# 組込みシステムに適した コンポーネントシステムTECSの最新状況 ～TOPPERS/ASP3標準搭載～


TOPPERS/ASP3 meets TECS TOPPERS Embedded Component System

安積 卓也
大阪大学大学院基礎工学研究科
takuya@sys.es.osaka-u.ac.jp

原稿作成協力：河田 智明 （名大）、大山 博司（TECS WG 主査 ）
引用：高田先生の資料（TOPPERSカンファレンス2016）

TOPPERS

# 目次

- TECSの基本
- TOPPERS/ASP3のSysLog, シリアルドライバ ログタスクの実装にTECSを標準採用
- TECSの利点
- その他の最新動向

TECSを使ってみませんか？

# TECS (TOPPERS Embedded Component System)

AUTOSARも
コンポーネントベース開発の一つ

- 組込みシステムでコンポーネントベース開発を実現
  - **再利用性**の向上→**生産性**の向上
  - インタフェースの明確な定義
    - C 言語のプロトタイプ宣言の曖昧さを TECS がカバー

  ->**さまざまなコードを自動生成可能**：例は後で説明

補足：
コンポーネント＝ソフトウェア部品

- その他の特徴
  - マルチインスタンス化が容易（デフォルト）
    - 同種のコンポーネントを複数生成
  - ダイナミックバインディング（相当）を実現
  - 関数テーブルを自動生成
  - カプセル化できる
    - 関数インタフェースのみで結合
  - 静的な生成と結合
    - 実行時オーバヘッド、メモリオーバヘッドの低減

ここを中心に説明



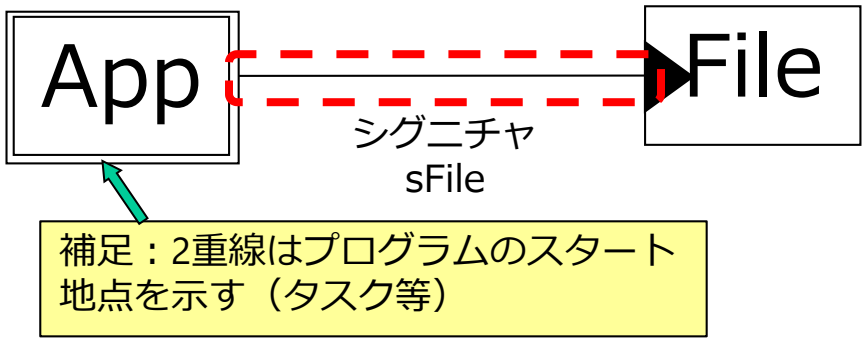
TOPPERS

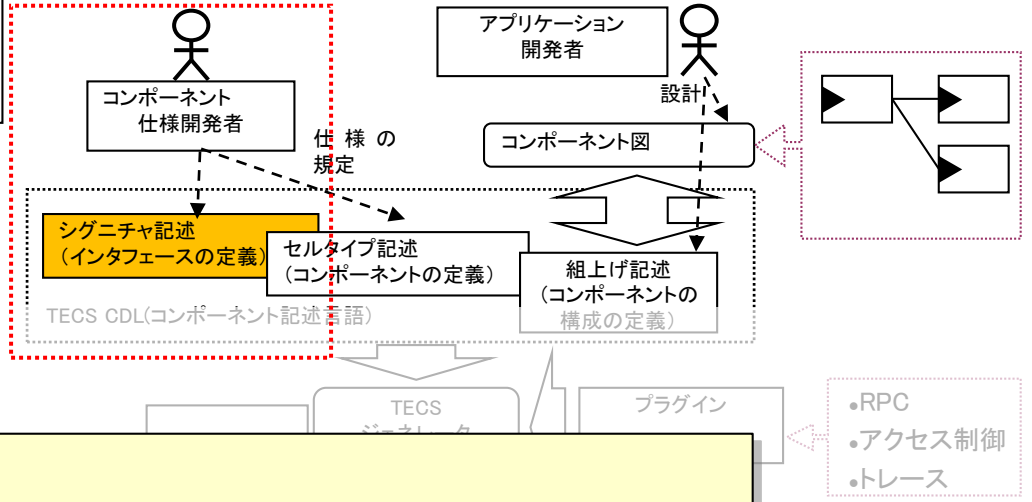
# TECS 開発の流れ

**これだけ知れば、始められる！**

- TECS CDL (コンポーネント記述言語)の記述
  - コンポーネント間のインタフェースの記述
    - シグニチャ (signature)記述
  - コンポーネントタイプの記述
    - セルタイプ (celltype) 記述
  - コンポーネントの設置と結合
    - 組上げ記述 (cell の記述)
      - ≒ コンポーネント図のテキスト表現
- C 言語の記述
  - 振る舞いの記述
    - セルタイプコード ＝ **C 言語**によるプログラム

# TECS CDL：インタフェースの記述（シグニチャ）




## シグニチャ記述

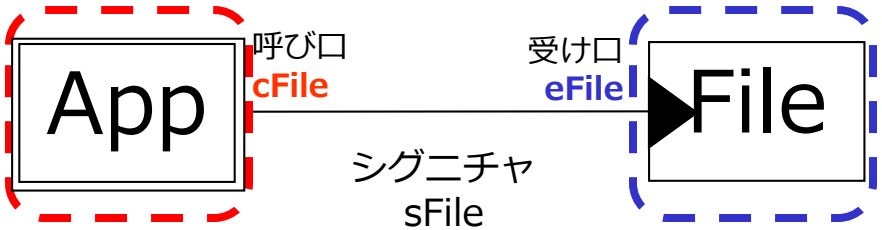
```
signature sFile {
  ER open( [in,string]char * fileName, [in]int16_t mode);
  ER close(void);
  ER read( [out,size_is(length),count_is(*count)]int8_t * buffer,
          [in]int32_t length, [out]int32_t *count);
  ER write( [in,size_is(length)]int8_t *buffer,
       [in]int32_t length, [out]int32_t *wroteLength);
  ER seek( [in]int32_t offset);
};
```

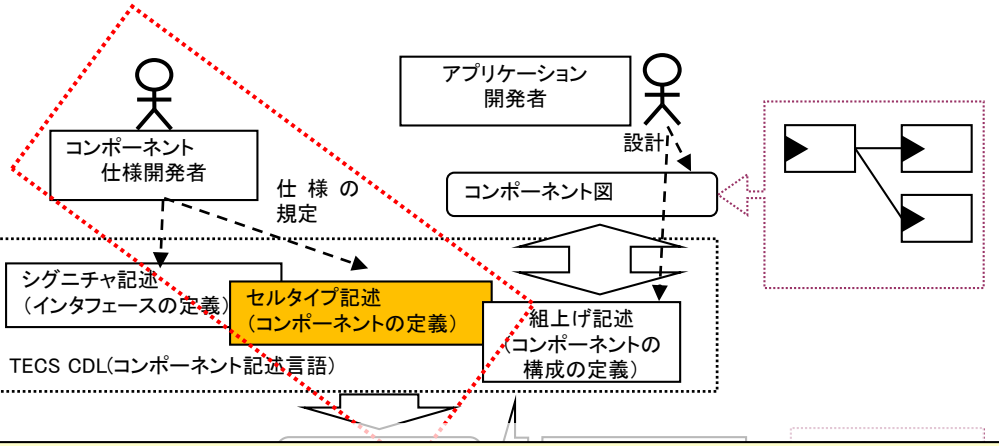
・[ ] 部分を取り除くと、C のプロトタイプ宣言になる
・in, out は入出力方向の指定、size_is, count_is, string はポインタの指定子で、配列長さ、有効要素数、文字列を指定



TOPPERS

# TECS CDL：コンポーネントの定義（セルタイプ）



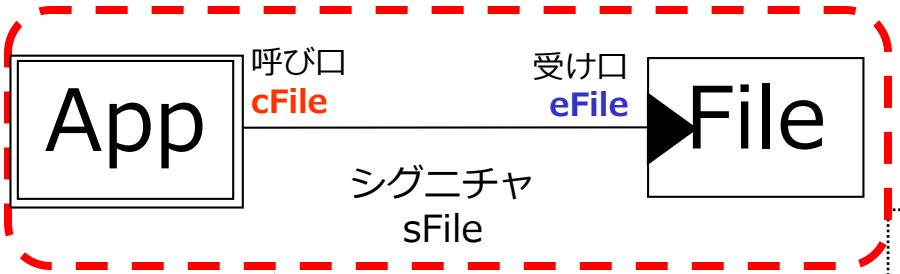

## セルタイプ記述

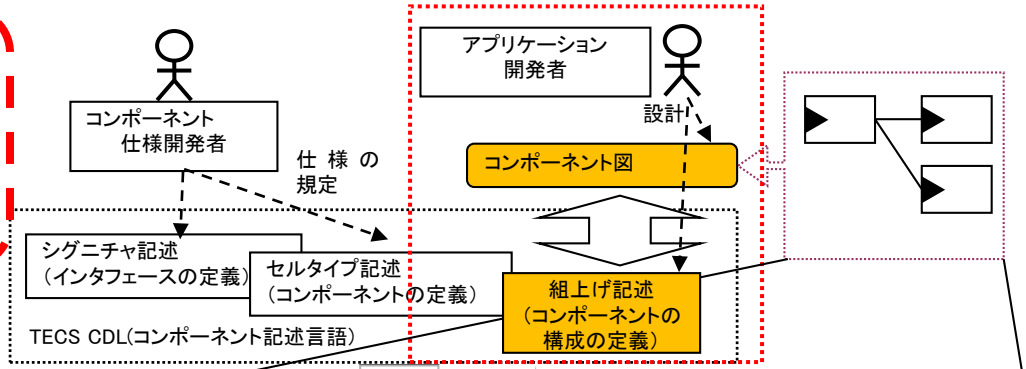
```
celltype tApp{
    // 呼び口(call) の設置
    call sFile cFile;
};
```

