Vol.2016-ARC-219 No.6
Vol.2016-SLDM-175 No.6
Vol.2016-EMB-40 No.6
2016/3/24

OSの機能のTECSプラグインを用いたコンポーネント化

河田 智明[1]　安積 卓也[2]　大山 博司[3]　高田 広章[4]

概要：本研究では TECS (組込みシステムに適したコンポーネントシステム) のプラグイン機能を利用して，TOPPERS/ASP3 カーネル（リアルタイム OS）のタイムイベント通知を TECS コンポーネント化した．近年，組込みシステムの大規模化・複雑化により，組込みソフトウェアの生産性が問題になってきている．この問題を解決するために TECS が提案されている．TECS は組込み OS 向けの静的 API を生成する機能を有している．しかし，タイムイベント通知など，複雑な静的 API の生成を要するカーネルの機能は，TECS の標準的なコンポーネントモデルでは対応できていない．本研究では複雑な静的 API を生成する TECS ジェネレータのプラグインを提案する．提案プラグインを利用することにより，複雑な静的 API の生成を利用者の利便性を害することなく実現でき，タイムイベント通知のコンポーネント化が行えることを示した．

キーワード：組込みシステム, リアルタイムシステム, コンポーネントベース開発

Componentization of Operating System Feature Using a TECS Plugin

Kawada Tomoaki[1]　Azumi Takuya[2]　Hiroshi Oyama[3]　Hiroaki Takada[4]

Abstract: In this study, we componentized the TOPPERS/ASP3 (a real-time operating system) time event notification using the plugin feature of TECS (TOPPERS Embedded Component System). With the increasing complexity of today's embedded systems, productivity of the systems is becoming a problem. To solve this problem, TECS has been proposed. While TECS supports generation of static API statements, it's limited and cannot generate certain complex statements, including ones of the TOPPERS/ASP3 time event notification. In this study, we propose the TECS generator plugin that generates complex static API statements, which we show can be used to componentize the TOPPERS/ASP3 time event notification.

Keywords: Embedded system, Real-time OS, Component-Based Software Engineering

1. はじめに

近年の組込みシステムは大規模化・複雑化がますます進んでいる．このため，組込みソフトウェアの生産性の低下が問題になってきており，大規模なソフトウェア開発に適した開発手法が求められている．こうした開発手法の一つとして，プログラムをそれぞれが再利用性の高い部品に分割する開発手法である，コンポーネントベース開発が知られている．この開発手法の利点は，ソフトウェアを既成のコンポーネントを組み合わせて構築できるため，大規模で複雑なシステムの開発コストを大幅に削減することができることである [1]．

コンポーネントベース開発を支援するためのシステムは，コンポーネントシステムと呼ばれている．コンポーネントシステムは様々なものが開発されているが，組込みシステムは CPU やメモリなどのリソース制約が厳しいため，組込みシステムに特化したものが必要となる [2]．その一つとして TECS (TOPPERS Embedded Component System)

[1] 名古屋大学工学部電気電子情報工学科
Department of Information Engineering, Nagoya University
[2] 大阪大学大学院基礎工学研究科
Graduate School of Engineering Science, Osaka University
[3] オークマ株式会社
OKUMA Corporation
[4] 名古屋大学大学院情報科学研究科
Graduate School of Information Science, Nagoya University

Vol.2016-ARC-219 No.6
Vol.2016-SLDM-175 No.6
Vol.2016-EMB-40 No.6
2016/3/24

が知られている [3].

TECS はリアルタイム OS のタスクなどのオブジェクトをコンポーネントとして扱う機能 [4] がある．リアルタイム OS はオブジェクトを静的 API によって静的に生成する機能があり，TECS はファクトリという機能により静的 API を自動生成できる．この機能を利用するコンポーネントを作成することで，リアルタイム OS の提供する機能をコンポーネントシステムの枠組みに取り込み，コンポーネントベース開発の利点を享受できるようになる．しかし，ファクトリによって生成できる静的 API の形式は限られており，複雑な静的 API については対応できていない．その一つが，TOPPERS/ASP3(ASP3 と略す) リアルタイムカーネルのタイムイベント通知 [5], 4.6(周期通知, アラーム通知) である．タイムイベント通知は静的 API の指定により，8 種類の通知方法を指定することができ，通知方法ごとに指定する引数は全く異なる．このような引数の多様性が TECS コンポーネント化の妨げとなっている．静的 API も対応できるようになると考えられる．

