Vol.2020-OS-148 No.6
2020/2/27

組込みシステム統合のための マルチコア制御基盤ソフトの検討

渡部 聡也[1,a)]　後藤 秀樹[2]　山中 直道[2]　鄭 俊俊[1]　毛利 公一[1]

概要：近年，様々な製品は，高機能，高性能化が要求されており，内蔵する組込み機器の数が増加している．内蔵する組込み機器が増加することにより，消費電力の増加やコストの増加，製品の内部スペースの圧迫，ネットワークの複雑化といった問題がある．この問題を解決するための方法として，複数の組込み機器で動作するソフトウェアを 1 つの組込み機器に集約する方法がある．しかし，複数のソフトウェアを 1 つに集約するとメモリ空間の競合やデバイスの競合といった問題が発生するため，複数のソフトウェアが互いに影響を及ぼさないように動作させる必要がある．複数のソフトウェアを動かす方法として，仮想化が用いられるが，仮想化ではエミュレーションによるオーバヘッドが発生する．また，仮想化で使用する VMM は規模として大きくなるため組込み機器の限られたメモリ資源を圧迫してしまう．以上の背景から本論文では，マルチコアプロセッサを用いて，複数の組込みソフトウェアを起動してハードウェアを直接割当てることでオーバヘッドを減らす制御基盤ソフトを提案し，提案した基盤ソフト上で複数組込みソフトウェアを動作させる機能について述べる．

1. はじめに

IoT や AI の普及により，家電や，車といった製品は高機能化，高性能化が要求されており、内蔵する組込み機器の数が増加している．製品が内蔵する組込み機器は，それぞれ連携することによって製品のシステムを構築している．よって，製品が内蔵する組込み機器が増えるだけでなく，組込み機器同士が連携するためのケーブル，ワイヤハーネスといった物理インタフェースの数も増加する．製品に内蔵される組込み機器や，物理インターフェースが増えることにより以下のような問題が起こる．

- 製品の内部スペースの圧迫
- 消費電力の増加
- 材料費などの製作コストの増加
- 製品全体の重量の増加
- ネットワークの複雑化

各組込み機器は，それぞれ制御を行うソフトウェアを搭載している．組込み機器が搭載するソフトウェアには，リアルタイム性が重視されるモータ制御などの単一の動作を行うアプリケーションや，FreeRTOS[1]，mbedOS[2]，Toppers/ASP[3] などのリアルタイム OS，高機能な汎用 OS がある．上記の問題を解決する方法として複数の組込みソフトウェアを 1 つの組込み機器に集約する方法がある．1 つの組込み機器に複数の組込みソフトウェアを統合を行うと図 1 に示すように組込み機器の数を削減することができる．また，電源を必要とする機器が 1 つになるため，組込み機器に電力供給を行うケーブルの数も削減することができる．さらに統合により，組込み機器間の通信インタフェースを削減することができる．

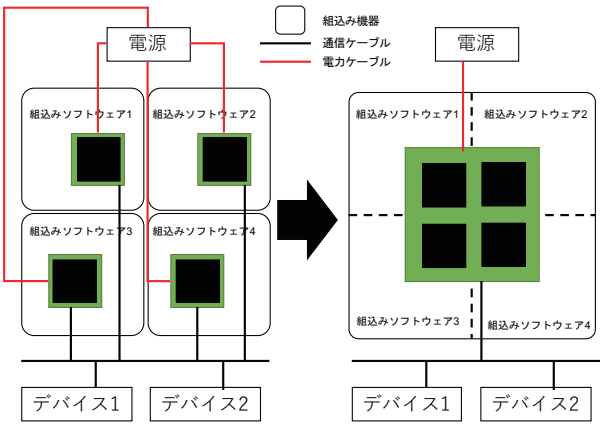

図 1　複数組込みソフトウェアの統合

現在，複数のソフトウェアを 1 つの機器で動作させる手法として仮想化手法が広く利用されている．2000 年代初

[1] 立命館大学
Ritsumeikan University
[2] アドソル日進株式会社
Ad-Sol Nissin Corporation
[a)] twatanabe@asl.cs.ritsumei.ac.jp

期に仮想化を実現するソフトウェア Xen[4] の登場を筆頭に，PC で用いられる x86 や，組込み機器で広く使用される ARM などに，CPU による仮想化のためのハードウェアサポートも実現された．

