TECSの使い方
Cygwin編

安積 卓也(名古屋大学)

目次

- TECS簡易パッケージ構造
- 準備
- Hello world (printf版)
  - コンポーネント記述

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のジェネレータ)

- ruby
- racc
  - http://i.loveruby.net/ja/projects/racc/からダウンロード
  - インストール方法
    :ruby setup.rb config
    :ruby setup.rb setup
    :ruby setup.rb install
- TECS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

準備(TECS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

チュートリアル:CygwinでTECSを使おう

- TECSを手軽に体験できるように、
  Cygwin上で動作する簡単なチュートリアルを
  準備しました

- Ex01:Hello Worldを実行する (printf版)
  - コンポーネント記述
  - 呼出し側の実装
  - 受け口関数の実装
- Ex02:出力文字を変更する:属性の使い方
- Ex03:セルを複数生成する

準備:tecsgenにシンボリックリンク張る

- tecspackage/bin/tecsgen.exeを利用する場合

tecs_package/tutorialで

:ln -s ../bin/tecsgen tecsgen ← tecsgenの シンボリックリンクの作成

- tecspackage/tecsgenを利用する場合

tecs_package/tutorialで

:ln -s ../tecsgen/tecsgen/tecsgen tecsgen

tecsgenの シンボリックリンクの作成

Makefileの説明1
tecsgen.exeを利用する場合の注意点

```
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
#
$(TIMESTAMP) : $(TECS_IMPORTS)
        $(TECSGEN) -I ../tecs/tecs-runtime/include -I ../tecs/tecs-runtime/workaround
                        -I ../tecs/tecs-runtime/target hello.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

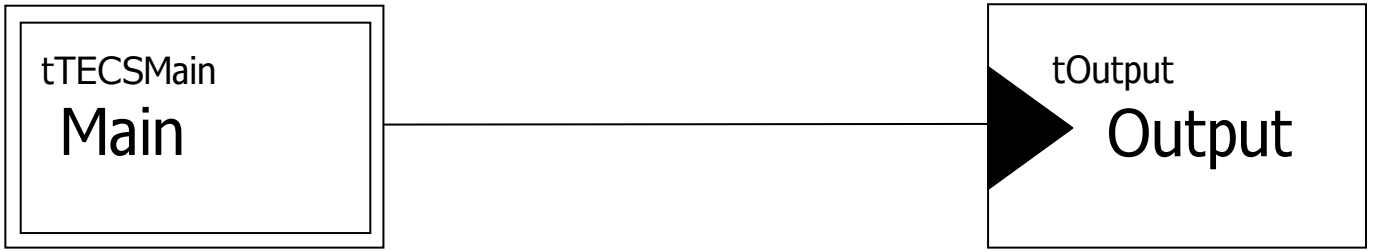
Hello World(Cygwin上で”Hello world“ を出力する)

tecs_package/tutorial/Ex01_helloで

:make tecs ⟸ TECSジェネレータの実行

:make ⟸ コンパイル

```
takuya@LENOVO-08092D61 ~/generator/tutorial/hello
$ ./hello.exe
Hello World
```



Hello Worldのコンポーネント記述

```
import_C( "tecs.h" );
```
← 型の定義

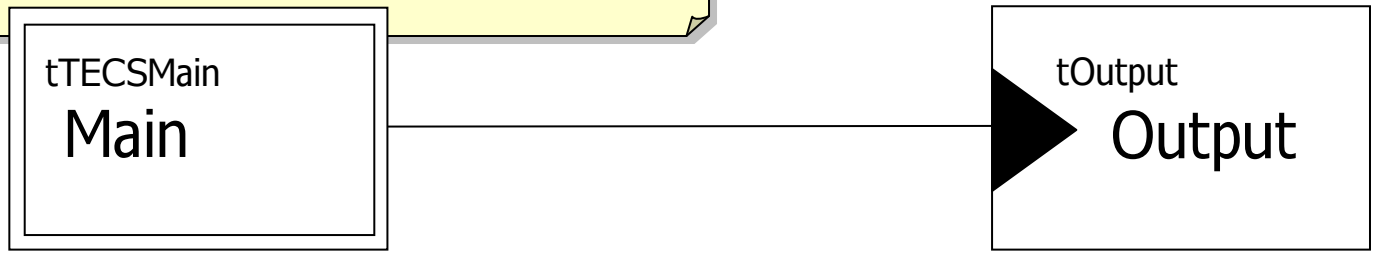
```
signature sPrint {
    void print( [in,string]const char_t *str );
};
```
←シグネチャの定義

