Reijo Lundahl

1541-LEVYASEMA

Commodore 64

AMERSOFT

Commodore 64
1541-LEVYASEMA

Reijo Lundahl

Commodore 64

1541-LEVYASEMA

COMMODOR

Reijo Lundahl

AMERSOFT

© Reijo Lundahl

Kannen kuvat Kuvapörssi ja Pf-studio

© Amer-yhtymä Oy AMERSOFT

Amer-yhtymä Oy Weilin+Göösin kirjapaino
Espoo 1986

ISBN 951-35-3707-7

SISÄLLYS

ESIPUHE 8

1 ENSIMMÄISET ASKELEET

Levykkeen alustaminen 9
Levykkeen sisällysluettelo 11
Ohjelman tallentaminen levykkeelle 14
Ohjelman lataaminen tietokoneen muistiin 16
Tallennuksen ja latauksen tarkistaminen 17
Levykkeen käsittelyohjeita 18

2 KOMENTOKANAVA

Levyaseman käyttöjärjestelmä 21
Komentokanava 21
Virheilmoituksen lukeminen 26

3 LEVYKETIETOJEN PÄIVITTÄMINEN

Levykkeen alustaminen – NEW 31
Levykkeen initialisointi – INITIALIZE 34
Tiedoston nimeäminen uudelleen – RENAME 35
Tiedoston tuhoaminen – SCRATCH 36
Tiedoston kopioiminen – COPY 38
VALIDATE-komento 40
Jokereiden käyttö 41

4 TIEDOSTOJEN KÄSITTELY

Eri tiedostotyypit 46
Tiedostojen käsittelyn päävaiheet 46
Tiedostojen avaaminen 47
  Looginen tiedostonumero 47
  Laitenumero 48
  Kanavanumero 49
  Tiedoston tunnusosa 49
Tiedostoon kirjoittaminen 52

Tiedoston lukeminen 53
Tiedoston lopun selvittäminen 55
Tiedoston sulkeminen 56

5 OHJELMATIEDOSTOT

Ohjelmatiedoston rakenne 57
Ohjelman lataus 59
Ohjelman tallennus 62
Korvaava tallennus 66
Ohjelmatiedoston kirjoittaminen ja lukeminen 67

6 PERÄKKÄISTIEDOSTOT

Johdanto 73
Peräkkäistiedostojen käsittely 75
Peräkkäistiedoston avaaminen 76
Kirjoittaminen peräkkäistiedostoon 77
Peräkkäistiedoston lukeminen 79
Peräkkäistiedoston sulkeminen 84
Tietueiden lisääminen APPEND-tilan avulla 85
Uuden tietueen sijoittaminen 86
Tiedoston korvaaminen 92
MODIFY-tila 93
Ohjelmatiedosto peräkkäistiedostoksi 95

7 SUHTEELLISET TIEDOSTOT

Suhteellisen tiedoston rakenne 100
Suhteellisen tiedoston avaus 101
P-komennon käyttö 104
Tietueen kirjoittaminen 105
Suhteellisen tiedoston lukeminen 108
Suhteellisen tiedoston sulkeminen 111
Indeksoidut suhteelliset tiedostot 111

8 SUORASAANTIKÄSKYT

Miten 1541 työskentelee? 115
Suorasaannin valmistelut 117
BLOCK-READ tai USER1 120
BUFFER-POINTER 123
BLOCK-WRITE tai USER2 124
MEMORY-READ 127
MEMORY-WRITE 129

MEMORY-EXECUTE 131
BLOCK-ALLOCATE 132
BLOCK-FREE 135
BLOCK-EXECUTE 136
USER-käskyt 137
Lisää teoriaa 139
Käyttäjän &-tiedostot 142

9 LEVYKKEEN SISÄINEN RAKENNE

Urat ja sektorit 145
Sektorin rakenne 146
Lohkokartta eli BAM 149
Sisällysluettelo 152
Ohjelmatiedoston rakenne 157
Peräkkäistiedoston rakenne 161
Suhteellisten tiedostojen rakenne 164
Muiden tiedostotyyppien rakenne 171

LIITE 1

1541-MUISTIKARTTA 173

LIITE 2

VIRHEILMOITUKSET 181

LIITE 3

TAULUKKOJA
Commodoren tiedostotyypit 189
Muunnostaulukko heksadesimaali/desimaali 190
Urille mahtuvien sektoreiden lukumäärä 190
Sisällysluettelon sektoreiden sisältö 190

LIITE 4

DOS 5.1 193

LIITE 5

LEVYKKEEN OHJELMAT 201

HAKEMISTO 205

ESIPUHE

Commodore 64 on kolmivuotisen elämänsä aikana saanut runsaasti ystäviä. Menestys johtuu pääasiassa siitä laajasta ohjelma- ja kirjallisuusvalikoimasta, jolla tätä konetta ja sen käyttäjiä on tuettu. Tietokoneen tehokas ja monipuolinen käyttö vaatii hyvät ja kohtuuhintaiset oheislaitteet. Koska useimmat hyötyohjelmat ja monet pelitkin ovat nykyään levykkeellä, on levyaseman hankinta usein ensimmäinen askel kohti monipuolisempaa tietokoneen käyttöä.

Levyaseman käytön tukena on aluksi vain aseman mukana tuleva käsikirja, joka on suppea eikä vastaa kaikkiin esille tuleviin kysymyksiin. Sen vuoksi levyasemaa ei useinkaan käytetä muuhun kuin ohjelmien lataamiseen ja tallentamiseen. Tämän kirjan tarkoituksena on laajentaa ja täydentää varsinaista käsikirjaa.

Kirja sopii sekä aloitteleville että pitemmälle ehtineille levyaseman omistajille. Kirjassa esitettyjen asioiden ja kirjan mukana olevien ohjelmien avulla voi jokainen oppia 1541-levyaseman tehokkaan käytön. Sen sijaan kirjassa ei käsitellä ohjelmien suojausta eikä suojauksien purkamista. Näin kirjan koko on pysynyt kohtuullisena.

Kiitän Jarmo Saajosta BACKUP-ohjelman luovuttamisesta käytettäväksi kirjan levykkeellä. Suuren kiitoksen on ansainnut myös AMERSOFT Oy:n henkilökunta minulle antamastaan mahdollisuudesta kirjan tekemiseen sekä kaikesta siitä tuesta, jota olen heiltä kirjan valmistusvaiheessa saanut.

Hämeenlinnassa, kesällä 1985

Reijo Lundahl

1. LUKU

ENSIMMÄISET ASKELEET

Aloitat kytkemällä levyaseman ja tietokoneen yhteen levyaseman käsikirjassa esitetyllä tavalla ja liität molemmat laitteet verkkoon. Virran kytkeminen laitteisiin on paras suorittaa kokoonpanosta riippuen seuraavasti:

C64 – levyasema tai
C64 – levyasema – kirjoitin tai
C64 – levyasema 1 – levyasema 2 tai
C64 – levyasema 1 – levyasema 2 – kirjoitin

LEVYKKEEN ALUSTAMINEN

Ensin alustamme uuden levykkeen. Kaupasta ostettu levyke ei sellaisenaan sovi käytettäväksi 1541-levyasemassa, vaan se on ensin formatoitava eli alustettava. Alustuksessa levykkeen pintaan merkitään ohjelmien tallentamista ja lataamista helpottavat urat ja sektorit sekä levykkeen sisällysluettelo. Tarkastelemme näitä asioita yksityiskohtaisemmin kolmannessa luvussa.

Alustus suoritetaan seuraavasti:

1) Kytke virta ja sijoita tyhjä levyke asemaan siten, että levykkeen etiketti on ylöspäin ja levykkeen reunassa oleva kolo on vasemmalla. Sulje aseman luukku.

2) Kirjoita seuraava komento:

Syntaksi:

```
OPEN 15,8,15,"N0:NIMI,ID"  <RETURN>
```

tai

```
OPEN 15,8,15  <RETURN>

PRINT#15,"N0:NIMI,ID"  <RETURN>
```

Esimerkkejä:

```
1) OPEN 15,8,15,"N0:OHJELMAT-1,01"

2) OPEN 15,8,15

   PRINT#15,"N0:LEVYKE 1,A1"

3) OPEN 2,8,15

   PRINT#2,"N0:APUOHJELMIA,XY"
```

Kuva 1.1

NIMI = levykkeelle annettava nimi. Nimessä voi olla korkeintaan 16 merkkiä. Siinä voidaan käyttää kirjaimia, numeroita, välilyöntejä ja grafiikkamerkkejä. Kertomerkki (*), kysymysmerkki (?), pilkku (,) ja kaksoispiste (:) ovat kiellettyjä merkkejä.

ID = levykkeen tunnuskoodi, joka muodostetaan kahdesta merkistä.

Käytä eri levykkeillä erilaisia tunnuskoodeja. Suurta levykemäärää on silloin helpompi hallita. Samanaiheiset levykkeet voidaan myös merkitä yhtenäisesti. Esimerkiksi pelilevykkeiden tunnuskoodeiksi sopivat P1, P2, P3 jne.

