TECS 対応版
TOPPERS/ASP カーネルについて

名古屋大学
安積卓也

目次

- TECS簡易パッケージ構造
- コンポーネント記述
- ASP+TECS
  - コンポーネント版のsample1.c（ASP）の実行例
  - カーネルオブジェクト
  - ログタスク＆シリアルドライバの例

TECS簡易パッケージ構造

- 入手先:http://www.toppers.jp/tecs.html
- tecs_package
  - asp+tecs ⇐ コンポーネント版のASP(Mac,Skyeye)
    - -doc
      - -asp+tecs_api.txt ⇐ コンポーネント化したカーネルオブジェクトのAPIリファレンス
    - -README.txt ⇐ サンプルの実行方法を記載
  - bin
    - -skyeye.exe ⇐ Skyeyeの実行ファイル
    - -tecgen.exe ⇐ TECSジェネレータ
  - tecsgen ⇐ TECSジェネレータのソース
  - tutorial ⇐ Cygwin上で動作するチュートリアル
  - README.txt

準備(TECSのインタフェースジェネレータ)

注:tecs_package/bin/tecsgen.exeを利用する場合下記の作業は不要

- Ruby
- Racc
  - 最新版をhttp://i.loveruby.net/ja/projects/racc/からダウンロード
  - インストール方法
    racc-1.4.5-all.tar.gzを展開したディレクトリで下記のコマンドを実行
    :ruby setup.rb config
    :ruby setup.rb setup
    :ruby setup.rb install
- TECSのジェネレータのインストール方法

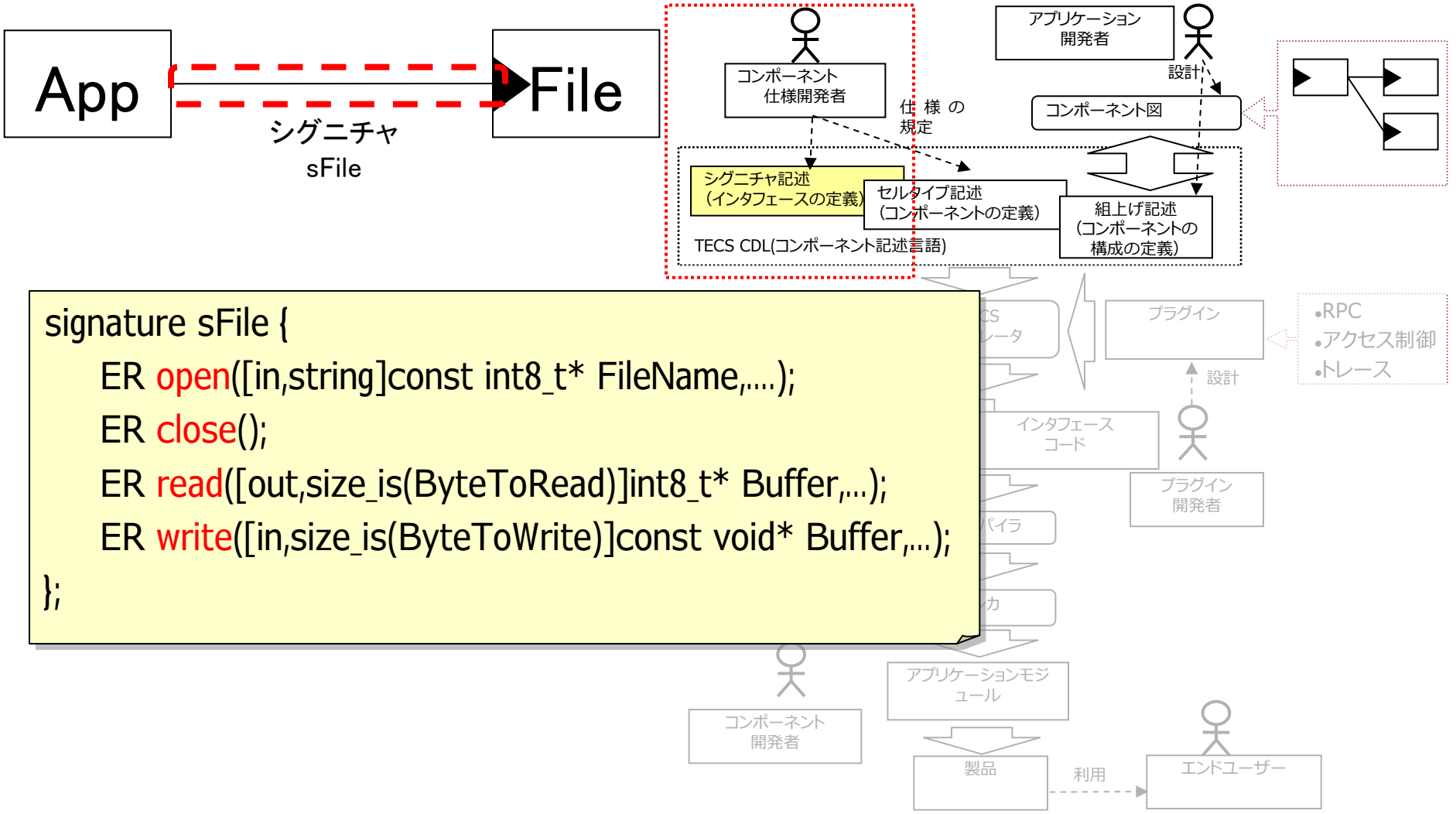
  tecs_package/tecsgenで

  :source set_env.sh ⟸ TECSジェネレータの環境設定

  :make ⟸ TECSジェネレータのコンパイル

  :tecsgen ⟸ インストールの確認

TECS：インタフェース定義（シグニチャ）

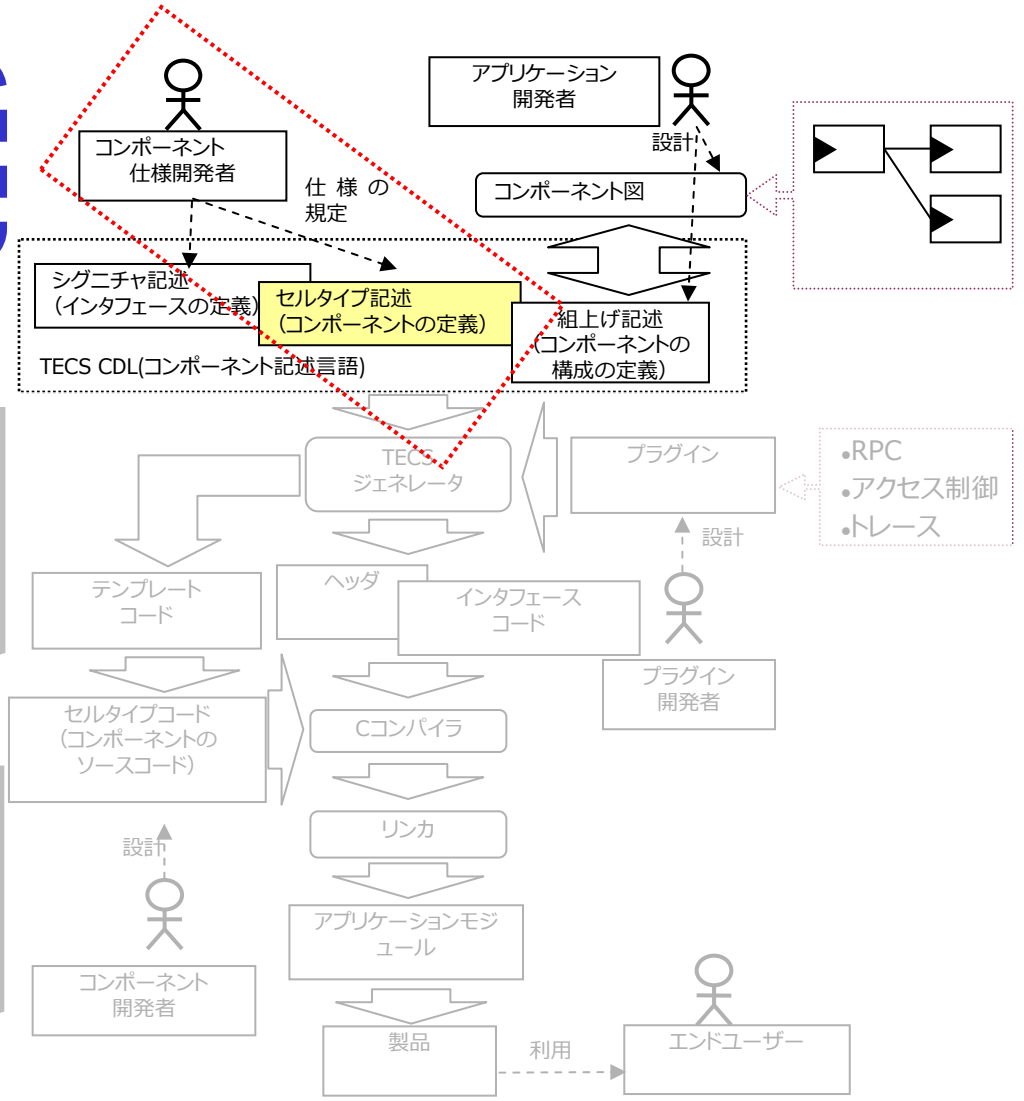



```
signature sFile {
    ER open([in,string]const int8_t* FileName,....);
    ER close();
    ER read([out,size_is(ByteToRead)]int8_t* Buffer,...);
    ER write([in,size_is(ByteToWrite)]const void* Buffer,...);
};
```

