Chapitre 4

Implémentation des routines de service du fichier IO.COM

Étudions les routines de service définies dans IO.COM. Les routines de service des interruptions de MS-DOS seront implémentées dans MSDOS.COM. Cependant toute la partie dépendant fortement du matériel l'est dans IO.COM.

4.1 Implémentation des pré-routines des entrées-sorties standard

4.1.1 Nouvelle routine de service de l'interruption 16h

Le BIOS fournit bien sûr une routine de service pour l'interruption `16h`, chargée de récupérer un caractère dans le tampon du clavier. Les concepteurs de MS-DOS ont cependant décidé de surcharger celle-ci.

4.1.1.1 Routine principale

La nouvelle routine de service est repérée par l'étiquette KBINT, comme nous l'avons vu :

```
402 : ;
403 : ; Console keyboard interrupt handler.
404 : ;
405 : KBINT:
406 :     PUSH  AX
407 :     PUSH  SI
408 :     MOV   AL,20H            ;End of Interrupt command
409 :     OUT   BASE+2            ;Send to slave
410 :     IN    DATA              ;Get the character
411 :     AND   AL,7FH
412 :     CMP   AL,"C"-"@"
413 :     JZ    FLSH
414 :     CMP   AL,"S"-"@"
415 :     JZ   FLSH
416 :     CMP   AL,"F"-"@"
417 :     JNZ   SAVKY
418 : FLSH:
419 :     CALL  13*3,BIOSSEG       ; Call I/O system keyboard buffer flush.
420 : SAVKY:
421 :     SEG   CS
422 :     MOV   SI,[REAR]          ;Pointer to rear of queue
423 :     CALL  INCQ
424 :     SEG   CS
425 :     CMP   SI,[FRONT]         ;Any room in queue?
426 :     JZ    QFULL
427 :     SEG   CS
428 :     MOV   [SI],AL            ;Put character in queue
429 :     SEG   CS
430 :     MOV   [REAR],SI          ;Save pointer
431 : LEAVINT:
432 :     POP   SI
433 :     POP   AX
434 :     IRET
435 : QFULL:
436 :     MOV   AL,7               ; BELL character.
437 :     CALL  3*3,BIOSSEG        ; Call I/O system console output function.
438 :     JMPS  LEAVINT
```

— On sauvegarde sur la pile les contenus de `AX` et de `SI`, on envoie `20h` au registre de la carte mère pour inhiber les interruptions, on lit l'octet situé sur le port des données de l'interface série et on met son septième bit à zéro (lignes 406–411).

Les ports de statut et des données du contrôleur des entrées-sorties série sont donnés par les constantes STAT et DATA. La position du bit de données disponible est donnée par la constante DAV :

```
81 : STAT:   EQU    BASE+7      ; Serial I/O status port.
82 : DATA:   EQU    BASE+6      ; Serial I/O data port.
83 : DAV:    EQU    2           ; Data available bit.
```

— Si le caractère lu est CTRL-C, CTRL-S ou CTRL-F, on vide le tampon du clavier, on fait pointer `SI` à la fin du tampon, on appelle la sous-routine INCQ, étudiée ci-dessous, pour incrémenter le pointeur du tampon de clavier, en le faisant pointer au début du tampon si on a dépassé l'indice maximal, et on regarde s'il reste de la place dans le tampon. Si le tampon est plein, on émet un bip. Sinon on place le caractère obtenu dans le tampon et on sauvegarde la valeur du tampon dans la variable REAR. Dans les deux cas, on restaure les valeurs de `SI` et de `AX` et on termine la routine (lignes 412–416 et 418–434).

Les variables FRONT et REAR spécifient la position de début et de fin dans le tampon circulaire. La variable QUEUE spécifie le début physique de la zone de la mémoire centrale consacrée au tampon. La constante ENDQ spécifie la fin physiquede la zone de la mémoire centrale consacrée au tampon. QSIZE est la taille du tampon :

```
77  : QSIZE:  EQU    80      ; Input queue size.
[...]
551 : FRONT:  DW    QUEUE
552 : REAR:   DW    QUEUE
553 : QUEUE:  DS    QSIZE
554 : ENDQ:   EQU   $
```

— Sinon on fait la même chose mais sans vider le tampon du clavier (lignes 416–417).

4.1.1.2 Sous-routine d'incrémentation (circulaire) du pointeur de la file d'attente

L'incrémentation (circulaire) du pointeur de la file d'attente s'effectue grâce à la routine INCQ :

```
537 : INCQ:
538 :     INC   SI
539 :     CMP   SI,ENDQ      ;Exceeded length of queue?
540 :     JB    RET
541 :     MOV   SI,QUEUE
542 :     RET
```

c'est-à-dire qu'on incrémente le pointeur et qu'on le fait pointer au début de la file d'attente si on a dépassé l'indice maximal.

4.1.2 Pré-routine de la fonction 0Bh de vérification du statut du clavier

4.1.2.1 Routine principale

La pré-routine de service de la fonction 0Bh de vérification du statut du clavier de l'interruption `21h`, repérée par l'étiquette STATUS, est différente suivant que la saisie s'effectue par interrogation ou par interruption :

```
356 : ;
357 : ; ************ CONSOLE INPUT ************
358 : ;
359 :
360 :     IF    INTINP-1   ; Non-interrupt driven input.
361 : STATUS:
362 :     IN    STAT
363 :     AND   AL,DAV
364 :     JZ    NOTHING     ; Jump if nothing there.
365 :     PUSHF             ; Save Z flag.
366 :     INB   DATA
367 :     AND   AL,7FH
368 :     SEG   CS
369 :     MOV   [QUEUE],AL  ; Put new character in buffer.
370 :     POPF              ; Return with Z flag clear.
371 :     RET   L
372 : NOTHING:
373 :     SEG   CS
374 :     MOV   AL,[QUEUE]  ; See if there's anything in the buffer.
375 :     NOT   AL          ; Set up the Z flag.
376 :     TEST  AL,80H
377 :     PUSHF
378 :     NOT   AL
379 :     POPF
380 :     RET   L
[...]
401 :     IF INTINP         ; Interrupt-driven input.
402 : ;
403 : ; Console keyboard interrupt handler.
404 : ;
[...]
440 : STATUS:
441 :     PUSH  SI
442 : ;See if printer ready
443 :     IN    PRNSTAT
444 :     AND   AL,TBMT
445 :     JZ    NOPRN
446 :     SEG   CS
447 :     MOV   SI,[PFRONT]
448 :     SEG   CS
449 :     CMP   SI,[PREAR]    ;Anything in print queue?
450 :     JNZ   SENDPRN
451 :     SEG   CS
452 :     CMP   B,[PRNFCB],-1 ;Print spooling in progress?
453 :     JZ    NOPRN         ;If not, nothing to print
454 : ;Print spooling in progress. Get next buffer
455 :     PUSH  DS
456 :     PUSH  CS
457 :     POP   DS
458 :     PUSH  AX
459 :     PUSH  CX
460 :     PUSH  DX
461 :     PUSH  [STKSAV]
462 :     PUSH  [STKSAV+2]
```

```
463 :     PUSH  [DMAADD]
464 :     PUSH  [DMAADD+2]
465 :     MOV   DX,PQUEUE
466 :     MOV   AH,26         ;Set DMA address
467 :     INT   33
468 :     MOV   DX,PRNFCB
469 :     MOV   CX,PBUFSIZ
470 :     MOV   AH,39         ;Read buffer
471 :     INT   33
472 :     OR    AL,AL
473 :     JZ    NOTEOF
474 :     MOV   B,[PRNFCB],-1 ;Turn off print spooling at EOF
475 : NOTEOF:
476 :     POP   [DMAADD+2]
477 :     POP   [DMAADD]
478 :     POP   [STKSAV+2]
479 :     POP   [STKSAV]
480 :     MOV   SI,CX
481 :     POP   DX
482 :     POP   CX
483 :     POP   AX
484 :     POP   DS
485 :     OR    SI,SI
486 :     JZ    NOPRN
487 :     ADD   SI,PQUEUE-1
488 :     SEG   CS
489 :     MOV   [PREAR],SI
490 :     MOV   SI,ENDPQ-1
491 : SENDPRN:
492 :     CALL   INCPQ
493 :     SEG   CS
494 :     MOV   [PFRONT],SI
495 :     SEG   CS
496 :     LODSB               ;Get character to print
497 :     OUT   PRNDATA
498 : NOPRN:
499 :     DI                  ; Disable interrupts while checking queue.
500 :     SEG   CS
501 :     MOV   SI,[FRONT]
502 :     SEG   CS
503 :     CMP   SI,[REAR]     ; Anything in queue?
504 :     JZ    NOCHR         ; Jump if nothing in queue.
505 :     CALL  INCQ
506 :     SEG   CS
507 :     LODSB               ;Get character (if there is one)
508 :     OR    SI,SI         ;Reset zero flag
509 : NOCHR:
510 :     EI
511 :     POP   SI
512 :     RET   L             ;Zero clear if we have a character
```

Rappelons que le choix entre l'interrogation ou par interruption s'effectue en mettant faux ou vrai à la variable INTINP avant assemblage du fichier IO.ASM.

Cas de l'interrogation

Commentaires.- 1^o) Dans le cas de l'interrogation (ligne 360) :

— On lit l'octet de statut dans le registre de statut des entrées-sorties sur l'interface série pour voir s'il y a une donnée valide sur le registre des données. S'il y en a une, on sauvegarde sur la pile le registre des drapeaux, on lit l'octet de données, qui se trouve alors dans `AL`, on le masque par `7Fh` pour mettre le septième bit à zéro, on confond le segment des

données avec celui de code, on place le nouveau caractère dans la variable QUEUE et on revient avec le drapeau ZF baissé pour indiquer qu'une donnée valide se trouve dans la variable QUEUE (lignes 362–371).

La variable QUEUE contient le caractère obtenu par la routine de statut. Elle est initialisée à `FFh`, c'est-à-dire avec le septième bit levé pour indiquer que la donnée s'y trouvant n'est pas valide :

```
398 : QUEUE:    DB    -1    ; For storing characters from STATUS to INP.
```

— S'il n'y a pas de données valides, on confond le segment des données avec le segment de code, on regarde s'il y a, par ailleurs, une donnée valide dans la variable QUEUE, c'est-à-dire avec le septième bit nul et, dans ce cas, on lève le drapeau ZF (lignes 364 et 372–380).

Cas de pilotage par interruption

Commentaires.- 2^o) Dans le cas du pilotage par interruption (ligne 401) :

— On sauvegarde le contenu de `SI` sur la pile, on lit l'octet de statut de l'imprimante pour voir s'il y a quelque chose à imprimer (lignes 440–444).

Les ports de statut et de données du contrôleur d'imprimante sont donnés par les constantes PRNSTAT et PRNDATA :

```
99 :      IF    SERIALPRN
100 : PRNSTAT:EQU    SIOBASE+1
101 : PRNDATA:EQU    SIOBASE+0
102 :     ENDIF
103 :
104 :     IF    PARALLELPRN
105 : PRNSTAT:EQU    BASE+13
106 : PRNDATA:EQU    BASE+12
107 :     ENDIF
```

La position du bit indiquant que le tampon est vide est donnée par la constante TBMT :

```
84 : TBMT:    EQU    1     ; Transmitter buffer empty bit.
```

— Si c'est le cas, on confond le segment des données avec le segment de code et on fait pointer `SI` sur le tampon de l'imprimante pour voir s'il y a des octets à imprimer (lignes 445–450).

