbx	tk_rcv_mbx (tmout==TMO_POL) tk_rcv_mbx_u (tmout_u==TMO_POL)

同期・通信機能(API)：メールボックス

機能	μITRON3.0	μITRON4.0	T-Kernel
メールボックスからの受信(タイムアウトあり)	trcv_msg	trcv_mbx	tk_rcv_mbx (tmout) tk_rcv_mbx_u (tmout_u)
メールボックスの状態参照	ref_mbx	ref_mbx	tk_ref_mbx

時間管理機能

μITRON3.0仕様	μITRON4.0仕様	T-Kernel
システムクロックは1985年1月1日を0とした1ミリ秒数のカウンタ	システム時刻はシステム初期化時に0に初期化したカウンタ	システムクロックは1985年1月1日を0とした1ミリ秒数のカウンタ
システムクロックを変更した場合に、それまで時間待ちしていたタスクや起動を待っていたハンドラの動作タイミングが狂う可能性がある	システム時刻を変更した場合にも、相対時間を用いて指定されたイベントの発生する実時刻は変化しない	システム時刻を変更した場合にも、相対時刻は変化しない
システムクロックのビット数を48ビットと推奨する	システム時刻のビット数に関する推奨値を定めない	システム時刻は64ビット符号付整数
周期起動ハンドラとアラームハンドラは定義する	周期ハンドラとアラームハンドラは生成する(ID番号で管理)	周期ハンドラとアラームハンドラは生成する(ID番号で管理)
	周期ハンドラに起動位相という概念を導入	周期ハンドラに起動位相という概念を導入

時間管理機能(API)

機能	μITRON3.0	μITRON4.0	T-Kernel
システム時刻の設定(実際の時間)	set_tim		tk_set_tim tk_set_tim_u
システム時刻の参照(実際の時間)	get_tim		tk_get_tim tk_get_tim_u
システム時刻の設定		set_tim	
システム稼働時間の参照		get_tim	tk_get_otm tk_get_otm_u
周期ハンドラの生成		cre_cyc	
周期ハンドラの生成(ID番号自動割付)		acre_cyc	tk_cre_cyc tk_cre_cyc_u
周期ハンドラの定義	def_cyc		
周期ハンドラの削除		del_cyc	tk_del_cyc
周期起動ハンドラの活性制御	act_cyc		

時間管理機能(API)

機能	μITRON3.0	μITRON4.0	T-Kernel
周期ハンドラの動作開始		sta_cyc	tk_sta_cyc
周期ハンドラの動作停止		stp_cyc	tk_stp_cyc
周期ハンドラの状態参照	ref_cyc	ref_cyc	tk_ref_cyc tk_ref_cyc_u

システム状態管理機能(API)

機能	μITRON3.0	μITRON4.0	T-Kernel
タスク優先順位の回転	rot_rdq	rot_rdq	tk_rot_rdq
実行状態のタスクIDの参照	get_tid	get_tid	tk_get_tid
CPUロック状態への移行	loc_cpu	loc_cpu	
CPUロック状態の解除	unl_cpu	unl_cpu	
ディスパッチ禁止	dis_dsp	dis_dsp	tk_dis_dsp
ディスパッチ許可	ena_dsp	ena_dsp	tk_ena_dsp
コンテキストの参照		sns_ctx	
CPUロック状態の参照		sns_loc	
ディスパッチ禁止状態の参照		sns_dsp	
ディスパッチ保留状態の参照		sns_dpn	
システムの状態参照	ref_sys	ref_sys	tk_ref_sys
省電力モード設定			tk_set_pow

非タスク部

μITRON3.0仕様	μITRON4.0仕様	T-Kernel
タスク独立部用のシステムコールの名称は、ixxx_yyyとする	非タスクコンテキスト用のサービスコールの名称は、ixxx_yyyとする	タスク独立部用のシステムコールは、タスク用のシステムコールと同じ名称
タスク独立部用のシステムコールの種類は実装依存		

非タスク部(API)

機能	μITRON3.0	μITRON4.0	T-Kernel
タスクの起動		iact_tsk	
タスクの起動			tk_sta_tsk
タスク優先度の変更	ichg_pri		
タスクの起床	iwup_tsk	iwup_tsk	tk_wup_tsk
待ち状態の強制解除	irel_wai	irel_wai	tk_rel_wai
強制待ち状態への移行	isus_tsk		
強制待ち状態からの再開	irsm_tsk		
強制待ち状態からの強制再開	ifrsm_tsk		
タスク例外処理の要求		iras_tex	
セマフォ資源の返却	isig_sem	isig_sem	tk_sig_sem
イベントフラグのセット	iset_flg	iset_flg	tk_set_flg
データキューへの送信(ポーリング)		ipsnd_dtq	
データキューへの強制送信		ifsnd_dtq	
メールボックスへの送信	isnd_msg		

非タスク部(API)

機能	μITRON3.0	μITRON4.0	T-Kernel
メッセージバッファへの送信	ipsnd_mbf		
固定長メモリブロックの獲得	ipget_blf		
可変長メモリブロックの獲得	ipget_blk		
タイムティックの供給		isig_tim	
タスクの優先順位の回転	irot_rdq	irot_rdq	tk_rot_rdq
実行状態のタスクIDの参照		iget_tid	tk_get_tid
CPUロック状態への移行		iloc_cpu	
CPUロック状態の解除		iunl_cpu	
タスクの強制待ち			tk_sus_tsk
タスクイベントの送信			tk_sig_tev
周期ハンドラの動作開始			tk_sta_cyc
周期ハンドラの動作停止			tk_stp_cyc
アラームハンドラの動作開始			tk_sta_alm tk_sta_alm_u
アラームハンドラの動作停止			tk_stp_alm

【講座】T-Kernel/ITRON入門テキスト「T-Kernel入門」

著者　TRON Forum