TOPPERSカンファレンス2020

TOPPERSプロジェクトの現状と方向性
+新しいプログラミング言語によるRTOSの実装

2020年6月12日

高田 広章

NPO法人 TOPPERSプロジェクト 会長
名古屋大学 未来社会創造機構 モビリティ社会研究所 教授
名古屋大学 大学院情報学研究科 教授
附属組込みシステム研究センター長
APTJ株式会社 代表取締役会長・CTO
Email: hiro@ertl.jp　URL: http://www.ertl.jp/~hiro/

AGENDA

TOPPERSプロジェクトの概要と活動指針（復習）

- これまでの開発成果，重点的に取り組んでいるテーマ
- 組込みソフトウェア開発効率化のためのTOPPERS

TOPPERSプロジェクトの最新の成果と活動

- この1年にリリースした開発成果
- WGの活動と主な成果，普及・広報活動の概要
- カーネル開発の状況・成果

新しいプログラミング言語によるRTOSの実装

- 取り組みの背景，使用した言語：Zig
- ZigによるRTOSの実装，コンフィギュレーション記述
- Zigの現時点での評価，Zigの課題

おわりに

TOPPERSプロジェクトの概要と活動指針 (復習)

TOPPERSプロジェクトとは?

TOPPERS = Toyohashi Open Platform for
Embedded and Real-Time Systems

プロジェクトの活動内容

- ITRON仕様の技術開発成果を出発点として，組込みシステム構築の基盤となる各種の高品質なオープンソースソフトウェアを開発するとともに，その利用技術を提供

組込みシステム分野において，*Linux*のように広く使われるオープンソースOSの構築を目指す!

プロジェクトの推進主体

- 産学官の団体と個人が参加する産学官民連携プロジェクト
- 2003年9月にNPO法人として組織化
- それ以前は，名古屋大学（2002年度までは豊橋技術科学大学）高田研究室を中心とする任意団体として活動

TOPPERSプロジェクトの狙い

決定版のITRON仕様OSの開発 完了

- ITRON仕様がかかえる過剰な重複投資と過剰な多様性の問題を解決（または軽減）

次世代のリアルタイムOS技術の開発

- 組込みシステムの要求に合致し，ITRONの良さを継承する次世代のリアルタイムOS技術を開発

Linuxと類似のOSをもう１つ作っても意味がない!

- オープンソースソフトウェア化により産学官の力を結集

組込みシステム開発技術と開発支援ツールの開発

- 高品質な組込みシステムの効率的な開発を支援

組込みシステム技術者の育成への貢献

- オープンソースソフトウェアを用いた教育コースや教材を開発し，それを用いた教育の場を提供

主な開発成果物

一般公開しているもの

第１世代カーネル

- TOPPERS/JSPカーネル，TOPPERS/FI4カーネル
- TOPPERS/ATK1（Automotiveカーネル バージョン1）
- TOPPERS/FDMPカーネル，TOPPERS/HRPカーネル

新世代（第２世代）カーネル

- TOPPERS/ASPカーネル，TOPPERS/SSPカーネル
- TOPPERS/FMPカーネル，TOPPERS/HRP2カーネル

第３世代カーネル（ITRON系）

- TOPPERS/ASP3カーネル，TOPPERS/HRP3カーネル
- TOPPERS/FMP3カーネル，TOPPERS/HRMP3カーネル

AUTOSAR関連

- TOPPERS/ATK2（Automotiveカーネル バージョン2）
- TOPPERS/A-COMSTACK，TOPPERS/A-WDGSTACK
- TOPPERS/A-RTEGEN

ミドルウェア

- TINET（TCP/IPスタック），FatFs for TOPPERS
- TOPPERS/ECNL（ECHONET Lite通信ミドルウェア）
- RLL（Remote Link Loader），DLM（Dynamic Loading Manager）

ツール，その他

- TECS（TOPPERS組込みコンポーネントシステム）
- TOPPERS BASE PLATFORM（ST，CV，RV）
- SafeG（デュアルOSモニタ），SafeG64
- EV3RT（LEGO Mindstorms EV3向けSPF）
- MDCOM（MultiDomain Communication Module）

