	TOPPERS Automotive Kernel 基本タスクスタック共通化検討		

0 背景となる状況

SH2系での割込み処理スタックの問題点

SH2E(SH1,SH2も同様)での割込み抑止は、割込みレベルでのみ行い、割込み禁止フラグを持たないため、割込み処理開始時には、より優先度の高い割込みが許可された状態である。
（M32等は、割込み処理開始時に、割込み禁止フラグがセットされている）
このため、複数の割込みが同時に発生した場合、割込みハンドラがタスクスタックから割込みスタックへ切り替える前に、多重割込みを受け付ける場合がある。
よって、タスクのスタックサイズ決定時は、多重割込みを受け付けることが可能なだけの予備のＲＡＭをタスク毎に確保する必要があり、タスク数が多い場合には、メモリを圧迫する要因となる。

この問題を解決すべく、タスクスタックを共通化する検討を行い、実装した。

なお、SH2A でも同様であるが、割込み時のレジスタ退避はレジスタバンク機能で実現し、スタックを使用しないため、スタック共通化の必然性は低い。

	TOPPERS Automotive Kernel 基本タスクスタック共通化検討		

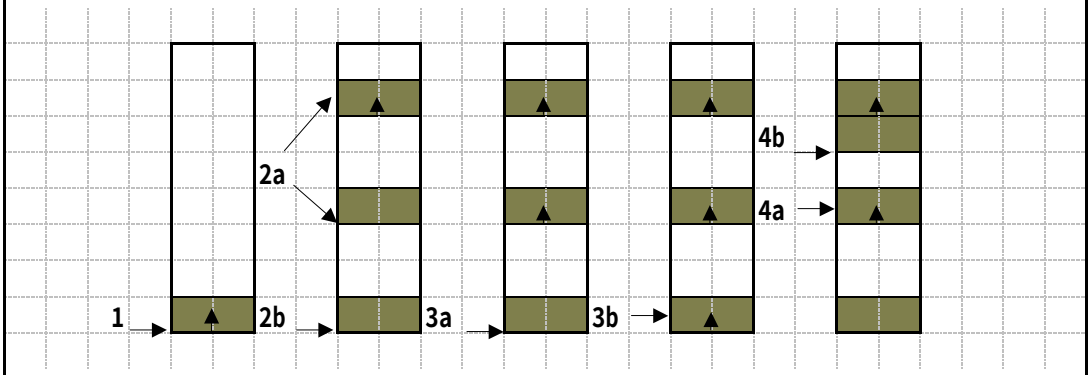
1．スタック管理概要

1.1 スタック操作個所

TOPPERS Automotive Kernel では、スタックポインタ(以下 SP) の操作は全て機種依存部で行っている。
設定個所と、設定値を以下に示す。

(1) リセット後
　SPは、HEWで設定したスタック(セクション S 末尾)
(2) OS実行中
　(2a) 実行すべきタスクがある場合、タスクスタックへ切り替える。
　　タスクが初回起動の場合に、タスク毎のSP初期値に設定。
　　ディスパッチ後の場合は、中断時に保存した SPを復元する。
　(2b) 実行すべきタスクが存在しない場合、タスクアイドル処理で、割込みスタックへ切り替える。
　　SPは、HEWで設定したスタック(セクション S 末尾)
　　(OS の起動処理は完了しているので SPを初期値にしてもよい)
(3) ISR2 起動時に、割込みスタックへ切り替える。
　(3a) タスクスタック使用中の場合は、割込みスタックに切り替える。
　　SPは、HEWで設定したスタック(セクション S 末尾)
　(3b) 割込みスタック使用中（タスクアイドル or 多重割込み)の場合は、そのまま使用する。

(4) ISR2 からのリターン時に、タスクスタックへ戻す。
　(4a) 元のタスクに戻る場合は、タスクスタックを戻す。
　　SPは、割込みスタックへの切り替え前のタスクスタック
　(4b) ディスパッチが必要な場合、別のタスクスタックへ切り替える。
　　SPは、(2a)のディスパッチ後と同様に、タスク毎の値。



なお、タスク切り替え時に、PC と SP を保存するための領域として、TCB を使用する。
以後、実行中の SP を保存する領域を TCB の SP 領域と記載する。