Komennon kirjoitettuasi painat RETURN-näppäintä, jolloin levyaseman punainen toimintavalo syttyy ja asemasta kuuluu pientä kolinaa. Alustus kestää noin 80 sekuntia, minkä jälkeen punainen valo sammuu ja levyasema lakkaa pyörimästä. Jos punainen valo jää palamaan levyaseman pysähtyessä, alustaminen ei ole onnistunut ja toimenpiteet on suoritettava uudestaan. Jos alustaminen ei onnistu useankaan yrityksen jälkeen, levyke on virheellinen.

3) Sulje komentokanava kirjoittamalla:

Syntaksi:

```
CLOSE 15  <RETURN>
```

Esimerkkejä:

1) CLOSE 15

2) Edellisen kohdan esimerkin 3) jälkeen suoritetaan sulkeminen komennolla:

```
CLOSE 2
```

Kuva 1.2

LEVYKKEEN SISÄLLYSLUETTELO

Levykkeen jokaiselle ohjelmalle on annettava nimi. Käyttöjärjestelmä tallentaa ohjelman tai tiedoston nimen, koon ja tyypin sisällysluetteloon, joka muodostetaan alustuksessa. Seuraavaksi tutustumme sisällysluetteloon kirjaan kuuluvan levykkeen avulla. Toimi seuraavasti:

1) Aseta kirjan mukana tuleva levyke levyasemaan.

2) Lataa sisällysluettelo tietokoneen muistiin komennolla LOAD "$",8 tai LOAD "$0",8
Paina RETURN.

3) Kun punainen valo sammuu ja päätteelle tulostuu teksti READY, kirjoitat komennon LIST ja painat RETURN-näppäintä.

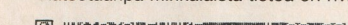
Päätteelle tulostuu sisällysluettelo:

```
0  [illegible]
9     "BACKUP"              PRG
8     "FIILI"               PRG
5     "TUTKI U&S"           PRG
6     "BAMBAM"              PRG
8     "SUPERDIR"            PRG
4     "MUUTA NIMI & ID"     PRG
4     "TARKASTA ID"         PRG
3     "TARKASTA LEVYKE"     PRG
3     "TARKASTA URA"        PRG
3     "TUTKI DRIVE"         PRG
```

```
5     "PELASTA"          PRG
5     "LUKITSE"          PRG
4     "VARAA"            PRG
5     "SULKEMATON"       PRG
1     "SAVEML"           PRG
2     "LUOML"            PRG
2     "SISLU 1"          PRG
1     "SISLU 2"          PRG
3     "LUO INDEKSI.1"    PRG
2     "LUO INDEKSI.2"    PRG
11    "REL.TESTI"        REL
2     "SEQ.TESTI"        SEQ
568 BLOCKS FREE.
```

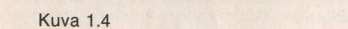
Kuva 1.3

Katsotaanpa minkälaista tietoa eri rivit sisältävät.



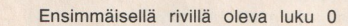
Kuva 1.4

Ensimmäisellä rivillä oleva luku 0 ilmoittaa levyaseman numeron. 1541-levyasemalla se pysyy aina samana.



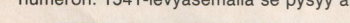
Kuva 1.5

Seuraavana on levykkeelle alustuksessa annettu nimi. Nimen perään on lisätty välilyöntejä niin, että sen pituus on tasan 16 merkkiä.



Kuva 1.6

Tämä on levykkeen ID-numero, joka annetaan alustuskomennon yhteydessä.



Kuva 1.7

Viimeisenä on levykkeen formaatti. Se on 1541-levyasemaa varten alustetuilla levykkeillä aina 2A.

```
568 BLOCKS FREE.
```

Kuva 1.8

Sisällysluettelon viimeinen rivi ilmoittaa levykkeen vapaiden lohkojen (sektoreiden) lukumäärän. Jokainen lohko voi sisältää 256 merkkiä. Koska tyhjällä alustetulla levykkeellä on 664 lohkoa, niin levykkeelle voidaan tallentaa kaikkiaan 664 * 256 = 169984 merkkiä (169,984 kilotavua).

Ensimmäisen ja viimeisen rivin välissä olevat rivit sisältävät eri ohjelmien tiedot.

```
9      "BACKUP"                PRG
```

Kuva 1.9

Ensimmäisenä olevan ohjelman tiedot näyttävät tällaisilta.

```
[9]    "BACKUP"                PRG
```

Kuva 1.10

Ensiksi on merkitty ohjelman (tiedoston) koko lohkoina. Kun tämä luku jaetaan neljällä, saadaan likimain ohjelman koko kilotavuina. Kun luku kerrotaan 256:lla, saadaan arvio ohjelman koosta bitteinä.

```
9      ["BACKUP"]              PRG
```

Kuva 1.11

Seuraavaksi on ilmoitettu ohjelman tai tiedoston nimi, jonka pituus saa olla korkeintaan 16 merkkiä.

```
9      "BACKUP"                [PRG]
```

Kuva 1.12

Viimeiseksi on merkitty tiedoston tyyppi.

Tavallisimmat tyypit ovat:

Ohjelmatiedosto	PRG
Peräkkäistiedosto	SEQ
Suhteellinen tiedosto	REL
Käyttäjän tiedosto	USR

Tiedostotyyppeihin palaamme neljännessä luvussa.

Jos et onnistu lataamaan levykkeen sisällysluetteloa, sammuta virta tietokoneesta ja levyasemasta ja kytke uudestaan. Yritä alusta uudelleen! Jos lataaminen ei vieläkään onnistu, vika on joko levykkeessä tai levyasemassa. Jos et saa ladattua muunkaan levykkeen sisällysluetteloa, ota yhteyttä jälleenmyyjän huoltoon.

OHJELMAN TALLENTAMINEN LEVYKKEELLE

Kirjoitettuasi ohjelman tietokoneen muistiin, haluat tallentaa sen myöhempää käyttöä varten levykkeelle. Tavalliset BASIC-ohjelmat tallennetaan SAVE-käskyllä, ja tallennus tapahtuu tietokoneen muistissa muistipaikasta 2049 eteenpäin. SAVE-käskyllä voidaan tallentaa myös sellaiset konekieliset ohjelmat, jotka alkavat muistipaikasta 2049. Muualla tietokoneen muistissa olevien ohjelmien tallentamista käsitellään neljännessä luvussa.

SAVE-käskyn käyttö:

Syntaksi:

```
SAVE "NIMI",8

tai

SAVE N$,8

Esimerkkejä:

1) SAVE "GRAF",8

2) N$="PELI-1"

   SAVE N$,8
```

Kuva 1.13

NIMI = ohjelman nimi
N$ = merkkijonomuuttuja, jonka arvona on ohjelman nimi
8 = levyaseman laitenumero

Ohjelman nimessä saa olla korkeintaan 16 merkkiä. Nimi kannattaa valita siten, että sen avulla on helppo muistaa minkälaisesta ohjelmasta on kysymys. Nimessä ei saa olla kertomerkkiä (*), kysymysmerkkiä (?) eikä pilkkua (,). Nimen ensimmäinen merkki ei voi olla AT-merkki (@). Nimi voi sisältää grafiikkamerkkejä ja käänteisiä merkkejä, mutta niiden käyttöä tulee välttää.

Kun olet kirjoittanut SAVE-käskyn ja painanut RETURN-näppäintä, syttyy punainen toimintavalo. Tallennuksen päätyttyä valo sammuu ja tietokone ilmoittaa olevansa valmis eli READY. Jos valo jää palamaan levyaseman pysähdyttyä, tallennus ei ole onnistunut. Ohjelma on kuitenkin vielä tietokoneen muistissa ja voit yrittää tallennusta toiselle levykkeelle. Jos kohdistin ei ole näkyvissä, painat samanaikaisesti RUN/STOP ja RESTORE. Jos tämäkään ei auta, sammuta virta tietokoneesta.

Jos tallennus ei onnistu, syynä on tavallisesti alustamaton levyke, levyke on täynnä, levykkeellä on jo samanniminen ohjelma tai levykkeessä on kirjoituksenestotarra. Seuraavassa luvussa tarkastelemme komentokanavan lukemista ja miten sen avulla voit selvittää virheen alkuperän.

Kirjoita seuraava lyhyt ohjelma ja harjoittele sen avulla SAVE-käskyn käyttöä. Tyhjennä ensin tietokoneen muisti NEW-käskyllä.

```
10 REM TESTI 1
20 PRINT "TALLENNAN OHJELMAN"
30 END

SAVE "TESTI 1",8

SAVING TESTI 1
```

Kuva 1.14

Kun olet kirjoittanut SAVE-komennon, tietokone tulostaa ilmoituksen SAVING TESTI. Kun tallennus on päättynyt, saat ilmoituksen READY.