Les variables PFRONT et PREAR spécifient la position de début et de fin dans le tampon circulaire. La variable PQUEUE spécifie le début physique de la zone de la mémoire centrale consacrée au tampon. PBUFSIZ est la taille du tampon d'impression :

```
78  :    PBUFSIZ:  EQU   128     ; Size of print buffer
[...]
555 :    PFRONT:   DW    PQUEUE
556 :    PREAR:    DW    PQUEUE
557 :    PQUEUE:   DS    PBUFSIZ
```

— S'il y a quelque chose à imprimer, on regarde le premier octet du FCB de l'imprimante pour voir s'il y a des données en impression différée. S'il y en a, on sauvegarde sur la pile l'adresse du segment des données, on confond le segment des données avec le segment de code, on sauvegarde sur la pile les contenus des registres `AX`, `CX`, `DX`, des valeurs d'adresses STKSAV et DMAADD, on place le premier mot de la file d'attente d'impression dans `DX`, on appelle la fonction 26 de l'interruption `21h` pour changer l'adresse de la zone de transfert, on appelle la fonction 39 de l'interruption `21h` pour lire le tampon, si on est arrivé à la fin du tampon on place `FFh` dans le premier octet du FCB de l'imprimante pour indiquer qu'il n'y a plus d'impression en attente, on restaure l'adresse de la zone de transfert et la valeur pointée par STKSAV, on fait pointer `SI` sur ? ?, on restaure les valeurs des registres `DX`, `CX`, `AX` et l'adresse du segment des données, si le contenu de `SI` est non nul, on lui ajoute ? ?, résultat que l'on place dans PREAR, on fait pointer `SI` à la fin de la file d'attente d'impression (lignes 450–490).

Le FCB de l'imprimante est repéré par PRNFCB :

```
559  :    PRNFCB:   DB    -1
560  :              DS    36
```

La constante DOSDIF donne la taille, en octets, des copies en mémoire centrale de MSDOS.COM et de IO.COM. Les constantes STKSAV donnent les décalages de la zone de sauvegarde de la pile et DMAADD de celui de la zone de transfert :

```
1930 :    DOSDIF:   EQU    16*(DOSSEG-BIOSSEG)
1931 :    STKSAV:   EQU    1701H+DOSDIF
1932 :    DMAADD:   EQU    15B4H+DOSDIF
```

La constante ENDPQ repère la fin physique de la file d'attente d'impression :

```
558  :    ENDPQ:    EQU    $
```

— On appelle la routine INCPQ, étudiée ci-dessous, pour incrémenter le pointeur de la file d'attente d'impression, on confond le segment des données avec le segment de pile, on sauvegarde la valeur du pointeur dans la variable PFRONT, on charge le caractère à imprimer dans `AL` et on le transmet au port des données de l'imprimante (lignes 491–497).

— Qu'il y ait eu un caractère à imprimer ou non, on inhibe les interruptions masquable le temps de vérifier la file d'attente, on place la position sauvegardée du pointeur dans la file d'attente d'impression dans `SI` et on regarde s'il y a un autre caractère dans la file d'attente. S'il y en a un, on appelle la sous-routine INCQ pour incrémenter le pointeur de la file d'attente, on charge le caractère dans `AL` et on lève le drapeau ZF. Dans tous les cas, on se met à nouveau à l'écoute des interruptions masquables, on sauvegarde la position du pointeur sur la pile et on termine la routine, avec le drapeau ZF baissé s'il y a un caractère (lignes 486 et 498–512).

4.1.2.2 Sous-routine d'incrémentation (circulaire) du pointeur de la file d'attente d'impression

L'incrémentation (circulaire) du pointeur de la file d'attente d'impression s'effectue grâce à la routine INCPQ :

```
544 : INCPQ:
545 :     INC SI
546 :     CMP SI, ENDPQ ;Exceeded length of queue?
547 :     JB RET
548 :     MOV SI, PQUEUE
549 :     RET
```

c'est-à-dire qu'on incrémente le pointeur et qu'on le fait pointer au début de la file d'attente si on a dépassé l'indice maximal.

4.1.3 Pré-routine de la fonction 08h de lecture d'un caractère sur le clavier sans écho à l'écran

La pré-routine de service de la fonction `08h` de lecture d'un caractère sur le clavier sans écho à l'écran de l'interruption `21h`, repérée par l'étiquette INP, est différente suivant que la saisie s'effectue par interrogation ou par interruption :

```
356 : ;
357 : ; ************ CONSOLE INPUT ************
358 : ;
359 :
360 :     IF    INTINP-1       ; Non-interrupt driven input.
[...]
382 : INP:
383 :     MOV   AL,-1
384 :     SEG   CS
385 :     XCHG  AL,[QUEUE]     ; Remove the character from the buffer.
386 :     AND   AL,AL
387 :     JNS   INRET          ; Return if we have a character.
388 : INLOOP:
389 :     IN    STAT           ; Wait till a character is available.
390 :     AND   AL,DAV
391 :     JZ    INLOOP
392 :  IN    DATA
393 :     AND   AL,7FH
394 : INRET:
395 : FLUSH:
396 :     RET   L
[...]
401 :     IF    INTINP         ; Interrupt-driven input.
[...]
514 : INP:
515 :     CALL  STATUS,BIOSSEG ; Get I/O system console input status.
516 :     JZ    INP
517 :     PUSH  SI
518 :     DI                   ; Disable interrupts while changing queue pointers.
519 :     SEG   CS
520 :     MOV   SI,[FRONT]
521 :     CALL  INCQ           ; Permanently remove char from queue
522 :     SEG   CS
523 :     MOV   [FRONT],SI
524 :     EI
525 :     POP   SI
526 :     RET   L
```

Rappelons que le choix entre l'interrogation ou par interruption s'effectue en mettant à faux ou vrai la variable INTINP avant assemblage du fichier IO.ASM.

Cas de l'interrogation

Commentaires.- 1^o) Dans le cas de l'interrogation :

- On retire le caractère de la file d'attente, c'est-à-dire qu'on place le caractère du début de la file d'attente dans `AL` et on l'y remplace par `FFh`, pour indiquer que ce n'est pas un caractère valide, puis on termine la routine s'il s'agit d'un caractère valide, c'est-à-dire dont le septième bit est nul (lignes 382–387).
- S'il ne s'agit pas d'un caractère valide, c'est que le tampon est vide. On attend alors qu'un caractère soit disponible, c'est-à-dire qu'on lit en boucle le registre de statut jusqu'à ce qu'il nous dise qu'un caractère se trouve sur le port des données (lignes 388–391).
- Lorsqu'un caractère s'y trouve, on le lit dans `AL`, on met son septième bit à zéro et on termine la routine (lignes 392–396).

Cas de pilotage par interruption

Commentaires.- 2^o) Dans le cas du pilotage par interruption (ligne 401) :

- On appelle la pré-routine STATUS jusqu'à ce qu'un caractère se trouve dans la file d'attente (lignes 514–516).
- On sauvegarde le contenu de `SI` sur la pile, on inhibe les interruptions masquables le temps de changer la position du pointeur sur la file d'attente, on place le caractère du début du tampon dans `SI`, on appelle la sous-routine INCQ pour incrémenter le pointeur du tampon, ce qui revient à supprimer le caractère de la file d'attente, on se met à nouveau à l'écoute des interruptions masquables, on restaure la valeur de `SI` et on termine la routine (lignes 517–526).

4.1.4 Pré-routine de la fonction 02h d'affichage d'un caractère à l'écran

La pré-routine de service de la fonction `02h` de l'interruption `21h` d'affichage d'un caractère à l'écran est repérée par l'étiquette OUTP :

```
563 : ;
564 : ; ************ Console and Printer Output ************
565 : ;
566 : OUTP:
567 :     PUSH  AX
568 : OUTLP:
569 :     IN    STAT
570 :     AND   AL,TBMT
571 :     JZ    OUTLP
572 :     POP   AX
573 :     OUT   DATA
574 :     RET   L
```

c'est-à-dire qu'on sauvegarde le contenu de `AX` sur la pile, qu'on attend qu'il y ait un caractère à afficher, en lisant en boucle le registre de statut jusqu'à ce qu'il indique qu'il y en a un dans le tampon d'affichage. À ce moment, on restaure la valeur de `AX`, on envoie le caractère sur le registre des données du contrôleur d'affichage et on termine la routine.

4.1.5 Pré-routine de la fonction 05h d'impression d'un caractère

La pré-routine de service de la fonction `05h` de l'interruption `21h` d'impression d'un caractère est repérée par l'étiquette PRINT :

```
576 : PRINT:
577 :     PUSH SI
578 :     SEG  CS
579 :     MOV  SI, [PREAR]
580 :     CALL INCPQ
581 : PRINLP:
582 :     SEG  CS
583 :     CMP  SI, [PFRONT]
584 :     JNZ  PRNCHR
585 : ;Print queue is full
586 :     PUSH AX
587 :     CALL STATUS,BIOSSEG     ;Poll and maybe print something
588 :     POP  AX
589 :     JMPS PRINLP
590 : PRNCHR:
591 :     SEG  CS
592 :     MOV  [PREAR], SI
593 :     SEG  CS
594 :     MOV  [SI], AL
595 :     POP  SI
596 :     RET  L
```

c'est-à-dire qu'on sauvegarde la valeur de `SI` sur la pile, qu'on place dans `SI` la position de la fin de la file d'impression et qu'on appelle la routine INCPQ pour l'incrémenter, puisqu'on va placer un caractère de plus dans la file d'attente. Si celle-ci est pleine, on commence par imprimer le contenu de la file d'attente d'impression : on sauvegarde le contenu de `AX` sur la pile, on appelle la pré-routine STATUS pour obtenir l'octet d'état et imprimer un octet le cas échéant, on restaure la valeur de `AX` et on recommence si on a imprimé un caractère, c'est-à-dire tant qu'on n'a pas vidé la file d'attente d'impression. À la fin, on sauvegarde la position dans la file d'attente dans la variable PREAR, on place le caractère dans la file d'attente d'impression, on restaure la valeur de `SI` et on termine la routine.

4.1.6 Pré-routine de la fonction 03h de lecture d'un caractère sur l'interface auxiliaire

La pré-routine de service de la fonction `03h` de l'interruption `21h` de lecture d'un caractère sur l'interface auxiliaire est repérée par l'étiquette AUXIN :

```
597 : ;
598 : ; ************ Auxiliary I/O ************
599 : ;
600 : AUXIN:
601 :     IN    AUXSTAT
602 :     AND   AL,DAV
603 :     JZ    AUXIN
604 :     IN    AUXDATA
605 :     RET   L
```

c'est-à-dire qu'on lit en boucle l'octet se trouvant sur le port de statut de l'interface auxiliaire et, lorsque celui-ci indique la présence d'une donnée disponible, on récupère l'octet de données sur le registre des données de l'interface auxiliaire, que l'on place dans `AL` et on termine la routine.