本研究の目的は，TECS ジェネレータのプラグイン [6] の一つであるセルタイププラグイン [7] を利用して，複雑な静的 API を生成するプラグインを提案し，利用者の利便性を害することなく実現することでタイムイベント通知のコンポーネント化を行う方法を示し，さらにその有用性を示すことである．

本研究では，まず利用者の利便性と実行効率を考慮しながらコンポーネントモデルの仕様を作成した．続いてこのコンポーネントモデルに基いて TECS ジェネレータのプラグイン `NotifierPlugin` の開発を行い，このプラグインを利用してタイムイベント通知のコンポーネント化を行った．最後に，作成したタイムイベント通知コンポーネントと TECS を利用せず直接静的 API を記述した場合で割り込み処理時間を比較することにより実行時オーバヘッドの評価を行うことで，提案プラグインの有用性を示した．

2. 対象

本章では，本研究でコンポーネント化の対象とした ASP3 タイムイベント通知と，TECS について述べる．

2.1 TOPPERS/ASP3 タイムイベント通知

ASP3 は TOPPERS プロジェクトが開発しているリアルタイムカーネルであり，TOPPERS 第 3 世代カーネルの一つである．タイムイベント通知は，TOPPERS 第 3 世代カーネル統合仕様書 [5] で定義されている時間管理機能の一つであり，通知方法を指定することで，タイムイベントをアプリケーションに通知できる機能である．一定周期で通知を行う周期通知と指定された時刻に通知を行うアラーム通知がある．通知方法として，(a) ハンドラ関数の呼び出し (b) 変数の設定 (c) 変数のインクリメント (d) タスクの起動 (e) タスクの起床 (f) イベントフラグのセット (g) データキューへの送信のうちいずれかを指定できる．通常通知は失敗することがあり (e.g. タスクが起動済)，エラー通知方法も同様に指定できるが，(a) は指定できず，(b)(g) ではユーザ指定値ではなくエラーコードが使用される．

```
CRE_CYC(CYC_ID, { TA_NULL,
  { TNFY_WUPTSK|TENFY_SETFLG, TASK_1, FLG_1, 0x0001U },
  50, 0 });
```

図 1　周期通知のカーネルコンフィギュレーション例
Fig. 1　Example of kernel configuration for creating a cyclic notification.

タイムイベント通知は，静的 API `CRE_CYC` または `CRE_ALM` をカーネルコンフィギュレーションファイルに記述して作成する．例えば，タスクの起動により通知を行い，失敗した場合はイベントフラグのセットにより通知を行う，属性 `TA_NULL` の周期通知を作成する静的 API の例を図 **1** に示す．

2.2 TECS

本節では，TECS の仕様書 [8] で規定されている開発プロセス，コンポーネントモデル及び実装モデルについて述べる．

2.2.1 TECS の開発プロセス

TECS を使用して開発を行う際には，TECS CDL という専用の言語を用い，コンポーネントの型 (セルタイプ) とそのインタフェース，及びコンポーネントのインスタンス (セル) の定義を行う．作成した TECS CDL はジェネレータに通すことにより，セル間の呼出しに必要なインタフェースコードが生成されるので，これとセルの振舞いを記述するセルタイプコードを一緒にビルドすることで，最終的なアプリケーションが得られる．

2.2.2 コンポーネントモデル

セルは TECS におけるコンポーネントのインスタンスである．セルには受け口と呼び口があり，呼び口を別のセルの受け口に結合することで，結合先のセルの機能を利用できるようになる．呼び口と受け口は配列として定義できる (呼び口配列, 受け口配列)，個別の要素を個別に結合できる．受け口・呼び口の型はシグニチャと呼び，受け口・呼び口を結合する際はシグニチャが一致している必要がある．なお，呼び口は結合を省略できない (オプショナル呼び口を除く)．

さらに，セルには属性・変数を含むことができる．属性はセルに含まれる定数値であり，プログラムや，後述するファクトリやプラグインから属性値を参照できる．変数はセルに含まれる内部変数であり，こちらは動的に値を変更できる．