TECS：コンポーネントの定義（セルタイプ）

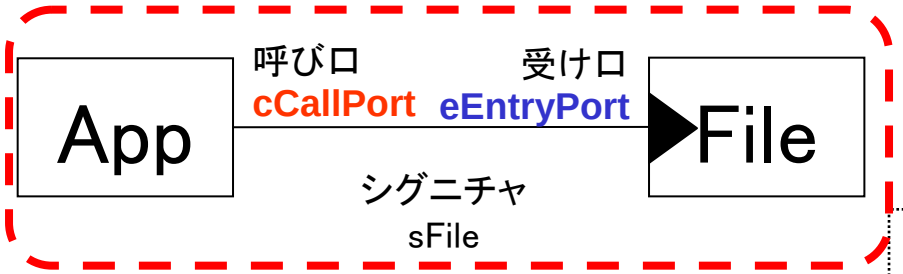

```
celltype tApp{
    call sFile cCallPort;
};
```

```
celltype  tFile{
    entry sFile eEntryPort;
};
```

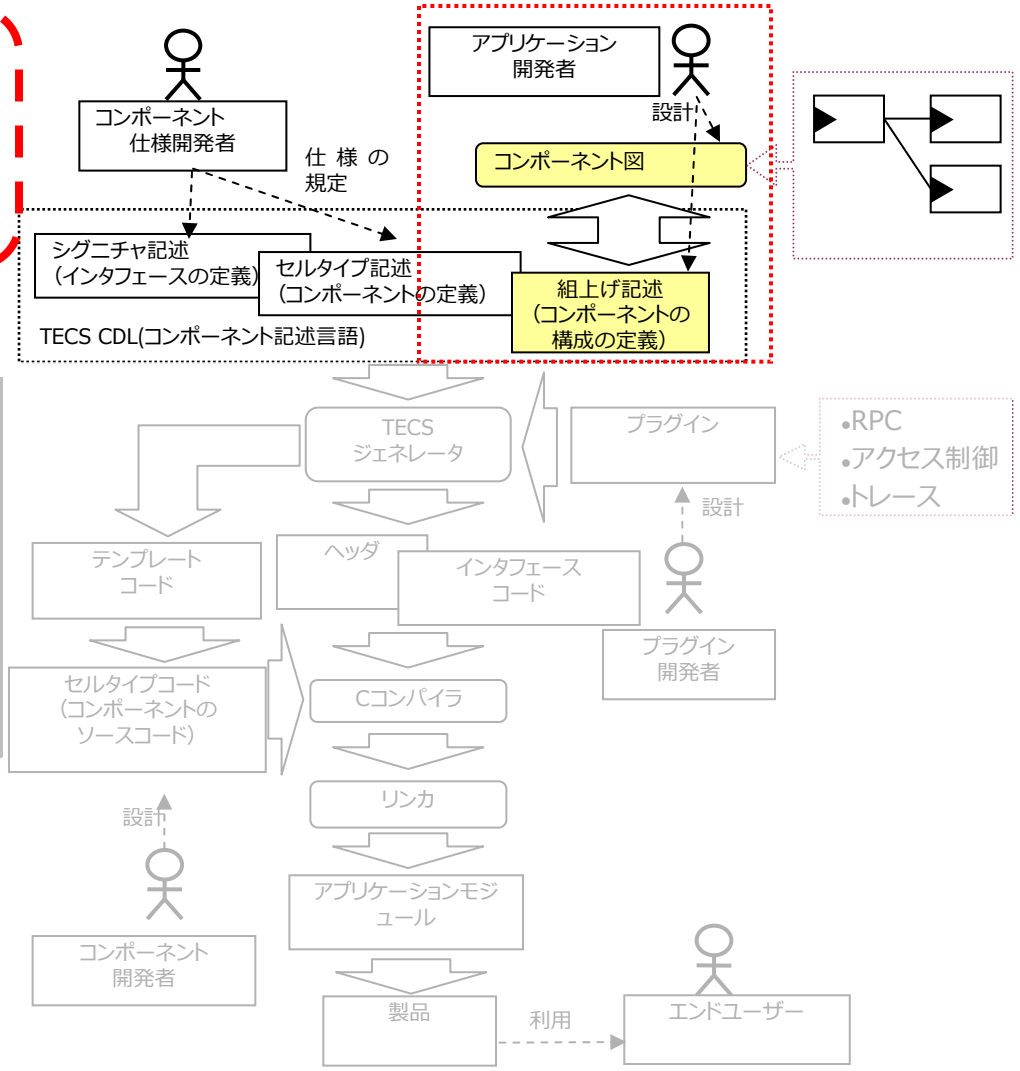


TOPPERS

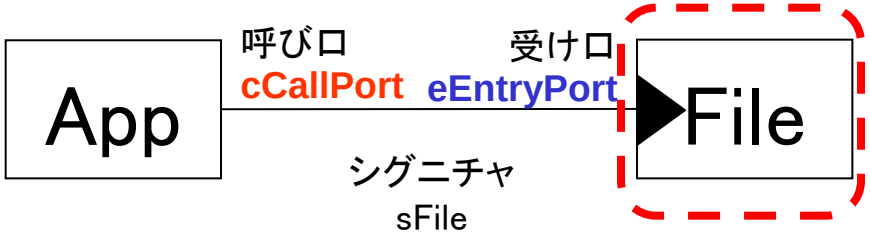
TECS：コンポーネントの構成の定義（組み上げ）

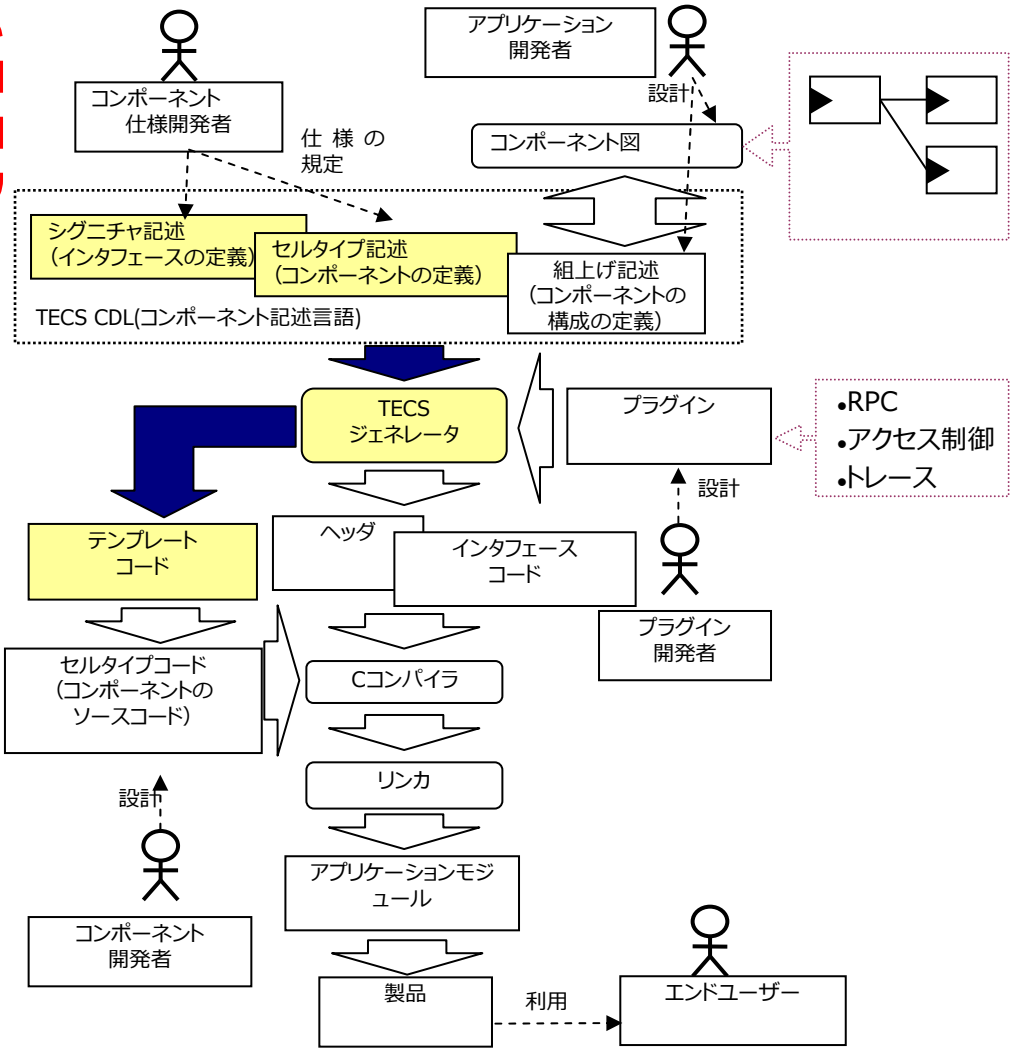



```
cell tFile File{
};
cell tApp App{
        cCallPort = File.eEntryPort;
};
```



