3A-1

組込み CPU 向け高信頼基盤ソフトウェアの開発

出原　章雄†　　山本　整†　　東山　知彦†　　落合　真一†

三菱電機株式会社 情報技術総合研究所†

1. はじめに

近年の組込み H/W の高性能化に伴い，従来は複数 H/W プラットフォーム(以下，PF)で実現していたシステムをマルチコアの単一 H/W PF で実現可能になりつつある．

こうしたシステムでは，一方の OS の障害が，他 OS に影響し，動作が停止する可能性がある．複数 PF で実現している場合，障害が発生した PF を再起動させるが，単一 H/W PF の場合，H/W 的な再起動は，正常動作中の他 OS にまで影響を与えてしまう．

そこで今回，一方の OS に障害が発生した際に，他方の OS に影響を与えることなく，再起動可能な基盤ソフトウェアを開発した．本稿ではこの内容について述べる．

2. 課題

今回の構成では，組込み向けのマルチコア CPU を使用し，各コア上で異なる OS を動作させること(以下，OS 分離動作)，および，複数 OS 動作時に一方の OS を再起動(以下，OS 再起動)させることを実現する．以下，これらの課題を検討する．

(1) OS 分離動作

(a) H/W リソース分離

各 OS で同じ H/W リソースを使用しないように H/W を設定するという課題がある．この課題を解決することにより，ある OS が動作中に他の OS の挙動を阻害することがなくなる．

(b) 複数 OS 起動

初期起動時における各 OS の起動方法について，H/W の制約等に応じて，OS イメージのロード方法や OS の起動順等を決定するという課題がある．

(2) OS 再起動

(a) デバイス初期状態化

通常，OS が再起動する際に，H/W PF 全体にリセット(以下，H/W リセット)を行うが，この処理は動作中の他 OS の動作を阻害する．そこで OS 再起動時には，H/W リセットではなく，OS の初期化処理から実施する必要がある．ここで，OS の終了時，デバイスドライバによっては，H/W リセットを想定した処理となっており，デバイスの状態を OS 起動前の初期状態に戻していない．こうしたドライバが存在する場合，OS 再起動時のデバイスの初期化処理に失敗することがある．そのため，デバイスの状態を OS 起動前の初期状態に戻した上で，OS 起動処理を実行するという課題がある．

(b) OS イメージ再利用化

OS 再起動時に使用する OS イメージについて，初期起動時に使用した OS イメージを再利用できるようにする，という課題がある．

以上をまとめると図 1の通りとなる．

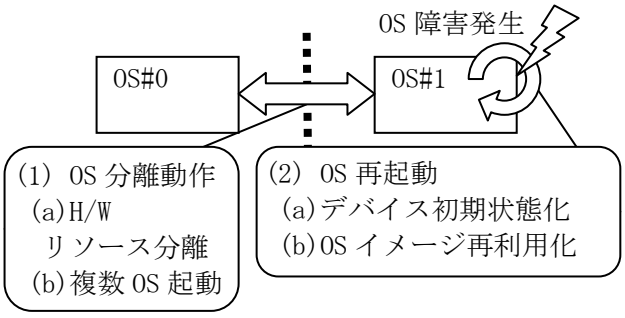

図 1　今回実現する OS 分離動作，OS 再起動

3. 設計と実装

今回ターゲットとした H/W およびベースとした S/W を表 1に示す．

表 1　ターゲット H/W およびベース S/W

H/W	PandaBoard CPU:OMAP™4430(Cortex™-A9(1GHz x 2)) MEMORY:LPDDR SDRAM 1GB
S/W	OMAP™4 用 Ubuntu11.04 OS:Linux(R) 2.6.38 ブートローダ:U-Boot 2011.06

3.1. OS 間分離動作

通常，Linux は単一 OS で動作しているとして，H/W PF 上に搭載されているデバイスを可能な限り利用する．そのため，同一 H/W PF 上で複数の OS を動作させるには，必要なデバイスを各 OS に割り当てるとともに，各 OS の起動についても設計する必要がある．本節ではこの内容について述べる．

3.1.1. H/W リソース分離

H/W リソースとして，シリアルデバイス，タイマ等，OS の基本動作に必要なデバイスを分離し，各 OS に割り当てた．また，メモリ領域についても，各 OS で重ならないように割り当てた．