セルタイプは，コンポーネントの型を表す概念であり，

Vol.2016-ARC-219 No.6
Vol.2016-SLDM-175 No.6
Vol.2016-EMB-40 No.6
2016/3/24

```
// セルタイプの定義
celltype tCellType1 {
  call sSignature cCallPort;    // 呼び口の定義
};
celltype tCellType2 {
  entry sSignature eEntryPort;  // 受け口の定義
  attr { int someAttr; };       // 属性の定義
  var { int someVar; };         // 変数の定義
};

// セルの定義
cell tCellType2 Cell2 { someAttr = 42; /* 属性値の指定 */ };
cell tCellType1 Cell1 { cCallPort = Cell2.eEntryPort; /* 結合
      */ };
```

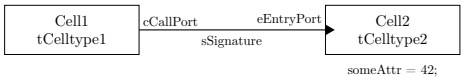


図 2　セルタイプとセルの記述例及び対応するコンポーネント図
Fig. 2　CDL describing celltypes and cells and the corresponding diagram.

同じ型のセルが共通して持つ性質を定義するものである．セルが持つ受け口・呼び口・属性・変数や，それらのシグニチャ・型は，セルタイプで定義され，各セルで属性の値や呼び口の結合先を個別に指定できる．セルの振舞いは，そのセルタイプに付随するセルタイプコードで，各受け口の実装関数 (受け口関数) により定義される．コンポーネント記述言語でのセルタイプとセルの記述例を図 **2** に示す．複合セルタイプという機能を用いて，複数のセルを組み合わせた新たなセルタイプを作ることもできる．複合セルタイプは，ジェネレータの処理過程で個別のセルに展開される．

シグニチャは，セル間の呼出しのインタフェースを記述するものである．シグニチャは関数ヘッダの集合 (0 個の場合もあり，意味的な結合を表すのに用いられる) として表される．シグニチャに記述する関数ヘッダは C 言語の関数ヘッダの拡張であり，受け渡し方向や，ポインタが指す配列のサイズを明示的に指定する点が異なる．

ファクトリ (図 **3**) は，セルが定義された際に，セルタイプで記述された文を外部ファイルに出力する機能であり，ITRON 仕様の OS でコンフィギュレーションファイルを生成する目的で使用する．

2.2.3　実装モデル

各セルの情報や状態は，RAM に確保される *CB* (Control Block, コントロールブロック) と ROM に確保される *INIB* (Initialize Block, 初期化ブロック) というメモリ領域に格納される．格納する情報がない場合は不要になるので確保されない (*CB*，*INIB* の最適化)．これらの領域の役割は次の通りである．

CB　変数の値と，INIB へのポインタが含まれる．

```
celltype tInterruptHandler {
  attr { INHNO inhNum; };
  factory { // セルのファクトリ
    write("tecsgen.cfg", "DEF_INH(%d,␣...);", inhNum);
  };
};
cell tInterruptHandler INH1 { inhNum = 1; };
```

```
DEF_INH(1, ...);
```

図 3　ファクトリ記述の例及び出力結果
Fig. 3　Factory description and the output result.

```
// セルタイプが使用されると，ExamplePlugin が呼び出される．
[generate(ExamplePlugin, "param1=1,␣param2=2")]
celltype tExampleCelltype { };
```

図 4　セルタイププラグインの使用例
Fig. 4　Example using a celltype plugin.

INIB　属性の値と，呼び口ディスクリプタが含まれる．呼び口関数の呼出しを行う際，呼び口ディスクリプタを参照することで，呼出し先の受け口関数を決定する．しかし，これにはオーバヘッドが伴うので，可能な場合は呼び口ディスクリプタを使用しない方法に最適化される (呼び口最適化)．

2.2.4　プラグイン

TECS ジェネレータは，Ruby で書かれたプラグインにより機能を拡張できる．プラグインの種類の一つが，セルタイププラグインであり，セルタイプで `generate` 指定を行うことでそのセルタイプが使用された際に独自の処理 (例えば，外部ファイルへの出力) を行うことができる．使用例を図 **4** に示す．

3.　タイムイベント通知のコンポーネント化

3.1　仕様検討

タイムイベント通知のコンポーネント化を行うにあたって，以下の点に考慮し，コンポーネントモデルを設計した．

一つは，実行速度, ROM/RAM 消費量である．組込み開発では，コストを抑えるために要件を満たす範囲内で最低限の性能で動作する設計を行うことが多いため，実行速度の低下やメモリ使用量の増加が起きると，要件を満たすためにより高性能なハードウェアが必要となり，コストの増大を招いてしまう．したがって，コンポーネントを使用することによるオーバヘッドは最小限に留める必要がある．

