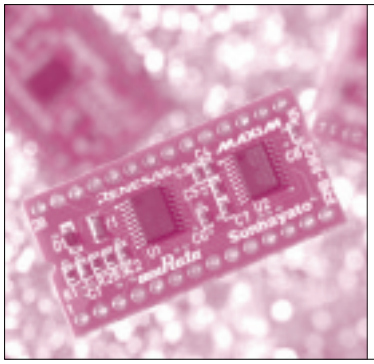

第9章　スタートアップ・ルーチンのしくみとスタック・フレームの役割を理解しよう

C プログラムの裏側を探ってみよう！

山本 秀樹
Hideki Yamamoto

マイコンのプログラム開発では，大半の場合C言語が使われていると思います．C言語はマイコンの違いをある程度隠蔽することができ，プログラマから見ると使いやすい言語です．

さらにアセンブリ言語の知識があれば，C言語で書いたプログラムが実際にはどのように実行されているのかを理解することができ，よりよいプログラムの開発に役立つのではないかと思います．

この章では，C言語で作成したプログラムの裏側を，アセンブリ言語の知識を使って眺めてみます．

スタートアップ・ルーチンの役割

C言語のプログラムでは，main関数内の最初の行を実行する時点ですでに，初期値のない外部変数は値が0になり，初期値のある外部変数はその値で初期化されています．この操作はいつの間に行われたのでしょうか．

アセンブリ言語で書いたプログラムでは，電源を投入してリセット・ベクタに書いたメモリにある命令が呼び出された時点では，RAMにどのような値が書かれているのかを保証できません．しかし，外部変数の値は，RAM上に書かれているはずです．このことから，マイコンがリセットされてからmain関数が呼び出されるまでの間に，C言語のプログラムを実行するために必要な処理が行われたことがわかります．

この処理を行うプログラムは，スタートアップ・ルーチン（またはCランタイム・ルーチン）と呼ばれます．HEWでC言語プログラムを作成した場合，デフォルトではncrt0.a30というファイルが用意され，ここにスタートアップ・ルーチンが書かれています．この内容をざっと見てみましょう．**リスト1**に，自動生成されたスタートアップ・ルーチンの抜粋を示します．

リセットされると，ラベルstartから実行が始まります．まずスタック・ポインタの初期化，マイコンのプロセッサ・モードの設定，可変ベクタ・テーブルの設定を行います．

その後は，マクロ・アセンブラ機能が使われているので，どのような命令が使われているのかわかりにくいのですが，初期値のない外部変数領域を0でクリアし，初期値のある外部変数領域にROM領域から初期値をコピーします．

次に，JSR命令でmain関数を呼び出します．

最後に，通常はmain関数の呼び出しから戻らないようにプログラムを作るのですが，もしmain関数の呼び出しから戻った場合，無限ループに入って暴走を防ぐようになっています．

このスタートアップ・ルーチンで行っているのは，最小限の処理です．これ以外に，例えばクロックを，リセット時に自動的に選択される低速クロックの8分周から，もっと高速のクロックに切り替える処理や，入出力ポートや他の内蔵モジュールの初期化処理を行いたいのではないかと思います．そのような処理をC言語のmain関数の最初で行うこともできますし，このスタートアップ・ルーチン内で行うように書き換えて，C言語のプログラムからはそのような処理を一掃してしまうこともできます．

スタック・フレームの役割

C言語の関数は，通常はサブルーチンとして実現されます．関数呼び出し，引数の受け渡し，関数内の自動変数といった，C言語でプログラムを作成するときには当たり前のように使っている機能について，どのように実現されているのかを見てみます．

HEWで**リスト2**のプログラムをビルドして，付録

Keywords
C言語，スタートアップ・ルーチン，Cランタイム・ルーチン，スタック，フレーム，volatile，ncrt0.a30，スタック・ポインタ，プロセッサ・モード，可変ベクタ・テーブル，マクロ・アセンブラ，JSR命令，ENTER命令

