シリーズ ソースコードを読む

芳之内 弘

LILOブートアップ・ソースコードの解析③

先月号で取りあげたSecond.Sの続きです。今月号では、グリーティングメッセージを表示するルーチン以降を解説していきます。

LILOのソースコード解析の第3回目です。今回は、先月号に引き続いてsecond.Sのソースコードを解読していきます。

なお、マップファイルの内のカーネルのMAPテーブルでは、表1の順にセクタアドレスが並んでいます。second.Sは、これを元にデータを読み込み処理を実行します。

表1 マップファイルの内のカーネルのMAPテーブル

順番	内容
(1)	フォールバックセクタのアドレス
(2)	オプショナルセクタのアドレス
(3)	カーネルオリジナルブートセクタのセクタアドレス
(4)	セットアップのセクタアドレス(複数個)
(5)	カーネルの最初のセクタアドレス
(6)	テーブルの次ページのセクタアドレス
(7)	最後のセクタアドレス
(8)	00 00 00 00 00 00 00 00 00 00 00 00(これはEOFを意味する)

リスト1 ※フローチャート1、2参照

```
0313                                        !グリーティングメッセージを表示するルーチン
0314  iloop:  seg     ss                    ! セグメント=SS=0x9000
0315           cmp     MSG_OFF+SSDIFF,#0     ! グリーティングメッセージの長さが0かどうか？
316            je      nomsg                 ! 0ならnomsg(0340)へジャンプ
0317           mov     bx,#crlf              ! BX=“改行”
0318           call    say                   ! 改行する
0319           seg     ss                    ! SEGMENT=SS=0x9000
0320           mov     cx,MSG_OFF+SSDIFF+2   ! CX=グリーティングメッセージファイルを読むためのマップファイルのCX値
0321           seg     ss
0322           mov     dx,MSG_OFF+SSDIFF+4   ! グリーティングメッセージのマップファイルのDX値
0323           mov     bx,#MAP               ! マップテーブルのバッファアドレス
0324           mov     al,#1                 ! 読み込むセクタ数=1
0325           call    sread                 ! マップファイルを1セクタ読み込む
0326           call    loadfile              ! マップファイルのセクタアドレス情報を使ってメッセージファイルを読み込む
0327           seg     ss                    ! SEGMENT=SS=0x9000
0328           mov     ax,#SYSSEG            ! AX=#SYSSEG=0x1000
0329           mov     ds,ax                 ! DS=0x1000
0330           xor     bx,bx                 ! BX=0 これはターミネーターになる
0331
0332           seg     ss                    ! SEGMENT=SS=0x9000
0333           xchg    bx,MSG_OFF+SSDIFF     ! BX=グリーティングメッセージ長のアドレス
                                             ! ディスクリプタテーブルのグリーティングメッセージ長=0
0334           mov     byte ptr (bx),#0      ! [DS：MSG_OFF+SSDIFF]=[0x1000:0xA028]=0
0335           xor     bx,bx                 ! BX=0ターミネタ
0336           call    say                   ! グリーティングメッセージを表示する
0337           mov     ax,cs                 ! DS,ESを元に戻す
0338           mov     ds,ax                 ! DS=CS=0xB000
0339           mov     es,ax                 ! ES=CS=0xB000
0340  nomsg:   mov   cmdbeg,#acmdbeg        ! cmdbeg=#acmdbeg(メッセージ“auto”のアドレス」)
0341           mov     si,#usrinpm           ! SI=#usrinpm(対話モード)
0342  niloop:  mov     bx,#msg_p             ! BX=“boot:”のアドレス
0343           call    say                   ! “boot:”を表示する
```

リスト1 （続き）

```
0344            mov     bx,#cmdline                     ! BX=コマンドラインの最初のアドレス
0345    clend:  mov     al,(bx)                         ! ALにコマンドの値を代入
0346            or      al,al                           ! AL=0かどうか？
0347            jz      cledne                          ! 0なら最後なのでcledne(0353)へ
0348            push    bx                              ! BX=cmdlineのポインタを待避させる
0349            call    display                         ! ALの文字を表示する
0350            pop     bx                              ! BX=cmdlineのポインタを戻す
0351            inc     bx                              ! BX=BX+1インクリメントして次の文字を指す
0352            jne     clend                           ! なぜjne？ 「jmp clend」では？
0353    cledne: mov     byte ptr prechr,#32             ! コマンドラインの直前にスペースをセット
                                                        ! precharはコマンドライン(command)の直前に配置
0354
0355    ! インプットループ
0356    ! キー入力処理ルーチン
0357    input:  seg     es                              ! SI=#userinpm(0341)またはSI=#DFCMDL+2
```

リスト2 ※フローチャート2参照

```
0357    input:  seg     es                              ! SI=#userinpm(0341)またはSI=#DFCMDL+2
```