二つ目は，利便性 (インタフェースの簡便性，エラーをいち早く検出できる機能の有無，及び後方互換性が含まれる) である．利便性が低いコンポーネントは，生産性の低下だけでなく，バグの増大による品質低下などの悪影響を生じる．このため，コンポーネントの利便性できるだけ高めることが重要である．

Vol.2016-ARC-219 No.6
Vol.2016-SLDM-175 No.6
Vol.2016-EMB-40 No.6
2016/3/24

```
// 周期通知を操作するためのシグニチャ
signature sCyclic {
    ER start(void);                                    // 動作開始
    ER stop(void);                                     // 動作停止
    ER refer([out]T_RCYC *pk_cyclicHandlerStatus); // 状態参照
};
// アラーム通知を操作するためのシグニチャ
signature sAlarm {
    ER start([in] RELTIM alarmTime);                   // 動作開始
    ER stop(void);                                     // 動作停止
    ER refer([out]T_RALM *pk_alarmStatus);             // 状態参照
};
// (非タスクコンテキスト用)
[context("non-task")] signature siAlarm {
    ER start([in] RELTIM alarmTime);                   // 動作開始
    ER stop(void);                                     // 動作停止
};
```

図 5 タイムイベント通知コンポーネント用のシグニチャ記述
Fig. 5 Signature descriptions for the time event notification component.

3.2 タイムイベント通知コンポーネントの生成方法

ユーザは，tCyclicNotifier, tAlarmNotifier タイプのセルを生成することで，それぞれ周期通知，アラーム通知を生成できる．従来のタイムイベントハンドラの tCyclicHandler と tAlarmHandler や，それ以外の tTask 等の既存のコンポーネントと同様である．

3.3 タイムイベント通知の操作方法

従来のタイムイベントハンドラコンポーネント [9] の tCyclicHandler 用にシグニチャ sCyclic，tAlarmHandler 用に sAlarm，siAlarm が定義されており，対応する受け口 eCyclic，eAlarm，eiAlarm を通じて制御することができた．このため，本研究で作成したタイムイベント通知コンポーネントでもこの仕様をそのまま踏襲し，図 5 に示すシグニチャと，対応する名称の受け口を用いる仕様とした．

3.4 通知方法の指定

各タイムイベント通知は，タイムイベント通知とエラー通知の通知先を指定するために，ciNotificationHandler, ciErrorNotificationHandler という呼び口 (ほとんどの通知方法でこの呼び口を使うため，共通呼び口と呼ぶ) に，通知先のセルの同じシグニチャの受け口を結合するか，あるいは属性値を指定することで使用する．このように共通呼び口を用いるようにしたのは，インタフェースを簡便にするためである．なお，共通呼び口は関数ヘッダが含まれないシグニチャ siNotificationHandler を用いる．このシグニチャは意味的な繋がりを表すのみであり，これを介した関数呼出しは行われない．

通知方法は，次の方法で決定する．(a) 図 **1** に示す通り，

表 1 通知方法と結合先セルタイプ・属性の組み合わせ
Table 1 Notification methods and required celltype to join and attributes.

通知方法	セルタイプ	属性
なし [1]	なし	なし
ハンドラ呼出し [2]	tTimeEventHandler	なし
変数の設定	なし	setVariableAddress
		setVariableValue[3]
変数のインクリメント	なし	incrementedVariableAddress
タスクの起動	tTask[4]	なし
タスクの起床	tTask[5]	なし
セマフォの資源の返却	tSemaphore	なし
イベントフラグのセット	tEventflag	flagPattern
データキューへの送信	tDataqueue	dataQueueSentValue[3]

[1] この通知方法はエラー通知でのみ使用できる．
[2] この通知方法はエラー通知では指定できない．
[3] これらの属性はエラー通知では指定できない．通常の通知に失敗した際のエラーコード値が代わりに使用される．
[4] 受け口 eiActivateNotificationHandler に結合する．
[5] 受け口 eiWakeUpNotificationHandler に結合する．

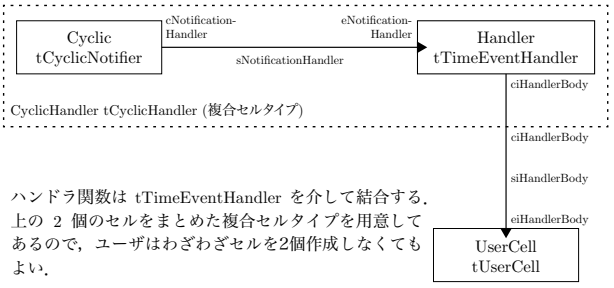


図 6 タイムイベント通知コンポーネントにハンドラ関数を結合する方法
Fig. 6 Joining a handler function to the time event notification component.