OHJELMAN LATAAMINEN TIETOKONEEN MUISTIIN

Ohjelma ladataan levykkeeltä tietokoneen muistiin LOAD-käskyllä. Oletetaan, että ohjelman alkuosoite on 2049 (tavallinen BASIC-ohjelma).

Syntaksi:

```
LOAD "NIMI",8
```

Kuva 1.15

Jos ohjelman alkuosoite on muu kuin 2049 (konekieliset ohjelmat), on LOAD-käskyä käytettävä seuraavasti:

Syntaksi:

```
LOAD "NIMI",8,1
```

tai

```
LOAD N$,8,1
```

Kuva 1.16

NIMI = ohjelman nimi
8 = levyaseman laitenumero
1 = kanavanumero (toisio-osoite)
N$ = merkkijonomuuttuja, jonka arvona on ohjelman nimi

Kun olet kirjoittanut latauskäskyn ja painanut RETURN-näppäintä, syttyy punainen toimintavalo. Kun ohjelma on ladattu tietokoneen muistiin, valo sammuu ja näyttöön tulostuu READY-ilmoitus. Jos tietokone viestii virheilmoituksen ?FILE NOT FOUND, antamasi ohjelman nimi ei ole levykkeen sisällysluettelossa (eikä ohjelma ole levykkeellä) tai olet kirjoittanut nimen väärin. Liitteeseen 2 on koottu kaikki mahdolliset virheilmoitukset selityksineen. Komentokanavan lukemista virheilmoituksen selvittämiseksi käsitellään seuraavassa luvussa.

Kun ohjelma on ladattu, voit listata sen LIST-komennolla näyttöön. Jos ohjelman ainoana rivinä on esimerkiksi 0 SYS 2045, kysymyksessä on joko käännetty BASIC-ohjelma tai

konekielinen ohjelma. Jos näyttöön tulostuu LIST-komennon jälkeen epämääräisiä merkkejä, kysymyksessä on konekielinen ohjelma, ja latauskäskystä on unohtunut toisio-osoite 1. Konekielisiä ohjelmia ei voi listata LIST-komennolla. Liikkeissä myytävät peli- ja hyötyohjelmat on lisäksi suojattu.

Useimmat peliohjelmat sisältävät automaattisen ohjelman suorituksen latauksen jälkeen (autostart), eikä RUN-käskyä tarvita ohjelman suorittamiseksi. Jos haluat säästää näppäilyjä ja suorittaa ohjelman heti latauksen jälkeen, voit toimia seuraavasti:

Kirjoita komento

LOAD "NIMI",8: tai LOAD "NIMI",8,1:

ja paina kaksoispisteen jälkeen SHIFT+RUN/STOP.

Toisio-osoitteen avulla käyttöjärjestelmä osaa ladata ohjelman siten, että se sijoittuu tietokoneen muistissa samaan paikkaan, mistä se on tallennettu. Toisio-osoitteeksi annetaan yleensä 1, mutta se voi olla mikä tahansa kokonaisluku 1–14.

TALLENNUKSEN JA LATAUKSEN TARKISTAMINEN

Kun olet tallentanut tai ladannut ohjelman, voit tarkistaa, onko tietokoneen muistissa oleva ohjelma täysin samanlainen kuin levykkeellä oleva ohjelma. Tarkistaminen suoritetaan VERIFY-käskyllä.

Syntaksi:

```
VERIFY "NIMI",8

VERIFY N$,8

tai

VERIFY "NIMI",8,1

VERIFY N$,8,1
```

Kuva 1.17

NIMI = ohjelman nimi
8 = levyaseman laitenumero
1 = kanavan numero (toisio-osoite)
N$ = merkkijonomuuttuja, jonka arvona on ohjelman nimi

Muistissa ja levykkeellä olevien ohjelmaversioiden pitää olla täysin samanlaiset, muuten kone antaa virheilmoituksen ?VERIFY ERROR.

VERIFY-komentoa ei yleensä tarvita, koska 1541-levyasema on luotettava. Jos epäilet levyaseman tai levykkeen kuntoa, voit VERIFY-käskyllä tarkistaa tallennuksen tai latauksen onnistumisen.

LEVYKKEEN KÄSITTELYOHJEITA

Virheiden syynä on usein levykkeiden huolimaton käsittely tai huono laatu. Levyn huonokuntoisuudesta johtuvien luku- ja kirjoitusvirheiden välttämiseksi olen koonnut seuraavat yleisohjeet.

1) Käytä hyvälaatuisia, yksipuolisia, yksi- tai kaksitiheyksisiä levykkeitä. Kaksipuoliset levykkeet sopivat tietysti yhtä hyvin, mutta ne ovat kalliimpia eikä kaksipuolisuudesta ole mitään hyötyä.

2) Vältä pölyn, lian ja tupakansavun joutumista levykkeen luku- ja kirjoitusalueelle.

3) Älä koske sormillasi levykkeen luku- ja kirjoitusalueeseen. Muista, että 1541-levyasema käyttää levykkeen takapuolta.

4) Säilytä levyke aina säilytyskuoressa.

5) Älä käytä levykettä liian kuumassa tai liian kylmässä tilassa. Normaali käyttölämpötila on huoneen lämpötila (n. 20 Celsius-astetta). Liika kuumuus saattaa vääristää levykkeen ja liika kylmyys voi aiheuttaa kosteuden tiivistymisen levykkeen pintaan.

6) Älä vie levykettä magneettikenttään. Televisiot, radiot ja muut sähköiset laitteet saattavat magneettikenttänsä avulla tuhota levykkeen pinnalla olevia tietoja.

7) Älä kirjoita levykkeen päällä olevaan etikettiin kuulakärkitai lyijykynällä. Kirjoita etiketti valmiiksi ennen kuin kiinnität sen levykkeeseen tai käytä erikoista etikettikynää.

8) Puhdista levyaseman luku- ja kirjoituspää säännöllisesti. Laitekauppiaalta saat tarkoitukseen sopivan puhdistuslevykkeen.

Nykyään käytetään mielellään yksipuolisen levykkeen molempia puolia. Ne saadaan käyttöön leikkaamalla levykkeen toiseen reunaan tarkistuskolo samaan kohtaan kuin toisessa reunassa. Tämä vähentää levykekustannuksia, mutta siihen liittyy myös vaaroja. Valmistaja tekee levykkeet tarkoin valitusta materiaalista. Huolellisten testausten jälkeen päätetään yksi- ja vastaavasti kaksipuolisten levykkeiden raaka-aineet. Valmistajat takaavat vain yksipuolisten levykkeiden yhden puolen laadun. Näin toinen puoli voi olla heikompilaatuinen. Mahdollinen läpikirjoittaminen on myös huomioitava. Kirjoittamistapahtumassa kirjoistuspää magnetisoituu sähkövirran vaikutuksesta. Muodostunut magneettikenttä voi olla niin voimakas, että se vaikuttaa myös levyn toiselle pinnalle kirjoitettuihin tietoihin. Lisäksi on muistettava, että varsinaisen levykeosan ulkoreuna kerää levykkeen pyöriessä pölyhiukkasia levykkeen keskiosista reunoille. Kun levyke käännetään, pyörimissuunta vaihtuu ja pöly voi joutua reunoilta keskelle. Kaksipuolisissa levyasemissa levyke pyörii aina samaan suuntaan.

2. LUKU

KOMENTOKANAVA

LEVYASEMAN KÄYTTÖJÄRJESTELMÄ

Levyaseman käyttöjärjestelmä (DOS, Disk Operating System) on konekielinen ohjelma, joka ohjaa levyasemaa. Se huolehtii levyaseman ja tietokoneen välisestä tiedonsiirrosta, hoitaa päivitystehtävät, esim. levykkeen alustamisen, tietojen kirjoittamisen levykkeen pinnalle ja tietojen lukemisen levykkeen pinnalta.

1541-levyasema on ns. älykäs oheislaite. Useimmissa tietokoneissa DOS ladataan levykkeeltä tietokoneen muistiin. Monet edellä mainituista tehtävistä toteutetaan silloin tietokoneen suorittimen avulla. Älykäs levyasema ei käytä pääkonetta tehtävien suorittamiseen. 1541-levyasemassa on 6502-mikrosuoritin, 2 K:n RAM-muisti, kaksi 6522 I/O -piiriä ja 15,8 K:n pysyvä ROM-muisti, johon DOS on tallennettu.

Levyasemaan sijoitettu DOS ei käytä tietokoneen muistia. Useat toiminnot voidaan suorittaa pääkoneen keskusyksiköstä riippumatta eikä levyaseman käyttö hidasta muita toimenpiteitä. Haittana on, että DOS-rutiineja on vaikea muuttaa käyttäjän omia tarkoituksia paremmin vastaaviksi ja DOS-version vaihtuessa ROM-piirit on vaihdettava.