（誌面の都合により387行目まで省略します。フローチャートを参照してください）

リスト3 ※フローチャート3、4参照

```
0388    noblnk: cmp     bx,#cmdline+CL_LENGTH           ! BX=cmdlineの最後のアドレスかどうか？(CL_LENGTH=78)
0389            je      input                           ! Yesのとき入力を無視してinput(0357)へ
0390            mov     (bx),al                         ! cmdlineコマンドバッファにALをストアする
0391            inc     bx                              ! BX=BX+1(cmdlineポインタをインクリメントする)
0392            push    bx                              ! BXを退避する
0393            call    display                         ! ALの文字を表示する
0394            pop     bx                              ! BXを回復する(cmdlineポインタ)
0395            cmp     bx,#cmdline+1                   ! 今回の入力は最初か？
0396            jne     input                           ! Noならinput(0357)へ
0397    #ifdef LCF_IGNORECASE
0398            call    upcase                          ! 最初なら大文字へ変換する
0399    #endif

シングルキーモードのチェック
0400            mov     cx,#IMAGES                      ! CX=#IMAGES(起動イメージの最大数)=19
                                                        ! ループの繰り返し回数=19回
0401            mov     di,#DESCR+2                     ! DI=ディスクリプタテーブル の3バイト目のアドレス=0x2202
0402            mov     ah,al                           ! AH=ALL(入力文字)
0403    sklp:   test    word ptr(di+FLAGS_OFF),#FLAG_SINGLE
                                                        ! Single Key Flagのビットテストを行う(#FLAG_SINGL=0x20なのでbit5のテスト)
                                                        ! DI=0x2202、FLAGS_OFF=0x30(Key Single Flag=[0x9000：0x2232])
                                                        ! lilo.confにシングルキーの記述があると、この値はbit5=1となる
                                                        ! すなわち、ディスクリプタテーブルの33バイト目の文字がbit5=1だとSingle Keyモード
0404            jz      sknext                          ! bit5=0だとsknext(0413)へ

シングルキーモード
0405            mov     al,(di)                         ! AL=ディスクリプタテーブルの3番目の文字(0401でセット)
0406    #ifdef LCF_IGNORECASE
0407            call    upcase                          ! 大文字に変換する
0408    #endif
0409            cmp     al,ah                           ! AH(キー入力文字)=AL？
0410            jne     sknext                          ! Noならsknext(0413)へ
0411            cmp     byte ptr (di+1),#0              ! テーブルの次の文字は0かどうか？
0412            je      cr                              ! YesならCR(0431)へ(シングルキーではCRは不要)
                                                        ! 起動イメージの選択は終了して、起動処理へ進む

マルチキーモード
0413    sknext: add     di,#DESCR_SIZE                  ! #DESCR_SIZE=0x34
                                                        ! ディスクリプタテーブルの1イメージの長さは0x34Bytes
                                                        ! DI=次のイメージのオフセットアドレス
0414            loop    sklp                            ! sklp(0403)から繰り返す
```

