「CP/M on an AVR(avrcpm)」で公開されているディスクイメージはちょっと問題があり、動作テスト程度にしか使えないので作り直した方がよい。さらにディスクサイズが 8 インチ標準 1 ドライブでは狭すぎるので拡張する。

完成したソースをアーカイブに添付するほうが簡単だが、CP/M に関する情報は少なくなってきており、応用が利くようにあえて作業の流れを記載しておく。アスキー出版の「応用 CP/M」(1982 年) から得られた情報を元にしている。(作業環境 OS は Windows)

準備するもの

■ cpmtools

CP/M ファイルシステムのディスクイメージを作成し、CP/M ファイルの登録、取り出しができる。

http://www.cpm8680.com/cpmtools/

【2013.1.1】 GUI 化したものを作成しました。 CpmtoolsGUI こちらも使ってみてください。

■ CP/M のシステムバイナリ

CPM.SYS は下記サイトから入手できる。
http://www.cpm.z80.de/binary.html
http://spritesmods.com/?art=avrcpm&page=4
http://www.mikrocontroller.net/articles/AVR_CP/M

■システムソースファイル

ipl.asm, bios.asm を下記から入手する。BIOS にシステムのリロードが実装されておりこちらをベースにする。
http://petersieg.bplaced.net/?AVR_CP%2FM の avrcpm_upd.zip

■ Z80 アセンブラ

Win32 用 Z80 アセンブラ (tniasm) は下記から入手。(ソースの終わりに改行を入れておく)
http://www.tni.nl/products/tniasm.html

■連結ツール

システムバイナリを dd コマンドで連結する。dd.exe は下記から入手。(cpmtools に既にある dll は上書きしない)

「コンピュータ広場 CS」→リンク切れのためコピーを配置→ dd.exe

■ディスクイメージを SD カードへ登録するツール

DDforWindows を使う。下記から入手。
http://blog.halpas.com/archives/1258

ディスクの設計

CP/M は階層ディレクトリを持たないので 1 ドライブで容量を拡張すると扱いにくくなる。パーティションを分け以下の構成とする。

[128 バイト / セクタ] × [64 セクタ / トラック] × [244 トラック] × 4 ドライブ

1 ドライブ当り約 2 メガバイト。計 8 メガバイトとなる。これならば実用上問題ないだろう。

■ディスクパラメータの算出

BIOS に設定するディスクパラメータを計算する。
パーティションで分ける場合、トラックは積算となるので各ドライブの諸元はそれぞれ

ドライブ	バイト	セクタ	トラック
A:	128	64	244
B:	128	64	488
C:	128	64	732
D:	128	64	976

CP/M ブロックサイズ (BLS) は、DSM が 256 以上になるので必然的に 2048 バイト以上となるが AVR のメモリが 4KB なので 2048 バイトしか選択の余地は無い。システムトラックはセクタを 64 としたので 1 トラックとしている。

BLS=2048

```
SPT:64
BSH:4 (2048=128*(2^4))
BLM:15 (0 〜 15: 2048/128-1=15)
EXM:0 (2048:DSM>255)
DSM A:{(128*64)*(244-1)/2048}   -1=971
DSM B:{(128*64)*(488-244)/2048} -1=975
DSM C:{(128*64)*(732-488)/2048} -1=975
DSM D:{(128*64)*(976-732)/2048} -1=975
DRM:127 (2bit*64-1)
AL0:192 (2bit: 0b11000000=192)
AL1:0 (0b00000000)
CKS:(DRM+1)/4=(127+1)/4=32
OFF A:1
OFF B:244
OFF C:488
OFF D:732
CSV A:32
CSV B:32
CSV C:32
CSV D:32
ALV A:(971+1)/8=122
ALV B:(975+1)/8=122
ALV C:(975+1)/8=122
ALV D:(975+1)/8=122
```

■パラメータ算出方法解説

- SPT(Sector Per Track)
  - 1 トラック当りのセクタ数
- BSH(Block SHift factor)
  - 1 ブロックのバイト数 (BLS) を 128 バイト × 2 の乗数で表す。BLS=128*2^BSH
- BLM(BLock Mask)
  - 1 ブロック内のセクタ数を 0 〜 で表す。BLS/128 - 1
- EXM(EXtent Mask)
  - 計算式は不明だが表が示されている。

BLS	DSM<256	DSM>255

1024	0	-
2048	1	0
4096	3	1
8192	7	3
16384	15	7

推測で、EXM=(BLS/1024/ ブロックポインタのサイズ)-1
ブロックポインタは、DSM<256 のとき 1 バイト、DSM>255 のとき 2 バイトになる
1 エクステント当り、ブロックポインタ用に 16 バイトの割り当てゆえに、
1 エクステント当りに指示できるバイト数 =BLS*16/ ブロックポインタのサイズ =1024 *(EXM+1)*16
STAT DSK: コマンドで表示される Records/Extent =(EXM+1)*128。(1Record=128 バイト)