KOMENTOKANAVA

Kaikki levyasemalle (käyttöjärjestelmälle) suunnatut käskyt lähetetään komentokanavan kautta. Komentokanavan numero on 15. Komentokanavan avulla voit

lukea mahdolliset virheilmoitukset;
huolehtia päivitystoimenpiteistä: levykkeen alustaminen, tiedoston poistaminen, tiedoston uudelleennimeäminen, tiedoston kopioiminen jne;
ohjata DOS:ia suoraan: tiedon kirjoittaminen määrätylle levykkeen uralle ja sektorille, tiedon lukeminen määrätystä paikasta jne.

KOMENTOKANAVAN KÄYTTÖVAIHEET

1) Komentokanavan avaaminen (OPEN)

2) Komennon lähettäminen DOS:ille (PRINT#)

3) DOS:in lähettämän ilmoituksen lukeminen (GET# tai INPUT#)

4) Komentokanavan sulkeminen (CLOSE)

Komentokanava avataan OPEN-käskyllä:

Syntaksi:

```
OPEN TN,LN,KN
```

```
Esimerkkejä:

1) OPEN 15,8,15

2) OPEN 2,8,15

3) 10 TN=5:LN=8:KN=15
   20 OPEN TN,LN,KN
```

Kuva 2.1

TN = looginen tiedostonumero (1–127)
LN = laitenumero (levyasemalle 8–11; arvo on 8, jos laitenumeroa ei ole muutettu)
KN = kanavanumero tai toisio-osoite (2–15)

Kanavanumerot 0 ja 1 on varattu DOS:in sisäiseen käyttöön, numerot 2–14 tiedonsiirtoon ja numero 15 on varattu komentokanavalle. Komentokanavaa avattaessa käytetään numeroa 15 usein myös loogisena tiedostonumerona muistamisen helpottamiseksi.

OPEN-käskyä voidaan käyttää sekä suorassa toimintatilassa (esimerkit 1 ja 2) että ohjelmointitilassa (esimerkki 3).

Jos kanavaa avatessasi saat ilmoituksen ?FILE OPEN, suljet kanavan komennolla CLOSE 15 ja suoritat avaamisen uudestaan.

Komento lähetetään PRINT#-käskyllä:

Syntaksi:

```
PRINT#TN,"KOMENTO"
```

Esimerkkejä:

```
1) PRINT#15,"N0:LEVY 1,L1"

2) PRINT#2,"I0"

3) 30 K$="N0:APU,99"
   40 PRINT#5,K$
```

Kuva 2.2

TN = looginen tiedostonumero, joka on sama kuin OPEN-käskyssä

KOMENTO = lähetettävä komento

Huomaa, että käskysanan ja #-merkin välissä ei saa olla välilyöntiä (siis EI: PRINT #). Käskyä PRINT# ei voi lyhentää ?#, vaan se on lyhennettävä P + SHIFT R.

OPEN- JA PRINT#-käskyt voidaan yhdistää seuraavasti:

```
OPEN 15,8,15,"KOMENTO"
```

Esimerkiksi:

```
OPEN 15,8,15,"N0:GAMES,GG"
```

Kuva 2.3

Kun komento on lähetetty DOS:ille, voit käyttää tietokonetta muihin tarkoituksiin. Levyasemaa ei kuitenkaan voi käyttää komennon suorittamisen aikana.

PRINT#-käskyä voi käyttää suoraan tai ohjelmassa. Tietojen lukeminen suoritetaan INPUT#- tai GET#-käskyllä.

Syntaksi:

```
INPUT#TN,ML

GET#TN,ML
```

Esimerkkejä:

```
1) 10 INPUT#15,VN,VI$,VU,VS

2) 30 GET#15,A$

3) 70 FOR I=1 TO 16
   80 GET#3,A$:PRINT A$;
   90 NEXT I
```

Kuva 2.4

TN = looginen tiedostonumero, joka on sama kuin OPEN-käskyssä
ML = yksi tai useampia muuttujia, jotka on erotettu toisistaan pilkulla

Käskysanan ja #-merkin välissä ei saa olla välilyöntiä. INPUT#-käskyn voi kirjoittaa lyhennettynä I + SHIFT N. GET#-käskyä ei voida kirjoittaa lyhennettynä ja sen avulla voidaan lukea vain yksi merkki kerrallaan. Luettavan tiedon ja muuttujan on tyypiltään vastattava toisiaan. Epävarmoissa tapauksissa on varminta käyttää merkkijonomuuttujia.

Komentokanava suljetaan CLOSE-käskyllä.

Syntaksi:

```
CLOSE TN
```

Esimerkkejä:

```
1) CLOSE 15
```

```
2) 10 TN=5
   20 CLOSE TN
```

Kuva 2.5

TN = looginen tiedostonumero, joka on sama kuin OPEN-käskyssä

Komentokanavan sulkeminen aiheuttaa KAIKKIEN avoinna olevien tiedostojen sulkeutumisen levyaseman puolella. Tämän vuoksi komentokanava on syytä avata ohjelman alussa ja sulkea viimeisenä ohjelman lopussa. Muista, että vaikka tiedostot ovat sulkeutuneet levyaseman puolella, ne ovat auki tietokoneen puolella, ellei niitä ole suljettu CLOSE-käskyllä.

Ohjelman lataaminen, tallentaminen, suorittaminen ja editoiminen sulkevat kaikki tiedostot tietokoneen puolella. Avoinna olevat tiedostot – myös komentokanava – sulkeutuvat, kun annat LOAD-, SAVE- tai RUN-käskyn. Ohjelmarivin lisääminen tai ohjelman muuttaminen sulkee komentokanavaa lukuun ottamatta kaikki kanavat. Vaikka tiedostot ovat kiinni tietokoneen puolella, ne ovat levyaseman puolella vielä auki. Tästä saattaa myöhemmin aiheutua sekaannusta.

Jos ohjelman suoritus pysähtyy ?SYNTAX ERROR -ilmoitukseen ja levyaseman punainen toimintavalo palaa, jokin tiedosto on ehkä auki. Kanava pitää sulkea ennen virheellisen kohdan korjaamista. Kanavan voi sulkea käskyllä CLOSE TN, jossa TN on tiedoston looginen tiedostonumero. Jos et muista tiedostonumeroa, voit hakea sen LIST-komennolla ohjelmasta. Voit sulkea kaikki kanavat kirjoittamalla komentorivin

```
CLOSE 15:OPEN 15,8,15:CLOSE 15
```

Jos et kiireessä huomaa, että punainen valo palaa ja korjaat virheen ennen kanavan sulkemista, tiedosto sulkeutuu tietokoneen puolella, mutta jää levyaseman puolella avoimeksi. CLOSE TN -käsky ei silloin auta, koska kanava (tiedosto) on suljettu tietokoneen puolella. Ainoana keinona on antaa edellä mainitut komennot tai käyttää C64:n muistipaikkaa 152, joka ilmoittaa avoinna olevien tiedostojen lukumäärän. Kirjoita POKE 152,1 ja sulje tiedosto tavalliseen tapaan.

VIRHEILMOITUKSEN LUKEMINEN

Komentokanavaa käytetään useimmiten virheilmoituksen lukemiseen. Lukemisen eri vaiheet ovat:

1) Komentokanavan avaaminen

2) Virheilmoituksen lukeminen

3) Virheilmoituksen tulostaminen näyttöön

4) Komentokanavan sulkeminen

Virheilmoitus muodostuu neljästä kentästä ja kunkin kentän sisältämä tieto luetaan yhden muuttujan arvoksi. Käytän seuraavia muuttujien nimiä:

VN = virheen numero (numeerinen muuttuja)
VI$ = virheen sanallinen, englanninkielinen kuvaus (merkkijonomuuttuja)
VU = levykkeen ura, jolla virhe on (numeerinen muuttuja)
VS = sektori, jolla virhe on (numeerinen muuttuja)

Kaikki muuttujat voivat olla merkkijonomuuttujia, esim. VN$, VI$, VU$ ja VS$. Toisena oleva, VI$ ei kuitenkaan voi olla numeerinen muuttuja.

Koska INPUT#- ja GET#-käskyjä ei voida käyttää suorassa toimintatilassa, virheilmoituksen lukeminen on suoritettava pienen ohjelman avulla. Näitä käskyjä simuloimalla virheilmoitus voidaan lukea suoraankin. Seuraava pieni ohjelma on tarkoitettu virheilmoituksen lukemiseen.

```
10 OPEN 15,8,15
20 INPUT#15,VN,VI$,VU,VS
30 PRINT VN;VI$;VU;VS
40 CLOSE 15
```

Kuva 2.6

RIVI	SELITYS
10	komentokanavan avaaminen
20	virheilmoituksen lukeminen
30	virheilmoituksen tulostaminen
40	komentokanavan sulkeminen

Harjoitellaanpa. Käytä varmuuden vuoksi harjoituslevykettä,

joka ei sisällä tärkeitä ohjelmia. Kirjoita seuraava virheellinen alustuskomento.

```
OPEN 15,8,15,"N0 TESTI,TT":CLOSE 15
```

Kuva 2.7

Punaisen valon pitäisi nyt vilkkua koska komennon (NO) ja nimen välistä puuttuu kaksoispiste. Kun suoritat edellisen ohjelman RUN-käskyllä, saat selville virheilmoituksen.

```
34 SYNTAX ERROR 0  0
```

Kuva 2.8

Virheen numero on 34 ja virheilmoituksesta SYNTAX ERROR voidaan päätellä, että kysymyksessä on kirjoitusvirhe. Ura ja sektori ovat 0 ja 0, koska virhe ei liity niihin.

