『RTOS基礎編』 - 座学』

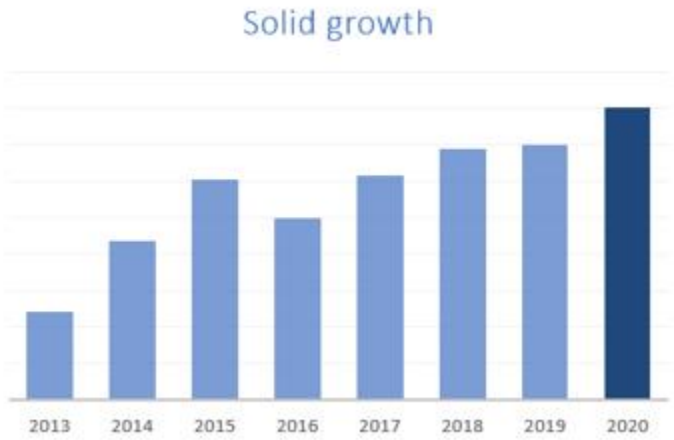
0:RTOSの動作の仕組みや、
RTOS上でのアプリケーションの作成について解説

イー・フォース株式会社　田村　聡

会社紹介

- **本社　　：東京**
- **開発拠点　　：東京 / インド**
- **事業内容**
  - Embedded事業
    - RTOSやミドルウェアの開発・販売
  - IoT事業
    - IoTプラットフォーム「iot-mos」の開発・販売
    - コンサルタント
- **沿革**

2006　:設立
国内のOSベンダとして初めて、ArmのCortex®-M/Aに対応し
「μC3/Compact」、「μC3/Standard」をリリース
2008　:TCP/IPスタック「μNet3」を開発
2013　:マルチコアデバイスに対応した「μC3/Standard+M」をリリース
2015　:産業用イーサネットへの取り組みをスタート
2016　:無線LANモジュール向けのSWパッケージを開発
2017　:RTOSとLinuxを共存させるソリューションをリリース
2018　:IoTプラットフォーム「iot-mos」を発表
2019　:Arm Cortex®-M33(TrustZone)に正式対応



セミナ紹介

■ このウェビナーで学んでほしいこと

- RTOS(リアルタイムOS)上でのアプリケーション作成の流れ
- RTOSの中で、μC3(弊社のRTOS製品)が採用しているμITRONの概要
- μITRONの動作の仕組み、特にタスクの動作について

■ こんな方にオススメです

- RTOSやμITRONの基礎について学んでみたい
- RTOS(特に、μITRON)を使ったSW開発を行う可能性があるので勉強しておきたい
- μC3(弊社のRTOS製品)がどんなものであるのか、概要について聴いてみたい

eForce

今日のプログラム

- 会社紹介とセミナ紹介
- 1. リアルタイム処理とは何か
- 2. RTOSを利用してアプリケーション構築するには
- 3. μITRONの概要と、メリットなどを紹介
- 4. マルチタスク機構の解説
  - 4-1. タスクのスケジューリング
  - 4-2. コンテキスト切替
- 5. eForce社製「μC3 (RTOS)」 の紹介
  - 5-1. μITRON4.0仕様準拠のRTOS
  - 5-2. μC3(RTOS) と μNet3(TCP/IP)

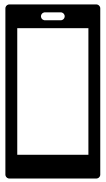
eForce

ソフトウェアの複雑化

～昔は、メカ的な動作で、機器を制御していた～

- その後、マイコンやセンサを搭載して、マイコン制御（ソフト制御)

- 組込みソフトウェアの複雑化
（RTOSのマルチタスクの役割）



組込みでのリアルタイム処理

- 組込みシステム :
  - 現実の物理世界が組込みシステムの対象
- 何が必要 :
  - 外部の環境(音・光・温度・状態)の変化への対応が必要
- 時間制約 :
  - 通信処理・温度制御・車の制御
  - 変化への応答に時間の制約がある
- リアルタイム処理 :
  - 定められた時間内に、処理(決められた手順)を完了する
- リアルタイムOS :
  - リアルタイム処理のアプリケーション構築を支援するOS

eForce

今日のプログラム

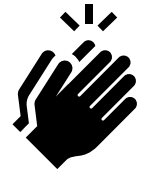
- 会社紹介とセミナ紹介
- 1. リアルタイム処理とは何か
- 2. RTOSを利用してアプリケーション構築するには
- 3. μITRONの概要と、メリットなどを紹介
- 4. マルチタスク機構の解説
  - 4-1. タスクのスケジューリング
  - 4-2. コンテキスト切替
- 5. eForce社製「μC3 (RTOS)」 の紹介
  - 5-1. μITRON4.0仕様準拠のRTOS
  - 5-2. μC3(RTOS) と μNet3(TCP/IP)

eForce

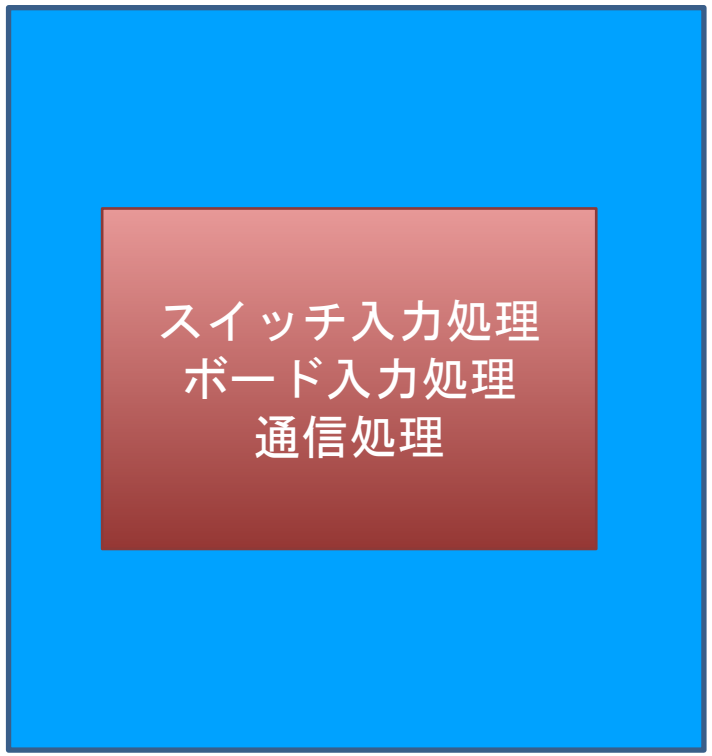
機能毎に処理を分割

- 複数の機能を動作させる組込みシステム
- タスクを利用して、機能毎に処理を分割

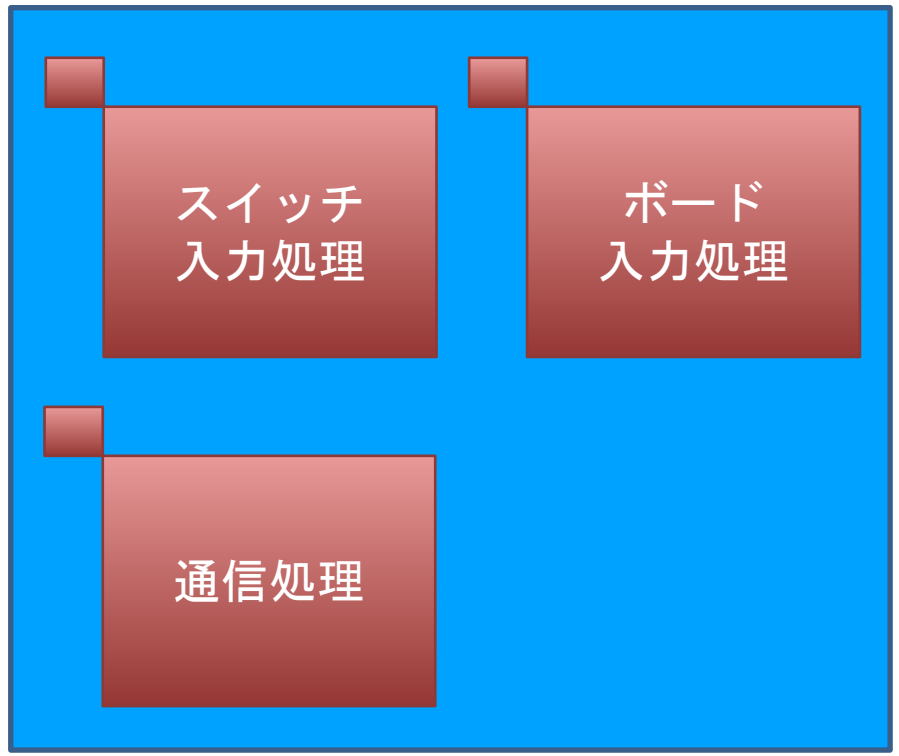
- 機能のモジュール化
- 複数名での並列開発
- 機能単位での保守性・拡張性



RTOSなし



RTOSあり

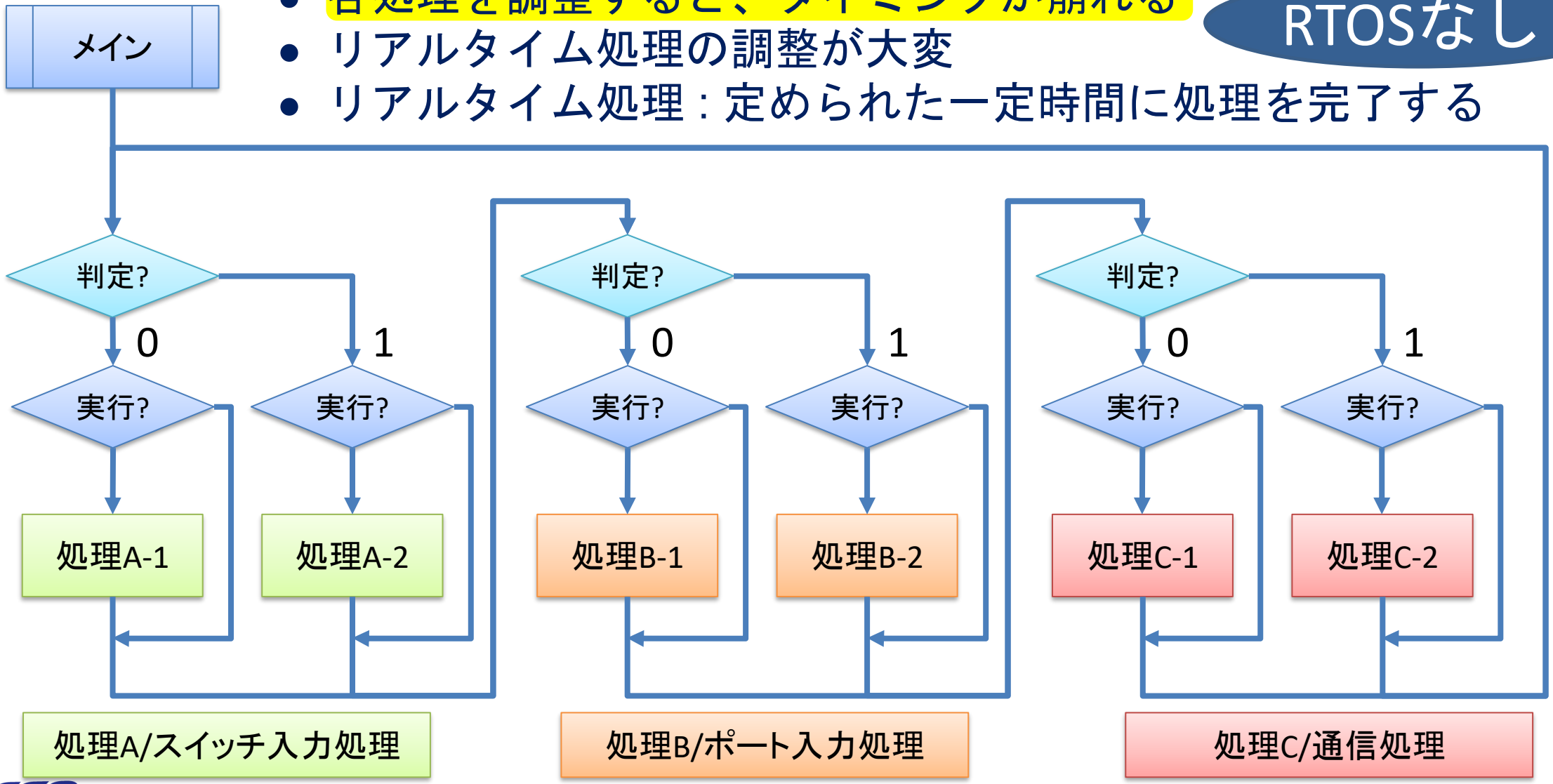


eForce

RTOSを利用しない場合

- 各処理を調整すると、タイミングが崩れる
- リアルタイム処理の調整が大変
- リアルタイム処理：定められた一定時間に処理を完了する

RTOSなし



eForce

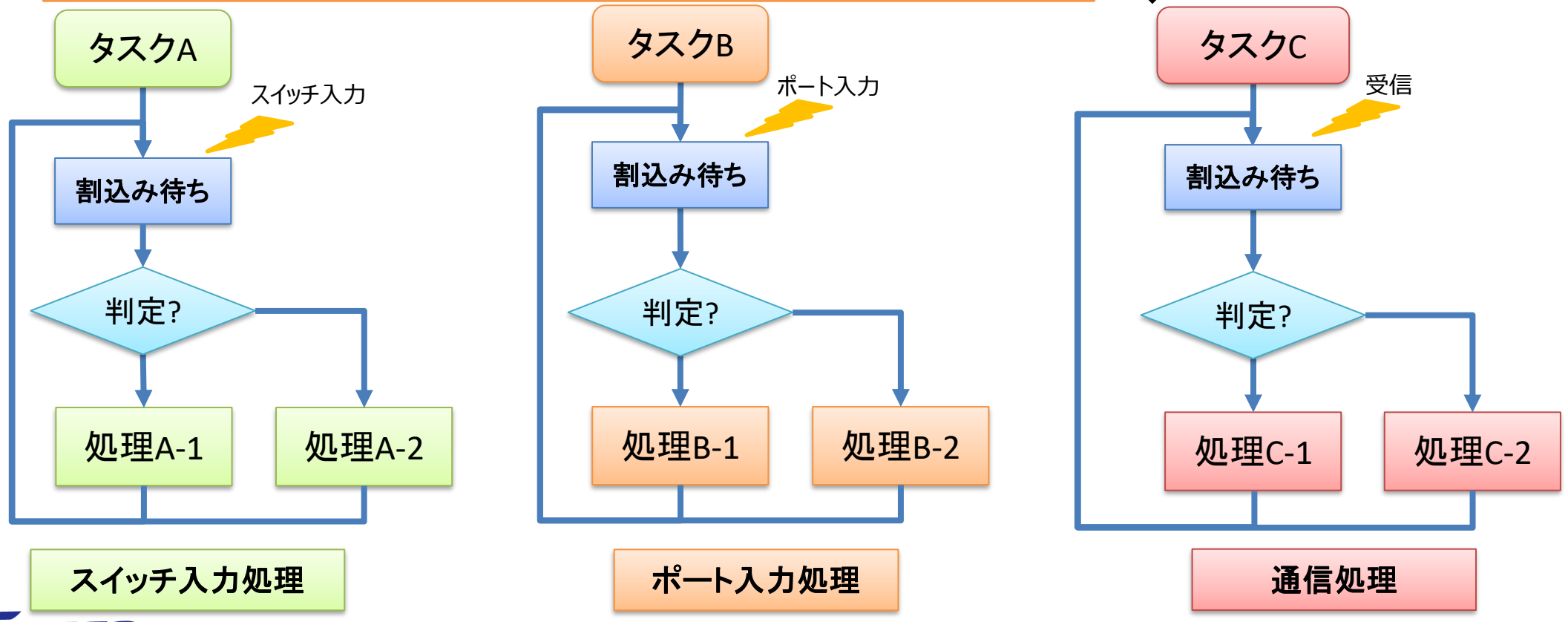
RTOSを利用する場合 (リアルタイム処理)

