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 ～普及期に入ったTECS～

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

TOPPERS

目次

- TECSの基本
- ASP3のSyslog, シリアルドライバ、ログタスクの実装に採用
- TECSの利点
- mrubyプラットフォーム

TECS (TOPPERS Embedded Component System)

- 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

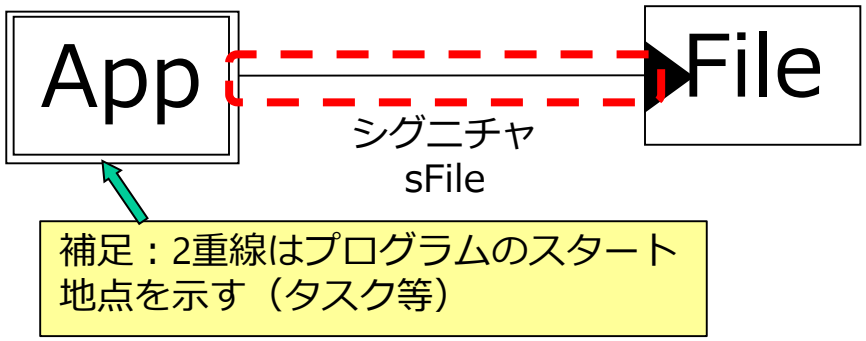
TOPPERS

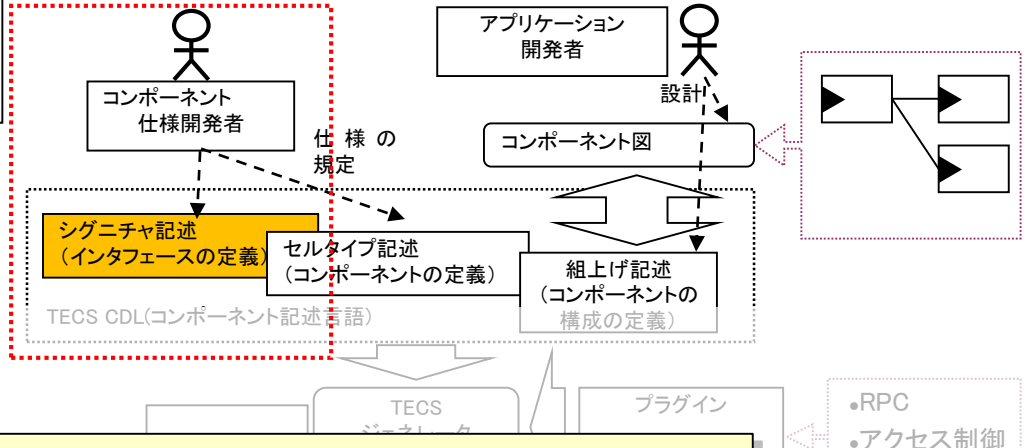
TECS 開発の流れ

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）





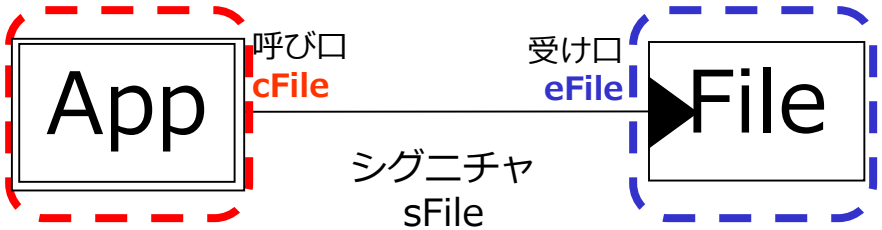
シグニチャ記述

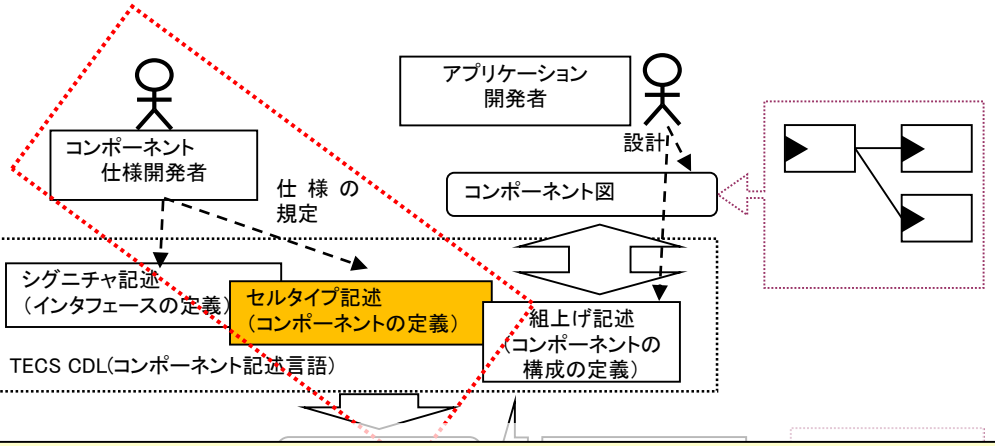
```
signature sFile {
  ER open( [in,string]char * fileName, [in]int16_t mode);
  ER close(void);
  ER read( [out,size_is(length),count_is(*count)]int8_t * buffer,
          [in]int32_t length, [out]int32_t *count);
  ER write( [in,size_is(length)]int8_t *buffer,
       [in]int32_t length, [out]int32_t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





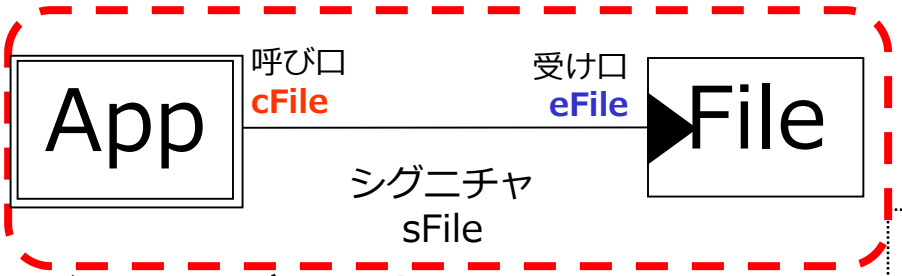
セルタイプ記述

```
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

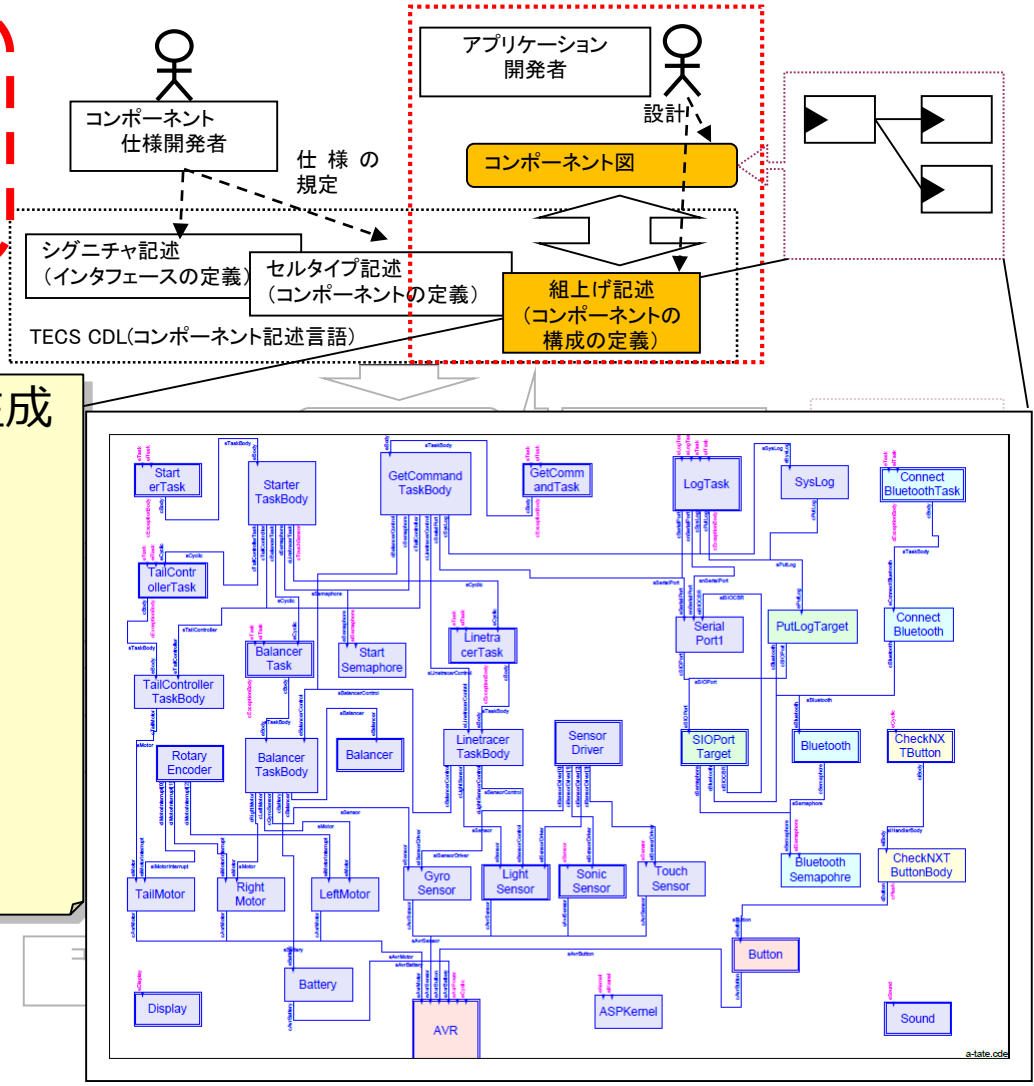
```
celltype  tFile{
    // 受け口(entry) の設置
    entry sFile eFile;
    attr {  // 属性：コンポーネント（セル）ごとの定数
            int16_t  buffer_len = 512;
    };
    var { // (内部)変数：セルごとの変数
        [size_is(buffer_len)]
                int8_t  *buffer;
        int   fd;  // ファイル記述子
    };
};
```

