組込みシステム用プロセッサのセキュリティ拡張向け基本ソフトウェア

M2023SE001 江塚俊介
指導教員：本田晋也

1　はじめに

IoT 機器の増加に伴い，ネットワークに接続されている組込み機器（デバイス）が増加している．そのため，ネットワークを経由した脅威に対して，機器やデータを守るためのセキュリティ対策が必要となってきている [1].

セキュリティ対策の機能を実現するためのプラットフォームとして TEE（Trusted Execution Environment）がある．既存プロセッサにセキュリティ拡張として，既存のプログラム実行モードに加えて，プロセッサに新たな実行モードである S-World を追加する．そして，S-World のみアクセス可能なメモリの領域を用意してセキュアなデータを保護する．

ARM 社の TrustZone-M は，小規模組込みシステム向けのセキュリティ拡張であり，各 World の割込み応答時間や World 切り替えの実行オーバーヘッドを小さくしたいという要求を満たすため，多くの機能をハードウェアで実現しており，ハイパーバイザーのような専用のソフトウェアは必要としない．現状，TrustZone-M は一般に使用可能な唯一の小規模組込みシステム向けのセキュリティ拡張でありハードウェアの脆弱性も発見されている [2]．TrustZone-M の脆弱性への対策として，ARM 社以外の小規模組込みシステム向けのプロセッサでのセキュリティ拡張を実現して利用する方法がある．

Cadence 社の Xtensa は，小規模組込みシステム向けのプロセッサであり，近年セキュリティ拡張が追加されたため，TEE を実現可能である．Xtensa のセキュリティ拡張は多くの機構をソフトウェアで実現するという特徴がある．そのため，TEE の実現方法として，TrustZone-M 向けの手法は使用できない．また，Xtensa は，ヘテロジニアスマルプロセッサにおけるサブプロセッサとして使われていることが多く，メインプロセッサがセキュリティ拡張を持つ場合，サブプロセッサとしてどのようなソフトウェアが必要であるかを明らかにする必要がある．

本研究では TrustZone-M とハードウェアアーキテクチャが異なる Xtensa のセキュリティ拡張を対象した，リアルタイム性を重視した TEE の実現を目的とする．まず，Xtensa のユースケースより，シングルプロセッサ及びヘテロジニアスマルプロセッサのそれぞれの状況において，TEE に求められる機能について整理する．次に，必要となる機能を実現するための TEE の実装を行い，実行オーバーヘッドやソフトウェアの複雑度を明らかにする．具体的には，RTOS ベースの TEE 及び N-World 間との通信機能を実現することにより，Xtensa のセキュリティ拡張の評価を実施する．

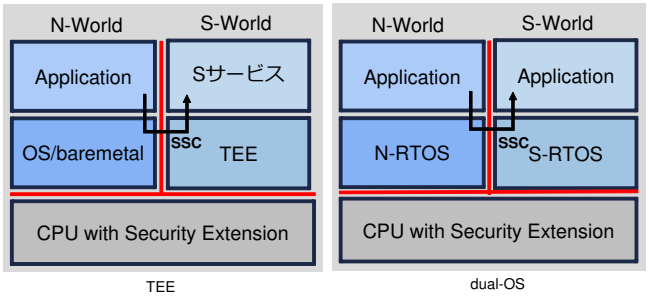

図 1　組込みシステム向けセキュリティ機構を用いたプラットフォームソフトウェアの構成

2　背景技術

2.1　組込みシステムのセキュリティ機構の利用

組込みシステム向けのセキュリティ機構を用いたプラットフォームソフトウェアの構成を図 1 に示す．プロセッサにセキュリティ拡張が追加され，既存のプログラム実行環境（N-World）から保護された環境（S-World）を用意し，データの暗号化や認証といった S サービスを提供する TEE を実行する [3]．TEE の構成には様々なバリエーションがあり，文献 [4] では，TEE を用いた構成を 3 つに分類している．S サービスは，N-World から SSC（Secure Service Call）として呼び出される（図 1 内の矢印）[5].