```
[singleton,active]
celltype tTECSMain {
    call sPrint cPrint;
    attr {
        char_t *msg = "Hello World¥n";
    };
};
```
←セルタイプの定義

```
celltype tOutput {
    entry sPrint ePrint;
};
```
←セルタイプの定義

```
cell tOutput Output {
};
```
← セルの生成

```
cell tTECSMain Main {
    cPrint = Output.ePrint;
};
```
← セルの生成



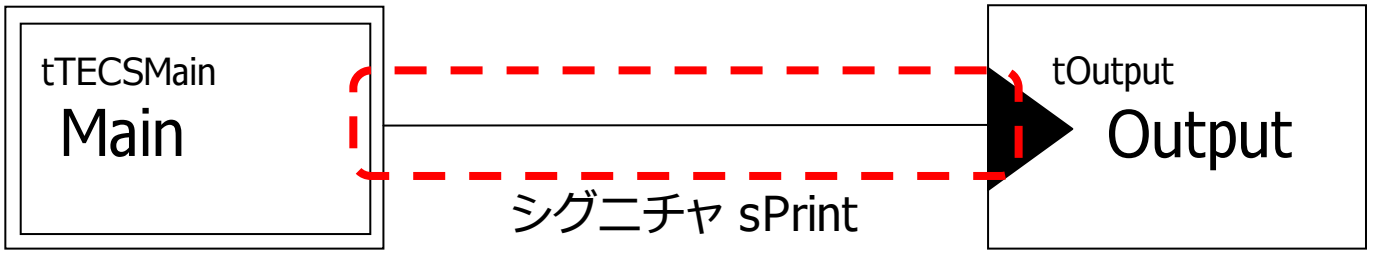
TECSのコンポーネント記述

- シグニチャ記述:**インタフェースの定義**

シグニチャsPrintはprint関数を持つ.

```
signature sPrint {
    ER print( [in,string]const char_t *str );
};
```

結合を関数で定義している



TECSのコンポーネント記述

- **セルタイプ**記述：コンポーネント（**セル**）の定義

セルタイプtTECSMainは、
呼び口cPrintを持つ.

```
[singleton]
celltype tTECSMain {
    call sPrint cPrint;
    attr {          ← 属性
      char_t *msg = "Hello World¥n";
    };
};
```

セルタイプtOutputは、
受け口ePrintを持つ.

```
celltype tOutput {
    entry sPrint ePrint;
};
```

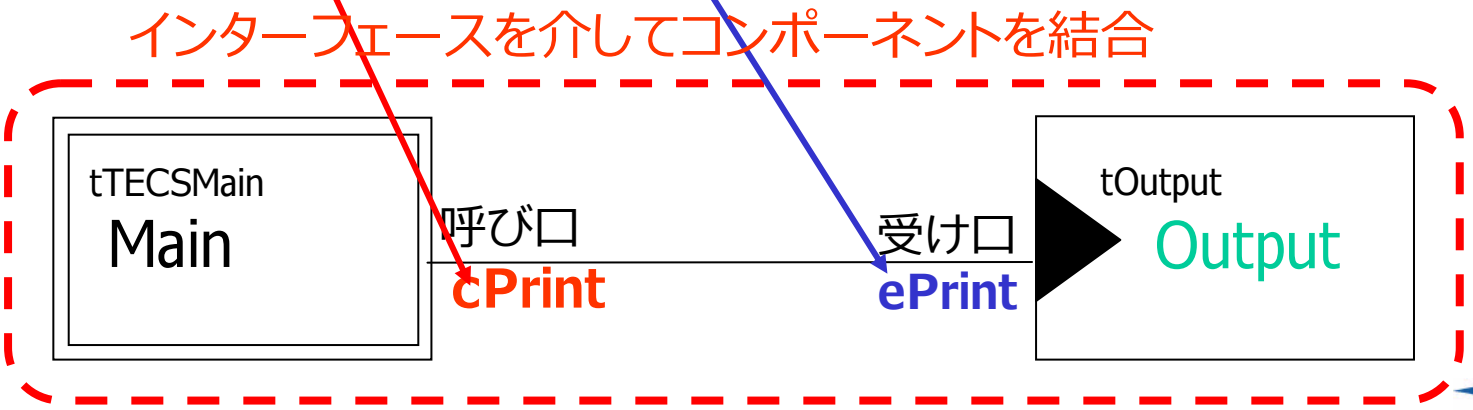
TOPPERS

TECSのコンポーネント記述

- 組上げ記述：セル同士の結合の定義
  （アプリケーションの構築）

セルTECSMainは，セルタイプtTECSMainで定義され，
自身の呼び口cPrintを，セルOutputの受け口ePrintと結合する．

