ASP3 に標準搭載された組込み コンポーネントシステム TECS の紹介

TOPPERS/ASP3 meets TECS TOPPERS Embedded Component System

名古屋大学大学院 情報科学研究科
河田 智明

はじめに – TECSの紹介

- TOPPERS/ASP3が今年2月にリリース
  - ティックレス時間管理
  - 実装の単純化 – タスク例外処理機能の削除
  - **TECSの標準搭載＆システムサービスのTECS化**

TOPPERS Embedded Component System (TECS)

- 組込み向けコンポーネントシステム
  - コンポーネントベース開発による**ソフトウェア再利用性・生産性**の改善
  - **低オーバヘッド** – コンポーネントの静的生成・結合
  - RPCインタフェースコード，トレース用コード等の自動生成
  - **カーネルの機能**をコンポーネントとして**利用可能**

本チュートリアルについて

- 本チュートリアルでは**TECSの基本的な使い方**から，ASP3+TECSを用いた**アプリケーション開発の方法**まで**実演を交えて**説明します
- 一般的なリアルタイムカーネルの基本的な使用法について理解されていることを前提として説明していきます

アジェンダ

1. TECS
   TECSの紹介

2. ASP3+TECS
   TOPPERS/ASP3の機能をTECSから利用する仕組みについて

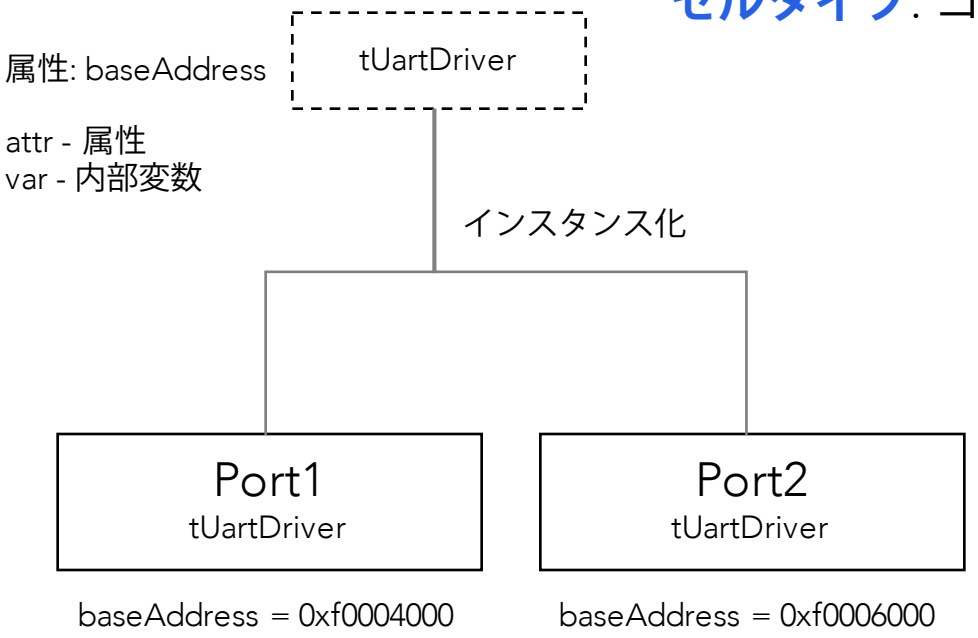
3. ASP3+TECS によるアプリケーション開発
   TECSアプリケーションの作成とシステムサービスのコンフィギュレーション方法について

1 TECS

TOPPERS Embedded Component System (TECS) の紹介

セル: TECSのコンポーネントの単位

セルタイプ: コンポーネントの型を表す.



tUartDriver.cdl (TECS CDL)

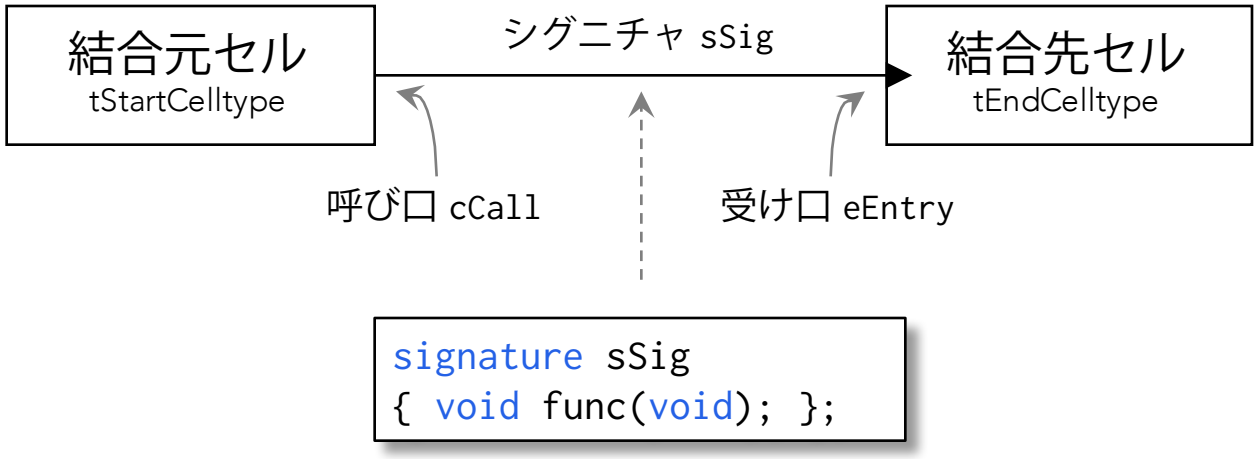
```
celltype tUartDriver {
  /* 属性の定義 */
  attr {
    void *baseAddress;
  };
};
```

```
cell tUartDriver Port1 {
};
cell tUartDriver Port2 {
  /* 属性値の指定 */
  baseAddress = ...
};
```

