第8章 CPUコアもソフト構造も共通のご利益にあやかってみる

MyプログラムをどのCortex-Mマイコンでも共通に動かすポイント

石岡 之也

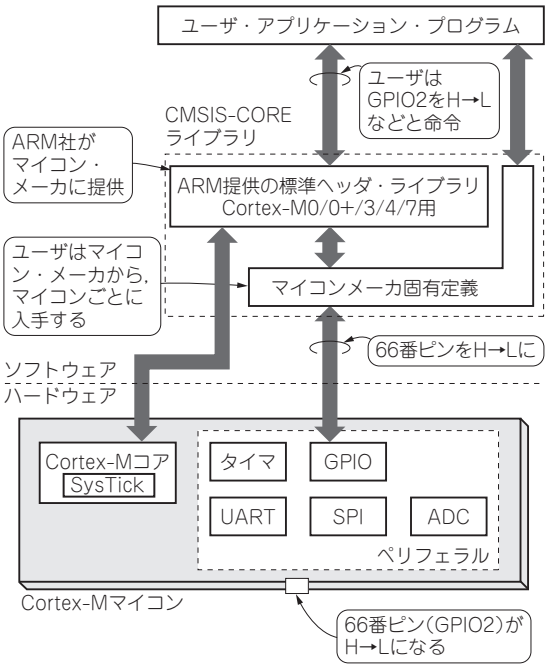

図1　CMSISソフトウェア規格の中でも特に重要なCMSIS-COREの移植のポイントを解説

第7章ではプログラム移植を簡単にするCortex-M用ソフトウェア・インターフェース規格CMSISの仕組みを紹介しました．

かつて，CMSISが提供される前のCortex-Mマイコン間の移植は，マイコン・メーカから出されるデータシートに記されたメモリ・マップやレジスタのアドレスから，各自で自分たちが使用するデバイスのアドレスやビットのオフセット値などを手作業でマクロで定義していました．

時は過ぎて，CMSIS規格の概念の下でCMSIS-COREに対応したソースコードが提供されてからは，Cortex-Mマイコン間の移植がしやすくなりました．

以上の経緯から，Cortex-M系マイコンの移植で最初に行わなければならない「CMSIS-CORE」の移植例のポイントを紹介します(**図1**)．

移植の要注意ポイント①…ベクタ・テーブル

Cortex-M系マイコンの起動直後にアクセスされるのが，ベクタ・テーブルです．ベクタ・テーブルは`startup_Device.s`ファイル内に置かれています(**図2**)．

リスト1は，IAR EWARM用のベクタ・テーブル部分の抜粋です．`sfe(CSTACK)`が0x00000000番地に配置され(**リスト1**の①)，以降，4バイトごとにハンドラのアドレスが配置されます．

ARMから提供されるテンプレートではCortex-Mコア固有の16個分が記述され，それ以降は`_IRQ Handler`で省略されていますが，実際のマイコン対応の`startup_Device.s`では，使用するマイコンが持つデバイスそれぞれのハンドラがベクタ・テーブルへ列記されています．

● 基本SysTickタイマ割り込みの例

ここではCortex-M系マイコンが共通して持ち[注1]，プログラム作成時にベースとなる時間を生成するSysTickタイマの割り込み(**リスト1**の②)の使い方を説明します．

▶`SysTick_Handler`の実処理を定義している箇所はプログラムが暴走しない記述になっている

`startup_Device.s`のさらに後方へソース表示を進めると**リスト2**の③のように`SysTick_Handler`の実処理を定義している箇所が現れます．

テンプレートでは，このままのコードを利用してもプログラムが暴走しないよう各割り込み(例外)ハンドラは呼び出されると自身のアドレスへジャンプする無限ループとなっています(**リスト2**の④)．

注1：Cortex-M0/M0+でオプション扱いになっているSysTickタイマが，実際には多くのM0/M0+で採用されているので共通と記している．