```
cell tOutput Output {
};
cell tTECSMain Main {
    cPrint = Output.ePrint;
};
```



TECSのコンポーネントの実装

- 実装ファイル：呼出し側の実装

```
[tTECSTask.c]
int
main()
{   呼び口名_関数名
    return cPrint_print( ATTR_msg );
}
```

呼び口関数の呼出し

```
signature sPrint {
    void print( [in,string]const char_t *str );
};
```

```
[singleton]
celltype tTECSMain {
    call sPrint cPrint;
    attr {
        char_t *msg = "Hello World¥n";
    };
};
```

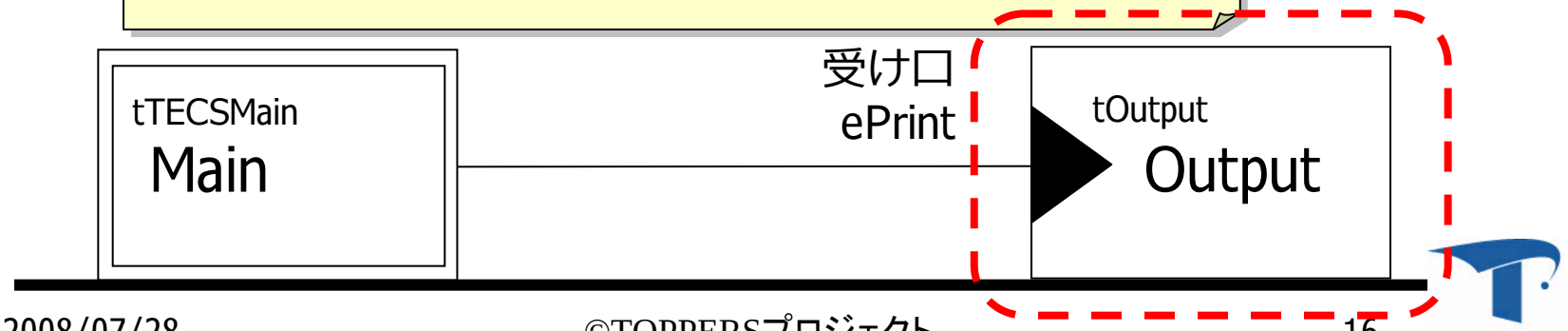
tecs_package/tutorial/Ex01-hello/src/tTECSMain.c

TOPPERS

TECSのコンポーネントの実装

- 実装ファイル：受け口関数のコードを書く

```
[tOutput.c]
ER  ePrint_print(CELLIDX idx, const char_t* str)
{
        CELLCB                  *p_cellcb;
        if (VALID_IDX(idx)) {
                p_cellcb = GET_CELLCB(idx);
        }
        else {
                return E_ID;
        }
        printf(str);        /* <<< 追記 */
        return E_OK;
}
```



hello/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

各セルタイプで自動生成されるファイル

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

- tTECSMain_factory.h
- tTECSMain_templ.c
- tTECSMain_tecsgen.c
- tTECSMain_tecsgen.h

- global_tecsgen.h ← 型の定義など

- sPrint_tecsgen.h ← 各シグニチャの定義

- tmp_C_src.c
- tmp_tecs.h

← 中間ファイル ジェネレータが使用

Ex02:出力文字を変更する:属性の使い方

- 出力文字をコンポーネント記述の属性を使用して変更する

tecs_package/tutorial/Ex02_hello-attribute/hello.cdl

```
[singleton]
celltype tTECSMain {
    call sPrint cPrint;
    attr {                          ← 属性
        char_t *msg = "Hello World¥n";
    };                                  ↖ デフォルト値
};                                      セルの生成時に指定されなかったら
                                        この値を使用する
```

tecs_package/tutorial/Ex02_hello-attribute/hello.cdl