仮想化では，ハードウェア資源を管理する VMM(Virtual Machine Monitor) がハードウェアの動作をエミュレーションすることでゲスト OS 間のリソースの競合による問題を解決し，ゲスト OS 間の保護を実現する．しかし，VMM を用いた複数ソフトウェアの実行では，VMM の仲介処理によるコンテキストスイッチやハードウェアエミュレーションによりゲスト OS の処理に大きく関与してしまうため，オーバヘッドが発生する [5]．組込みソフトウェアは，リアルタイム性が重視されるため，エミュレーション処理のオーバヘッドを最小限に抑える必要がある．

組込み機器へ向けた，SPUMONE[6] などの VMM が開発されたが，ゲスト OS ごとに排他的ににリソースを割り当てて，ハードウェアの直接操作を許可しているためゲスト OS 間の保護が考慮されていない．また，SafeG は，ARM プロセッサにおける TrustZone を利用することで OS 間の保護を行う．SafeG は，規模として小さく，リアルタイム性が考量されておりオーバヘッドが最小限に抑えられている．しかし，SafeG はリアルタイム性が必要とされるソフトウェアと汎用 OS などの高機能なソフトウェアを 1 つの機器で安全に動作させるデュアルシステムを目的として開発されているため，多数のソフトウェアを集約して動作させることを目的とはしていない．

近年，CPU の処理の効率化が重視されるようになり，1 つの CPU に複数のコアを乗せるマルチコアが注目されるようになった．マルチコアは，シングルコアよりも省電力で同程度の処理を行うことができる．また，マルチコアプロセッサは，コアごとに別の処理を行うことができる．マルチコア環境において，組込みソフトウェアを各コアで動作させれば，複数のシステムを 1 つの機器上で動作させることにより，上記にあげた組込み機器増加による様々な問題は解決する．

以上の背景から，我々は，マルチコア上で複数の組込みソフトウェアを動作させるためにシステム全体へ影響を与える命令のみをエミュレーションしつつ，プロセッサコア，メモリ，デバイスの割当ての管理とパーティショニング，そして仮想的な通信路の提供を実現する，軽量なソフトウェアプラットフォームの開発を行っている．

本論文では，組込み機器で広く使用される ARM プロセッサ上に複数組込みソフトウェアを制御するプラットフォームのプロトタイプ実装を行い，既存の製品で広く使用される組込みソフトウェアである TOPPES プロジェクトの RTOS，ASP3[3] と組込みシステムに見立てたベアメタルで動くソフトウェアを同時に動かし動作を確認した．

以下，本論文では，2 章で複数のシステムを動作させる関連研究について述べ，3 章で提案手法で構築するソフトウェアプラットフォームについて述べる．4 章で複数ソフトウェアを起動する実装について述べ，5 章で今回構築した提案手法ソフトウェア上での複数の組込みソフトウェアの動作について述べる．6 章で今後の課題について述べ，7 章でまとめる．

2. 関連研究

複数のシステムソフトウェアを動作させる研究は，既存研究においても行われている．1 つのコンピュータで複数のシステムを動作させる研究として Xen[4]，SPUMONE[6]，DARMA[9][10][11],SafeG[7] がある．本章では，関連研究について述べる．

Xen は，ゲスト OS に対して Xen 上で動作するための改変を施し，CPU，メモリ，デバイスといったハードウェアの仮想化を行う．電源投入後最初起動するドメイン 0 と呼ばれるゲスト OS が，ドメイン U と呼ばれる他のゲスト OS のデバイスへのアクセスを受け取りエミュレーション処理を行う．具体的には，ドメイン U がデバイスへのアクセスを行う場合，ドメイン U からドメイン 0 へハイパーコールと呼ばれる命令を発行し，デバイスへのアクセスと処理を依頼する．ハイパーコールを受け取ったドメイン 0 は，ドメイン 0 のデバイスドライバを用いてドメイン U から受け取った処理を行う．

ハードウェアへのアクセスをドメイン 0 が管理することにより，ゲスト OS 間のリソースの保護が実現される．

Xen は，メモリの仮想化，デバイスのエミュレーション処理を行う仮想化を前提にしており，動作させるゲスト OS に大きく関与する．仮想化して動作をエミュレーション範囲が大きすぎるため，オーバヘッドが生じるというデメリットがあるまた，Xen は，リソースを潤沢に確保できるデスクトップコンピュータやサーバで使用する目的で設計されているため，規模が大きくメモリリソースを大量に消費するという問題がある．