- 属性は、デフォルトの値を指定できる(未指定の場合、セル定義時に値指定が必須)
- (内部)変数は、属性を参照して初期化できる

```
celltype  tFile{
    // 受け口(entry) の設置
    entry sFile eFile;
    attr {  // 属性：コンポーネント（セル）ごとの定数
            int16_t  buffer_len = 512;
    };
    var { // (内部)変数：セルごとの変数
        [size_is(buffer_len)]
                int8_t  *buffer;
        int   fd;  // ファイル記述子
    };
};
```

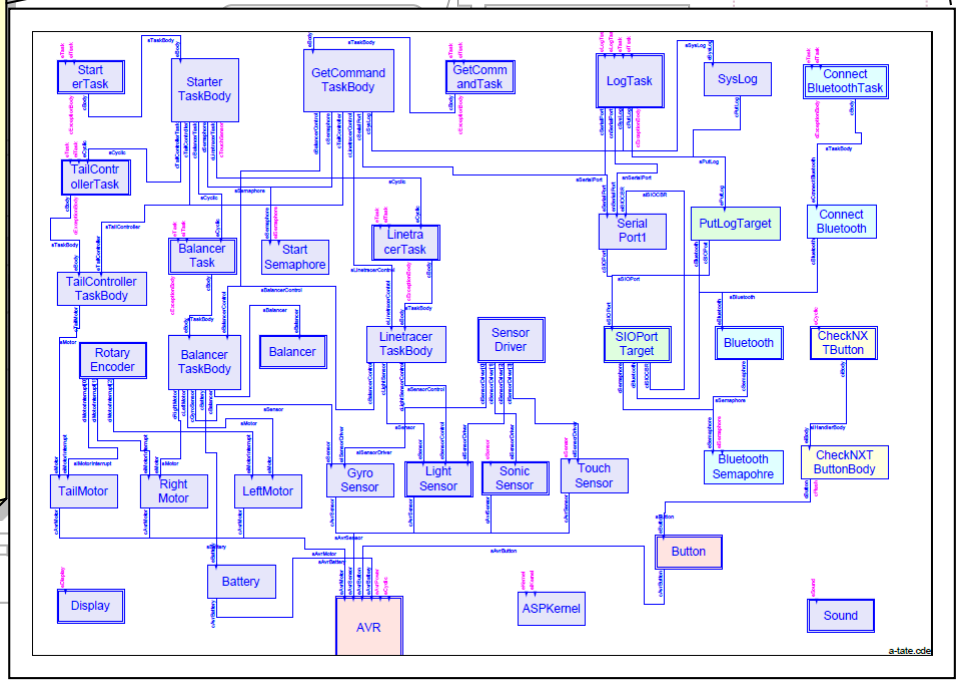
# TECS CDL：コンポーネントの設置と結合




## 組上げ記述

```
// コンポーネント（セル）の静的な生成
cell tFile File{
    buffer_len = 64; // 属性
};
cell tApp App{
    //呼び口を受け口に結合
    cFile = File.eFile;
};
```

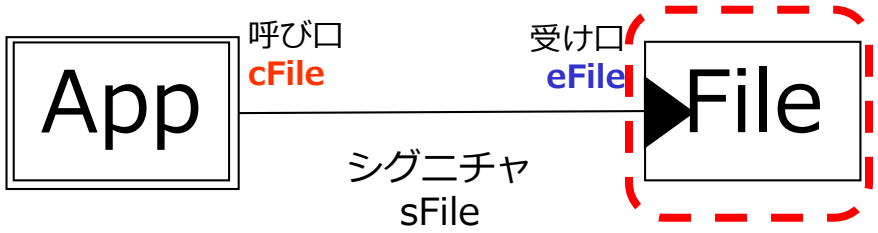



tecscde(GUIツール)



TOPPERS

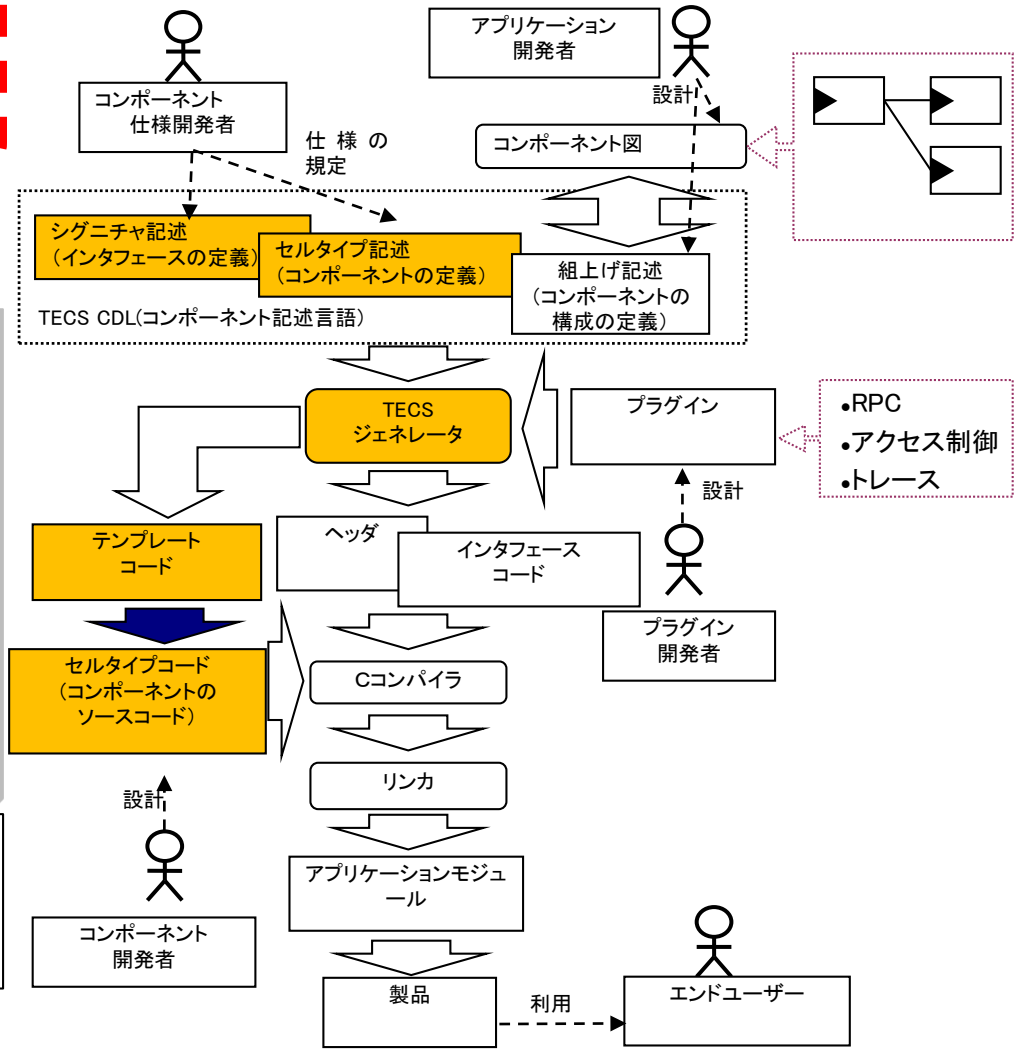
# C言語 ：振る舞いの記述




## テンプレート

```
[tFile.c]
#include “tFile_tecsgen.h”
// 受け口関数 (受け口名)_(関数名)
ER        eFile_open( … )
{
        コンポーネントの振舞いを記述
}
ER        eFile_close()
                  :
```

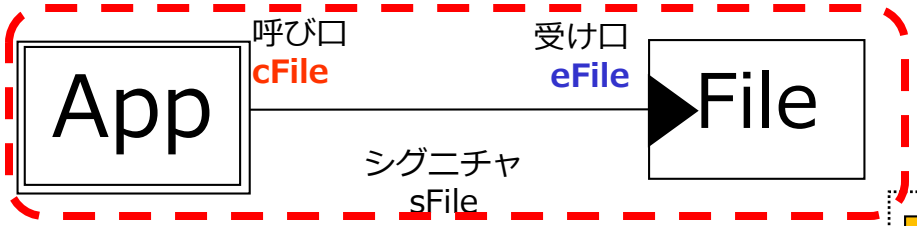
- TECS ジェネレータがテンプレートを生成するので、それを埋める形でセルタイプコードを作成できる

