1A-04

組込みシステムにおけるプログラム異常動作時のシステム継続手法の検討

加藤　寿和†　　山本　整†　　水口　武尚†

三菱電機株式会社　情報技術総合研究所†

1 はじめに

産業機器のような組込みシステムでは，適用先装置の特性に応じて個別の制御を実現する必要がある．多様化する個々のニーズに迅速に対応するためには，共通的な機能を提供する S/W（以下，ベース S/W）に対して，アドオン S/W により，機能を拡張する手法が考えられる[1]．

第 3 者が開発するアドオン S/W を組み込む場合，不具合などによりアドオン S/W が割り当てられた実行時間を超過して動作（以下，オーバーラン）すると，ベース S/W に制御が戻らず，システムが暴走する可能性がある．マルチタスク環境において，アドオン S/W を専用タスクで動作させる場合には，当該タスクを強制終了することで制御をベース S/W に戻すことも可能だが，既存タスクの処理の一部として関数単位でアドオン S/W を実行させる場合には，当該タスクを強制終了するとアドオン S/W の後に行われるベース S/W の処理が行われず，やはりシステムが暴走する可能性がある．

本稿では，マルチタスク環境において，既存タスクの一部として実行するアドオン S/W がオーバーランした場合に，システムのタスクスケジューリングに不整合なく，アドオン S/W の実行直前に制御および実行状態を戻し，システム動作を継続可能とする手法について，検討および試作評価した結果について報告する．

2 課題

アドオン S/W がオーバーランした場合にシステムを継続動作させるためには，オーバーランを検知した割込みハンドラや他のタスクにより，アドオン S/W の関数呼び出し直前の実行ポイントに制御および実行状態を戻した上で，アドオン S/W を無効化し，ベース S/W の後続処理を実行する必要がある．

前記に対して，C 言語の標準ライブラリでは，任意の実行ポイントに制御および実行状態を復元させる setjmp/longjmp 関数が提供されている[2]．setjmp 関数により任意の実行ポイントの実行状態を保存した上で，longjmp 関数を実行すると，即時に保存しておいた実行状態を復元し，その実行ポイントに制御を移すことができる．

しかし，マルチタスク環境において setjmp 関数を実行したタスクと異なる実行コンテキスト（割込みハンドラやタスク）から longjmp 関数を実行する場合，即時に実行状態を復元することになる．そのため，システムが持つコンテキストスイッチ処理でのタスク管理データの不整合が発生するとともに，longjmp 関数を呼び出した実行コンテキストの実行ポイントに戻ることが不可能となる．また，システムが行うタスクスケジューリングの枠組みと異なったコンテキストスイッチを行うことになるため，タスク間の優先度逆転が発生する可能性が考えられる．

3 方式検討

2 章で述べた課題を解決し，システムのタスクスケジューリングに不整合を発生させることなく，アドオン S/W の実行直前に制御および実行状態を戻し，システム動作を継続可能とする手法について検討した．

図 1 に提案手法の概要を示す．オーバーランが発生した際に，即時に longjmp 関数相当の処理を行うのではなく，実行状態を復元するための準備（スタックフレームの構築）に留めておき，システムによるタスクスケジューリングのタイミングで実行状態を復元することで，システムのコンテキストスイッチ処理との整合性を図ることが可能となる．

図 1 における(1)～(3)の詳細について，以下で説明する．

Study of system fault tolerance when add-on programs operate abnormally for Embedded Systems