- 機能単位に分割して、各タスクに処理を実装
- 各タスクでは、イベント駆動型の処理を構築

↓

- アプリケーションのリアルタイム化への貢献

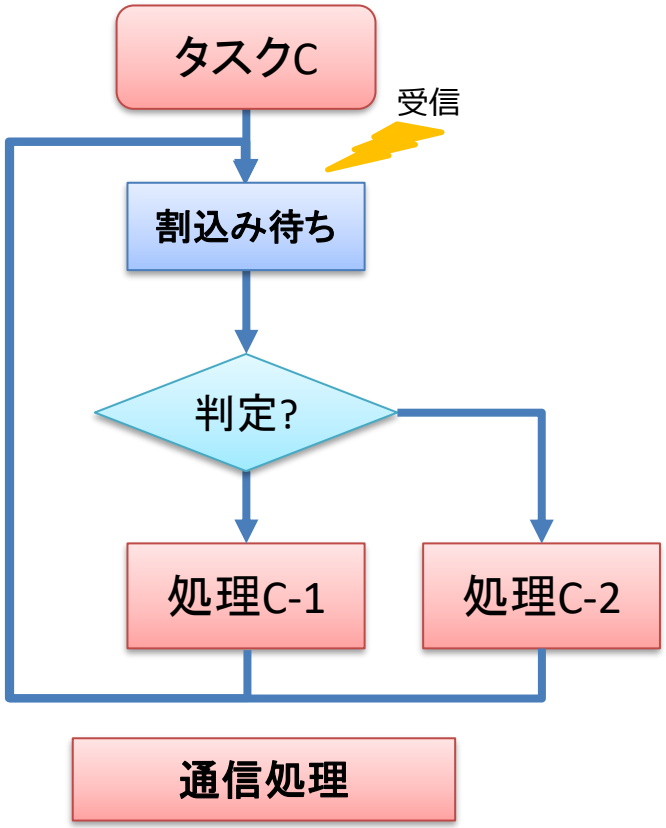
RTOSあり



eForce

関数ベースで処理を構築、割込みも活用

RTOSあり




```
void isr(VP_INT exinf)
{
  割込みサービスルーチン処理
      「タスク待ち解除」
}
```

```
void TaskC(VP_INT exinf)
{
      for(;;) {
          割込み待ち;
          if 判定
              処理C-1;
          else
              処理C-2;
      }
}
```

生成

eForce社のRTOSでは、
- コンフィグレータが
- 関数(スケルトン関数)を生成

構築

アプリケーションの構築
- タスク処理の構築
- 割込み処理の構築

変化

割込みサービスルーチン
- 外部の環境(音・光・温度・状態)の変化に対応する為

eForce

今日のプログラム

- 会社紹介とセミナ紹介
- 1. リアルタイム処理とは何か
- 2. RTOSを利用してアプリケーション構築するには
- 3. μITRONの概要と、メリットなどを紹介
- 4. マルチタスク機構の解説
  - 4-1. タスクのスケジューリング
  - 4-2. コンテキスト切替
- 5. eForce社製「μC3 (RTOS)」 の紹介
  - 5-1. μITRON4.0仕様準拠のRTOS
  - 5-2. μC3(RTOS) と μNet3(TCP/IP)

eForce

μITRON4.0仕様

仕様

最新バージョンはVer. 4.03.03

オープンアーキテクチャー

仕様が公開されており、これを使って誰でも自由に作成することができる
多くの商用μITRONが存在する
→ 各社実装が違うため微妙に仕様が異なっている部分もある

弱い標準化

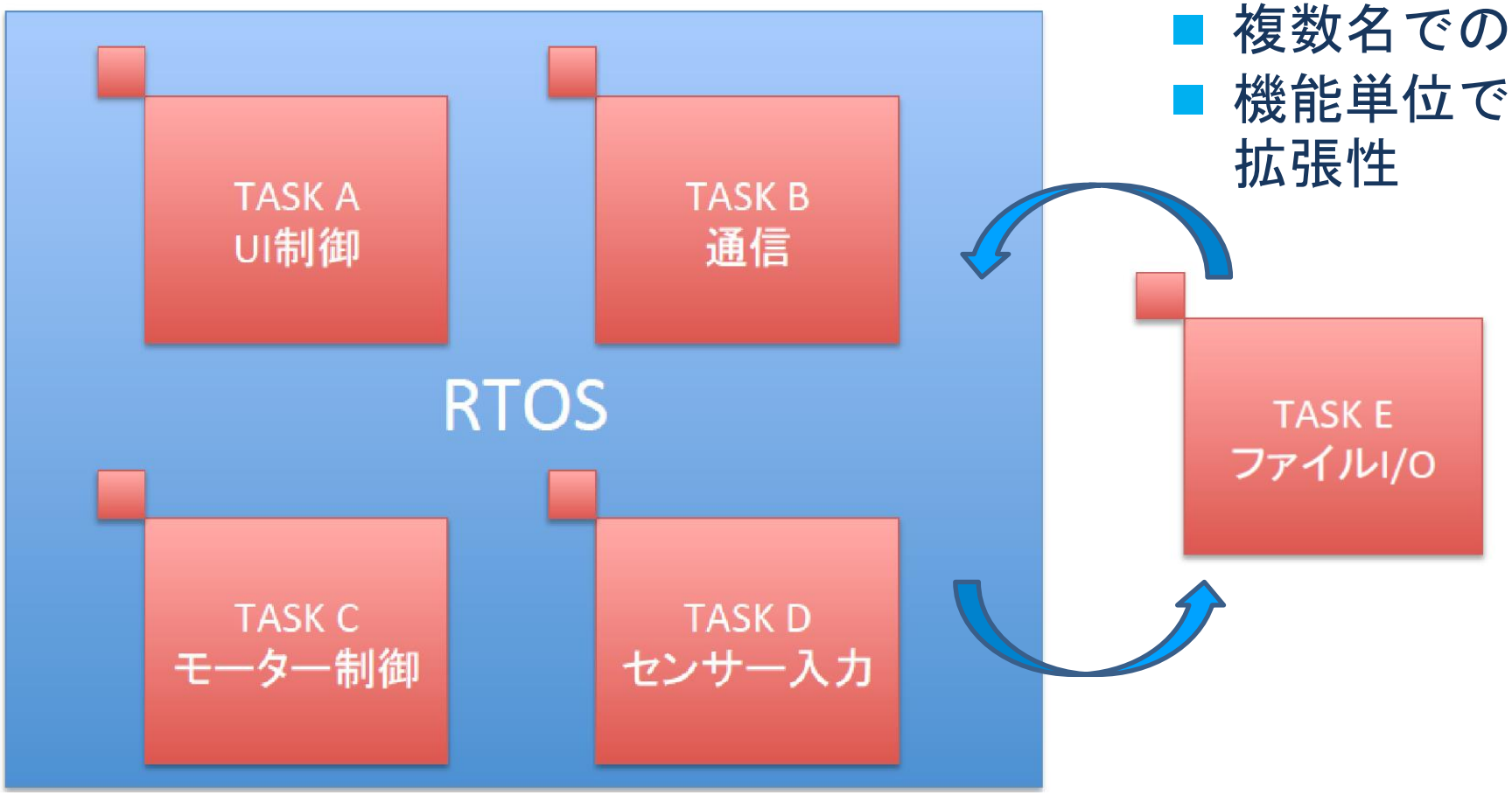
使用するハードウェアやミドルウェアなどの規定が無い
→ 多くのμITRONミドルウェアが存在するが、互換性が無い実装依存部が多く、性能もまちまち

eForce

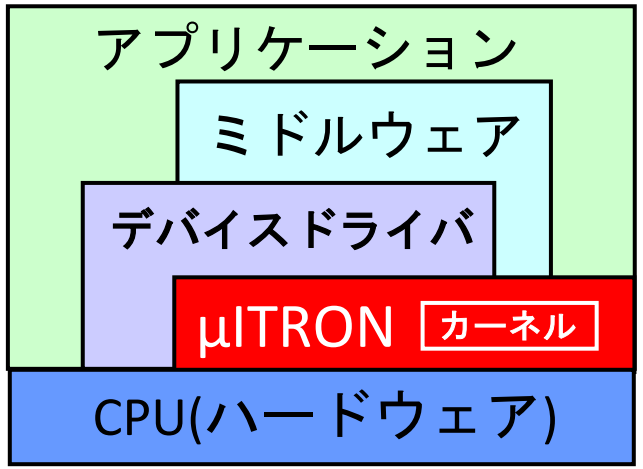
保守性と効率向上

タスク分割による保守性と、開発効率の向上

- 機能のモジュール化
  - 機能の追加/削除が容易
- 複数名での並列開発
- 機能単位での保守性・拡張性

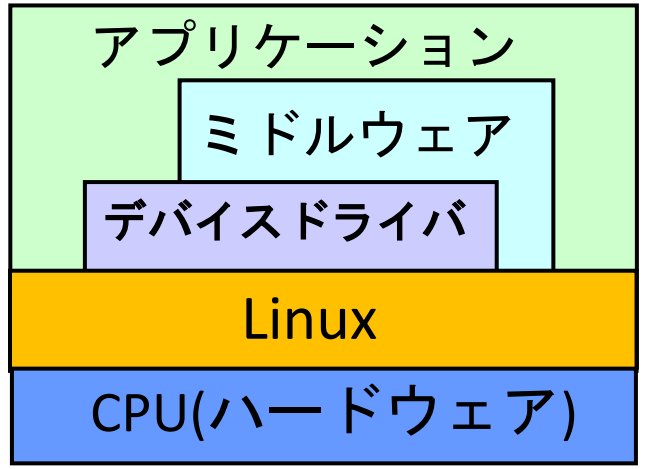


リアルタイム処理が得意なμITRON !!!



- μITRONの中心となるモジュール
- 「リアルタイムカーネル」と呼ばれることも

※特徴:
割込み処理もリアルタイム



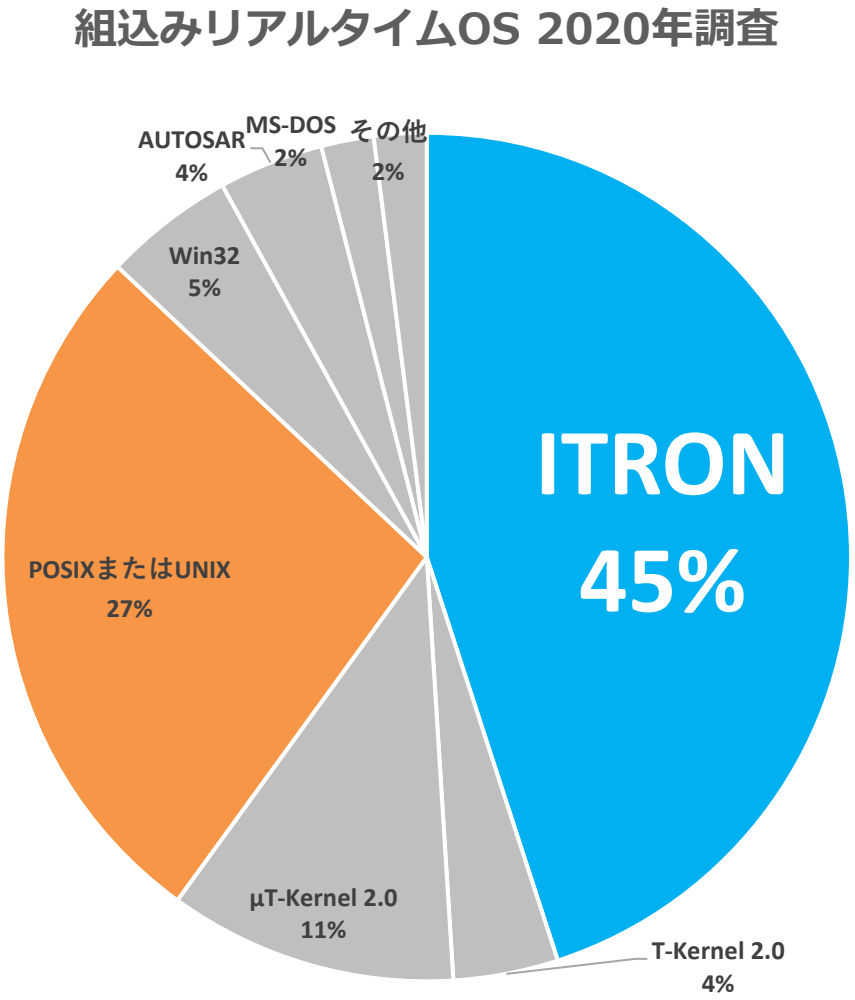
リアルタイム処理は、苦手

- **μITRONの主要な機能**
  - タスクスケジューリング
  - タスク間協調
  - メモリ管理
  - 割込み管理
  - 時間管理

システムコール（OS提供関数）で実現

カーネルは、機能単位で構築され、コードサイズの節約も可能

国内はITRONが24年連続シェアNo.1



出典：トロンフォーラム

ITRONの位置づけ

- ITRON APIは国内で最も使用されているAPI
- 幅広い用途での採用実績
（産業機器、医療機器、IoT、通信、自動車、OA等）
- サポートベンダ数もNo.1
（OSベンダ、Middlewareベンダ、CPUベンダ等）