Jos kaikki on kunnossa, virheilmoituksena on

```
0 OK 0  0
```

Kuva 2.9

Kaikki virheilmoitukset on koottu selityksineen liitteeseen 2.

Ohjelmassa virhekanava on hyvä lukea aina tiedoston avaamisen, tiedoston sulkemisen, tiedostoon kirjoittamisen ja tiedostosta lukemisen jälkeen. Edellä esitetty ohjelma kannattaa sijoittaa aliohjelmaksi.

```
10 OPEN 15,8,15
20 OPEN 2,8,2,"0:TIEDOSTO,S,W"
30 GOSUB 600
40   .
     .
     .
100 PRINT#2,A$
110 GOSUB 600
     .
     .
599 REM VIRHEILMOITUS
```

```
600 INPUT#15,VN,VI$,VU,VS
610 IF VN<20 THEN RETURN
620 PRINT "LEVYASEMAVIRHE !!"
630 PRINT VN;VI$;VU;VS
640 CLOSE 2:CLOSE 15
650 END
```

Kuva 2.10

RIVI	SELITYS
10	komentokanavan avaaminen
20	tiedoston avaaminen
30	siirtyminen aliohjelmaan
40	paluu aliohjelmasta tälle riville
100	tiedostoon kirjoittaminen
110	siirtyminen aliohjelmaan
599	aliohjelman alku
600	virheilmoituksen lukeminen
610	jos virhenumero < 20, kaikki on kunnossa
620	virheilmoitus
630	virheilmoituksen tulostaminen
640	tiedoston ja komentokanavan sulkeminen
650	ohjelman lopettaminen

GET# -käskyllä virheilmoitus voidaan lukea esimerkiksi seuraavasti:

```
10 OPEN 15,8,15
20 GET#15,A$
30 PRINT A$;
40 IF A$<>CHR$(13) THEN 20
50 CLOSE 15
```

Kuva 2.11

RIVI	SELITYS
10	komentokanavan avaaminen
20	virheilmoituksen (yhden merkin) lukeminen
30	yhden merkin tulostaminen
40	jos merkki ei ole CHR$(13) (RETURN), uuden merkin lukeminen
50	komentokanavan sulkeminen

Jos INPUT#- tai GET#-käskyä käytetään suorassa toimintatilassa, kone antaa virheilmoituksen ?ILLEGAL DIRECT ERROR. Simuloimalla, matkimalla ohjelmointitilaa tai käyttämällä C64:n ROM-rutiineja näitä käskyjä voidaan käyttää suorassakin tilassa.

Muistipaikat 57–58 sisältävät BASIC-ohjelman käsiteltävänä olevan rivinumeron. Kun muistipaikkaan 58 sijoitetaan arvo 0, saadaan tietokone uskomaan, että kysymyksessä on ohjelma. GET#-käskyä voidaan silloin käyttää tavalliseen tapaan. Muista painaa kummankin rivin jälkeen RETURN-näppäintä.

```
CLOSE 15
POKE58,0:OPEN15,8,15:FORT=1TO30:GET#15,A
$:?A$;:IFST=0THENNEXT
```

Kuva 2.12

Jos olet varma, että komentokanava ei ole auki, voit jättää ensimmäisen rivin CLOSE 15 pois, muuten se on kirjoitettava.

C64:n GET# ROM -rutiinia käyttämällä voidaan toimia seuraavasti:

```
CLOSE15:OPEN15,8,15
FORT=1TO30:SYS(43909)15,A$:?A$;:IFST=0TH
ENNEXT
```

Kuva 2.13

tai

```
CLOSE15:OPEN15,8,15
FORT=1TO30:SYS43906#15,A$:?A$;:IFST=0THE
NNEXT
```

Kuva 2.14

Jos komentokanava on auki, voidaan kahdesta jälkimmäisestä esimerkistä jättää ensimmäinen rivi pois.

Myös INPUT#-käskyä voidaan simuloida.

```
CLOSE15
POKE58,0:OPEN15,8,15:SYS(43941)15, VN,VI
$,VU,VS:?VN;VI$;VU;VS
```

Kuva 2.15

Huomaa muuttujan VN edellä oleva välilyönti.

ROM-rutiinin avulla:

```
CLOSE15
POKE58,0:POKE198,1:OPEN15,8,15:INPUT#15,
 VN,VI$,VU,VS:?VN;VI$;VU;VS
```

Kuva 2.16

Huomaa muuttujan VN edellä oleva välilyönti.

Jos kanava 15 ei ole auki, voidaan kahdesta jälkimmäisestä esimerkistä jättää rivi CLOSE 15 pois.

3. LUKU

LEVYKETIETOJEN PÄIVITTÄMINEN

Tässä luvussa käsittelen levykkeen päivittämiseen kuuluvia toimenpiteitä, joista tavallisimmat ovat

1) Levykkeen alustaminen

2) Tiedoston poistaminen levykkeeltä

3) Tiedoston uudelleennimeäminen

4) Tiedoston kopioiminen

5) Levykkeen initialisointi

6) Levykkeen sisällysluettelon saattaminen ajan tasalle (Validate)

Harvinaisempia päivitystoimenpiteitä ovat esim. levykkeen nimen muuttaminen ja tiedoston lukitseminen. Näitä asioita käsitellään kahdeksannessa luvussa.

Levyaseman mukana tulevalla levykkeellä olevaa apuohjelmaa DOS 5.1 ja sen käyttöä selvitetään liitteessä 4.

LEVYKKEEN ALUSTAMINEN – **NEW**

1541-levyasema ei tunnista uutta, käyttämätöntä levykettä, ellei sitä ensin alusteta eli formatoida. Alustamalla luodaan puitteet, joissa DOS kykenee toimimaan. Alustuksen suorittaa DOS.

Levykkeen alustaminen suoritetaan NEW-komennolla, joka lähetetään komentokanavan kautta.

Syntaksi:

```
NEW0:NIMI,ID
```

tai

```
N0:NIMI,ID
```

Kuva 3.1

NIMI = levykkeen nimi
ID = levykkeen tunnuskoodi

Komentosanan jäljessä oleva 0 viittaa levyasemaan 0 (jäänne Commodoren muista levyasemista) ja sen voi jättää pois. Jotkut käyttäjät ovat raportoineet, että komennon yhteydessä käytetty 0 varmistaa komentojen suorittamista.

Levykkeen nimessä saa olla korkeintaan 16 merkkiä (ei kertomerkkiä (*), kysymysmerkkiä (?), eikä pilkkua (,)). Eräät graafiset ja käänteiset merkit voivat saada aikaan omituisen sisällysluettelon. Niiden käyttöä on vältettävä, jos on vähänkin epävarma.

Levykkeen ID-numero muodostuu kahdesta merkistä. Jokaisella levykkeellä pitää olla oma ID-numeronsa (katso INITIALIZE), jotta vältytään harmittavilta sekaannuksilta.

Esimerkkejä:

```
NEW0:LEVYKE 1,L1

N0:PELIT,PP

10 N$="N0:DISKETTI-1,D1":OPEN 15,8,15
20 PRINT#15,N$
30 CLOSE 15
```

Kuva 3.2

Alustaminen kestää noin 80 sekuntia. Jos haluat ottaa aikaisemmin käytössä olleen levykkeen uudelleen käyttöön ja poistaa siitä kaikki tiedostot, voit käyttää NEW-komennon nopeampaa muotoa, jossa ID-numero jätetään pois. Tällä komennolla voit alustaa vain sellaisen levykkeen, joka on jo kerran alustettu.

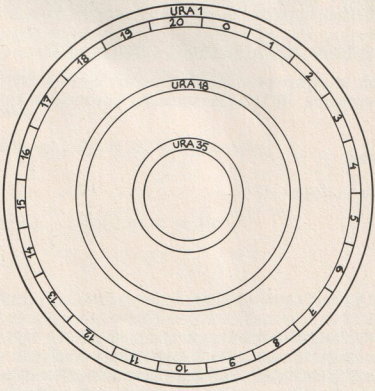
Esimerkki:

```
10 N$="N0:DISKETTI-1":OPEN 15,8,15
20 PRINT#15,N$
30 CLOSE 15
```

Kuva 3.3

Mitä alustamisen aikana tapahtuu? Ensin levyasema tulkitsee komennon. Jos komennon syntaksi on oikea, varsinainen alustaminen alkaa. Muussa tapauksessa punainen valo vilkkuu ja kone antaa virheilmoituksen ?SYNTAX ERROR.