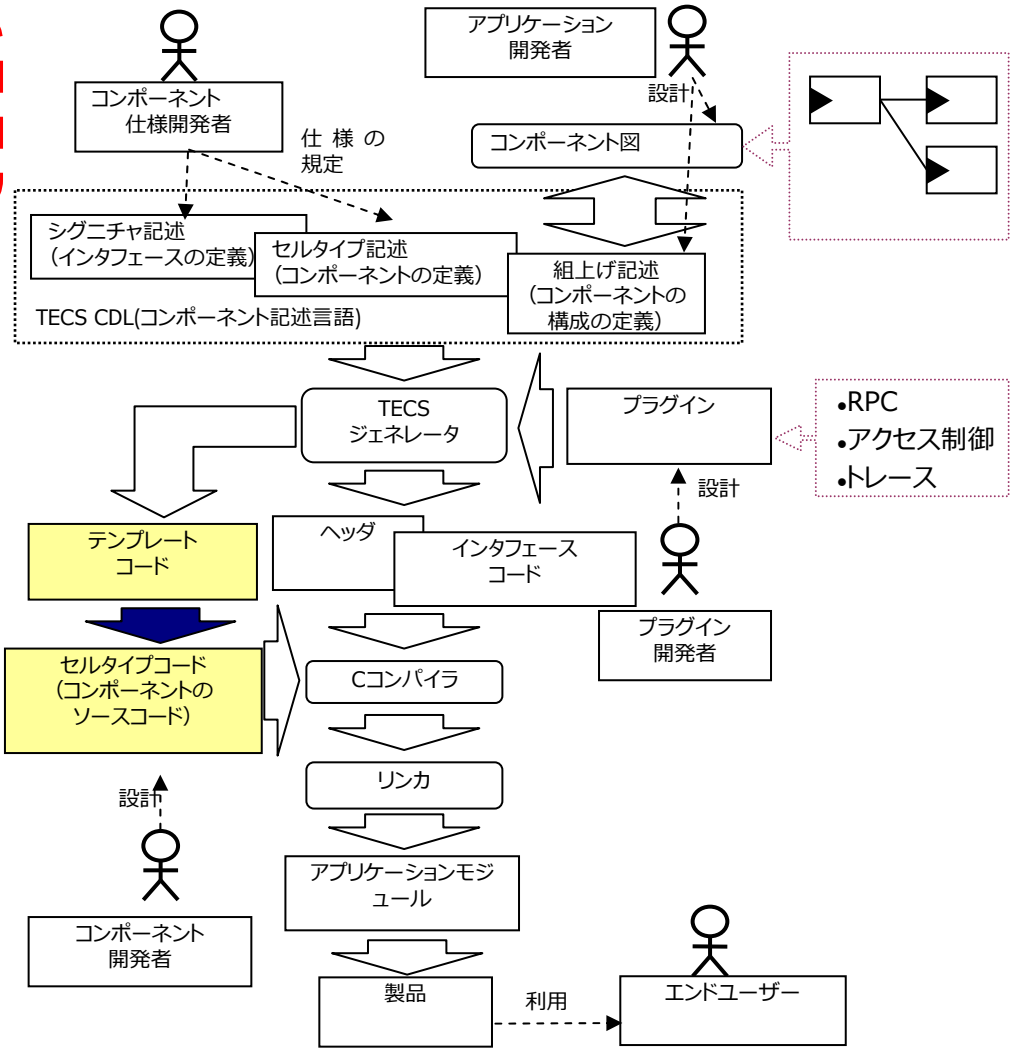
TECSジェネレータ:テンプレートコード





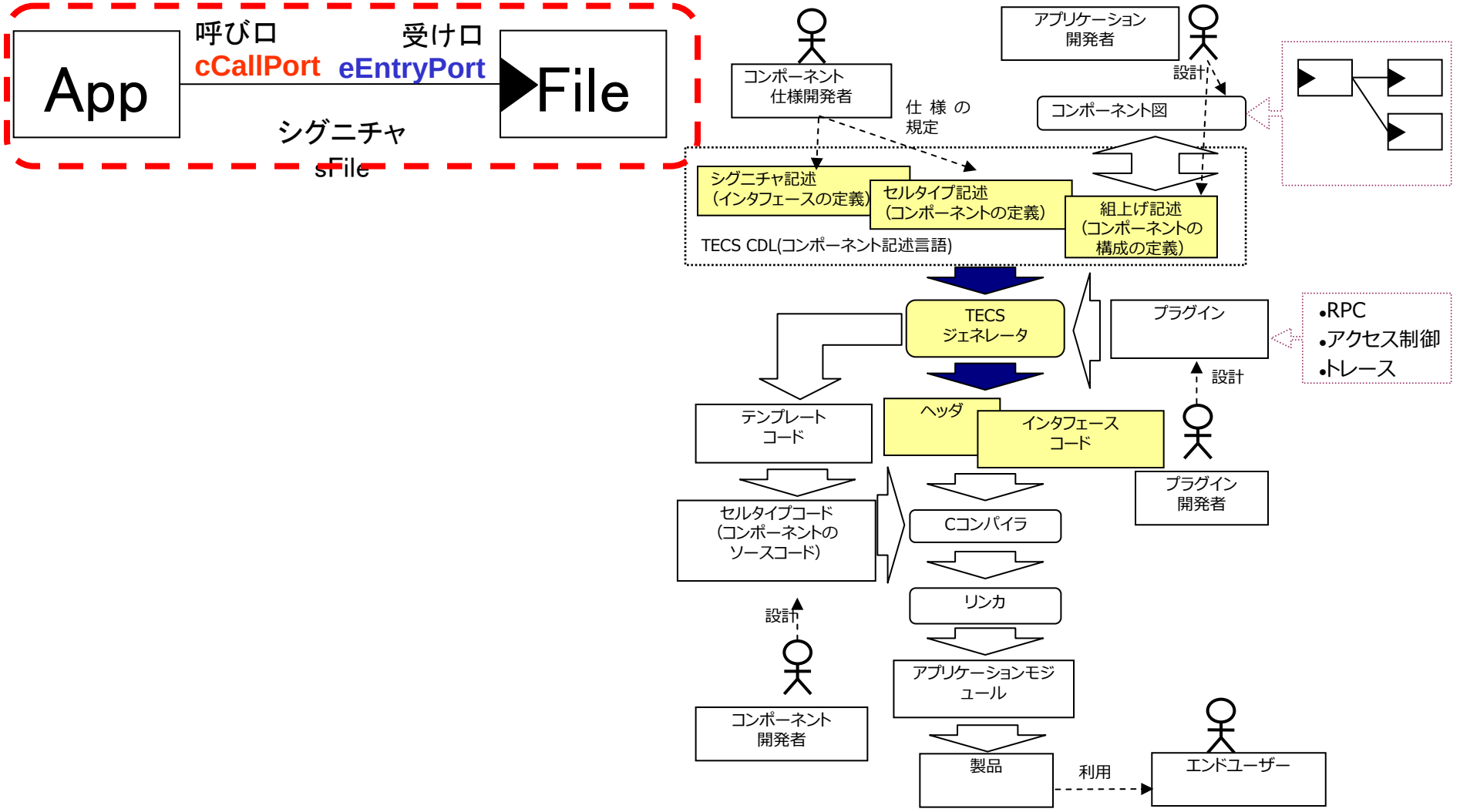

```
テンプレートコード[tFile.c]
#include “tFile_tecsgen.h”
ER      eEntryPort_open( … )
{
        /* ここにコードを書く */
}
ER      eEntryPort_close( …)
                :
```

TECS：コンポーネントのソースコードの実装

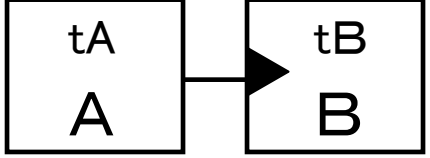



```
[tFile.c]
#include “tFile_tecsgen.h”
ER        eEntryPort_open( … )
{
    コンポーネントの振舞いを記述
}
ER        eEntryPort_close( …)
                :
```

TECSジェネレータ:ヘッダ インタフェースコード



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

tB_eEnt_func1_skel
tB_eEnt_func2_skel
tB_eEnt_func3_skel

受け口スケルトン関数

```
ER      tB_eEnt_func1_skel( struct
                    tag_sSig1_VDES *epd)
{
    struct tag_tB_eEnt_DES *lepd
        = (struct tag_tB_eEnt_DES *)epd;
    return tB_eEnt_func1( lepd->idx );
}
```

cCall1_func1( )

cCall1_func2( )

受け口
ディスクリプタ

&tB_eEnt_MT

&tB_B_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 cCall1_func1( ) ¥
                tA_cCall1_func1( p_cellcb )
#define tA_cCall1_func1( p_that ) ¥
            (p_that)->cCall1->VMT->¥
            func1( (p_that)->cCall1 )
```

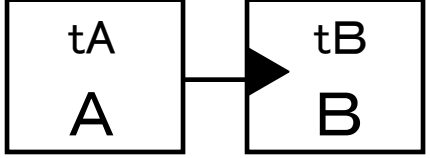
呼び側のセルＣＢ

```
typedef struct tag_tA_CB {
    /* call port */
    struct tag_sSig1_VDES *cCall1;
    struct tag_sSig2_VDES *cCall2;
} tA_CB;
```

```
ER          eEnt_func1( tB_IDX idx)
{
    ER      ercd_ = E_OK;
    tB_CB *p_cellcb;
    if( tB_VALID_IDX( idx ) ){
        p_cellcb = GET_CELLCB(idx);
    }else{
        return E_ID;
    }
     /*処理*/
    return ercd_;
}
```


TOPPERS

結合の実装構造の標準形



呼び側

受け側

cCall1_func1( )

cCall1_func2( )

受け口
ディスクリプタ

&tB_eEnt_MT

&tB_B_CB

受け口
関数テーブル

tB_eEnt_func1_skel

tB_eEnt_func2_skel

tB_eEnt_func3_skel

受け口関数テーブルへの
ポインタ

受け側のセルＣＢ

受け口スケルトン関数