µC3はµITRON仕様のRTOS

ミドルウェアの活用

システムに、機能を追加したい！しかも短期間で！
Ethernet, WIFI, USB, SD, Bluetooth, GUIなど

- 多くのミドルウェアはRTOS上で動作します。

μNet3
micro net cube

小さなメモリで動作するTCP/IPスタック

μC3
WLAN SDK

ソフトウェア内蔵無線LANモジュール
アプリケーション開発キット

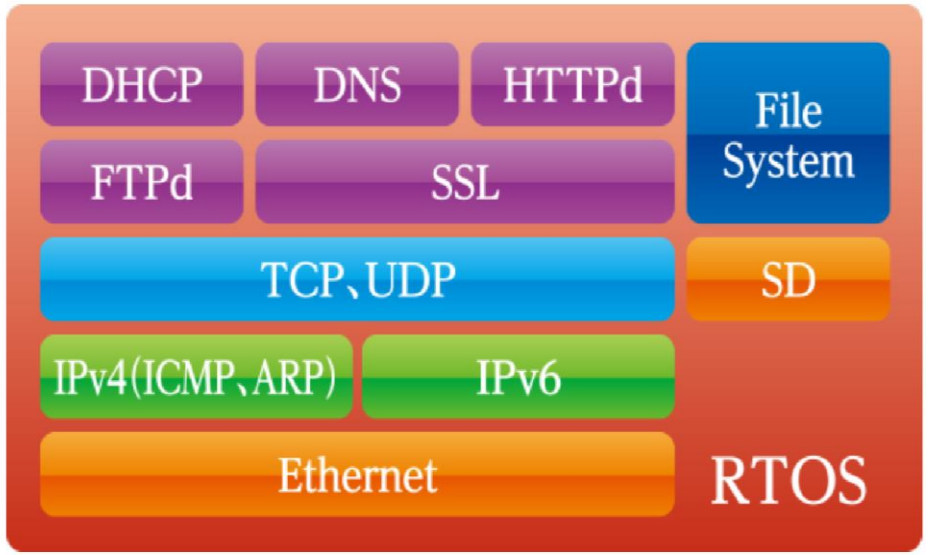
MatrixQuest USB/host

USB1.1/2.0対応ホストドライバ

動作確認済みOS

OS依存部をWrapper層に集約することにより高い移植性を実現しています。
Wrapper層は以下に記載されているOS上で動作検証を行っています。
以下に記載されていないOSの対応や実績については お問い合わせ ください。

- 各種μITRON ： NORTi、μC3、TOPPERSJSP/ASP他
  各種半導体メーカ製μITRON
- 各種T-Kernel ： T-Kernel、μT-Kernel、eT-Kernel
- その他OS ： VxWorks、CMSIS-RTOS RTX



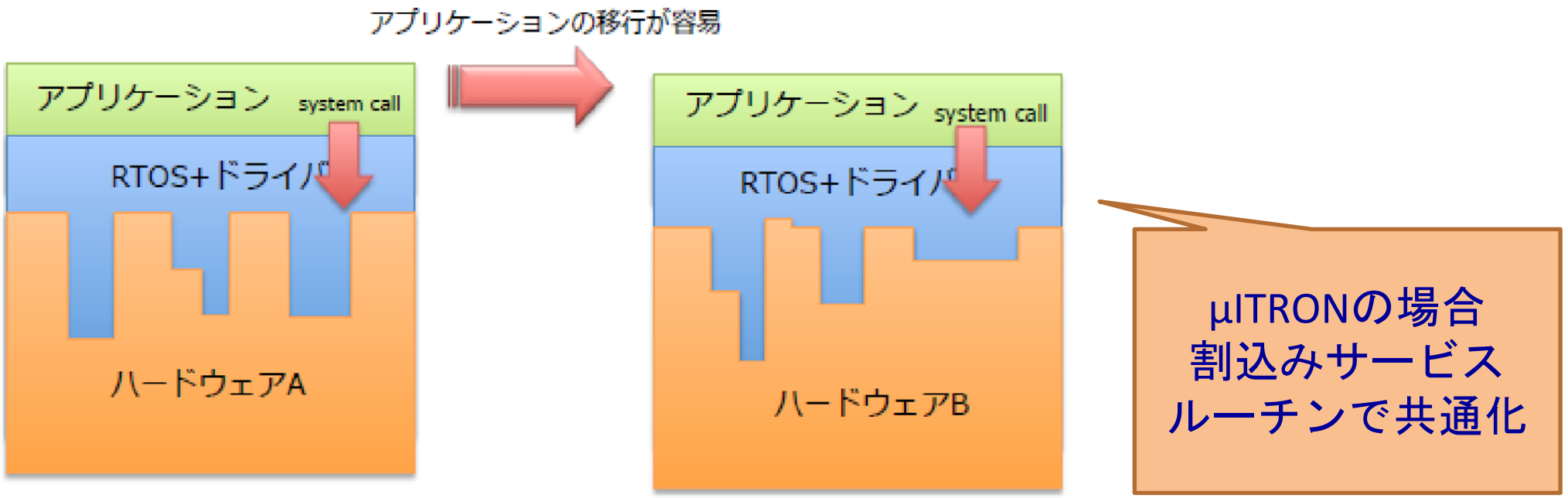
eForce

開発効率の向上

資産を有効活用して、開発効率を上げる

- 社内の資産をもとにした開発効率の向上
- ソフトウェアのメンテナンス効率の向上

- ハードウェア(マイコン)変更時などでも、移植が容易
- HWの違いをRTOSが吸収



ここまでのまとめ「RTOSを利用した開発の特長」

- 組込み機器 複雑化➡RTOSによる開発の支援

- RTOSの利用:タスクの活用
  - 機能単位に処理を分割して開発
  - リアルタイム処理の開発のしやすさ

- RTOSを利用した開発の特長
  - 保守性や開発効率の向上
  - メンテナンス、移植作業
  - ミドルウェアの活用

～開発工数・コストの削減～

eForce

今日のプログラム

- 会社紹介とセミナ紹介
- 1. リアルタイム処理とは何か
- 2. RTOSを利用してアプリケーション構築するには
- 3. μITRONの概要と、メリットなどを紹介
- 4. マルチタスク機構の解説
  - 4-1. タスクのスケジューリング
  - 4-2. コンテキスト切替
- 5. eForce社製「μC3 (RTOS)」 の紹介
  - 5-1. μITRON4.0仕様準拠のRTOS
  - 5-2. μC3(RTOS) と μNet3(TCP/IP)

eForce

リアルタイムOSの特徴

- 時間管理 : システム時刻管理・周期処理の機能
  - 定周期処理
  - タイミングの制御

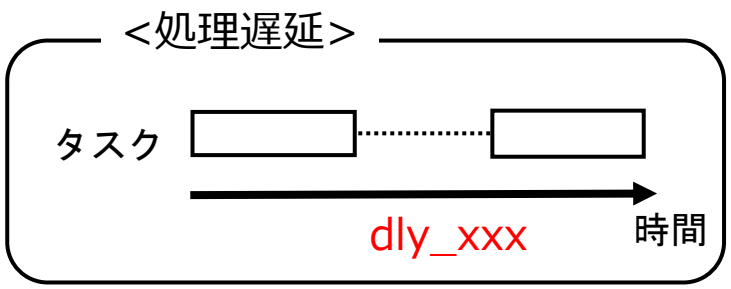
- スケジューラの仕組み :
  - 優先度の高いタスクから実行
  - 最初に実行待ち状態に入ったタスクから実行(FCFS方式)

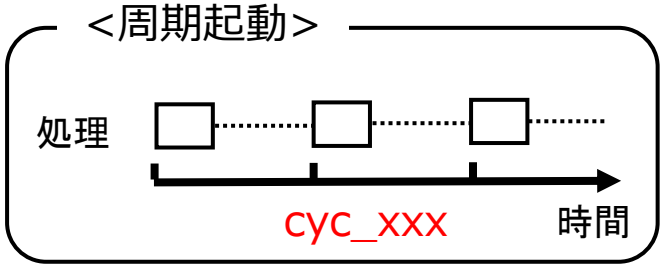
- マルチタスク :
  - タスク単位での機能分割
  - 各タスクには、優先度を設定できる
    - 重要な機能を優先的に実行

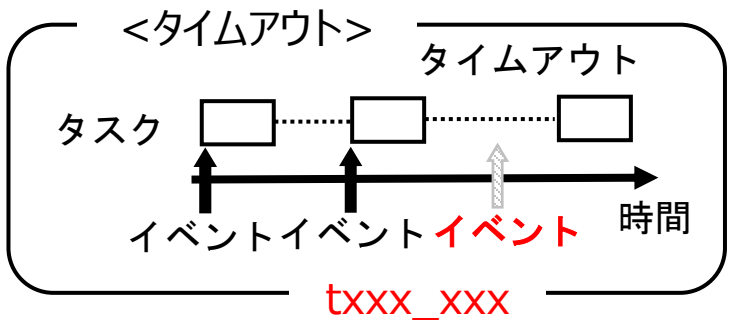
時間管理の仕組み

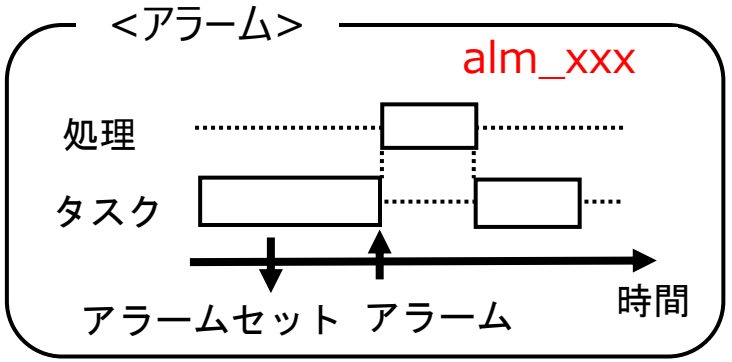
- カーネルタイマを利用して時間を管理
  - 処理遅延
  - 周期的な動作　（周期起動）
  - イベント待ちのタイムアウト
  - 時間経過の監視　（アラーム）

- 周期ハンドラ
- アラームハンドラ(uC3/Standard)



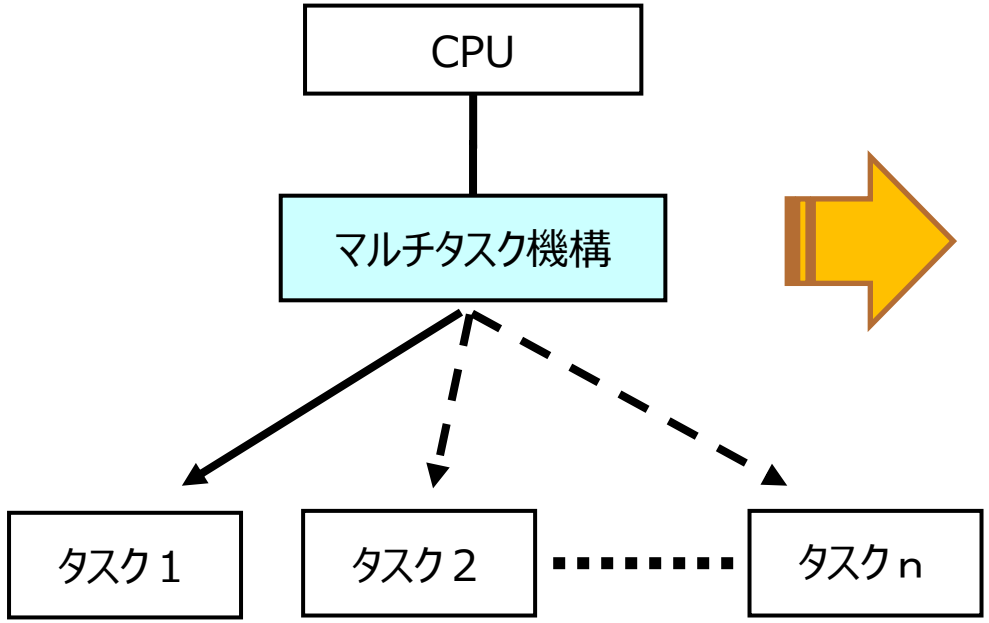


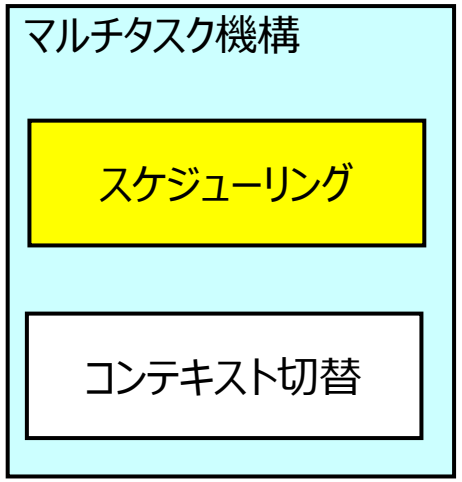




マルチタスク機構

- マルチタスク機構
  - CPUでは、ある時に、実行できるタスクはひとつのみ
  - それを時分割で制御して、複数のタスクを擬似的に平行実行させるための機構





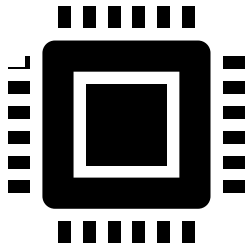
ここで、紹介します。

- スケジューリング
  タスクの実行を制御
- コンテキスト切替
  タスクの切替を制御

eForce

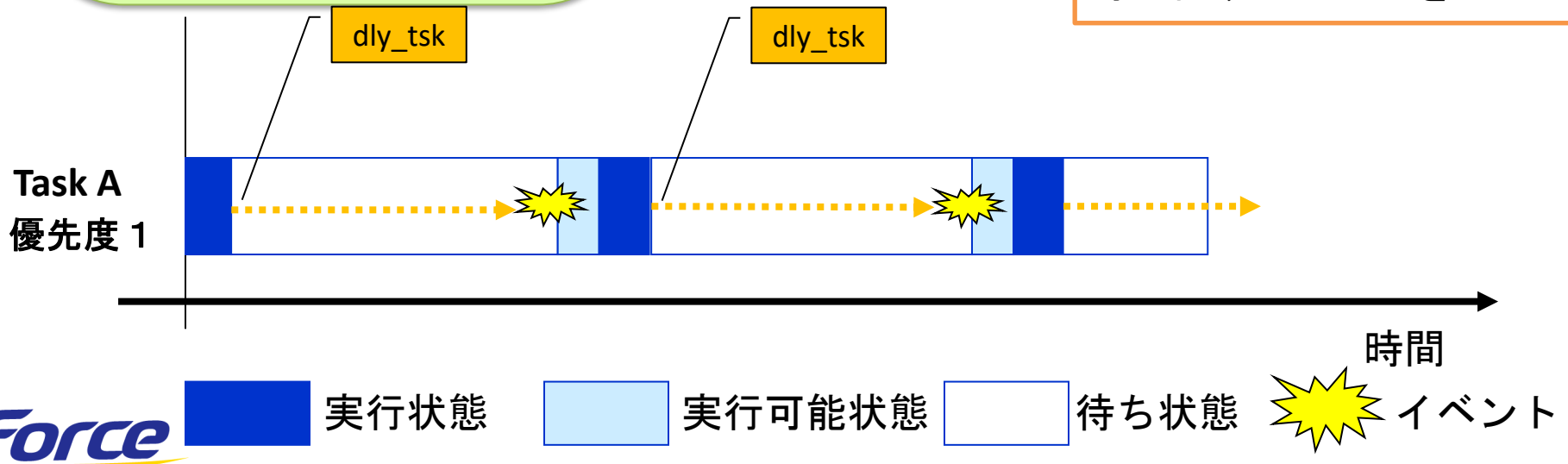
LEDの点滅サンプルで考える：時間管理

```
void TaskA(VP_INT exinf)
{
        for(;;) {
        LED点灯処理;
        dly_tsk(500);
        LED消灯処理;
        dly_tsk(500);
        }
}
```



dly_tsk: 自タスクの遅延
【書式】ER ercd = dly_tsk (RELTIM dlytim)
【パラメータ】RELTIM dlytim 自タスクの遅延時間
【解説】自タスクをシステムコールが呼び出された時刻からdlytim で指定される時間の間、時間経過待ち状態に遷移させます。