Aluksi askelmoottori kuljettaa luku- ja kirjoituspään levykkeen keskustasta uloimmalle reunalle. Alustuksen alussa kuuluva kolina aiheutuu pään osumisesta mekaaniseen pysäyttimeen ja on aivan normaalia. Näin varmistutaan siitä, että ollaan uloimmalla reunalla, uralla 1. Levykkeen pyöriessä kierroksen, luku- ja kirjoituspään ohittaa renkaan muotoinen alue, jota kutsutaan uraksi (leveys n. 0,4 mm). Aluksi uralta 1 pyyhitään kaikki aiempi tieto pois. Sen jälkeen kirjoitettava uusi tieto jakaa uran sektoreihin.



Kuva 3.4

Levykkeellä on kaikkiaan 35 uraa. Sektoreiden lukumäärä vaihtelee seuraavasti:

URA	SEKTORIT	SEKTOREIDEN KOKONAISMÄÄRÄ
1 – 17	0 – 20	21
18 – 24	0 – 18	19
25 – 30	0 – 17	18
31 – 35	0 – 16	17

Kun kaikki urat on käyty läpi ja uudet tiedot on merkitty sektoreihin, tarkistetaan, että tallennus on tapahtunut oikein. Tämän jälkeen luku- ja kirjoituspää siirtyy uralle 18 sektorille 0. Tälle sektorille tallennetaan levykkeen nimi ja ID-numero sekä muodostetaan taulukko, joka ilmoittaa kullakin sektorilla vapaana olevat tietolohkot: BAM (Block Availability Map). Tyhjä sisällysluettelo merkitään uran 18 sektorille 1.

NEW-komennon nopeampaa muotoa käytettäessä luku- ja kirjoituspää siirtyy heti uralle 18 sektorille 0, jonne kirjoitetaan levykkeen uusi nimi, vanha ID-numero sekä tyhjä BAM. Uran 18 sektorille 1 merkitään tyhjä sisällysluettelo. Tämä käy huomattavasti nopeammin kuin perusteellinen alustus.

LEVYKKEEN INITIALISOINTI – INITIALIZE

Tässäkin on kysymyksessä eräänlainen alustus. Nyt alustetaan levyasema. INITIALIZE-komento lyhennetään kirjaimella I.

Syntaksi:

```
INITIALIZEØ

tai

IØ
```

Kuva 3.5

Kun INITIALIZE-komento annetaan, levykkeen nimi ja BAM luetaan uralta 18 sektorilta 0 levyaseman sisäiseen muistiin (alueelle $0700–$07FF). Samalla päivitetään levyaseman muistipaikoissa $12/$13 oleva ID. Levyaseman tärkeimmät muistipaikat ovat liitteessä 1.

Tavallisesti DOS suorittaa initialisoinnin automaattisesti aina levyketoimenpiteiden yhteydessä. Levykettä vaihdettaessa ja levyvirheiden yhteydessä on kuitenkin hyvä suorittaa initialisointi erikseen. Näin voit varmistua siitä, että levyaseman muistissa oleva BAM ja levykkeen BAM vastaavat toisiaan. Jos kahdella levykkeellä on sama ID-numero, levyasema ei erota niitä toisistaan ilman I-komentoa. Jos I-komentoa ei anneta, levyasema toimii virheellisen BAM:in pohjalta ja arvokasta tietoa saattaa tuhoutua.

Esimerkkejä:

```
OPEN 15,8,15
PRINT#15,"I0"
CLOSE 15

tai

OPEN 15,8,15,"I0":CLOSE 15
```

Kuva 3.6

TIEDOSTON NIMEÄMINEN UUDELLEEN – **RENAME**

Levykkeellä olevan tiedoston tai ohjelman nimi on usein muutettava. Nimen muuttaminen suoritetaan RENAME-komennolla, joka voidaan lyhentää kirjaimella R.

Syntaksi:

```
RENAME0:UUSI NIMI=VANHA NIMI

tai

R0:UUSI NIMI=VANHA NIMI
```

Kuva 3.7

Tiedoston uudessa nimessä saa olla korkeintaan 16 merkkiä ja sitä koskevat samat rajoitukset kuin tiedoston nimiä yleensä. Vanha nimi on kirjoitettava täsmälleen sellaisena kuin se esiintyy levykkeen sisällysluettelossa. Jokereita (katso 3.7) ei saa käyttää. RENAME-komennolla voidaan nimetä uudestaan minkätyyppinen tiedosto tahansa. Avoinna olevaa tiedostoa ei kuitenkaan voi nimetä uudelleen.

Kun komento annetaan, DOS tarkistaa sisällysluettelosta sekä uuden että vanhan nimen. Jos vanhaa nimeä ei löydy, saadaan virheilmoitus ?FILE NOT FOUND. Jos uusi nimi on jo sisällysluettelossa, saadaan virheilmoitus ?FILE EXISTS.

Esimerkkejä:

```
OPEN 15,8,15
PRINT#15,"R0:PELI2=PELI1"
CLOSE 15

OPEN 15,8,15,"R0:ASIA/U=ASIA/V":CLOSE 15

10 OPEN 2,8,15
20 UN$="KOLMAS":VN$="TOINEN"
30 N$="R0:"+UN$+"="+VN$
40 PRINT#2,N$
50 CLOSE 2
```

Kuva 3.8

TIEDOSTON TUHOAMINEN – **SCRATCH**

Tiedosto tai ohjelma voidaan poistaa levykkeeltä SCRATCH-komennolla, joka lyhennetään kirjaimella S.

Syntaksi:

```
SCRATCH0:NIMI

tai

S0:NIMI
```

Kuva 3.9

Yhdellä SCRATCH-komennolla voidaan poistaa useita tiedostoja (jopa kaikki) samalla kertaa käyttämällä jokereita * ja ? (katso 3.7). Tällöin on noudatettava erikoista varovaisuutta.

Esimerkkejä:

```
OPEN 15,8,15
PRINT#15,"S0:PELI2"
CLOSE 15
```

```
OPEN 15,8,15,"S0:TIEDOT":CLOSE 15
```

```
10 OPEN 5,8,15
20 TN$="SYNTAX"
30 N$="S0:"+TN$
40 PRINT#5,N$
50 CLOSE 5
```

Kuva 3.10

Kun SCRATCH-komento annetaan, DOS tarkistaa levykkeen sisällysluettelon ja etsii annetun tiedoston. Kun tiedosto on löytynyt, DOS tarkistaa tiedoston tyypin. Jos kysymyksessä on tavallinen tiedosto (PRG, SEQ, REL tai USR), tyyppiä ilmoittava tavu saa arvon $00 eikä tiedosto enää esiinny sisällysluettelossa. Tämän jälkeen DOS selvittää tiedoston käytössä olleet sektorit ja merkitsee ne vapaiksi BAM:iin. Tiedostoa ei pyyhitä pois, vaan sen käyttämä tila merkitään vapaaksi. Jos tuhoamisen jälkeen tallennat levykkeelle uuden tiedoston, sen tiedot asettuvat syntyneeseen tyhjään tilaan eivätkä sijoitu sisällysluettelon loppuun.

Koska tiedostoa ei pyyhitä pois, vahingossa tuhottu tiedosto voidaan vielä pelastaa. Jos tuhoamasi ohjelma oli viimeisenä ladattu tai tallennettu etkä tuhoamisen jälkeen ole tehnyt mitään, voit kokeilla komentoa:

```
LOAD "*",8
```

Kuva 3.11

Tällä komennolla voit saada viimeksi käsitellyn ohjelman ladattua tietokoneen muistiin. Jos ohjelma latautuu, tallenna se ennen kuin teet mitään muuta. Jos tämä keino ei auta, voit yrittää pelastaa tilanteen kirjaan liittyvällä levykkeellä olevan PELASTA-nimisen ohjelman avulla.

Levykkeellä oleva tiedosto voidaan lukita siten, että tyyppiä ilmoittavan tavun 6. bitti asetetaan ykköseksi. Lukitun tiedoston tyypin perään on sisällysluettelossa merkitty <-merkki, esim. PRG<. Lukittua tiedostoa ei voida tuhota SCRATCH-komennolla.

Jos tiedostoa ei ole suljettu, tyyppiä ilmoittavan tavun 4. bitti on nolla. Sisällysluettelossa tällaisen tiedoston tyypin eteen on merkitty *-merkki, esim. *SEQ. Jos tiedosto, jota ei ole suljettu kunnolla, tuhotaan SCRATCH-komennolla, tyypin ilmoittava

tavu saa arvon $00. Tiedostoa ei jäljitetä eikä sen käyttämää tilaa merkitä BAM:iin vapaaksi. Tällainen tiedosto on syytä tuhota VALIDATE-komennolla. Jos käytät vahingossa SCRATCH-komentoa, sen jälkeen on muistettava antaa VALIDATE-komento.

TIEDOSTON KOPIOIMINEN – COPY

COPY-komennon avulla voit ottaa tiedostosta varmuuskopion. Varmuuskopio tallennetaan alkuperäisen tiedoston kanssa samalle levykkeelle. COPY-komento lyhennetään kirjaimella C.