```
ER      tB_eEnt_func1_skel( struct
                    tag_sSig1_VDES *epd)
{
    struct tag_tB_eEnt_DES *lepd
        = (struct tag_tB_eEnt_DES *)epd;
    return tB_eEnt_func1( lepd->idx );
}
```

受け口関数

```
ER          eEnt_func1( tB_IDX idx)
{
    ER      ercd_ = E_OK;
    tB_CB *p_cellcb;
    if( tB_VALID_IDX( idx ) ){
        p_cellcb = GET_CELLCB(idx);
    }else{
        return E_ID;
    }
     /*処理*/
    return ercd_;
}
```

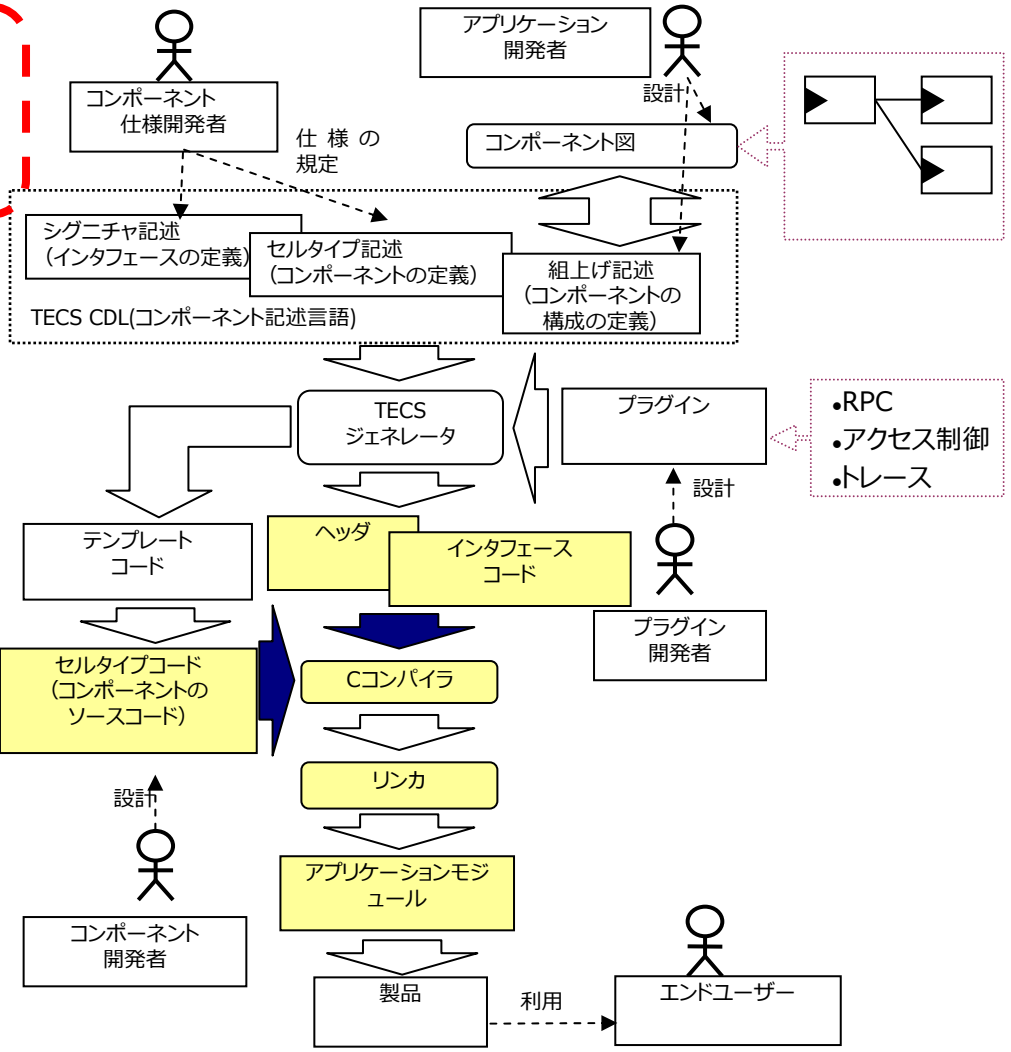
```
/* 呼び口関数マクロ（短縮形） */
#define cCall1_func1( ) ¥
            tA_cCall1_func1( p_cellcb )
#define tA_cCall1_func1( p_that ) ¥
        (p_that)->cCall1->VMT->¥
        func1(p_that->cCall1 )
```

呼び側のセルＣＢ

```
typedef struct tag_tA_CB {
    /* call port */
    struct tag_sSig1_VDES *cCall1;
    struct tag_sSig2_VDES *cCall2;
} tA_CB;
```


TOPPERS

コンパイル



TOPPERS

genにジェネレータが自動生成するファイル

- Makefile.depend ← 依存関係
- Makefile.tecsgen ← 自動生成したファイル用のMakefile
- Makefile.templ ← Makefileのテンプレート
- global_tecsgen.h ← 型の定義など
- sFile_tecsgen.h ← 各シグニチャの定義

各セルタイプで自動生成されるファイル

- tFile_factory.h ← ファクトリ用ヘッダ
- tFile_templ.c ← テンプレート
- tFile_tecsgen.c ← セルの結合コード
- tFile_tecsgen.h

- tApp_factory.h
- tApp_templ.c
- tApp_tecsgen.c
- tApp_tecsgen.h

- tmp_C_src.c
- tmp_tecs.h

← 中間ファイル ジェネレータが使用

ASP+TECS

コンポーネント版のsample1（ASP）の実行

- コンポーネント化された部分
  - カーネルオブジェクト
  - ログタスク
  - syslog
  - シリアルドライバ
- 準備項目
  - tecsgenにシンボリックリンクを張る
    - tecs_package/asp+tecs/で
      - tecspackage/bin/tecsgen.exeを利用する場合
        :ln -s ../bin/tecsgen tecsgen
      - tecspackage/tecsgenを利用する場合
        :ln -s ../tecsgen/tecsgen/tecsgen tecsgen

arm-elf-gccにパスを通してください

  - arm-elf-* （コンパイラ等）（skyeye版を実行する場合）
    - GNUWING:http://www.embedded.jp/gnuwing/index.html からダウンロードできます

ASP+TECSの構成

- asp+tecs
  - include
    - kernel.cdl ⟸ カーネルオブジェクトのコンポーネント記述
  - doc
    - asp+tecs_api.txt ⟸ カーネルオブジェクトAPIドキュメント
  - syssvc
    - tSerialPort.cdl ⟸ シリアルポート
    - tLogTask.cdl ⟸ ログタスク
  - tecs_kernel ⟸ カーネルオブジェクトのコンポーネント化実装コード
  - target ⟸ ターゲット依存部
    - at91skyeye_gcc/macosx_gcc
      - target_syssvc_decl.cdl
      - target_syssvc_inst.cdl
      - tSIOPort*.cdl
      - tPutLog*.cdl

コンポーネント版sample1の生成方法

注:tecs_package/asp+tecs/sample1_skyeyeを利用する場合下記の作業は不要

tecs_package/asp+tecs/で

:mkdir obj

:cd obj

:mkdir sample

:cd sample

:../../configure -T at91skyeye_gcc

ターゲット名
Macの場合は
macosx_gcc

Makefile
tSample1.c,tSample1.h
tSample.cfg,tSample1.cdl
の生成

:make tecs ← ジェネレータの実行

:make depend ← コンフィギュレータの実行＆依存関係の生成

:make ← コンパイル

:cp ../../target/at91skyeye_gcc/skyeye.conf ./ ← Skyeyeの場合のみ

TOPPERS

tSample1のMakefileの説明1
tecsgen.exeを利用する場合の注意点

```
APPLNAME = tSample1
APPL_CDL = $(APPLNAME).cdl      ← コンポーネント記述
TECSGEN_CPP = 'gcc -E -D TECS'  ← ジェネレータに渡すCPP
#cygwinのgccがシンボリックリンクの場合は、
#下記のようにシンボリックリンク先(gcc-3,gcc-4など)を直接指定する必要がある.
#TECSGEN_CPP = 'gcc-3 -E -D TECS'
#TECSGEN_CPP = 'gcc-4 -E -D TECS'
TECSGEN = $(SRCDIR)/tecsgen.exe -c $(TECSGEN_CPP)  -k euc
#                                  ↖ ジェネレータの実行コマンド
#  TECSインタフェースジェネレータの実行
#
tecs.timestamp: $(APPL_CDL) $(TECS_IMPORTS)
        $(TECSGEN) -R -D TECS -D TECS_CPP ¥
                        $(INCLUDES) $(APPL_CDL)
      touch tecs.timestamp
.PHONY: tecs
tecs: tecs.timestamp    ← tecs.timestampと比較して実行するか判断する
```

Makefileの説明2

- tecs_package/tecsgenを利用する場合
  下記のようにMakefileを変更する

```
#tecsgen.exeを利用する場合
TECSGEN_CPP = 'gcc -E -D TECS'
#cygwinのgccがシンボリックリンクの場合は、
#下記のようにシンボリックリンク先(gcc-3,gcc-4など)を直接指定する必要がある.
#TECSGEN_CPP = 'gcc-3 -E -D TECS'
#TECSGEN_CPP = 'gcc-4 -E -D TECS'
#TECSGEN = $(SRCDIR)/tecsgen.exe -c $(TECSGEN_CPP)
```

TECSGENをコメントアウト