TEE 以外のセキュリティ拡張を利用したソフトウェアの機構として，dual-OS がある [6]．dual-OS は，TEE とは異なり，両 World でそれぞれアプリケーションを実行する．N-World では既存アプリケーションを実行し，S-World ではリアルタイム OS（S-RTOS），リアルタイム性が必要な処理，セキュアサービスを実行する．

2.2　Xtensa とセキュリティ拡張

Xtensa は機能や命令を必要に応じて変更可能なコンフィギュラブルプロセッサである．AI や音声処理を高速に処理する命令を追加可能である．Xtensa にはセキュリティ拡張として Cadence Tensilica Xtensa LX Secure Mode（XLS）を有している．XLS の拡張内容は次の通りである．

(拡張 1) ステータスレジスタに実行 World を表すビットを追加

(拡張 2) メモリの特定領域を Secure 領域に設定可能

(拡張 3) S-World で MPU の設定をロック可能

（拡張 1）は，ステータスレジスタに World の状態を示すビットが追加されており，バスアクセス時には実行中の World の情報がメモリや周辺回路に送られる．また，N-World ではビットの変更が不可能となっている．プロセッサのレジスタは両方の World で共有しており，World を切り替え時にはソフトウェアにより保存・復帰する．

（拡張 2）については，Secure 領域のメモリや周辺回路は S-World からのみアクセス可能である．割込みベクターは全て Secure 領域に配置される．Xtensa には low 割込みと mid 割込みの 2 種類の割込みがあるが，全て S-World で受け付けることになる．N-World で割込みを受け付けたい場合は，S-World で受け付けた後，N-World への遷移処理をソフトウェアで実現する必要がある．

2.3 ヘテロジニアスマルチプロセッサ

小規模な組込みシステムにおいても機械学習処理や DSP 処理等の負荷の高い処理が増加している．そのため，必要に応じた専用命令を持つプロセッサが必要となる．

一方，組込みシステムにおいては，近年 ARM 社のプロセッサである Cortex-M を用いることが一般的になっている．Cortex-M は，専用の命令を追加することが不可能であり，上記の要求を満たすことができない．

そこで，Cortex-M に加えて，命令の追加が可能なコンフィギュラブルプロセッサを組み合わせるヘテロジニアスマルチプロセッサが用いられる．Cortex-M はメインプロセッサとして通常のソフトウェアを実行し，コンフィギュラブルプロセッサは応用毎の処理を高速化する命令を追加しサブプロセッサとしてメインプロセッサから特定の処理をオフロードして実行する．ヘテロジニアスマルチプロセッサは，メインプロセッサとサブプロセッサの命令セットが異なるため，それぞれ独立したバイナリを実行する．

現状市場で販売されているヘテロジニアスマルチプロセッサの多くは，TrustZone-M を備えた Cortex-M を使用しているものが多い．

3 セキュリティ拡張を持つヘテロジニアスマルチプロセッサ向けのソフトウェア構成

本章では，メインプロセッサ及びサブプロセッサ共にセキュリティ拡張を持つヘテロジニアスマルチプロセッサのソフトウェア構成を検討する．

ヘテロジニアスマルチプロセッサにおいて，メインプロセッサでセキュリティ拡張を用いた構成は，図 1 の TEE と dual-OS に分類できる．

3.1 メインプロセッサで TEE を実行するケース

本ケースのソフトウェア構成を図 2 の上段に示す．メインの処理は N-World で実行され，セキュリティに関する処理は S-World の S サービスを呼び出し，AI などの処理はサブプロセッサを使用する．メインプロセッサの S-World で実行される S サービスからサブプロセッサの処理を呼び出すケースは考えにくい．そのため，サブプロセッサは，メインプロセッサの N-World のみ利用可能であればよく，サブプロセッサはメインプロセッサの S-World のみ使用可能なリソースにアクセス不可とすればよい．

3.2 メインプロセッサで dual-OS を実行するケース

本ケースのソフトウェア構成を図 2 の下段に示す．メインプロセッサでは，S-World で信頼性やリアルタイム性が

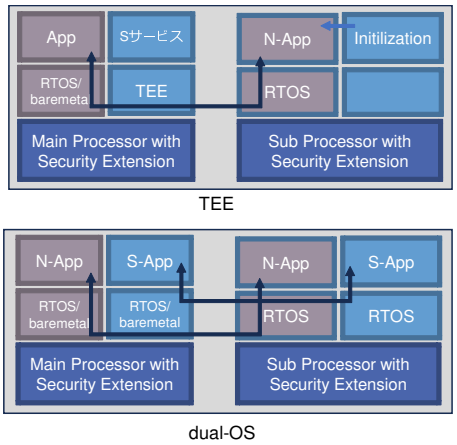

図 2 ヘテロジニアスマルチプロセッサ向けのソフトウェア構成

高い既存のアプリケーションを実行し，N-World ではセキュリティの脅威に晒される可能性があるネットワーク関連等のアプリケーションを実行する．それぞれの World のアプリケーションは，AI やデジタル信号処理をサブプロセッサに依頼することになる．そのためサブプロセッサはメインプロセッサの各 World の要求を処理独立して処理可能な dual-OS とする必要がある．また，プロセッサ間での World 間通信が必要になる．