リスト3 （続き）

```
0415            jmp     input                   ! キー入力を繰り返す(0357)
0416
0417    todelch:br      delch                   ! delch(0534)1文字削除へ
0418    todell: br      delline                 ! delline(0545)1行削除へ

コマンドの実行／割り込みキーのチェック
0422    nul:    push    bx                      ! BXを退避(BXはcmdlineポインタ)
0423            mov     ax,old_del              ! AX=old_del
0424            call    waitsh                  ! 次のキーかあるいはタイムアウトまで待つ(1260)
                                                ! [Shift]か[Control]か[Alt]が押されるか、あるいは[CapsLock]か[ScrllLock]がONのとき
                                                ! CY=1にセットされる
0425            pop     bx                      ! BXを元に戻す(cmdlineポインタ)
0426            jnc     crnul                   ! タイムアウトではCY=0なのでcrnul(0432)へジャンプ
0427            mov     bx,#msg_int             ! キー入力なので" *Interrupted* "を表示
0428            call    say
0429            br      iloop                   ! iloop(0314)対話モードへ戻る

ブートイメージ確定、起動開始
0431    cr:     mov     cmdbeg,#mcmdbeg         ! cmdbeg=#mcmdbegメッセージ" BOOT_IMAGE "のアドレス
0432    crnul:  mov     byte ptr break,#0       ! ブレークフラグ=0
0433            mov     ax,cs                   ! AX=CS=0x9B00
0434            mov     es,ax                   ! ES=CS =0x9B00
0435            xor     al,al                   ! AL=0
0436            mov     (bx),al                 ! BXがポイントするcmdline=0にセット(ターミネータ)

comdlineからlkcbuffへコピー
0437            mov     si,#cmdline             ! SI=#cmdline(コマンドラインのアドレス)
0438            mov     di,#lkcbuf              ! DI=#lkcbuf(バッファのアドレス)
0439            mov     byte ptr dolock,#0      ! dolock=0(ロックを無効にする)
0440            push    es                      ! ES=0x9B00を退避
0441            mov     ax,ds                   ! AX=DS=0x9B00
0442            mov     es,ax                   ! ES=DS=0x9B00
0443    cpsav:  lodsb                           ! AL=[DS:SI]のデータ(SI=SI+1)
0444            stosb                           ! ALの内容を[ES:DI]へ書き込む(DI=DI+1)
0445            or      al,al                   ! ALが0ならZF=1になる
0446            jnz     cpsav                   ! ALが0でないときはcpsav(0443)へ繰り返す
0447            pop     es                      ! ESを戻す
0448            cmp     bx,#cmdline             ! BXはコマンドのポインタなので、
                                                ! BX=#cmdlineはコマンドラインが空を意味する
0449            je      notrspc                 ! Yesならコマンドがないので最初のイメージを起動
                                                ! notrspc(0454)へ
0450            cmp     byte ptr (bx-1),#32     ! コマンドがスペースかどうか？
0451            jne     notrspc                 ! Noならnotrspc(0454)へ
0452            dec     bx                      ! BX=BX-1でコマンドポインタを前へずらす
0453            mov     byte ptr (bx),al        ! コマンドラインのスペースの位置にAL=0を代入

コマンドラインの中に" vga= "、" kbd "、" lock= "、" mem= "をスキャンする
0454    notrspc:mov     si,#cmdline             ! SI=コマンドラインのアドレス
0455            mov     di,si                   ! DI=SI=#cmdline(コマンドラインのアドレス)
0456            cld                             ! 繰り返しでDI、SIは増加するようにセット
0457    chkvga: cmp     word ptr (si),#0x6776   ! " vga "かどうか" vga= "をチェック
0458            jne     vsktk                   ! Noでvsktk(0464)へ
0459            cmp     word ptr (si+2),#0x3d61 ! 次のワードが" a= "か？
0460            jne     vsktk                   ! 違うときはvsktk(0464)へ
0461            call    setvga                  ! VGA モードを設定する(1584)
0462            jc      toiloop                 ! CY=1はエラーなのでtiloop(0495)を経てiloop(0314)へ
0463            jmp     vskdb                   ! vskdb(0476)へ(最後のブランクを捨てる)
0464    vsktk:  cmp      word ptr (si),#0x626b  ! " kb "かどうか？ " kbd= " をチェック
0465            jne     vsktl                   ! No.ならvsktl(0464)へ
0466            cmp     word ptr (si+2),#0x3d64 ! 次のワードは" d= "かどうか？
0467            jne     vsktl                   ! No.でvsktl(0470)へ
0468            call    putkbd                  ! putkbd(1509)
0469            jmp     vskdb                   ! vskdb(0476)へ最後のブランクを捨てる
0470    vsktl:  cmp     word ptr (si),#0x6f6c   ! " lo "かどうか？ " lock "をチェック
0471            jne     vsktm                   ! Noでvsktm(0478)へ　次のブランクへ
0472            cmp     word ptr (si+2),#0x6b63 ! 次のワードは" ck "か？
0473            jne     vsktm                   ! Noでvsktm(0487)へ　次のブランクへ
0474            mov     byte ptr dolock,#1      ! ロックを有効にする
0475            add     si,#4                   ! SI=SI+4　SIは次のコマンドを指す
0476    vskdb:  dec     di                      ! DI=DI-1　最後のブランクを捨てる
0477            jmp     vsknb                   ! vslmb(0483)へ
0478    vsktm:  cmp     word ptr (si),#0x656d   ! " me "? " mem= "をチェックするか？
```

