Appendix B

RAMでプログラムを動かす

山際伸一

組み込み機器は，利用者がスイッチをONすると自動的にプログラムを実行し始めるように設計しなければなりません．このため，起動時のソフトウェアはROMに配置します．しかし，起動後は，命令コードをRAMに配置させて動作させる機器もあります．（筆者）

● 命令コードをRAMに配置する理由

パソコンのようにソフトウェアを入れ替えながら使うことがない組み込み機器で，なぜ命令コードをRAMに配置するのでしょうか．

いちばんの理由は，ROMのアクセス時間がRAMよりも長いというケースが多いことです．例えば，RAMなら0ウェイトで動作できるが，ROMでは1ウェイト以上必要になる場合があります．このときは，RAMを使ったほうが性能が向上します．

メモリのアクセス時間に差がなくてもRAMを用いたほうがよい場合もあります．本誌の付属基板に搭載されているADuC7000シリーズの場合，内蔵のROM（フラッシュ・メモリ）のデータ・バス幅は16ビットですが，RAMのデータ・バス幅は32ビットです．ARMプロセッサは，32ビット・アーキテクチャです．32ビット長の命令コードをフェッチするとき，ROMでは2クロック必要ですが，RAMなら1クロックですむということになります．このことから，ソフトウェアがRAMに配置されていたほうが高速に動作することがわかります．

● RAMで命令コードを動作させる手順

命令コードをRAMに配置させるためには，あらかじめROMに用意したコード群をRAMに転送します．この方法を**図1**に示します．

まず，電源投入またはリセット後，決められたアドレスから実行を始めます（**図1**の①）．ここはROM領域です．次にROM領域に書かれた命令コードをRAMに転送します（**図1**の②）．そして，RAM上の命令コードの入り口のアドレスに分岐します（**図1**の③）．

しくみとしては単純ですが，これを実現するソフトウェアを開発するときには，注意が必要です．

まず，ROMで動作するソフトウェアをそのままRAMに転送しても，正しく動作しないということです．ROMとRAMが配置されるアドレスは異なります．分岐命令のジャンプ先やデータの参照先は，RAMで動作するアドレスになっている必要があります．

例えば，ROMアドレス0x00000000から配置されるコードの中に0x00000014に分岐する命令があるとします．これはROM上で正しく動作するコードとします．これをRAMアドレス空間で動作させようと，アドレス0x00010000に転送したらどうなるでしょうか．正しく動作させるためには，この分岐先は0x00010014でなければなりません．0x00000014に分岐してしまったらROMに戻ってしまい，目的を達成できないことになります．

ROMには，RAMで動作することを考慮したコードを書き込んでおく必要があります．つまり，起動時にROMで動作するコードとRAMで実行するコードは，別々に用意する必要があるということです．このため，スタートアップ・ルーチンが2種類必要になります．