Les ports de statut et des données de l'interface auxiliaire sont donnés par les constantes AUXSTAT et AUXDATA :

```
89 :     IF    SERIALAUX
90 : AUXSTAT:EQU    SIOBASE+3
91 : AUXDATA:EQU    SIOBASE+2
92 :     ENDIF
93 :
94 :     IF    PARALLELAUX
95 : AUXSTAT:EQU    BASE+13
96 : AUXDATA:EQU    BASE+12
97 :     ENDIF
```

4.1.7 Pré-routine de la fonction 04h d'envoi d'un caractère sur l'interface auxiliaire

La pré-routine de service de la fonction `04h` de l'interruption `21h` d'envoi d'un caractère sur l'interface auxiliaire est repérée par l'étiquette AUXOUT :

```
607 : AUXOUT:
608 :     PUSH  AX
609 : AUXLP:
610 :     IN    AUXSTAT
611 :     AND   AL,TBMT
612 :     JZ    AUXLP
613 :     POP   AX
614 :     OUT   AUXDATA
615 :     RET   L
```

c'est-à-dire qu'on sauvegarde la valeur du registre `AX`, contenant l'octet à envoyer, sur la pile et qu'on lit en boucle l'octet se trouvant sur le port de statut de l'interface auxiliaire. Lorsque celui-ci indique qu'il est prêt à recevoir une donnée, on restaure la valeur du registre `AX` dont on envoie l'octet de poids faible sur le registre des données de l'interface auxiliaire et on termine la routine.

4.2 Implémentation des pré-routines concernant les lecteurs de disquette

4.2.1 Pré-routine de la fonction 25h de lecture de secteurs sur la disquette

4.2.1.1 Routine principale

La fonction de lecture READ lit plusieurs secteurs de la disquette.

Avant d'appeler la routine, `AL` doit contenir le numéro du lecteur de disquette, `DS:BX` l'adresse en mémoire centrale à laquelle transférer ce qui est lu, `CX` le nombre de secteurs à lire et `DX` le numéro linéaire du premier secteur à lire.

Après son exécution, le drapeau CF est baissé si le transfert s'est effectué avec succès. Sinon il est levé, `CX` contient le nombre de secteurs qui n'ont pas été transférés et `AL` contient un code d'erreur :

— `02h` si la disquette n'est pas prête ;
— `04h` si le CRC n'est pas correct ;
— `06h` si la recherche du secteur a été infructueuse ;
— `08h` si le secteur n'a pas été trouvé :
— `12h` pour une erreur autre que celles ci-dessus :

```
816 : ;
817 : ; Disk read function.
818 : ;
819 : ; On entry:
820 : ;   AL = Disk I/O driver number
821 : ;   BX = Disk transfer address in DS
822 : ;   CX = Number of sectors to transfer
823 : ;   DX = Logical record number of transfer
824 : ; On exit:
825 : ;   CF clear if transfer complete
826 : ;
827 : ;   CF set if hard disk error.
828 : ;   CX = number of sectors left to transfer.
829 : ;   AL = disk error code
830 : ;        0  = write protect error
831 : ;        2  = not ready error
832 : ;        4  = "data" (CRC) error
833 : ;        6  = seek error
834 : ;        8  = sector not found
835 : ;        10 = write fault
836 : ;        12 = "disk" (none of the above) error
837 : ;
838 : READ:
839 :     CALL  SEEK     ;Position head
840 :     JC    ERROR
841 :     PUSH  ES       ; Make ES same as DS.
842 :     MOV   BX,DS
843 :     MOV   ES,BX
844 : RDLP:
845 :     CALL  READSECT ;Perform sector read
846 :     JC    POPESERROR
847 :     INC   DH       ;Next sector number
848 :     LOOP  RDLP     ;Read each sector requested
849 :     CLC            ; No errors.
850 :     POP   ES       ; Restore ES register.
851 :     RET   L
```

— On appelle la sous-routine SEEK, étudiée ci-dessous, pour positionner la tête de lecture (ligne 839).
— Si on n'y parvient pas, on va à la partie ERROR, étudiée ci-dessous, pour placer dans `AL` le double de la position dans la table des erreurs du code d'erreur contenu dans `AH` (ligne 840).
— Sinon on confond le segment supplémentaire avec le segment des données et on appelle la routine READSECT, étudiée ci-dessous, pour lire un secteur (lignes 841–845).
— Si on n'y est pas parvenu on va à la partie POPESERROR, étudiée ci-dessous en même temps que ERROR, pour restaurer la valeur de `ES` et placer dans `AL` le double de la position dans la table des erreurs du code d'erreur contenu dans `AH` (ligne 846).
— Sinon on incrémente `DH` et on recommence pour lire le secteur suivant (lignes 847–848).
— Lorsqu'on est arrivé à lire tous les secteurs, on baisse l'indicateur de retenue pour indiquer qu'il n'y a pas eu d'erreur, on restaure la valeur de `ES` et on termine la routine (lignes 849–851).

4.2.1.2 Traduction du numéro linéaire en caractéristiques physiques

La numérotation linéaire des secteurs est une abstraction du système d'exploitation. La fonction SEEK traduit le numéro linéaire du premier secteur en caractéristiques physiques.

Avant d'appeler la routine, `AL` doit contenir le numéro du lecteur de disquette, `DS:BX` l'adresse en mémoire centrale à laquelle transférer ce qui est lu, `CX` le nombre de secteurs à lire, `DX` le numéro linéaire du premier secteur à lire.

Après son exécution, `AH` contient l'octet de sélection du lecteur de disquette, `DL` le numéro de piste, `DH` le numéro de secteur sur cette piste, `DS:SI` pointe sur la zone de transfert, `CS:DI` sur le compteur de piste et `CX` contient toujours le nombre de secteurs à lire :

```
942  : ;
943  : ; Function:
944  : ;     Seeks to proper track.
945  : ; On entry:
946  : ;     Same as for disk read or write above.
947  : ; On exit:
948  : ;     AH = Drive select byte
949  : ;     DL = Track number
950  : ;     DH = Sector number
951  : ;     SI = Disk transfer address in DS
952  : ;     DI = pointer to drive's track counter in CS
953  : ;     CX unchanged (number of sectors)
954  : ;
955  : SEEK:
956  :       MOV  SI,BX          ; Save transfer address
957  :       CBW
958  :       MOV  BX,AX          ; Prepare to index on drive number
959  :
960  :       IF   WD1791         ; If two disk formats per drive.
961  :       SHR  AL             ; Convert to physical disk drive number.
962  :       ENDIF
963  :
964  :       CALL CHKNEW         ; Unload head if changing drives.
965  :       SEG  CS
966  :       MOV  AL,[BX+DRVTAB] ; Get drive-select byte.
967  :
968  :       IF   CROMEMCO16FDC
969  :       CALL MOTOR          ; Wait for the motors to come up to speed.
970  :       ENDIF
971  :
972  :       OUTB DISK+4         ; Select drive.
973  :
974  :       IF   CROMEMCO
975  :       OR   AL,80H         ; Set auto-wait bit.
976  :       ENDIF
977  :
978  :       MOV  AH,AL          ; Save drive-select byte in AH.
979  :       XCHG AX,DX          ; AX = logical sector number.
980  :       MOV  DL,26          ; 26 sectors/track unless changed below
981  :
982  :       IF   SCP
983  :       TEST DH,SMALLBIT    ; Check if small disk.
984  :       JZ   BIGONE         ; Jump if big disk.
985  :       MOV  DL,18          ; Assume 18 sectors on small track.
986  :       TEST DH,DDENBIT     ; Check if double-density.
987  :       JZ   HAVSECT        ; Jump if not.
988  :       MOV DL,SMALLDDSECT  ; Number of sectors on small DD track.
989  :       JP   HAVSECT
990  : BIGONE:
991  :       TEST DH,DDENBIT     ; Check if double-density.
```

```
992  :       JZ   HAVSECT        ; Jump if not.
993  :       MOV  DL,LARGEDDSECT ; Number of sectors on big DD track.
994  :       ENDIF
995  :
996  :       IF   TARBELLDD      ; Tarbell DD controller.
997  :       TEST DH,DDENBIT     ; Check for double-density.
998  :       JZ   HAVSECT
999  :       MOV  DL,LARGEDDSECT ; Number of sectors on DD track.
1000 :      ENDIF
1001 :
1002 :      IF   CROMEMCO4FDC
1003 :      TEST DH,SMALLBIT    ; Check if small disk.
1004 :      JNZ  HAVSECT        ; Jump if not.
1005 :      MOV  DL,18          ; 18 sectors on small disk track.
1006 :      ENDIF
1007 :
1008 :      IF   CROMEMCO16FDC
1009 :      TEST DH,SMALLBIT    ; Check if small disk.
1010 :      JNZ  BIGONE         ; Jump if big disk.
1011 :      MOV  DL,18          ; Assume 18 sectors on small track.
1012 :      TEST  DH,DDENBIT    ; Check if double-density.
1013 :      JZ   HAVSECT        ; Jump if not.
1014 :      MOV  DL,SMALLDDSECT ; Number of sectors on small DD track.
1015 :      JP   HAVSECT
1016 : BIGONE:
1017 :      TEST DH,DDENBIT     ; Check if double-density.
1018 :      JZ   HAVSECT        ; Jump if not.
1019 :      MOV  DL,LARGEDDSECT ; Number of sectors on big DD track.
1020 :      ENDIF
1021 :
1022 : HAVSECT:
1023 :      DIV  AL,DL          ; AL = track, AH = sector.
1024 :      XCHG AX,DX          ; AH has drive-select byte, DX = track & sector.
1025 :      INC  DH             ; Sectors start at one, not zero.
1026 :      SEG  CS
1027 :      MOV  BL,[BX+TRKPT]  ; Get this drive's displacement into track table.
1028 :      ADD  BX,TRKTAB      ; BX now points to track counter for this drive.
1029 :      MOV  DI,BX
1030 :      MOV  AL,DL          ; Move new track number into AL.
1031 :      SEG  CS
1032 :      XCHG AL,[DI]        ; Xchange current track with desired track
1033 :      OUT  DISK+1         ; Inform controller chip of current track
1034 :      CMP  AL,DL          ; See if we're at the right track.
1035 :      JZ   RET
1036 :      MOV  BH,2           ; Seek retry count
1037 :      CMP  AL,-1          ; Head position known?
1038 :      JNZ  NOHOME         ; If not, home head
1039 : TRYSK:
1040 :      CALL HOME
1041 :      JC   SEEKERR
1042 : NOHOME:
1043 :      MOV  AL,DL          ; AL = new track number.
1044 :      OUT  DISK+3
1045 :      MOV  AL,1CH+STPSPD  ; Seek command.
1046 :      CALL MOVHEAD
1047 :      AND  AL,98H         ; Accept not ready, seek, & CRC error bits.
1048 :      JZ   RET
1049 :      JS   SEEKERR        ; No retries if not ready
1050 :      DEC  BH
1051 :      JNZ  TRYSK
1052 : SEEKERR:
1053 :      MOV  AH,AL          ; Put status in AH.
```

```
1054 :      TEST AL,80H          ; See if it was a Not Ready error.
1055 :      STC
1056 :      JNZ  RET             ; Status is OK for Not Ready error.
1057 :      MOV  AH,2            ; Everything else is seek error.
1058 :      RET
```

— On sauvegarde la valeur de `BX`, le décalage de la zone de transfert, dans `SI` (ligne 956).
— On place le numéro de lecteur de disquette dans `BX` (lignes 957–958).
— Dans le cas d'un contrôleur de lecteur de disquette WD1791, on convertit le numéro de lecteur de disquette (lignes 960–962).
— On appelle la sous-routine CHKNEW, étudiée ci-dessous, pour déplacer la tête de lecture-écriture s'il s'agit d'un nouveau lecteur de disquette (ligne 964).
— On place l'octet de sélection du lecteur de disquette dans `AL` (lignes 965–966).
— Si le contrôleur de lecteur de disquette est d'un certain type, on attend que son moteur atteigne sa vitesse de croisière, grâce à la sous-routine MOTOR étudiée ci-dessous (lignes 968–970).
— On sélectionne le lecteur de disquette sur le contrôleur de lecteur de disquette (ligne 972).