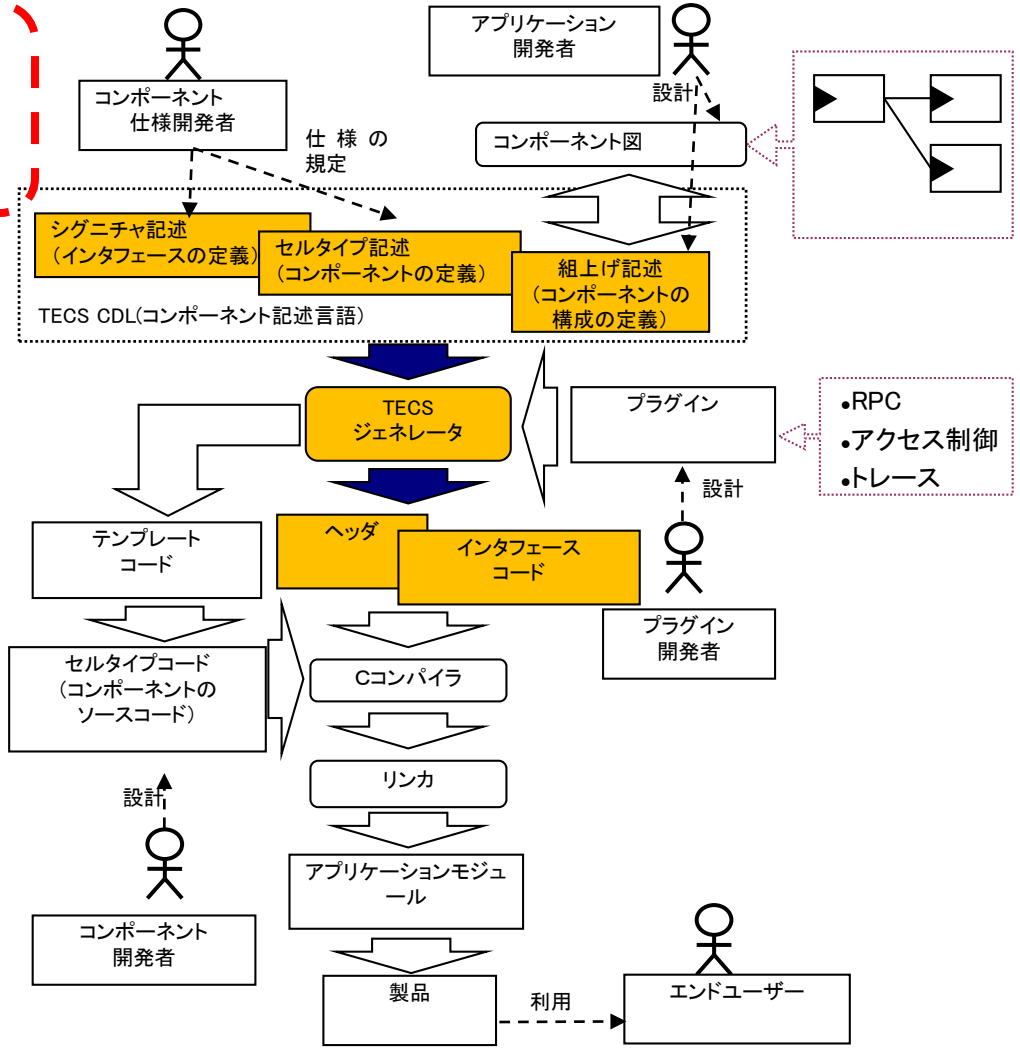





TOPPERS

# TECSジェネレータ：インタフェースコード生成




- コンポーネント間をつなぐインタフェースコードを TECSジェネレータが自動生成
- 結合状況に応じて関数テーブルを生成したり、属性や変数等に応じてROM部(定数)、RAM 部（RAM を初期化するコード）を生成

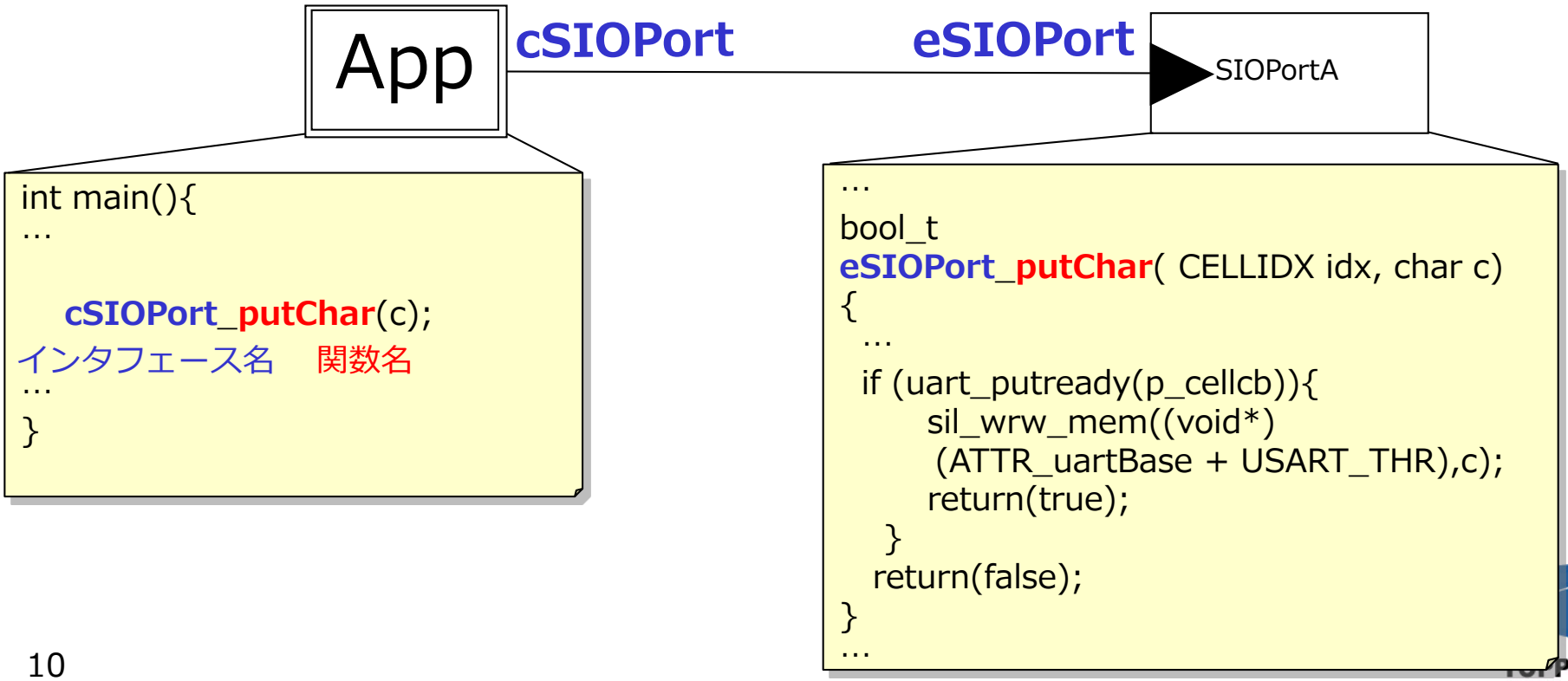


TOPPERS

# 再利用を上げる仕組み：App

補足：SIO：Serial Input Output
TOPPERSでは、ターゲット依存部のシリアルドライバ
※実際のアプリケーションはSerialPort（ターゲット非依存）を利用することが多い。

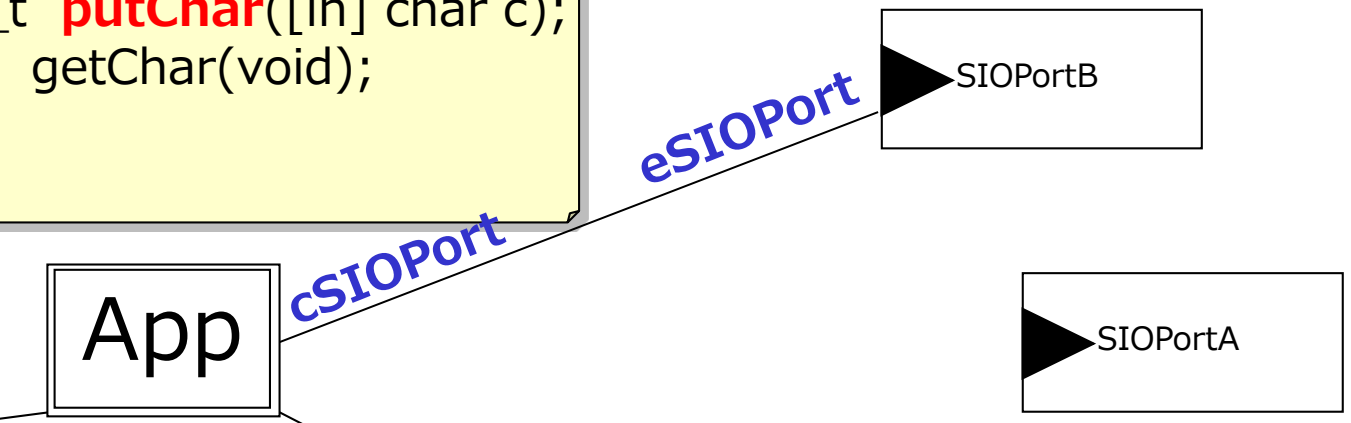
インタフェース定義
```
signature sSIOPort {
        void    open(void);
        void    close(void);
        bool_t  putChar([in] char c);
        char    getChar(void);
        …
};
```