TECS CDL：コンポーネントの設置と結合



組上げ記述

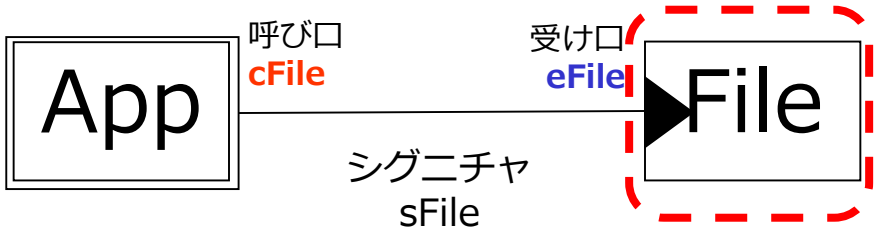
```
// コンポーネント（セル）の静的な生成
cell tFile File{
    buffer_len = 64; // 属性
};
cell tApp App{
    //呼び口を受け口に結合
    cFile = File.eFile;
};
```



tecscde(GUIツール)

TOPPERS

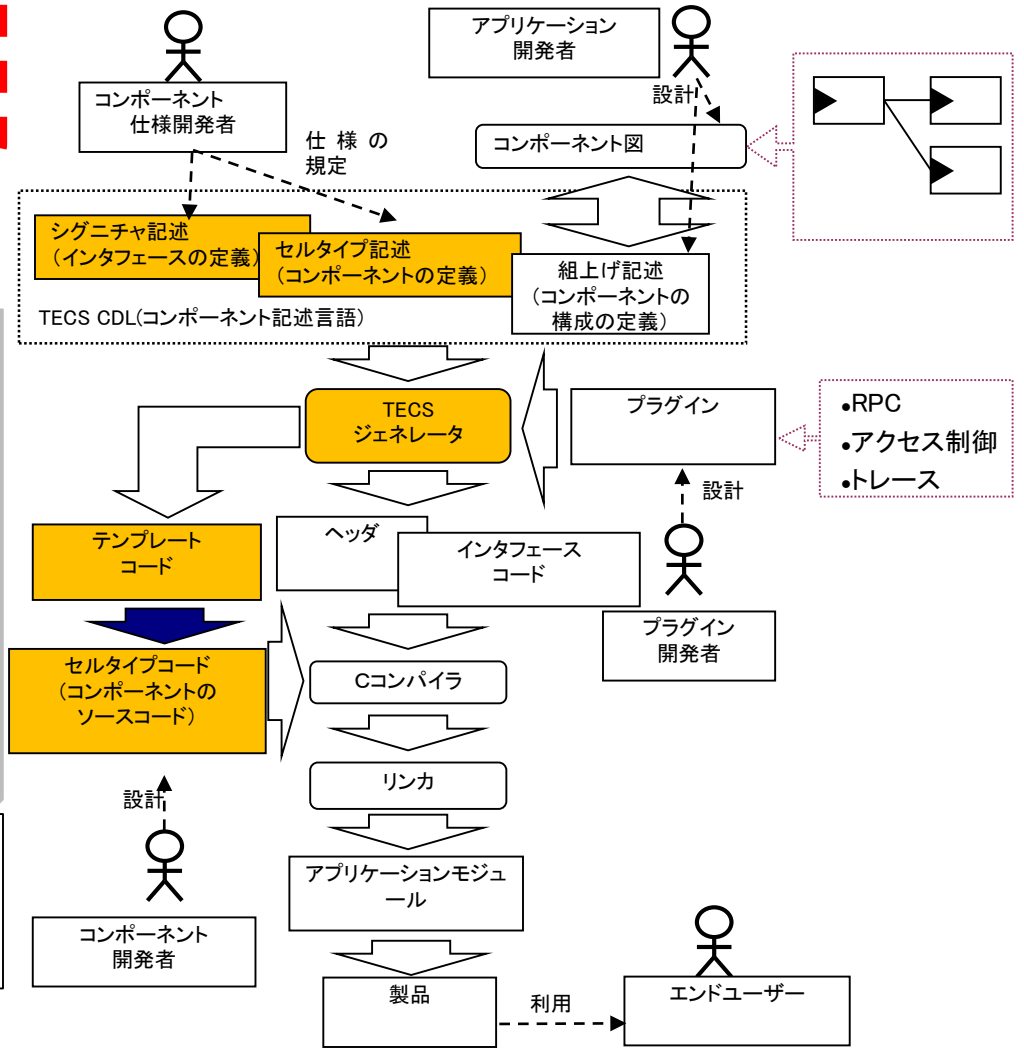
C言語 ：振る舞いの記述



テンプレート

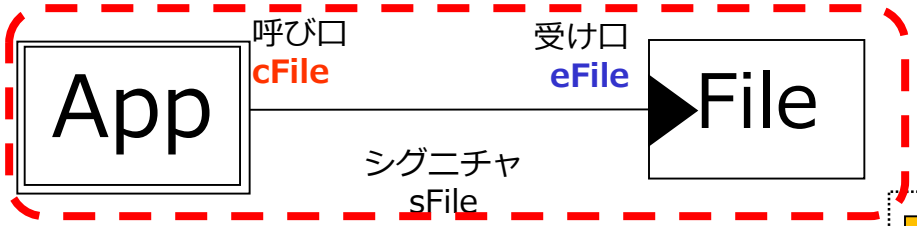
```
[tFile.c]
#include “tFile_tecsgen.h”
// 受け口関数 (受け口名)_(関数名)
ER        eFile_open( … )
{
        コンポーネントの振舞いを記述
}
ER        eFile_close()
                  :
```