  La constante DISK spécifie le numéro de port du premier registre du contrôleur de disquette, qui dépend du contrôleur de lecteur de disquette. Par exemple, pour SCP, on a :

```
640 : DISK:    EQU    0E0H
```

— Pour un certain type de contrôleur de lecteur de disquette, il faut lever le bit d'attente (lignes 974–976).
— On sauvegarde l'octet de sélection du lecteur de disquette dans `AH` et on échange `AX` et `BX` pour obtenir dans `AX` le numéro logique de secteur (lignes 978–980).
— Pour les contrôleurs de lecteur de disquette SCP, qui admettent plusieurs types de disquette, on ajuste en fonction de ce type (lignes 982–994).

  Les constantes SMALLBIT, BACKBIT, DDENBIT (pour bit de double densité) et DONEBIT sont définies pour SCP de la façon suivante :

```
635 :      IF    SCP
636 : SMALLBIT:   EQU    10H
637 : BACKBIT:    EQU    04H
638 : DDENBIT:    EQU    08H
639 : DONEBIT:    EQU    01H
640 : DISK:       EQU    0E0H
641 :      ENDIF
```

— On fait de même pour le contrôleur de lecteur de disquette Tarbell (lignes 996–1000).

  Les constantes BACKBIT, DDENBIT, DONEBIT et DISK sont définies pour le contrôleur de lecteur de disquette Tarbell de la façon suivante :

```
643 :      IF    TARBELL
644 : BACKBIT:    EQU    40H
645 : DDENBIT:    EQU    08H
646 : DONEBIT:    EQU    80H
647 : DISK:       EQU    78H
648 :      ENDIF
```

— On fait de même pour le contrôleur de lecteur de disquette Cromemco 4 (lignes 1002–1006).

  Les constantes SMALLBIT, BACKBIT, DDENBIT, DONEBIT et DISK sont définies pour le contrôleur de lecteur de disquette Cromenco de la façon suivante :

```
650 :      IF    CROMEMCO
651 : SMALLBIT:   EQU    10H
652 : BACKBIT:    EQU    0FDH      ; Send this to port 4 to select back.
653 : DDENBIT:    EQU    40H
654 : DONEBIT:    EQU    01H
```

```
655 : DISK:       EQU    30H
656 :     ENDIF
```

— On fait de même pour le contrôleur de lecteur de disquette Cromemco 16 (lignes 658–672).

Les constantes SMALLDDSECT et LARGEDDSECT sont définies de la façon suivante :

```
658 :     IF    SMALLDS-1
659 : SMALLDDSECT:    EQU    8
660 :     ENDIF
661 :
662 :     IF    SMALLDS
663 : SMALLDDSECT:    EQU    16
664 :     ENDIF
665 :
666 :     IF    LARGEDS-1
667 : LARGEDDSECT:    EQU    8
668 :     ENDIF
669 :
670 :     IF    LARGEDS
671 : LARGEDDSECT:    EQU    16
672 :     ENDIF
```

— On divise le numéro logique de secteur par le nombre de secteurs par piste pour obtenir le numéro de piste dans `AL` et le numéro de secteur sur cette piste dans `AH`, on place l'octet de sélection du lecteur de disquette dans `AX` et les numéros de piste et de secteur dans `DX`, on incrémente `DH` car les numéros de secteur sur une piste commencent à 1 et non à zéro, on récupère dans `BL` le déplacement dans la table TRKPT du contrôleur de lecteur de disquette adéquat, puis on fait pointer `DI` sur la table TRKTAB adéquate, on place le nouveau numéro de piste dans `AL`, on échange le numéro de piste en cours et celui désiré et on informe le contrôleur de lecteur de disquette de celui-ci (lignes 1022–1033.

La table TRKPT dépend du contrôleur de lecteur de disquette. Pour chacun des contrôleurs, la table spécifie les lecteurs de disquette, parmi les 16 possibles, l'utilisant. Par exemple le contrôleur PerSCi 277 ou 299 est utilisé pour les lecteurs de disquette numéros 4 et 5 :

```
1474 : ; TRKPT is a table of bytes which indicates which TRKTAB entry each
1475 : ; disk I/O driver should use.  Since each physical drive may be used for
1476 : ; more than one disk I/O driver, more than one entry in TRKPT may point
1477 : ; to the same entry in TRKTAB.  Drives such as PerSci 277s which use
1478 : ; the same head positioner for more than one drive should share entrys
1479 : ; in TRKTAB.
[...]
1484 :  IF SCP*COMBIN*FASTSEEK
1485 : ;
1486 : ; A PerSci 277 or 299 and one 5.25-inch drive.
1487 : ;
1488 : DRVTAB: DB 00H,08H,01H,09H,10H,18H,00H,08H,01H,09H
1489 : TRKPT: DB 0,0,0,0,1,1,0,0,0,0
```

La table TRKTAB spécifie, pour chaque contrôleur de lecteur de disquette et chaque lecteur de disquette reliée à ce contrôleur, le numéro de tête de lecture-écriture utilisée, en un octet. Par exemple, pour le contrôleur PerSCi 277 ou 299, on initialise le numéro de tête de lecture-écriture utilisée pour les lecteurs de disquette numéros 4 et 5 à `FFh` :

```
1470 : ; TRKTAB is a table of bytes used to store which track the read/write
1471 : ; head of each drive is on.  Each physical drive should have its own
1472 : ; entry in TRKTAB.
[...]
1484 :  IF SCP*COMBIN*FASTSEEK
1485 : ;
1486 : ; A PerSci 277 or 299 and one 5.25-inch drive.
```

```
1487 : ;
1488 : DRVTAB:   DB 00H,08H,01H,09H,10H,18H,00H,08H,01H,09H
1489 : TRKPT:    DB 0,0,0,0,1,1,0,0,0,0
1490 : TRKTAB:   DB -1,-1
```

— On termine la routine si on se trouve sur la bonne piste (lignes 1034–1035).
— Sinon on réessaie. Pour cela on place 2 dans `BH` et on regarde si on est à une position connue de la tête de lecture-écriture. Si ce n'est pas le cas, on appelle la sous-routine HOME, étudiée ci-dessous, pour placer la tête de lecture-écriture sur la piste 0 (lignes 1034 et 1036–1038).
— Si on n'y parvient pas, on place l'octet de statut dans `AL`, et si le septième bit est levé, on lève l'indicateur de retenue CF. Si l'octet de statut est correct, on termine la routine. Sinon on place le code d'erreur `02h` dans `AH` et on termine la routine (lignes 1041 et 1052–1058).
— On place le nouveau numéro de piste dans `AL`, que l'on transmet au registre du contrôleur. On place l'octet de commande de recherche dans `AL` et on appelle la sous-routine MOVHEAD, étudiée ci-dessous, pour déplacer la tête de lecture-écriture. Si on a réussi, on termine la routine. Si on a une erreur, on procède comme ci-dessus, le cas échéant en décrémentant `BH` et en réssayant la recherche (lignes 1043–1051).

4.2.1.3 Détermination du changement de lecteur de disquette

La sous-routine CHKNEW vérifie si le numéro de lecteur de disquette transmis en paramètre, *via* le registre `AL`, est celui du lecteur de disquette par défaut, levant le drapeau ZF si c'est le cas :

```
779 : CHKNEW:
780 :     MOV   AH,AL       ; Save disk drive number in AH.
781 :     SEG   CS          ; AL = previous disk drive number,
782 :     XCHG  AL,[CURDRV] ;  make new drive current.
783 :     CMP   AL,AH       ; Changing drives?
784 :     JZ    RET
785 : ;
786 : ; If changing drives, unload head so the head load delay one-shot will
787 : ; fire again. Do it by seeking to the same track with the H bit reset.
788 : ;
789 :     IN    DISK+1      ; Get current track number
790 :     OUT   DISK+3      ; Make it the track to seek to
791 :     MOV   AL,10H      ; Seek and unload head
792 :     CALL  DCOM
793 :     MOV   AL,AH       ; Restore current drive number
794 :     RET
```

— On sauvegarde le numéro transmis en paramètre dans `AH` et on le compare avec le numéro du lecteur de disquette par défaut (lignes 780–783).
— S'il s'agit du lecteur de disquette par défaut (le drapeau ZF est levé), on termine la routine (ligne 784).
— Sinon, on récupère le numéro de piste en cours, depuis le registre adéquat du contrôleur de lecteur de disquette, qu'on prend comme numéro de piste à chercher et on appelle la sous-routine DCOM, étudiée ci-dessous, pour déplacer la tête de lecture-écriture du lecteur de disquette, on restaure la valeur du lecteur de disquette par défaut et on termine la routine (lignes 785–794).

4.2.1.4 Sous-routine de déplacement de la tête de lecture-écriture

La sous-routine DCOM déplace la tête de lecture-écriture du lecteur de disquette :

```
1413 : DCOM:
1414 :    OUT  DISK
1415 :    PUSH AX
1416 :    AAM          ;Delay 10 microseconds
1417 :    POP  AX
1418 : GETSTAT:
1419 :    IN   DISK+4
1420 :    TEST AL,DONEBIT
1421 :
1422 :    IF   TARBELL
1423 :    JNZ  GETSTAT
1424 :    ENDIF
1425 :
1426 :    IF   SCP+CROMEMCO
1427 :    JZ   GETSTAT
1428 :    ENDIF
1429 :
1430 :    IN   DISK
1431 :    RET
```

On envoie le contenu de `AL` au registre du contrôleur du lecteur de disquette, on sauvegarde le contenu de `AX` sur la pile, on effectue une instruction fictive qui prend 10 microsecondes et on restaure la valeur de `AX`. On récupère la valeur du registre de statut du contrôleur de disquette, et on boucle jusqu'à ce que la tête soit positionnée. On lit alors la valeur dans le registre du contrôleur de disquette et on termine la routine.