システムコール
RTOSの機能を利用するために、アプリケーションから呼び出すAPIのことを示します。



タスクの状態遷移とディスパッチ：基本の状態遷移

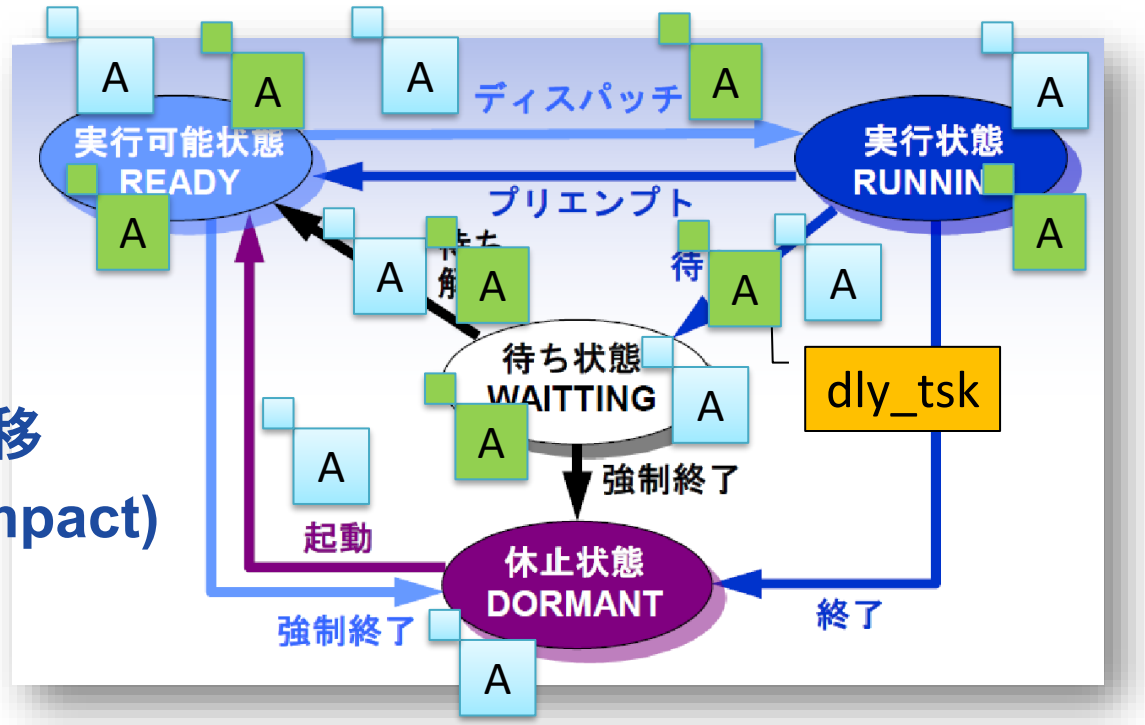
```
void TaskA(VP_INT exinf)
{
    for(;;) {
        LED点灯処理;
        dly_tsk(500);
        LED消灯処理;
        dly_tsk(500);
    }
}
-----    中略    -----
int main(void)
{
    /*μC3システム起動*/
    start_uC3();

}
```

dly_tsk: 自タスクの遅延

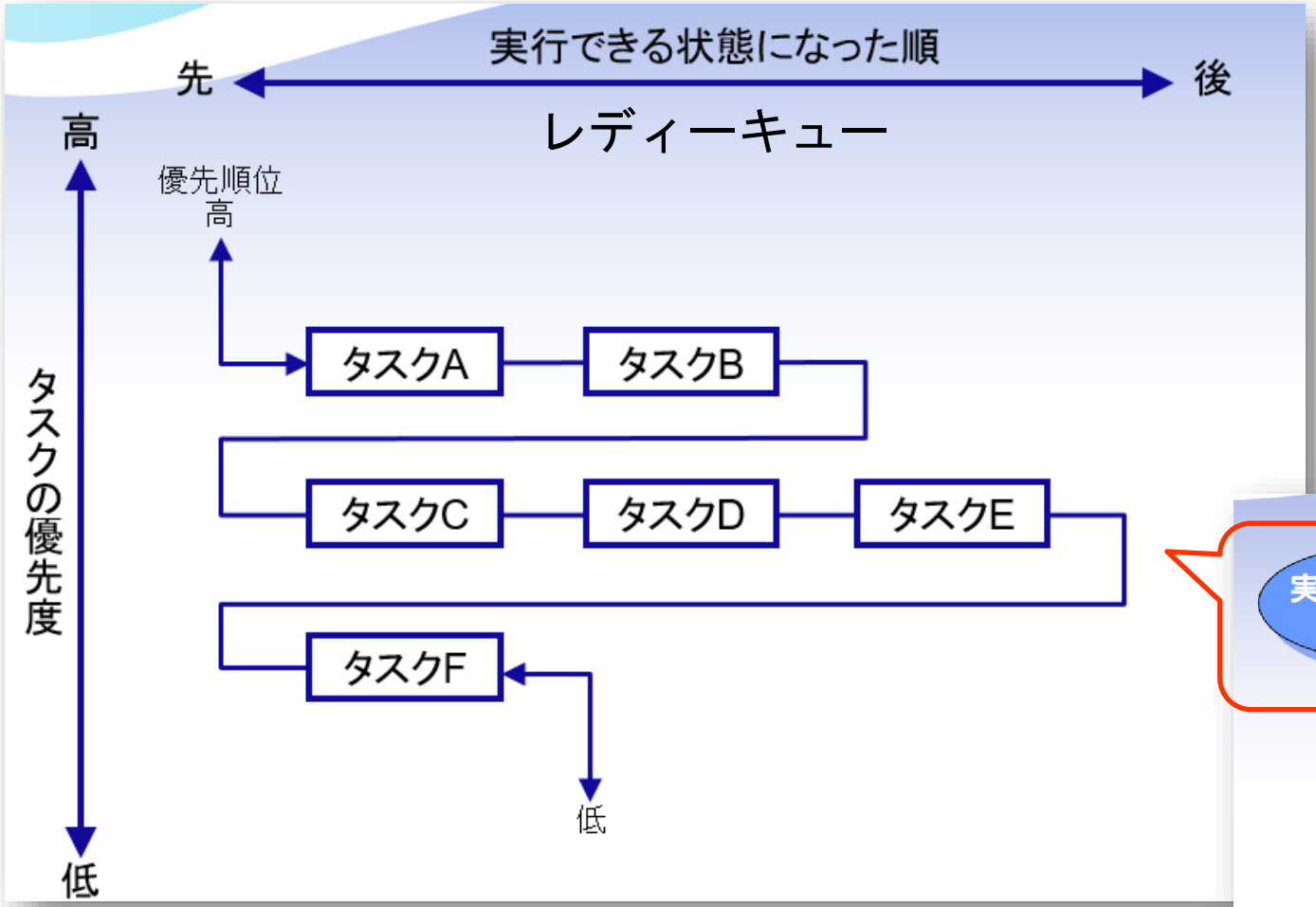
【解説】自タスクをシステムコールが呼び出された時刻からdlytim で指定される時間の間、時間経過待ち状態に遷移させます。

ディスパッチ:
プロセッサが、実行しているタスクの切り替えを行うこと



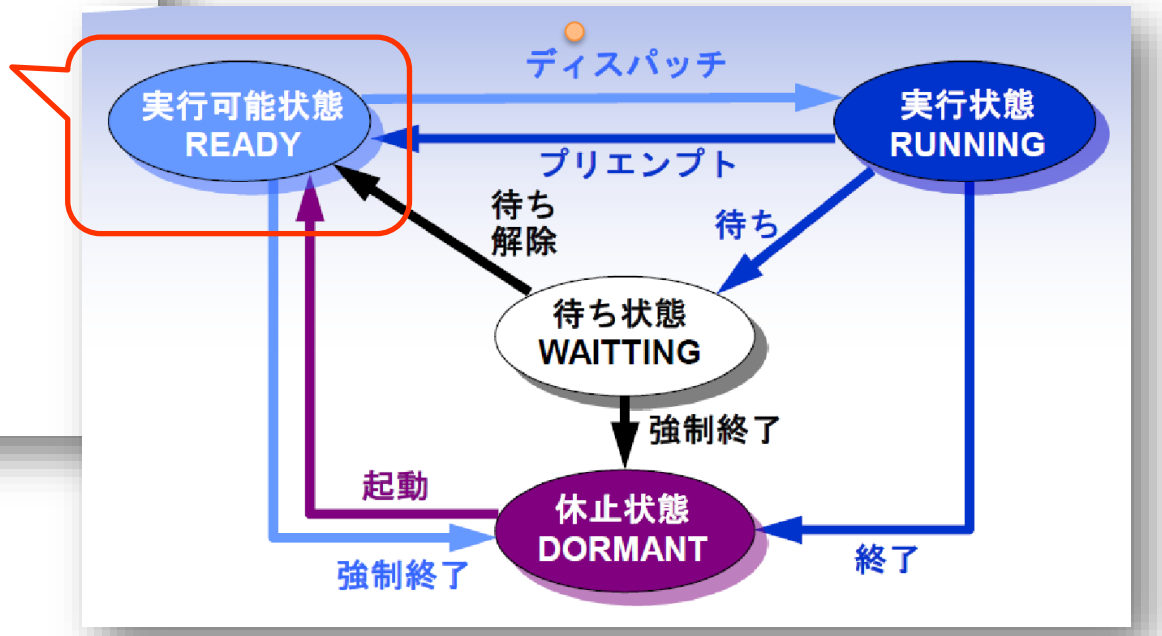
基本の状態遷移
(参考例: μC3/Compact)

タスクの優先度とレディーキュー：REDY状態のタスク遷移

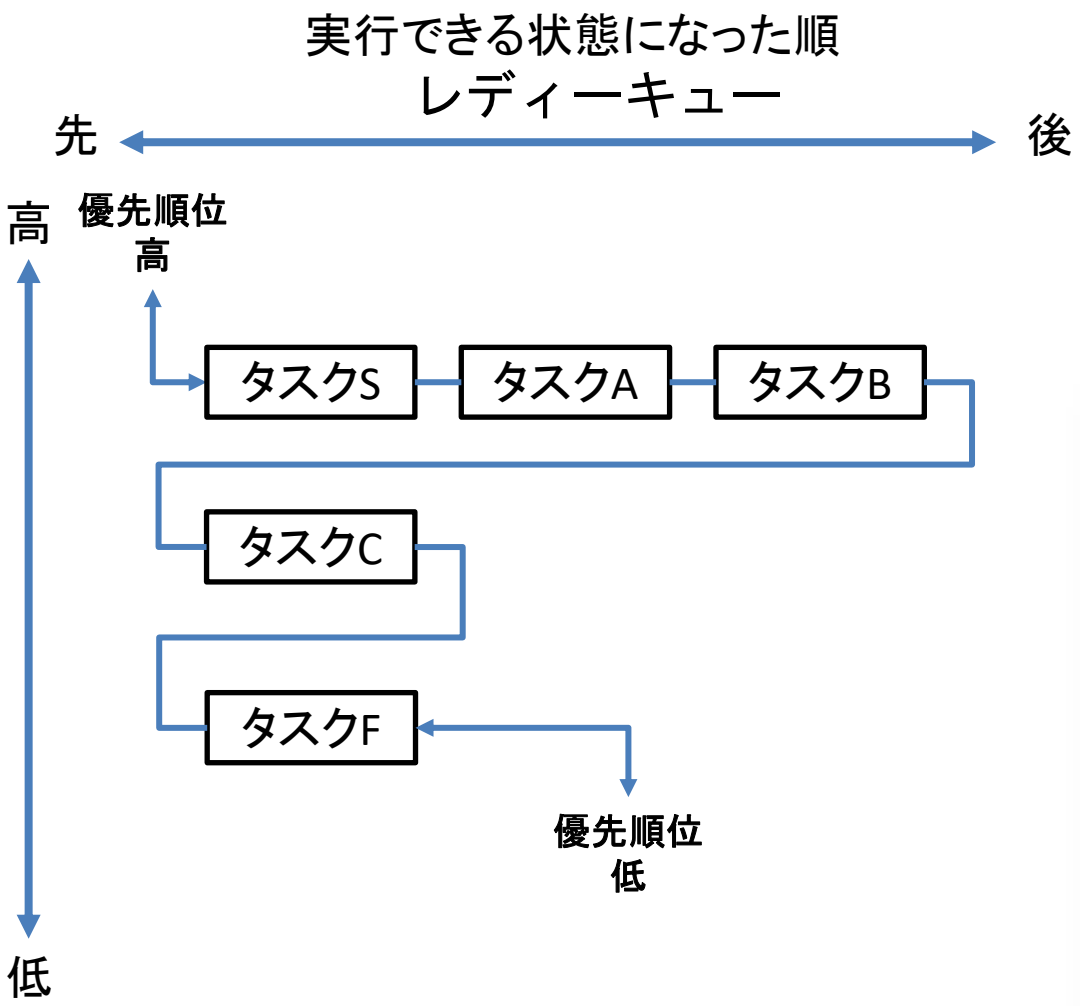


基本の状態遷移

(参考例: μC3/Compact)