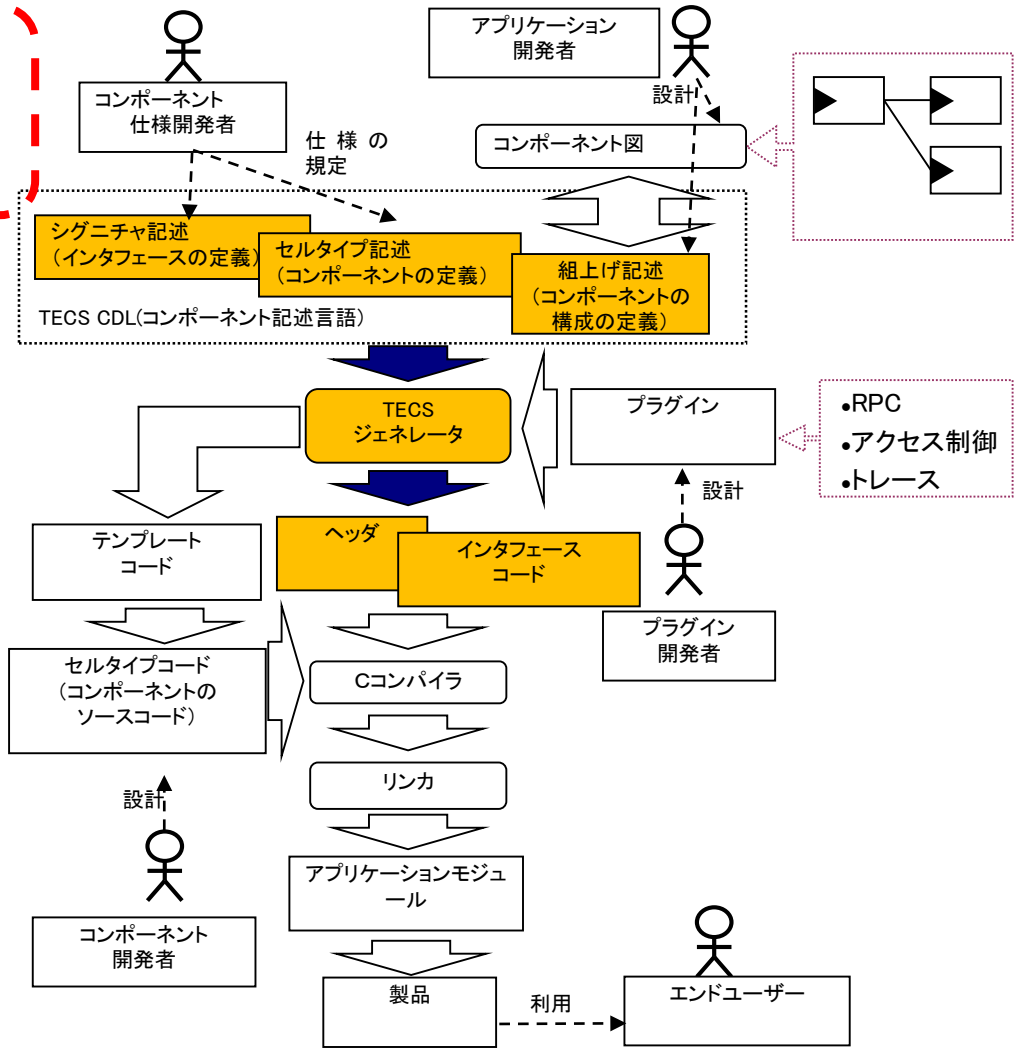
- TECS ジェネレータがテンプレートを生成するので、それを埋める形でセルタイプコードを作成できる



TOPPERS

TECSジェネレータ：インタフェースコード生成





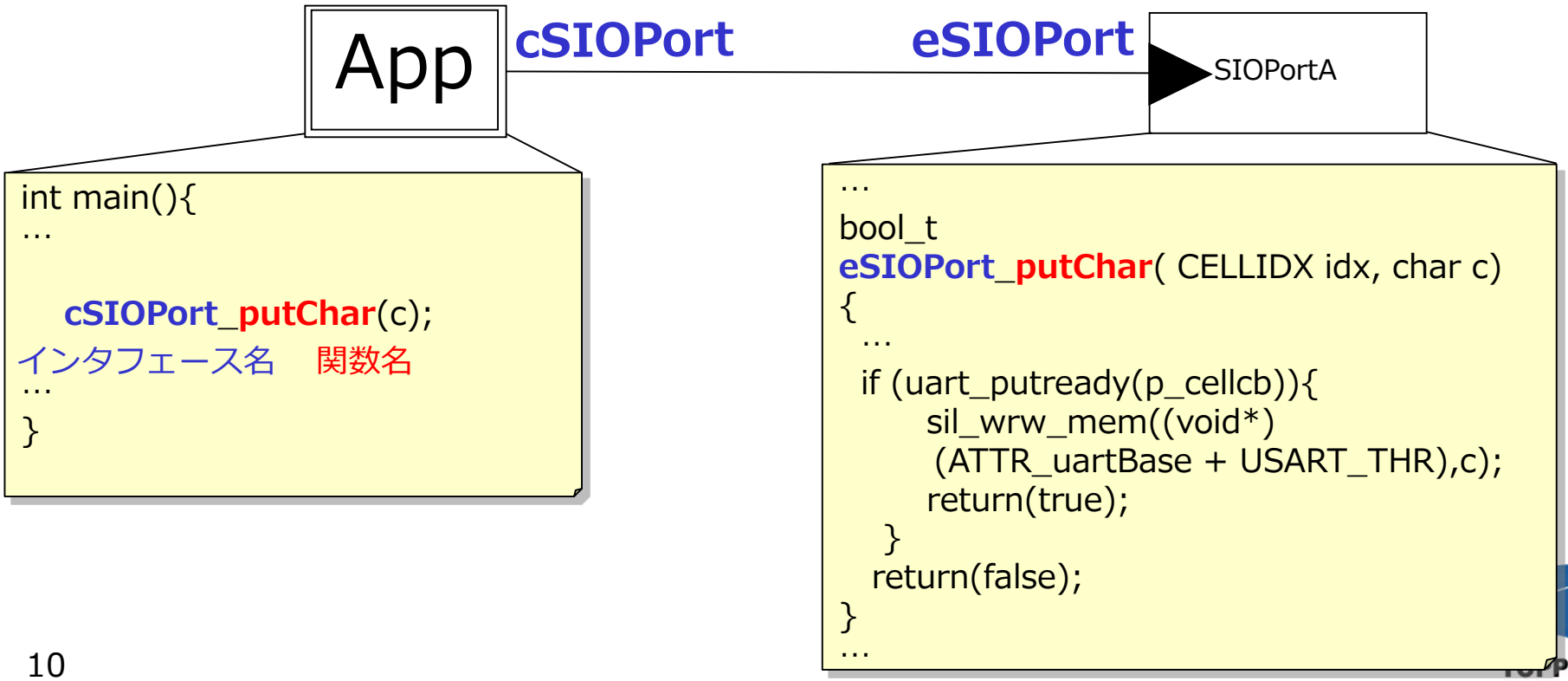
- コンポーネント間をつなぐインタフェースコードを TECSジェネレータが自動生成
- 結合状況に応じて関数テーブルを生成したり、属性や変数等に応じてROM部(定数)、RAM 部（RAM を初期化するコード）を生成

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。

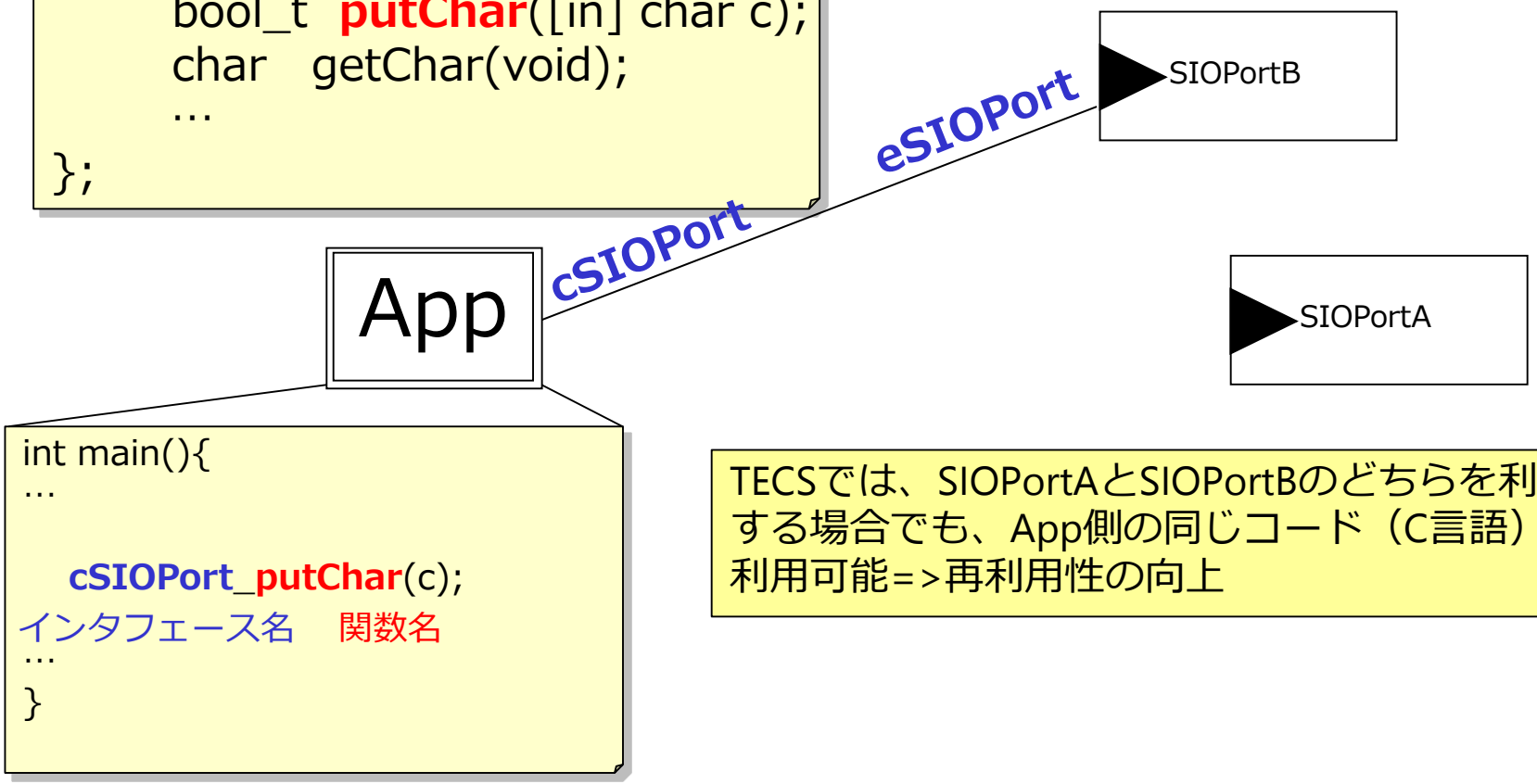
インタフェース定義
```
signature sSIOPort {
        void    open(void);
        void    close(void);
        bool_t  putChar([in] char c);
        char    getChar(void);
        …
};
```