教育コンテンツ

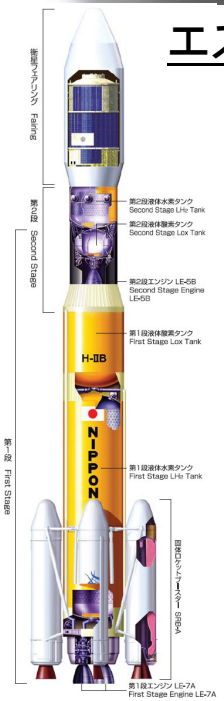
- 初級・中級実装セミナー教材
- 基礎1・基礎2・基礎3実装セミナー教材
- 基礎ハードウェア設計セミナー教材
- ETロボコン向けTOPPERS活用セミナー教材

開発成果物の主な利用事例

エスクード（スズキ）

スカイラインハイブリッド（日産）

IPSiO GX e3300（リコー）



H-IIB（JAXA）

Cell³iMager duos（SCREENホールディングス）

OSP-P300（オークマ）

SoftBank 945SH（シャープ）

UA-101（Roland）

PM-A970(エプソン)

次の10年を見据えた活動指針 (2011年度に策定)

Smart Futureのための組込みシステム技術

- 組込みシステム技術を，持続可能なスマート社会の実現（Smart Future）のための重要な要素技術の1つと位置づけ，その研究開発と普及に取り組む
- それに向けての研究開発課題
  - Safety & Security
  - Ecology（高エネルギー効率）
  - Connectivity

コンソーシアムによるオープンソースソフトウェア開発

- 同じ技術に関心を持つプロジェクトメンバによりコンソーシアムを結成し，複数組織の協力によりソフトウェアを開発
- 開発したソフトウェアは，TOPPERSプロジェクトからオープンソースソフトウェアとして公開