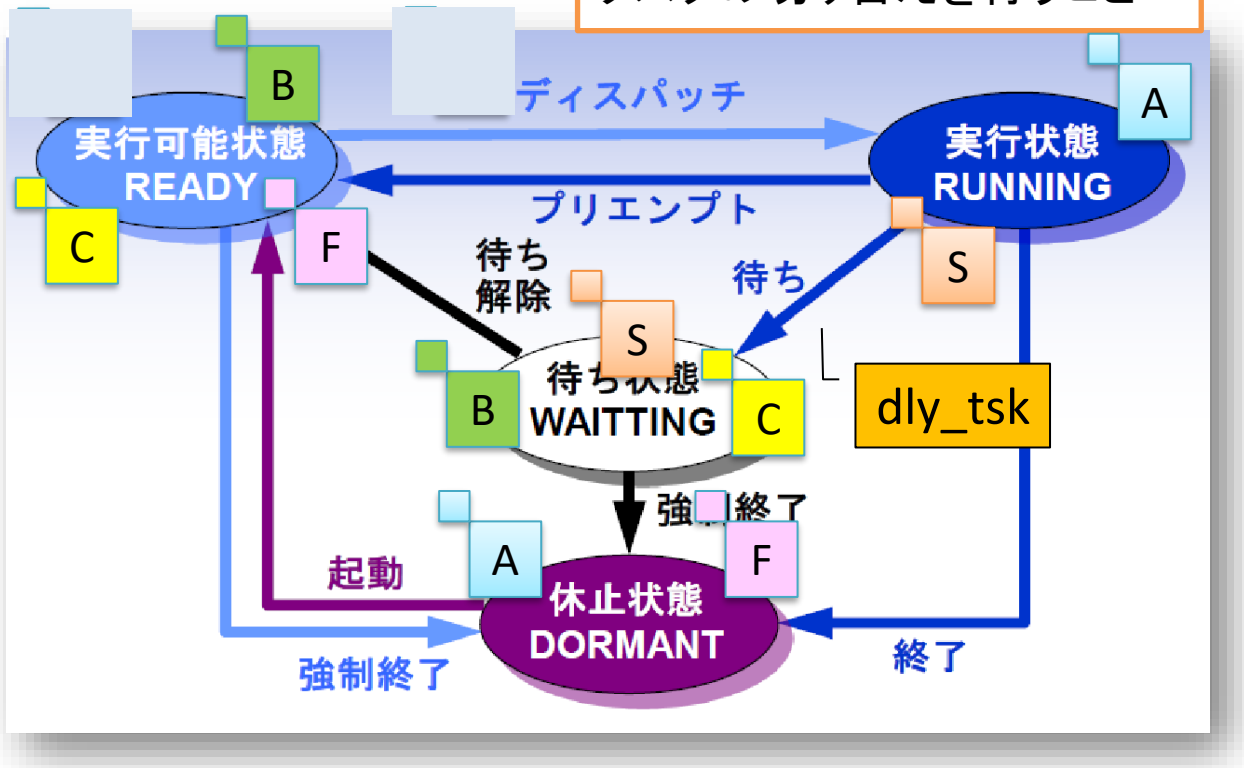
FCFS :最初に実行可能状態に入ったタスクから実行



dly_tsk: 自タスクの遅延

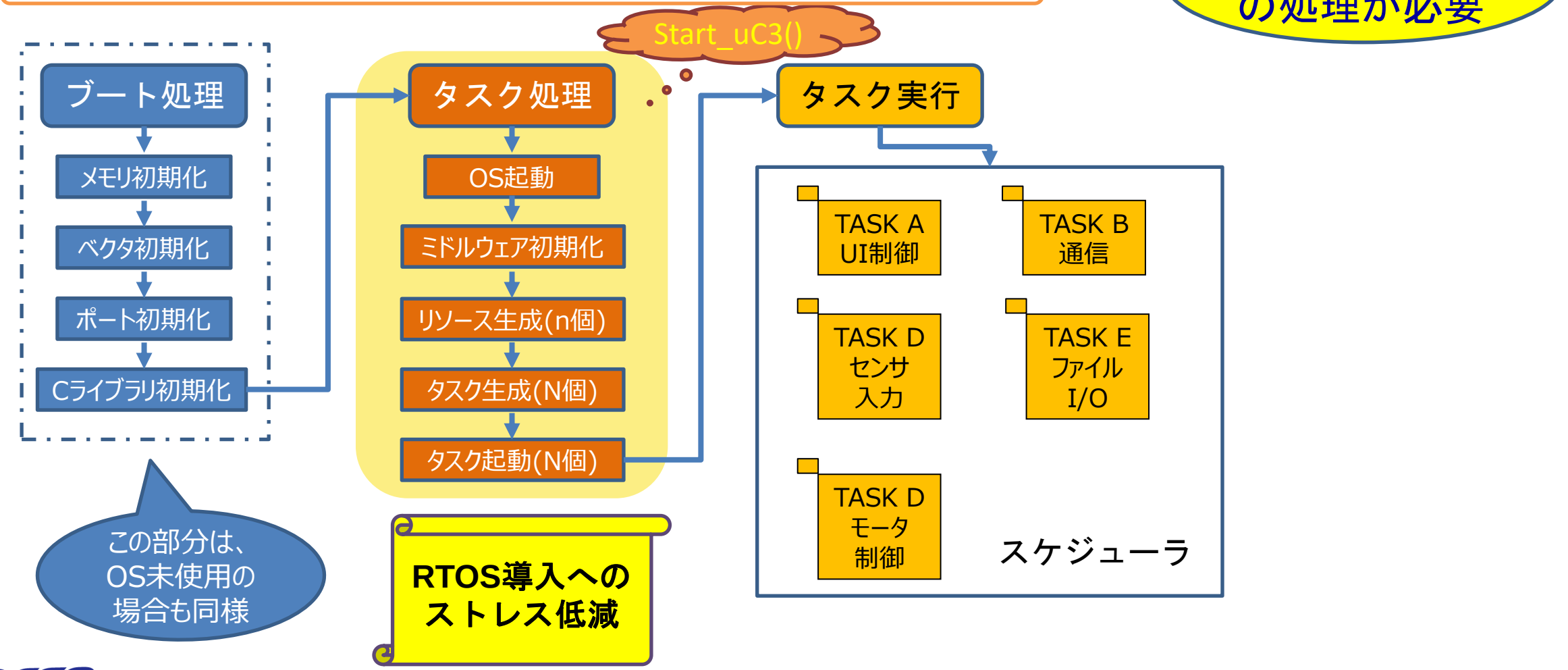
【解説】自タスクをシステムコールが呼び出された時刻からdlytim で指定される時間の間、時間経過待ち状態に遷移させます。

ディスパッチ:
プロセッサが、実行している
タスクの切り替えを行うこと

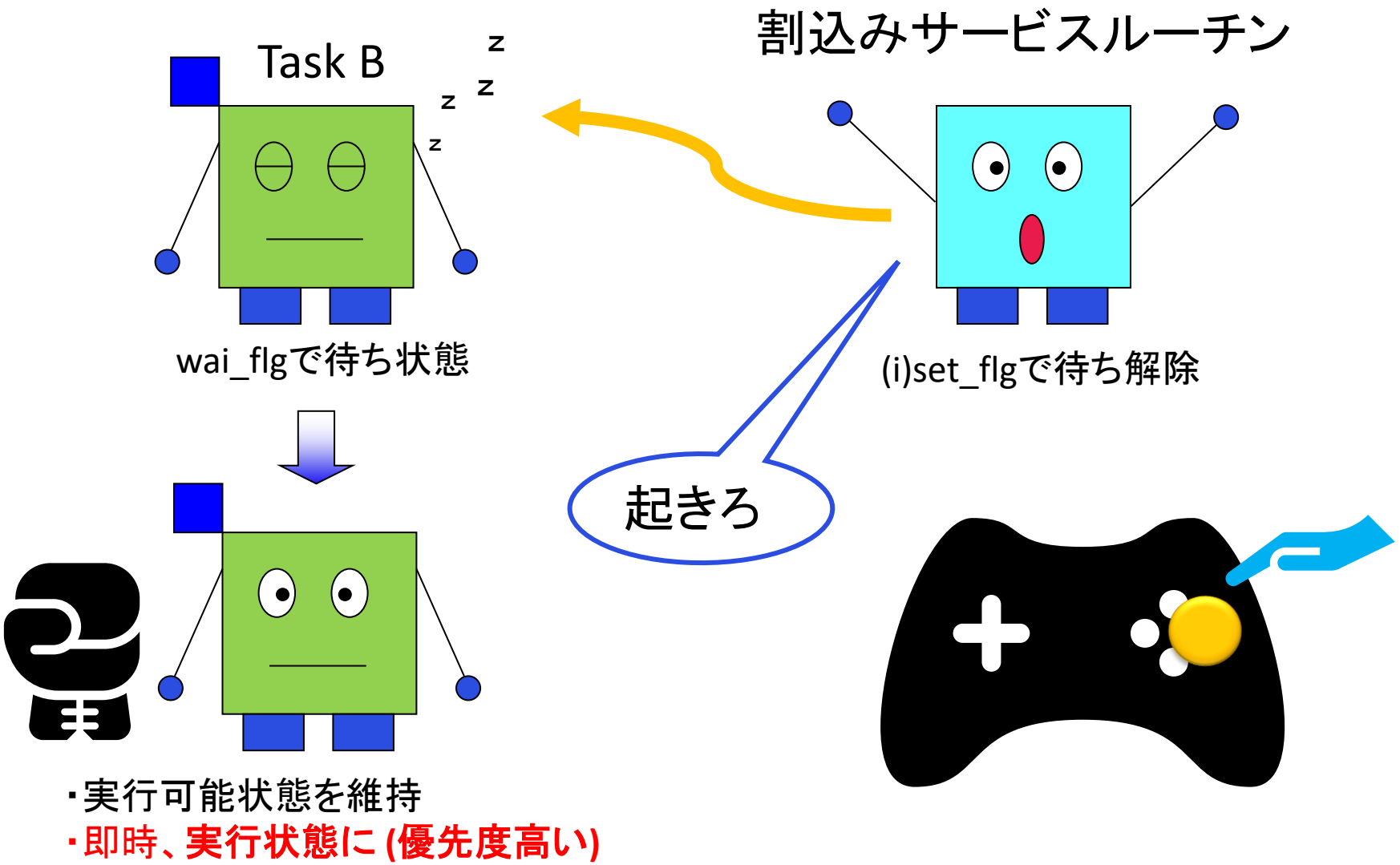


RTOSの動作開始

- 真中の処理が必要になる
  (eForce社のRTOSでは、コンフィグレータが処理を構築・生成)



同期処理：優先度の設定も併用



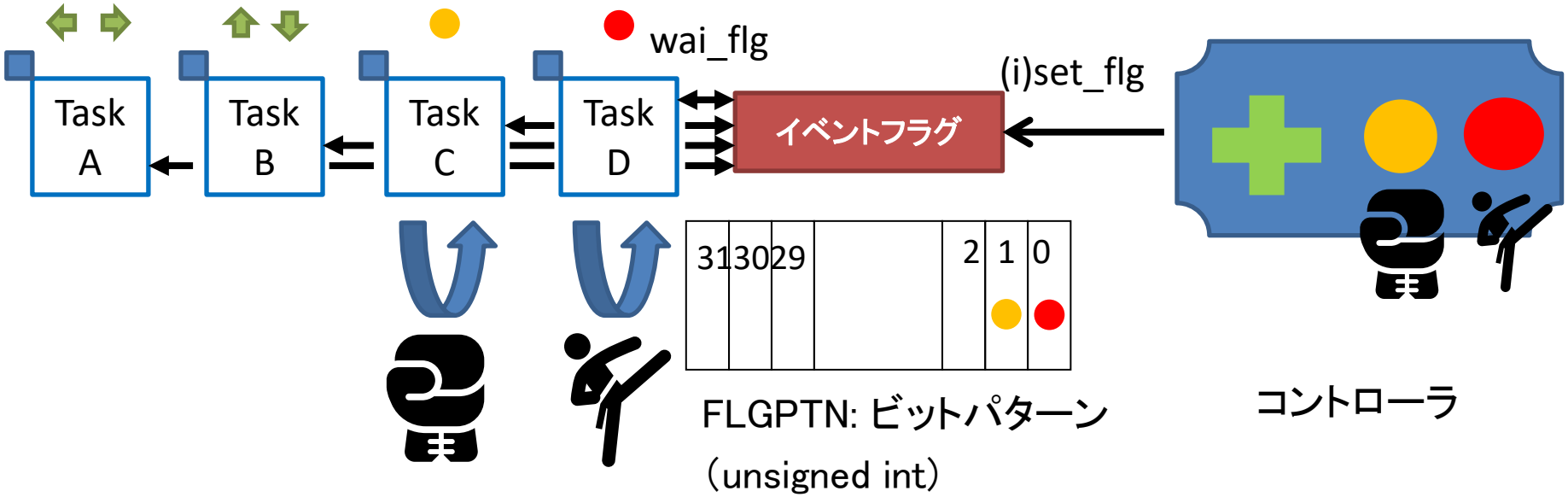
・実行可能状態を維持

・即時、**実行状態に (優先度高い)**

eForce

条件待ち合わせ(イベントフラグの動作紹介)

システムコール
RTOSの機能を利用するために、アプリケーションから呼び出すAPIのことを示します。



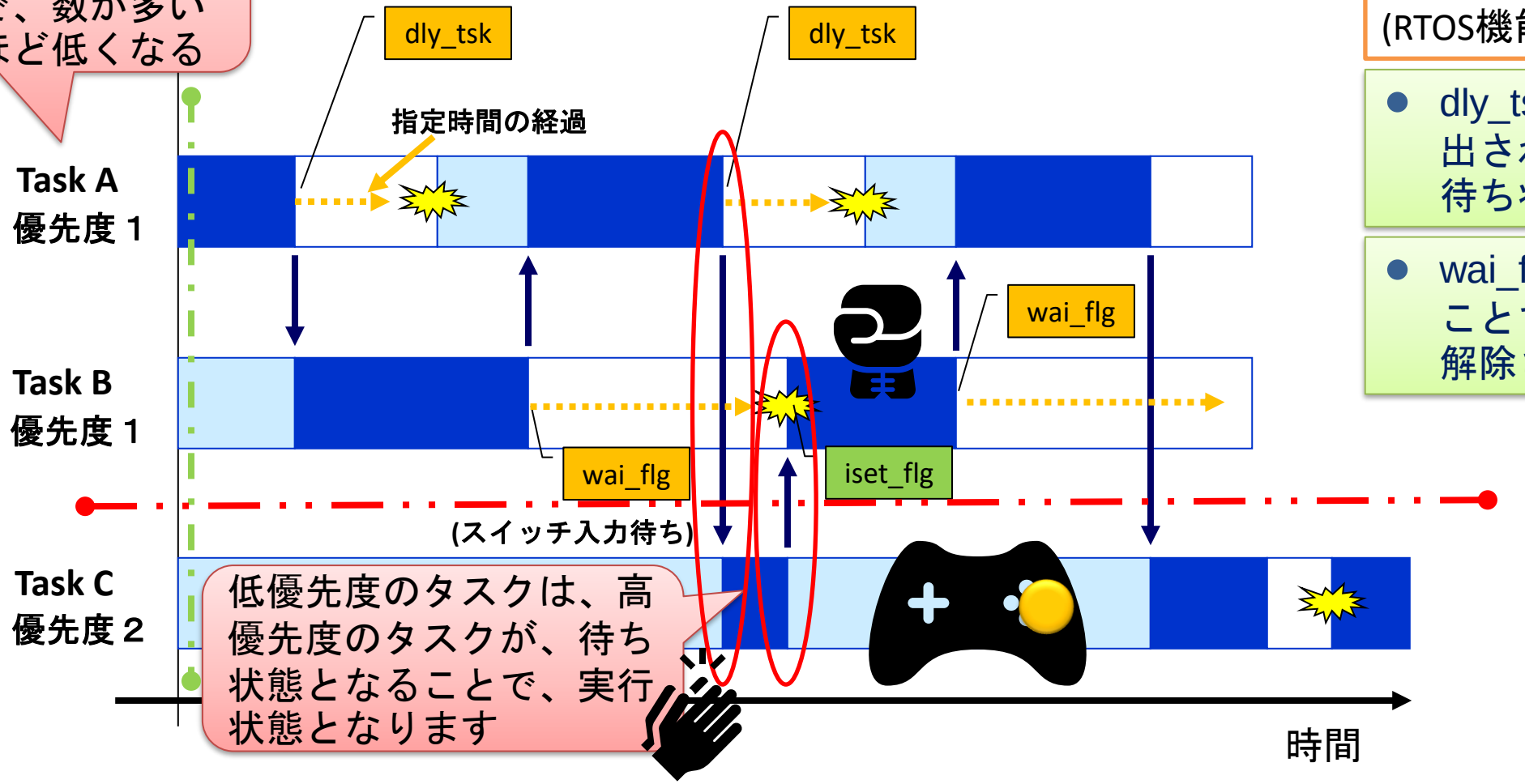
- イベントフラグ

- イベントフラグを利用して、コントローラのボタンの情報を通知
- 受け取り側で、ボタンに応じた処理の構築が可能

優先度ベースのRTOSならではのメリット!!!