SPUMONE は，RTOS と汎用 OS を 1 つのデバイスで動作させるための，組込み向けに提案された CPU のみを仮想化する軽量な VMM である．SPUMONE は，各 OS が，デバイスを直接操作できるようになっており，ゲスト OS への干渉を最小限に抑えている．2 つの OS を動作させた際に，一方の OS が障害発生時に障害から復旧するために再起動を行う場合がある．1 つの OS が他の OS に影響を与えないように再起動をする必要があることから，仮想 CPU を 1 つずつ各 OS に占有させて，1 つの OS で再起動する必要がある場合は，仮想 CPU を初期化することで他の OS に影響を与えずに再起動する．また，OS 間で通信を行うためのインタフェースを複数用意しており，仮想 CPU 間の割込みを利用する方法と共有メモリを用いる方法で実現している．CPU のみを仮想化することにで，仮想化に

よるオーバヘッドを最小限に抑えているが，デバイスの操作を直接許可しており，ソフトウェア間の保護を行わない問題がある．また，仮想 CPU の割り当てスケジューリングによるオーバヘッドが発生する．

DARMA は，ナノカーネルによって，OS を複数動作させることを目的としており，ナノカーネルが，シングルプロセッサを時間分割して複数 OS を切り替える．割込みが発生した場合は，ナノカーネルが割込みを横取りし，各 OS に振り分けて通知する．DARMA では，最初にホスト OS が起動し，ホスト OS のデバイスドライバをロードする機能によりナノカーネルがメモリ上にロードされる．その後，ゲスト OS はナノカーネルと同様にホスト OS デバイスドライバをロードする機能を用いてメモリ上にロードされる．ホスト OS とゲスト OS でのデバイスの競合を防ぐために，デバイスを排他的に割り当てる．各 OS が直接デバイスを操作することを許しており仮想化を行わないため，オーバヘッドが最小限に抑えられている．しかし，ナノカーネル上で動作させるゲスト OS は，最初に起動したホスト OS の機能により，上位のメモリ番地にロードされるためソフトウェアの改変が必要である．また，ナノカーネルとゲスト OS のメモリ上への配置は，ホスト OS の機能に頼るため，ホスト OS の機能に依存するといった課題がある．さらに，プロセッサを時間分割するため，プロセッサの割り当てスケジューリングによるオーバヘッドが発生する．

SafeG は，ARM プロセッサのセキュリティ拡張機能における TrustZone を利用したデュアルシステムのための組込み向けハイパーバイザである．TrustZone により領域をセキュアワールドとノーマルワールドに分け，セキュアワールドで制御用の OS，ノーマルワールドで汎用 OS を動作させるように設計されている．セキュリティ拡張機能によりノーマルワールドからセキュアワールドへのアクセスは禁止されているため，汎用 OS が攻撃されたとしてもその影響は制御用 OS 行かないようになっている．SafeG は，制御 OS と汎用 OS の 2 つを動作ささせることを目的として作られているため，多数のソフトウェアを集約して動作させることを目的とはしていない．

以上で述べた関連研究の問題を解決するために．本論文では，マルチコア環境において，以下に示すアプローチを採るソフトウェアプラットフォームを構築する．

- 各コアを組込みソフトウェアに占有させる．
- 組込みソフトウェアにハードウェアを直接操作させる．
- 複数のソフトウェアに影響を与える命令とハードウェア操作を制御する．
- 組込みソフトウェアをロードし起動する機能を備える．

コアを占有させ，ハードウェアを直接操作させることで，制御プラットフォームが介在するオーバヘッドを削減する．また，最小限の命令とデバイス操作を制御対象とすることで，プラットフォームの規模が小さくなるため，メモリ資源の圧迫を抑えることが可能である．さらに，組込みソフトウェアをロードし起動する機能を備えることで，他のソフトウェアに依存しない運用が可能となる．

3. マルチコア制御基盤ソフト

本章では，提案手法ソフトウェアである複数の組込みシステムをマルチコアで制御するためのソフトウェアプラットフォームの概要と機能，統合対象のソフトウェアについて述べる．