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);
        …
};
```



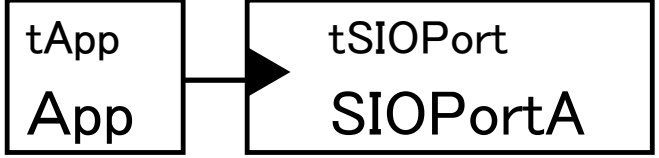

```
int main(){
…

    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 )
```

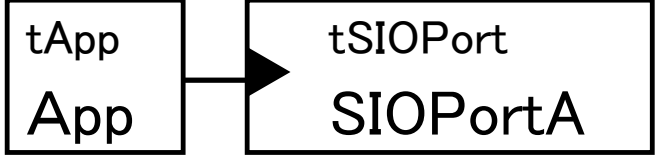
呼び側のセルＣＢ

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


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

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)

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);
}
```

```
typede  struct tag_tApp_CB {
    /* c  ll port */
    stru  t tag_sSIOPort_VDES *cSIOPort;
}  tApp  _CB;
```


TOPPERS

TECSの適用イメージ①：プラットフォームへの適用

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

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, FMP, HRP2, ATK, …)

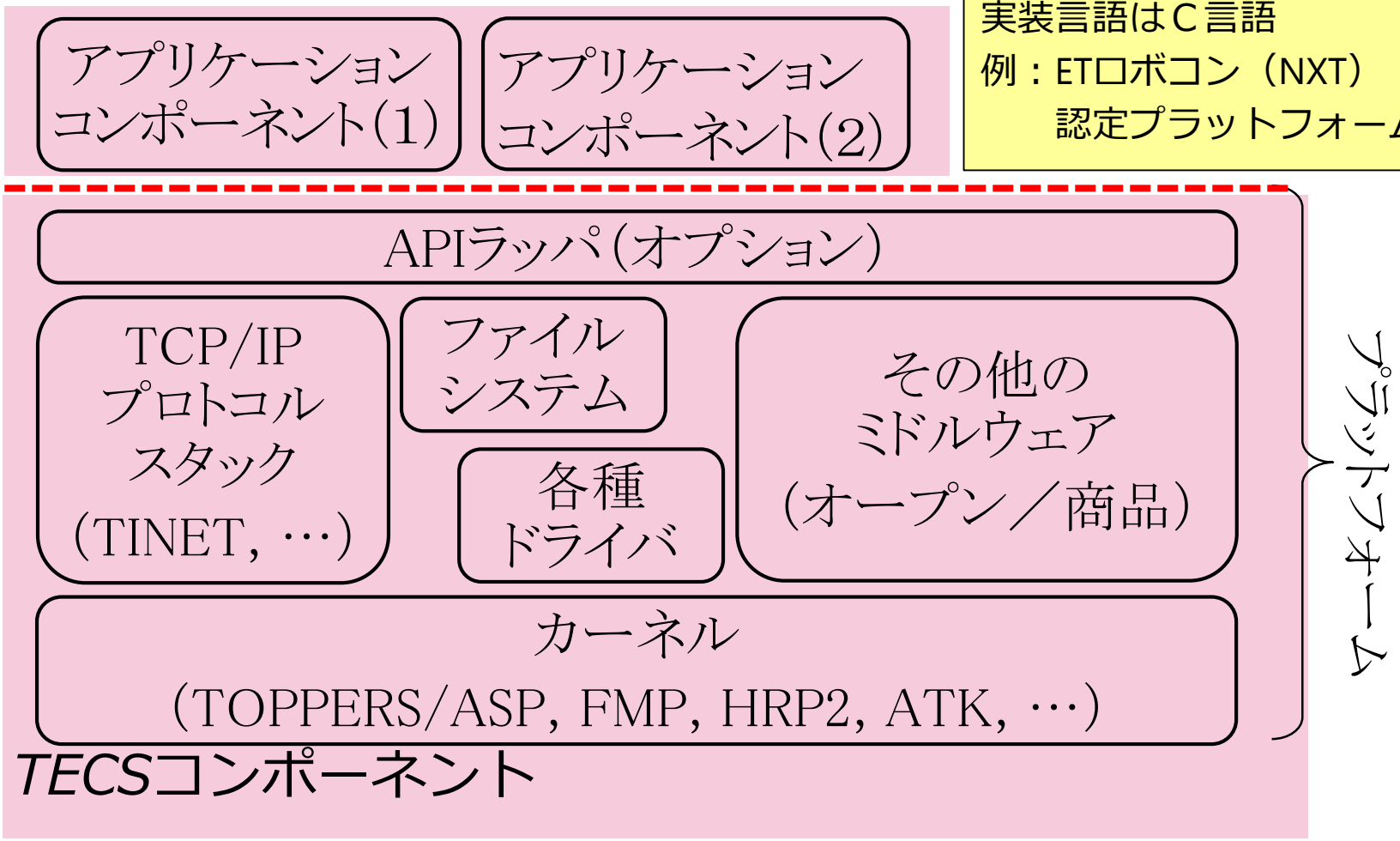
プラットフォーム

*TECS*コンポーネント

TOPPERS

TECSの適用イメージ②：システム全体への適用

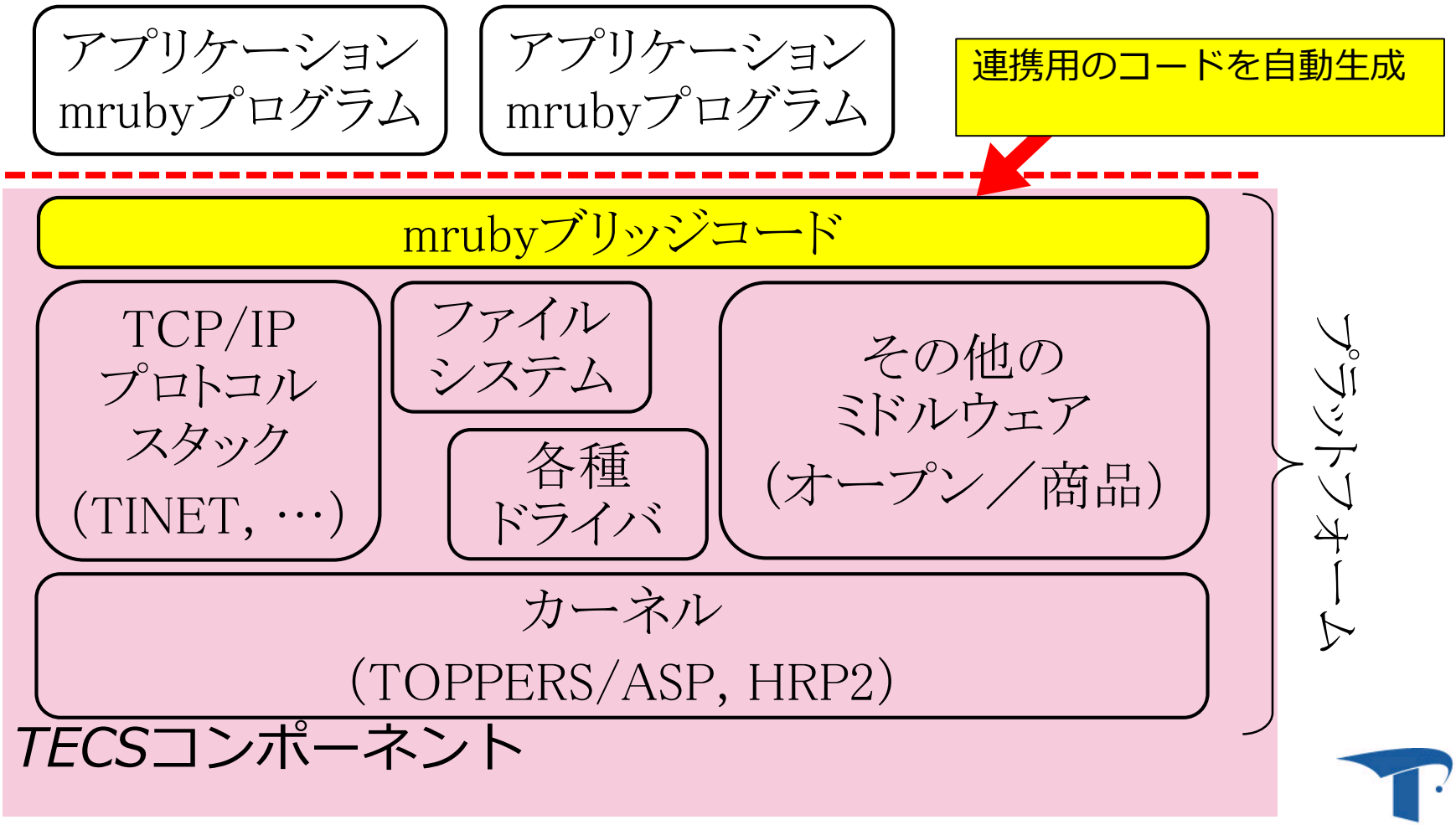
アプリケーションを含めすべての
要素をTECS仕様に準拠
（再利用性高）
実装言語はＣ言語
例：ETロボコン（NXT）
　　認定プラットフォーム



