ダウンロード・データあります

Cortex-AのMMUで不正なメモリ書き込み/読み出し/実行を禁止する方法

徳山 琢郎

従来の組み込みシステムにおけるCortex-Aの使われ方

● リアルタイムOSではあまりMMUが使われていない

近年，Arm Cortex-Aを使った組み込みシステムが普及してきました．Cortex-AではOSとしてLinuxが使われるようになってきていて，そのLinuxではMMU（Memory Management Unit）の高度な機能を存分に活用したメモリ保護が実現されています．

一方で，リアルタイムOS（RTOS）を使ったシステムでは，MMUによるメモリ保護がサポートされているケースがほとんどないのが現状です．これまでのリアルタイムOSのターゲット環境にはマイコンと呼ばれるシンプルなCPUが使われていました．こういったターゲットCPUの環境では，速度優先かつリソース制約ありという条件の元，メモリ保護の機能自体が実装されていないことがほとんどでした．こういった歴史から，高機能なCortex-Aを使ったシステムでも，単純に同じスタイルが引き継がれているようです．

そこで本稿では，Cortex-A CPU上にリアルタイムOSを載せたシステムで，これまでほとんど使われていなかったMMUの機能を活用することのメリット（特にデバッグ時のメリット）を解説します．

● Cortex-Aでリアルタイム OSを使う理由

本稿を執筆する際の議論で，Cortex-Aでリアルタイム OSを使うという事例は少ないのではないか，ましてや64ビットCPU（Armv8-Aアーキテクチャ）は使われないのではないかという意見がありました．

しかし，大規模なデータをリアルタイムに扱いたいデジカメやプリンタと言ったイメージング機器や，Linuxでは実現できないようなリアルタイム性が求められる産業用製造装置といった分野では，64ビットのCPUを含めて引き続きリアルタイムOSが使われることが多いようです．

● Cortex-AでMMUを使う理由

とは言うものの，リアルタイムOSでのMMUの利用は，「扱いが難しそうだから無理してMMUを使わなくてもよいのではないか」とか「MMUなんて使ったら遅くなるのではないか．また，リアルタイム性が阻害されるのではないか」という意見もあると思います．

しかし，Cortex-Aでデータ・キャッシュを有効にするには，MMUを有効にする必要があります．実際にやってみれば分かることなのですが，キャッシュ無効では，有効時と比較し，実行速度が5～10倍遅くなります．そのためキャッシュを有効にしないという選択肢は非現実的で，Cortex-AはMMUを有効にして使うものということになります．

そういった背景から，どうせ有効にして使うのだから，MMUをリアルタイムOSとして使いやすい範囲で，とことん使ってやろうというのが，この記事の背景です．

Cortex-AのMMUの特徴

最初に，Cortex-AのMMUの特徴を簡単に解説します．

● Cortex-AのMMUの機能

Cortex-Aに搭載されているMMUの主な機能は以下の通りです

- 仮想アドレスと物理アドレスのアドレス変換機能
- プロセスのための空間を実現する機能＋実行モード
- アクセス権に応じたメモリ保護

● 仮想アドレスと物理アドレスのアドレス変換機能

MMU有効時にCPUが実行する命令からのアクセスは，常に仮想アドレスになります（仮想アドレスと物理アドレスが一致するような状況であっても，CPUからのアクセスは仮想アドレスが使われている，となる）．

一方で，ハードウェアとしてのメモリやペリフェラル・レジスタに振られているアドレスは物理アドレス