第1章 コンパイラ・オプション/アーキテクチャの違いを意識したコードの移植/追加機能の利用

M3からM33へ環境移行時のポイント

森本 登志子

この章では，Cortex-M3からCortex-M33への移行を行う場合のポイントについて解説します．

移行においては最低限コンパイラや統合開発環境（IDE）などの設定や，デバイス固有に関わる設定およびその他必要に応じたソースコードの修正が必要になります．

一方で，アーキテクチャ上，追加や拡張された機能を利用できるように，意図的にソースコードを変更することで機能や性能などの面で，より良いアプリケーションを目指すことも移行時のポイントの1つとなります．

開発環境の変更

さまざまな開発ツール・ベンダからArm開発用のソフトウェアが提供されています．

Armv8-Mアーキテクチャへの移行を行う際，仮に同じ開発ツールを使い続ける場合であってもある程度の変更は必要になります．しかし，どの程度であるかは開発ツールによって違いがあります．

単純にCPUなどを指定するオプションを切り替えるだけで基本的な移行ができるケースの他，Armv8-Mをサポートするために処理系が大きく変更され，バージョンアップが必要なケースもあります．

Arm純正のコンパイラの場合は後者になります．そこで，どういった変更が必要になるかの概要をArm純正の開発環境を例に紹介します．

● 言語処理系の移行

Arm純正のコンパイラでは，Armv8-Mアーキテクチャ向けプログラムを開発する場合，Arm Compiler 6を使用する必要があります．

Armv7-Mまでのアーキテクチャをターゲットとして，開発を行っていた場合，1世代前のArm Compiler 5を使用している場合が多いと思います．その場合，Arm Compiler 6への移行が必要になります．Arm Compiler 6はLLVMベースのコンパイラです．そのため，Arm Compiler 5と比較すると，コンパイラに対するオプションをはじめ，ソースコード内のディレクティブなどの書き換えが必要となるケースがあります．

また，アセンブリ言語はArmv7-MとArmv8-Mでは同じT32命令セットが使用できるため互換性があります．ただし，Armv8-Mアーキテクチャをターゲットとしたプログラムを開発する場合，Arm Compiler 6に含まれるコンパイラの実行形式armclangの機能の1つである，armclang integrated assemblerによるアセンブルが必要となります．これに伴い，ソースコード上にあるアセンブリ言語以外の部分，例えばディレクティブなどの書式などに違いが発生し，書き換えを行う必要があります．

このような処理系上の違いは開発ツール・ベンダのドキュメントに記載されています．Arm Compiler 6の場合はArm Compiler for Embedded Migration and Compatibility Guideと呼ばれるマニュアルです．

● 処理系変更に伴うIDEへの変更

言語処理系の移行に伴い，IDE側でもGUI上の変更が行われていることがあります．例えばArm純正の開発ツールMDK-Armに含まれるuVisionでは，プロジェクトにおいてビルドするコンパイラのバージョンをArm Compiler 5（**図1**）からArm Compiler 6（**図2**）に変更すると，コンパイラのオプションを指定するタブが自動的に切り替わり，Arm Compiler 6にあったオプションが指定できるようになっています．

IDEを利用してビルドする場合，基本的なオプションの違いについては単純にGUI上のチェック・ボックスやドロップダウン・リストなどから選択するオプションを切り替えるだけで自動的に適切なオプションが選択されます．

一方でmakefileなどを利用してビルドする場合は，オプションを記述する必要が出てきます．その場合，先に紹介をした，Migration and Compatibility Guideのような処理系のマニュアルを参照し，オプションなどの差異を確認して移行先の要件に合わせた設定変更が必要となります．