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

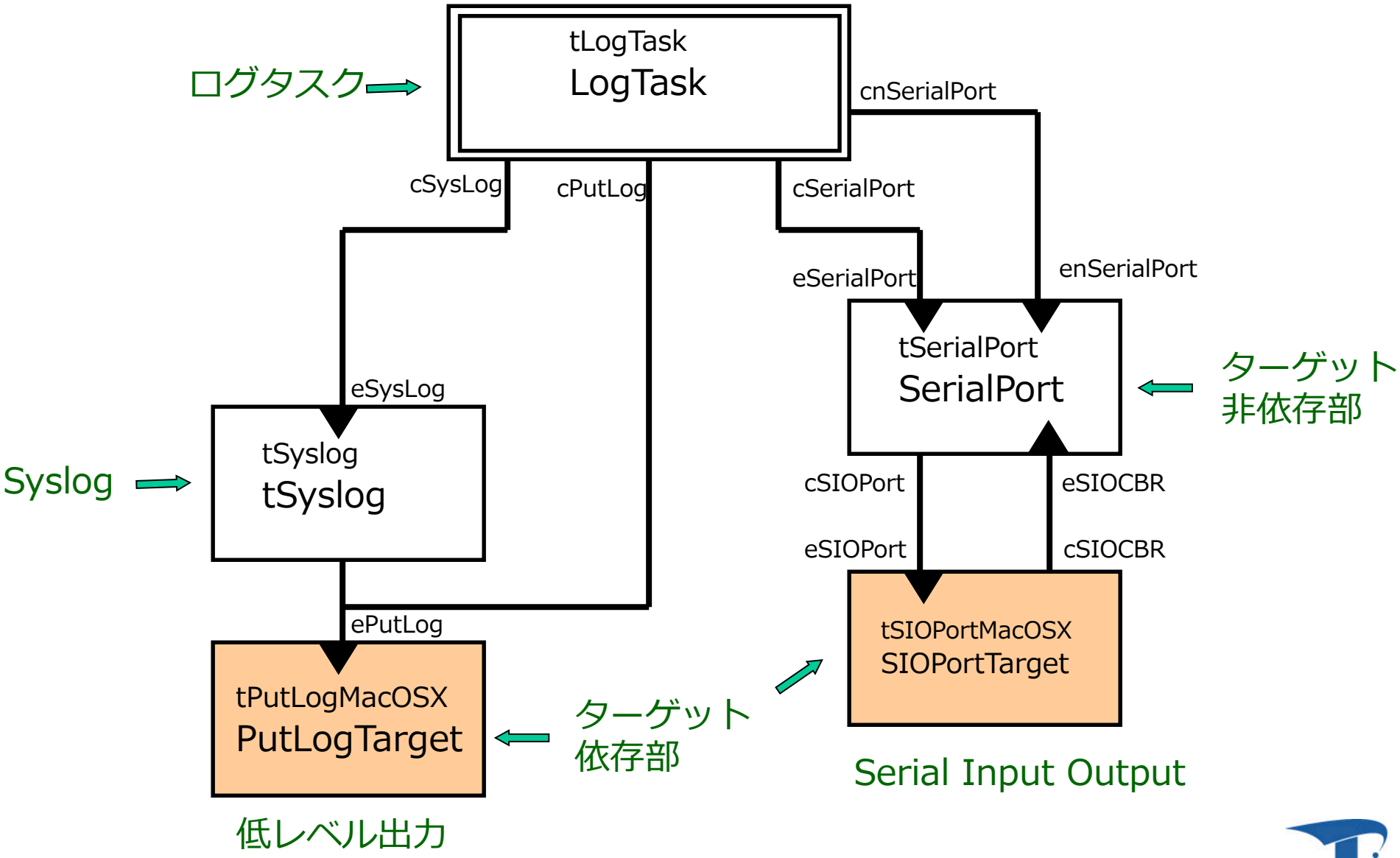


TOPPERS

TECSがASP3のSyslog, シリアルドライバ、ログタスクの実装に標準採用（予定）

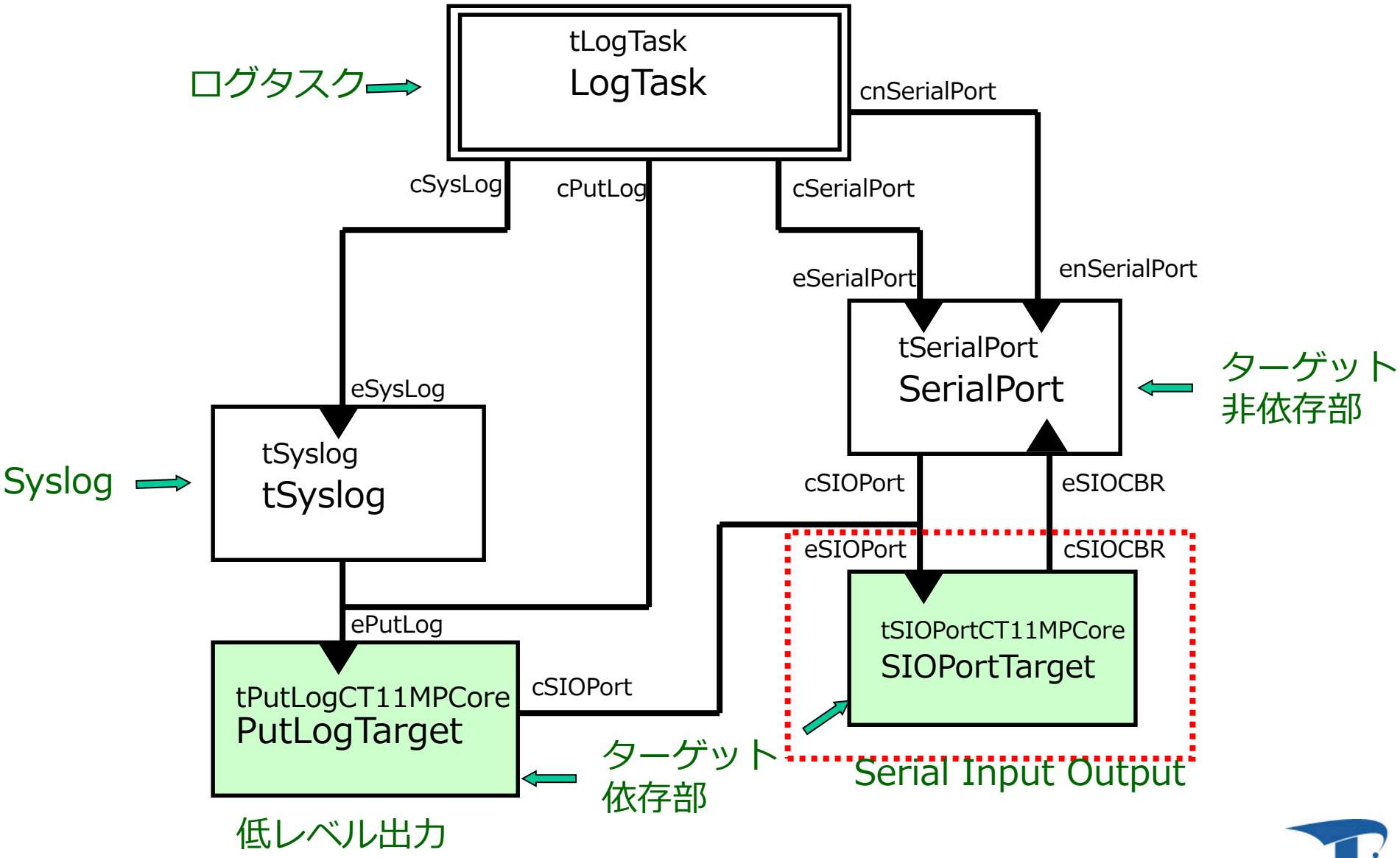
- ○ Syslogの出力先（シリアル、Bluetooth、LCD等）をプログラムを修正せずに変更可能
- △ ターゲット依存部のポーティングには、TECSの基本を覚える必要がある
  - ○TECSのすべての機能を使いこなすには、時間がかかるが、ポーティングに必要な最低限の知識であれば、それほど時間はかからない
- ○ ポーティング初心者には、何を実装すればよいかが明確になる