1.2 基本タスクスタック共用

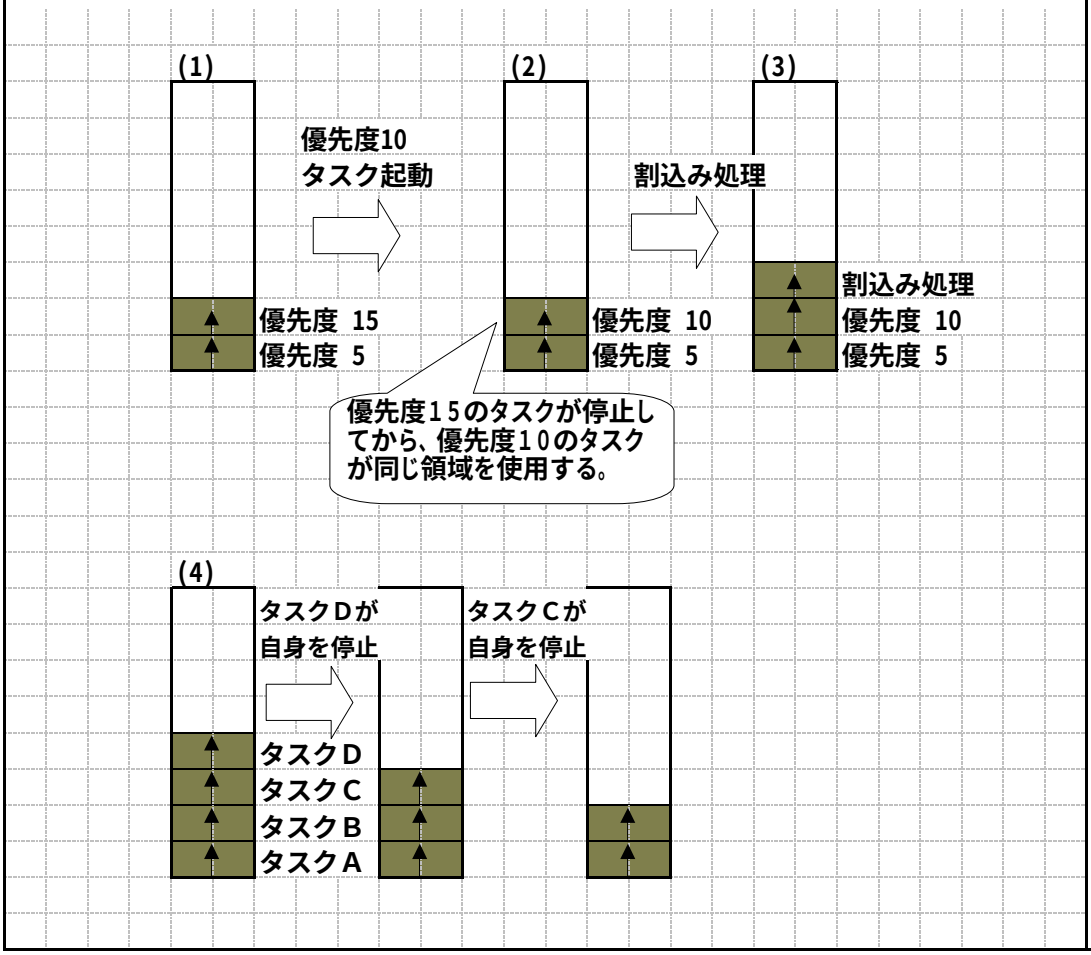
基本タスクが、スタックを共用できる理由を以下に示す。

(1) Running 状態の基本タスクは、より優先度の高いタスクによってのみプリエンプトされる。優先度の高いタスクが基本タスクの場合、一旦プリエンプトされたタスクは、より優先度の高いタスクが終了するまで実行できない。
これは、優先度の低いタスクのスタックポインタが変化しないことを意味する。
（個別の基本タスクのプリエンプションを関数コールと見なして、スタックを使用し、基本タスク群を一つの拡張タスクの様に管理する。）

(2) (1) の状態から、最優先のタスクよりも低いレベルのタスクを起動した場合、最優先のタスクが停止してから、実行を開始する。よって、スタックは、最低レベルのタスクが使用した末尾から使用する。

(3) 割込み処理中は、タスクの処理は行わない。
最優先度のタスクが起動したことと同じ考え方でよい。

(4) タスクを停止するときは、自分自身のみであり、他のタスクを停止することはできない。
よって、タスクスタックを解放する際は、常にスタックトップから解放することになり、不連続の領域が発生することはない。



1.3 基本タスクで共通スタックを使用することにした場合の検討

基本タスクを実行する際の、スタックに関連する処理ごとに検討する。

（1）SUSPEND → READY
ActivateTaskにより、タスクを READY にする場合。
SP初期値とPC初期値(activate_r) をTCBにセットする。
基本タスクの場合のSPは、共通スタックを使用することになるため、
RUNNING になったときに確定する。ここでは初期化しない。

（2）READY → RUNNING （初回起動）
dispatcher により、実行を開始する。
SPとPCをTCBから読み込む。
基本タスクの場合のSPは、TCB ではなく共通スタックトップのアドレスを
セットする。
また、初回起動時特有の処理として、タスク実行開始時のSPを保存する。
(理由：タスク停止時に、共通スタックトップのアドレスを元に戻すため。
基本タスクがプリエンプトされたときにスタックポインタを保存し、実行を
再開するときに、スタックポインタを戻すやり方では、中間レベルの基本
タスクが起動されていた場合には、対応できない。
よって、タスクが実行を開始する際のスタックポインタを記憶し、タスク
終了時に、スタックポインタを戻すことで実現する。タスク終了の
teminatetask が、自分自身を終了する仕様であることに依存して
いるため、OS仕様変更の際には、適用できない。)