4 Xtensa 向け dual-OS

本章では，メインプロセッサで dual-OS を実行するケースを対象に，サブプロセッサとして Xtensa を使用する際に必要となる XLS を利用した dual-OS の設計と実装について述べる．

4.1 dual-OS の構成

dual-OS では，S-World のリアルタイム性を損なわないようにするため，S-World 側の処理が全て N-World より優先して実行する必要がある．そこで以下を行う．

- 全ての low 割込みは N-World への割込み，全ての mid 割込みは S-World への割込みとして扱う．
- S-World 実行時は，low 割込みは常に全て禁止する．
- N-World 実行時は，mid 割込みは常に許可する．
- S-World から N-World への遷移方法は Normal 実行関数（call_ns()）のみとする．

S-World で，call_ns() を呼び出すタイミングは，ユーザーが選択可能であり，様々なスケジューリングが実現可能となる．Secure 処理を優先的に行う IDLE スケジューリングは，S-RTOS の最優先度タスクから call_ns() を呼び出すことで実現可能である．

4.2 dual-OS の実現

前節の設計で説明した内容の実現方法について述べる．

S-World 実行時は，low 割込みを禁止する．これは S-RTOS の割込み許可 API を変更することにより実現する．

割込み受付時の遷移シーケンスの概要を図 3 に示す．S-World 実行時の mid 割込みは，オリジナルの RTOS と同じ扱いとなり，ISR を呼び出す．N-World 実行時の low

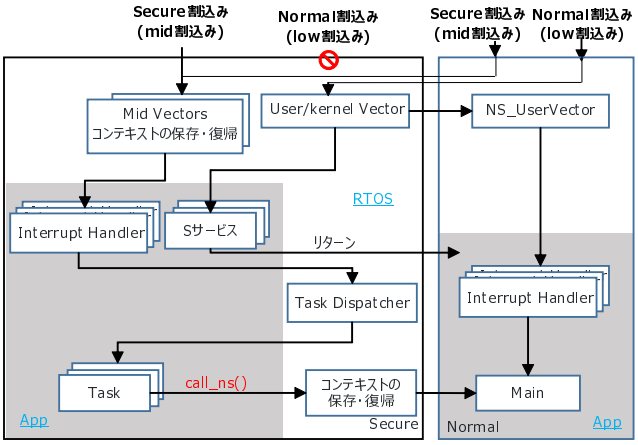

図 3 割込み遷移シーケンス

割込みは，S-RTOS の割込みベクター処理後，N-World に遷移し ISR を実行するよう変更する．N-World 実行時の mid 割込みは，N-World のコンテキストを保存及び S-World のコンテキストの復帰処理を追加し，その後 ISR を実行する．

call_ns() は，S-World のコンテキストを保存した後，mid 割込み受付時に保存した N-World のコンテキストを復帰後，N-World に遷移する．

4.3 SSC の実現

SSC の実現はソフトウェア割込みを発生させる syscall 命令を使用する．SSC 毎に，ID を割り付けて識別する．また SSC 毎に，呼び出しに必要な処理を行い syscall 命令を呼び出す，veneer 関数を用意する．SSC を実装するにあたり次のことを行った．

N-World での veneer 関数呼び出し

veneer 関数では，引数と SSC の ID をレジスタに保存，syscall を発行し S-World に割込みを入れる．レジスタを使用する理由として，メモリを使用すると，S-World 側でそのメモリの正当性のチェックが必要になるためである．

S-World の関数呼び出しまでの実行パス

図 3 より S-World のベクターで syscall 呼び出しと判断したら，ISR と分岐する．そして N-World のコンテキストを保存し，SSC 呼び出し関数を実行する．