Syntaksi:

```
COPY0:NIMI 2=0:NIMI 1

tai

C0:NIMI 2=0:NIMI 1
```

Kuva 3.12

NIMI 1 on tiedoston alkuperäinen nimi ja NIMI 2 on varmuuskopion nimi. Nimet eivät saa olla samoja eikä niissä saa käyttää jokereita *, ?, pilkkua (,) tai kaksoispistettä (:). Alkuperäisen nimen pitää olla täsmälleen samassa muodossa kuin levykkeen sisällysluettelossa.

COPY-komennolla ei voida ottaa varmuuskopiota suhteellisesta tiedostosta (REL). Ennen komennon antamista on varmistettava, että levykkeellä on tarpeeksi tilaa kopiota varten.

Esimerkkejä:

```
OPEN 15,8,15
PRINT#15,"C0:PELI2=PELI1"
CLOSE 15

OPEN 15,8,15,"C0:TIEDOT=TIEDOT.1"
CLOSE 15
```

Kuva 3.13

Kun COPY-komento annetaan, DOS etsii sisällysluettelosta sekä alkuperäisen tiedoston että kopion nimeä. Jos alkuperäistä ei löydy, se antaa virheilmoituksen ?FILE NOT FOUND. Jos sisällysluettelosta löytyy kopion kanssa samanniminen tiedosto, saadaan virheilmoitus ?FILE EXISTS. Jos kaikki on kunnossa, kopion nimi merkitään sisällysluetteloon. DOS lukee sitten alkuperäisen tiedoston ja kirjoittaa sen uudestaan levykkeelle. Tämän jälkeen BAM ja sisällysluettelo päivitetään ajan tasalle.

COPY-komennosta ei ole tässä muodossa paljoakaan hyötyä. Jos haluat varmuuskopion jostakin tiedostosta, voit käyttää kirjan levykkeellä olevia kopiointiohjelmia. Koko (suojaamattomasta) levykkeestä saat varmuuskopion BACKUP-ohjelmalla.

COPY-komennon avulla 2–4 tiedostoa voidaan liittää toisiinsa yhdeksi isoksi tiedostoksi.

Syntaksi:

```
COPY0:ISO=0:PIENI1,0:PIENI2,0:PIENI3

tai

C0:ISO=PIENI1,PIENI2,PIENI3,PIENI4
```

Kuva 3.14

Kaikkien yhdistettävien tiedostojen PIENI1–PIENI4 pitää olla samalla levykkeellä ja levykkeellä tulee olla tarpeeksi tilaa tiedostoa ISO varten. Yhdistettävien tiedostojen nimet pitää kirjoittaa täysin samalla tavalla kuin sisällysluettelossa. Yksikään tiedosto ei saa olla suhteellinen tiedosto. Jos yhdistettävät tiedostot ovat erityyppisiä, yhdistämällä saadun tiedoston tyypiksi tulee ensimmäisen tiedoston PIENI1 tyyppi.

BASIC-kielisiä ohjelmatiedostoja ei voida liittää (MERGE) toisiinsa tällä tavoin. Ainoastaan ensimmäinen ohjelma toimii, muut käyttävät vain muistia. Tässä muodossa COPY-komento on hyödyllinen vain peräkkäistiedostoja yhdistettäessä.

Yhdistämällä saadulla tiedostolla voi olla virheellinen (liian suuri) koko sisällysluettelossa, koska koko saadaan laskemalla yhteen yhdistettävien tiedostojen sisällysluetteloon merkityt lohkot. Jos viimeinen tiedoston käyttämä lohko on vajaa, summan arvo on todellista arvoa suurempi.

VALIDATE-KOMENTO

Tämän komennon avulla merkitään vapaiksi sellaiset sektorit, jotka eivät kuulu mihinkään tiedostoon mutta jotka on BAM:iin merkitty varatuiksi. Sektorin voi merkitä tahallisesti varatuksi käyttämällä ns. suorasaantikäskyjä (BLOCK-ALLOCATE, 8. luku). VALIDATE-komennolla sektorin saa uudestaan käyttöön. Komento lyhennetään kirjaimella V.

Syntaksi:

```
VALIDATE0

tai

V0
```

Kuva 3.15

Esimerkkejä:

```
OPEN 15,8,15
PRINT#15,"V0"
CLOSE 15

tai

OPEN 15,8,15,"V0":CLOSE 15
```

Kuva 3.16

Tiedosto, jota ei ole suljettu (sisällysluettelossa esim. *SEQ), tuhotaan käyttämällä VALIDATE-komentoa. SCRATCH-komento muuttaa vain tällaisen tiedoston tyyppiä osoittavan tavun eikä vapauta tiedostoon kuuluvia sektoreita BAM:ista.

Jos BAM ei ole kunnossa, uusi ohjelma tai tiedosto saattaa mennä levykkeellä ennestään olevan tiedoston päälle, tai jotkut sektorit voivat kuulua kahteen tai useampaan tiedostoon. Tällaiset ristitiedostot ovat sekavia. BAM:issa esiintyy häiriöitä useimmiten sen seurauksena, että SCRATCH-komennolla tuhotaan tiedostoja, joita ei ole suljettu, tai käytetään SAVE-käskyn korvaavaa muotoa levykkeen ollessa melkein täynnä.

VALIDATE-komentoa ei voi käyttää satunnaistiedostoja sisältävän levykkeen kanssa. Satunnaistiedosto ei ole sama kuin suhteellinen tiedosto.

Joillakin levykkeillä on suojauksen osana (esim. pelit) tahallaan varatuiksi merkittyjä sektoreita. VALIDATE-komentoa ei tällöin pidä käyttää.

VALIDATE-komentoa suorittaessaan DOS merkitsee ensin levyaseman RAM-muistiin tyhjän BAM:in, jossa kaikki sektorit on merkitty vapaiksi. Sitten se hakee sisällysluettelon ensimmäisen tiedoston nimen ja alkuosoitteen ja jäljittää ura-sektori-osoittimien avulla tiedoston merkiten BAM:iin tiedoston käyttämät sektorit (lohkot) varatuiksi. Kun kaikki tiedostot on käyty läpi, levykkeelle kirjoitetaan uusi BAM. Tiedostoja, joita ei ole suljettu, ei jäljitetä. Niiden käyttämät sektorit ovat uudessa BAM:issa vapaina.

Jos levykkeellä on lukuvirhe, VALIDATE-komento keskeytyy. Levykkeen sisällysluettelo ei muutu, joten lukuvirheestä ei aiheudu harmia.

JOKEREIDEN KÄYTTÖ

Komento voidaan kohdistaa samalla kerralla usean tiedoston joukkoon kun käytetään ns. jokereita: * ja ?.

Kertomerkillä * voidaan korvata tiedoston nimen jäljessä kuinka monta merkkiä tahansa. Kysymysmerkillä ? voidaan korvata mikä tahansa yksittäinen merkki.

Esimerkkejä:

```
1) S0:DAT*      2) S0:??DAT
```

Kuva 3.17

Ensimmäisen esimerkin komento tuhoaa kaikki tiedostot, joiden nimi alkaa kirjaimilla DAT. Nimessä voi olla kuinka monta merkkiä tahansa (ei kuitenkaan yli 16). Toisen esimerkin komento tuhoaa kaikki ne tiedostot, joiden nimessä on 5 merkkiä. Kaksi ensimmäistä merkkiä saavat olla mitkä tahansa ja kolme viimeistä ovat DAT.

```
S0:*
```

Kuva 3.18

Tämä komento tuhoaa kaikki levyllä olevat tiedostot (vastaten nopeaa NEW-komentoa).

```
1) S0:?A????     2) S0:?A*
```

Kuva 3.19

Ensimmäisen esimerkin komento tuhoaa kaikki ne tiedostot, joiden nimessä on 6 merkkiä ja toisena merkkinä on A. Toisen esimerkin komento tuhoaa kaikki tiedostot, joiden nimen toisena merkkinä on A.

SCRATCH-komennon yhteydessä pitää jokereita käyttää erittäin varovaisesti, jotta tärkeitä tiedostoja ei vahingossa tuhoudu.

On myös mahdollista tuhota kaikki määrätyn tyyppiset tiedostot. Tällöin SCRATCH-komentoa käytetään seuraavasti:

```
SCRATCH0:*=XXX  tai  S0:*=XXX
```

Kuva 3.20

XXX = PRG, SEQ, REL tai USR (myös P, S, R tai U)

Esimerkkejä:

```
1) S0:*=SEQ     2) S0:*=P
```

Kuva 3.21

Edellinen komento tuhoaa kaikki peräkkäistiedostot, jälkimmäinen kaikki ohjelmatiedostot.