● ADuC7000シリーズの独特な方法

使用するマイコンによって独特な方法を用いなければならないこともあります．

ADuC7000シリーズの場合，リセットの直後は0x00000000

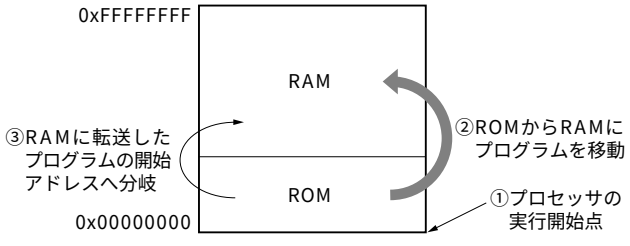

図1　RAMで動作するプログラムの初期の処理

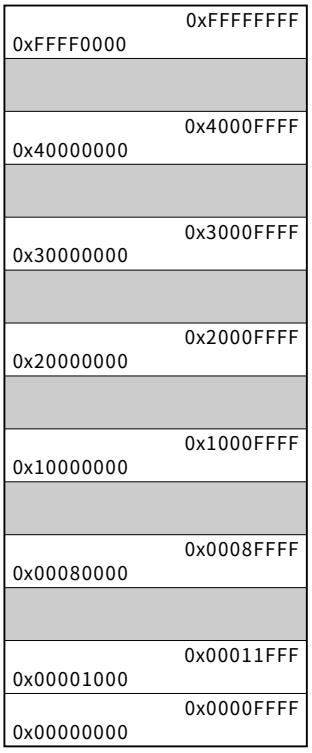

図2　メモリ・マップ

から命令の実行を始めます．また，REMAP レジスタの 0 ビット目に 1 を書くと，0x00000000 が ROM から RAM に変わります．REMAP レジスタへの書き込みは 0x00008000 以降の命令で行う必要がある，という制約があります．

このようなしくみでは，二つの異なるコードを同一アドレスに配置しなくてはならなくなるため，リンク時にエラーになってしまいます．

図2 に ADuC7000 シリーズのメモリ・マップを示します．0x00080000 からが ROM（フラッシュ・メモリ）の領域です．0x00000000 からの領域は，SRAM またはフラッシュ・メモリの再配置空間です．つまり，0x00000000 からの領域と 0x00080000 からの領域は，物理的には同一のメモリになり，同じ内容が見えるということになります．

そこで，以下の手順を実行するコードを記述すればよいことになります．

1）0x00000000 から起動し，すぐに 0x00080000 へ分岐
2）REMAP レジスタに 1 を書く
3）RAM で実行する命令コードを ROM から RAM にコピー
4）0x00000000 に分岐

リンク時には，最初に ROM から実行されるスタートアップ・ルーチンを 0x00080000 から配置します．スタートアップ・ルーチンの中には，0x00080000 から領域に分岐する命令を記述して

▶ リスト1
startup2.s

```
    .text

    .extern     main
    .extern     _sp_base
#=================================================
#       Initialize vectors
#
##      0x00      Reset
##      0x04      Undefined Instruction
##      0x08      Software Interrupt
##      0x0c      Abort(prefetch)
##      0x10      Abort(data)
##      0x14      Reserved
##      0x18      IRQ
##      0x1c      FIQ
#=================================================

#   この _startup はアドレス 0x00080000 を保持する
    B    _startup

    .org(0x20)
_startup:
#   プログラムを ROM から RAM 領域にコピーする
#   再配置する際にはプログラムは 0x00080000-0x0008FFFF で
#   動作していなければならない
    LDR  R1, =0xFFFF0220
    LDR  R2, =1
    STR  R2, [R1]
#   RAM で動作するためのプログラムのコピー
    LDR  R1, = _sstart_ram
    LDR  R2, = 0x0
    LDR  R3, = _end
copy_loop:
    LDR  R4, [R1]
    STR  R4, [R2]
    ADD  R1, R1, #4
    ADD  R2, R2, #4
    CMP  R1, R3
    bne  copy_loop


#   再配置後の 0x00000000 番地，つまり RAM の先頭に分岐
#   （リセット直後の状態を再現する）
    b    _sstart_ram
```

おきます．実際には，0x00000000 から読み出されて実行することになりますが，0x00000000 のアドレス空間から脱出できることになります．

● LED 点滅のプログラムを RAM で実行

ここまで解説した方法を使って，第 6 章の演習 2 で説明した LED 点滅のプログラムを RAM 上で動作させてみます．

ここで使用するコードは，

1）リセット後に ROM で動作するスタートアップ・ルーチン
2）RAM に分岐した後に実行するスタートアップ・ルーチン

です．これは，ROM と RAM にプログラムを配置するためのリンカ・スクリプトです．それぞれ順に見ていきます．

リスト1 に，リセット後，ROM から実行されるコードを示します．アセンブリ言語で書かれています．**図3** に示すアルゴリズムで必要なコードを ROM から読み出し，RAM に転送しています．ここで，_sstart_ram と _end はリンカ・スクリプトでアドレスが設定されます．つまり，_sstart_ram は RAM で実行されるコードの塊の先頭で，_end はその塊の最後の