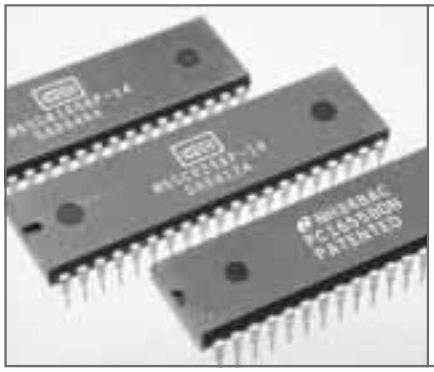
連　載



マイコン・システムのしくみを基礎から理解する

6502マイコン・ボード製作記

〈最終回〉ブート・ローダとモニタ・プログラムの作成

桑野　雅彦
Masahiko Kuwano

最終回は，ブート・ローダとモニタ・プログラムの作成，そして記号化BASICインタプリタの作成例を解説します．

ブート・ローダの作成

ブート・ローダはアセンブリ言語で記述しました．クロスCコンパイラcc65の環境が構築できたのでC言語で記述したいところですが，Cで書いた場合，簡単な演算でもライブラリがあれこれと付いてきてしまいます．

プログラムをダウンロードして動かすならば，多少大きくなってもかまいませんが，今回はスイッチを使って手作業で書き込むわけですからたいへんです．そこで，cc65のアセンブラでブート・ローダを作成する方針を立てました．

■ cc65のオブジェクト・フォーマットの確認

最初に，cc65が出力したオブジェクトのフォーマットを調べてみましょう．まず，第11回(2007年3月号)で実行したテスト・プログラムをアセンブリ言語で書いてみます(**リスト12－1**)．ファイル名はchkprog.sとしました．

コンフィグレーション・ファイルはchkprog.cfgです(**リスト12－2**)．

リスト12－1　第11回(2007年3月号)で実行したテスト・プログラムをアセンブリ言語で記述

```
;-----------------------------------------
;-- PEACH-Iテストプログラム                --
;--                                       --
;-- スイッチ（ジャンパ）の状態を読んで     --
;-- 出力端子に設定                        --
;--                                       --
;-- LED1: OUT1                            --
;-- LED2: OUT2                            --
;-- SW1 : RI                              --
;-- SW2 : DCD                             --
;-----------------------------------------
.segment        "ZEROPAGE"      ;.ゼロページ領域
msrdat:         .byte   0
outdat:         .byte   0

.segment        "CODE"          ;.コード領域の開始
                                ;.cfgファイルで、ROM領域（$FF00～）
                                  に割り付け
;
; PC16550の内部レジスタ
;
        SIO_MSR = $8006         ; モデムステータスレジスタ
        SIO_MCR = $8004         ; モデムコントロールレジスタ

        MSR_RI = $40            ; MSR[6]がRIビット （SW1）
        MSR_DCD = $80           ; MSR[7]がDCDビット（SW2）

        MCR_OUT1 = $04          ; MCR[2]がOUT1ビット(LED1)
        MCR_OUT2 = $08          ; MCR[3]がOUT2ビット(LED2)


startup:                        ; // while(1){ SIO_MCR =
                                     (SIO_MSR & 0xc0) >> 4; }
        lda     SIO_MSR         ; Areg=SIO_MSR;
        and     #$C0            ; Areg=Areg & 0xc0;
        lsr     a               ; Areg >>=1;
        lsr     a               ; Areg >>=1;
        lsr     a               ; Areg >>=1;
        lsr     a               ; Areg >>=1;
        sta     SIO_MCR         ; SIO_MCR = Areg
        jmp     startup         ; goto startup; }

.segment        "VECTORS"       ; 割り込みベクタ領域
                                ; .cfgファイルで$FFFA～に割り付け
vector_nmi:
        .addr   startup
vector_reset:
        .addr   startup
vector_irqb:
        .addr   startup
```