重点的に取り組んでいるテーマ

※ SPF：ソフトウェア
プラットフォーム

次世代のリアルタイムカーネル技術

- TOPPERS第3世代カーネル（ITRON系）
- 車載システム向けRTOS（AUTOSAR OS仕様ベース）

ソフトウェア部品化技術

- TECS（TOPPERS組込みコンポーネントシステム）

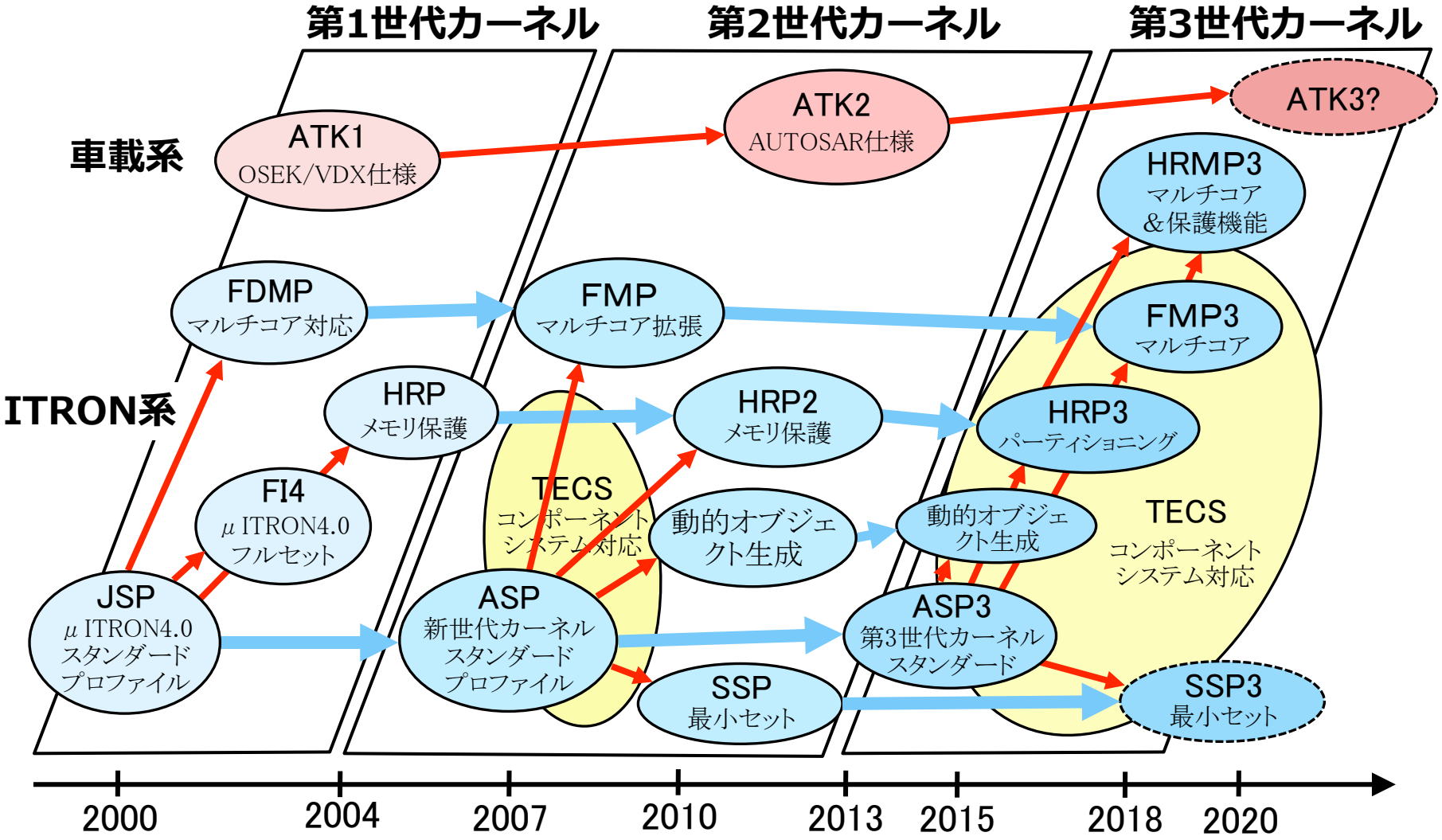
組込みシステム向けSPFと開発支援ツール

- TOPPERS BASE PLATFORM, ホームネットワーク
- シミュレーション環境（箱庭），開発支援ツール
- 車載制御システム向けSPF（AUTOSAR仕様ベース）
- 仮想化技術（SafeG），ドメイン間通信（MDCOM）

技術者育成のための教材開発

- プラットフォーム技術者の育成
- ETロボコン向けSPFと教材の提供

TOPPERSカーネル開発ロードマップ



組込みソフトウェア開発効率化のためのTOPPERS

応用分野毎のプラットフォームの構築・活用と共通化

- 組込みソフトウェアプラットフォーム構築のための，品質が高く使いやすい(… 使うことで開発の効率化ができる)部品と構築技術を提供する

設計資産の再利用を促進する仕組みの構築

- ソフトウェアコンポーネント技術(TECS)
- TOPPERSソフトウェア自身の再利用(業界レベルでの設計資産の蓄積)

人材育成

- オープンな教材(セミナー資料とソフトウェア)
- “お手本としてのTOPPERS”

新しい技術の導入 追加

- 新しい技術を導入した開発

TOPPERSプロジェクトの最新の成果と活動

この1年にリリースした主な開発成果

! 特定ターゲット向けの簡易パッケージは除く

一般公開（2019年6月～2019年12月）

6月10日	mruby on EV3RT+TECS β2.2.0
6月14日	TECS V1.7.0
6月20日	MDCOM Release1.1.0
7月1日	SafeG64 バージョン1.0.0（最初の一般公開）
7月10日	TOPPERS/EV3RT Version β7.3
10月7日	TOPPERS第3世代カーネル（ITRON系）統合仕様書 Release 3.4.0
10月7日	TOPPERS/ASP3 Release 3.5.0
10月7日	TOPPERS/HRP3 Release 3.2.0
10月7日	TOPPERS/HRMP3 Release 3.1.0
10月10日	TOPPERS/FMP3 Release 3.1.0