通知方法ごとに結合先のセルタイプと必須属性の組み合わせが決まっているので，これらを順に照合し，厳密に一致する通知方法を採用する．(b) 指定が不足している場合や，余剰な指定を行った場合はエラーを発生させる．

3.5 ハンドラ関数の指定方法

ハンドラ関数の指定は，図 6 に示しているとおり，tTimeEventHandler を経由してハンドラ受け口を結合する仕様とした．このままではユーザの手間が増えるため，タイムイベント通知セルと tTimeEventHandler を一つにまとめた複合セルタイプを用意した．この複合セルタイプは従来のコンポーネントと同様のインタフェースであり，従来のコードとの互換性も保てる．

3.6 エラー通知

エラー通知では ForError を末尾に付けた属性名が使用される．表 1 の脚注で示しているとおり，エラー通知では一部の属性が指定できなくなる．さらに，この他にいくつか特別な扱いが必要である．

- 通知方法 “ハンドラ関数”，“変数の設定”，及び “変数のインクリメント” ではエラーが発生し得ないためエ

Vol.2016-ARC-219 No.6
Vol.2016-SLDM-175 No.6
Vol.2016-EMB-40 No.6
2016/3/24

ラー通知は指定できない．

- 逆にエラーが発生する場合でもエラー通知を指定しないことは許され，この場合エラーは無視されてしまう．このため，属性が明示的に指定されていなければ警告を出力するようにした．

3.7 NotifierPlugin の実装

以上で述べたコンポーネントモデルに基づき，ASP3 タイムイベント通知のコンポーネント化を行った．

タイムイベント通知のコンポーネント化の中心となるのが，本研究で提案するセルタイププラグイン NotifierPlugin である．このプラグインの役割はタイムイベント通知セルの静的 API を生成することである．このプラグインは対象となる各セルに対し，次の処理を行う．

(1) 各ハンドラについて次の処理を行う．ハンドラは通常通知とエラー通知を指す内部名称で，“通常のハンドラ”, “エラーハンドラ” がある．
  (a) 共通呼び口に設定された結合 (ない場合もある) を取得する．
  (b) 呼び口の結合先の受け口・セルタイプや，指定されている属性からハンドラタイプを決定する．ハンドラタイプは同じ通知方法でも通常通知とエラー通知で動作が違うものを区別するために導入した概念である．
  (c) いずれのハンドラタイプにも合致しなければエラーを出力する．
  (d) エラー通知の指定忘れや，余剰指定を検査し，警告やエラーを出力する．
  (e) 当該ハンドラに対する静的 API 記述を生成する．
(2) 最終的な静的 API 記述をセルタイプで指定したフォーマットに基づいて生成し，出力する．

3.8 ハンドラ関数を静的 API に指定する方法

ハンドラ関数は，siHandlerBody シグニチャのハンドラ受け口を tTimeEventHandler を経由してタイムイベント通知コンポーネントに結合することで指定する．静的 API で指定できるハンドラ関数の型は void(*)(intptr_t) のみである．しかし，対応する受け口関数はそれが受け口配列か，あるいはセルタイプがシングルトンであるかに応じて必要な引数が変化するため，静的 API に受け口関数を直接指定することはできず，関数呼出しの変換を行うアダプタ関数が必要である．この実現方法の一つは，単一のアダプタ関数を用意し，目的の受け口関数を，ジェネレータが生成する呼び口関数により呼出す方法である．この方法は非常に簡単であるが，評価を行ったところ，呼び口最適化が適用されない場合は 20 サイクル程度の無視できないオーバヘッドが生じることが分かった．