Keywords
ブート・ローダ，モニタ，BASICインタプリタ，ゼロ・ページ，ベクタ，6502，XMODEM

リスト12−2　リスト12−1のコンフィグレーション・ファイル

```
MEMORY {
        ZP: start = $0, size = $100, type = rw define = yes;
        ROM: start = $FFE0 ,size = $0100, file = "ChkProg.OUT";
}
SEGMENTS {
        CODE:           load = ROM, type = ro;
        ZEROPAGE:       load = ZP, type = zp, define = yes;
        VECTORS:        load = ROM, type = ro, start = $FFFA;
}
```

リスト12−3　cc65のアセンブラで生成されたChkProg.OUTのダンプ

```
-d
2DFF:0100  AD 06 80 29 C0 4A 4A 4A-4A 8D 04 80 4C E0 FF 00   ...) .JJJJ...L...
2DFF:0110  00 00 00 00 00 00 00 00-00 00 E0 FF E0 FF E0 FF   ................
```

プログラムは$FFE0からで，ベクタ領域は$FFFAからの領域に配置します．これでコマンド・ラインから次のように入力します．

```
cl65 --listing --cpu 65C02 --mapfile ChkProg.map -C chkprog.cfg -t none chkprog.s
```

マップ・ファイルやリスト・ファイルが不要ならば，--listingや--mapfile ChkProg.mapの部分は不要です．

これで生成されたChkProg.OUT(MEMORYセクションで指定したファイル名)をコマンド・プロンプトのDEBUGで読み込んでダンプしたのが**リスト12−3**です．第11回でハンド・アセンブルで作ったものとまったく同じ，単にベタなバイナリ・ファイルであることがわかります．したがって，これを読み込んでRAM上に展開して，指定アドレスから実行するようにすればよいわけです．

■ ダウンロードにXMODEMプロトコルを採用

次に考えなくてはならないのは，上記のベタなバイナリ・ファイルの受け渡し方法です．

Windowsに標準添付のハイパーターミナルには，XMODEM，YMODEM，ZMODEM，Kermitなどのプロトコルがサポートされています．本稿では，このハイパーターミナルでもっとも簡単なXMODEMを使用することにしました．

至近距離での直結伝送であり，エラーの発生はほとんどないという前提で，単にバイナリ・データを受け取るだけなら比較的簡単に実装できます．

XMODEMでは，エラー・チェックがチェックサムであることや，データと制御コードの区別がつきにくいなどということもあり，まじめにエラー処理を行おうとするといろいろと面倒なことになるのですが，今回はエラーは起きないという前提で，非常に簡略化した実装を行いました．

ブロック番号チェックやチェックサムのチェックは省略し，常にACK応答するようにしています．また，ホストから送られてきたデータの先頭バイトが$01でなければEOT(End of Transmission)とみなしてACKを返して終了するというやり方にしました．これでプログラムもだいぶ小さくすることができます．

■ 実行開始アドレスとベクタ情報の設定方法

● ダウンロード・プログラムの実行開始アドレスとベクタ情報

プログラム本体をダウンロードできる目処が立ちました．cc65が出力したベタのイメージのバイナリ・ファイルを，Windows標準のハイパーターミナルでXMODEMを使って送るだけなので扱いは簡単です．

さて，ここで問題になるのが，実行開始アドレスとベクタ情報をどうするかということです．実行開始アドレスを別途与えるのでは面倒なので，ダウンロードするファイルに埋め込んでおくほうがよいでしょう．

また，ベクタ情報も問題です．シリアル・ポート経由でモニタ・プログラムをダウンロードしているとき，実行しているプログラム(ブート・ローダ)は$FF00〜$FFFFの領域にあります．当然，ベクタ情報もこのブート・ローダのものになっているので，これをモニタ・プログラムのものと差し替えなくてはなりません．

ところが，XMODEMではダウンロード先のアドレス指定はありません．$FFFA〜$FFFFのベクタ領域の情報をもたせようとすると，現在実行中のプログラムを上書きしていかないとベクタ領域まで届きません．こんなことをすればいきなり暴走してしまいます．

● ヘッダ情報を付加したダウンロード・ファイルを使用する

そこで，ダウンロード・ファイルの先頭に16バイトのヘッダ情報を付加しました．**図12−1**に，ブー