タスク優先度は、1が最高で、数が多いほど低くなる

Task A = Task B > Task C :『優先度』



システムコール
アプリケーションから呼び出すAPI(RTOS機能)を示す

- dly_tsk:システムコールが呼び出された時刻から、時間経過待ち状態に遷移
- wai_flg: (i)set_flgを呼び出すことで、タスクの待ち状態が解除される

イベント発生時

待ち状態に遷移するシステムコール

実行状態　実行可能状態　待ち状態　イベント

RTOS動作 (タスクスケジューリング) のまとめ

- タスク優先度は1が最高で、数が大きいほど低くなる
  - 参考例 : (μC3/Compact:16) (μC3/Standard:31)

- システムコールを呼ばなければ、
  低優先度のタスクに状態が遷移しなくなる可能性がある

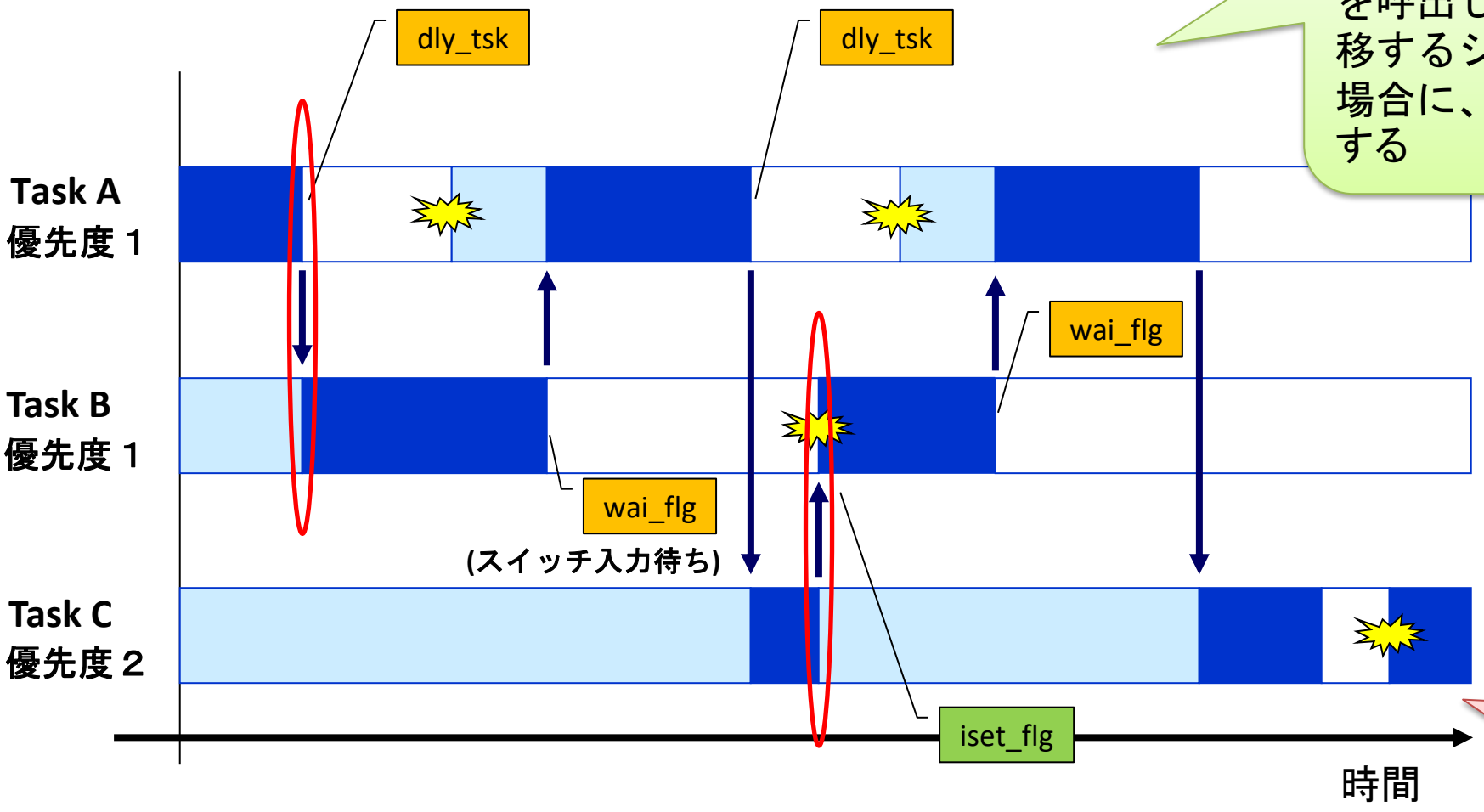
- プリエンプト・ディスパッチにより、RTOSがタスク切替を実施する

- タスク間協調の紹介
  - 複数のタスクが連携してシステムを構築

eForce

タスクスケジューリング、詳細説明

～タスクスケジューリング例～



イベント発生時にシステムコールを呼出した場合や、待ち状態に遷移するシステムコールを呼出した場合に、タスクの切替わりが発生する

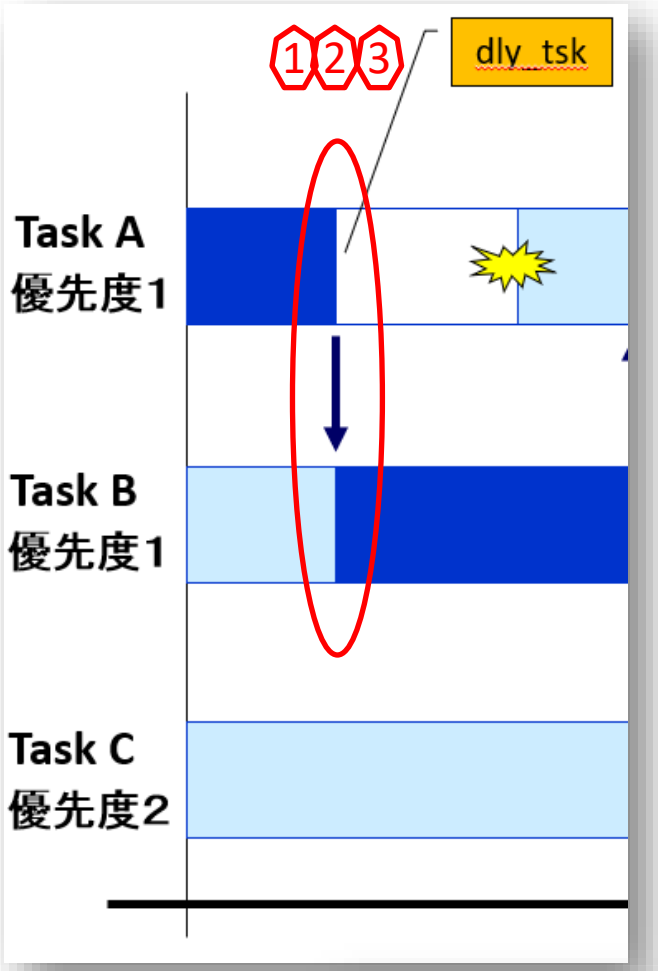
イベント発生時

待ち状態に遷移するシステムコール

赤の2か所のタスク遷移について説明をします

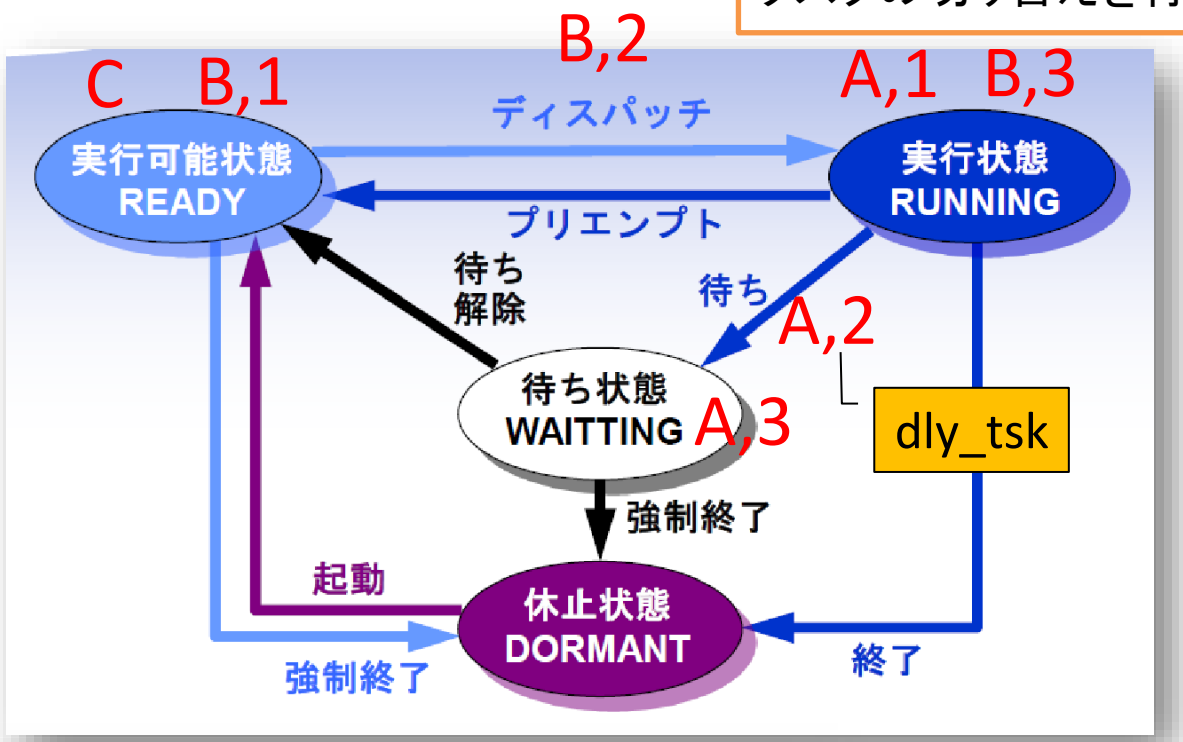
実行状態　実行可能状態　待ち状態　イベント

タスクスケジューリング、詳細説明



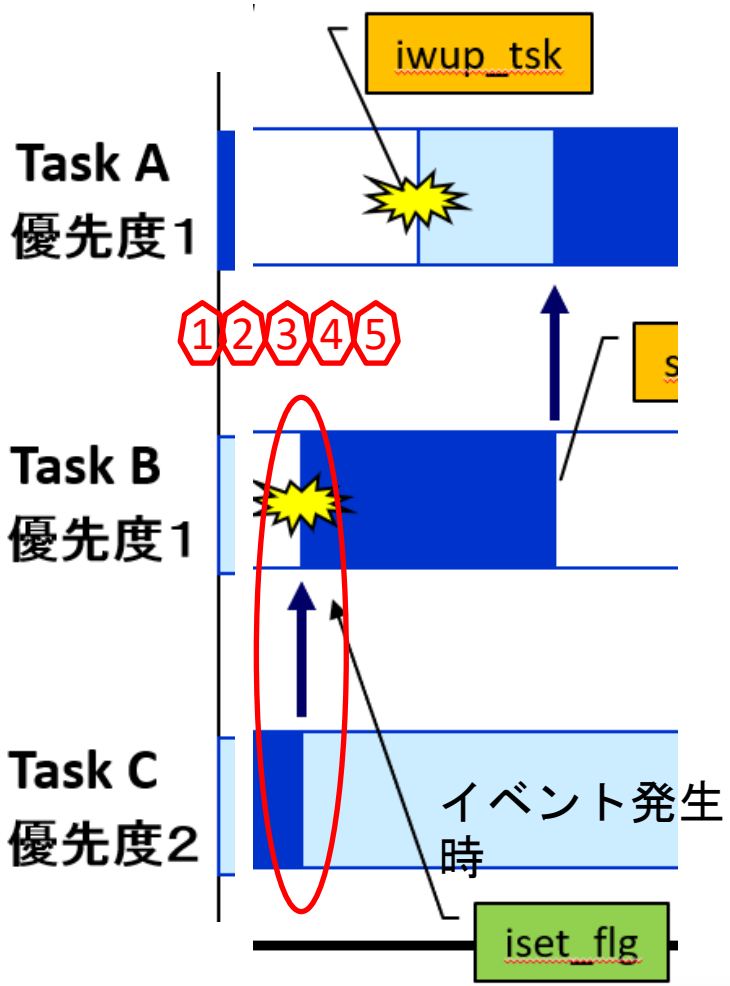
～タスクスケジューリングと状態遷移～

ディスパッチ:
プロセッサが、実行している
タスクの切り替えを行うこと



実行状態　実行可能状態　待ち状態

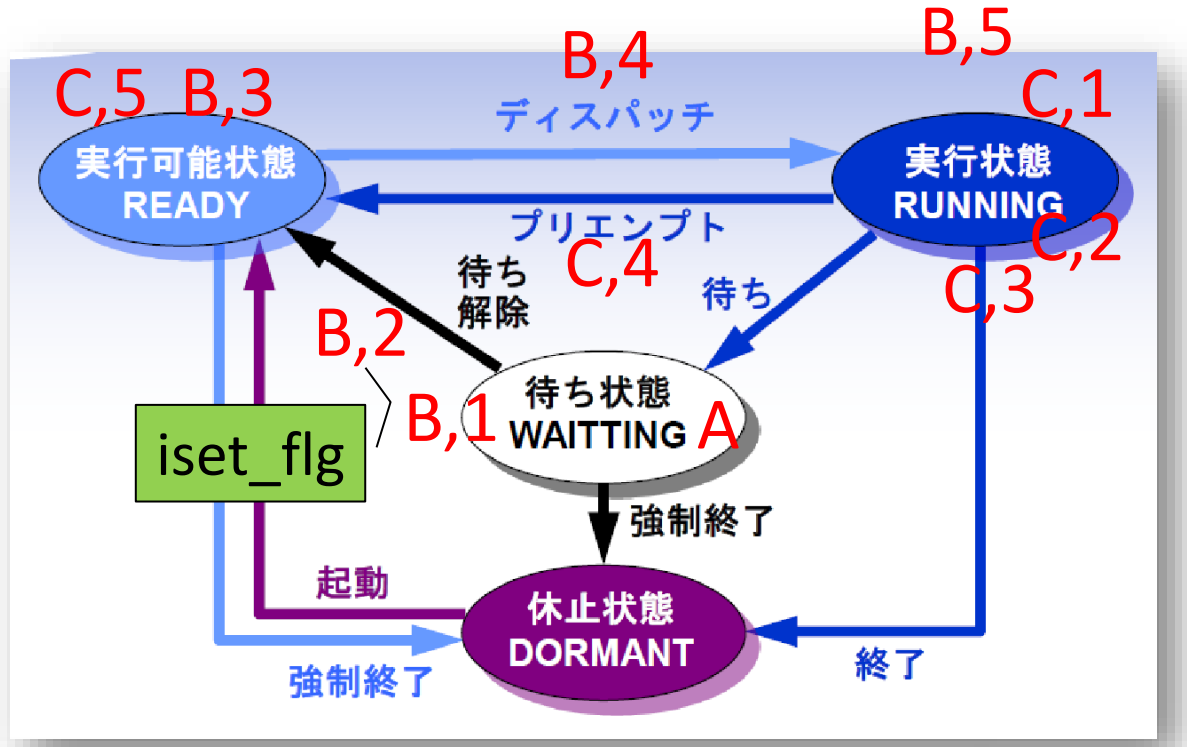
タスクスケジューリング、詳細説明



～タスクスケジューリングと状態遷移～

- 不均衡: B,3 > C,3
- 均衡　: C,5 < B,5

「プリエンプト」:「先取りする、差し替える」の意



実行状態　実行可能状態　待ち状態

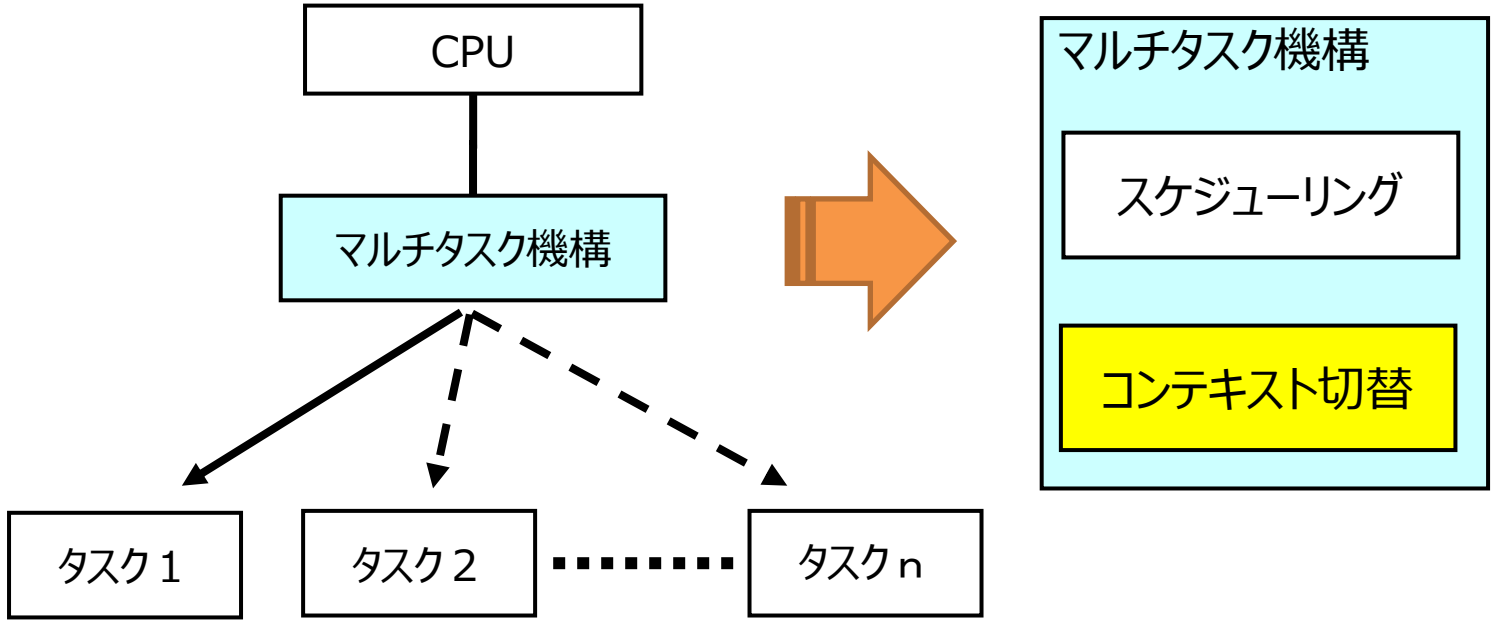
今日のプログラム

- 会社紹介とセミナ紹介
- 1. リアルタイム処理とは何か
- 2. RTOSを利用してアプリケーション構築するには
- 3. μITRONの概要と、メリットなどを紹介
- 4. マルチタスク機構の解説
  - 4-1. タスクのスケジューリング
  - 4-2. コンテキスト切替
- 5. eForce社製「μC3 (RTOS)」 の紹介
  - 5-1. μITRON4.0仕様準拠のRTOS
  - 5-2. μC3(RTOS) と μNet3(TCP/IP)

eForce

マルチタスク機構

- マルチタスク機構
  - CPUでは、ある時に、実行できるタスクはひとつのみ
  - それを時分割で制御して、複数のタスクを擬似的に平行実行させるための機構



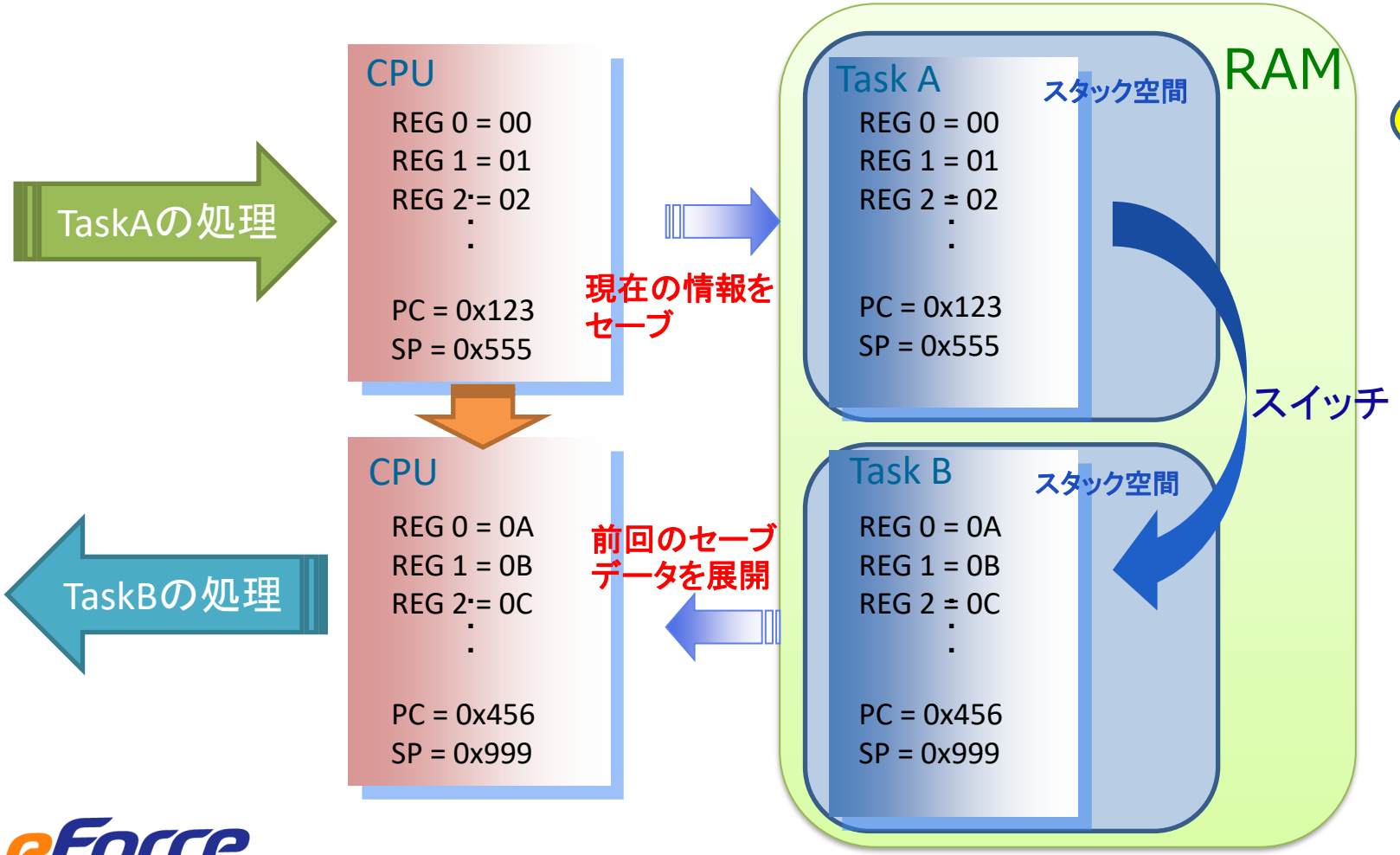
ここで、紹介します。

- スケジューリング