# 再利用を上げる仕組み：App

インタフェース定義

```
signature sSIOPort {
        void    open(void);
        void    close(void);
        bool_t  putChar([in] char c);
        char    getChar(void);
        …
};
```



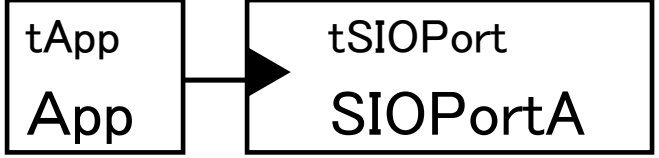

```
int main(){
…

    cSIOPort_putChar(c);
インタフェース名    関数名
…
}
```

TECSでは、SIOPortAとSIOPortBのどちらを利用する場合でも、App側の同じコード（C言語）を利用可能=>再利用性の向上



TOPPERS

# 結合の実装構造の標準形




呼び側

受け側

受け口関数テーブル

tSIOPort_eSIOPort_open_skel
tSIOPort_eSIOPort_close_skel
tSIOPort_eSIOPort_putChar_skel
tSIOPort_eSIOPort_getChar_skel

受け口スケルトン関数

```
ER      tSIOPort_eSIOPort_putChar_skel( struct
            tag_sSig1_VDES *epd, char c)
{
    struct tag_tSIOPort_eSIOPort_DES *lepd
        = (struct tag_tSIOPort_eSIOPort_DES *)epd;
    return tSIOPort_eSIOPort_putChar( lepd->idx, c );
}
```

cSIOPort_putChar(c )

受け口
ディスクリプタ

&tSIOPort_eSIOPort_MT

&tSIOPort_SIOPortA_CB

受け口関数テーブルへの
ポインタ

受け側のセルＣＢ

受け口関数

```
/* 呼び口関数マクロ（短縮形） */
#define cSIOPort_putChar( c ) ¥
            tSIOPort_cSIOPort_putChar( p_cellcb, c)
#define tApp_cSIOPort_putChar( p_that, c  ) ¥
            (p_that)->cSIOPort->VMT->¥
            putChar( (p_that)->cSIOPort, c )
```

呼び側のセルＣＢ

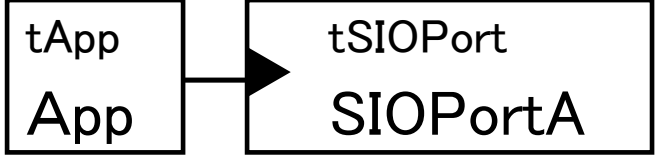
```
bool_t eSIOPort_putChar(CELLIDX idx, char c)
{
    CELLCB    *p_cellcb;
    //エラーチェック省略
    if (uart_putready(p_cellcb)){
        sil_wrw_mem((void*)
      (ATTR_uartBase + USART_THR),c);
        return(true);
    }
    return(false);
}
```

```
typedef struct tag_tApp_CB {
    /* call port */
    struct tag_sSIOPort_VDES *cSIOPort;
}  tApp_CB;
```



TOPPERS

# 結合の実装構造の標準形




呼び側

受け側

受け口関数テーブル

cSIOPort_putChar(c )

tSIOPort_eSIOPort_open_skel
tSIOPort_eSIOPort_close_skel
tSIOPort_eSIOPort_putChar_skel
tSIOPort_eSIOPort_getChar_skel

受け口スケルトン関数

tSIOPort_eSIOPort_putChar_skel( struct tag_sSig1_VDES *epd, char c)

struct tag_tSIOPort_eSIOPort_DES *lepd
= (struct tag_tSIOPort_eSIOPort_DES *)epd;
return tSIOPort_eSIOPort_putChar( lepd->idx, c );

コンポーネントの構成によっては

&tSIOPort_eSIOPort_MT
&tSIOPort_SIOPortA_CB

受け口関数テーブルへのポインタ

受け側のセルＣＢ

最適化で省略可

受け口関数

```
bool_t eSIOPort_putChar(CELLIDX idx, char c)
{
   CELLCB    *p_cellcb;
    //エラーチェック省略
    if (uart_putready(p_cellcb)){
        sil_wrw_mem((void*)
      (ATTR_uartBase + USART_THR),c);
        return(true);
    }
   return(false);
}
```

```
typedef struct tag_tApp_CB {
   /* call port */
   struct tag_sSIOPort_VDES *cSIOPort;
} tApp_CB;
```

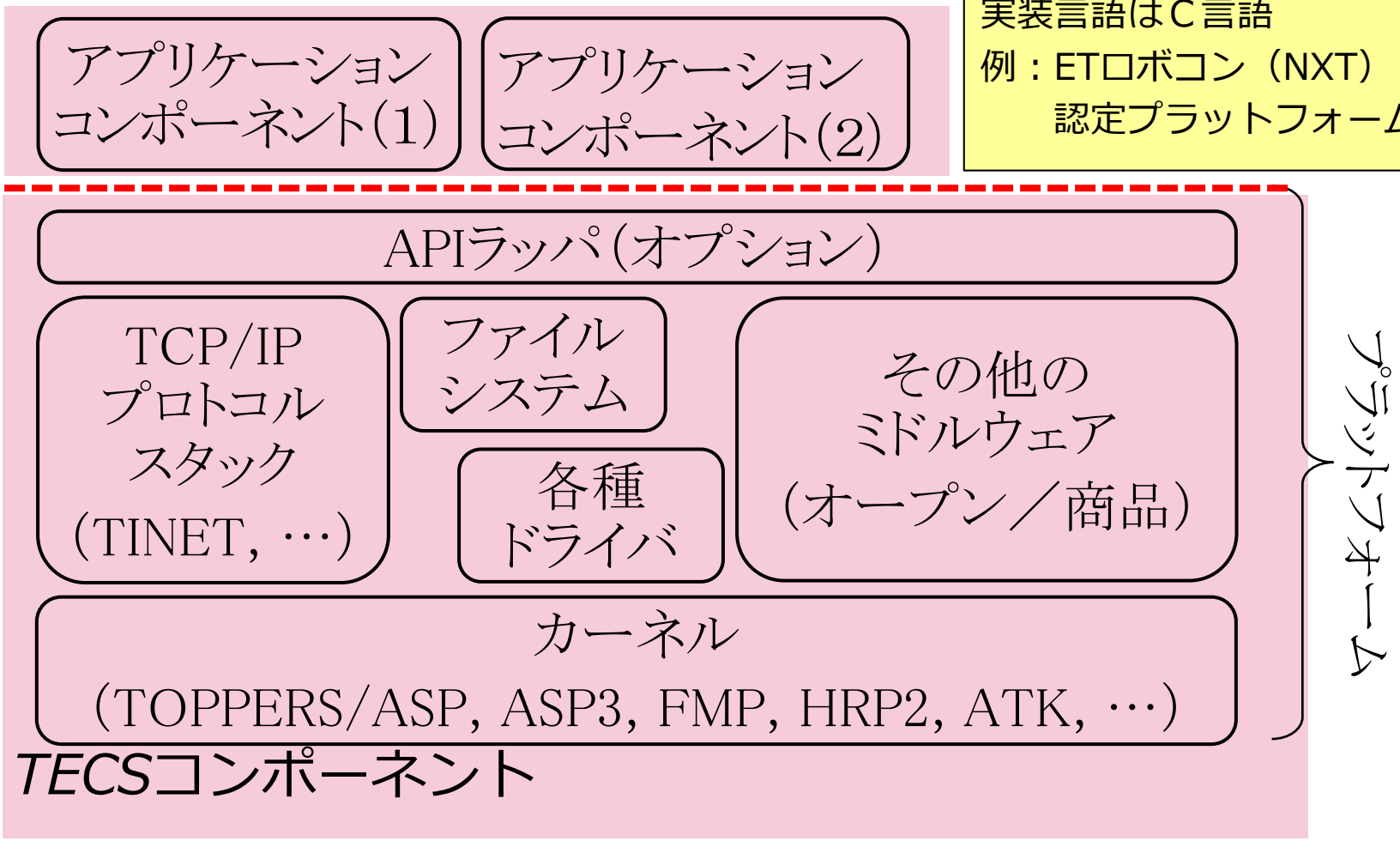


TOPPERS

# TECSの適用イメージ①：システム全体への適用

ASP3/HRP3等のアプリケーションが共通で利用できる

アプリケーションを含めすべての要素をTECS仕様に準拠
（再利用性高）
実装言語はＣ言語
例：ETロボコン（NXT）
　　認定プラットフォーム






TOPPERS

# TECSの適用イメージ②：プラットフォームへの適用

## レガシーコードも利用可能

アプリケーション
モジュール(1)