リスト3 （続き）

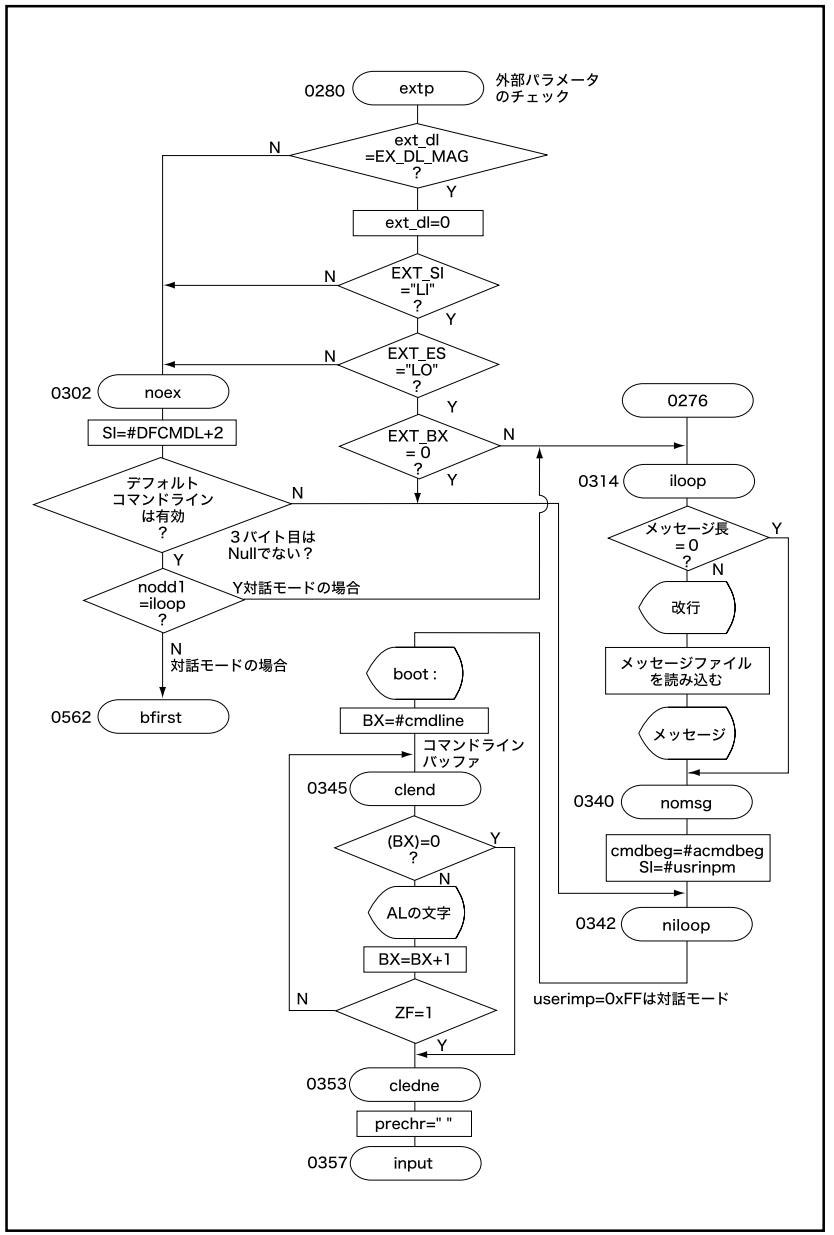
```
0479            jne     vsknb                   ! Noでvsknb(0483)へ　次のブランクへ
0480            cmp     word ptr (si+2),#0x3d6d ! 次のワードは‘mo’か？
0481            jne     vsknb                   ! Noでvsknb(0483)へ　次のブランクへ
0482            call    getmem                  ! ユーザーメモリのリミットを取得
0483   vsknb:   lodsb                           ! AL=[DS:SI]コマンドラインのデーターをロード(SI=SI+1)
```