  タスクの実行を制御
- コンテキスト切替
  タスクの切替を制御

eForce

コンテキストの切替、概要

- タスク内のアプリケーションは、タスクが切り替わったことを意識せずに、処理を記述する。



RTOSの仕組み

- コンテキスト(Context)
  - [タスク]、[タイムイベントハンドラ]、[割込みハンドラ]で、それぞれ持つ
  - コンテキストの切り替え・再開のために、必要なデータを退避・復元する

スタック領域を準備する

- RTOSでは、スタック領域を準備する必要がある。
- ユーザが設計して、領域のサイズを指定する。

スタックとは？

ローカル変数で使用(大きな配列などは要注意)

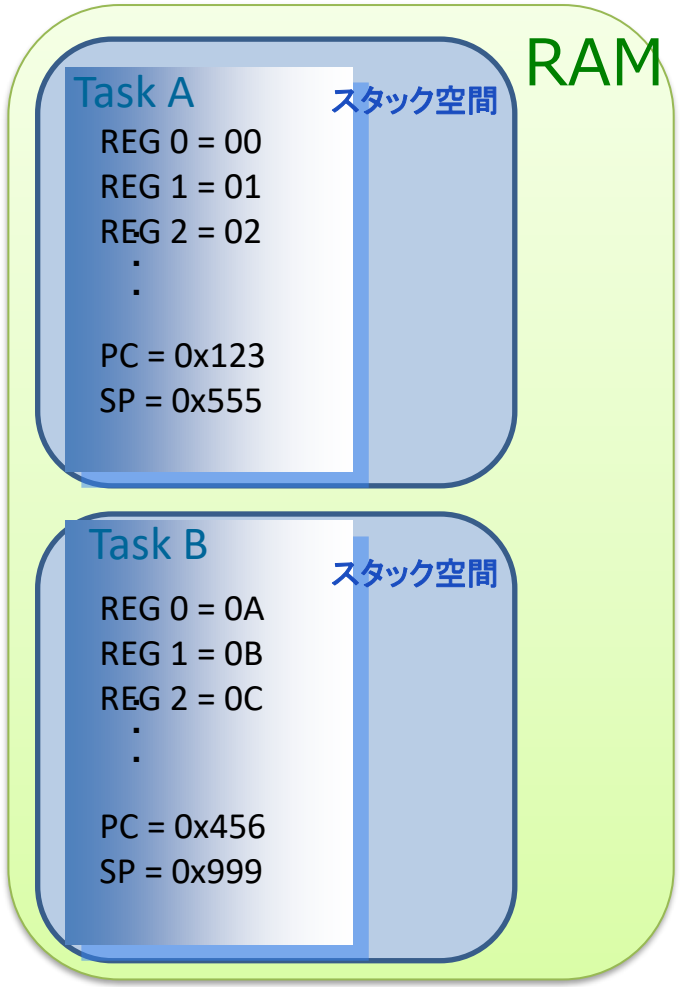
コンテキストの切替の際に、レジスタの値を退避する

関数の呼び出しがネストすると使われる場合もある

※注意※

スタックがあふれると

- 書き換えてはいけないメモリを書き換えてしまう
  - プログラムが暴走する
  - 他のタスクの変数を書き換える可能性も
  - 原因究明が難しい
  - スタックサイズを大きくして解決することもある



タスク間協調の紹介

- タスク間協調の必要性
  - タスク単位では、独立した処理を実行
  - 複数のタスクが連携することでシステムを構成している

タスク間の協調が必要

- タスク間協調の種類

種類	特徴	μITRONによる手段（例）
排他	資源待ち合わせ	セマフォ、（ミューテックス） など
同期	条件待ち合わせ	イベントフラグ
通信	タスク間でのデータ交換	データキュー、メールボックス、（メッセージバッファ）

eForce

今日のプログラム

- 会社紹介とセミナ紹介
- 1. リアルタイム処理とは何か
- 2. RTOSを利用してアプリケーション構築するには
- 3. μITRONの概要と、メリットなどを紹介
- 4. マルチタスク機構の解説
  - 4-1. タスクのスケジューリング
  - 4-2. コンテキスト切替
- 5. eForce社製「μC3 (RTOS)」 の紹介
  - 5-1. μITRON4.0仕様準拠のRTOS
  - 5-2. μC3(RTOS) と μNet3(TCP/IP)

eForce

μITRON4.0仕様

仕様

◆ μITRON4.0仕様準拠のRTOS

優先度ベースのRTOS

タスク単位で優先度を設定し、RTOSは、優先度の高いタスクから、実行します。

FCFSでスケジューリング

最初に実行可能状態に入ったタスクから実行(FCFS方式)します。

eForce

データ型

μITRONで使用するデータ型

B	符号付き8ビット整数
H	符号付き16ビット整数
W	符号付き32ビット整数
UB	符号無し8ビット整数
UH	符号無し16ビット整数
UW	符号無し32ビット整数
VB	データタイプが定まらない8ビットの値
VH	データタイプが定まらない16ビットの値
VW	データタイプが定まらない32ビットの値
VP	データタイプが定まらないものへのポインタ
FP	プログラムの起動番地（ポインタ）
INT	プロセッサに自然なサイズの符号付き整数
UINT	プロセッサに自然なサイズの符号無し整数
BOOL	真偽値（TRUEまたはFALSE）
FN	機能コード（符号付き整数）
ER	エラーコード（符号付き整数）
ID	オブジェクトのID番号（符号付き整数）