アプリケーション
モジュール(2)

アプリケーションは、
既存と同様の実装（C言語）
TECSをライブラリとして利用
後述のASP3はこの構成も
サポート

API

APIラッパ(オプション)

TCP/IP
プロトコル
スタック
(TINET, …)

ファイル
システム

各種
ドライバ

その他の
ミドルウェア
(オープン／商品)

カーネル
(TOPPERS/ASP, ASP3, FMP, HRP2, ATK, …)

プラットフォーム

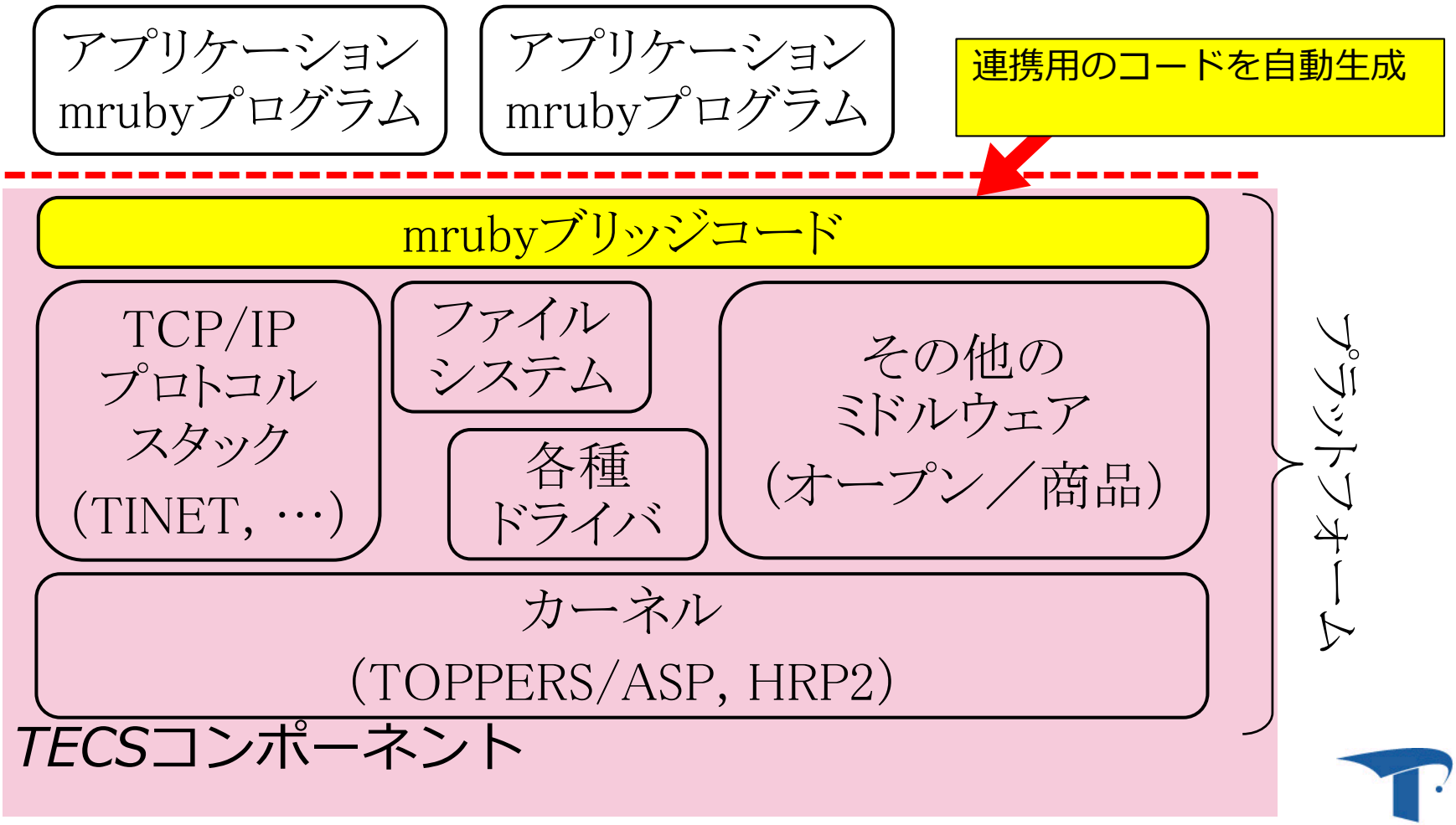
*TECS*コンポーネント



TOPPERS

# TECSの適用イメージ③：自動生成機構を利用

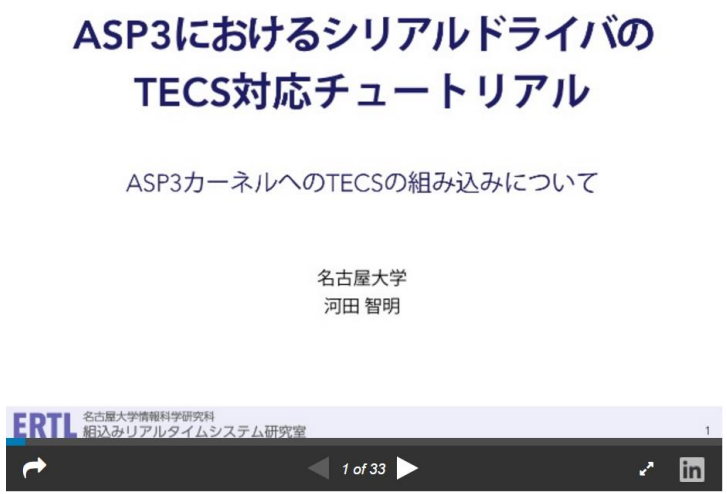
mruby：組込みシステム向けにRuby軽量化したスクリプト言語
プログラマはプラットフォーム側（C言語）の知識がなくても利用可能
例：ETロボコン（EV3）：認定プラットフォーム



# TECSがASP3に標準採用：SysLog, シリアルドライバ、ログタスクの実装

- ○ SysLogの出力先（シリアル、Bluetooth、LCD等）をプログラムを修正せずに変更可能
- △ ターゲット依存部のポーティングには、TECSの基本を覚える必要がある
  - ○ TECSのすべての機能を使いこなすには、時間がかかるが、ポーティングに必要な最低限の知識であれば、それほど時間はかからない
- ○ ポーティング初心者には、何を実装すればよいかが明確になる