3.1 提案手法概要

複数の組込みソフトウェアを 1 つのハードウェアプラットフォーム上で動作させるためには，メモリ空間に適切に割り当てた後に起動する必要がある．また，提案手法ソフトウェアが起動した各ソフトウェアが，デバイスの競合やメモリ空間の重複により互いに干渉しないように制御する必要がある．複数の組込みソフトウェアに対して影響を及ぼすような命令が発行された時に，ハンドリングを行いシステム全体へ影響を与えないようにする必要がある．さらに，組込み機器は，使用できるリソースが限られていため，メモリ領域を圧迫しない程度の軽量なソフトウェアプラットフォームにする必要がある．よって，提案手法で構築する制御基盤ソフトは，複数システムを上記を満たすために以下で述べる動作を行う．

提案手法ソフトウェアは，組込みシステムを起動するために，起動するシステムを 2 次記憶から探し出し，メモリに読み込む．メモリに読み込んだ組込みシステムを任意のコアに配置して，対象のコアで動作を開始する．メモリに読み込んだシステムを起動する際に，提案手法ソフトウェアが起動後に変えてしまった CPU の設定などを起動直後の状態に戻し，起動直後の状態をエミュレーションする．複数システムが起動した後に，1 つのデバイスに対して競合が発生する恐れがあるため，競合が発生しないよう管理する．また，ハードウェアプラットフォーム上のすべてのデバイスを把握して，どのシステムがどのデバイスを利用する可能性があるかを管理する．複数のシステムで同じデバイスを使用する方法としてデバイスの処理をエミュレーションする仮想化があるが，デバイスの処理のエミュレーションを行うとオーバヘッドになる．さらに，デバイスの数によっては処理内容が多くなってしまうため，メモリリソースを大量に消費してしまう．よって，デバイスの仮想化は必要最低限のデバイスに対して行い，デバイスの競合が発生した場合は，排他制御を行い組込みソフトウェア間の制御を行う．また，1 つの組込みシステムで障害などが発生した際に，ハードウェアリセットを伴う再起動を行う命令が発行される場合がある．ハードウェアリセットが行われると電源供給が一時的に停止してしまい他のシステムにも影響が出てしまう．このような，提案手法ソフトウェ

ア上で動作する複数のソフトウェアへ影響を及ぼす命令を横取りして，例外処理を行う．

これらのことを踏まえ，必要となる機能を表 1 に示す．

表 1　必要となる機能

分類	機能
複数システム起動	ローダ機能
	起動エミュレーション機能
競合管理	メモリ管理機能
	デバイス管理機能
システム全体の制御	命令トラップ機能
	例外ハンドリング機能

3.2　統合対象

提案手法では，1 つの製品内の組込み機器の数を削減するために，それぞれの組込み機器上で動作するシステムを 1 つの機器に統合する．統合するソフトウェアは，以下のものを想定する．

- 組込み向け汎用 OS
- リアルタイム OS
- ベアメタルプログラム

提案手法ソフトウェアにおいて，表 1 に示す機能だけでは対処できない項目が存在する．そのため，対処できない項目に対しては，動作させるシステムのソフトウェアを改変することにより対応する．よって，上記のソフトウェアの内，改変が許されるものを対象とする．ソフトウェアの改変な必要な箇所を以下で述べる．

CPUID により変わる処理

組込み機器に限らずマルチコアを持つプロセッサは，コア毎に CPUID を持ち，電源投入後 BSP(Boot strap processor) と呼ばれる CPUID0 のコアが最初に起動するため，プログラムは，BSP のコアで動作を開始することを想定して実装される．提案手法上では，CPUID0 以外でも起動するため動作させる．よって，ソフトウェアで特定の CPUID をもつコアでしか動作しない処理を改変する必要がある．

CPU 機能の設定を変更する処理

ソフトウェアの中には，起動時にプロセッサの初期化を行うものがある．提案手法ソフトウェアが設定したレジスタの値を変更する可能性があるため，CPU の設定を変更しないように起動シーケンスの処理を変更する必要がある．

メモリレイアウトの競合

組込みシステムは，動作するメモリ番地が決まっているため，複数のシステムを動作させる際互いに同じメモリ番地を使用する可能性がある．よって，メモリ番地の競合が起こらないようにメモリレイアウトを改変する必要がある．

3.3　提案手法に必要な機能

提案手法では複数の組込みシステムを制御するためのソフトウェアを配置する．図 2 にシステムの構成図を示す．マルチコア環境において，各コアをその上で動作するシステムがそれぞれ占有し，その下で提案手法ソフトウェアが制御を行う．提案手法では，各組込みシステムにコアを占