SSC 呼び出し関数

SSC の関数テーブルを用意し，SSC の ID に応じて引数をセット，SSC の関数を呼び出す．

SSC からのリターン処理

戻りアドレス，N-World のコンテキストの復帰を行い，セキュアサービスで使用したレジスタをゼロクリアした後，N-World にリターンする．

4.4 World 間通信ライブラリ

World 間通信の通信ライブラリとして RPMsg を用いる．RPMsg はヘテロジニアスマルチプロセッサ向けの World 間通信として広く用いられており，エンドポイントを用いて通信を行う．本研究では，互換性を考慮してベアメタル環境で動作するように変更を行った．ベアメタルでは，受信時にユーザ独自のコールバック関数を使用してデータを受信する．

5 評価

本章では，実装した dual-OS がセキュリティ拡張を使用しない環境（single-OS）で実行される RTOS と比較して実行オーバーヘッドの増加がどの程度か評価する．またヘテロジニアスマルチプロセッサ環境での RPMsg の評価を行い，Xtensa に与える影響を考える．

5.1 評価環境及び評価項目

メインプロセッサに ARM 社の Cortex-A，サブプロセッサは FPGA 上に Xtensa を実装した kr260 マイコンボードを用いた．動作周波数は Cortex-A は 1.5 GHz，Xtensa は 100MHz である．

評価項目は次の通りである．

- タスク切り換え時間
  single-OS 実行時，dual-OS の S-World 実行時，N-World 実行時のそれぞれで，高優先度のタスクをタスク起動 API を呼び出した後，高優先度のタスクが実行されるまでの時間を計測した．
- 割込み応答時間
  各 World 実行中に割込みを受け付けて ISR が実行されるまでの時間．
  - SOS-M：single-OS mid 割込み
  - SOS-L：single-OS low 割込み
  - DOS-S-M：dual-OS S-World 実行時の mid 割込み
  - DOS-N-M：dual-OS N-World 実行時の mid 割込み
  - DOS-N-L：dual-OS N-World 実行時の low 割込み
- SSC 呼び出し時間
  空の SSC を呼び出してリターンするまでの時間．
- RPMsg 実行時間
  RPMsg の送信 API を実行してから受信コールバック関数でデータを受信するまでの時間（表 1）．XS-XN は即座に割込みが入らないため評価外とした．

表 1 RPMsg 実行時間 項目

	送信元	送信先
XN-XS	Xtensa の N-World	Xtensa の S-World
AN-XN	Arm の N-World	Xtensa の N-World
XN-AN	Xtensa の N-World	Arm の N-World
AS-XS	Arm の S-World	Xtensa の S-World
XS-AS	Xtensa の S-World	Arm の S-World

5.2 評価結果：タスク切り替え時間

全項目 20,000 回ずつ計測を行った．実行時間を比較すると実行時間に差はなかった．これは dual-OS を実現す

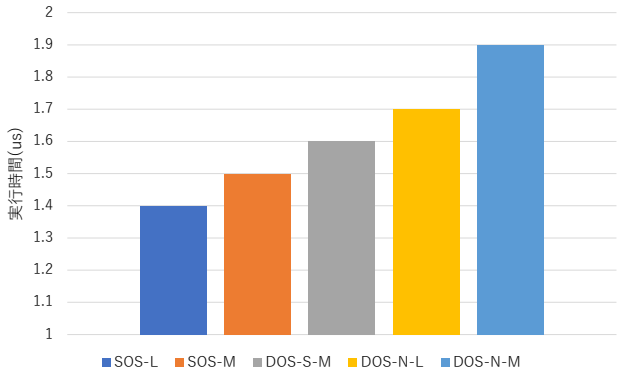

図 4　割込み応答時間

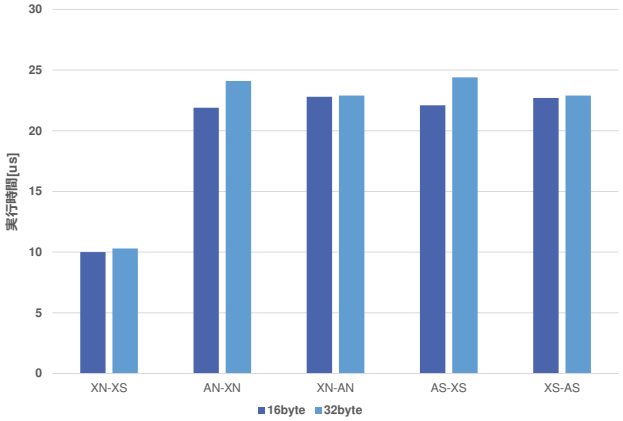

図 5　RPMsg 実行時間

るにあたり OS 内部の変更を最小とし，ハンドラのエントリ部分の変更で済んだため，タスク切り換えのシーケンスは増加していないことがわかる．

なお，dual-OS（N-World）では 1 回目に大きな値がみられた．これは N-World はプログラムがキャッシュが有効なメモリに置かれているためである．