リスト4 ※フローチャート5参照

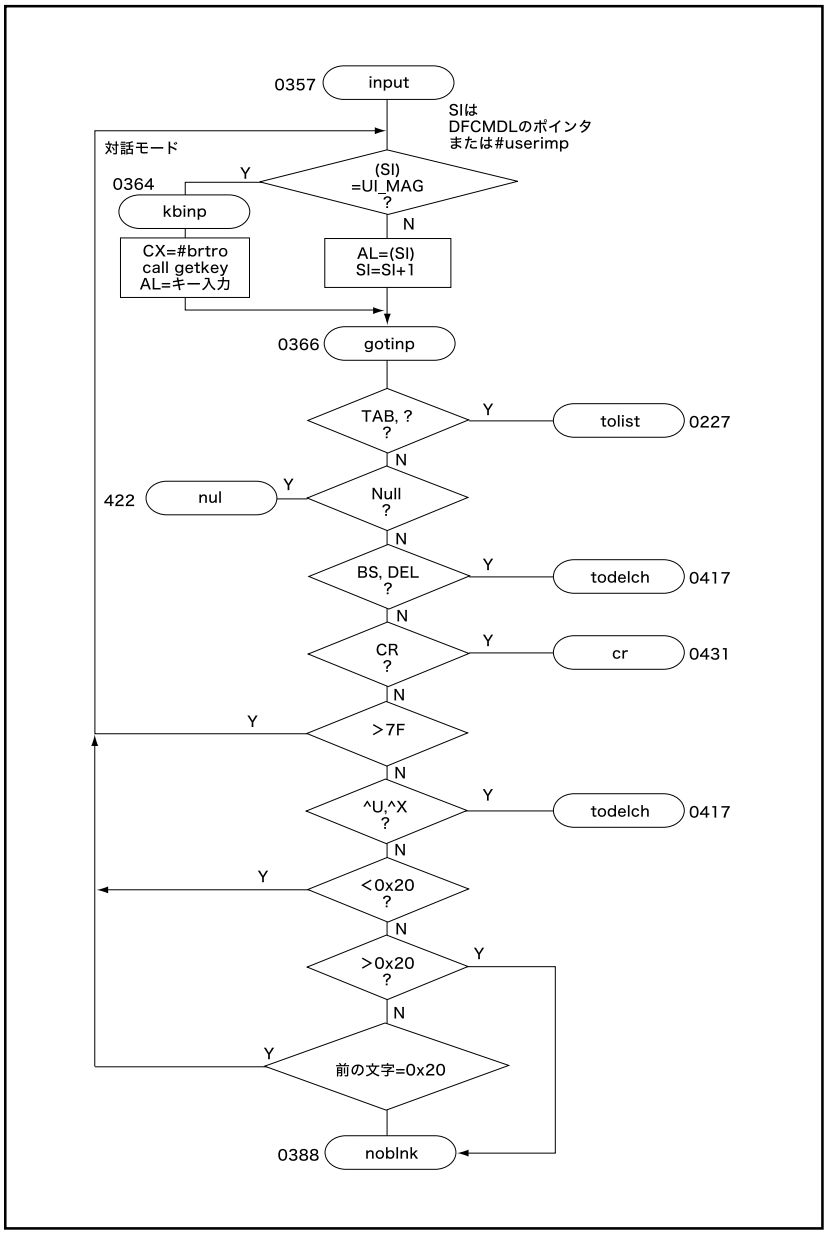
```
0483   vsknb:   lodsb                           ! AL=[DS:SI]コマンドラインのデーターをロード(SI=SI+1)
0484            stosb                           ! [ES:DI]=AL(DI=DI+1)
0485            cmp     al,#32                  ! ALはスペースか？
0486            je      chkvga                  ! Yesでchkvga(0457)へ(他のオプションのチェック)
0487            or      al,al                   ! AL=0か？(オプションの最後か？)AL=0でZF=1にセットされる
0488            jnz     vsknb                   ! AL=0でないときvsknb(0483)へ
0489            mov     bx,#crlf                ! BX=メッセージ改行のアドレス
0490            call    say                     ! 改行する

“vga=”、“kbd”、“lock=”、“mem=”のスキャンはここまで

0491            cmp     di,#cmdline+1           ! DI=#cmdine+1？(cmdlineは空か？)
0492   emptyl:  je      bfirst                  ! Yesでbfirst(0562)へ　最初のイメージを起動する
0493            jmp     bcmd                    ! Noでbcmd(0501)へ　他のイメージを起動する
0494
0495   toiloop:br       iloop                   ! iloop(0314)へ
0496   toinput:br       input                   ! input(0357)へ
0497
0498

他のブートイメージを見つけ、起動する
0501   bcmd:    mov     cx,#IMAGES              ! CX=#IMAGES(最大イメージ数)=19
                                                ! 繰り返し数=19(すべて調べる)
0502            mov     bx,#DESCR+2             ! BX=ポインタ(最初のイメージ名)
0503   nextn:   mov     si,bx                   ! SI=イメージ名のポインタ
0504            mov     di,#cmdline             ! DI=cmdlineの最初
0505   nextc:   mov     al,(si)                 ! AL=イメージ名の文字
0506            or      al,al                   ! AL=0か？　AL=0でZF=1にセットされる
0507            jz      dscend                  ! Yes(AL=0)でdscend(0522)へ
0508                                            ! 次の文字を得る
0509   #ifdef LCF_IGNORECASE
0510            call    upcase                  ! ALを大文字へ変換する
0511   #endif
0512            mov     ah,al                   ! AH=AL
0513            mov     al,(di)                 ! AL=cmdlineの文字
0514   #ifdef LCF_IGNORECASE
0515            call    upcase                  ! ALを大文字へ変換する
0516   #endif
0517            cmp     al,ah                   ! AL=AHか？　cmdlineとディスクリプタテブルの文字が同じか？
0518            jne     skipn                   ! Noでskipn(0526)次のイメージ名を取り込む
0519            inc     si                      ! SI=SI+1　ポインタを次の文字へ
0520            inc     di                      ! DI=DI+1
0521            jmp     nextc                   ! nextc(0505)へ(AL=AHなので次の文字をチェ ツク)
0522   dscend:  cmp     byte ptr (di),#32       ! comdlinはスペースか？
0523            je      boot                    ! Yesでboot(0577)へ(起動イメージを確定)
0524            cmp     byte ptr (di),#0        ! cmdlineは0？
0525            je      boot                    ! Yesでboot(0577)へ
0526   skipn:   add     bx,#DESCR_SIZE          ! BX=BX+DESCR_SIZE=0x34
0527            loop    nextn                   ! 次のイメージ名の先頭からチェックする
0528            mov     bx,#msg_nf              ! 起動イメージが見つからなかったので
0529            call    say                     !“No such image, Shows a list”を表示する
0530            br      iloop                   ! ilop(0314)へ戻る。(再入力する)

(今回は省略)

タイムアウト後にイメージを起動
0556   brto:    mov     bx,#crlf                ! BX=crlfのアドレス
0557            call    say                     ! 改行する
0558            jmp     brfrst                  ! bfirst(0562)へ
0559
0560                                            ! 最初のイメージからブート
```