セルタイプを**インスタンス化**することにより,
セル (コンポーネントのインスタンス) が得られる

セルの結合

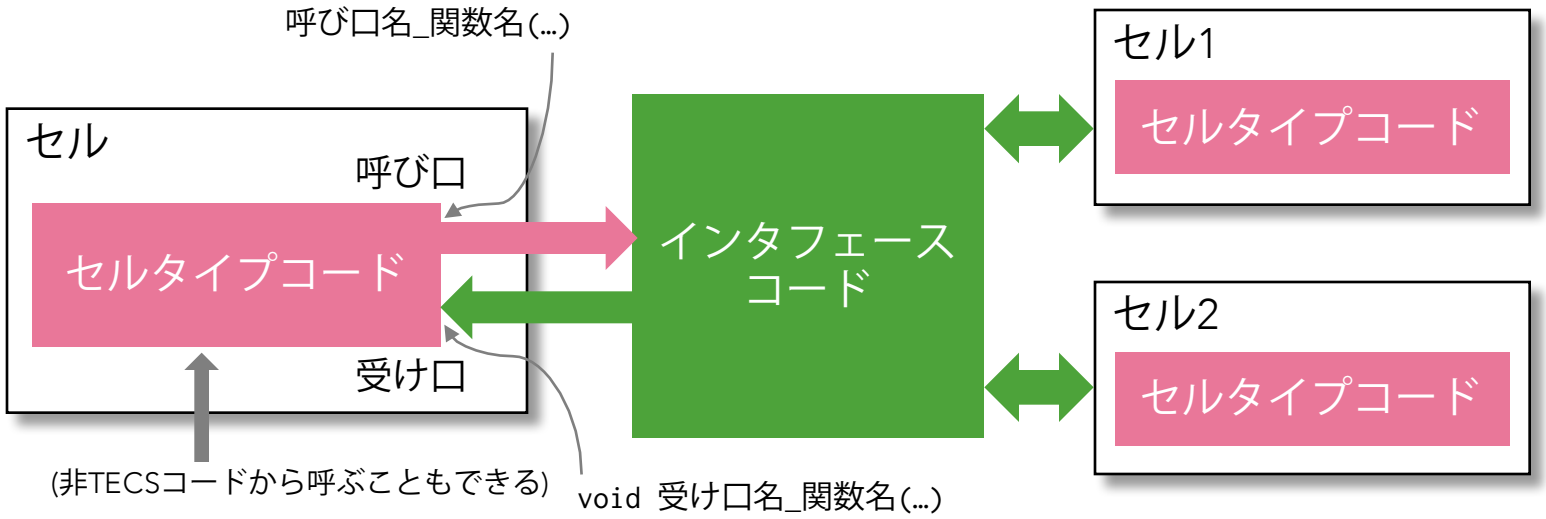
- **セル** (コンポーネント) の**呼び口**を他のセルの**受け口**に結合することで，結合先のセルの機能を利用することができる
- 受け口・呼び口のインタフェースはC言語の関数呼出しの集合であり，その型の定義を**シグニチャ**と呼ぶ



実演: セル・セルタイプの定義

- 既存のセルタイプのインスタンス化
- 独自のセルタイプの定義とインスタンス化
- 受け口/呼び口の定義と結合

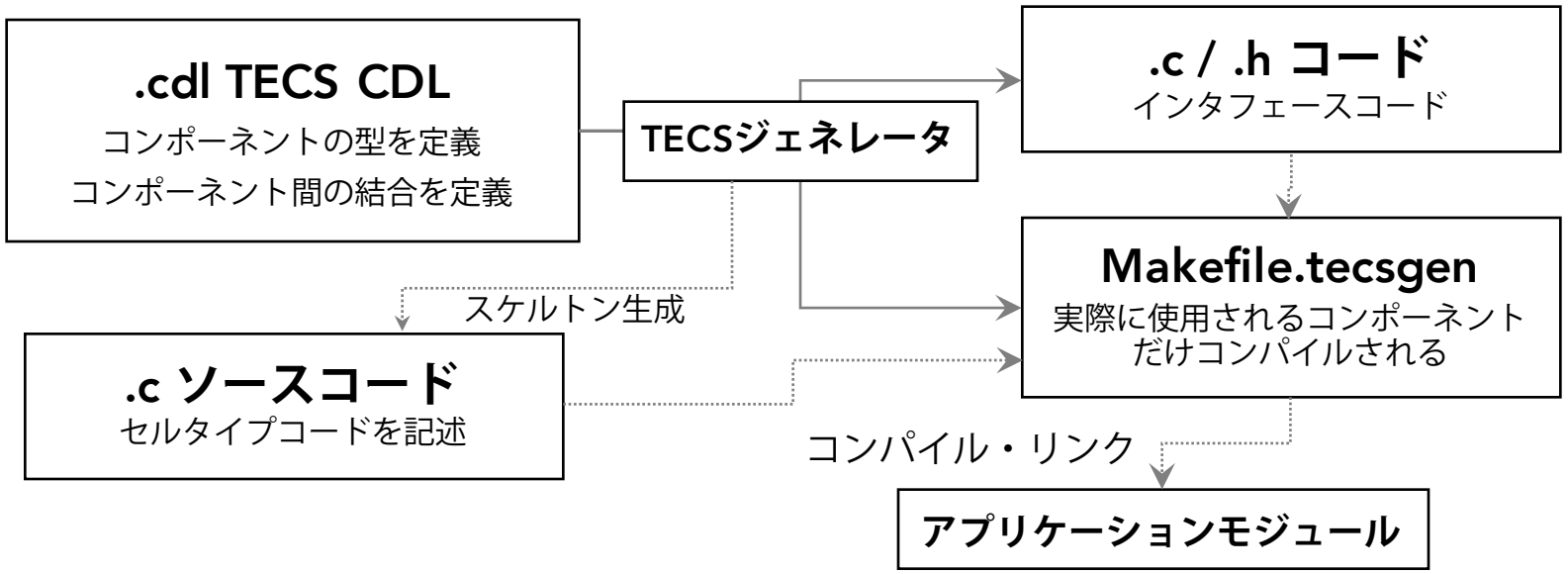
インタフェースコードの自動生成



- **セルタイプコード** ― コンポーネントの振舞いをユーザが記述
  - セルの受け口が呼ばれた時の動作を，受け口関数として記述
  - 非TECSコードからの呼出しを受け持つことも
- インタフェースコード ― TECS CDLの結合情報から**自動生成**

TECSの開発ワークフロー

- **TECS ジェネレータ** (tecsgen) により各種コードを自動生成
  - コンポーネント間のインタフェースコード
  - コンポーネント用の Makefile