```
#tecsgen.rb (ruby + racc)を利用する場合は下記のRUBYLIBとTECSGENの定義を利用する
RUBYLIB = $(SRCDIR)/tecsgen/tecsgen
TECSGEN =$(RUBY) $(SRCDIR)/tecsgen/tecsgen/tecsgen.rb -L $(RUBYLIB)
```

こちらのRUBYLIBとTECSGENを利用

コンポーネント版tSample1の実行

:../../bin/skyeye.exe -e asp.exe ← Skyeyeの実行

```
TOPPERS/ASP Kernel Release 1.4.0 for AT91SKYEYE(ARM) (Jun  2 2009, 09:37:14)
Copyright (C) 2000-2003 by Embedded and Real-Time Systems Laboratory
                            Toyohashi Univ. of Technology, JAPAN
Copyright (C) 2004-2009 by Embedded and Real-Time Systems Laboratory
            Graduate School of Information Science, Nagoya Univ., JAPAN

System logging task is started.
Sample program starts.
task1 is running (001).   |
task1 is running (002).   |
task1 is running (003).   |
task1 is running (004).   |
task1 is running (005).   |
task1 is running (006).   |
task1 is running (007).   |
```

↖ Ctrl-Cで終了

TOPPERS

カーネルオブジェクトのコンポーネント化

- カーネルオブジェクト一覧

アクティブなセルタイプ

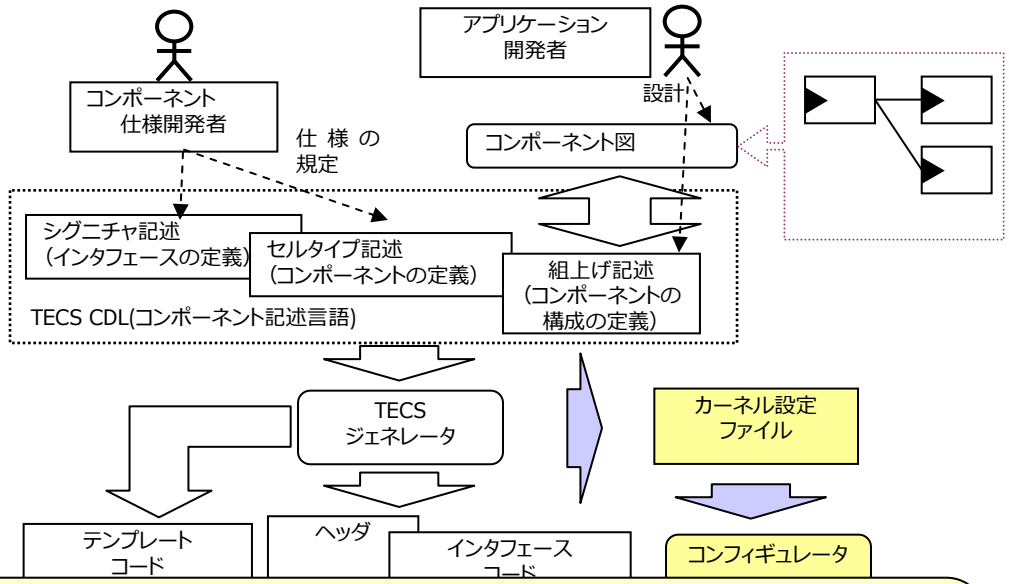
- タスク
- 初期化処理ルーチン
- 終了処理ルーチン
- 割込みサービスルーチン
- 割込み要求ライン
- 周期ハンドラ
- アラームハンドラ

その他のセルタイプ

- セマフォ
- イベントフラグ
- データキュー
- 優先度データキュー
- 固定長メモリプール
- カーネル

カーネルオブジェクトのコンポーネント化

- カーネルオブジェクト
  - タスク
  - セマフォ
  - データキュー
- 利点
  - 静的APIを自動生成
  - 部品化を促進



タスク

```
CRE_TSK(TSKID_tTask_LogTask_Task,
        { TA_ACT, &tTask_CB_tab[0], tTask_start_task, 3, 131072, NULL });
```

周期ハンドラ

```
CRE_CYC( CYCHDLRID_tCyclicHandler_CyclicHandler,
{ TA_NULL, &tCyclicHandler_CB_tab[0], tCyclicHandler_start, 2000, 0 } );
```

セマフォ

```
CRE_SEM(SEMID_tSemaphore_SerialPort1_ReceiveSemaphore, { TA_NULL, 0, 1 });
```

タスクのシグニチャ記述(一部抜粋)

```
/* タスク本体を呼び出すためのシグニチャ*/
signature sTaskBody {
        void    main(void);
};
/* タスク例外処理ルーチン本体を呼び出すシグニチャ*/
signature sTaskExceptionBody {
        void    main([in] TEXPTN pattern);
};
/* タスクを操作するためのシグニチャ（タスクコンテキスト用）*/
signature sTask {
        ER              activate(void);
        ER_UINT         cancelActivate(void);
        ER              terminate(void);
        …
};
/*  タスクを操作するためのシグニチャ（非タスクコンテキスト用）    */
signature siTask {
        ER              activate(void);
        ER              wakeup(void);
        …
};
```

tecs_package/asp+tecs/include/kernel.cdl

タスクのセルタイプ定義(前半)

```
[active]
celltype tTask {
    [inline]     entry sTask   eTask;    /* タスク操作(タスクコンテキスト用)*/
    [inline]     entry siTask  eiTask;   /* タスク操作(非タスクコンテキスト用)*/
                 call sTaskBody          cBody;  /* タスク本体 */
    [optional] call sTaskExceptionBody        cExceptionBody;
                                        /* タスク例外処理ルーチン本体 */

    attr{
    };
    factory {
    };
    FACTORY {
    };
};
```

[inline]で指定された受け口はinline関数として実装する
生成されるテンプレートコードもinline関数が生成される

[optional]で指定された呼び口は未結合を許す
セルタイプコード(C言語の実装コード)では、
呼び出す前に結合されているかの確認をする必要がある

タスクのセルタイプ定義(後半)

```
celltype tTask {
      ...
  attr{
    ID      id = C_EXP("TSKID_$id$");
    [omit] ATR              taskAttribute = C_EXP("TA_NULL");
    [omit] ATR              exceptionAttribute = C_EXP("TA_NULL");
    [omit] PRI              priority;
    [omit] SIZE             stackSize;
  };
  factory {  ⟸  各セルごとに出力
    write("tecsgen.cfg",
          "CRE_TSK(%s, { %s, $cbp$, tTask_start_task, %s, %s, NULL });",
          id, task_attribute, priority, stack_size);
    };
  FACTORY {  ⟸  セルタイプで一度だけ出力
    write("tecsgen.cfg", "#include ¥"$ct$_tecsgen.h¥"");
  };
};
```

C_EXP(“”);はC言語での表現
コンポーネント記述で解釈しないことを示す

ファクトリ

- セル（コンポーネント）生成時に静的APIなどを開発者が意図的に出力するために使用する

tecs_package/asp+tecs/include/kernel.cdl

```
celltype tTask {
    ...
factory {
    write("tecsgen.cfg",
        "CRE_TSK(%s, { %s, $cbp$, tTask_start_task, %s, %s, NULL });",
        id, taskAttribute, priority, stackSize);
};
FACTORY {
    write("tecsgen.cfg", "#include ¥"$ct$_tecsgen.h¥"");
};
};
```

出力したいファイル名

出力内容
printf形式で出力

ファクトリでの名前置換

id … セルタイプ名とセル名を '_' で連結したものに置換
$cell$ … セル名に置換
cb … セルの CB の C 言語名に置換
cbp … セルのCB へのポインタ(CB が生成されない場合はNULL に置換)
cb_proto … セルの CB の C 言語名（プロトタイプ宣言用）に置換
ct … セルタイプ名に置換
idx … セルの CB の IDX (idx_is_id の場合は整数、そうでない場合は CB へのポインタ)に置換
ID … セルの ID(idx_is_id の場合 IDX に一致)に置換
$$ … $ に置換

例えば、セルタイプ名**tTask**、セル名**MainTask**の場合

```
ID  id = C_EXP("TSKID_$id$");
write("tecsgen.cfg","CRE_TSK(%s, { %s, $cbp$, tTask_start_task, %s, %s, NULL });",
            id, task_attribute, priority, stack_size);
```

tecsgen.cfg

```
CRE_TSK(TSKID_tTask_MainTask, { TA_ACT, &tTask_CB_tab[1],
        tTask_start_task, MAIN_PRIORITY, STACK_SIZE, NULL });
```

セルタイプ名　**セル名**

tecsgen.cfg

tSample1.cfg

```
INCLUDE("target_timer.cfg");
INCLUDE("tecsgen.cfg");
INCLUDE("syssvc/banner.cfg");

#include "tSample1.h"
#ifdef CPUEXC1
DEF_EXC(CPUEXC1, { TA_NULL, cp
#endif /* CPUEXC1 */
```

