C-023

組込ソフトウェア開発効率化のための 実機レス開発環境の開発 Virtual software development environment for embedded software

佐田 康文†　宮崎 剛†　中島 信一†
Yasufumi Sata†, Tsuyoshi Miyazaki†, Shinichi Nakajima†

1. はじめに

近年，組込ソフトウェアはハードウェアの高性能化，IoT 化の進展等により，大規模化，複雑化が進んでいる[1]. こうした中，ユーザが必要とする組込機器製品をタイムリーに提供するため，開発期間の短縮が求められている.

組込ソフトウェアの開発においては，CPU やメモリなどのハードウェアとそれを用いた専用の開発設備が必要となる．しかし，組込ソフトウェアとハードウェアの並行開発などの理由から，専用の開発設備を入手，開発環境構築することが困難なケースがあり，その結果開発リードタイムが長期化するという課題がある.

上記課題を解決し，組込ソフトウェアの開発効率向上を実現するため，ハードウェアや専用の開発設備を必要とせず，リアルタイム OS（Real-Time Operating System，RTOS）を用いたアプリケーションの実行・デバッグが可能な開発環境の開発を進めている．既に RTOS を専用の開発設備を必要とせずに PC 上でデバッグする実機レス開発環境を開発し，報告を行った[2].

組込機器では外部のデバイスへアクセスし，測定や制御を行う．これらの機能の開発を実機レス開発環境上で実現するためには，外部のデバイスにアクセスするデバイスアクセス機能の実装が必要である.

そこで，RTOS のデバイス管理機能に組み込む形で，実機レス開発環境のデバイスアクセス機能の開発を行った．これにより，外部のデバイスにアクセスする組込機器の実機レス開発環境での開発が可能となった.

本稿では，実機レス開発環境のデバイスアクセス機能の開発の詳細について報告する.

2. 実機レス開発環境の概要

2.1 要件

組込ソフトウェアを専用の開発設備なしで開発するため実機レス開発環境が満たすべき要件は以下の通りである.

(1) PC の Windows 上で動作すること.
(2) 汎用的な統合開発環境を利用し，RTOS アプリケーションの実行・デバッグが可能であること.
(3) RTOS アプリケーションは対象ハードウェア向けに再ビルドすることにより，対象ハードウェア上で実行可能であること.
(4) 外部デバイスと連携して動作する RTOS アプリケーションの実行・デバッグが可能であること.

2.2 実機レス開発環境の機能

図 1 に実機レス開発環境の概略構造を示す.

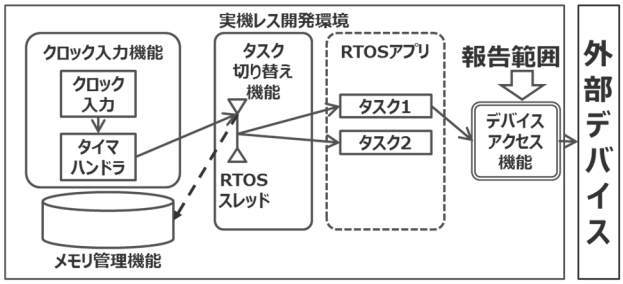

図 1　実機レス開発環境の概略構造

実機レス開発環境は，メモリ管理機能・タスク切り替え機能・クロック入力機能・デバイスアクセス機能の 4 つの機能を持つ.

メモリ管理機能は，RTOS スレッドが使用するスタック領域を静的に確保したメモリ領域に切り替え，RTOS から管理可能にする機能である.

タスク切り替え機能は，1 つの RTOS スレッド内で複数の RTOS アプリのタスクを切り替えてマルチタスクを実現する機能である.

クロック入力機能は，タイマ割込をプロセス間通信で模擬し，RTOS のタイマハンドラを呼び出す機能である.

デバイスアクセス機能は， RTOS のシステムコール経由で外部デバイスへのアクセスを可能にする機能である.

実機レス開発環境を利用する開発者が開発するアプリケーションは RTOS アプリ(図 1 の点線部)に組み込み，開発する。この部分を実機用にコンパイルすれば，実機で動作する.

メモリ管理機能，タスク切り替え機能，クロック入力機能は以前に報告済のため，今回はデバイスアクセス機能(図 1 の二重線部)の開発について説明する.

3. デバイスアクセス機能

本章では，デバイスアクセス機能の開発の詳細について述べる．統合開発環境は Windows との親和性が高い Visual Studio を，RTOS は組込機器製品で広く採用されている μT-Kernel[3]を対象とした.

3.1 デバイスアクセス機能の構造

図 2 にデバイスアクセス機能の構造を示す.

デバイスアクセス機能は，仮想 RTOS アプリプロセス，模擬デバイスプロセスに分けて実装する．図 2 の点線で囲まれた部分が新規に開発した部分である.