・DSM(Disk Size Max)
　・ブロックの最大数 - 1。
・DRM(DiRectory entries Max)
　・ディレクトリエントリーの数 - 1。(AL0,AL1 のビット 1 の数)*BLS/32-1
・AL0,AL1
　・この 2 バイトで 1 ビットを 1 ブロックとみなしたビットパターンでディレクトリエントリ (=1) を表す

AL0	AL1
11000000	00000000

これから分かることは、
ディレクトリーエントリとして確保できる領域は最大 16 ブロックまで。
AL0、AL1 が意味するものは何か？これも未確認で推測だが、
CP/M システムはディスクから一律に 16 ブロックのディレクトリーエントリ領域を確保する？
AL0、AL1 はマップであり、確保した領域のうちビット 1 の部分を使用する？
空きは、ディレクトリーエントリが物理損傷した場合のリカバリー用ではないか。
→確認してみた。ビットの位置を変更してみたが変化無し。
ディレクトリーエントリ領域はビット分しか確保していなかった。
構想のみで実装しなかったのか？

・CKS(ChecK vector Size)
　・(DRM+1)/4。ディレクトリチェック用ワークエリアサイズ。
・OFF(track OFFset)
　・システムトラック数。パーティションに分けるときにも利用できる。
・CSV(Check Scratchpad Vector)
　・=CKS
・ALV(ALlocation scratchpad Vector)
　・(DSM+1)/8。1 ビット 1 ブロックとみなすマップを作り使用済 / 未使用エリアを管理するためのワークエリア。

なおこれらのパラメータは CP/M が動作する環境があれば DISKDEF マクロによって自動的に算出することもできる。→ DISKDEF マクロの使い方メモ

ソースの修正

■ ipl.asm の修正

予め 43 行を

```
ld a,1  ;execute DMA READ
out (22),a
```

とし、21 〜 24 行を削除しておく。

26 行あたり。読み込むセクタ数。修正によって少し増えるので多めにとっておく。

```
ld b,55    ;
```

49 行あたり。1 トラック当りのセクタ数を 26 から 64 へ変更

```
cp 64    ;
```

■ bios.asm の修正

予め、76,77 行を削除。31 〜 34 行を削除。

32 行あたり。システムのセクタ数。44。もとのコード ($-ccp) の意味がわからないので修正。

67 行あたり。追加。理由は後述。

```
ld  (cdisk),a
ld  (odisk),a  ;add neko Java
jp  gocpm
```

101 行あたり。ipl 同様。1 トラック当りのセクタ数を 26 から 64 へ変更

```
cp  64  ;if sector >= 64 then change tracks.
```

108 行あたり。トラック数が 256 以上 (トラックのポインタが 2 バイト) になるので b レジスタに正しい値を入れる。

```
push  hl
ld  b,0  ;add neko Java.
call  settrk
```

home:。トラックが 2 バイト指定になったことによる。

```
home:
  push af
  ld a,0
  out (16),a
  out (17),a  ;add neko Java.
  pop af
  ret
```

seldsk:。4 ドライブ化。前のカレントドライブを odisk に保持し、不正なドライブを選択したときは前のドライブに戻す。これをやっておかないとエラーメッセージ表示後、リブートの無限ループになる。

```
seldsk:      ;mod for 4drive. neko Java
  push af
  ld a,c
  cp 0
  jr nz,seldsk_1
  ld hl,dph0
  jr seldsk_ok
seldsk_1:
  cp 1
  jr nz,seldsk_2
  ld hl,dph1
  jr seldsk_ok
seldsk_2:
  cp 2
  jr nz,seldsk_3
  ld hl,dph2
  jr seldsk_ok
seldsk_3:
  cp 3
  jr nz,seldsk_na
  ld hl,dph3
seldsk_ok:
  ld (odisk),a
  pop af
  ret
seldsk_na:
  ld hl,0
  ld a,(odisk)
  ld (cdisk),a
  pop af
  ret
```

settrk:。トラックが 2 バイト指定になったことによる。

```
settrk:
  push af
  ld a,c
  out (16),a
  ld a,b    ;add neko Java
  out (17),a  ;add neko Java
  pop af
  ret
```

read:、write:。AVR がディスクエラーを返せるので取り込むようにする。

```
read:
  ld a,1
  out (22),a
  in a,(23)  ;mod neko Java
  ret

write:
  ld a,2
  out (22),a
  in a,(23)  ;mod neko Java
  ret
```

ディスクパラメータヘッダ。下記を参考に、以下 dph1,dph2,dph3 を定義する。

```
;Disk Parameter Header
dph0:
  dw trans ;XLT: Address of translation table
  dw 0 ;000: Scratchpad
  dw 0 ;000: Scratchpad
  dw 0 ;000: Scratchpad
  dw dirbuf ;DIRBUF: Address of a dirbuff scratchpad
  dw dpb0 ;DPB: Address of a disk parameter block
  dw chk0 ;CSV: Address of scratchpad area for changed disks
  dw all0 ;ALV: Address of an allocation info scratchpad
```