リスト1　スタートアップ・ルーチン ncrt0.a30（抜粋）

```
;------------------------------------------------------
; for R8C/Tiny
;------------------------------------------------------

start:
;------------------------------------------------------
; after reset,this program will start
;------------------------------------------------------
        ldc     #istack_top,    isp     ;set istack pointer
        mov.b   #02h,0ah
        mov.b   #00h,04h                ;set processer mode
        mov.b   #00h,0ah
        ldc     #0080h, flg
        ldc     #stack_top,     sp      ;set stack pointer
        ldc     #data_SE_top,   sb      ;set sb register
        ldintb  #VECTOR_ADR

;======================================================
; NEAR area initialize.
;------------------------------------------------------
; bss zero clear
;------------------------------------------------------
        N_BZERO bss_SE_top,bss_SE
        N_BZERO bss_SO_top,bss_SO
        N_BZERO bss_NE_top,bss_NE
        N_BZERO bss_NO_top,bss_NO

;------------------------------------------------------
; initialize data section
;------------------------------------------------------
        N_BCOPY data_SEI_top,data_SE_top,data_SE
        N_BCOPY data_SOI_top,data_SO_top,data_SO
        N_BCOPY data_NEI_top,data_NE_top,data_NE
        N_BCOPY data_NOI_top,data_NO_top,data_NO

        ldc     #stack_top,sp

;======================================================
; Call main() function
;------------------------------------------------------
        ldc     #0h,fb          ; for debuger

        .glb    _main
        jsr.a   _main           ← main関数の呼び出し

.endif ; __R8C__

;======================================================
; exit() function
;------------------------------------------------------
        .glb    _exit
        .glb    $exit
_exit:                          ; End program
$exit:
        jmp     _exit
```

マイコンにダウンロードしてみました．**図1**に，ダウンロードしたプログラムを示します．この画面は，HEWのエディタ画面のモードを，混合モードに切り替えることで得られます．なお，本特集ではC言語でプログラムを作成する方法について説明していませんが，ご了承ください．

関数fは四つの引数をとります．逆アセンブル結果を見ると，main関数では第4引数からスタックに格納しています．ここで使っているコンパイラでは，第1，第2引数をレジスタR1，R2に格納し，それ以降の引数は最後の引数からスタックに格納するようです．

リスト2　スタック・フレームを確認するプログラム

```
/* トランジスタ技術 2005年4月号 */
/* 第9章 スタックフレーム */

int f(int, int, int, int);

void main(void)
{
        int x;

        x = f(0, 10, 1, 5);
}

int f(int a1, int a2, int a3, int a4) {
        int     buf[10];
        int     i;

        for (i = a1; i < a2; i += a3)
                buf[i] = a4;

        return a1+a2;
}
```

このふるまいは，コンパイラの実装に依存します．

次に，JSR命令によりf関数をサブルーチンとして呼び出しています．

呼び出された関数では，ENTER命令によりスタック上に自動変数領域を作っています．この領域は，配列bufの領域として使われます．

この結果からわかるように，スタックはサブルーチンの戻り番地やレジスタの一時待避だけに使われるのではありません．関数呼び出しでは，引数，戻り番地，レジスタ待避，自動変数領域が，ある決まった構造でスタック上にとられます．このようにスタック上にある決まった構造の領域が取られたものは，スタック・フレームと呼ばれます．

スタック上の自動変数領域や引数の領域には，CPUのFBレジスタを使ってアクセスしています．FBレジスタはスタック・フレームの基準点を指すレジスタとして使われます．

コンパイラの最適化による意図しない動作を抑止するvolatile宣言

C言語で作成したプログラムをコンパイルするとき，通常は最適化が行われます．この最適化はプログラム実行速度を向上したり，プログラム・サイズを小さくするのに役立ちます．しかし，マイコンのプログラムを開発する場合には，最適化によってソース・コードとは異なる実行プログラムになってしまうことがあり，注意が必要になります．

SFRを操作する場合がその典型例でしょう．ここ