リスト4 （続き）

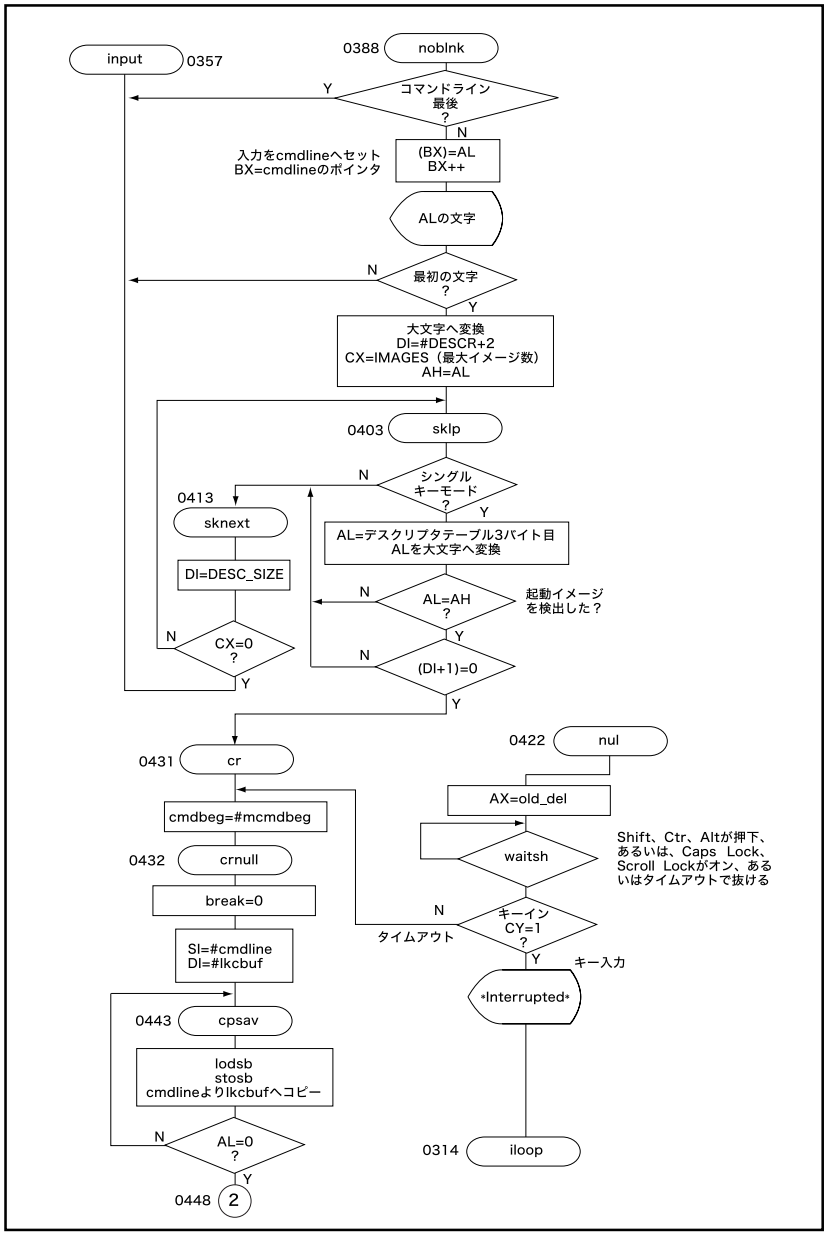
```
0561
0562  bfirst: mov    byte ptr lkcbuf,#0              ! lkcbuf=0バッファをクリア
0563          cmp     byte ptr cmdline,#0            ! コマンドラインは0か？
0564          jne     bcmd                           ! Noでbcmd(0501)へ(コマンドがあるので)
                                                     ! 起動イメージを表示して選択するルーチン
0565  brfrst: mov    bx,#DESCR+2                     ! コマンドラインは空なので最初のイメージを起動
                                                     ! BX=ディスクリプタテーブルの最初のイメージのアドレス
0566          mov     si,bx                          ! SI=BX  最初のイメージの名前をcmdlineへコピーする
0567          mov     di,#cmdline                    ! DI=コマンドラインのアドレス
0568          cld                                    ! SIはlodsbで増加する
0569  bfcpl:  lodsb                                  ! ディスクリプタテーブルから1文字、ALへロードする
                                                     ! SI=SI+1になる
0570          mov     (di),al                        ! cmdlineへALをストアする
                                                     ! すなわち、ディスクリプタテーブルからcmdlineへコピーする
0571          inc     di                             ! DI=DI+1  cmdlineのポインタを1増やす
0572          or      al,al                          ! AL=0か？
0573          jnz     bfcpl                          ! Noでbfcpl(0569)へ(次の文字をコピーする)

BXはディスクリプタテーブルで見つけたブートイメージのポインタパスワードの有りなしのチェック
0577  boot:   mov     si,#cmdline                    ! SI=コマンドラインバッファの最初のアドレス
0578  locopt: lodsb                                  ! AL=[DS:SI]  SI=SI+1になる
0579          or      al,al                          ! AL=0か？  AL=0でZF=1になる
0580          je      optfnd                         ! Yesでoptfnd(0586)へ(オプションなし)
0581          cmp     al,#32                         ! ALがスペースか？
0582          jne     locopt                         ! Noでlocopt(0578)繰り返す
0583          cmp     (si),#0                        ! 次の文字は0か？
0584          jne     optfnd                         ! Noでoptfnd(0586)へ(次のオプション有り)
0585          mov     byte ptr (si-1),#0             ! コマンドラインのスペースを0に置き換える
0586  optfnd: dec     si                             ! SI=SI-1(SIは文字がNULLのアドレスを指している)
0587          mov     options,si                     ! options=SI(オプションのポインタ)
0588          cmp     byte ptr (bx+MAX_IMAGE_NAME+1),#0 ! パスワードを宣言しているか？
                                                     ! BX=DESCR+2  MAX_IMAGE_NAME=15  ディスクリプタテーブルのオフセット18のバイト
0589          je      toboo                          ! Noでtoboot(0594)へ
0590          test    (bx+FLAGS_OFF),#FLAG_RESTR     ! パスワードあり？
                                                     ! BX=ディスクリプタテーブルのブートイメージのポインタ(FLAGS_OFF=0x30)
                                                     ! フラグ=#FLAG_RESTR=2か？
0591          jz      dopw                           ! Noでdopw(0595)へ  パスワードをチェックする
0592          cmp     byte ptr (si),#0               ! 他のオプションはあるか？
0593          jne     dopw                           ! Yes(ある)でdopw(0595)  パスワードをチェックする
0594  toboot: br      doboot  !doboot(0656)へ        ! パスワードなしでブート処理

パスワードのチェック
0595  dopw:   push    bx
```