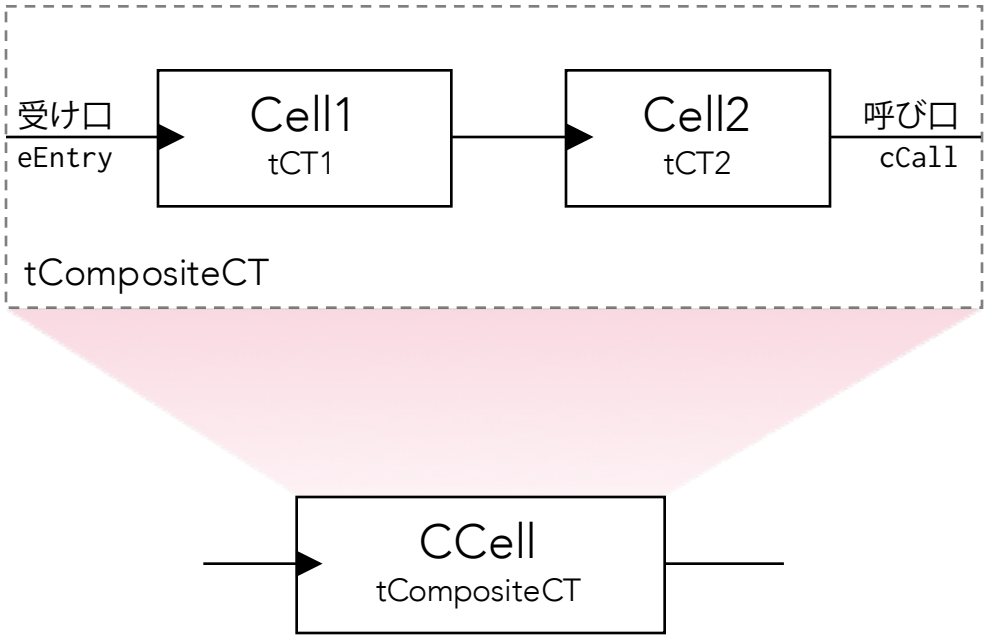


実演: ジェネレータの実行 ~ コンパイル

- TECS ジェネレータの実行
  - インタフェースコードの生成
  - セルタイプコードのスケルトン生成
- セルタイプコードの作成
- コンパイル
- 実行

複合セル: 入れ子構造を持つコンポーネントの定義

- **複合セル** — 複数のセルをまとめて一つのセルタイプとして定義する機能
  - 単純な機能を組み合わせて複雑なコンポーネントを階層的に実現できる




```
composite tCompositeCT {
  entry sSig eEntry;
  call sSig cCall;
  cell tCT1 Cell1 { ... };
  cell tCT2 Cell2 {
    cCall => composite.cCall;
  };
  eEntry => Cell1.eEntry;
};
```

```
cell tCompositeCT CCell {
  cCall = ...
};
```

利用例: 複雑なデバイスドライバを複合セルで作り，
デバイス毎にインスタンス化

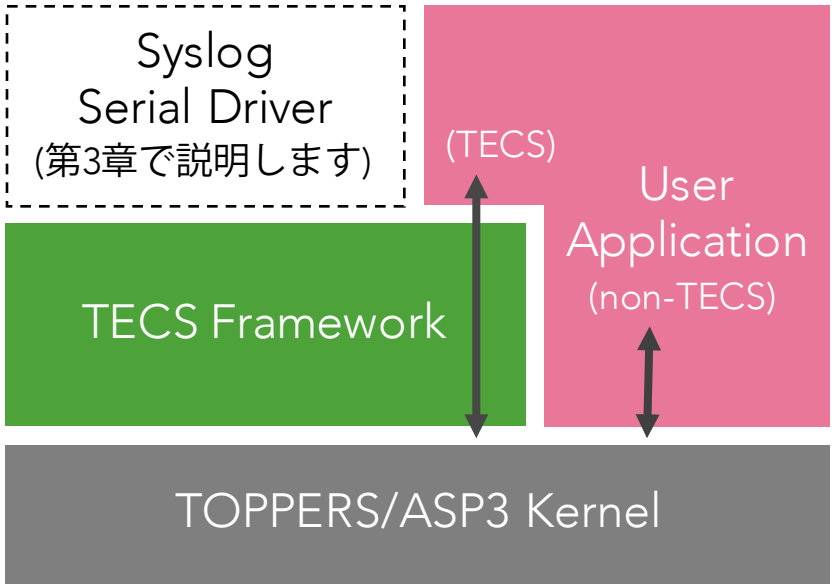
2 ASP3+TECS

TOPPERS/ASP3の機能をTECSから利用する仕組みについて

TOPPERS/ASP3+TECS

- ASP3が提供する**静的/実行時API**(サービルコール)の**ラッパー**として実現
- ASP3の機能の大部分を**コンポーネント**として提供
  - タスク
  - タスク間同期機能
  - タイムイベント通知
  - 割込みライン操作・ISR
  - CPU例外/初期化/終了ハンドラ

→ コンポーネントの一部としてカーネルオブジェクトを含められる



カーネル機能のコンポーネント化の仕組み

セルタイプ定義

```
celltype tTask {
  entry  sTask eTask;
  entry  siTask  eiTask;
  call   sTaskBody cTaskBody;
  /* 中略 */
  factory {
    write("tecsgen.cfg", "CRE_TSK(…)");
  };
};
```

ブリッジコード: 実行時APIを提供

```
start (intptr_t){
  /* 中略 */
  cTaskBody_start();
}
```

```
eTask_activate(...){
  return
    act_tsk(ATTR_id);
}
```

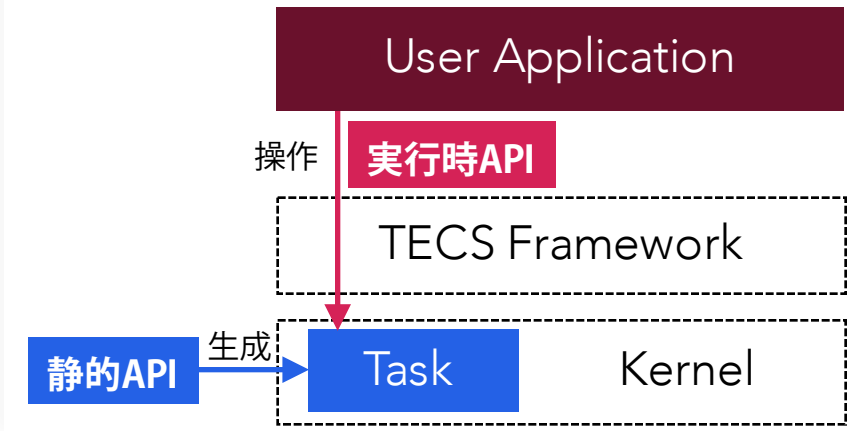
セル定義

```
cell tTask MainTask { /* 例 */
  cTaskBody = MyCell.eMainTaskBody;
  priority = 5;
};
```

静的API記述

```
CRE_TSK(TSKID_tTask_MainTask, { TA_NULL,
&tTask_CB_tab[1], start, MID_PRIORITY,
STACK_SIZE, NULL });
```