ASP3：ログタスク＆シリアルドライバの例（Mac）



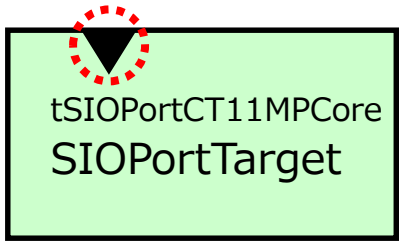
TOPPERS

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);
    }
```

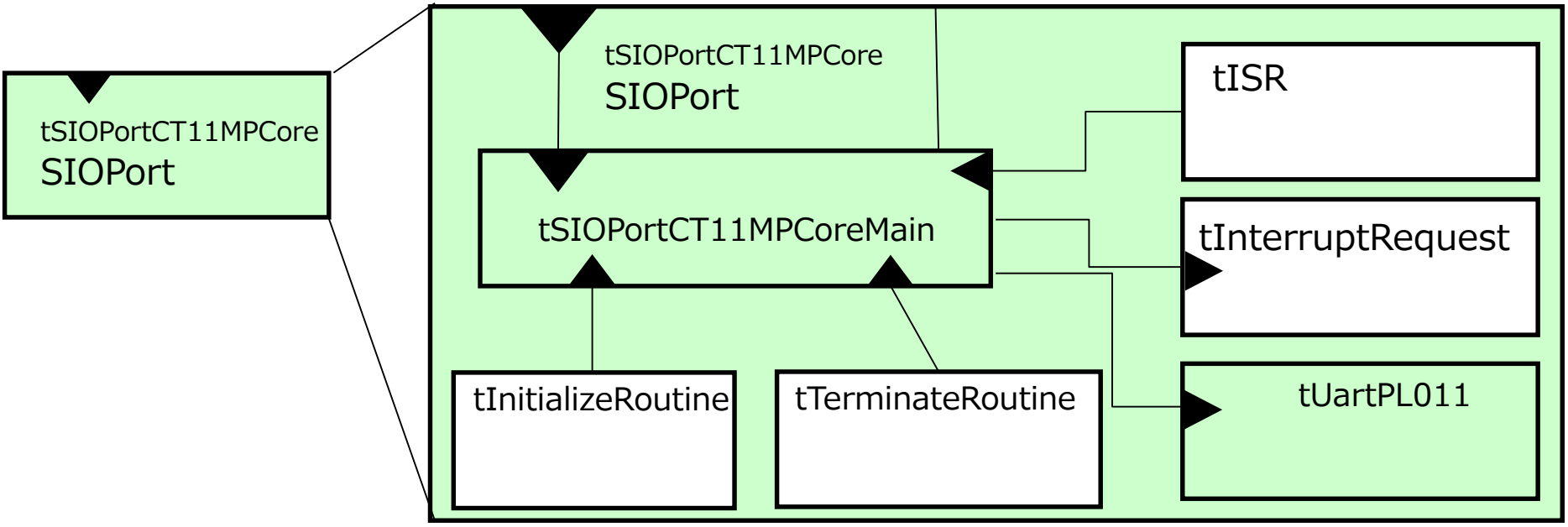
ここにドライバの
コードを実装

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


TOPPERS

TECSの利点：複合コンポーネント&cfg

- 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

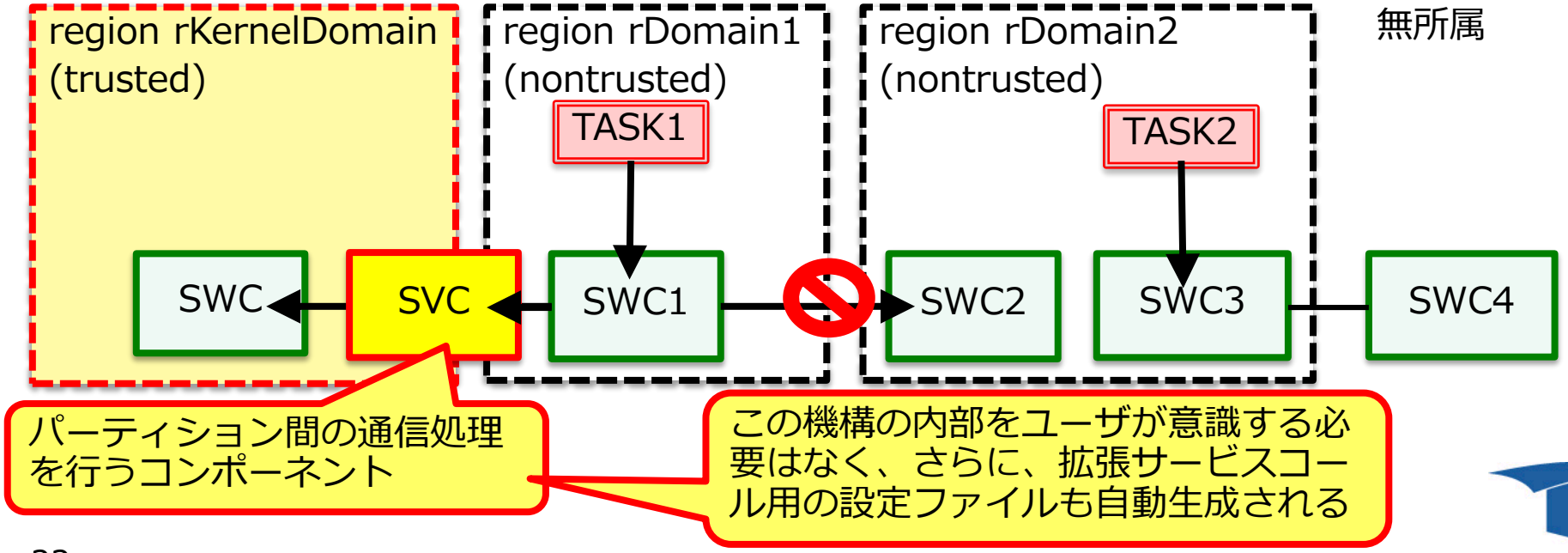


カーネルの設定ファイルの生成

