

1. SH72513 ターゲット依存部の概要

1.1 ターゲットプロセッサとシステム

ターゲットプロセッサ (CPU) として、SH-2A コアを内蔵した SH72513 をサポートした。
内部リソースは、システムタイマとして、CMTチャンネル0 を使用した。

他に、サンプルプログラムの動作用として、以下を割当てた。

- ・ コマンド送受信用
SCIA
- ・ サンプルカウンタ用
CMT(コンペアマッチタイマ) チャンネル1
- ・ ハードウェアタイマ ISR1 用
ATU-III(アドバンスドタイマユニット) ダウンカウンタD00
- ・ ハードウェアタイマ ISR2 用
ATU-III ダウンカウンタD10

ターゲットシステム (SYS) は、(株)ルネサステクノロジの SH72513を
搭載した評価用ボード 「SDK72513」をサポートする。

1.2 開発環境と実行環境

開発環境には、(株)ルネサステクノロジの開発環境(HEW ver.4)を用いる。
実行環境としては、SDKボード搭載の外付けフラッシュメモリに書き込んで
実行する。

High-performance Embedded Workshop Version 4.04.01.001
SuperH RISC engine Standard Toolchain (V.9.0.3.0)

2. SH-2A ターゲット依存部の機能

2.1 リソース

SH-2A ターゲット依存部では、次のプロセッサの内部リソースを使用する。

- ・ CMT(コンペアマッチタイマ) チャンネル0, チャンネル1
- ・ SC11 (115200bps, 8bit, NonParity, 1stop)
(通信速度は SDKボード内蔵のモニタプログラムと同一設定)
- ・ ATU-III(アドバンスドタイマユニット) タイマDダウンカウンタD00, D10

ターゲットシステムには、LED, LCD が取り付けられているが、使用しない。

2.2 割込み

2.2.1 割込み優先度

TOPPERS Automotive Kernelでは、プロセッサの割り込み制御として、割込み全禁止・
許可機能と割込みマスクレベルでの禁止・許可機能の両方を持つことを想定し
ている。しかし、SH-2E では、割込みマスクレベル機能しか持たない。そのため、
割込み全禁止・許可に対しては、割込みマスクレベルを最大に上げることで、
実現する。

2.2.2 割込み関連のカーネル内部関数

割込み全禁止は、カーネル内部の lock_cpu() が使用している。lock_cpu() は
カーネルリソースの更新時に使用するため、この内部で、割込みマスクレベルを
変更(set_ipr())することがある。
よって、lock_cpu() 中に割込みマスクレベル変更要求が来た場合には、変数に
値を保持し、unlock_cpu() 時に割込みマスクレベルとして、設定する。

2.2.3 割込みハンドラでの割込み要因クリア

SH72513 では、周辺デバイスへの書き込み要求とCPUの実行は非同期であるため、
周辺デバイスの割り込み要因クリアが書込まれる前に、割込みからリターンする
ことがありうる。この場合、再度割り込みが発生する。
よって、割込み要因クリア後に当該レジスタを読み込むことで、周辺デバイスと
CPUとを同期させる。

2.3 プログラムのRAM実行

SDK72513ボードでは、マイコン外付けのFLASH ROMを使用するため、内蔵ROMと
比較し、処理速度が低下する。
本サンプルでは、リセット直後の初期化処理にて、ROMの内容をRAMにコピーして
実行するようにしている。

2.4 基本タスクでのスタック共通化

2.4.1 スタック共通化の必要性

本サンプルでは、基本タスクでのスタックを共通化する処理を組み込んでいる。
(詳細は、別紙「基本タスクスタック共通化検討」を参照)

TOPPERS Automotive Kernel では、割込みハンドラの入口処理にて、タスクスタックから割込み
スタックへの切り替えを行ため、最小限のレジスタを退避する。

SH2 系プロセッサでは、割込み処理では、より上位の割り込みを許可する仕様で
あるため、レジスタの退避中に上位の割込みが発生した場合、多重にスタックへ
レジスタを退避することとなる。

つまり、各タスクで割込みレベル数分のRAMを確保する必要がある。

これを回避するため、WAITING 状態が無い基本タスクで共通スタックを使用する。共通スタックは割込みスタックと兼用とする。

効果としては、同時に実行することはない同一優先度のタスクにおいても、スタックを個別に確保する必要がないため、さらにメモリ占有量を削減できる。

2.4.2 スタック共通化での注意点

OILの設定で、タスク毎にスタックサイズを指定するが、基本タスクか拡張タスクかの識別には、イベントの使用有無で行い、基本タスクには、4を指定する。しかし、誤って、拡張タスクに4を指定した場合には、他のタスクのスタック破壊に到るため、注意が必要となる。

3. サンプルプロジェクトのビルド方法

サンプルプロジェクトは、2つのプロジェクトで構成してある。

- ・ libkernel
カーネル共通部分をビルドして、ライブラリ化するプロジェクト。
- ・ sample
サンプルプログラムと SDK72513用システムライブラリをビルドし、実行形式にするためのプロジェクト。

libkernel プロジェクトでビルドすると debug フォルダ下に libkernel.rel が作成される。

sample プロジェクトでビルドすると、SG フォルダ下の sample1.oil を参照し、SG を起動する。
その後、コンパイルし、libkernel.rel と結合する。

4. 独自プロジェクトのビルド方法

サンプルプロジェクトでは、オプション類を HEW の標準から変更している。
変更箇所は、以下のとおり。

- (1) インライン関数展開をコンパイラの最適化オプションで指定する。
- (2) インラインアセンブラを有効にするため、コンパイラの出力ファイルを *.obj から *.src へ変更する。

また、多数のインクルードディレクトリを指定しているため、プロジェクトを新規に構築するよりも、sample の環境を引き継いだ別のプロジェクトを作成することが効率的である。

それには、HEW の機能である「プロジェクトテンプレートに変換」してから、新規プロジェクトを作成するとよい。

詳細は、Help Library のクイックスタートガイドの下記の記述を参照のこと。

TN 0014: 独自のプロジェクトジェネレータを作成するには?