5.3　評価結果：割込み応答時間

割込み応答時間を図 4 に示す．全項目 1,000 回ずつ計測を行った．

DOS-N-L のみ試行回数の 1，2 回目の実行時間が大きくなった．その他の項目では 1,000 回同じ実行時間となった．

dual-OS で S-World に発生する実行オーバーヘッドについては，DOS-N-M を DOS-S-M と比較すると約 1.2 倍実行時間が増加している．DOS-N-M を SOS-M と比較して実行時間が増加しているのは図 3 から ISR 処理の前に N-World のコンテキストを保存，S-World のコンテキストを復帰するためである．また DOS-N-L を SOS-L と比較して実行時間が増加してるのは一度 S-World へ遷移し OS 処理を挟むためである．

5.4　評価結果：SSC 呼び出し時間

SSC 呼び出し時間は 840ns であった．また同じ関数を N-World に配置，実行したときの時間は 250ns であった．SSC は通常の関数呼び出しの約 3 倍実行時間がかかる．

5.5　評価結果：RPMsg 実行時間

RPMsg 実行時間を図 5 に示す．データサイズを 16byte，32byte で全項目 1,000 回ずつ計測を行った．

Xtensa の N-World が関わる項目では，5.2 節と同様の理由で 1 回目の実行時間が大きくなった．XN-XS は他の項目と比較して高速である．RPMsg に必要な共有メモリの領域に高速なローカルメモリを使用しているためである．

5.6　評価結果：S-RTOS の変更量

S-RTOS の変更量は，割込み処理が 5ヶ所，OS 処理は 4ヶ所であった．

6　おわりに

本研究では，Xtensa 向け dual-OS の設計及び実装と，World 間通信として SSC，プロセッサ間の World 間通信として RPMsg を実現した．また dual-OS 環境及び SSC，RPMsg の性能評価を行った．

今後は，メインプロセッサで汎用 OS を実行するヘテロジニアスマルチプロセッサのサブプロセッサとして Xtensa プロセッサを使用できるように，S-RTOS の改良，ソフトウェア構成の再構築を考えている．

参考文献

[1] 独立行政法人情報処理推進機構 (IPA) : *IoT*開発におけるセキュリティ設計の手引き, (2024). `https://www.ipa.go.jp/security/iot/iotguide.html`, (参照 2024-09-23).

[2] Zheyuan Ma, Xi Tan, Lukasz Ziarek, Ning Zhang, Hongxin Hu, and Ziming Zhao : *Return-to-Non-Secure Vulnerabilities on ARM Cortex-M TrustZone: Attack and Defense*, ACM/IEEE Design Automation Conference (2023).

[3] Jinwen Wang, Ao Li, Haoran Li, Chenyang Lu, Ning Zhang : *RT-TEE: Real-time System Availability for Cyber-physical Systems using ARM TrustZone*, 2022 IEEE Symposium on Security and Privacy (SP) (2022).

[4] Xi Tan, Zheyuan Ma, Sandro Pinto, Le Guan, Ning Zhang, Jun Xu, Zhiqiang Lin, Hongxin Hu, Ziming Zhao : *Where's the "up"?! A Comprehensive (bottom-up) Study on the Security of Arm Cortex-M Systems*, 18th USENIX WOOT Conference on Offensive Technologies (WOOT 24), pp.149-169 (2024).

[5] Pan Dong, Zhe Jiang, Alan Burns, Yan Ding, Jun Ma : *Work-In-Progress: Real-Time RPC for Hybrid Dual-OS System*, 2019 IEEE Real-Time Systems Symposium (RTSS) (2019).

[6] 小森工，本田晋也：*SafeG-M: ARMv8-M TrustZone* を利用した組込み向けデュアル *OS* 実行環境，コンピュータソフトウェア，vol.41, pp.97-114 (2024).