```
cell tTECSMain Main {               ← セルの生成
    cPrint = Output.ePrint;
    msg ="Change Output¥n"; /* 出力文字列 */;
};                  ↖ この値が優先される
```


TOPPERS

実装での属性の使い方

属性アクセスマクロ
属性xxxはATTR_xxxのように
属性名の先頭にATTR_を
付けることで使用できる

例えば、コンポーネント記述で

msgと宣言された属性は、
ATTR_msg　でアクセスできる

変数アクセスマクロ

変数xxxはVAR_xxxのように
変数名の先頭にVAR_を
付けることで使用できる

Ex02_hello-attribute/src/tTECSMain.c

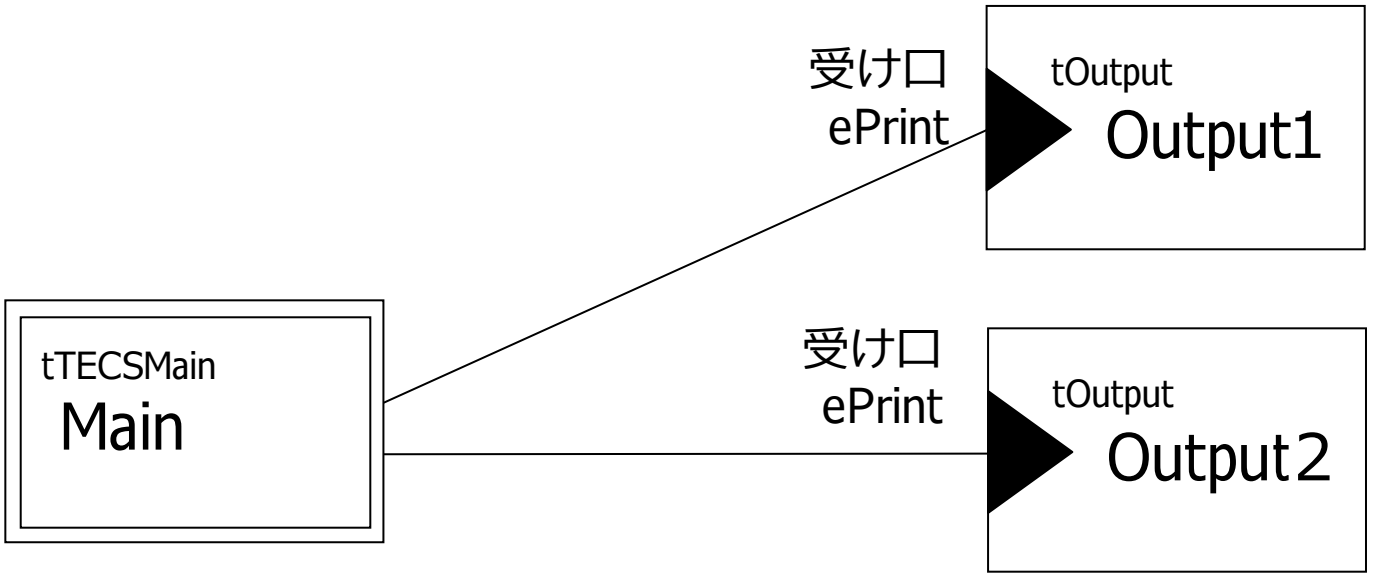
```
[tTECSMain.c]
int
main
{
      return cPrint_print( ATTR_msg );
}
```

Ex02_hello-attribute/hello.cdl

```
[singleton]
celltype tTECSMainTask {
    call sPrint cPrint;
    attr {
        char_t *msg = "Hello World¥n";
    };
};
```

Ex03:セルを複数生成する

- 一つのセルタイプから複数のセルを生成する

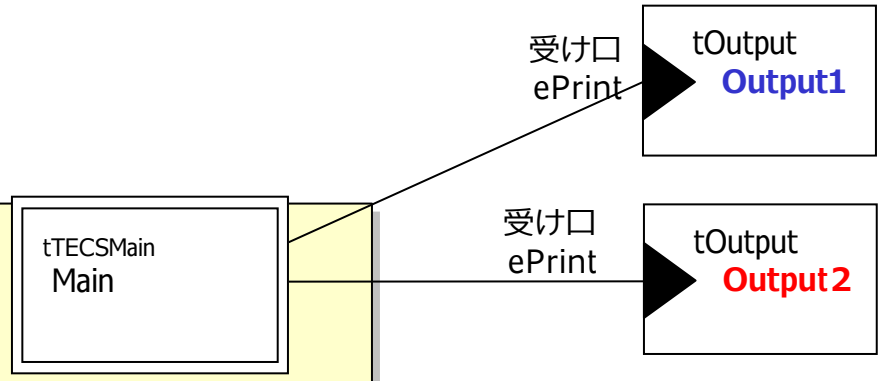


組上げ記述の書き方

tecs_package/tutorial/Ex03_hello-instaces/hello.cdl

```
/* セルのインスタンス化 */
cell tOutput Output1 {
    cell_name = "Output1";
};
/* セルのインスタンス化 */
cell tOutput Output2 {
    cell_name = "Output2";
};

/* セルのインスタンス化 */
cell tTECSMain Main{
    cPrint1 = Output1.ePrint; /* セルの結合 */
    cPrint2 = Output2.ePrint; /* セルの結合 */
};
```




```
[singleton]
celltype tTECSMain {
    call sPrint cPrint1;
    call sPrint cPrint2;
    attr {
        char_t *msg = "Hello World¥n";
    };
};
```