```
#include "cb_type_only.h"
#include "tTask_tecsgen.h"
CRE_TSK(TSKID_tTask_LogTask_Task, { TA_ACT, &tTask_CB_tab[0],
            tTask_start_task, 3, 131072, NULL });
DEF_TEX(TSKID_tTask_LogTask_Task, { TA_NULL,
             tTask_start_exception });
...
#include "tInitializeRoutine_tecsgen.h"
ATT_INI({ TA_NULL, NULL, tInitializeRoutine_start });
#include "tTerminateRoutine_tecsgen.h"
ATT_TER({ TA_NULL, NULL, tTerminateRoutine_start });
#include "tISR_tecsgen.h"
ATT_ISR({ TA_NULL, NULL, 2, tISR_start, 1 });
CFG_INT( 2,{ TA_NULL, -2});
#include "tCyclicHandler_tecsgen.h"
CRE_CYC( CYCHDLRID_tCyclicHandler_CyclicHandler, {
  TA_NULL, &tCyclicHandler_CB_tab[0], tCyclicHandler_start, 2000, 0 } );
#include "tAlarmHandler_tecsgen.h"
CRE_ALM(ALMHDLRID_tAlarmHandler_AlarmHandler,
            { TA_NULL, &tAlarmHandler_CB_tab[0], tAlarmHandler_start});
CRE_SEM(SEMID_tSemaphore_SerialPort1_ReceiveSemaphore, { TA_NULL, 0, 1 });
CRE_SEM(SEMID_tSemaphore_SerialPort1_SendSemaphore, { TA_NULL, 1, 1 });
#include "syssvc/tLogTask.h"
ATT_TER({ TA_NULL, 0, tLogTask_terminate });
```

周期タスク

```
void handler(void)
{
    iact_tsk(タスクID);
}
```

```
void task(intptr_t exinf)
{
    タスク処理
}
```

```
CRE_CYC(CALCID_CYCHDR, {
TA_NULL, 0, vCalcIdHandler, 10, 0 });
```

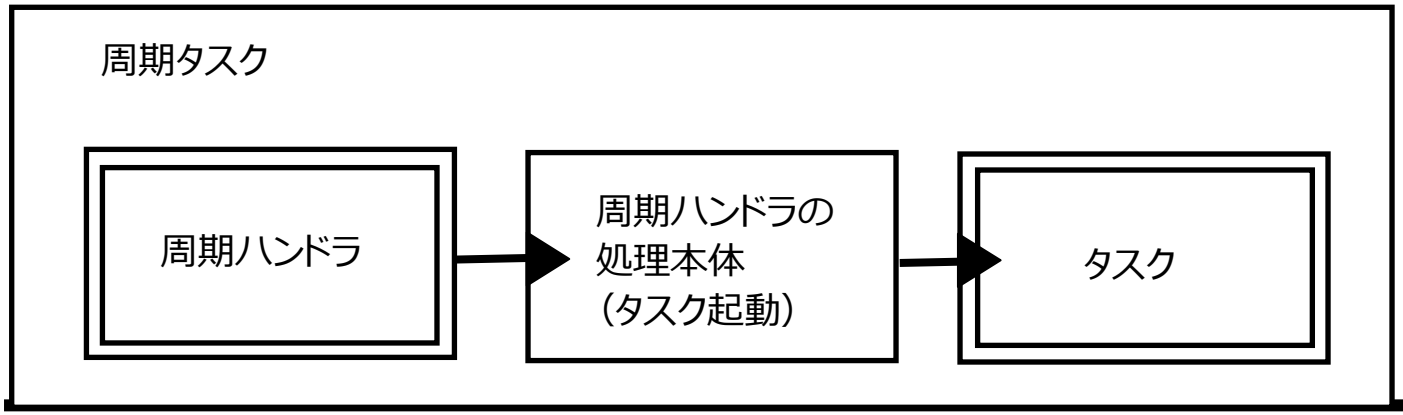
```
CRE_TSK(TASK1, { TA_NULL, 0, task,
MID_PRIORITY, STACK_SIZE, NULL });
```

周期タスク：複合セルタイプ（複合コンポーネント）

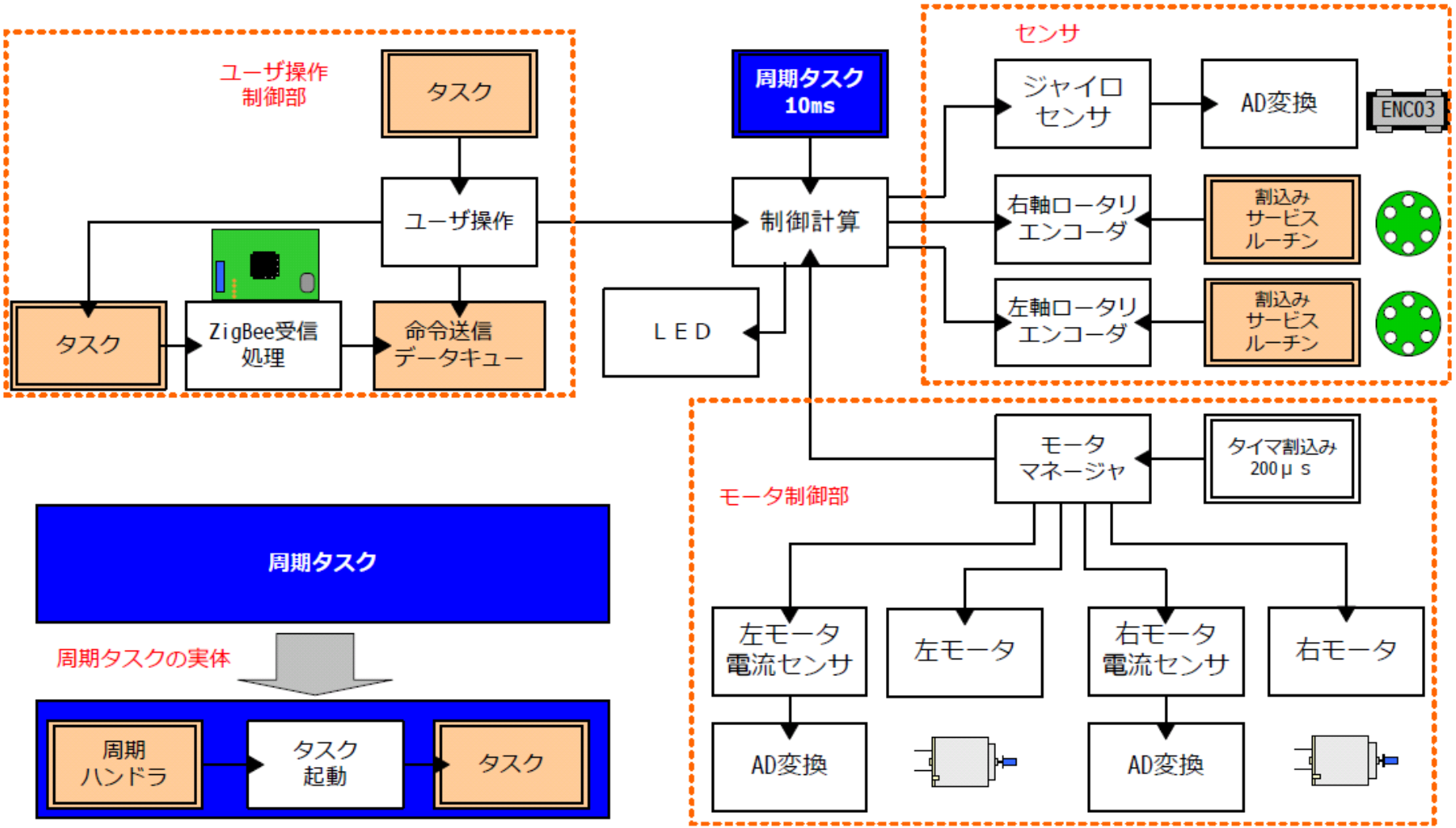
- 複数のコンポーネントを一つのコンポーネントとして扱う技術
- 利用者は周期タスクセルを生成するだけでよい

周期タスクの使用例