そこで，受け口ごとにアダプタ関数を自動的に生成し，直接受け口関数を呼び出すことによって実現するようにした．受け口関数は最大 2 個の引数を指定する必要がある．一つは CELLIDX (非シングルトンセルタイプのみ)，もう一つは int_t (受け口配列) のみである．アダプタ関数は intptr_t 型の引数を受け取ることができるので，この値を受け口関数の引数に渡せばよい．しかし，intptr_t に CELLIDX と int_t の両方の値を含めることはできないので，一方をアダプタ関数の引数で渡し，もう一方は指定される値ごとに別のアダプタ関数が生成されるようにした．

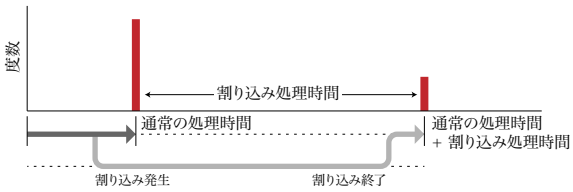


図 7 ヒストグラムを用いて割り込み処理時間を計測する方法
Fig. 7 Method to measure the interrupt processing time using histograms.

4. 性能の評価

タイムイベント通知コンポーネントを使用した場合と直接静的 API を記述した場合で割り込み処理時間を計測することにより，実行時オーバヘッドの計測を行った．

4.1 評価環境

計測は，ASP3 カーネルを STMicroelectronics 社の Cortex-M4 ベースの MPU を搭載した開発ボード NUCLEO-F401RE 用にポーティングして行った．

タイムイベント通知の割り込み処理時間の計測を行うために，50 マイクロ秒周期で動作する周期通知を作成し，通知方法をハンドラ関数に設定した[*1]．指定したハンドラ関数にはディザリング (詳細は後述する) のために wait_randomly への呼出しのみを追加した．周期通知を動作させた状態で，ビジーループを回し，ループの各反復に掛かったサイクル数を Cortex-M4 内蔵 SysTick タイマを用いて求め，ヒストグラムを得た．ループの本体で割り込みが発生するとその分処理時間が伸びるため，二つのピークのそれぞれの期待値の差を求めることで割り込み処理時間を求めることができる (図 7)．

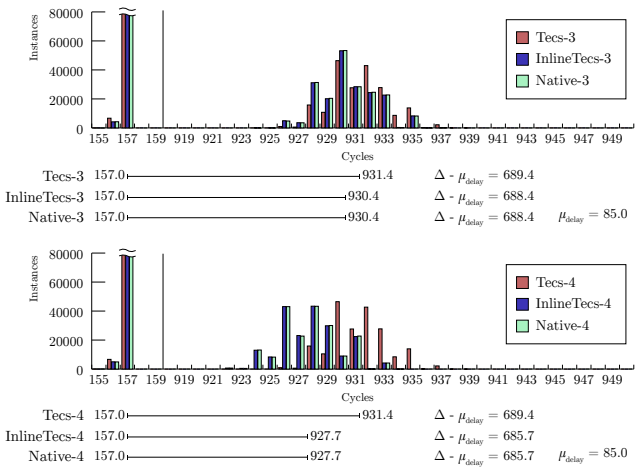
テスト条件は次の 3 通りである．

Tecs tCyclicHandler により周期通知を作成し，タイプ tMain のシングルトンセルの受け口に結合した．

InlineTecs Tecs と同様であるが，結合先の受け口はインライン関数化した別の受け口である．

Native 静的 API CRE_CYC で周期通知を作成した．

[*1] ハンドラ関数以外の通知方法は，TECS で記述しても直接書くのと全く同じ静的 API が得られるため，計測を行う意義がないと考えため，計測していない．



各テストケースについて，ループの反復時間のヒストグラムを示している．各ヒストグラムには二つのピークがあり，それぞれの期待値をヒストグラムの下の図に示している．二つのピークの差 (横棒の長さ) が割り込み処理時間 Δ である．この値から `wait_randomly` の平均実行時間の測定値 μ_{delay} を減じた値を右に示してある．

図 8　計測用ループ反復時間のヒストグラム
Fig. 8　Histograms of measured iteration time.