ディスクパラメータブロック。下記を参考に、以下 dpb1,dpb2,dpb3 を定義する。dpb1 以降の DSM=975、OFF(積算) に注意。

```
dpb0:
  dw 64 ;SPT: sectors per track
  db 4 ;BSH: data allocation block shift factor
  db 15 ;BLM: Data Allocation Mask
  db 0 ;Extent mask
  dw 971 ;DSM: Disk storage capacity
  dw 127 ;DRM, no of directory entries
  db 192 ;AL0
  db 0 ;AL1
  dw 32 ;CKS, size of dir check vector
  dw 1 ;OFF, no of reserved tracks
```

スキューはかけないのでここの記述は意味なし。適当。

```
trans:
  db 0
```

後は dirbuf:(=128)、chk0: 〜 chk3:(=CSV)、all0: 〜 all3:(=ALV) のバイト数を算出した値に設定する。

```
dirbuf:
  ds 128
chk0:
  ds 32
.
.
all0:
  ds 122
.
.
```

最後に以下を追加。前のカレントドライブ保持用。

```
odisk:
  ds 1  ;add neko Java
```

■ファームの修正

z80cpm.h の修正。最後の方。SECT_CNT と BLOCK_SIZE を修正。

```
/* Disk parameters ------------------------------------- */
#define SDC_CLST_SIZE  512  // fixed for SDC.
#define SECT_SIZE    128  // fixed for CP/M.
#define SECT_CNT    64  // CP/M sector size.
#define BLOCK_SIZE    2048  // CP/M block size.
```

z80cpm.c の修正。166 行。var_calc() 内。トラックのハイバイトが 0 に固定してあるのでコメント化する。

```
  //Track_no_h = 0;
  Track_no = ((WORD)Track_no_h << 8) + (WORD)Track_no_l;
```

アーカイブに添付の Makefile でビルドする。

■ diskdefs ファイルの修正

cpmtools の diskdefs ファイルを定義する。同様に cpm_b,cpm_c,cpm_d を定義。それぞれ tracks と

boottrk は異なる (積算になる) ので注意。

```
# CP/M ドライブ A アクセス用 --
diskdef cpm_a
  seclen 128
  tracks 244
  sectrk 64
  blocksize 2048
  maxdir 128
  skew 1
  boottrk 1
  os p2dos
end
```

ディスクイメージを作成する。

以下作業ディレクトリを下記と仮定する

```
[ ツール    ] D:\cpmtools
[ ソース    ] D:\cpmtools\z80
[A: ファイル ] D:\cpmtools\z80\a
[B: ファイル ] D:\cpmtools\z80\b
[C: ファイル ] D:\cpmtools\z80\c
[D: ファイル ] D:\cpmtools\z80\d
```

■環境設定、作業準備

コマンドプロンプトから、

```
set CPMTOOLS=
set CPMTOOLS=D:\cpmtools
set PATH=%CPMTOOLS%;%PATH%
d:
cd D:\cpmtools\z80
```

■アセンブル

```
tniasm ipl.asm ipl.bin
tniasm bios.asm bios.bin
```

■連結

```
dd conv=sync bs=128  count=1 if=ipl.bin > cpm.bin
dd conv=sync bs=128 count=44 if=CPM.SYS >> cpm.bin
dd conv=sync bs=128  count=10 if=bios.bin >> cpm.bin
```

■ディスクイメージ作成

```
mkfs.cpm -f cpm_d diskimage
mkfs.cpm -f cpm_c diskimage
mkfs.cpm -f cpm_b diskimage
mkfs.cpm -f cpm_a -b cpm.bin -L test diskimage
```

■ファイルコピー (a-d フォルダのファイルをイメージへ)

```
cpmcp -f cpm_a diskimage ./a/*.* 0:
cpmls -f cpm_a diskimage
cpmcp -f cpm_b diskimage ./b/*.* 0:
cpmls -f cpm_b diskimage
cpmcp -f cpm_c diskimage ./c/*.* 0:
cpmls -f cpm_c diskimage
cpmcp -f cpm_d diskimage ./d/*.* 0:
cpmls -f cpm_d diskimage
```

■イメージを SD カードに書き込む

DDforWindows を使う。読出す場合は SDC の容量一杯までやってしまうが途中で ESC キーで止めることができる。

■イメージからファイルを取り出す場合の例

```
cpmcp -f cpm_a diskimage 0:*.* コピー先ディレクトリ
```

AVR と Z80 で CP ／ M へ戻る

CpmtoolsGUI へ

関連サイト

CP/M on an AVR

http://spritesmods.com/?art=avrcpm&page=1

改善版

http://petersieg.bplaced.net/?AVR_CP%2FM ← BIOS のベースにしたのはこちらのサイト。

http://www.mikrocontroller.net/articles/AVR_CP/M

- ($-ccp) の $ には現在アドレス位置が入りますから (bios-ccp) と読み替えます -CCP) - samllaCircuit (2011 年 09 月 09 日 23 時 32 分 07 秒)
- ご指摘ありがとうございます。アセンブラは詳しくないので。修正不要ということで削除しておきます。 - 管理人 (2011 年 09 月 10 日 03 時 12 分 33 秒)