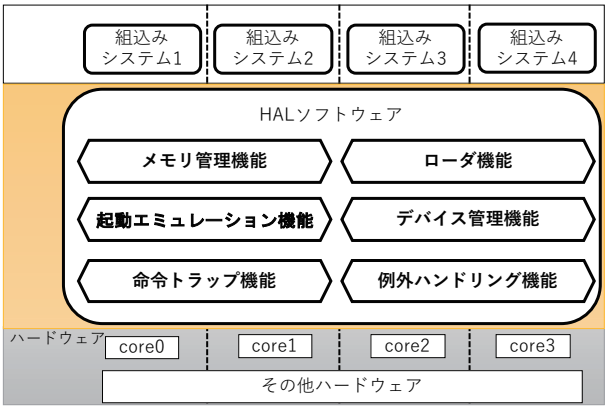

図 2　提案手法により動作する複数の組込みシステム

有させて動作させる．これを実現するために必要となる機能は以下の通りである．

ロード機能

ロード機能は，２次記憶からメモリ上に組込みシステムを読み込むための機能である．２次記憶のデバイスを読み込み，ファイルシステムを解析する．ファイルシステムの解析結果からファイル名を取得し，ロード対象の組込みシステムの実行ファイルを探す．そして，ロード対象の組込みシステムを，他のシステムのメモリ領域に被らないように物理メモリ空間上にロードする．

起動エミュレーション機能

起動エミュレーション機能は，メモリに配置した組込みシステムを起動するための機能である．コア番号を指定してコア間割込みを行い，対象コアが割込みハンドラから組込みシステムを起動させる．組込みシステムの実行を開始する前に，提案手法ソフトウェアが使用したプロセッサのコプロセッサのレジスタ値とプロセッサ情報を保持するレジスタ，汎用レジスタの値などの値を保存し，電源投入直後の値に戻す．

メモリ管理機能

メモリ管理機能は，各組込みシステムが使用する領域を管理する．すべてのシステムが使用するメモリ領域を把握することにより，ロード機能でシステムをロードする際に使用するメモリ領域の重複が発生しない場所に配置することができる．

デバイス管理機能

デバイス管理機能は，複数システム間でデバイスの競合が起きないようにデバイスの管理を行う．ハードウェア上に存在する MMIO で制御できるデバイスを全て把握し，どの MMIO 領域をどのシステムが使用するかの管理を行う．

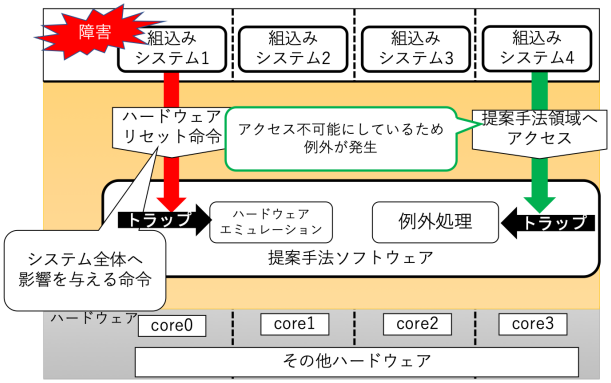

図 3　命令トラップ機能

ハードウェアを操作するための MMIO へのアクセスを制御して，アクセス可能なデバイスを制限する．これにより，1 つのデバイスに対しての競合の発生を防ぐ．1 つのデバイスを複数のシステムで共有する必要がある場合，CPU のロック機能や命令エミュレーションを用いて制御を行う．

命令トラップ機能

命令トラップ機能は，他のシステムへ影響を与えるような命令や提案手法ソフトウェアが管理するメモリ領域や，デバイスへアクセスする命令を横取りする．トラップ処理を図 3 に示す．トラップすべき他システムへ影響を与える命令は各システムで必ず共有しなければならないメモリや，電源といったハードウェアへのアクセス命令である．その他のデバイスは，組込みシステム設計時に各システムの使用するデバイスを排他的に設計することで解決する．横取りした命令は，以下で述べる例外ハンドリング機能で処理し，命令発行元のシステムに返す．