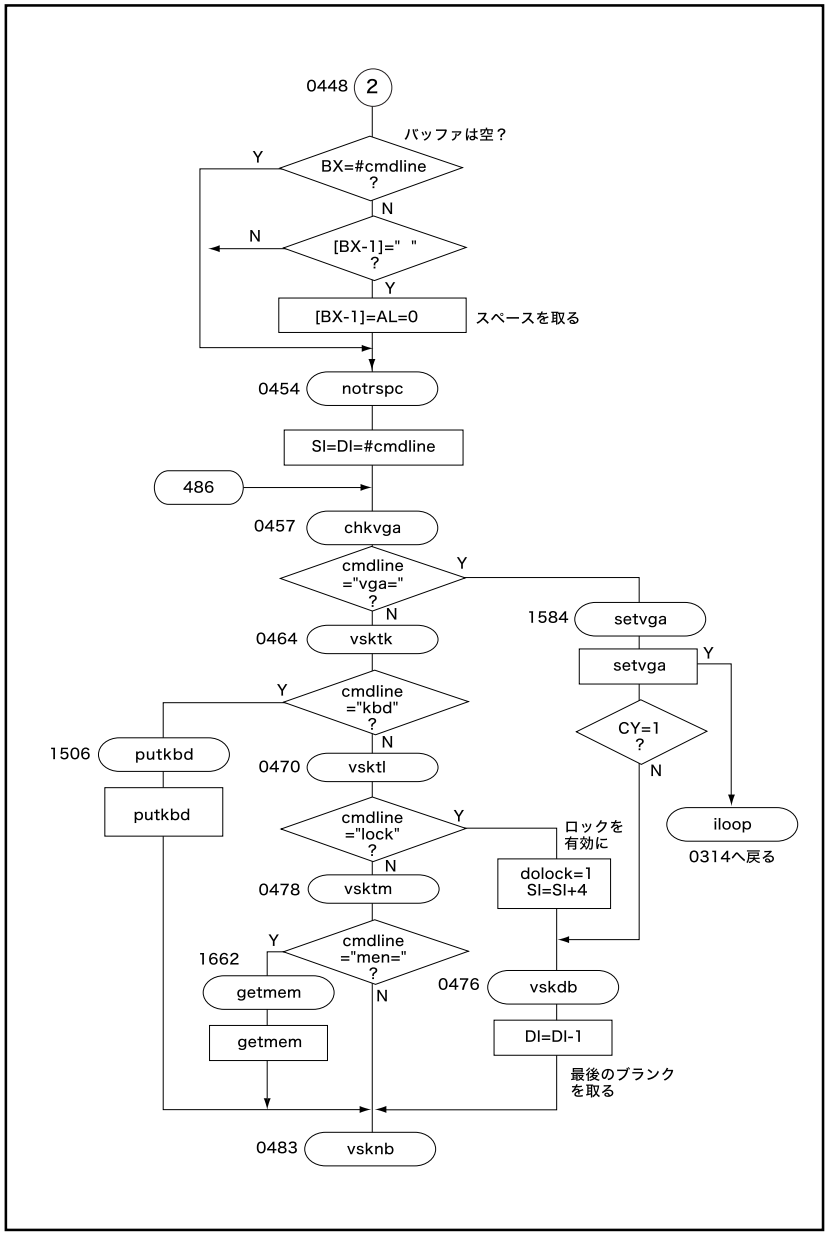
フローチャート1



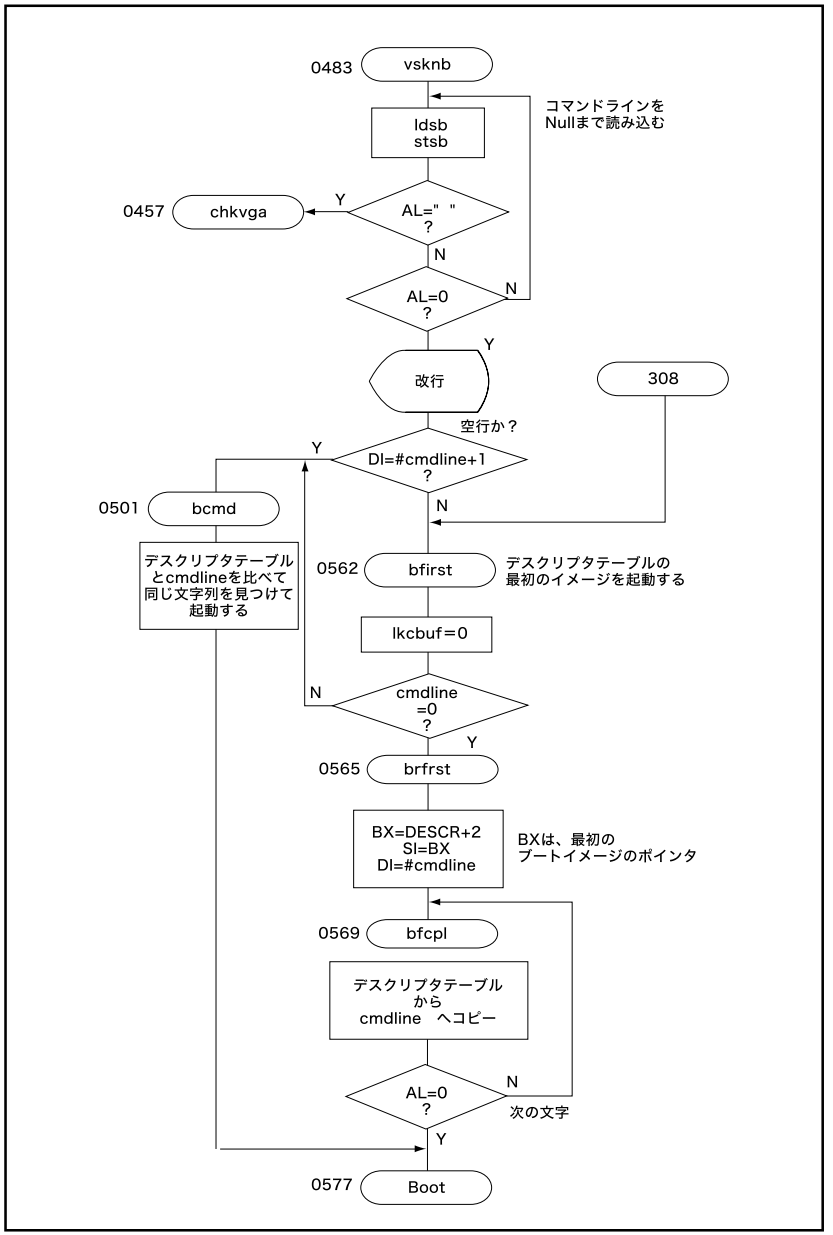