4.2.1.5 Sous-routine de démarrage du moteur d'un lecteur de disquette

La sous-routine MOTOR permet au moteur du lecteur de disquette d'atteindre sa vitesse de croisière.

Elle est définie lignes 796–815 :

```
796 :      IF    CROMEMCO16FDC
797 : MOTOR:
798 :      PUSH  AX
799 :      MOV   AH,AL
800 :      IN    DISK+4     ; See if the motor is on.
801 :      TEST  AL,08H
802 :      MOV   AL,AH
803 :      OUTB  DISK+4     ; Select drive & start motor.
804 :      JNZ   MOTORSON   ; No delay if motors already on.
805 :      PUSH  CX
806 :      MOV   CX,43716   ; Loop count for 1 second.
807 : MOTORDELAY:           ;  (8 MHz, 16-bit memory).
808 :      AAM              ; 83 clocks.
809 :      AAM              ; 83 clocks.
810 :      LOOP  MOTORDELAY ; 17 clocks.
811 :      POP   CX
812 : MOTORSON:
813 :      POP   AX
814 :      RET
815 :      ENDIF
```

4.2.1.6 Sous-routine de positionnement sur la piste 0

La sous-routine HOME place la tête de lecture-écriture sur la piste zéro. Elle dépend du contrôleur de lecteur de disquette utilisé. Par exemple, pour SCP elle est définie lignes 1309–1347 :

```
1309 : ;
1310 : ; Subroutine to restore the read/write head to track 0.
1311 : ;
1312 :     IF   SCP+CROMEMCO+TARBELL*(FASTSEEK-1)
1313 : HOME:
1314 :     ENDIF
1315 :
1316 :     IF   FASTSEEK*CROMEMCO
1317 :     TEST AH,SMALLBIT      ; Check for large disk.
1318 :     JNZ  RESTORE          ; Big disks are fast seek PerSci.
1319 :     ENDIF
1320 :
1321 :     MOV  BL,3
1322 : TRYHOM:
1323 :
1324 :     IF   SCP*FASTSEEK
1325 :     MOV  AL,AH            ; Turn on Restore to PerSci.
1326 :     OR   AL,80H
1327 :     OUTB DISK+4
1328 :     ENDIF
1329 :
1330 :     MOV  AL,0CH+STPSPD    ; Restore with verify command.
1331 :     CALL DCOM
1332 :     AND  AL,98H
1333 :
1334 :     IF   SCP*FASTSEEK
1335 :     MOV  AL,AH            ; Restore off.
1336 :     OUTB DISK+4
1337 :     ENDIF
1338 :
1339 :     JZ   RET
1340 :     JS   HOMERR           ; No retries if not ready
1341 :     MOV  AL,58H+STPSPD    ; Step in with update
1342 :     CALL DCOM
1343 :     DEC  BL
1344 :     JNZ  TRYHOM
1345 : HOMERR:
1346 :     STC
1347 :     RET
```

— On place 3 dans `BL`, le contenu de `AH` dans `AL`, on lève le septième bit de `AL` et on transmet au registre du contrôleur du lecteur de disquette (lignes 1321 et 1325–1327).
— On place ? ? dans `AL` et on appelle la sous-routine DCOM pour déplacer la tête de lecture-écriture du lecteur de disquette (lignes 1330–1331).

La constante STPSPD est définie ligne 70 :

```
70 : STPSPD:    EQU    0
```

— On masque le résultat `AL` obtenu pour n'en garder que les bits 3, 4 et 7 (ligne 1332).
— On restaure la valeur de `AL`, que l'on envoie au registre du contrôleur de lecteur de disquette (lignes 1335–1336).
— Si ? ?, on termine la routine (ligne 1339).
— On ne réessaie pas si le lecteur n'est pas prêt (ligne 1340).
— Sinon on appelle à nouveau la sous-routine DCOM pour déplacer la tête de lecture-écriture du lecteur de disquette, on décroît `BL` et on réessaie à nouveau si on n'y est pas parvenu

(lignes 1341–1344).
— On lève l'indicateur de retenue CF et on termine la routine (lignes 1346–1347).

4.2.1.7 Sous-routine de préparation au déplacement de la tête de lecture-écriture

La sous-routine MOVHEAD prépare le déplacement la tête de lecture-écriture du lecteur de disquette dans le cas de certains contrôleurs de lecteur de disquette :

```
1399 : ;
1400 : ; Subroutine to move the read/write head to the desired track.
1401 : ; Usually falls through to DCOM unless special handling for
1402 : ; PerSci drives is required in which case go to FASTSK.
1403 : ;
1404 :     IF    SCP+CROMEMCO+TARBELL*(FASTSEEK-1)
1405 : MOVHEAD:
1406 :     ENDIF
1407 :
1408 :     IF    CROMEMCO*FASTSEEK
1409 :     TEST  AH,SMALLBIT     ; Check for PerSci.
1410 :     JNZ   FASTSK
1411 :     ENDIF
1412 :
1413 : DCOM:
```

4.2.1.8 Sous-routine de traitement des erreurs

La sous-routine ERROR, un code d'erreur étant fourni dans `AH`, renvoie dans `AL` le double de sa position dans le table des erreurs :

```
868 : POPESERROR:
869 :       POP  ES       ; Restore ES register.
870 : ERROR:
871 :       MOV  BL,-1
872 :       SEG  CS
873 :       MOV  [DI],BL  ; Indicate we don't know where head is.
874 :       MOV  SI,ERRTAB
875 : GETCOD:
876 :       INC  BL       ; Increment to next error code.
877 :       SEG  CS
878 :       LODB
879 :       TEST AH,AL    ; See if error code matches disk status.
880 :       JZ   GETCOD   ; Try another if not.
881 :       MOV  AL,BL    ; Now we've got the code.
882 :       SHL  AL       ; Multiply by two.
883 :       STC
884 :       RET  L
885 :
886 : ERRTAB:
887 :       DB   40H      ;Write protect error
888 :       DB   80H      ;Not ready error
889 :       DB   8        ;CRC error
890 :       DB   2        ;Seek error
891 :       DB   10H      ;Sector not found
892 :       DB   20H      ;Write fault
893 :       DB   7        ;"Disk" error
```

— On place `FFh` à la position pointée par `DI` pour indiquer que l'on ne connaît pas la position de la tête de lecture-écriture (lignes 871–873).
— On fait pointer `SI` sur la table des codes d'erreur (ligne 874).
— On incrémente `BL` pour aller au code d'erreur suivant et on charge le premier octet du code d'erreur. On recommence jusqu'à ce qu'on arrive sur le code d'erreur entré en paramètre, *via* `AH`. (lignes 875–880).
— On place alors le contenu de `BL` dans `AL`, on le multiplie par deux, on lève l'indicateur de retenue CF et on termine la routine (lignes 881–884).

4.2.1.9 Sous-routine de lecture d'un secteur

La sous-routine de lecture d'un secteur est READSECT :

```
1180 : READSECT:
1181 :     CALL  SETUP
1182 :     MOV   BL,10         ; Retry count for hard error.
1183 :     XCHG  DI,SI         ; Transfer address to DI.
1184 :     PUSH  DX            ; Save track & sector number.
1185 :     MOV   DL,DISK+3     ; Disk controller data port.
1186 : RDAGN:
1187 :     OR    AL,READCOM
1188 :     OUT   DISK
1189 :
1190 :     IF    CROMEMCO
1191 :     MOV   AL,AH         ; Turn on auto-wait.
1192 :     OUT   DISK+4
1193 :     ENDIF
1194 :
1195 :     MOV   BP,DI         ; Save address for retry.
1196 :     JMPS  RLOOPENTRY
1197 : RLOOP:
1198 :     STOB                ; Write into memory.
1199 : RLOOPENTRY:
1200 :
1201 :     IF    SCP
1202 :     IN    DISK+5        ; Wait for DRQ or INTRQ.
1203 :     ENDIF
1204 :
1205 :     IF    TARBELL+CROMEMCO
1206 :     IN    DISK+4
1207 :     ENDIF
1208 :
1209 :     IF    TARBELL
1210 :     SHL   AL
1211 :     INB   DX            ; Read data from disk controller chip.
1212 :     JC    RLOOP
1213 :     ENDIF
1214 :
1215 :     IF    SCP+CROMEMCO
1216 :     SHR   AL
1217 :     INB   DX            ; Read data from disk controller chip.
1218 :     JNC   RLOOP
1219 :     ENDIF
1220 :
1221 :     EI                  ; Interrupts OK now
1222 :     CALL  GETSTAT
1223 :     AND   AL,9CH
1224 :     JZ    RDPOP
1225 :     MOV   DI,BP         ; Get origainal address back for retry.
1226 :     MOV   BH,AL         ; Save error status for report
1227 :     MOV   AL,0
1228 :     DEC   BL
1229 :     JNZ   RDAGN
1230 :     MOV   AH,BH         ; Put error status in AH.
1231 :     STC
1232 : RDPOP:
1233 :     POP   DX            ; Get back track & sector number.
1234 :     XCHG  SI,DI         ; Address back to SI.
1235 :
1236 :     IF    TARBELL
1237 : FORCINT:
```

```
1238 :     MOV   AL,0D0H      ; Tarbell controllers need this Force Interrupt
1239 :     OUT   DISK         ;  so that Type I status is always available
1240 :     MOV   AL,10        ;  at the 1771/1793 status port so we can find
1241 : INTDLY:                ;  out if the head is loaded.  SCP and Cromemco
1242 :     DEC   AL           ;  controllers have head-load status available
1243 :     JNZ   INTDLY       ;  at the DISK+4 status port.
1244 :     ENDIF
1245 :
1246 :     RET
```

Elle dépend du contrôleur de lecteur de disquette. Par exemple pour SCP :

— On appelle la sous-routine SETUP, étudiée ci-dessous, pour préparer la lecture, on place 10 dans `BL`, le nombre d'essais qui sera effectué, on place l'adresse de transfert dans `DI`, on sauvegarde les numéros de piste et de secteur sur la pile, on place le port des données du contrôleur de lecteur de disquette dans `DL`, on lève le bit de lecture et on transmet l'octet d'état au contrôleur du lecteur de disquette (lignes 1181–1188).

La valeur des constantes READCOM et WRITECOM dépend du contrôleur. Par exemple pour CSP elle sont définies de la façon suivante :

```
622 : WD1791:   EQU    SCP+TARBELLDD+CROMEMCO16FDC
623 : WD1771:   EQU    TARBELLSD+CROMEMCO4FDC
624 :
625 :     IF    WD1791
626 : READCOM:  EQU    80H
627 : WRITECOM: EQU    0A0H
628 :     ENDIF
```

— On sauvegarde la valeur de `DI` dans `BP` pour pouvoir la réutiliser lors du prochain essai et on passe à l'instruction suivante pour entrer dans la boucle (lignes 1195–1196).

— On lit la valeur du port du contrôleur de lecteur de disquette (lignes 1201–1203).