ATR	オブジェクト属性（符号無し整数）
STAT	オブジェクトの状態（符号無し整数）
MODE	システムコールの動作モード（符号無し整数）
PRI	優先度（符号付き整数）
SIZE	メモリ領域のサイズ（符号無し整数）
TMO	タイムアウト指定（符号付き整数，時間単位は実装定義）
RELTIM	相対時間（符号無し整数，時間単位は実装定義）
SYSTEM	システム時刻（符号無し整数，時間単位は実装定義）
VP_INT	データタイプが定まらないものへのポインタまたはプロセッサに自然なサイズの符号付き整数
ER_BOOL	エラーコードまたは真偽値
ER_ID	エラーコードまたはID番号（負のID番号は表現できない）
ER_UINT	エラーコードまたは符号無し整数（符号無し整数の有効ビット数はUINTより1ビット短い）
FLGPTN	イベントフラグのビットパターン（符号無し整数）
T_MSG	メールボックスへのメッセージヘッダ
INTNO	割込み番号
IMASK	割込みマスク

eForce

デモでの活用

通信

μC3で迷路脱出

ロボットカーのソフトウェア構成

メインタスク	起動処理 スタート待ち処理
センサタスク	センサから、走行状態を決定
モータタスク	ロボットカーの走行を制御
通信タスク	コントローラと通信処理
LEDタスク	状態をLED表示

コントローラのソフトウェア構成

表示タスク	画面表示を制御
通信タスク	ロボットカーと通信処理

タスク分割による保守性と開発効率の向上

- 機能毎
- 時間管理
- 同期

メリット1 機能毎に開発ができる マルチタスクOSだからこそ、次のことが可能となった。

- 例1 センサ部 センサを4系統から6系統に増やす改善を行えた。
- 例2 モータ部 迷路走破の改善を独立して行えた。
- 例3 通信部 テストから最終段階に向けて容易に通信内容を変更できた。

メリット2 時間の管理が容易にできる

- 例1 センサ値を周期的に取得ができた。
- 例2 センサ判定状態を、判定中から確定に移行するなどの時間待ちが容易に処理できた。

メリット3 タスク間の同期がとれる

- 例1 モータタスクから、制御終了通知などを、他のタスクに実施できた。

eForce

今日のプログラム

- 会社紹介とセミナ紹介
- 1. リアルタイム処理とは何か
- 2. RTOSを利用してアプリケーション構築するには
- 3. μITRONの概要と、メリットなどを紹介
- 4. マルチタスク機構の解説
  - 4-1. タスクのスケジューリング
  - 4-2. コンテキスト切替
- 5. eForce社製「μC3 (RTOS)」 の紹介
  - 5-1. μITRON4.0仕様準拠のRTOS
  - 5-2. μC3(RTOS) と μNet3(TCP/IP)

eForce

µC3のラインアップ

	µC3 micro c cube Compact	µC3 micro c cube Standard	µC3 micro c cube Standard+M	µC3 micro c cube +Linux
仕様	**µITRON4.0** 自動車制御プロファイル	**µITRON4.0** スタンダードプロファイル		
追加仕様	-	-	マルチコア対応	OpenAMP対応
特長	極小フットプリント	高い割込み応答性能	AMP型のマルチコア対応	Linuxとの共存が可能
対応CPUコア	arm ・Cortex®-M0/M0+/M3/M4/M7/M33 Renesas ・RX intel ・Nios Ⅱ	arm ・Cortex®-M4/M7/M33, A5/A7/A8/A9/A15/A53/A73, R4/R5 Renesas ・Renesas RX 700	arm ・Arm Cortex®-A7/A9/A15/A53	arm ・Cortex®-A53/A53 ・Cortex®-A53/R5 ・Cortex®-A53/M4 ・Cortex®-A9/A9 ・Cortex®-A7/A7 ・Cortex®-A7/M4
主な 対応デバイス （詳しくは製品ガイドを参照下さい）	STMicro STM32 NXP Kinetis, LPCxxx Renesas RA, RX200/600/700 Intel Nios Ⅱ Cypress PSoC Sililabs EFR32 Xilinx Microblaze（planning）	Renesas RZ/A, RZ/T, RZ/N STMicro STM32 NXP i.MX RT, i.MX6/7/8 TI AM335x, 57x, 65x Intel SoC Xilinx Zynq-7000, MPSoC	Renesas RZ/G1E, G1N, G1M TI AM57x NXP i.MX6/7, LS1043A Xilixn Zynq-7000, MPSoC Intel SoC	Renesas RZ/G1E NXP i.MX6/7/8M Mini STMicro STM32MP157 TI AM65x Xilinx Zynq-7000, MPSoC Intel SoC

eForce

µC3/Compact（ワンチップマイコン向けRTOS）

µC3 micro c cube Compact **はµITRON4.0 自動車制御プロファイルを採用**

オーバーヘッドやメモリ使用量の削減を目的とした機能セット

～　他社製品にはないの３つの特徴　～

1．コンフィグレータによる初期設定・コードの自動生成

・**GUIツール**により直接のコーディングが不要
・コーディングミスを防いで**ロケットスタート**

開発期間短縮 品質向上

2．最小2.4Kバイトの極小フットプリント

・ROM/RAMを極限までダウンサイジング
・FreeRTOSと比べて、RAM使用量は約3割減

BOMコスト減

3．業界トップクラスのサポートＣＰＵ

・Arm Cortex®-M、Renesas RX、
Intel Nios Ⅱ等をベースにした国内外**50種類以上**の
マイコンシリーズに対応

実績 No.1

・スケジュール通りに開発を行いたい
（EETimes 2019 Embedded Markets Studyにおける、開発時のお悩みNo.1）
・ライフサイクルコストを抑えたい

市場ニーズ

eForce が提供する

・コンフィグレータ（無償）
・OSプラグイン（無償）
・1営業日以内のサポート（保守）

を活用する事によって
開発期間の短縮や
ライフサイクルコストの低減が可能

お客様の声

・OSが原因のバグはなかった
・遅延なく製品を出荷できた
・コーディング時間を最大20%削減
・トラブルフリーです

ユーザ事例集はコチラ
https://www.eforce.co.jp/case/

〇こんな心配をされている方には
「無償評価版」や「ハンズオンセミナー」がおススメ

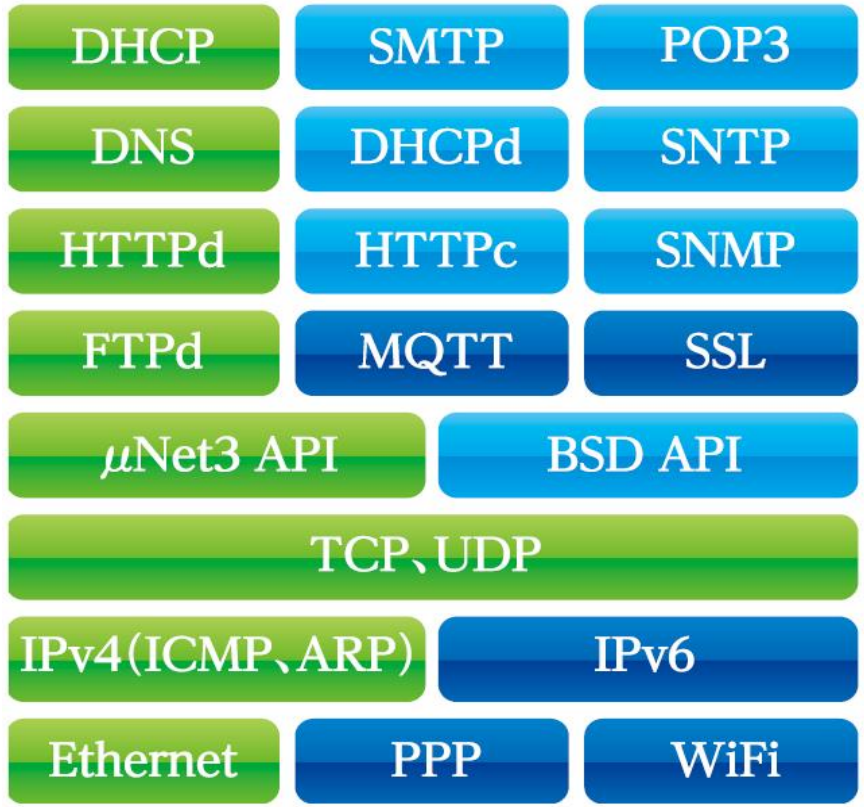
☑RTOSの基礎を学びたい
☑µC3の開発イメージを掴みたい
☑費用を掛けずに評価したい

無償評価版や ハンズオンセミナーを ご活用下さい

eForce

µNet3（TCP/IPスタック）

プロトコル構成



対応プロトコル一覧は
HP(下記URL)を参照下さい
https://www.eforce.co.jp/unet3/

他社製品にはない µNet3 の３つの特徴

1．コンフィグレータによる初期設定・コードの自動生成

・**GUIツール**により直接のコーディングが不要
・コーディングミスを防いで**ロケットスタート**

2．市場実証済みSDK

・WLAN-SDKや産業用イーサネットSDKなど、
市場実証済みSDKの提供が可能

**開発期間短縮
品質向上**

3．暗号エンジンのサポート

・All SWでの暗号化通信の実現はもちろん、
プロセッサが内蔵している
暗号エンジンにも対応

CPU負荷低減

こんな心配をされている方には無償評価版や製品の無償貸出がおススメ

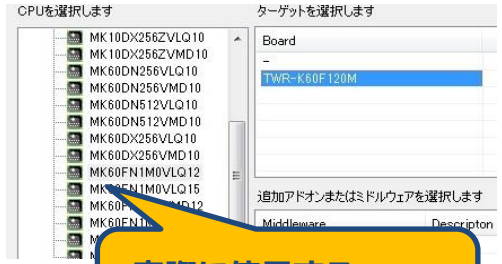
☑TCP/IPスタックの勉強をしてみたい
☑µNet3の開発イメージを掴みたい
☑費用を掛けずに評価したい

**無償評価版や貸出は
info@eforce.co.jpまで**

eForce

μC3/ConfiguratorによってOS + TCP/IPを簡単設定！　主に、μC3/Compact向け

①CPU型番選択



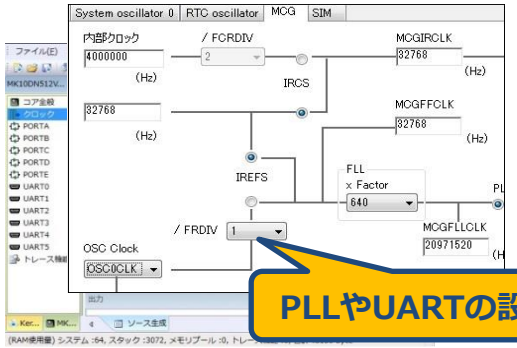
実際に使用する
デバイスの型番を選択

②カーネルの設定

RAMの使用量が
一目瞭然👀

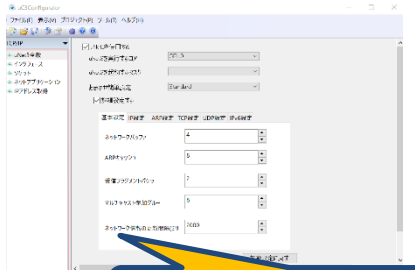
(RAM使用量) システム :174, スタック :4352, メモリプール :0, トレース機能 :0, 合計 :4526 byte

③内蔵ペリフェラルの設定



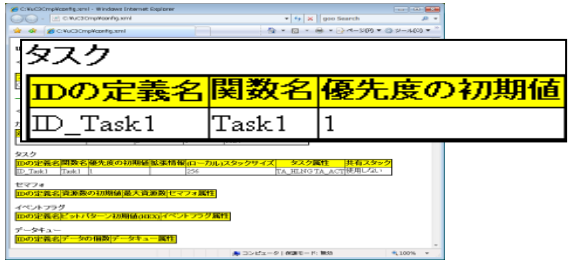
PLLやUARTの設定

④TCP/IPの設定



IPアドレス等の設定が可能

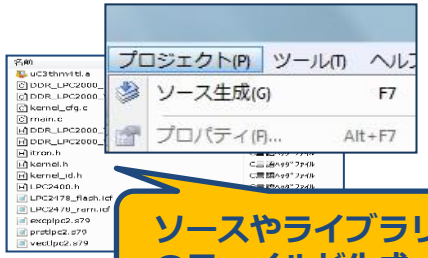
⑤ブラウザで設定内容をチェック



タスク

IDの定義名	関数名	優先度の初期値
ID_Task1	Task1	1

⑥ファイルの生成



ソースやライブラリ
のファイルが生成

⑦コードの自動生成

```
/*********************************
        Task1
 *********************************/
void Task1(VP_INT exinf)
{
}
```

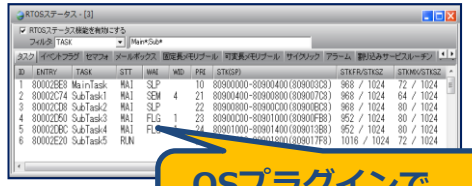
タスクが起動するまでの
コードが自動生成

⑧Appの記述

```
/*********************************
        Task1
 *********************************/
void Task1(VP_INT exinf)
{
    for (;;) {
        REG_FIO1.LONG.SET = 0x00002000;
        REG_FIO1.LONG.CLR = 0x00040000;
        dly_tsk(1000);
        REG_FIO1.LONG.CLR = 0x00002000;
        REG_FIO1.LONG.SET = 0x00040000;
        dly_tsk(1000);
    }
}
```

⑨Build & Download

⑩Debug



OSプラグインで
リソース情報を可視化

μC3/Configurator → 無償バンドル
OSプラグイン → 無償ダウンロード
https://www.eforce.co.jp/download#dl02

eForce

まとめ

- RTOSを利用した開発
  - タスクを利用した開発
  - リアルタイム処理の開発
  - 保守性・開発効率の向上、ライフサイクルコストの低減

- RTOS上での開発に必要となる知識
  - タスクは、優先度をもとにしたスケジューリング
  - タスク利用時には、待ち状態への遷移や協調処理が必要

- eForce社製「μC3 (RTOS)」 の概要
  - 幅広い用途で採用実績のある「μITRON4.0仕様準拠」
  - 開発期間の短縮を実現する1chipマイコン向けのRTOS「μC3/Compact」
  - マルチコア対応やLinuxとの共存など幅広い対応が特徴の「μC3/Standard」
  - 開発期間短縮の実現や様々なSDKが用意された「μNet3」

eForce

Q&A

ご視聴有り難う御座いました

- 時間内に回答できなかったご質問に関しましては、
  後日、アンケートと共に回答を送付させて頂きます。

- アンケートに回答頂けた方には、
  本日のセミナー資料のダウンロードURLを送付させて頂きます。

本日のご参加、誠に有り難う御座いました。

ご清聴ありがとうございました

eForce