```
cell tCyclicTask ControlTask{
    cBody = タスク.eBody;
    cyclicTime = 10;
    priority = C_EXP("MID_PRIORITY");
    stackSize = C_EXP("STACK_SIZE");
};
```

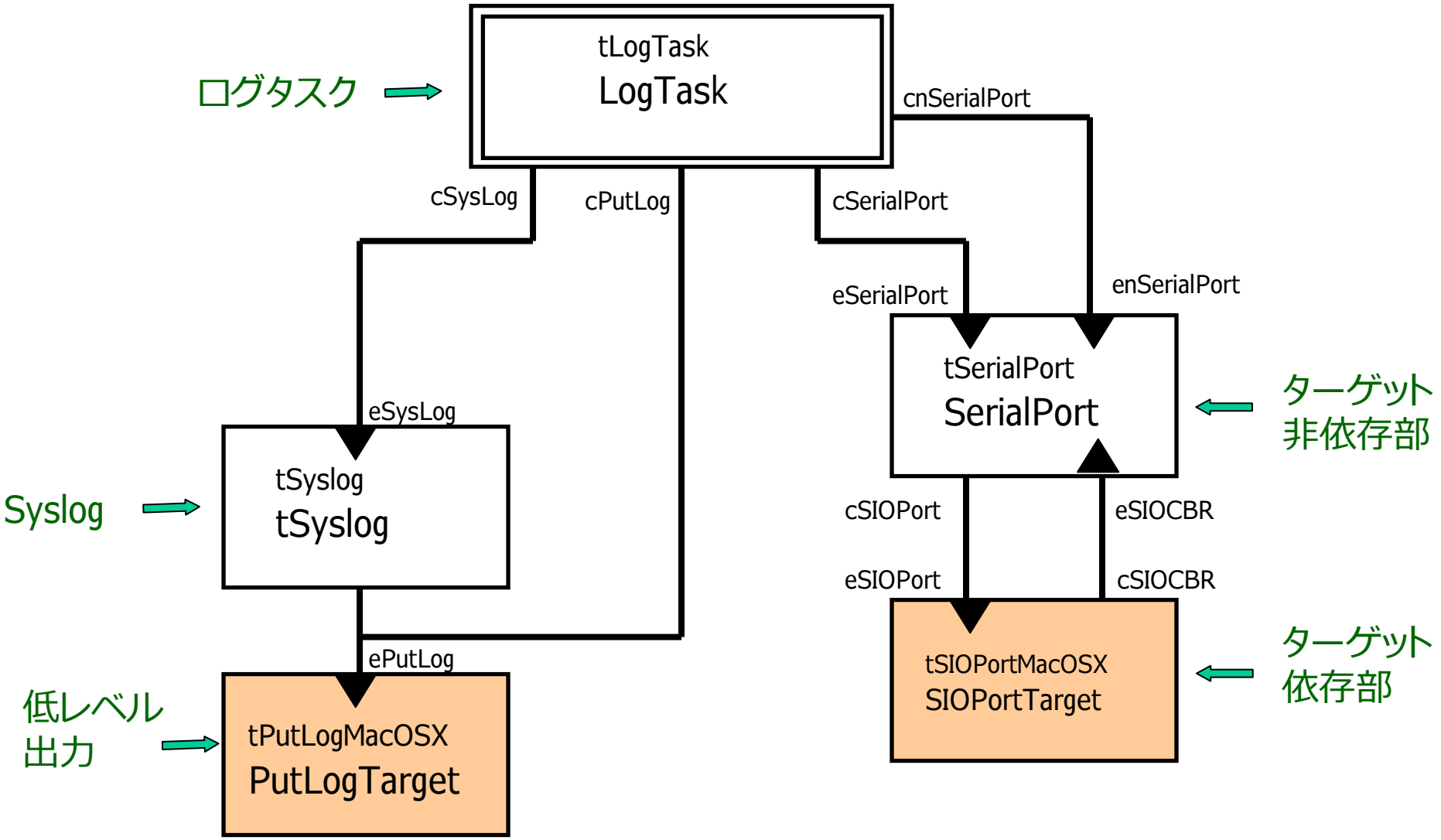


puppy2

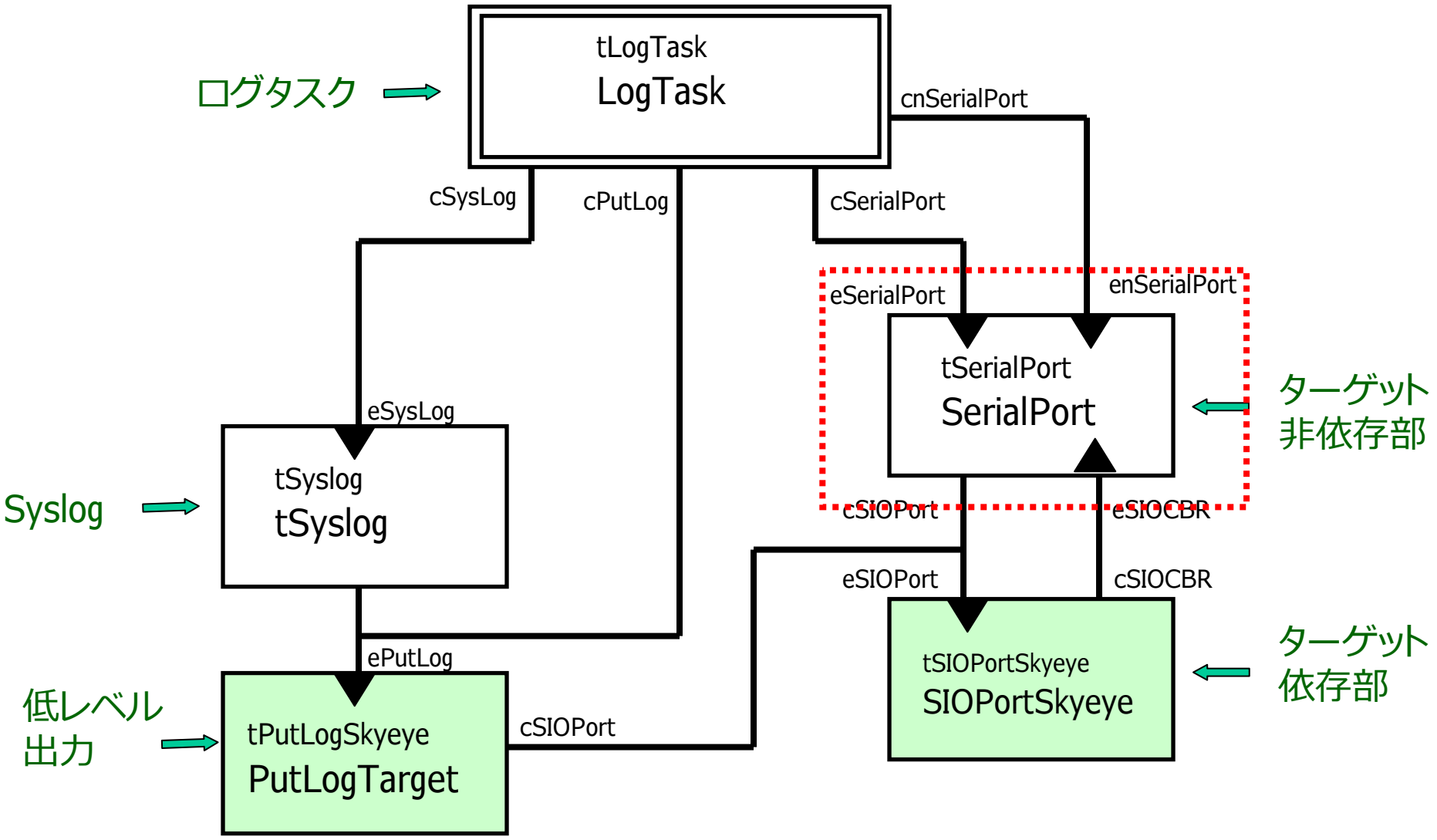


TOPPERS

ログタスク＆シリアルドライバの例（Mac）



ログタスク＆シリアルドライバの例（Skyeye）



セルタイプの定義（tSerialPort：非依存部）一部抜粋

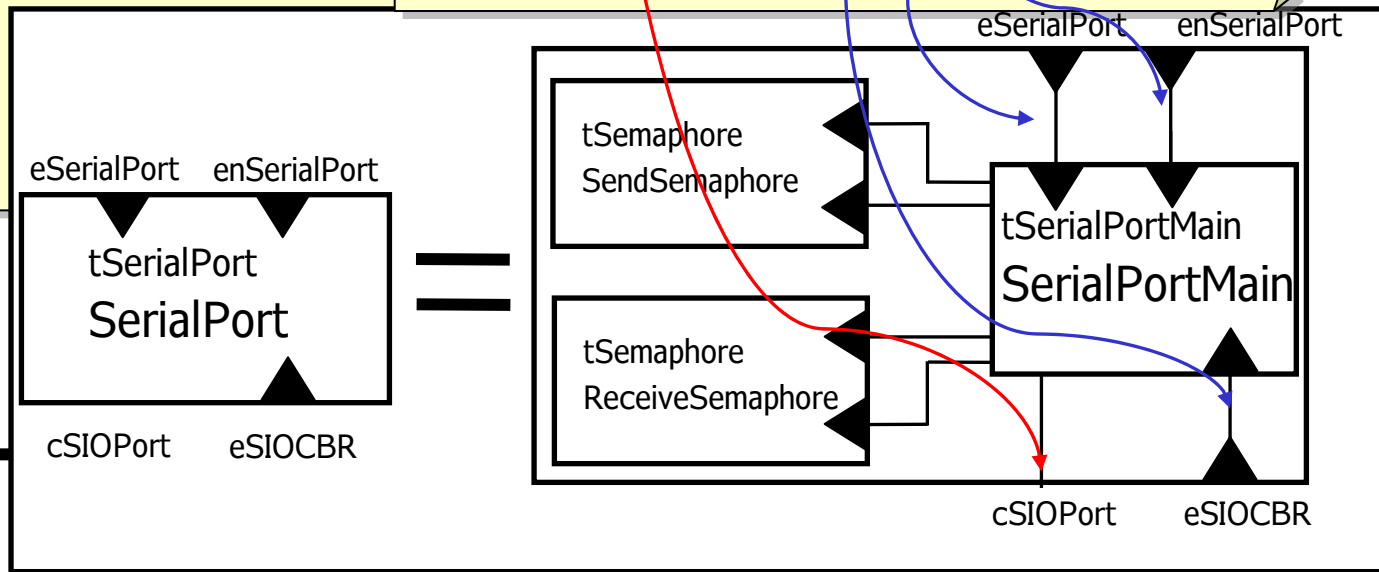
tecs_package/asp+tecs/syssvc/tSerialPort.cdl

```
composite tSerialPort{
  entry sSerialPort          eSerialPort;
  entry snSerialPort         enSerialPort;

  call sSIOPort   cSIOPort;  /* 簡易SIOドライバとの接続 */
  entry siSIOCBR             eiSIOCBR;

  attr {
        uint_t    receiveBufferSize = 256;   /* 受信バッファサイズ */
        uint_t    sendBufferSize = 256;      /* 送信バッファサイズ */
  };
  /* 受信用のセマフォ */
  cell tSemaphore ReceiveSemaphore{
        attribute = C_EXP("TA_NULL");
        count = 0;
        max =1;
  };
  /* 送信用のセマフォ */
  cell tSemaphore SendSemaphore{
        attribute = C_EXP("TA_NULL");
        count = 1;
        max =1;
};
```