Toshikazu Kato†, Hitoshi Yamamoto†, Takehisa Mizuguchi†

† Information Technology R&D Center, Mitsubishi Electric Corporation

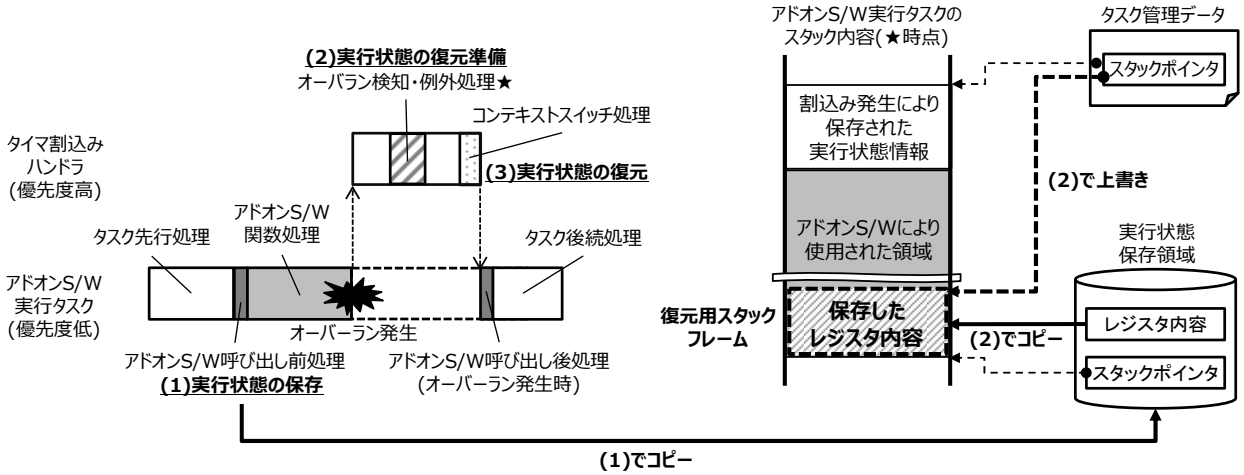

図 1　提案手法の概要

(1) 実行状態の保存

アドオン S/W の関数を呼び出す直前の実行状態をメモリ上の静的記憶領域（実行状態保存領域）に保存する．本処理は setjmp 関数の処理に相当する．その際，setjmp 関数では通常保存対象ではない CPU の制御・状態レジスタ値や FPU のレジスタ値も保存する．

(2) 実行状態の復元準備

アドオン S/W がオーバーランした際，本稿ではタイマ割込みハンドラにおいてオーバーランを検知，その例外処理を行うことを想定するが，タイマ割込みハンドラにおいて当該タスクのスタック領域上に，保存しておいた実行状態を復元させるためのスタックフレームを，システムのコンテキストスイッチ処理が実行状態を復元させる際のスタックフレームの構成に合わせて構築する．加えて，システムのコンテキストスイッチ処理により当該タスクを次回再開する際に参照するスタックポインタ値（一般的にはタスク管理データに保存される）を，本処理で構築したスタックフレームの先頭を指すように上書きする．この時点では，実行状態を復元するための準備に留めて，処理を終了する．

(3) 実行状態の復元

システムのタスクスケジューリングにより，当該タスクよりも高優先度のタスクがあればそれらのタスクが先に実行される．そして，当該タスクがスケジューリングされる際に，実行状態の復元処理が実行される．その際，システムのコンテキストスイッチ処理により，(2)で構築したスタックフレームから実行状態を復元することで，(1)の実行状態保存処理の呼び出し元に遷移する．なお，(1)を実行した際の関数の戻りと(3)により実行状態を復元した際の関数の戻りとを区別するための返り値を返すことで，アドオン S/W を無効化するか否かを判定できる．

4 実装と評価

提案手法を実装し，性能評価を実施した．実施にあたり，CPU として Arm Cortex-R4（動作周波数：600MHz，命令/データキャッシュ無効）を搭載したボードを使用した．性能評価では，3 章で述べた(1)と(2)の処理に要する時間を複数回測定し，最悪値を取得した．

評価結果を表 1 に示す．マイクロ秒オーダーの周期で動作することが多い制御機器において，十分に小さい実行オーバーヘッドで提案手法を実現可能であると考える．

表 1　評価結果

項目	結果(最悪値)
(1)実行状態の保存	73ns
(2)実行状態の復元準備	237ns

5 おわりに

本稿では，マルチタスク環境においてアドオン S/W がオーバーランした場合に，システムのタスクスケジューリングに不整合なく，アドオン S/W の実行直前に制御および実行状態を戻し，システム動作を継続可能とする手法を検討した．

試作評価の結果，システムに対して，十分小さい実行オーバーヘッドで実現可能であることを確認した．

参考文献

[1] 野原 他,"組み込みシステム向けのダイナミックリンク手法に関する研究", 情報処理学会第 71 回全国大会, 1L-4, 2009

[2] "ISO/IEC 9899:2011 Information technology - Programming languages – C"