  →シリアルドライバ移植チュートリアルを公開

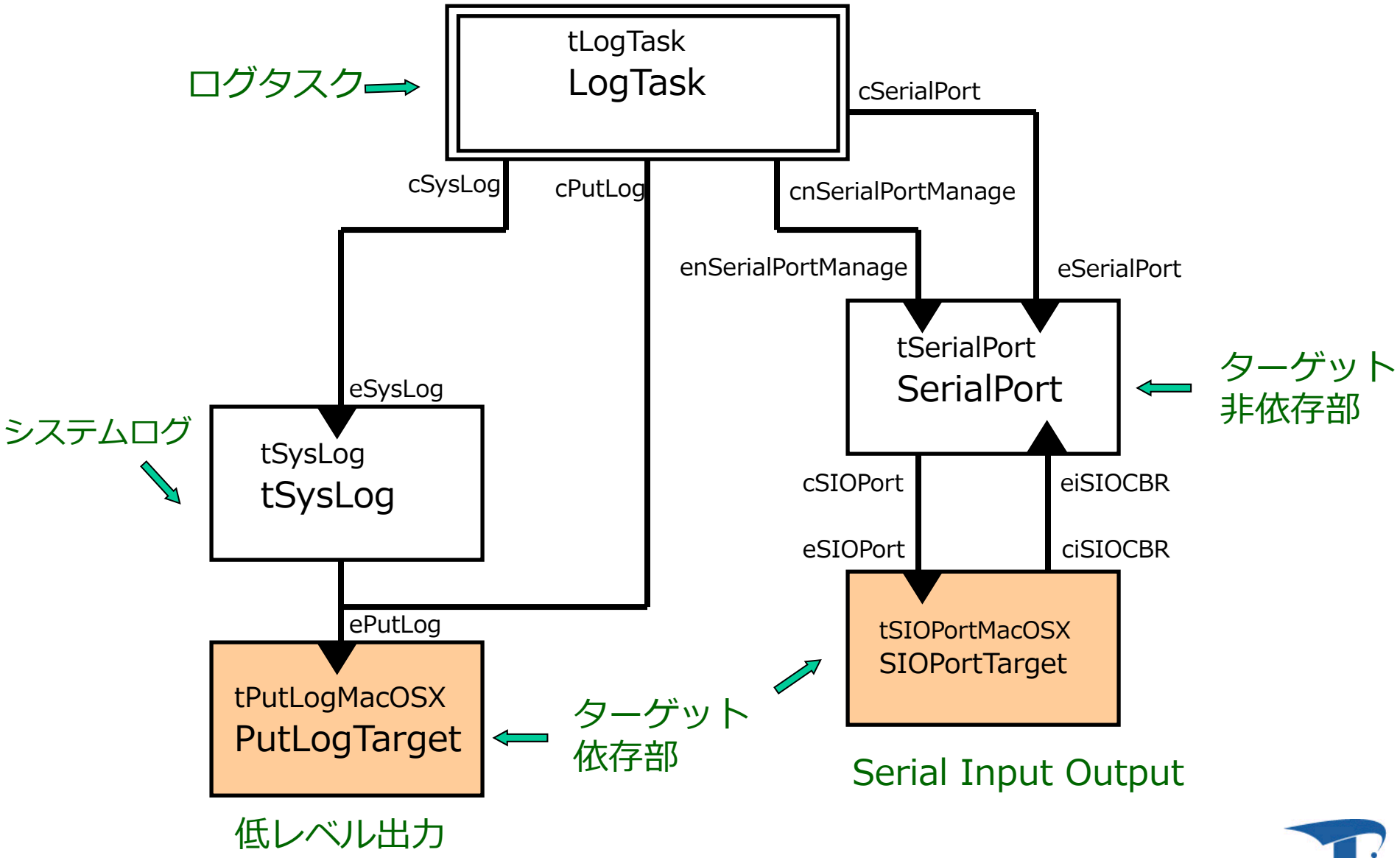



https://sites.google.com/site/tecsmanual/tutorial/asp3-tecs-tutorial



TOPPERS

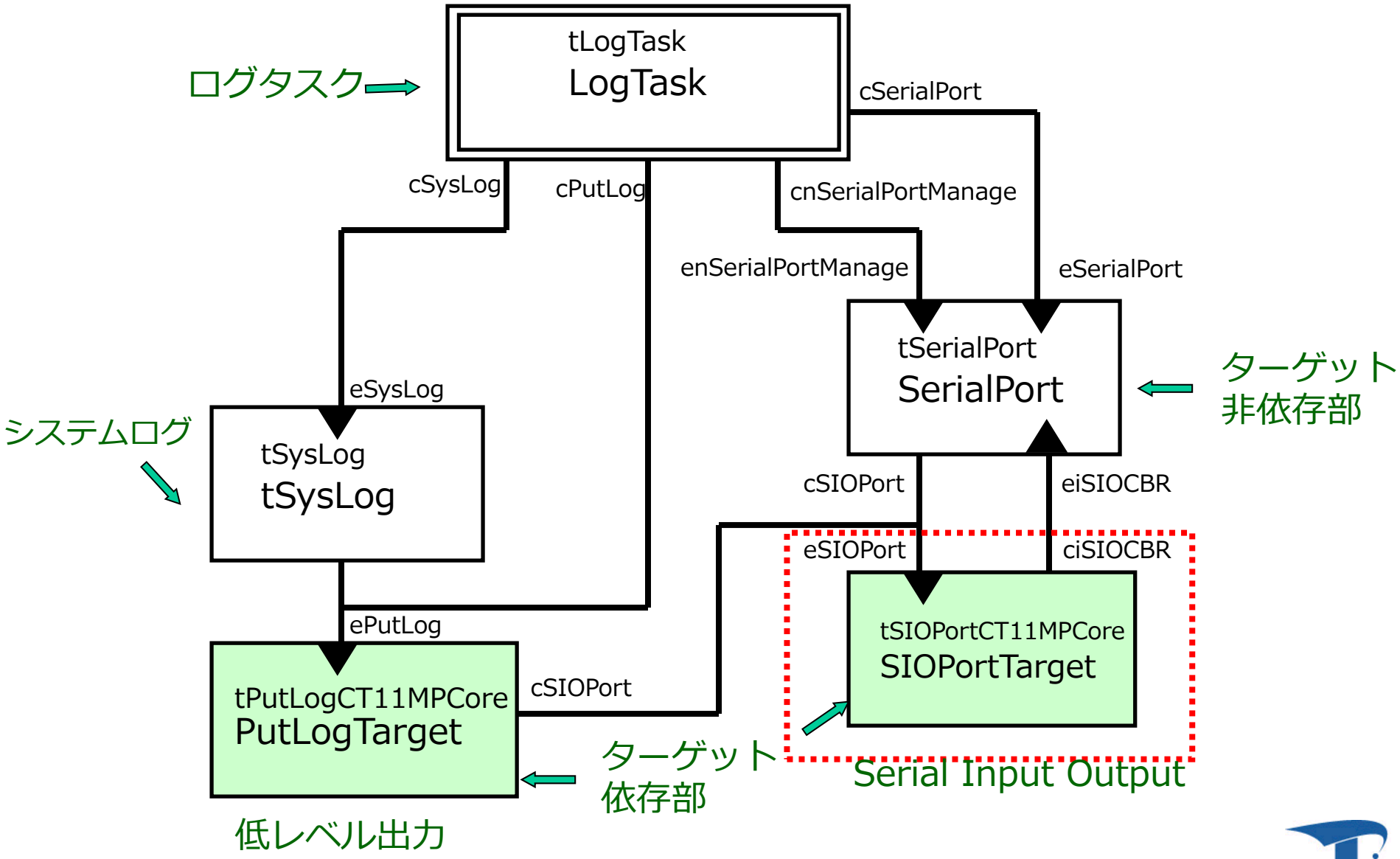
# ASP3：ログタスク＆シリアルドライバの例（Mac）






TOPPERS

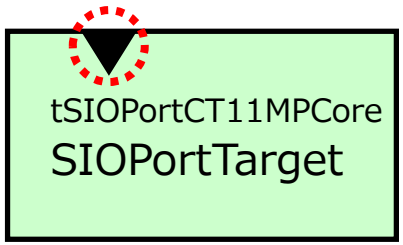
# ASP3：ログタスク＆シリアルドライバの例（ARM）






TOPPERS

# SIOPortのポーティング例




TECSジェネレータが
テンプレートコードを生成

インタフェース定義

```
signature sSIOPort {
        void    open(void);
        void    close(void);
        bool_t  putChar([in] char c);
        int_t   getChar(void);
        …
};

celltype tSIOPortCT11MPCoreMain {
    entry    sSIOPort eSIOPort;
        …
}
```

```
void
eSIOPort_open(CELLIDX idx)
{
    CELLCB          *p_cellcb;
    if (VALID_IDX(idx)) {
        p_cellcb = GET_CELLCB(idx);
    }
```

ここにドライバの
コードを実装

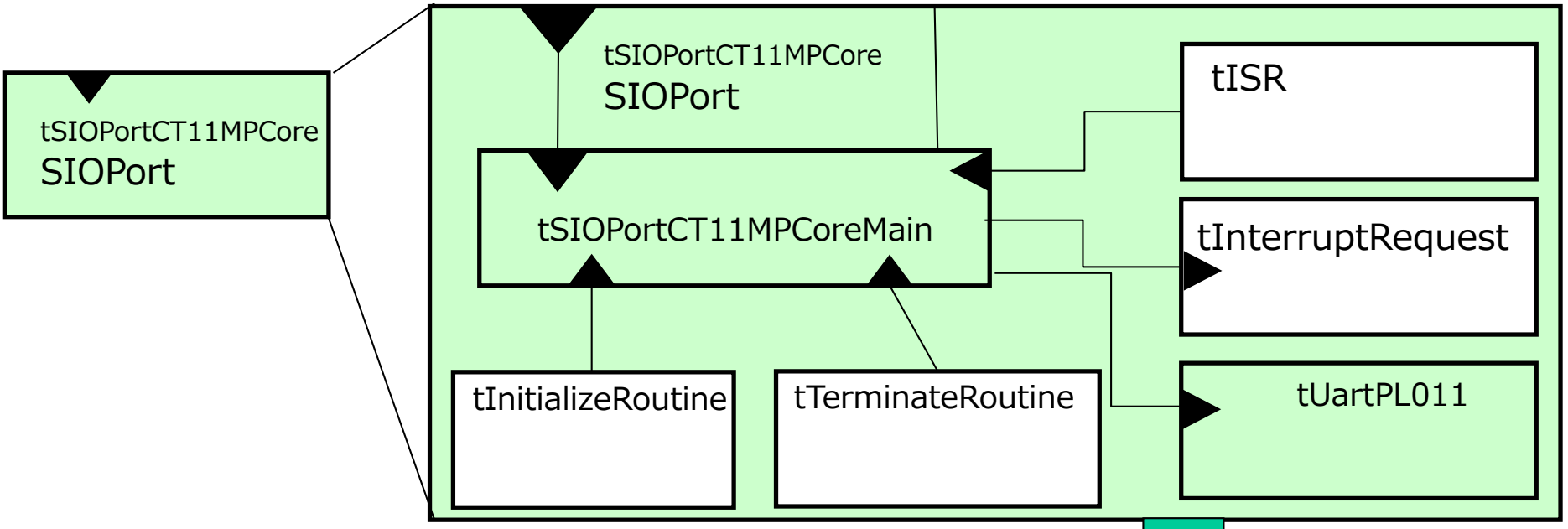
```
}
void
eSIOPort_close(CELLIDX idx)
{
    …
```



