6A-02

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

佐田　康文†　宮崎　剛†　中島　信一†

富士電機株式会社†

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．そこで,本稿では,上述した実機レス開発環境の詳細について報告する．

2 実機レス開発環境の設計

2.1 要件

専用の開発設備を不要とするために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．

2.2 ソフトウェア制御方式

図 1 に実機レス開発環境における RTOS アプリケーション（仮想 RTOS アプリ）のソフトウェア制御方式を示す．

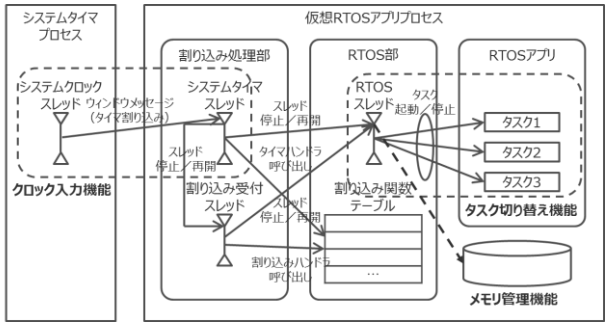

図 1　仮想 RTOS アプリ　ソフトウェア制御方式

仮想 RTOS アプリは，仮想 RTOS アプリプロセスとシステムタイマプロセスから成る．

仮想 RTOS アプリプロセスは割り込み処理部，RTOS 部，RTOS アプリから構成され，割り込み処理部の各スレッドは割り込み待ち状態で停止し，RTOS 部の RTOS スレッドが常時動作し続けることにより RTOS アプリを制御・管理する．タイマ割り込み等の割り込みが発生した場合，発生した割り込みを処理するスレッドが自身より下位の優先度のスレッドを全て停止させた後，割り込みハンドラを処理する．ハンドラ処理完了後，停止させたスレッドを順次再開させる．本プロセス内では，動作中のスレッドは常に 1 つのみとなる．

3　実機レス開発環境の開発

2 章の設計に基づき，実機レス開発環境の開発を実際に行ったので，その詳細について述べる．

本開発においては，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[2] を対象とした．

RTOS はハードウェアに依存したメモリ管理，タスク管理，システムタイマ機能を持つが，Windows 上で動作させる場合，Windows の仕組みにより実行されるため RTOS 本来の動作と異なる．

このことから，Windows・PC のハードウェア向けのメモリ管理機能，タスク切り替え機能，クロック入力機能を実装している．

3.1　メモリ管理機能

通常，RTOS は，実機 CPU で利用可能な範囲（0

Virtual software development environment for embedded software

Yasufumi Sata†, Tsuyoshi Miyazaki†, Shinichi Nakajima†
†Fuji Electric Co.Ltd

番地アドレス～メモリ空間終端）のメモリ領域を管理し，OS 資源（タスク，メモリプールなど）へのメモリの割り当て，開放を行う．

しかし，Windows プロセス内では，メモリ管理は Windows プロセスが行うため，RTOS が必要とするアドレス・サイズのメモリ領域を自由に使用できる保証はない．そのため RTOS のメモリ管理機能による OS 資源へのメモリ割り当て，開放ができない．そこで，Windows プロセス内に，実機相当の大きさのメモリ領域を静的に確保し，この領域を RTOS 処理管理のメモリ領域として使用した．これにより RTOS による OS 資源へのメモリの割り当て，開放を可能とした．

3.2 タスク切り替え機能

RTOS はシングルコアで動作しており，複数のタスクを切り替えてシーケンシャルに動作させることによってマルチタスクを実現している．

このタスクの切り替えを Windows で実現する場合は，Windows カーネルのマルチスレッドを使用してもマルチタスクが実現可能だが，RTOS のタスク管理やスケジューリングの仕組みと動作が異なってしまう．

これを，RTOS スレッド内で，RTOS のタスク管理を用いてタスクを切り替えることにより，シーケンシャルに動作させることを可能とした．

このコンテキスト退避・復元は C 言語のコンテキスト退避・復元関数を用いて実装した．

3.3 クロック入力機能

仮想 RTOS アプリの RTOS スレッドは，CPU から直接クロック供給（ハードウェアタイマ割り込み）を受けて，タイマカウンタをカウントアップすることができない．そこで，以下のようにシステムタイマ機能を構成する．

(1) 別プロセス（システムタイマプロセス）に，CPU からのクロック供給を模擬するシステムクロックスレッドを設ける．本スレッドより，定周期で仮想 RTOS アプリプロセスにタイマ割り込みを入れる．タイマ割り込みは，ウィンドウメッセージにて通知する．
(2) 仮想 RTOS アプリプロセスは，ウィンドウメッセージ（タイマ割り込み）を受け付けるため，システムタイマスレッドにウィンドウを設け，ウィンドウメッセージ通知後，通常のタイマ割り込みとしてタイマ割り込みハンドラを呼出し，タイマカウンタをカウントアップする．

これにより仮想 RTOS アプリの RTOS スレッドにおけるタイマカウンタ　カウントアップを実現した．

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．

実機レス開発環境上での RTOS アプリケーションとして,トロンフォーラムが提供するμT-Kernel 向けテストスイートを動作させ,各要件に対して,以下のように要件を満たしている結果となった．

表 1　実機レス開発環境の動作検証結果

要件	結果
(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．	○

○：要件を満たしている

5 おわりに

本稿では，実機レス開発環境の概要について述べ，RTOS アプリケーション向けの実機レス開発環境の開発について述べた．

Windows・PC のハードウェア向けのメモリ管理機能，タスク切り替え機能，クロック入力機能を実装することにより，RTOS アプリケーション向けの実機レス開発環境を実現した．今後は，汎用 OS 向けアプリケーションの実機レス開発環境の開発を進める予定である．

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. https://www.ipa.go.jp/files/000065315.pdf.
[2] トロンフォーラム，“μT-Kernel2.0,” https://www.tron.org/ja/tron-project/what-is-t-kernel/mt-kernel/.