例外ハンドリング機能

例外ハンドリング機能は，命令トラップ機能でトラップした命令や，提案手法ソフトウェア上で発生した例外に対処するための機能である．トラップ機能でハードウェアリセット命令を横取りした場合は，ハードウェアリセット命令をエミュレーションするために CPU のレジスタ，コプロセッサの初期化を行う．

4. 提案手法実装

提案手法ソフトウェアは，組込みで広く使用される ARM プロセッサ Cortex-A を対象にして構築する．本論文では，表 2 に示す環境で，動作するように実装を行なった．本章では，現在実装が完了している提案手法ソフトウェアの複数システムを起動する実装について述べる．

表 2　動作環境

項目	内容
ボード	Raspberry Pi3 Model B
コア	Cortex-A53
アーキテクチャ	aarch32

4.1 動作させるシステムへの影響を抑える実装

2 章で述べた通り，既存組み込みシステムは，メモリ番地 0 から CPUID0 のコアで実行を開始する．よって，動作させるソフトウェアの改変箇所を最低限に抑えるために，提案手法ソフトウェアは起動後に自身の再配置を行うように実装を行なった．提案手法ソフトウェアは，起動後にメインプログラムをメモリの低位番地から高位番地に再配置して，CPUID3 で実行を開始する．CPUID0 から 2 のコアは WFI(Wait For Interrupt) 命令により，割込みが来るまで待機状態にする．

4.2 複数システム起動のための実装

提案手法ソフトウェアは，プロセッサ間割込みを利用することで，組込みシステムを起動する．シェルから起動するシステム名とロードする番地，起動する CPUID を指定して，ロードコマンドを実行することで実行ファイルがメモリ上にロードされ，起動コマンドを実行することで実行が開始される．提案手法ソフトウェアがシステムを任意のコアで起動する実装について，コア 1 での起動を例として以下で述べる．

(1) シェル上からロードするプログラムとロード先のメモリ番地を指定する．そして，2 次記憶のロード対象の実行ファイルをメモリの指定番地へロードする．
(2) core0 から core1 にプロセッサ間割込みで割込みを発生させて，同時にロードしたメモリ番地を通知する．
(3) core1 の割込みハンドラ内で CPU のレジスタ，コプロセッサレジスタを起動直後の状態に初期化する．
(4) core1 のプログラムカウンタを実行ファイルの配置番地にセットしてロードした組込みソフトウェアの実行を開始する．

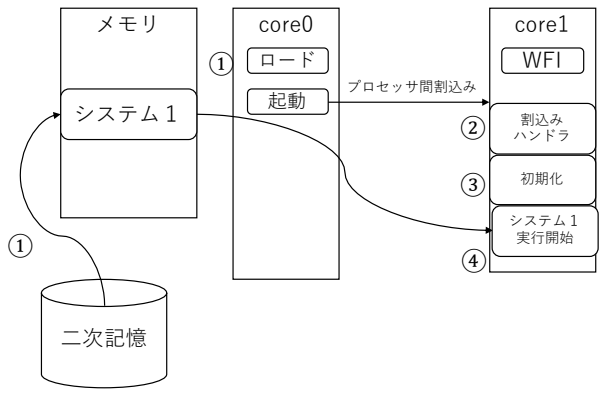

図 4　システム動作シーケンス

5. 複数組込みシステム動作確認

本章では，提案手法ソフトウェア上で行った動作検証について述べる．

5.1 検証内容

図 5 に示す構成で動作検証を行った．検証用に 3 つのプログラムを用意した．用意したプログラムはそれぞれ，MMIO で競合が起こらないように，同じデバイスを使用しないプログラムである．コア 3 で動作する提案手法ソフトウェアの shell からコア 0，コア 1，コア 2 にそれぞれ，TOPPERS ASP3，ベアメタルで動作する気温表示プログラム，ベアメタルで動作する I2C によるカウンター表示プログラムを起動して動作を確認した．

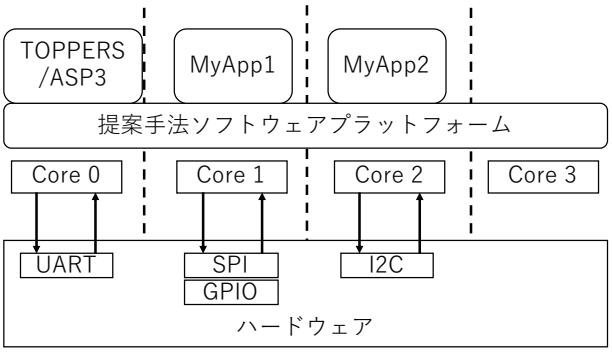

図 5 動作検証

5.2 動作検証結果

TOPPERS ASP3 のログ出力を確認しつつ，気温表示，カウンターがそれぞれ動作することを確認した．これにより各コアにソフトウェアを割り当てて動作することが確認できた．