ジェネレータを通す



tTask: タスクコンポーネント

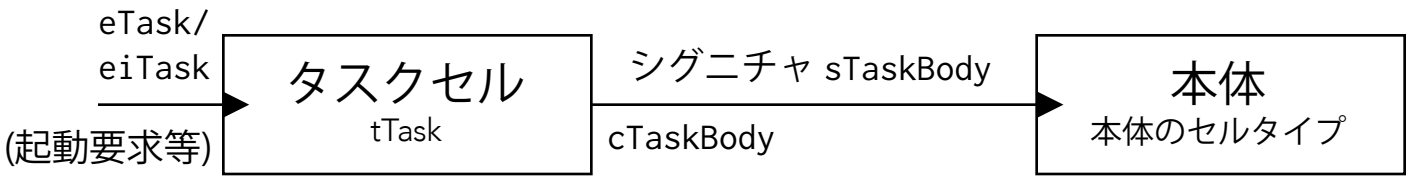
- 呼び口 cTaskBody がタスク本体として呼ばれる
- 受け口 eTask/eiTask から起動 (activate), 起床 (wakeup), 等の要求
- 属性 priority でタスク属性指定

```
celltype tTask {
  entry  sTask eTask;
  entry  siTask  eiTask;
  call   sTaskBody cTaskBody;
  attr {
    ATR    attribute = C_EXP("TA_NULL");
    PRI    priority;
    size_t stackSize; };
  /* 中略 */
};
```

```
cell tTask MainTask { /* 例 */
  cTaskBody = MyCell.eMainTaskBody;
  priority = 5;
};
```

出力:

```
CRE_TSK(/* 中略 */);
```



t(Cyclic|Alarm)Notifier: タイムイベント通知

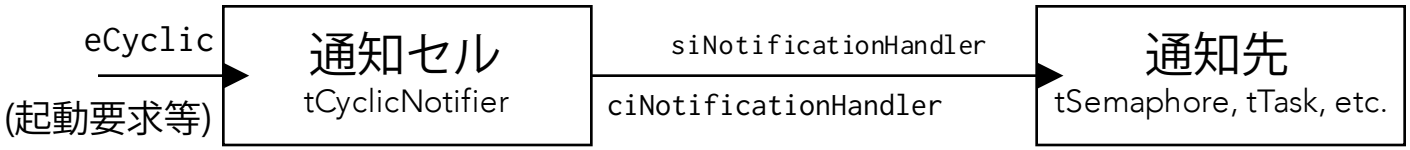
- ASP3で新しく追加されたAPI
  - **8種類の通知方法**: ハンドラ関数, 変数設定/インクリメント, タスク起動/起床, セマフォ返却, イベントフラグ設定, データキュー送信
- 仕様を静的APIに合わせる為に**変則的なコンポーネントモデル**を採用
  - 結合先によって通知方法を指定
  - 次のスライドで説明

```
celltype tCyclicNotifier {
  entry  sCyclic eCyclic;
  call   siNotificationHandler
    ciNotificationHandler;
  attr {
    ATR    attribute = C_EXP("TA_NULL");
    RELTIM cycleTime;
    RELTIM cyclePhase; };
  /* 中略 */
};
```

```
cell tCyclicNotifier Cyclic { /* 例 */
  ciNotificationHandler =
    Target.eiNotificationHandler;
};
```

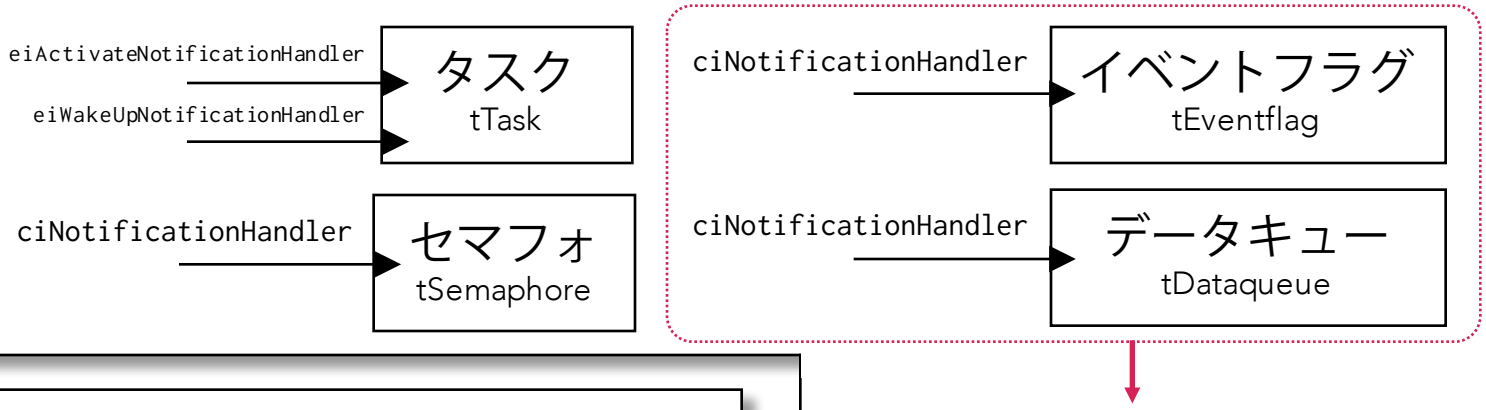
出力:

```
CRE_CYC(/* 中略 */, {..., Target});
```



タイムイベントの通知先の指定

タスク起動/起床, セマフォ返却, イベントフラグ設定, データキュー送信



変数設定/インクリメント

```
cell tCyclicNotifier Cyclic {
  setVariableAddress = C_EXP("&var");
  setVariableValue = 0xc0ffee;
};
```

```
cell tCyclicNotifier Cyclic {
  incrementedVariableAddress = ...;
};
```