```
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：メモリ保護の設定（HRP2）

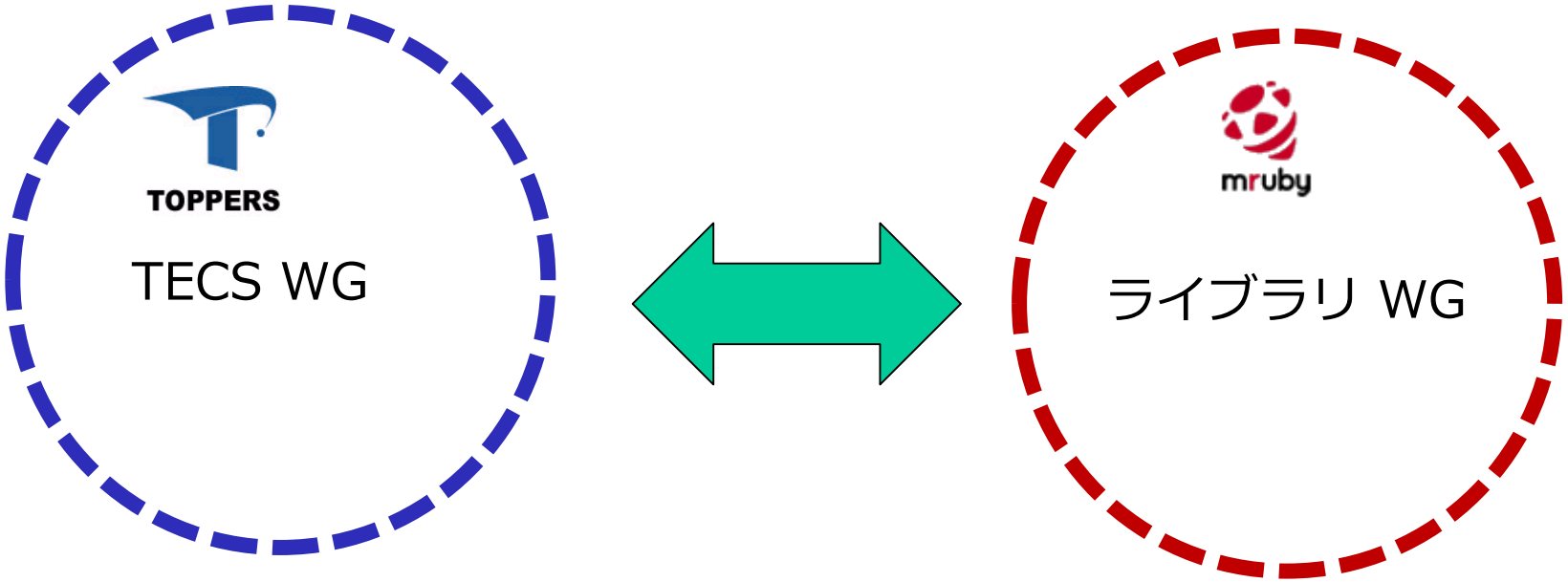
- コンポーネント記述からHRP2カーネルの設定ファイルを自動生成する
  - リージョン（グループ化）を保護ドメインに対応させる
    - リージョンに所属するタスクおよび、コンポーネント（セル）固有のデータを、対応する保護ドメインに所属させる
  - **設定ファイル**をTECSジェネレータにより出力



TOPPERS

mrubyプラットフォーム

軽量Rubyフォーラムと TOPPERSの協業



- 相互に広めあう
- 両技術を組み合わせて活用する

協業内容（2012年～）

- TOPPERS開発者会議（2012/10/21-22）
- ETスペシャルセッション（2012/11/15）
  - TECSの概要とmrubyとの連携（安積）
- TOPPERS TECS 合宿@松江（2013/3/22-23）
- ESEC（2013/5/8-10）
  - TOPPERSブース
- TOPPERSカンファレンス（2013/6/21）
- SWEST15：mrubyセッション（2013/8/22-23）
- TOPPERS TECS 合宿@福岡（2014/3/23-24）
- TOPPERS TECS 合宿@有馬（2015/3/22-23）
- SWEST17：mrubyセッション（2015/8/27-28）

- **mruby向けEV3プラットフォームの構築**
  ETロボコン認定プラットフォーム

TOPPERS

EV3用プラットフォーム

https://www.toppers.jp/tecs.html#mruby_ev3rt

- 名古屋大学により開発
- デバドラなどをLinuxなどから移植

その他のOSS…

LMS2012
(LinuxベースのVM)

leJOS
(Java VM)

MonoBrick
(.Net framework)

ev3dev
(Linux)