```
        /* シリアルポートの制御部 */
  cell tSerialPortMain SerialPortMain{
        /* 呼び口の結合 */
        cReceiveSemaphore  = ReceiveSemaphore.eSemaphore;
        ciReceiveSemaphore = ReceiveSemaphore.eiSemaphore;
        cSendSemaphore  = SendSemaphore.eSemaphore;
        ciSendSemaphore = SendSemaphore.eiSemaphore;
        /* 呼び口の委譲 */
        cSIOPort => composite.cSIOPort;
        /* 属性 */
        receiveBufferSize = composite. receiveBufferSize;
        sendBufferSize  = composite.sendBufferSize ;
  };
  /* 受け口の委譲 */
  composite.eSerialPort  => SerialPortMain.eSerialPort;
  composite.enSerialPort => SerialPortMain.enSerialPort;
  composite.eiSIOCBR     => SerialPortMain.eiSIOCBR;
};
```

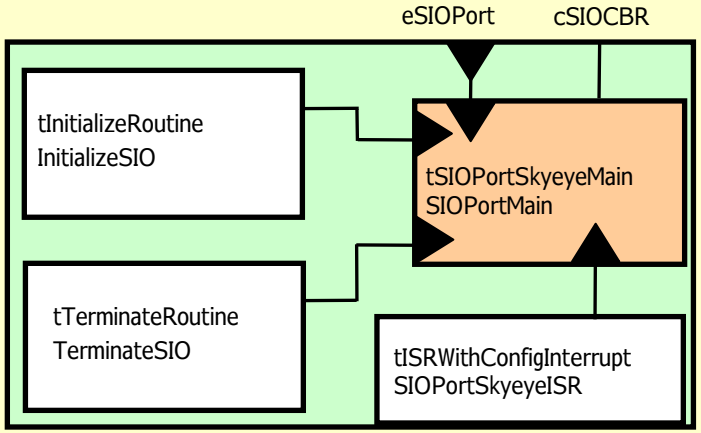


セルタイプの定義（tSIOPortSkyeyeMain：依存部）一部抜粋

```
celltype tSIOPortSkyeye {      tecs_package/asp+tecs/target/at91skyeye_gcc/tSIOPortSkyeye.cdl
      entry sSIOPort     eSIOPort;
      call siSIOCBR      ciSIOCBR;

      entry sInitializeRoutineBody eInitialize; /* 初期化処理 */
      entry sTerminateRoutineBody eTerminate;   /* 終了処理 */
      entry siHandlerBody eiISR;  /* 割込みサービスルーチン */

      attr {
            /*Skyeye用のアドレス*/
            void*  uartBase = C_EXP("(USART0_BASE)");
      };
      var {
            bool_t       openFlag;                  /* オープン済みフラグ */
            bool_t       receiveFlag;               /* 受信文字バッファ有効フラグ */
            char_t       receiveBuffer;             /* 受信文字バッファ */
            bool_t       recieveReady;              /* 受信通知コールバック許可フラグ */
            bool_t       sendReady;                 /* 送信通知コールバック許可フラグ */
      };
};
```



組上げ記述

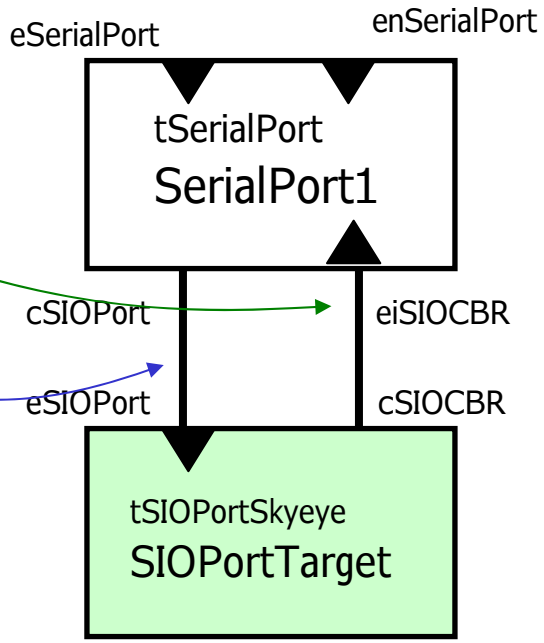
tSample.cdl

```
cell tSerialPort SerialPort1;

cell tSIOPortSkyeye SIOPortTarget {
    cSIOCBR = SerialPort1.eiSIOCBR;
};
cell tSerialPort SerialPort1 {
    cSIOPort = SIOPortTarget.eSIOPort;
    receiveBufferSize = 256;
    sendBufferSize = 256;
};
```

セルの結合

属性の初期化



TOPPERS

属性と変数

tecs_package/asp+tecs/syssvc/tSerialPort.cdl

```
celltype tSerialPort {
  /*一部抜粋*/
  attr {
        uint_t      receiveBufferSize = 128;  /* 受信バッファサイズ */
        uint_t      sendBufferSize = 128;  /* 送信バッファサイズ */
    };
  var {
        [size_is(receiveBufferSize )] char_t      *receiveBuffer ;  /* 受信バッファ */
        [size_is(sendBufferSize )] char_t         *sendBuffer;  /* 送信バッファ */
  };
};
```

属性のデフォルト値

属性の定義

変数の定義

sendBufferSizeの送信バッファを確保

tSample.cdl

```
cell tSerialPort SerialPort1 {
    /*一部抜粋*/
    receiveBufferSize = 256;
    sendBufferSize = 256;
};
```

ジェネレータが自動的に送信バッファを確保する

char_t tSerialPortMain_SerialPort1_SerialPortMain_CB_receiveBuffer_INIT[256];

属性の値を設定する
設定しない場合はデフォルト値(この場合128)を使用

TOPPERS

受け口関数の実装

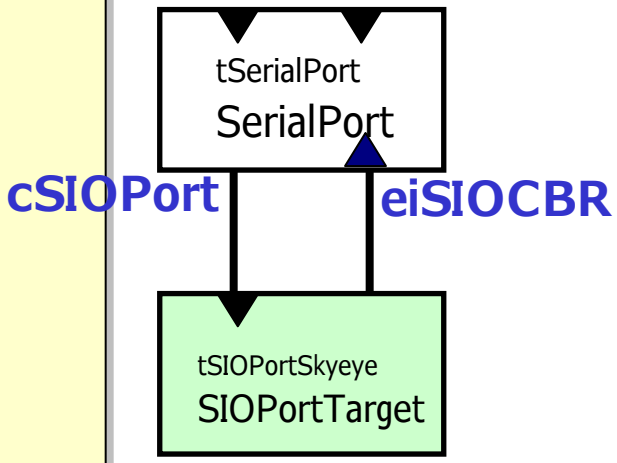
tecs_package/tutorial/asp+tecs/syssvc/tSerialPort.c

```
void
  受け口名_関数名
eiSIOCBR_readySend(CELLIDX idx)
{
      CELLCB            *p_cellcb;
      assert(VALID_IDX(idx));
      p_cellcb = GET_CELLCB(idx);
      if (VAR_rcv_fc_chr != '¥0') {
            /*
             *  START/STOP を送信する.
             */    呼び口名_関数名
            (void) cSIOPort_putChar(VAR_rcv_fc_chr);
            VAR_rcv_fc_chr = '¥0';
      }
   /*省略*/
}
```

結合先のコンポーネントを呼び出すように定義される

```
#define cSIOPort_putChar( c ) ¥
   tSIOPortSkyeyeMain_eSIOPort_putChar( ¥
```

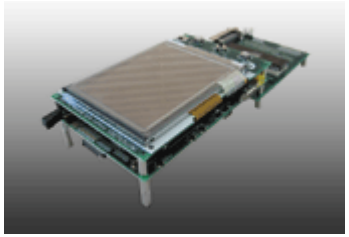
```
signature siSIOCBR {
      void   readySend(void);
      void   readyReceive(void);
};
```




```
signature sSIOPort {
      void   open(void);
      void   close(void);
      bool_t putChar([in] char_t c);
      int_t  getChar(void);
      void   enableCBR([in] uint_t cbrtn);
      void   disableCBR([in] uint_t cbrtn);
};
```

コンポーネント化の事例

- ログタスク ← 作成済み
- シリアルドライバ ← 作成済み
- カーネルオブジェクト ← 作成済み
- ファイルシステム ← プロトタイプ
- トレース ← プロトタイプ
- RPCチャネル ← プロトタイプ
- メモリアロケータ ← プロトタイプ
- PUPPY ← この後紹介
- PUPPY2 ← この後紹介
- LEGO MINDSTORMS NXT



LEGO MINDSTORMS NXT

http://www.toppers.jp/software.html

デモ中

産学連携パビリオン
UI-010

名古屋大学
高田研究室

TECSの講座　http://www.nces.is.nagoya-u.ac.jp/NEP/

- 講座：組込みシステムのコンポーネントベース開発入門
  （TECSの使い方）
- 日時：2010/3/11(木), 12(金)
  申し込み開始は2010年2月頃
- 講師：高田広章（名古屋大学）
  大山博司（オークマ）
  鵜飼敬幸（ヴィッツ）
  安積卓也（名古屋大学）
- 人数：限定10名
- 参加費：無料