TOPPERS

# TECSの利点：
# 複合コンポーネント&静的API自動生成

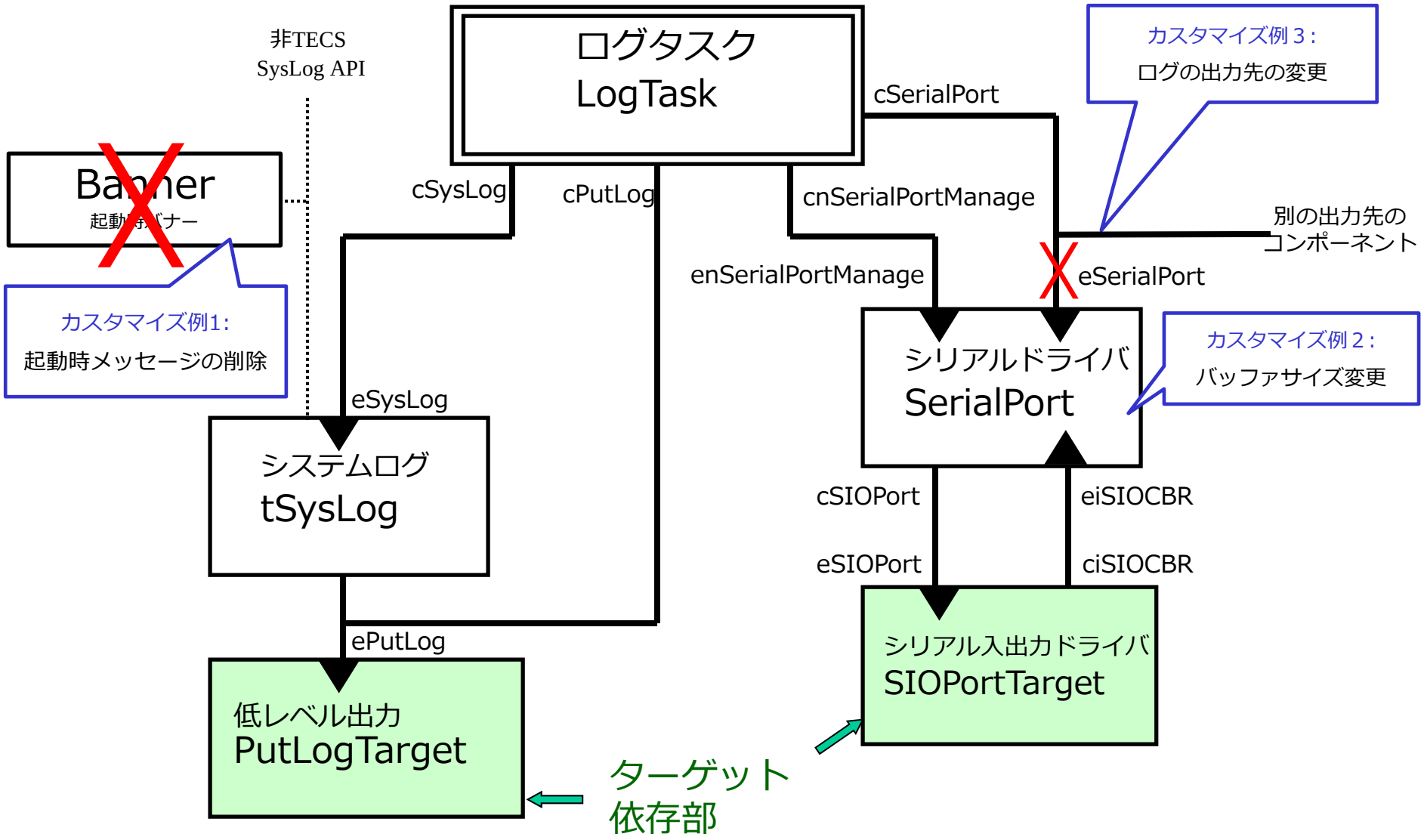
- 利用者はSIOPortが複数のコンポーネントで構成されていることを意識せずに利用可能




カーネルの設定ファイル(静的API)の生成

```
CFG_INT(EB_IRQNO_UART0, { TA_NULL, -2 });
CRE_ISR(ISRID_tISR_SIOPortTarget_base_ISRInstance, { TA_NULL, &tISR_CB_tab[0],
EB_IRQNO_UART0, tISR_start, 1 });
ATT_INI({ TA_NULL, NULL, tInitializeRoutine_start });
ATT_TER({ TA_NULL, NULL, tTerminateRoutine_start });
```

# TECSの利点：システムサービスのカスタマイズ



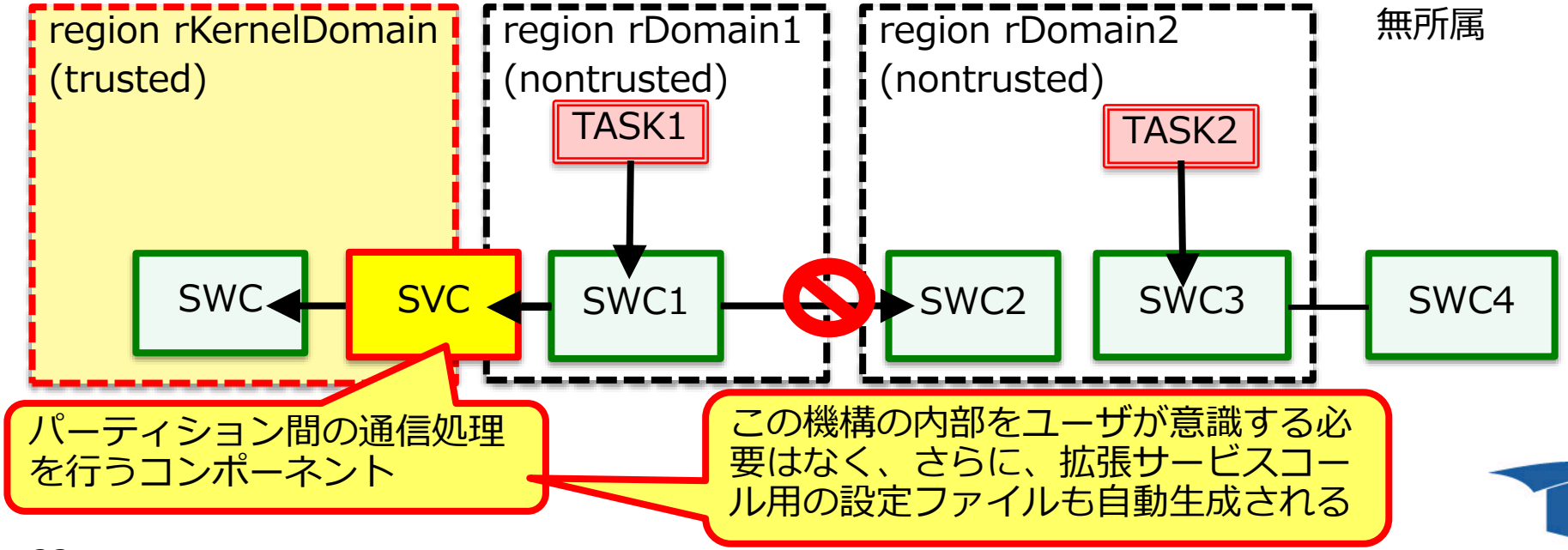

C言語を編集しなくて構成を変更できる



TOPPERS

# TECSの利点：メモリ保護の設定（HRP2/3）

- コンポーネント記述からHRP2/3カーネルの設定ファイルを自動生成する
  - リージョン（グループ化）を保護ドメインに対応させる
    - リージョンに所属するタスクおよび、コンポーネント（セル）固有のデータを、対応する保護ドメインに所属させる
  - **メモリ設定ファイル、拡張サービスコール、静的API**をTECSジェネレータにより出力






TOPPERS

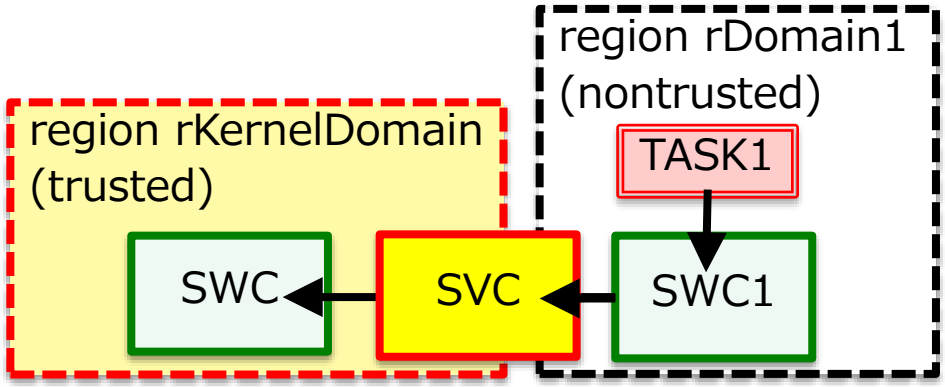
# TECSの導入で改善した点：拡張サービスコール対応

## TECSを利用しない場合のコード：拡張サービスコール

```
ER_UINT
extsvc_syslog_wri_log(intptr_t prio, intptr_t p_syslog,   intptr_t par3, intptr_t par4,
                      intptr_t par5, ID cdmid)   {
  ER_UINT ercd;
  if (!EXTSVC_PROBE_MEM_READ(p_syslog, SYSLOG)) {
    ercd = E_MACV;
  }
  else {
    ercd = (ER_UINT) _syslog_wri_log((uint_t) prio, (const SYSLOG *) p_syslog);
  }
  return(ercd);
}
```