属性により送信データ・フラグパターンを指定

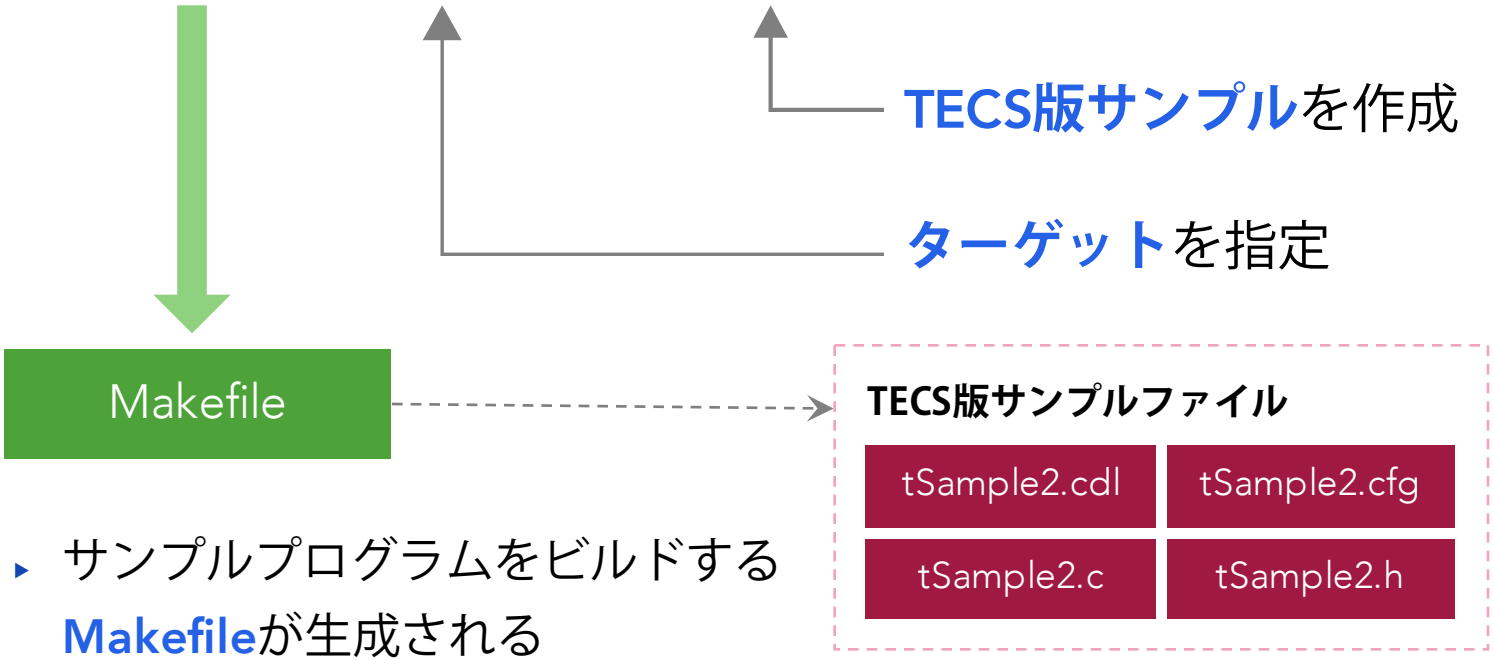
```
cell tCyclicNotifier Cyclic {
  dataqueueSentValue = 0xfaceb00c;
};
```

```
cell tCyclicNotifier Cyclic {
  flagPattern = 0x4d;
};
```


TOPPERS

サンプルプロジェクトの構築

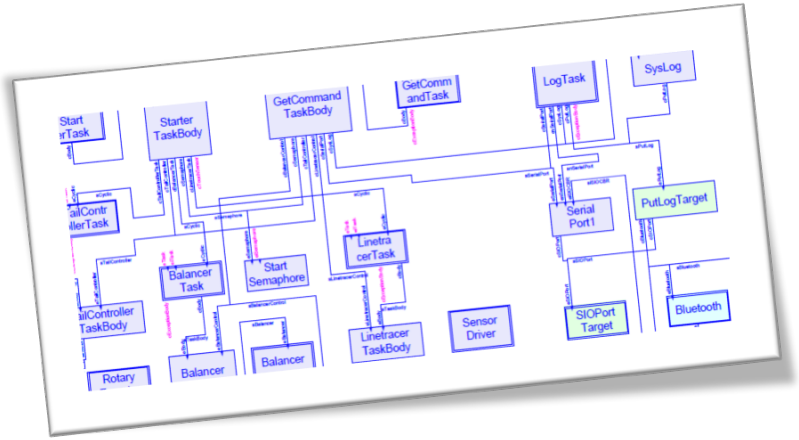
```
mkdir OBJ && cd OBJ
../configure.rb –T macosx_gcc -t
```



- サンプルプログラムをビルドする**Makefile**が生成される
- ソースファイルを適宜修正・移動して使用

実演: ASP3+TECS

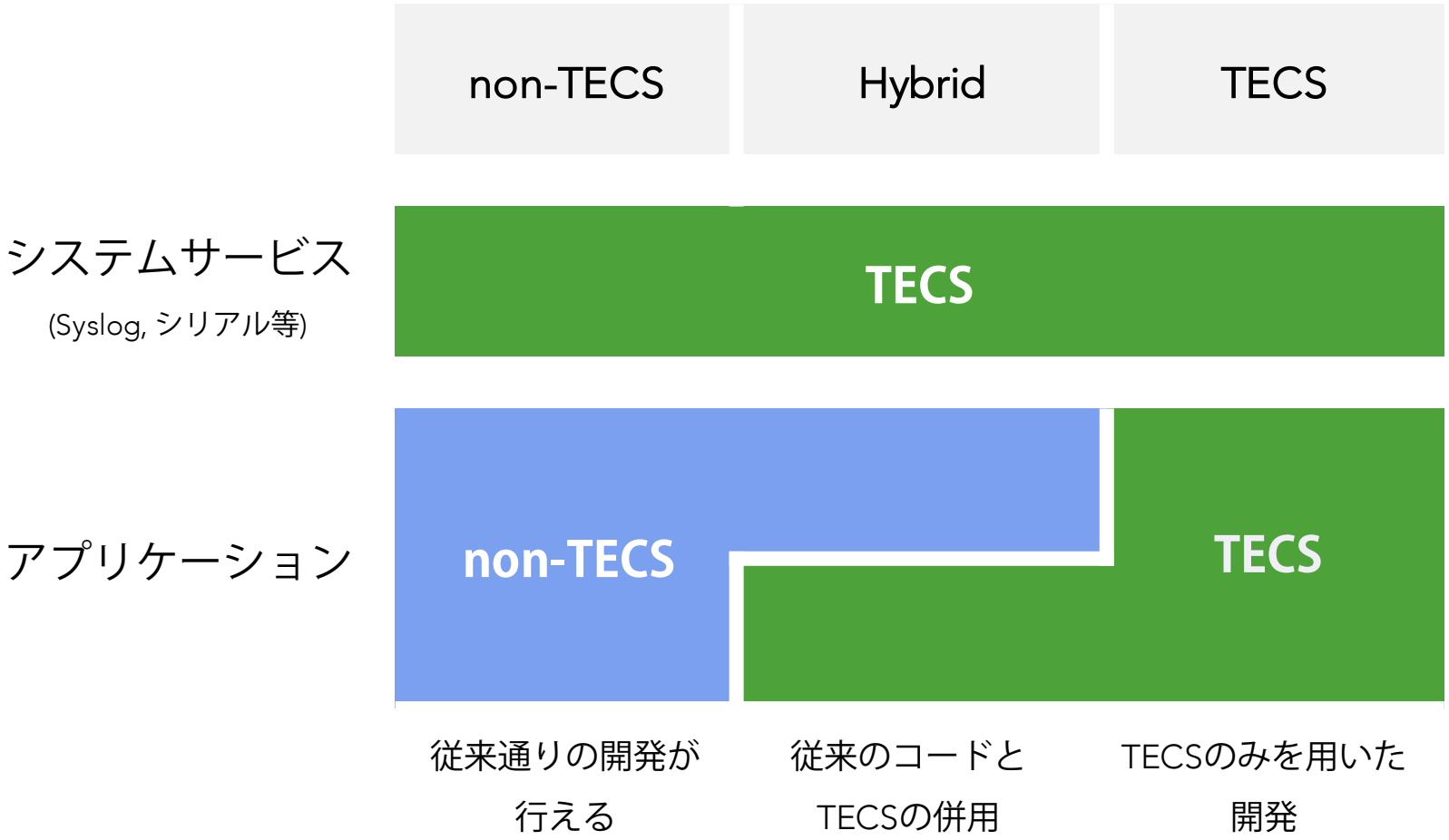
- ASP3+TECSサンプルプロジェクトの作成
- カーネルオブジェクトコンポーネントの生成
- カーネルオブジェクトの操作
- 生成される静的APIを観察する
- 実際に動かしてみる