一般公開（2020年1月～2020年5月）

4月1日	TOPPERS第3世代カーネル（ITRON系）統合仕様書 Release 3.4.1
4月1日	TOPPERS/HRMP3 Release 3.1.1
4月15日	TOPPERS/EV3RT Version 1.0
5月13日	TOPPERS BASE PLAFORM（ST/RV） V1.4.1
	TOPPERS BASE PLAFORM（CV） V1.1.1

早期リリース（2019年6月～2020年5月）

9月4日	TOPPERS/SSP3カーネル Release 3.A.0
12月6日	TOPPERS BASE PLATFORM（RV） V0.1.1
12月20日	TOPPERS BASE PLATFORM（RV） V0.1.2

WGの活動と主な成果

教育WG

- 新基礎3セミナー，上級1セミナー
- TOPPERS BASE PLATFORM（ST，CV，RV）

TECS WG

- TECSジェネレータ＋プラグイン（継続的に拡張・改良）
- TECSUnit，TECSEditor，mruby on EV3RT+TECS

海外展開WG

- TOPPERS新世代カーネル統合仕様書の英語化

ホームネットワークWG

- Azure IoT Hubとの接続

箱庭WG

- 単体ロボット向けシミュレータ（題材：ETロボコン）
- ROS・マルチECU向け，ロボット間協調動作向け

普及・広報活動の概要

カンファレンス実行委員会

- TOPPERSカンファレンス

展示会運営委員会

- ET/IoT，ET/IoT-West，ESEC，ARMシンポジウム

TOPPERS開発者会議実行委員会

- TOPPERS開発者会議
- TOPPERS活用アイデア・アプリケーション開発コンテスト
- オープンソースカンファレンス（OSC）

広報戦略TF，エコシステム

- 外部（CQ出版等）との連携によるセミナー開催
- ウェブサイトの見直し

TOPPERS/HRMP3カーネルがJAXAの次世代宇宙機向けマイクロプロセッサに採用

昨年11月20日付けでプレス発表

- HRMP3カーネルが，JAXAが三菱重工に委託して開発を進めている次世代宇宙機向けマイクロプロセッサ用のRTOSに採用されることに
- 次世代宇宙機向けマイクロプロセッサの概要
  - シリコンオンインシュレータ（SOI）技術を用いて耐放射線特性を持たせたSoC
  - ルネサスエレクトロニクス設計のRXv2 CPUコア（倍精度FPU付き）を2つ搭載
  - 2021年度に開発を完了する計画で，開発が進行中
- HRMP3カーネルのポーティングと品質確保は，JAXAの信頼性基準に則って，三菱重工が名古屋大学の協力を得て実施中

TOPPERS/HRMP3カーネル

位置づけ

“High Reliable Multiprocessor Profile”

- HRP3カーネルをマルチプロセッサ向けに拡張したもの（HRP3カーネルの上位互換）
- FMP3カーネルに対して保護機能／パーティショニング機能を追加したもの（FMP3カーネルの上位互換）
- これまで，ITRON系では，マルチプロセッサ対応と保護機能の両方を持つカーネルは開発していなかった

開発の背景

- 近年，高い信頼性が求められる組込みシステムの分野においても，マルチプロセッサの適用が求められている
- 高い信頼性が求められる組込みシステムを開発する産業分野からの要望を受けて開発

適用対象となるターゲットハードウェア

- プロセッサ数が2～8個程度のホモジニアスなマルチプロセッサシステム

リリース履歴と計画

- 2019年3月28日　Release 3.0.0 … 最初の一般公開
- 2019年10月6日　Release 3.1.0 … バージョン3.1を長期サポートバージョンに
- 2020年3月31日　Release 3.1.1
- 未定　Release 3.2.0

この1年の開発

- 信頼性の向上
  - TTSP3（第3世代カーネル向けのTOPPERS Test Suite Package）の開発
  - カーネル付属のテストプログラム（testディレクトリ）の充実
  - カーネル付属のコンフィギュレータのテスト（test_cfgディレクトリ）の開発
    - ✓エラー検出に関するテストが進んだ