— On divise la valeur de `AL` par 2, on lit la valeur sur le port des données du contrôleur de lecteur de disquette, que l'on place dans la zone de transfert en mémoire centrale tant qu'il n'y a pas d'indicateur de retenue (lignes 1215–1219 et 1198).

— On rétablit les interruptions masquables et on appelle la routine GETSTAT, partie de la routine DCOM, pour positionner la tête de lecture-écriture (lignes 1221–1222).

— Si on a échoué, on réessaie : on redonne à `DI` la valeur sauvegardée, on sauvegarde le code d'erreur dans `BH`, on place 0 dans `AL`, on décroît le nombre d'essais restants à effectuer et, si on n'a pas atteint 0, on recommence (lignes 1223–1229).

— Si on ne réussit pas au bout de dix essais, on place le code d'erreur dans `AH` et on lève l'indicateur de retenue (lignes 1230–1231).

— Sinon on replace les numéros de piste et de secteur dans `DX`, l'adresse de transfert dans `SI` et on termine la sous-routine (lignes 1232–1234 et 1246).

4.2.1.10 Sous-routine de préparation à la lecture d'un secteur

La préparation à la lecture d'un secteur est effectuée par la sous-routine SETUP :

```
1060 : SETUP:
1061 :     MOV  BL,DH           ; Move sector number to BL to play with
1062 :
1063 :     IF   SCP+CROMEMCO16FDC
1064 :     TEST AH,DDENBIT      ; Check for double density.
1065 :     JZ   CHECKSMALL      ; Not DD, check size for SD.
1066 :     ENDIF
1067 :
1068 :     IF    TARBELLDD
1069 :     TEST  AH,DDENBIT     ; Check for double density.
1070 :     JZ    CHECK26        ; Not DD.
1071 :     ENDIF
1072 :
1073 :     IF    WD1791
1074 :
1075 :     IF    (SCP+TARBELL)*LARGEDS+SCP*SMALLDS
1076 :     MOV   AL,AH          ; Select front side of disk.
1077 :     OUT   DISK+4
1078 :     ENDIF
1079 :
1080 :     IF    CROMEMCO*(LARGEDS+SMALLDS)
1081 :     MOV   AL,0FFH        ; Select front side of disk.
1082 :     OUT   04H
1083 :     ENDIF
1084 :
1085 :     CMP   BL,8           ; See if legal DD sector number.
1086 :     JBE   PUTSEC         ; Jump if ok.
1087 :
1088 :     IF    (LARGEDS-1)*((SMALLDS*(SCP+CROMEMCO))-1)
1089 :     JP    STEP           ; If only SS drives, we gotta step.
1090 :     ENDIF
1091 :
1092 :     IF    SCP*LARGEDS*(SMALLDS-1)
1093 :     TEST  AH,SMALLBIT    ; Check for 5.25 inch disk.
1094 :     JNZ   STEP           ; Jump if small because SMALLDS is off.
1095 :     ENDIF
1096 :
1097 :     IF    SCP*SMALLDS*(LARGEDS-1)
1098 :     TEST  AH,SMALLBIT    ; Check for 8 inch disk.
1099 :     JZ    STEP           ; Jump if large because LARGEDS is off.
1100 :     ENDIF
1101 :
1102 :     IF    CROMEMCO16FDC*LARGEDS*(SMALLDS-1)
1103 :     TEST  AH,SMALLBIT    ; Check for 5.25 inch disk.
1104 :     JZ    STEP           ; Jump if small because SMALLDS is off.
1105 :     ENDIF
1106 :
1107 :     IF    CROMEMCO16FDC*SMALLDS*(LARGEDS-1)
1108 :     TEST  AH,SMALLBIT    ; Check for 8 inch disk.
1109 :     JNZ   STEP           ; Jump if large because LARGEDS is off.
1110 :     ENDIF
1111 :
1112 :     IF    LARGEDS+SMALLDS*(SCP+CROMEMCO)
1113 :     SUB   BL,8           ; Find true sector for back side.
1114 :     CMP   BL,8           ; See if ok now.
1115 :     JA    STEP           ; Have to step if still too big.
1116 :
1117 :     IF    SCP+TARBELLDD
1118 :     MOV   AL,AH          ; Move drive select byte into AL.
```

```
1119 :     OR    AL,BACKBIT      ; Select back side.
1120 :     OUT   DISK+4
1121 :     ENDIF
1122 :
1123 :     IF    CROMEMCO16FDC
1124 :     MOV   AL,BACKBIT      ; Select back side.
1125 :     OUT   04H
1126 :     ENDIF
1127 :
1128 :     JP    PUTSEC
1129 :     ENDIF
1130 :
1131 :     ENDIF
1132 :
1133 :     IF    SCP
1134 : CHECKSMALL:
1135 :     TEST  AH,SMALLBIT     ; See if big disk.
1136 :     JZ    CHECK26         ; Jump if big.
1137 :     ENDIF
1138 :
1139 :     IF    CROMEMCO
1140 : CHECKSMALL:
1141 :     TEST  AH,SMALLBIT     ; See if big disk.
1142 :     JNZ   CHECK26         ; Jump if big.
1143 :     ENDIF
1144 :
1145 :     IF    SCP+CROMEMCO
1146 :     CMP   BL,18           ; See if legal small SD/SS sector.
1147 :     JA    STEP            ; Jump if not.
1148 :     ENDIF
1149 :
1150 : CHECK26:
1151 :     CMP   BL,26           ; See if legal large SD/SS sector.
1152 :     JBE   PUTSEC          ; Jump if ok.
1153 : STEP:
1154 :     INC   DL              ; Increment track number.
1155 :     MOV   AL,58H          ; Step in with update.
1156 :     CALL  DCOM
1157 :     SEG   CS
1158 :     INC   B,[DI]          ; Increment the track pointer.
1159 :     MOV   DH,1            ; After step, do first sector.
1160 :     MOV   BL,DH           ; Fix temporary sector number also.
1161 : PUTSEC:
1162 :     MOV   AL,BL           ; Output sector number to controller.
1163 :     OUT   DISK+2
1164 :     DI                    ; Interrupts not allowed until I/O done
1165 :
1166 :     IF    SCP+CROMEMCO
1167 :     INB   DISK+4          ; Get head-load bit.
1168 :     ENDIF
1169 :
1170 :     IF    TARBELL
1171 :     INB   DISK
1172 :     ENDIF
1173 :
1174 :     NOT   AL
1175 :     AND   AL,20H          ; Check head load status
1176 :     JZ    RET
1177 :     MOV   AL,4
1178 :     RET
```

Le code dépend du contrôleur de lecteur de disquette. Par exemple pour un SCP :

— On sauvegarde le nombre de secteurs, entré en paramètre *via* `DH`, dans `BL` (ligne 1061).
— S'il ne s'agit pas d'une disquette double densité, on vérifie la taille de la disquette. S'il s'agit d'une disquette de grande capacité et si le nombre de secteurs est inférier à 26, on incrémente le numéro de piste, on appelle la sous-routine DCOM pour placer la tête de lecture-écriture, on incrémente le pointeur de piste, on initialise `DH` à 1 pour se placer sur le premier secteur, ainsi que `BL`. Dans tous les cas on spécifie ce numéro de secteur au contrôleur du lecteur de disquette, on inhibe les interruptions masquables, on place le bit de déplacement de la tête sur le registre du contrôleur et on vérifie que le bit est bien positionné dans `AL` (lignes 1063–1066, 1133–1137, 1150–1164, 1166–1168 et 1174–1178).
— Sinon on replace le contenu de `AH` dans `AL` et on le communique au registre du contrôleur de lecteur de disquette. Si `BL` est inférieur à 8, on spécifie ce numéro de secteur au contrôleur de lecteur de disquette, on inhibe les interruptions masquables, on place le bit de déplacement de la tête sur le registre du contrôleur de lecteur de disquette et on vérifie que le bit est bien positionné dans `AL` (lignes 1075–1078 et 1085–1086).

4.2.2 Pré-routine de la fonction 26h d'écriture de secteurs sur la disquette

4.2.2.1 Routine principale

La fonction d'écriture WRITE écrit un certain nombre de secteurs sur une disquette.

Avant d'appeler la routine, `AL` doit contenir le numéro du lecteur de disquette, `DS:BX` l'adresse en mémoire centrale depuis laquelle transférer ce qui doit être écrit, `CX` le nombre de secteurs à écrire, `DX` le numéro du premier secteur à écrire.

Après son exécution, le drapeau CF est baissé si le transfert s'est effectué avec succès. Sinon il est levé, `CX` contient le nombre de secteurs qui n'ont pas été transférés et `AL` contient un code d'erreur :

- `00h` si la disquette est protégée en écriture ;
- `02h` si la disquette n'est pas prête ;
- `04h` si le CRC n'est pas correct ;
- `06h` si la recherche du secteur a été infructueuse ;
- `08h` si le secteur n'a pas été trouvé ;
- 10 s'il y a eu une faute d'écriture ;
- 12 pour aucune des erreurs ci-dessus :

```
852 : ;
853 : ; Disk write function.
854 : ; Registers same on entry and exit as read above.
855 : ;
856 : WRITE:
857 :       CALL  SEEK      ;Position head
858 :       JC    ERROR
859 : WRTLP:
860 :       CALL  WRITESECT ;Perform sector write
861 :       JC    ERROR
862 :       INC   DH        ;Bump sector counter
863 :       LOOP  WRTLP     ;Write CX sectors
864 :       CLC             ; No errors.
865 : WRITERET:
866 :       RET   L
```

- On appelle la sous-routine SEEK pour traduire le numéro logique du premier secteur en caractéristiques physiques (ligne 857).
- Si on n'y parvient pas, on va à la partie ERROR pour placer dans `AL` le double de la position dans la table des erreurs du code d'erreur contenu dans `AH` (ligne 858).
- On appelle la routine WRITESECT, étudiée ci-dessous, pour écrire un secteur (ligne 860).
- Si on n'y est pas parvenu on va à ERROR (ligne 861).
- Sinon on incrémente `DH` et on recommence pour écrire le secteur suivant (lignes 862–863).
- Lorsqu'on est arrivé à écrire tous les secteurs, on baisse l'indicateur des retenues pour indiquer qu'il n'y a pas eu d'erreur et on termine la routine (lignes 864–866).