引数の数が違う

サービスコールの本体




TECSの場合は上記コードを自動生成するため
ASP 3 とHRP3で共通のプログラムが利用可能



TOPPERS

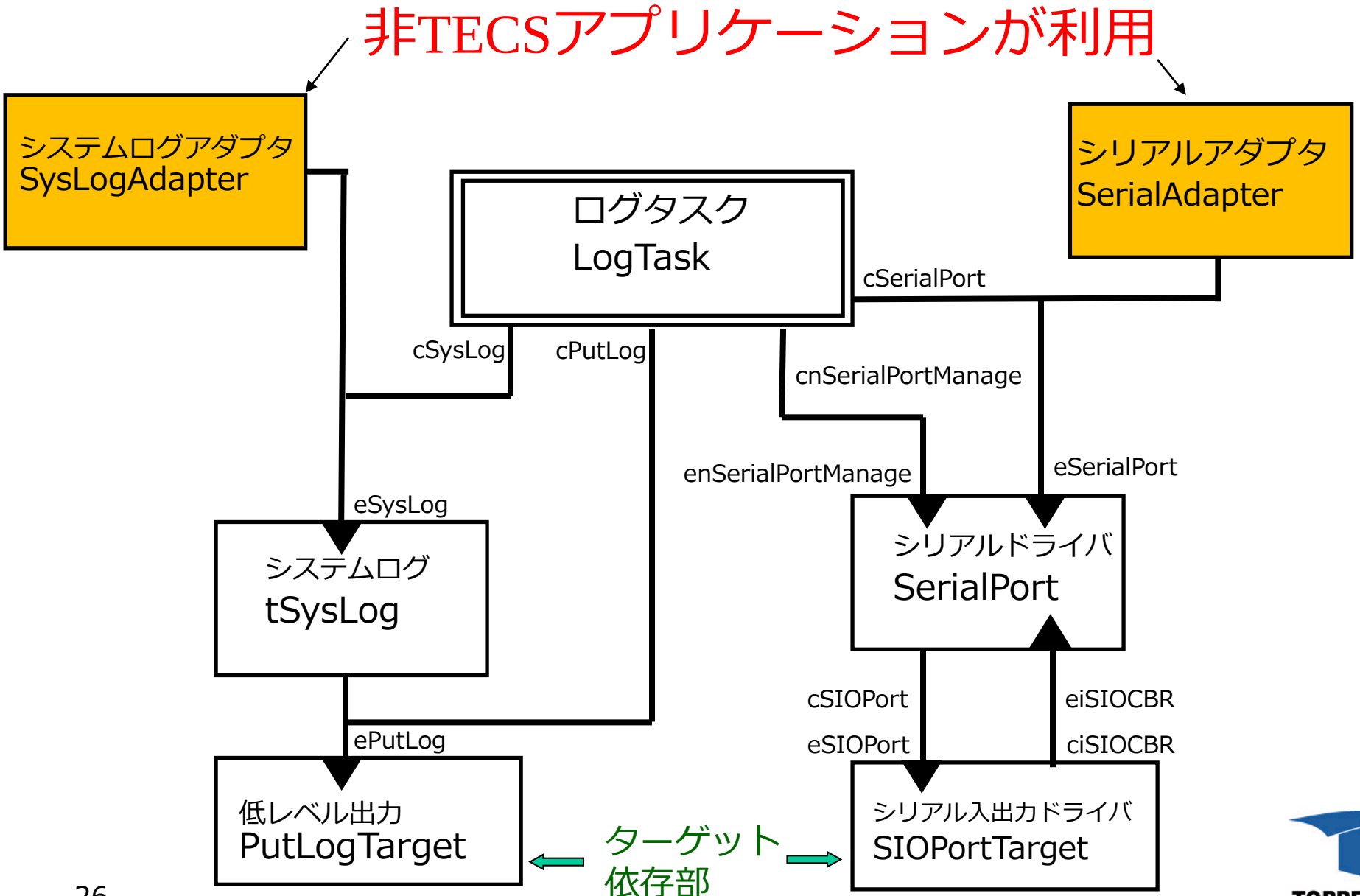
# TECSの導入で改善した点

TECS導入前のコード

```
#ifndef SERIAL_RCV_BUFSZ1
#define SERIAL_RCV_BUFSZ1 256 /* ポート1の受信バッファサイズ */
#endif /* SERIAL_RCV_BUFSZ1 */

#ifndef SERIAL_SND_BUFSZ1
#define SERIAL_SND_BUFSZ1 256 /* ポート1の送信バッファサイズ */
#endif /* SERIAL_SND_BUFSZ1 */

static char rcv_buffer1[SERIAL_RCV_BUFSZ1];
static char snd_buffer1[SERIAL_SND_BUFSZ1];

#if TNUM_PORT >= 2 /* ポート2に関する定義 */

#ifndef SERIAL_RCV_BUFSZ2
#define SERIAL_RCV_BUFSZ2 256     /* ポート2の受信バッファサイズ */
#endif /* SERIAL_RCV_BUFSZ2 */

＜同様の定義が続く（TNUM_PORTが4まで対応）＞
```

ifdefを利用しなくても記述可能に
（ジェネレータが自動生成）

# アダプタの提供
# 非TECSアプリケーションを利用する仕組み






TOPPERS

# アダプタの提供
# 非TECSアプリケーションを利用する仕組み

アダプタがC言語の関数を提供する
これまでのレガシーコードが利用可能

シリアルアダプタ
SerialAdapter

シリアルアダプタが提供する関数の例

```
ER_UINT
serial_rea_dat(ID portid, char *buf, uint_t len)
{
  if (sns_dpn()) {    /* コンテキストのチェック */
          return(E_CTX);
  }
  if (!(1 <= portid && portid <= N_CP_cSerialPort)) {
          return(E_ID);       /* ポート番号のチェック */
  }
          return(   cSerialPort_read(portid - 1, buf, len)   );
}
```

シリアルアダプタの接続先の
シリアルドライバを呼び出す

アプリケーション開発者はTECSを
知らなくても利用可能

# その他の最新動向

- mrubyプラットフォーム
  - mruby on EV3RT + TECS
    - ETロボコン認定プラットフォーム
    - mruby教育教材
      - http://www.afrel.co.jp/archives/21493
- 動的結合
  - テストコード
  - TINET
- TOPPERS/ATK2（AUTOSAR OS）対応
- ECHONET連携開始
- リファレンス・マニュアル整備
  - 随時更新中
    http://tecs-docs.readthedocs.io/ja/latest/asp3/index.html



TOPPERS

# TECSリファレンスマニュアル：随時更新中

http://tecs-docs.readthedocs.io/ja/latest/asp3/index.html



tMyAnotherCellType.c¶

```
// タスクの起動
cTask_activate();

// タスクの現在状態の参照
T_RTSK taskStatus;
cTask_refer(&taskStatus);
```

なお、非タスクコンテキスト内では、`eTask` の代わりに `eiTask` を使用する必要があります。

## リファレンス

### セルタイプ

*celltype* **tTask**

タスクの生成、制御及び状態の取得を行うコンポーネントです。

本コンポーネントは `CRE_TSK` 静的API [NGKI1023] によりタスクの生成を行います。静的APIの引数の値には、一部を除き属性値が用いられます。

*attr* ID **id**

タスクのID番号の識別子 (詳しくはカーネルオブジェクトのID番号を参照) を `C_EXP` で囲んで指定します (省略可能)。

指定しない場合、`TSKID_` で始まる識別子が自動生成され、使用されます。

*attr* ATR **attribute**

タスク属性 [NGKI3526] を `C_EXP` で囲んで指定します。複数個指定する場合、ビット毎の論理和演算子を用いて `C_EXP("TA_ACT | TA_NOACTQUE")` のようにして指定します。何も指定しない場合は指定を省略するか、`0` を指定します。

TOPPERS統合仕様書と連携
2017：ESECターゲット



TOPPERS

# 参考資料等

- TECSプレゼン資料
  https://sites.google.com/site/tecsmanual/tutorial/
  - ASP3+TECS移植チュートリアル
    https://sites.google.com/site/tecsmanual/tutorial/asp3-tecs-tutorial

- TECS基本説明の動画（ETロボコン向けセミナーの一部）
  - https://www.toppers.jp/etrobo_seminar.html

- わからないときは
  - TOPPERS 会員の皆さま
    - com-wg@toppers.jp… TECS WG の ML
    - dev@toppers.jp … 開発者 ML
  - 非会員の皆さま
    - users@toppers.jp … ユーザー ML



TOPPERS