（3）RUNNING → READY
ActivateTask により、他のタスクを起動する場合、dispatch() を
コールする。
この処理では、現在のSPとPCをTCBに保存する。
基本タスクの場合のSPは、TCB ではなく共通スタックトップのアドレスに
保存する。
（ここで保存したスタックトップのアドレスは、拡張タスクが実行中の間
ISR2 用スタックとして使用する。）

（4）RUNNING → READY （ISR2経由)
ISR2 処理で、他のタスクをRUNNINGとした場合。
ISR2処理から戻ってきたときのSPと割込み復帰処理のアドレスをTCBに
保存する。
基本タスクの場合のSPは、TCB ではなく共通スタックトップのアドレスに
保存する。

（5）READY → RUNNING （実行再開）
dispatcher により、実行を再開する。
SPとPCをTCBから読み込む。
基本タスクの場合のSPは、TCB ではなく共通スタックトップのアドレスを
セットする。

（6）RUNNING → SUSPEND
TeminateTask, ChainTask により、自タスクを停止する。
基本タスクの場合は、タスク実行開始時のSPを復元して、共通スタック
トップにする。（タスク実行開始時のSP == プリエンプトされたタスクのSP)

以上から、TCBにSPを保存する必要はないため空き領域にできる。
一方、タスク実行開始時のSPを保存する必要がある。
よって、TCBのSP退避領域の意味付けを基本タスクと拡張タスクで変更する。
基本タスク：
タスク実行開始時のSP初期値 (初回起動処理の activate_r で保存する)
拡張タスク：
タスクプリエンプト時のSP

1.4 割込みで共通スタックを使用することにした場合の検討

割込みスタック関連処理は、以下の作業が必要となる。

（1）割込み時、スタック設定処理
実行レベルがタスクの場合、割込みスタックへ切り替える。
このときのSPは、割り込みスタックトップ。
これを以下のとおりに修正する。
基本タスクの場合、スタックは切り替えない。
拡張タスクの場合、共通スタックトップへ切り替える。
（拡張タスクが実行中の場合は、基本タスクがプリエンプトされた状態であり、中断時の SP == 共通スタックトップ となっている。）

（2）タスクアイドル時、スタック設定処理
タスクアイドルでは、無条件に割込みスタックへ切り替えて処理を行う。
このときのSPは、割り込みスタックトップ。
ここは修正しない。
基本タスクには WAITING は無いので、アイドル状態であるとは、
基本タスク全てが SUSPEND になっている。
よって、共通スタックトップ == SP初期値　とする。

	TOPPERS Automotive Kernel 基本タスクスタック共通化検討		

２．実装

追加する変数は、以下とする。
common_stack_top　割込みスタックトップ

2.1 タスク起動時(make_active)

修正せず。
基本タスクの場合、SP領域の値をSPに設定することは無いので、どんな値でも問題なし。

2.2 タスク実行（start_dispatch)

基本タスクの場合、SPとして「共通スタックトップ」を入れる。
初回実行の場合に SP 初期値を保存する処理が必要であるため、activate_r の処理で、SPを TCB の SP 領域に保存する。

2.3 タスク中断（dispatch)

基本タスクの場合、SPを「共通スタックトップ」に保存し、TCBのSP領域には保存しない。
割込み処理後のディスパッチ処理(ret_int)も同様にする。

2.4 タスク停止（exit_and_dispatch)

TerminateTask, ChainTask 処理からコールされる停止処理では、TCB の SP 領域の値を取り出し、「共通スタックトップ」に保存する。
（これで、タスク実行前のスタックトップアドレスに戻る。なぜならば、今停止したタスクが、基本タスクでは最優先のタスクであるから）

停止処理に引き続き実行する start_dispatch では、次に実行するタスクが基本タスクなら、
このアドレスがSPとなる。
拡張タスクならば、当該タスクのスタックに切り替わる。

2.5 割込み入口処理

実行レベルがタスクだった場合、SPに「共通スタックトップ」を設定する。
なお、基本タスクの場合、SPの操作は必要ない。

2.6 スタック使用量修正

OIL ファイルにて記述する STACKSIZE は、基本タスクの場合 4 にする。
(現状のカーネルでは、スタックサイズ０を想定していないため、スタック領域の配列サイズが０となり、コンパイルエラーとなる。)

2.7 共通スタックサイズ検討

スタックの使用量は、タスクから ActivateTask したときと、ISR2からの ActivateTask では、異なるため、算出する場合には、以下を考慮すること。
ISR2経由だと、レジスタの全退避(R0-R14,PR,MACH,MACL,GPR,SR,PC) 84byte
経由しないと、Ｃに従った退避（R8-R14,PR,MACH,MACL,GPR,PC) 48byte