† Fuji Electric Co.Ltd

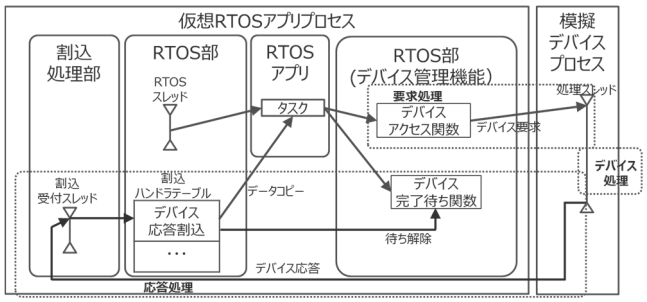

図 2　デバイスアクセス機能の構造

仮想 RTOS アプリプロセスにはデバイスアクセス機能として，RTOS のデバイス管理機能に登録する関数と，模擬デバイスプロセスからの割込を受け付ける機能を実装する．

RTOS のデバイス管理機能に登録する関数は，デバイスアクセス関数とデバイス完了待ち関数を外部デバイス毎に実装し，登録する．登録した関数は，RTOS のデバイス管理機能のシステムコールから呼び出される．

模擬デバイスプロセスからの割込を受け付ける機能として割込受付スレッドと，デバイス応答割込の割込ハンドラを実装する．

模擬デバイスプロセスには，外部デバイスに実際にアクセスする機能を実装する．これは，開発する RTOS アプリが変わってもアクセスするデバイスが同じであれば模擬デバイスプロセスを流用可能とするためである．

外部デバイスは PC の外付け機器であっても良いし，PC 上で動くデバイスのエミュレータであっても良い．PC の外付け機器にアクセスする場合は，Windows のデバイスドライバを用いてアクセスする．

仮想 RTOS アプリプロセスと模擬デバイスプロセスの間は Windows のプロセス間通信の仕組みであるウインドウメッセージで通信する．

3.2 デバイスアクセス機能の処理

デバイスアクセス機能の処理は要求処理・デバイス処理・応答処理に分けられる．以下では，処理毎に説明する．

3.2.1 要求処理

要求処理は，仮想 RTOS アプリプロセスから，模擬デバイスプロセスに対してデバイスへのアクセスの要求を行う処理である．

要求処理は，RTOS アプリからシステムコール経由でデバイスアクセス関数を呼び出すことによって開始される．

デバイスアクセス関数はデバイス要求を作成し，模擬デバイスプロセスに通知する．

デバイス要求の内容は，アクセスの種別(読み出し・書き込みなど)や，書き込みデータなどからなり，外部デバイスによって異なる．

デバイスアクセス関数はデバイス要求の模擬デバイスプロセスへの通知が完了すると復帰する．

デバイスアクセス関数からの復帰後，呼び出し元の RTOS アプリは，システムコール経由でデバイス完了待ち関数を呼び出す．デバイス完了待ち関数内で RTOS の待ち状態となり，応答処理の完了を待つ．

3.2.2 デバイス処理

デバイス処理は，模擬デバイスプロセス内で実際に外部デバイスにアクセスする処理である．

仮想 RTOS アプリプロセスからのデバイス要求を受信した模擬デバイスプロセスは，要求内容を解析し，外部デバイスにアクセスする．デバイス処理の内容は外部デバイスによって異なる．

3.2.3 応答処理

応答処理は，模擬デバイスプロセスから，仮想 RTOS アプリプロセスに対してデバイス処理の結果を通知する処理である．

模擬デバイスプロセスは，外部デバイスにアクセスした結果を，デバイス応答として仮想 RTOS アプリプロセスに通知する．

デバイス応答の内容は，処理結果や，読み出しデータなどからなり，外部デバイスによって異なる．

仮想 RTOS アプリプロセスでは，デバイス応答を割込受付スレッドで受信し，デバイス応答割込の割込ハンドラを呼び出す．

割込ハンドラは，受信したデバイス応答の内容を，デバイス完了待ち関数の呼び出し元の RTOS アプリの領域にコピーし，RTOS アプリの待ち状態を解除する．

待ちが解除された RTOS アプリは，デバイス応答の内容を元に処理を続行する．

4. 検証

開発した実機レス開発環境について，2.1 の各要件を満たしているかの検証を行った．

要件(1)，(2)，(3)については，Windows 上の Visual Studio で RTOS アプリケーションの一つであるトロンフォーラムが提供する μT-Kernel 向けテストスイートの実行・デバッグが可能なことで確認した．

要件(4)については，外部デバイスとシリアル通信を行うサンプルにて，正常に外部デバイスと連携して動作する RTOS アプリケーションの実行・デバッグが可能なことで確認した．

5. おわりに

本稿では，RTOS アプリケーション向けの実機レス開発環境の概要について述べ，デバイスアクセス機能の開発について述べた．

デバイスアクセス機能を実装したことによって，外部デバイスにアクセスする RTOS アプリの実機レス開発環境での開発が可能となった．

参考文献

[1] 情報処理推進機構，“「組込みソフトウェアに関する動向調査」調査報告書”，2018.

[2] 佐田 康文　宮崎 剛　中島 信一，”組込ソフトウェア開発効率化のための実機レス開発環境の提案”, 情報処理学会第 81 回全国大会，2019.

[3] トロンフォーラム，“μT-Kernel2.0,” https://www.tron.org/ja/tron-project/what-is-t-kernel/mt-kernel/.