3 ASP3+TECS によるアプリケーション開発

TECSアプリケーションの作成とシステムサービスのコンフィギュレーション方法について

アプリケーションの開発スタイル

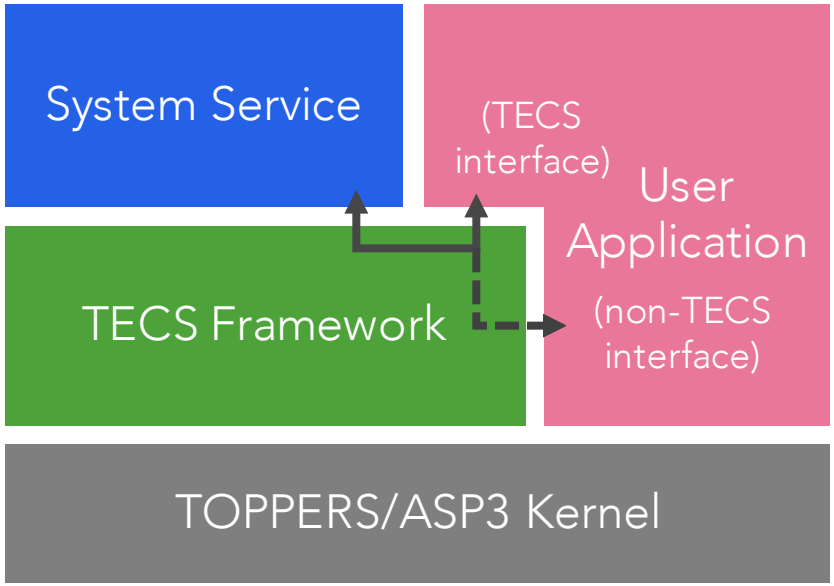


TOPPERS/ASP3 システムサービスのTECS化

システムサービス

- Syslog, シリアル入出力ドライバ, シリアルインタフェースドライバ
- 起動時メッセージ (バナー)

- ASP3ではシステムサービスを **TECSで実装**
- コンフィギュレーションは **TECS CDL**で行う
  - 不要な機能の無効化
  - Syslog出力先の切り替え等



SysLog API

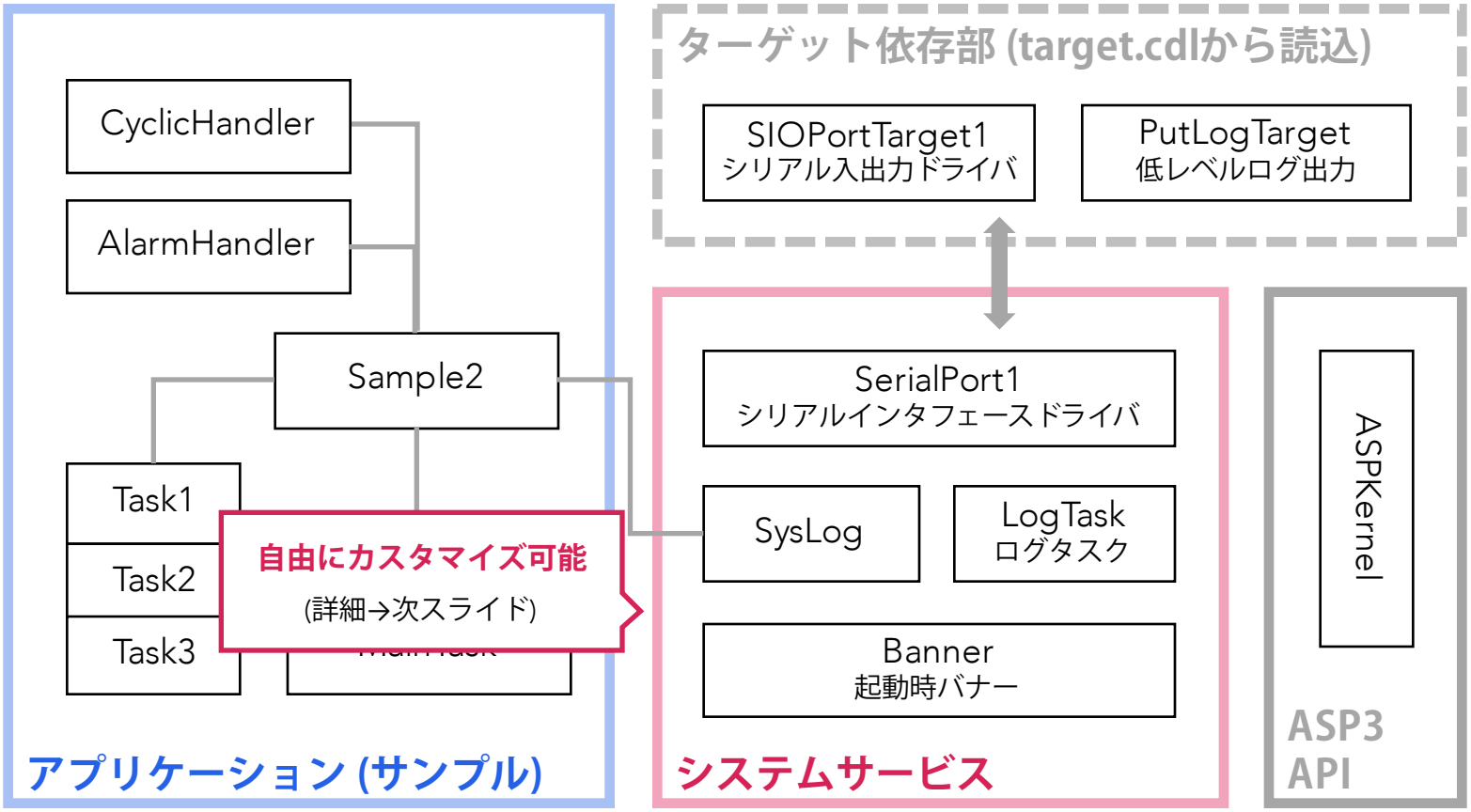
- 利便性・互換性を確保する為に，SysLogは**従来通りのC API**を用意
- 内部的にはSysLogセルの受け口を直接呼び出している

```
Inline void syslog_1(uint_t prio, uint_t type, LOGPAR arg1){
    SYSLOG logbuf;
    logbuf.logtype = type;
    logbuf.logpar[0] = arg1;
    (void) tSysLog_eSysLog_write(prio, &logbuf);
}
```