4.2.2.2 Sous-routine d'écriture d'un secteur

L'écriture d'un secteur est effectué par la sous-routine WRITESECT :

```
1248 : WRITESECT:
1249 :     CALL  SETUP
1250 :     MOV   BL,10
1251 :     PUSH  DX         ; Save track & sector number.
1252 :     MOV   DL,DISK+3  ; Disk controller data port.
1253 : WRTAGN:
1254 :     OR    AL,WRITECOM
1255 :     OUT   DISK
1256 :
1257 :     IF    CROMEMCO
1258 :     MOV   AL,AH      ; Turn on auto-wait.
1259 :     OUT   DISK+4
1260 :     ENDIF
1261 :
1262 :     MOV   BP,SI
1263 : WRLOOP:
1264 :
1265 :     IF    SCP
1266 :     INB   DISK+5
1267 :     ENDIF
1268 :
1269 :     IF    TARBELL+CROMEMCO
1270 :     INB   DISK+4
1271 :     ENDIF
1272 :
1273 :     IF    SCP+CROMEMCO
1274 :     SHR   AL
1275 :     LODB            ; Get data from memory.
1276 :     OUTB  DX        ; Write to disk.
1277 :     JNC   WRLOOP
1278 :     ENDIF
1279 :
1280 :     IF    TARBELL
1281 :     SHL   AL
1282 :     LODB            ; Get data from memory.
1283 :     OUTB  DX        ; Write to disk.
1284 :     JC    WRLOOP
1285 :     ENDIF
1286 :
1287 :     EI              ; Interrupts OK now.
1288 :     DEC   SI
1289 :     CALL  GETSTAT
1290 :     AND   AL,0FCH
1291 :     JZ    WRPOP
1292 :     MOV   SI,BP
1293 :     MOV   BH,AL
1294 :     MOV   AL,0
1295 :     DEC   BL
1296 :     JNZ   WRTAGN
1297 :     MOV   AH,BH      ; Error status to AH.
1298 :     STC
1299 : WRPOP:
1300 :     POP   DX         ; Get back track & sector number.
1301 :
1302 :     IF    TARBELL
1303 :     JMPS  FORCINT
1304 :     ENDIF
1305 :
1306 :     IF    SCP+CROMEMCO
```

```
1307 :      RET
1308 :      ENDIF
```

Son code dépend du contrôleur de lecteur de disquette. Par exemple pour SCP on a :

— On appelle la sous-routine SETUP pour se préparer à écrire, on place 10 dans `BL`, le nombre d'essais à effectuer, le port des données du contrôleur du lecteur de disquette dans `DL`, on lève le bit d'écriture et on transmet l'octet d'état au contrôleur du lecteur de disquette (lignes 1249–1255).
— On sauvegarde la valeur de `DI` dans `BP` pour sa réutilisation lors du prochain essai et on entre dans la boucle (lignes 1262–11263).
— On lit la valeur du port du contrôleur du lecteur de disquette (lignes 1265–1267).
— On divise la valeur de `AL` par 2, on lit un octet dans la zone de transfert en mémoire centrale, que l'on place sur le port des données du contrôleur du lecteur de disquette et on recommence tant que l'indicateur de retenue n'est pas levé (lignes 1273–1278 et 1263).
— On rétablit les interruptions masquables, on décrémente `SI` et on appelle la routine GETSTAT, partie de la routine DCOM, pour positionner la tête d'écriture (lignes 1287–1289).
— Si on a échoué, on réessaie : on redonne à `SI` la valeur sauvegardée, on sauvegarde le code d'erreur dans `BH`, on place 0 dans `AL`, on décroît le nombre d'essais qui restent à effectuer et, si on n'a pas atteint 0, on recommence (lignes 1290–1296).
— Si on ne réussit pas au bout de dix essais, on place le code d'erreur dans `AH` et on lève l'indicateur de retenue (lignes 1297–1298).
— Sinon on replace les numéros de piste et de secteur dans `DX` et on termine la sous-routine (lignes 1299–1300 et 1306–1308).

4.2.2.3 Fonction de détection de changement de disquette

La fonction DSKCHG dit si on a changé de disquette.

Avant d'appeler la routine, `AL` doit contenir le numéro du lecteur de disquette.

Après son exécution, `AH` contient :

— `FFh` si la disquette a été changée ;
— `00h` si on n'est pas parvenu à le savoir ;
— `01h` si elle ne l'a pas été.

S'il n'y a pas eu d'erreur, le drapeau CF est baissé et `AL` contient le numéro de lecteur de disquette.

Sinon le drapeau CF est levé et `AL` contient un code d'erreur :

— `00h` si la disquette est protégée en écriture ;
— `02h` si la disquette n'est pas prête ;
— 12 pour aucune des erreurs ci-dessus :

```
673 : ;
674 : ; Disk change function.
675 : ; On entry:
676 : ; AL = disk drive number.
677 : ; On exit:
678 : ; AH = -1 (FF hex) if disk is changed.
679 : ; AH = 0 if don't know.
680 : ; AH = 1 if not changed.
681 : ;
682 : ; CF clear if no disk error.
683 : ; AL = disk I/O driver number.
684 : ;
685 : ; CF set if disk error.
686 : ; AL = disk error code (see disk read below).
```

```
687 : ;
688 :     IF    WD1771
689 : DSKCHG:
690 :     MOV   AH,0     ; AH = 0 in case we don't know.
691 :     SEG   CS
692 :     CMP   AL,[CURDRV]
693 :     JNZ   RETL
694 :     PUSH  AX       ; Save drive number.
695 :
696 :     IF    CROMEMCO
697 :     INB   DISK+4
698 :     ENDIF
699 :
700 :     IF    TARBELL
701 :     INB   DISK
702 :     ENDIF
703 :
704 :     AND   AL,20H   ; Look at head load bit
705 :     POP   AX
706 :     JZ    RETL
707 :     MOV   AH,1     ; AH = 1, disk not changed.
708 : RETL:
709 :     CLC            ; No disk error.
710 :     RET   L
711 :     ENDIF          ; End of 1771 DSKCHG.
712 :
713 :     IF    WD1791
714 : DSKCHG:
715 :     MOV   AH,0      ; AH = 0 in case we don't know.
716 :     SEG   CS
717 :     CMP   AL,[CURDRV]
718 :     JNZ   DENSCHK   ; Check density if not same drive.
719 :     PUSH  AX
720 :
721 :     IF    SCP+CROMEMCO
722 :     INB   DISK+4
723 :     ENDIF
724 :
725 :     IF    TARBELL
726 :     INB   DISK
727 :     ENDIF
728 :
729 :     AND   AL,20H     ; Look at head load bit
730 :     POP   AX
731 :     JZ    DENSCHK    ; Check density if head not loaded.
732 :     MOV   AH,1       ; AH = 1, disk not changed.
733 :     MOV   BX,PREVDENS
734 :     SEG   CS
735 :     XLAT             ; Get previous density
736 :     CLC              ; No disk error.
737 :     RET   L
738 : DENSCHK:
739 :     CALL  CHKNEW     ; Unload head if selecting new drive.
740 :     CBW
741 :     XCHG  AX,SI
742 :     ADD   SI,PREVDENS
743 :     MOV   CX,4       ; Try each density twice
744 :     MOV   AH,0       ; Disk may not have been changed.
745 : CHKDENS:
746 :     SEG   CS
747 :     MOV   AL,[SI]    ; Get previous disk I/O driver number.
748 :     MOV   BX,DRVTAB
```

```
749 :     SEG   CS
750 :     XLAT            ; Get drive select byte for previous density
751 :
752 :     IF    CROMEMCO16FDC
753 :     CALL  MOTOR       ; Wait for motor to come up to speed.
754 :     ENDIF
755 :
756 :     OUT   DISK+4      ; Select disk
757 :     MOV   AL,0C4H     ; READ ADDRESS command
758 :     CALL  DCOM
759 :     AND   AL,98H
760 :     IN    DISK+3      ; Eat last byte to reset DRQ
761 :     JZ    HAVDENS     ; Jump if no error in reading address.
762 :     NOT   AH          ; AH = -1 (disk changed) if new density works.
763 :     SEG   CS
764 :     XOR   B,[SI],1    ; Try other density
765 :     LOOP  CHKDENS
766 :     MOV   AX,2        ; Couldn't read disk at all, AH = 0 for don't
767 :     STC               ;  know if disk changed, AL = error code 2 -
768 :     RET   L           ;  disk not ready, carry set to indicate error.
769 :
770 : HAVDENS:
771 :     SEG   CS
772 :     LODSB             ; AL = disk I/O driver number.
773 :     CLC               ; No disk error.
774 :     RET   L
775 :
776 : PREVDENS:DB    1,3,5,7,9,11,13  ; Table of previous disk I/O driver numbers.
777 :     ENDIF             ; End of 1793 DSKCHG function.
778 :
779 : CHKNEW:
780 :     MOV   AH,AL       ; Save disk drive number in AH.
781 :     SEG   CS          ; AL = previous disk drive number,
782 :     XCHG  AL,[CURDRV] ;  make new drive current.
783 :     CMP   AL,AH       ; Changing drives?
784 :     JZ    RET
785 : ;
786 : ; If changing drives, unload head so the head load delay one-shot will
787 : ; fire again. Do it by seeking to the same track with the H bit reset.
788 : ;
789 :     IN    DISK+1      ; Get current track number
790 :     OUT   DISK+3      ; Make it the track to seek to
791 :     MOV   AL,10H      ; Seek and unload head
792 :     CALL   DCOM
793 :     MOV   AL,AH       ; Restore current drive number
794 :     RET
```

Le code dépend du contrôleur de lecteur de disquette. Par exemple pour un SCP :

— On place 0 dans `AH` puisque, pour l'instant, on ne sait pas si la disquette a été changée. On compare le numéro de lecteur de disquette passé en paramètre avec celui en cours (lignes 714–717).

— S'il s'agit du même, on sauvegarde la valeur de `AX`, donc essentiellement du lecteur de disquette, sur la pile, on lit l'octet d'état sur le registre du contrôleur de lecteur de disquette, on n'en garde que le bit de chargement de la tête et on restaure la valeur de `AX` (lignes 719–723 et 729–731).

— Si la position de la tête de lecture-écriture n'a pas été changée, on en déduit qu'on n'a pas changé de disquette. On place donc 1 dans `AH`, on fait pointer `BX` sur la table des numéros des lecteurs de disquette précédents, on indique le numéro, on baisse l'indicateur de retenue puisqu'il n'y a pas d'erreur et on termine la routine (lignes 731–737).

— Sinon on appelle la sous-routine CHKNEW pour voir si le numéro de lecteur de diquette transmis en paramètre, *via* le registre `AL`, est celui du lecteur de disquette par défaut, levant le drapeau ZF si c'est le cas et, sinon, pour récupèrer le numéro de piste en cours et déplacer la tête de lecture-écriture du lecteur de disquette (lignes 738–739).
— On place le numéro de lecteur de disquette dans `AX`, on l'échange avec la valeur actuelle de `SI`, qu'on ajoute à l'adresse du début de la la table des numéros des lecteurs de disquette précédents, on place 4 dans `CX`, nombre d'essais à effectuer et 0 dans `AH` car, pour l'instant, on ne sait pas si on a changé de disquette (lignes 740–744).
— On place le numéro du lecteur de disquette précédent dans `AL`, on fait pointer `BX` sur la table DRVTAB, on récupère l'octet de sélection dans `AL` et on l'envoie au contrôleur de lecteur de disquette pour sélectionner le lecteur de disquette (lignes 745–750 et 756).