A development of reliable software platform for embedded CPU
† Akio Idehara, Hitoshi Yamamoto, Tomohiko Higashiyama and Shinichi Ochiai, Information Technology R&D Center, Mitsubishi Electric Corporation

3.1.2. 複数 OS 起動

複数 OS を起動する場合，OS イメージのロードの処理等を行うブートローダを拡張し，各 OS を起動可能にするか，OS を拡張し，他方の OS を起動するかを検討する必要がある．今回の構成では，OS 再起動に際してブートローダの介在をなくすため，一方の OS から他方の OS を起動することとした．

3.2. OS 再起動

Linux では復旧不可能な障害が発生すると OS パニックとなり，汎用レジスタの状態などを出力した後，OS 機能を停止する．今回の構成では，一方の OS 障害が発生した際に，他方の OS に影響を与えることなく，OS 再起動を行う必要があるため，OS パニック処理に修正を行った．本節ではこの内容について述べる．

3.2.1. デバイス初期状態化

Linux が OS 機能を停止する際の処理について，デバイスドライバによっては H/W リセットを想定した S/W 処理となっている．そのため，各デバイスの状態を OS 起動前の状態に戻すかどうかは，各ドライバの実装レベルによる．今回使用した Linux のバージョンでは MMU(Memory Management Unit)に対応が必要であった．

MMU は OS 終了処理にて，H/W リセットを想定した処理となっており，MMU 機能の無効化が実施されていなかった．しかし，OS 終了時に，単に MMU の機能を無効にした場合，物理アドレスと仮想アドレスのマッピングがずれてしまうため，以降の処理が継続できなる．そこで，OS 終了時に MMU の機能を無効にする処理については物理アドレスと仮想アドレスが等しいマッピングになるように変更した．これにより，MMU の機能を無効にした後も，OS 終了処理が継続可能となる．

3.2.2. OS イメージ再利用化

OS 再起動時について，再起動を実施するたびに同一の OS イメージを使用することとした．通常の ARM(R)用 Linux は，OS 起動時に OS イメージ内の GOT(Global Offset Table)領域を書き換えるため，再利用できない．そこで，OS イメージを ROM 化して使用可能となるオプション(CONFIG_ZBOOT_ROM)を有効にし，GOT 領域が変更されないようにした．

4. 評価と考察

4.1. ソースコード変更量

ソースコードの変更量を表 2に示す．

表 2　ソースコード変更量

	変更量	
	追加	削除
OS#0	35 行	3 行
OS#1	436 行	21 行
計	471 行	24 行

全体で 500 行程度の修正となり，小規模な修正となった．

4.2. OS 障害時再起動確認

OS パニックを生成する簡易的なドライバを用いて OS 障害を模擬し，他方の OS に影響することなく，OS が再起動可能となることを確認した．

4.3. OS 再起動時間

今回開発した OS の再起動時間，および，初回起動時間を表 3に示す．

表 3　OS 起動時間内訳

	再起動	初回起動
OS イメージ展開～カーネル処理	1.6	2.0
カーネル処理～ユーザランド処理	1.3	1.3
ユーザランド処理～ログイン表示	1.6	1.8
合計	4.6	5.1

(単位[s])

今回開発した OS の再起動にかかる時間は 5 秒程度であり，OS の初回起動と同程度の起動時間となった．これにより，一般的な処理時間で OS 再起動を実現できた．なお，初回起動と比較して再起動が 0.5 秒速い点について，初回起動時は，複数の OS が同時に起動しており，キャッシュ処理等に負荷がかかっているためと考えられる．

5. おわりに

今回，一方の OS に障害が発生した際に，他方の OS に影響を与えることなく，OS の再起動が可能となる基盤 S/W を開発した．今後は，OS 間メモリ保護や H/W デバイス保護等，CPU 内蔵のセキュア機能を活用することにより，より堅牢性の高い基盤 S/W の開発を行う予定である．

参考文献

[1] 茂田井他，「組込み向け CPU 仮想化技術対応ハイパーバイザの設計」，ESS2012 2012.10
[2] 菅井他，「シングルチップマルチプロセッサ上のハイブリッド OS 環境の実現」，情処第 66 回全国大会 2004.3