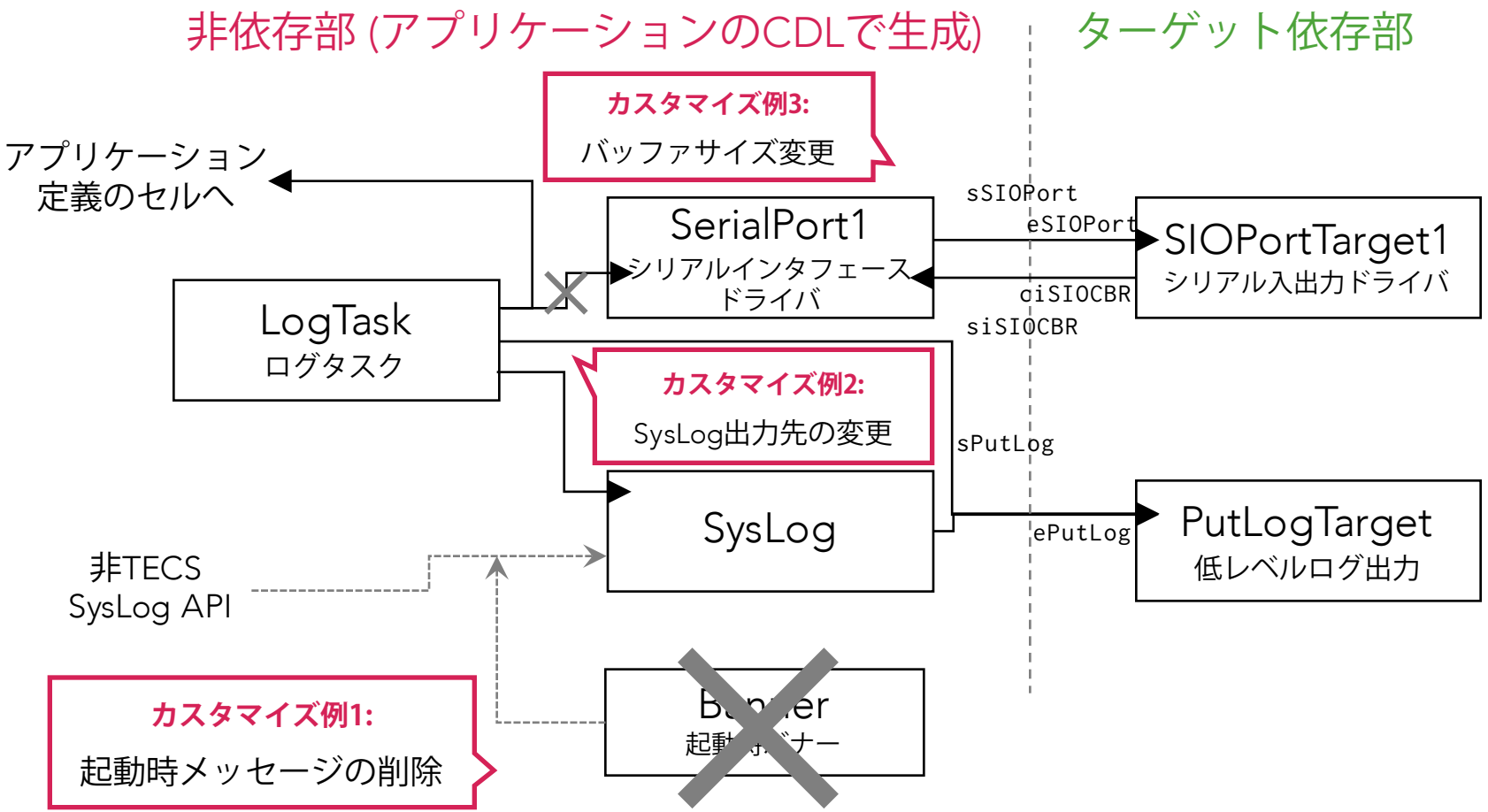
これ以外の C API を使用するには，tSysLogAdapterセルの生成が必要になります (user.txt 8.1.8).

```
void syslog(uint_t prio, const char *format, ...) {
    /* 中略 */
    (void) tSysLog_eSysLog_write(prio, &logbuf);
}
```