  Un contrôleur de lecteur de disquette contrôlant plusieurs lecteurs de disquette, DRVTAB est la table de l'octet envoyé au contrôleur de lecteur de disquette pour choisir le lecteur de disquette, dire s'il s'agit d'une disquette 8 ou 5,25 pouces, simple ou double densité. Elle dépend du contôleur. On a, par exemple, pour SCP :

```
1462 : ; DRVTAB is a table of bytes which are sent to the disk controller as drive-
1463 : ; select bytes to choose which physical drive is selected for each disk I/O
1464 : ; driver.  It also selects whether the disk is 5.25-inch or 8-inch, single-
1465 : ; density or double-density.  Always select side 0 in the drive-select byte if
1466 : ; a side-select bit is available.  There should be one entry in the DRVTAB
1467 : ; table for each disk I/O driver.  Exactly which bits in the drive-select byte
1468 : ; do what depends on which disk controller is used.
1469 : ;
[...]
1484 :  IF SCP*COMBIN*FASTSEEK
1485 : ;
1486 : ; A PerSci 277 or 299 and one 5.25-inch drive.
1487 : ;
1488 : DRVTAB:    DB    00H,08H,01H,09H,10H,18H,00H,08H,01H,09H
```

— On place l'octet de commande `C4h` dans `AL`, on appelle la routine DCOM pour déplacer la tête d'écriture, on en baisse le dernier bit et on lit le registre d'état (lignes 757–760).
— S'il y a eu une erreur, on place `FFh` pour indiquer que la disquette a été changée et on essaie une autre densité de disquette (lignes 762–765 et 745).
— Si on échoué dix fois, on a zéro dans `AH` car on ne sait pas si on a changé de disquette, le code d'erreur `02h` dans `AX`, l'indicateur de retenue CF est levé et on termine la routine (lignes 766-768).
— Si on réussit, on récupère le numéro de lecteur de disquette dans `AL`, on baisse l'indicateur de retenue pour indiquer qu'il n'y a pas d'erreur et on termine la routine (lignes 761 et 770–774).

4.3 Implémentation des pré-routines de gestion de la date et de l'heure

4.3.1 Pré-routine de la fonction 2Bh d'initialisation de la date

La pré-routine de la fonction `2Bh` de l'interruption `21h` d'initialisation de la date est repérée par l'étiquette SETDATE :

```
340 : SETDATE:
341 :     XCHG  AX,DX         ;Put date in DX
342 :     MOV   AL,0BH        ;Select Counter 3 load register
343 :     OUT   STCCOM
344 :     XCHG  AX,DX
345 :     OUT   STCDATA
346 :     MOV   AL,AH
347 :     OUT   STCDATA
348 :     MOV   AL,44H        ;Load counter 3
349 :     OUT   STCCOM
350 : POINTSTAT:
351 :     PUSH  AX
352 :     MOV   AL,1FH        ;Point to status register
353 :     OUT   STCCOM        ;   so power-off glitches won't hurt
354 :     POP   AX
355 :     RET   L
```

— On place la date passée en paramètre dans `DX` et l'octet `0Bh` dans `AL`, que l'on transmet au registre du temporisateur (lignes 341–343).
— On replace la date passée en paramètre dans `AX` et on en transmet l'octet de poids faible au registre des données du temporistaeur (lignes 344–345).
— On lui transmet ensuite l'octet de poids fort (lignes 346–347).
— On spécifie au compteur 3 de démarrer (lignes 348–249).
— On sauvegarde la date passée en paramètre sur la pile, on passe l'octet de commande `1Fh` au registre du temporisateur pour que les problèmes d'alimentation ne perturbent pas le décompte, on restaure la valeur de `AX` et on termine la routine (lignes 352–355).

4.3.2 Pré-routine de la fonction 2Dh d'initialisation de l'heure

4.3.2.1 Routine principale

La pré-routine de la fonction `2Dh` de l'interruption `21h` d'initialisation de l'heure est repérée par l'étiquette SETTIME :

```
301 : SETTIME:
302 :     PUSH  CX
303 :     PUSH  DX
304 :     CALL  LOAD0       ;Put 0 into load registers to condition timer
305 :     MOV   AL,43H      ;Load counters 1 & 2
306 :     OUT   STCCOM
307 :     POP   DX
308 :     POP   CX
309 :     CALL  LOAD
310 :     MOV   AL,43H
311 :     OUT   STCCOM      ;Load counters 1&2
312 :     CALL   LOAD0
313 :     MOV   AL,27H      ;Arm counters 1,2,3
314 :     OUT   STCCOM
315 :     JP    POINTSTAT
316 :
317 : LOAD0:
318 :     XOR   CX,CX
319 :     MOV   DX,CX
320 : LOAD:
321 :     MOV   AL,09       ;Counter 1 load register
322 :     CALL  OUTDX
323 :     MOV   AL,0AH      ;Counter 2 load register
324 :     MOV   DX,CX
325 : OUTDX:
326 :     OUT   STCCOM      ;Select a load register
327 :     MOV   AL,DL
328 :     CALL  OUTBCD
329 :     MOV   AL,DH
330 : OUTBCD:
331 :     AAM               ;Convert binary to unpacked BCD
332 :     SHL   AH
333 :     SHL   AH
334 :     SHL   AH
335 :     SHL   AH
336 :     OR    AL,AH       ;Packed BCD
337 :     OUT   STCDATA
338 :     RET
```

— On sauvegarde les contenus de `CX` et de `DX` sur la pile et on place 0 dans le registre de chargement du minuteur (lignes 302–304).

Pour cela :

— on place 0 dans `CX` et dans `DX`, on envoie `09h` au registre du minuteur, on convertit 0 de binaire en BCD non compacté, puis en BCD compacté et on le transmet au registre des données du minuteur (lignes 317–322, 325–328 et 330–338) ;

— on envoie `0Ah` au registre du minuteur, on convertit 0 de binaire en BCD non compacté, puis en BCD compacté et on le transmet au registre des données du minuteur (lignes 323–328 et 330–338).

— On envoie `43h` au registre du minuteur, on restaure les valeurs de `DX` et `CX` et on appelle la sous-routine LOAD, étudiée ci-dessous, pour transmettre les valeurs de `DX` et `CX`, sous forme BCD compacté aux registres 1 et 2 du minuteur (lignes 305–309, 320–322, 325–328, 330–338, 323–328 et 330–338).

— On envoie `43h` au registre du minuteur, on place 0 dans `CX` et `DX` et on transmet les valeurs de `DX` et `CX`, sous forme BCD compacté aux registres 1 et 2 du minuteur (lignes 310–312, 317–322, 325–328, 330–338, 323–328 et 330–338).
— On envoie `27h` au registre du minuteur, on sauvegarde la valeur de `AX`, on envoie `1Fh` au registre du minuteur, on restaure la valeur de `AX` et on termine la routine (lignes 313–315).

La partie POINTSTAT de la routine SETDATE a été étudiée ci-dessus.

4.3.2.2 Sous-routine de chargement

La routine de chargement est LOAD :

```
317 : LOAD0:
318 :     XOR    CX,CX
319 :     MOV    DX,CX
320 : LOAD:
321 :     MOV    AL,09     ;Counter 1 load register
322 :     CALL   OUTDX
323 :     MOV    AL,0AH    ;Counter 2 load register
324 :     MOV    DX,CX
325 : OUTDX:
326 :     OUT    STCCOM    ;Select a load register
327 :     MOV    AL,DL
328 :     CALL   OUTBCD
329 :     MOV    AL,DH
330 : OUTBCD:
331 :     AAM              ;Convert binary to unpacked BCD
332 :     SHL    AH
333 :     SHL    AH
334 :     SHL    AH
335 :     SHL    AH
336 :     OR     AL,AH     ;Packed BCD
337 :     OUT    STCDATA
338 :     RET
```

— On place 0 dans les registres `CX` et `DX` (lignes 318 et 319).
— On place 9 dans le registre `AL` que l'on envoie sur le port de commande du minuteur, on place 0 dans `AL`, que l'on convertit en BCD puis en BCD compacté, que l'on envoie sur le port des données du minuteur (lignes 321-322 et 325–338).
— On place 10 dans le registre `AL` 0 dans `DX`, que l'on envoie sur le port de commande du minuteur, on place 0 dans `AL`, que l'on convertit en BCD puis en BCD compacté, que l'on envoie sur le port des données du minuteur (lignes 323–338).

4.3.3 Pré-routine de la fonction 2Ch de récupération de l'heure

La pré-routine de la fonction `2Ch` de l'interruption `21h` de récupération de l'heure est repérée par l'étiquette GETTIME :

```
265 : ;
266 : ; ************ Time and Date ************
267 : ;
268 : GETTIME:
269 :     MOV   AL,0A7H          ;Save counters 1,2,3
270 :     OUT   STCCOM
271 :     MOV   AL,0E0H          ;Enable data pointer sequencing
272 :     OUT   STCCOM
273 :     MOV   AL,19H           ;Select hold 1 / hold cycle
274 :     OUT   STCCOM
275 :     CALL  STCTIME          ;Get seconds & 1/100's
276 :     XCHG  AX,DX
277 :     CALL  STCTIME          ;Get hours & minutes
278 :     XCHG  AX,CX
279 :     IN    STCDATA
280 :     MOV   AH,AL
281 :     IN    STCDATA
282 :     XCHG  AL,AH            ;Count of days
283 :     JP    POINTSTAT
284 :
285 : STCTIME:
286 :     CALL  STCBYTE
287 :     MOV   CL,AH
288 : STCBYTE:
289 :     IN    STCDATA
290 :     MOV   AH,AL
291 :     SHR   AH
292 :     SHR   AH
293 :     SHR   AH
294 :     SHR   AH
295 :     AND   AL,0FH           ;Unpack BCD digits
296 :     AAD                    ;Convert to binary
297 :     MOV   AH,AL
298 :     MOV   AL,CL
299 :     RET
```

— On envoie `A7h` au registre du minuteur pour sauvegarder les compteurs 1, 2 et 3, on envoie `E0h` au même registre pour démarrer le cadencement et on envoie `19h` au même registre pour maintenir le cycle du compteur 1 (lignes 269–274).
— On récupère les secondes et les centièmes de seconde dans `DX`. Pour cela, on lit l'octet de données du minuteur, que l'on place dans `AH`, que l'on divise par 16, on ne garde que les 4 bits de poids forts de `AL`, ce qui donne deux chiffres BCD dans `AX`, que l'on convertit en binaire ; on place le contenu de `AL` dans `AH` et celui de `CL` dans `AL` ; on place le contenu de `AH` dans `CL` et on recommence. On sauvegarde le résultat obtenu dans `DX` (lignes 275, 285–286, 288-299, 287–299, 287–299 et 276).
— On récupère de façon analogue l'heure et la minute, que l'on place dans `CX` (lignes 277, 286, 288–299, 287–299 et 278).
— On lit les deux octets de données suivants du minuteur, ce qui donne dans `AX` le nombre de jours écoulés depuis le 1^{er} janvier 1980, on sauvegarde la valeur de `AX`, on envoie `1Fh` au registre du minuteur, on restaure la valeur de `AX` et on termine la routine (lignes 279–283).

4.4 Implémentations des fonctions FLUSH et MAPDEV

La fonction FLUSH ne fait rien puisque sa routine de service :

```
395 : FLUSH:
396 :     RET   L
```

se contente de terminer la routine.

Il en est de même de la fonction MAPDEV :

```
127 : MAPDEV:
128 :     RET   L
```