6. 課題

組込みソフトウェアの中には，単体で機能が完結するものに限らず，組込みソフトウェアが互いに通信を行い連携するものがある．互いに通信を行う組込みソフトウェアを統合すると物理的なインターフェイスを削減することが可能であるが，組込みソフトウェア同士が通信するためのインターフェイスがなくなってしまう．機器同士がイーサネットなどを用いて行なっていた通信を，コア間割込みを利用したプロセッサ間通信や，共有メモリ領域を利用した通信，通信用ハードウェアをエミュレーションした通信に置き換える必要がある．よって提案手法ソフトウェア上の機能として，組込みソフトウェアが通信を行うためのインタフェースを提供する機能が必要となる．

7. おわりに

本論文では，複数の組込みソフトウェアをマルチコア環境において制御するソフトウェアプラットフォームについて述べた．提案した手法では，仮想化ではなく，マルチコア環境において物理コアを直接起動する組込みソフトウェアに割り当てることにより，複数組込みソフトウェアを動作させる．組込みシステム統合のためのマルチコア制御基盤ソフトに必要な機能のうち，組込みソフトウェアを 2 次記憶から読み込むローダ機能と CPU の初期化を行い電源投入直後の状態をエミュレーションする起動エミュレーション機能のプロトタイプ実装を行なった．動作確認の結果，デバイスの競合が起こらないように配慮して組込みソフトウェアを動作させることで，それぞれのソフトウェアが干渉することなく動作することが確認できた．

参考文献

[1] Inc. or its affiliates Amazon Web Services. FreeRTOS - Market leading RTOS (Real Time Operating System) for embedded systems with Internet of Things extensions. https://www.freertos.org/.
[2] Arm Limited (or its affiliates). Mbed OS — Mbed. https://www.mbed.com/en/platform/mbed-os/.
[3] TOPPERS Project Inc. TOPPERS プロジェクト／ASP3. https://www.toppers.jp/asp3-kernel.html.
[4] Barham, P.and Dragovic, B.and Fraser, K.and Hand, S.and Harris, T.and Ho, A.and Neugebauer, R.and Pratt, I., Warfield, and A. Xen and the art of virtualization. In *In Proceedings of the nineteenth ACM symposium on Operating systems prin- ciples*, pp. 164–177, 2003.
[5] 渡邉和樹, 片山吉章, 松本利夫, 瀧本栄二, 樫山武浩, 毛利公一. 仮想計算機モニタ xen における rtos 向け割込み通知機構. コンピュータシステム・シンポジウム ComSys2011 論文集, pp. 22–31, 2011.
[6] W. Kanda, Y. Yumura, Y. Kinebuchi, K. Makijima, and T. Nakajima. Spumone: Lightweight cpu virtualization layer for embedded systems. In *2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*, Dec 2008.
[7] 中嶋健一郎, 本田晋也, 手嶋茂晴手, 高田広章. セキュリティ支援ハードウェアによるハイブリッド os システムの高信頼化. 電子情報通信学会論文誌. D, 情報・システム, Vol. J93-D, No. 2, pp. 75–85, feb 2010.
[8] R. Wilhelm and J. Reineke. Embedded systems: Many cores — many problems. In *7th IEEE International Symposium on Industrial Embedded Systems (SIES'12)*, pp. 176–180, June 2012.
[9] 佐藤雅英, 関口知紀, 新井利明, 井上太郎, 宮崎義弘, 中橋晃文, 梅都利和. ナノカーネル方式による異種 os 共存 技術 「darma」 の実装. 全国大会講演論文集, p. 59, 1999.
[10] 新井利明, 関口知紀, 佐藤雅英, 井上太郎, 中村智明, 岩尾秀樹. ナノカーネル方式による異種 os 共存技術「darma」の提案. 全国大会講演論文集, 1999.
[11] 加藤直, 齋藤雅彦, 大野洋, 中村智明, 上脇正, 井上太郎. 組込み向けデュアル os 実行システム darma の開発. 全国大会講演論文集, 1999.