アプリケーション用CDLファイルの構造



システムサービスのカスタマイズ



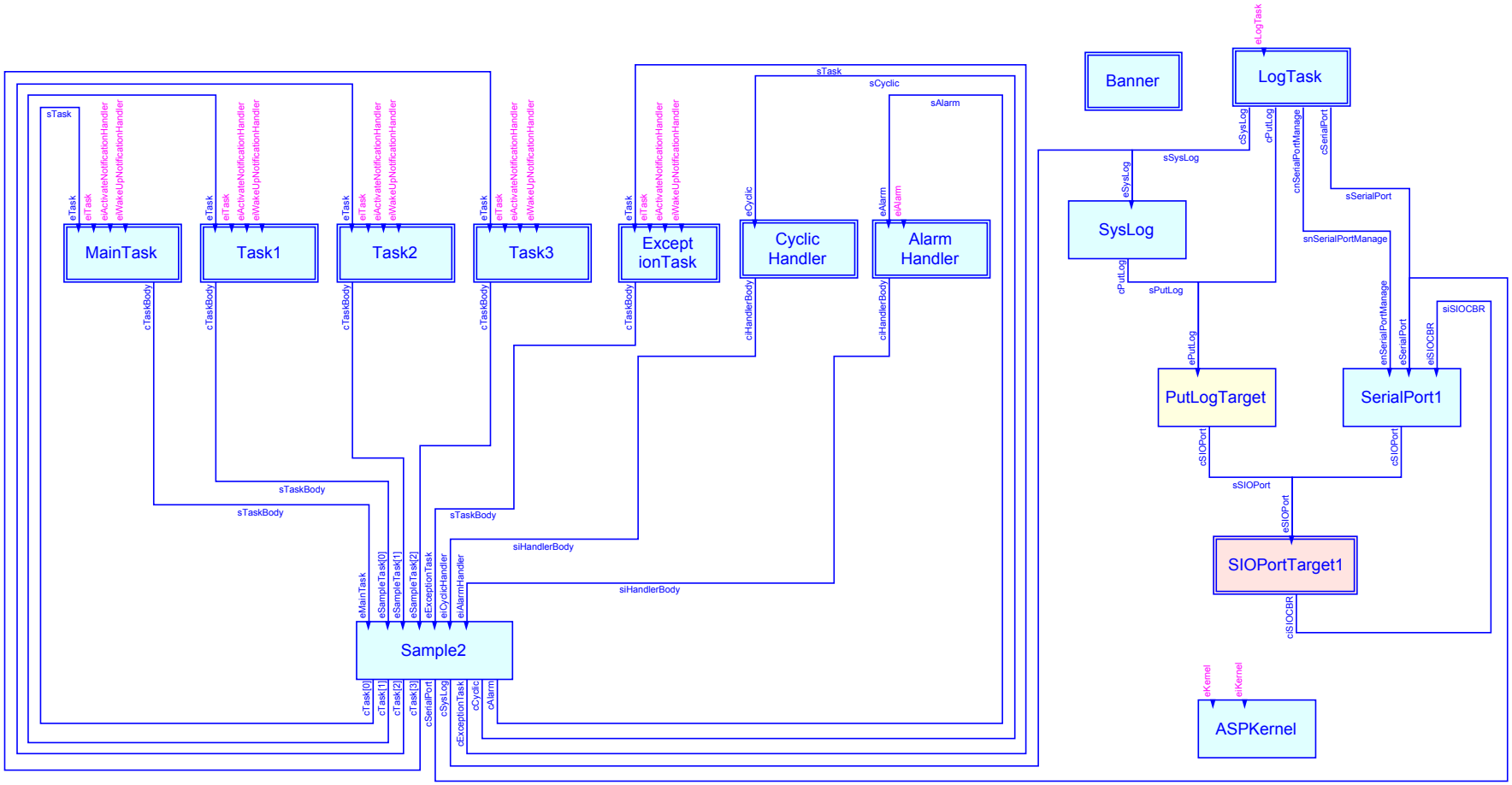
実演: システムサービスのカスタマイズ

- システムサービスのカスタマイズ
  - パラメータのカスタマイズ
  - 機能の無効化
- 実際に動かしてみる

まとめ

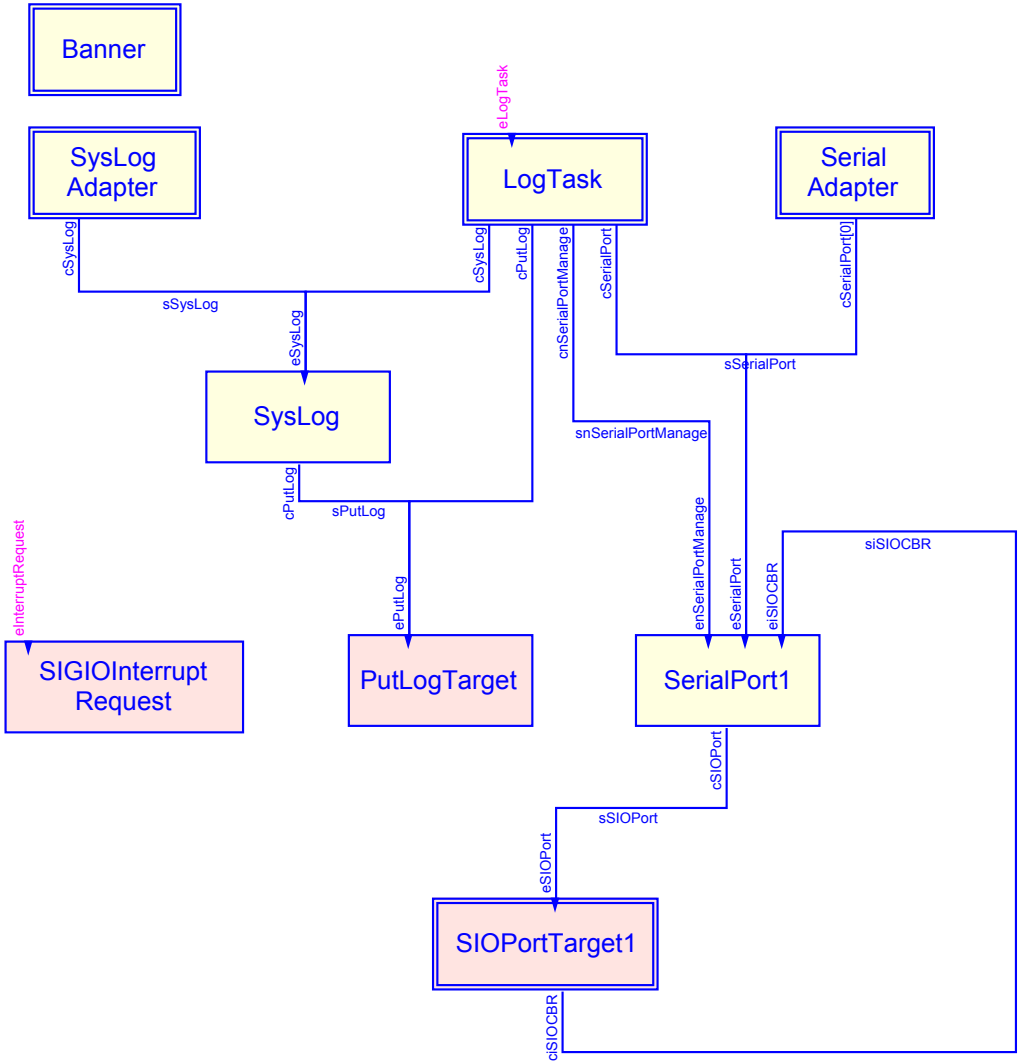
- 本チュートリアルでは**TECSの基本的な使い方**や，ASP3+TECSを用いた**アプリケーション開発の方法**について解説しました
- ASP3+TECSの使用法は付属のユーザマニュアル等で参照できます
  - システムサービス — doc/user.txt “8 システムサービス”
  - 各種セルタイプの定義については tecs_kernel/kernel.cdl を参照して下さい

付録 サンプルプロジェクトのコンポーネント図



TOPPERS

付録 OS X版ターゲット依存部のコンポーネント図



TOPPERS