フローチャート2



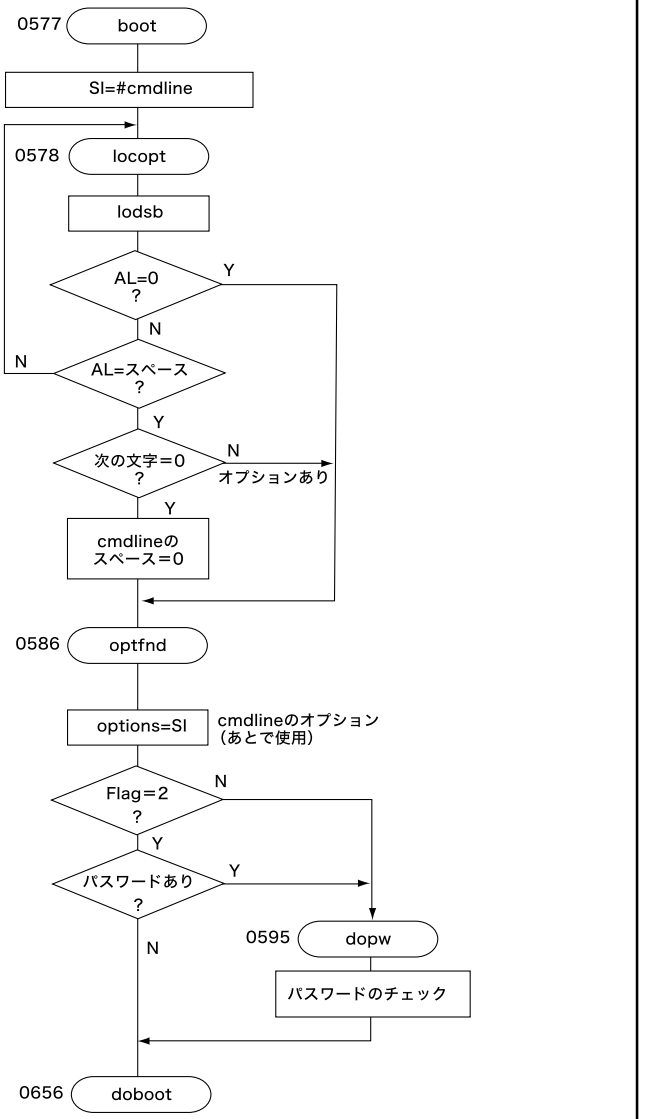
フローチャート3



フローチャート4



フローチャート5



フローチャート6