MINDSTORMS EV3用
TOPPERSプラットフォーム

EV3RT

今回の
プラットフォーム

TOPPERS/HRP2

TOPPERS/ASP

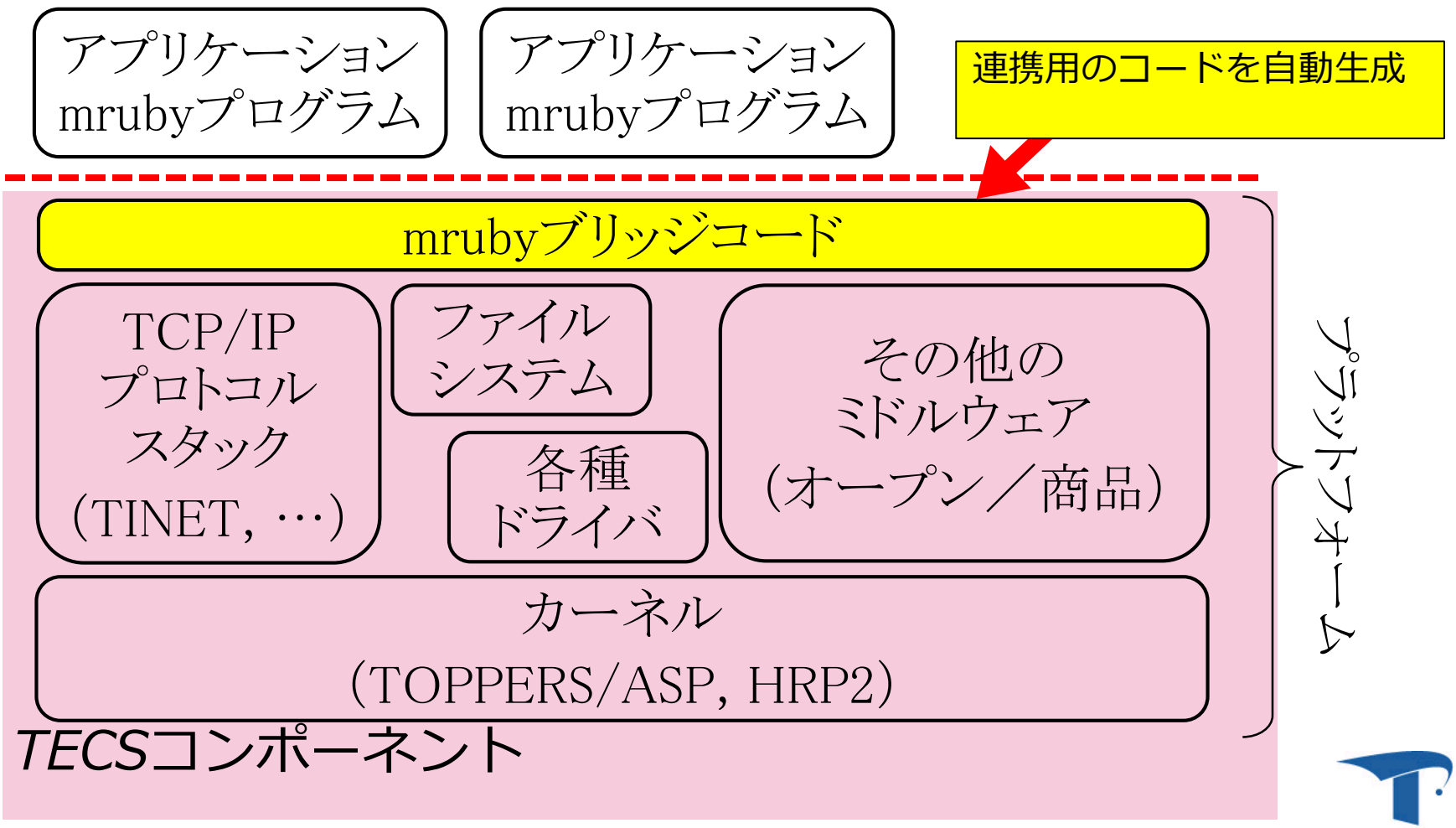
TECS仕様

- TECS WGにより開発
- mrubyの動作環境

TOPPERS

TECSの適用イメージ③：他言語（mruby）との連携

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



TOPPERS

mruby⇒TECSブリッジ

補足：mrubyプラットフォームでは、
setPowerをpower=にリネームして利用

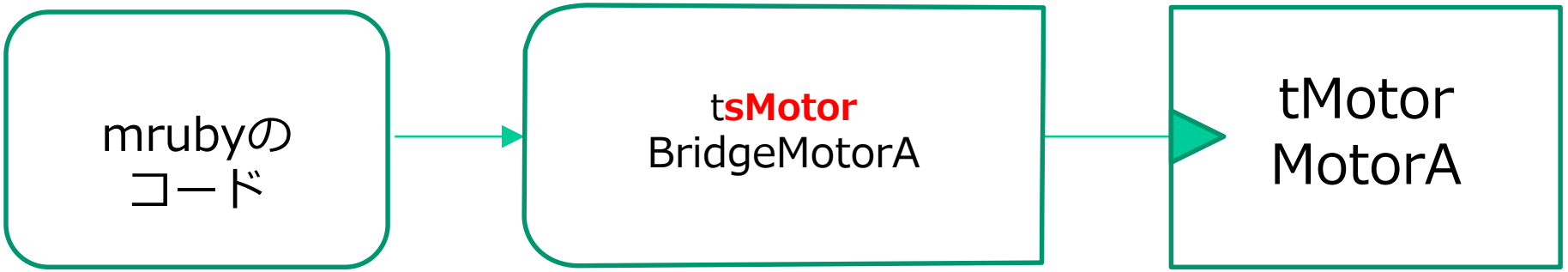
- mruby ⇒ C 言語のI/Fの自動生成
  - TECSのインタフェース定義を利用
  下記はモータを利用する例

モータのインタフェース定義

```
signature sMotor{
        ER setPower([in]int power);
        ER stop([in] bool_t brake);
        …
};
```

(呼び元=クライアント側)

(呼び先=サーバー側)



mruby オブジェクト
振る舞いはmruby で記述

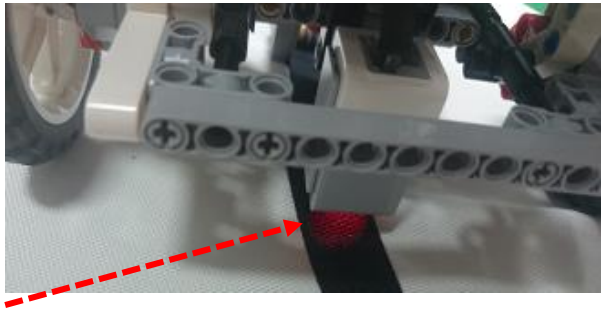
ブリッジセル
mruby オブジェクト/
TECS コンポーネント
自動生成!!

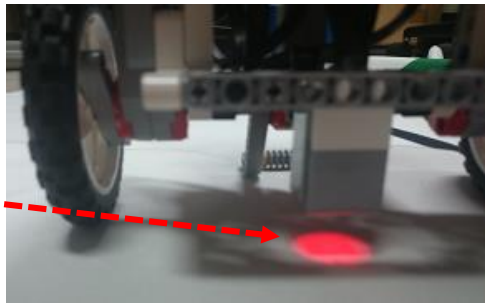
セル

TOPPERS

ev3way_sample

- 操作手順
  - 電源を入れる
  - 黒色のライン上にカラーセンサを移動
  - タッチセンサを押す：黒色の値を取得
  - 白色の上にカラーセンサを移動
  - タッチセンサを押す：白色の値を取得
    ：しっぽを下ろす
  - ライン上移動
  - タッチセンサを押す：ライントレーススタート





ev3way_sample.rb：初期化

```
begin
  LCD.puts "ev3way_sample.rb"
  LCD.puts "--- mruby version ---"
  Speaker.volume = 1
  forward = turn = 0
  # initialize sensors
  $sonar = UltrasonicSensor.new(SONAR_SENSOR)
  $color = ColorSensor.new(COLOR_SENSOR)
  $color.reflect
  $touch = TouchSensor.new(TOUCH_SENSOR)
  $gyro  = GyroSensor.new(GYRO_SENSOR)
  # initialize motors
  $motor_l = Motor.new(LEFT_MOTOR)
  $motor_r = Motor.new(RIGHT_MOTOR)
  $motor_t = Motor.new(TAIL_MOTOR)
  $motor_t.reset_count
  # Signal calibration
  LED.color = :orange
…
```

ひとつしかないもの（ポート番号指定不要）は、クラスメソッドとして直接呼び出す

ポート番号を指定して初期化（インスタンス化）

ev3way_sample.rb：黒色、白色の取得

```
# Calibration
  $black_value = color_calibration
  LCD.puts "black::#{$black_value}"
  $white_value = color_calibration
  LCD.puts "white::#{$white_value}"

  threshold = (($black_value + $white_value) / 2).round
  # wait start
  LCD.puts "Ready to start"
```

ライントレースの
基準値を計算

タッチセンサが押されるまで待つ

カラーセンサn回取得し、
平均値を取得

```
def color_calibration(n=10)
    loop {
        break if $touch.pressed?
        RTOS.delay(10)
    }
    col = 0
    n.times { col += $color.reflect}
    col = (col / n).round
    Speaker.tone(:a4, 200)
    RTOS.delay(500)
    col
end
```

ev3way_sample.rb：スタート準備

```
# wait start
    LCD.puts "Ready to start"
    loop {
        #initialize tail
        tail_control(TAIL_ANGLE_STAND_UP)
        RTOS.delay(10)
        # Touch sensor start
        break if $touch.pressed?
    }
    # reset motor encoder
    $motor_l.reset_count
    $motor_r.reset_count
    # reset Gyro sensor
    $gyro.reset

    # Signal start status
    LED.color = :green
```

しっぽの位置を指定された角度に保つ
（フィードバック制御）

```
def tail_control(angle)
        pwm = ((angle - $motor_t.count) * P_GAIN).to_i
        pwm = (pwm > PWM_ABS_MAX)  ? PWM_ABS_MAX :
        (pwm < -PWM_ABS_MAX) ? -PWM_ABS_MAX : pwm
        $motor_t.power = pwm
        $motor_t.stop(true) if pwm == 0
end
```

目標値

現在の値

ev3way_sample.rb：ライントレース

障害物まで一定の距離以下になると止まる

```
# main loop
  forward = turn = 0
  loop {
    start = RTOS.msec
    # up tail
    tail_control(TAIL_ANGLE_DRIVE)
    if sonar_alert
      forward = turn = 0
    else
      # Line trace
      turn = $color.reflect >= threshold ? 20 : -20
      forward = 30
    end
    …
}
```

```
def sonar_alert
  $sonar_counter += 1
  if $sonar_counter == 10
    distance = $sonar.distance
    $sonar_alert = distance <=
        SONAR_ALERT_DISTANCE
        && distance >= 0
    $sonar_counter = 0
  end
  $sonar_alert
end
```

サンプルでは、30に固定

カラーセンサと閾値と比較し
どちらかに曲がる
ここを変更すると、
自前のライントレースが可能

ev3way_sample.rb：倒立制御

```
# main loop
  loop {
      start = RTOS.msec
     …
     # call balance_control API
     pwm_l, pwm_r = Balancer.control(
     forward.to_f,
     turn.to_f,
     $gyro.rate.to_f,
     GYRO_OFFSET,
     $motor_l.count.to_f,
     $motor_r.count.to_f,
     Battery.mV.to_f)
     $motor_l.stop(true) if pwm_l == 0
     $motor_l.power = pwm_l
     $motor_r.stop(true) if pwm_r == 0
     $motor_r.power = pwm_r
     wait = 4 - (RTOS.msec - start)
     RTOS.delay(wait) if wait > 0
  }
```

バランサの返り値が2つ

C言語で実装されたバランサを呼び出す

4ミリ秒周期で実行
現状1ミリ秒程度で処理完了
mubyでも十分制御可能

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