今後のカーネル開発計画

リリース済みのカーネルと統合仕様書の改善

- 継続的に機能拡張・改善を実施
- 設計書などのドキュメントの充実を継続的に実施

TOPPERS/FMP3カーネル，HRMP3カーネルのTECS対応

- TECS対応が進行中（TECS側の開発が進んだ）

開発中のカーネル

- TOPPERS/SSP3カーネル … 早期リリース中

メニーコアプロセッサ向けのカーネル（名称未定）

- FMPカーネルはメニーコアプロセッサにも有用なことが示せているが，FMP3カーネルでは対象から除外
- 開発したいが，ターゲットシステムを何にするかが課題（昨年度から進捗なし）

新しいプログラミング言語による RTOSの実装

取り組みの背景

組込みシステム開発技術の革新

- 情報技術が進む中で，新しい技術を取り込んで，組込みシステム開発技術も革新させていくべき
  - 組込みシステム開発技術の “デジタルトランスフォーメーション”
- オープンソースソフトウェアは，新しい開発技術をトライアルするには良い題材
  - トライアルした成果をオープンに議論できる

問題意識：いつまでもC言語で良いのか？

- 多くの組込みシステム開発者が感じている問題意識
- 新しいプログラミング言語は常に登場しており，いつが乗り換え時かは難しい

使用した言語：Zig

Zigとは？（Zigのウェブサイトより） ☞ https://ziglang.org/

- Zig is a general-purpose programming language and toolchain for maintaining **robust**, **optimal**, and **reusable** (+ **maintainable**) software.

言語の位置付け

- C言語を現代風に改良・拡張したもの
  - OSカーネル，組込みシステム，リアルタイムシステムは想定されている用途

なぜZigを選んだか？

- 静的APIを置き換えられる可能性を感じたため
  - コンパイル時のコード実行の機能
- RTOSの記述に向いていると思われるため
  - すべての資源を明示的に管理できる

他の新しいプログラミング言語との位置付け

- 組込みシステムにも向いた言語として，RustやGoが注目されているが…
  - “Go is *better* Java/C#”
  - “Rust is *better* C++”
  - “Zig is *better* C”

  ☞ https://kristoff.it/blog/why-go-and-not-rust/
- メモリの扱い
  - Go：ガベージコレクタ（GC）を持つ
  - Rust：言語仕様（「値の所有者」の概念の導入）により，静的なメモリ管理を可能に
  - Zig：明示的にalloc/freeを呼ぶ（複数のアロケータを持つことができ，どのアロケータを使うかも明示する）

Zigの言語設計における基本的な方針

- プログラムの振る舞いをすべて記述
  - 隠れた制御フローがない（プログラムは見た目の通りに動く）
  - 手動のメモリ管理（メモリの扱いを明示的に記述）
- 性能を重視しつつ，安全性も重視する
  - 性能と安全性は同時には両立できないため，ビルドオプションにより，性能を取るか安全性を取るかを選択（ReleaseSafeとReleaseFast/ReleaseSmall）
  - ただし，メモリ安全な言語にはなっていない … 課題
- シンプルな文法（小さい言語仕様）
  - ただし，ビルトイン関数と標準ライブラリは別（ビルトイン関数は100個くらいある．標準ライブラリはそれなりの規模．これは仕方がない）
- C言語との共存を容易に

Zig言語の重要な特徴（設計方針）

- コンパイル時のコード実行（comptime）
  - 最適化と合わせて，プリプロセッサが不要に
- 安全性の向上
  - ポインタの種類分け
  - optional type（NULLポインタの代替）
  - 安全なunion
- 記述性の向上
  - 新しいエラー処理のアプローチ
  - defer文，ラベル付きのbreak文とcontinue文（goto-less）
  - ジェネリックなデータ構造と関数
  - コンパイル時のリフレクション（メタプログラミング）
  - 非同期関数，並行性のサポート（☆未調査）
- 単体テスト記述の統合
- ビルドシステム（Makefileの代替）をZig自身で記述できる