それぞれについて，測定対象の周期通知のほかに，ダミーとなる周期通知を作成した．ダミーの個数が 2 個の場合 (Tecs-3 等) と 3 個の場合 (Tecs-4 等) について計測を行った．

一度計測を行ったところ，測定結果の平均がまるで “量子化” されたかのように，明確に 2 種類の値に別れる現象が起きた．このような現象が発生した原因をはっきりと特定することはできなかったが，このような現象は，評価の妨げになる．量子化が行われた信号の入力信号の期待値を推定する方法として，量子化を行う前にディザリング信号を加えるという方法が知られている [10]．そこで，ハンドラ関数内で正規分布に従ってランダムに待ち時間を発生させる関数 `wait_randomly` を呼出し，ディザリングを行うようにした．これによって測定値がずれるので，あらかじめこの関数の平均待ち時間 μ_{delay} を計測し，測定値から引いて補正するようにした．

4.2　結果・考察

ループの反復時間のヒストグラムを図 8 に示す．

図 8 の結果が示す通り，タイムイベント通知コンポーネントのハンドラ関数呼出しのオーバヘッドは 4 サイクル未満と非常に小さいことが確認できた．さらに，受け口をインライン化したテストケース (InlineTecs) については，オーバヘッドが完全に無くなっていることが分かる．

5.　おわりに

本論文では，TECS ジェネレータプラグインを作成することにより複雑な静的 API のコンポーネント化が行えることを示すために，プラグインの実装アプローチ，実装アプローチの決定理由，及びそのプラグインを使用した TECS セルタイプを作成する方法に付いて述べた．さらに，作成したタイムイベント通知でハンドラ関数を呼び出す際の割り込み処理時間を計測することにより，タイムイベント通知の実行時オーバヘッドが十分に小さく，受け口をインライン関数化することによりほとんどオーバヘッドを無くすことが可能であることを示した．

現時点における提案プラグインの問題点の一つは，ほとんどの通知方法では通知先を結合によって指定できるのにも関わらず，“変数の設定”・“変数のインクリメント” のみ，変数のアドレスを属性で直接指定しなければならい点である．TECS ではコンポーネント間の結合は関数結合に限られているため，変数を指定できるようにするには工夫が必要である．このため，変数の指定をコンポーネントの枠組みに取り込むことが，今後の課題の一つである．

参考文献

[1] Bose, D.: Component Based Development, *CoRR*, Vol. abs/1011.2163 (online), available from ⟨http://arxiv.org/abs/1011.2163⟩ (2010).
[2] Crnkovic, I.: Component-based Software Engineering for Embedded Systems, *Proceedings of the 27th International Conference on Software Engineering*, ICSE '05, ACM, pp. 712–713 (2005).
[3] 安積卓也，山本将也，小南靖雄，高木信尚，鵜飼敬幸，大山博司，高田広章：組込みシステムに適したコンポーネントシステムの実現と評価，コンピュータソフトウェア，Vol. 26, No. 4, pp. 39–55 (2009).
[4] Azumi, T., Takada, H., Ukai, T. and Oyama, H.: Wheeled inverted pendulum with embedded component system: a case study, *Proceedings of the 13th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC)*, IEEE, pp. 151–155 (2010).
[5] NPO 法人 TOPPERS プロジェクト：TOPPERS 第 3 世代カーネル（ITRON 系）統合仕様書，release 3.0.0 edition (2016). `https://www.toppers.jp/docs/tech/tgki_spec-300.pdf`.
[6] Azumi, T., Oyama, H. and Takada, H.: Memory allocator for efficient task communications by using RPC channels in an embedded component system, *Proceedings of the 12th IASTED International Conference on Software Engineering and Applications*, pp. 204–209 (2008).
[7] 長原裕希，大山博司，安積卓也，西尾信彦ほか：分散インテントプラグイン: 組込み機器向け分散インテントの自動生成機構，組込みシステムシンポジウム 2013 論文集，Vol. 2013, pp. 114–122 (2013).
[8] NPO 法人 TOPPERS プロジェクト：組込みコンホーネントシステム TECS 仕様書，暫定第一版 (v1.0.2.32) edition (2011). `http://www.toppers.jp/download.cgi/tecs_package-20120608.tar.gz`.
[9] NPO 法人 TOPPERS プロジェクト：TECS 簡易パッケージ．`https://www.toppers.jp/tecs.html#e-package`.
[10] Widrow, B. and Kollár, I.: *Quantization Noise: Roundoff Error in Digital Computation, Signal Processing, Control, and Communications*, Cambridge University Press, Cambridge, UK (2008).