開発者

- Andrew Kelley
  - Zig開発に対する寄附だけで生活しているようだ
  - 寄附は，5ドル/月のプランから（最大400ドル/月）

ライセンス

- 言語処理系は，MITライセンス（一条項BSDライセンスと同等）によるオープンソースソフトウェア

最新版

- Release 0.6.0（2020年4月にリリース）
- 毎日スナップショットがリリースされる

派生した言語

- Zen
  - Zig 0.3.0から派生
  - 帝都久利寿氏（日本在住）らが開発

ZigによるRTOSの実装

実装の概要

- TOPPERS/ASP3カーネル（ターゲットプロセッサ：ARM）のカーネル本体をZigで実装
  - libkernel.aの全体をZig＋インラインアセンブラで記述することに成功（.cファイル，.Sファイルは追放）
- システムコンフィギュレーションファイル（静的API）とコンフィギュレータ（Ruby）もZigで実装（パス3は除く）
  - Zigの特性や制限から，一部の仕様を変更
- アプリケーションとシステムサービス（C言語＋TECS）は修正なし
  - Zigで記述することも可能（やればできる）
- ビルドシステム（Makefile）には修正を加えた
  - ここもZigにしたいが，現時点ではできていない

Zigによるコンフィギュレーション記述

静的APIによるコンフィギュレーション記述

```
INCLUDE("tecsgen.cfg");

#include "test_sem2.h"

CRE_TSK(TASK1, { TA_ACT, 1, task1, MID_PRIORITY, STACK_SIZE, NULL });
CRE_TSK(TASK2, { TA_NULL, 2, task2, HIGH_PRIORITY, STACK_SIZE, NULL });
CRE_TSK(TASK3, { TA_NULL, 3, task3, LOW_PRIORITY, STACK_SIZE, NULL });
CRE_ALM(ALM1, { TA_NULL, { TNFY_HANDLER, 1, alarm1_handler }});
CRE_SEM(SEM1, { TA_NULL, 1, 1 });
CRE_SEM(SEM2, { TA_TPRI, 0, 1 });
```

Zigによるコンフィギュレーション記述

```
usingnamespace @import("../kernel/kernel_cfg.zig");

const tecs = @import("../" ++ TECSGENDIR ++ "/tecsgen_cfg.zig");

usingnamespace @cImport({
    @cDefine("UINT_C(val)", "val");
    @cInclude("test_sem2.h");
});

fn configuration(comptime cfg: *CfgData) void {
    tecs.configuration(cfg);
    cfg.CRE_TSK("TASK1", CTSK(TA_ACT, 1, task1, MID_PRIORITY, STACK_SIZE, null));
    cfg.CRE_TSK("TASK2", CTSK(TA_NULL, 2, task2, HIGH_PRIORITY, STACK_SIZE, null));
    cfg.CRE_TSK("TASK3", CTSK(TA_NULL, 3, task3, LOW_PRIORITY, STACK_SIZE, null));
    cfg.CRE_ALM("ALM1", CALM(TA_NULL, NFY_TMEHDR(1, alarm1_handler)));
    cfg.CRE_SEM("SEM1", CSEM(TA_NULL, 1, 1));
    cfg.CRE_SEM("SEM2", CSEM(TA_TPRI, 0, 1));
}

//  静的APIの読み込みとコンフィギュレーションデータの生成
//  以下は変更する必要がない.
<7行省略 (後に掲載)>
```

Zigによるコンフィギュレーション記述 – 拡大図

```
CRE_TSK(TASK1, { TA_ACT, 1, task1, MID_PRIORITY,
                                    STACK_SIZE, NULL });
```

システムコンフィギュレーション記述を行う関数

コンフィギュレーションデータを格納するデータ構造

```
fn configuration(comptime cfg: *CfgData) void {
    tecs.configuration(cfg);
    cfg.CRE_TSK("TASK1", CTSK(TA_ACT, 1, task1,
                        MID_PRIORITY, STACK_SIZE, null));
}
```

データ構造にタスクの生成情報を追加する関数

これらの記述をすべてコンパイル時に実行し，コンフィギュレーションデータを静的に生成

Zigによるコンフィギュレーション記述の利点

- コンフィギュレータのパス1は不要に（パス1とパス2を，1つのパスで実現できる）
  - カーネル構成・初期化ファイル（kernel_cfg.c）をZigで生成するのではなく，直接データ構造を生成するため
- C言語，静的API，Ruby（コンフィギュレータの実現言語）の記述を，すべてZigで記述できる（パス3は除く）
- コンフィギュレーション記述とカーネル本体を一体でコンパイルすることで，さらなる最適化が可能
  - コンフィギュレーションに最適化されたカーネルのバイナリコードが生成される
    - ✓例えば，セマフォが1つだけなら，セマフォの初期化処理はループを回らない
  - テスト（品質保証）の観点からは，良いとは限らない

実装したRTOSの性能（資料作成時点での測定結果）

！以下の記述はいずれも，C言語による実装と比べて

- プログラムサイズ（ASP3のサンプルプログラム）

ビルド条件	C (NDBUG)	Zig (ReleaseSafe)	Zig (ReleaseFast)	Zig (ReleaseSmall)
プログラムサイズ	34,544B	39,559B	34,898B	34,294B

  - ReleaseSafeは，15%程度大きい（やむをえない）
  - ReleaseFast/Smallは，ほぼ同じ
- 実行速度（act_tskによるタスク切換え，GR-PEACH）

ビルド条件	C (NDBUG)	Zig (ReleaseSafe)	Zig (ReleaseFast)	Zig (ReleaseSmall)
実行時間	1.4μsec	1.3μsec	1.2μsec	1.2μsec

  - ReleaseSafeでもやや高速
  - 関数を積極的にインライン展開するためと思われる

Zigの現時点での評価

- 記述が，小さくはならないが，きれいにはなっており，安全性や保守性は向上していると思われる
  - バリエーションの記述方法には，工夫とルール化が必要（後述）
  - 低レベルのビルトイン関数が便利
  - クロスコンパイル環境は問題なし
  - C言語（.hファイル，.cファイル），Ruby言語，静的APIの記述を，ほぼすべてZigに統一できた
- 少しではあるが，性能が向上したのは驚き
- リモート開発環境におけるpanic時の情報の出し方には，改善の余地
  - 開発者が実現した方法は，メモリを大量に使う（アイデアは面白いが）
- 課題が解決されれば，C言語から乗り換えたい気持ち

Zigの課題

- まだまだ未完成の言語（Release 0.6.0）
  - 未実装の機能，制限事項も多数残っている
  - 言語処理系の不具合も多い
- ドキュメントの不足（Zenのドキュメントの方が読みやすい）
  - 不具合なのか仕様なのか判断できない
  - 言語マニュアルの中にも未完成部分がある
  - 標準ライブラリを見て，使い方を探った
- エラーメッセージが不親切
  - エラーメッセージの出方（特に出る順序）が違う
  - 正しく動かすまでに試行錯誤が必要
- ターゲットプロセッサが限られる
  - llvmが対応していることが必要条件
- メモリ安全でないこと（安全でない部分の明確化）

今後の取り組み

- もう少し完成度を上げて，公開（または，会員向けにリリース）する計画
- Zigの将来性があるなら，さらに発展させたいところ
  - ご意見をお聞かせください
- 要望があれば，実装の内容について説明する場を設ける
  - 要望があれば，お寄せください

謝辞

- TOPPERS/ASP3カーネルをZigで実装するにあたり，Zigについていろいろ教えてくれた河田智明君@高田研究室に感謝します
  - Zig処理系の不具合の切り分け
  - 不具合の回避策の提示
- Zigの開発者のAndrew Kelley氏らにも感謝します

おわりに

おわりに ～ 恒例のお願い

TOPPERSへの入会のお願い

- *TOPPERSのユーザやサポータには，プロジェクトへの入会をお願いしたい*

利用事例の報告に関するお願い

- 利用事例を紹介することは，さらなる採用の促進やプロジェクトの発展につながる → ユーザにも利益に
- *TOPPERSのユーザには，利用報告をお願いしたい*

TOPPERSの活動へのご意見を募集中

- IoT時代における組込みシステムに必要な技術は？
- TOPPERS第3世代カーネルに入れるべき機能は？