Jokereita voidaan käyttää myös sisällysluetteloa ladattaessa.

```
LOAD "$:PE*",8
```

Kuva 3.22

Kaikki kirjaimilla PE alkavat tiedostot saadaan tällä komennolla sisällysluettelosta näyttöön. Kaikki ohjelmatiedostot saadaan näyttöön komennolla

```
LOAD "$:*=PRG",8
```

Kuva 3.23

Määrätty ohjelma saadaan sisällysluetteloon komennolla

```
LOAD "$:NIMI",8
```

Kuva 3.24

NIMI = ohjelman nimi

Sisällysluettelon ensimmäinen ja viimeinen rivi saadaan seuraavilla komennoilla:

```
LOAD "$$",8  tai  LOAD "$:XXXX",8
```

Kuva 3.25

Jos edellistä komentoa käytetään kaksi kertaa peräkkäin, saadaan virheilmoitus FILE NOT FOUND. Jälkimmäisessä komennossa voi merkkien XXXX paikalla olla mikä tahansa merkkijono, joka ei esiinny ohjelman nimenä.

Jokereiden käyttöä kannattaa harjoitella, sillä ne helpottavat ja nopeuttavat päivitystoimenpiteitä.

4. LUKU

TIEDOSTOJEN KÄSITTELY

Tavallisten ohjelmien (BASIC- tai konekielisten ohjelmien) lisäksi levykkeelle voidaan tallentaa tietoa, jota erityyppiset ohjelmat voivat käyttää hyväkseen.

Tiedosto muodostuu useista toisiinsa läheisesti liittyvistä tiedoista. Tiedosto voidaan muodostaa esimerkiksi kirjahyllyn kirjoista. Tiedosto jaetaan tietueisiin. Kirjahyllytiedostossa muodostavat yhden kirjan tiedot tietueen. Tietue jaetaan edelleen kenttiin. Tietueessa voi olla useita kenttiä. Kirjahyllytiedoston tietue eli yhden kirjan tiedot voidaan jakaa viiteen kenttään seuraavasti:

1) kirjan tekijä

2) kirjan nimi

3) kirjan kustantaja

4) ilmestymisvuosi

5) kirjan sisältökuvaus

Kun tietueet kirjoitetaan tiedostoksi levykkeelle, eri tietueet ja niiden kentät tulevat peräkkäin. Ne erotetaan toisistaan erottimella, joka on CHR$(13) (RETURN), CHR$(44) (pilkku) tai CHR$(57) (puolipiste).

Tiedostoja on kahta päätyyppiä: peräkkäistiedosto ja suhteellinen tiedosto (relatiivinen tiedosto). Peräkkäistiedostossa tietueet on kirjoitettu peräkkäin. Halutessamme lukea 40. tietueen tiedot, meidän pitää ensin lukea sitä edeltävät 39 tietuetta. Suhteellisenkin tiedoston tiedot ovat peräkkäin, mutta mää-

rätyn tietueen tietoihin päästään käsiksi suoraan muita tietueita lukematta.

1541-levyasemalla voidaan käyttää myös ns. satunnaistiedostoja, joita ei pidä sotkea suhteellisiin tiedostoihin. Tällöin käytetään ns. suorasaantikäskyjä ja kullakin tietueella on levyn pinnalla määrätty paikkansa (ura & sektori). Näitä asioita käsitellään kahdeksannessa luvussa.

ERI TIEDOSTOTYYPIT

Seuraavassa luetellaan tavallisimmat tiedostotyypit. Sulkeisiin on merkitty tiedoston sisällysluettelossa näkyvä tunnus.

ohjelmatiedosto	(PRG)
peräkkäistiedosto	(SEQ)
suhteellinen tiedosto	(REL)
käyttäjän tiedosto	(USR)

Tiedosto, jota ei ole suljettu, on merkitty sisällysluetteloon tyypin edessä olevalla *-merkillä, esim. *SEQ, *PRG. Suhteellinen tiedosto ei voi esiintyä sisällysluettelossa sulkemattomana.

Lukittu tiedosto on merkitty sisällysluetteloon tyypin jäljessä olevalla <-merkillä, esim. PRG<, REL<. Tiedoston lukitsemista käsitellään yhdeksännessä luvussa ja ohjelmassa nimeltä LUKITSE.

Sisällysluettelossa voi tiedoston tyyppinä olla myös DEL (deleted, poistettu). Kysymyksessä ei ole tuhottu tiedosto vaan Commodoren dokumentoimaton tiedostotyyppi, joka on erittäin harvinainen. Sitä käytetään joskus suojauksen osana ja sen tarkoituksena on yleensä vain aiheuttaa sekaannusta ja epätietoisuutta. Yhdeksännessä luvussa käsitellään sisällysluettelon rakennetta tarkemmin.

TIEDOSTON KÄSITTELYN PÄÄVAIHEET

Tiedoston käsittelyssä voidaan erottaa kolme päävaihetta:

1) tiedoston avaaminen

2) tiedoston päivittäminen (tietueen kirjoittaminen, tietueen lukeminen, tietueen poistaminen jne.)

3) tiedoston sulkeminen

Tiedostoja käsitellään seuraavassa ainoastaan levyaseman kannalta. Muiden oheislaitteiden (kasettiasema, kirjoitin jne.) osalta saat tietoa laitteiden käsikirjoista.

TIEDOSTON AVAAMINEN

Tiedostoa avattaessa oheislaitteen ja ohjelman välille tarvitaan tiedonsiirtoyhteys tai -kanava. Yhteyden määräämiseksi pitää avauskomennon mukana ilmoittaa tarvittavat parametrit:

Looginen tiedostonumero, jolla viitataan kyseiseen yhteyteen.

Laitenumero, joka ilmoittaa kysymyksessä olevan oheislaitteen.

Kanavanumero tai toisio-osoite, jolla ilmoitetaan toimintatapa (kirjoittaminen tai lukeminen).

Tiedoston tunnus, joka muodostuu tiedoston nimestä, tiedoston tyypistä ja käsittelytavasta.

Suhteellisilla tiedostoilla ilmoitetaan lisäksi tietueen pituus.

Tiedoston avaaminen suoritetaan OPEN-käskyllä.

Syntaksi:

```
OPEN TN,LN,KN,"TU"
```

Kuva 4.1

TN = looginen tiedostonumero (1–127)
LN = laitenumero (levyasemalla 8–11)

KN = kanavanumero eli toisio-osoite (2–14 ovat tietokanavia, 15 on komentokanava)
TU = tiedoston tunnusosa

Looginen tiedostonumero

Koska kymmenen tiedostoa voi olla samanaikaisesti auki eri oheislaitteille, tiedostot pitää erottaa toisistaan. Esimerkiksi yksi tiedosto voi olla auki kirjoittimelle, yksi kasettiasemalle ja kaksi levyasemalle. Nämä tiedostot erotetaan toisistaan loogi-

sen tiedostonumeron avulla. Kun tiedostoon viitataan avaamisen jälkeen, käytetään avauskäskyn yhteydessä ilmoitettua loogista tiedostonumeroa. Kun tiedosto on suljettu, tiedostonumero on vapaa muuhun käyttöön. Kahdessa samanaikaisesti auki olevalla tiedostolla pitää olla erilaiset loogiset tiedostonumerot.

Jos tiedosto avataan käyttämällä esim. tiedostonumeroa 2 ja suljetaan esim. tietueen kirjoittamisen jälkeen, sama tiedosto voidaan avata uudestaan käyttämällä toista tiedostonumeroa. Tiedostonumero on liitetty vastaavaan tiedostoon vain avaamisen ja sulkemisen välisen ajan.

Looginen tiedostonumero on 1–127. Voit käyttää myös numeroita 128–255, mutta ne aikaansaavat hieman erilaisen toiminnan. Näitä tiedostonumeroita käytettäessä tietokone lähettää rivin lopussa RETURN-merkin CHR$(13) lisäksi myös rivinvaihtomerkin CHR$(10). Tämä menettely vie tilaa eikä tiedostoa voida lukea INPUT#-käskyllä. Jokaisen luettavan tiedon ensimmäisenä merkkinä on tällöin CHR$(10).

Vaikka auki olevia tiedostoja voi kerralla olla kymmenen, levyasemalle saa kerralla olla auki vain neljä. Seuraavaan taulukkoon on koottu tiedot siitä, kuinka monta erityyppistä tiedostoa voi kerralla olla auki.

PRG tai SEQ	REL	SUORASAANTI-TIEDOSTOT	YHTEENSÄ
0	0	0 – 4	0 – 4
0	1	0 – 2	1 – 3
1	1	0 – 1	2 – 3
2	0	0 – 2	2 – 4
3	0	0 – 1	3 – 4

Suorasaantitiedostot käsitellään kahdeksannessa luvussa. Huomaa, että kaksi suhteellista tiedostoa ei voi olla samanaikaisesti auki.

Laitenumero

Tämä numero ilmoittaa laitteen, jonka kanssa tietoa välitetään. Commodoren oheislaitteet käyttävät seuraavia laitenumeroita:

1 = kasettiasema
2 = RS-232C (V24)-liitäntä, modeemi
3 = näyttöpääte

4 – 5 = kirjoitin
8 – 11 = levyasema

Jos haluat käyttää kahta 1541-levyasemaa samanaikaisesti, toisen laitenumero on muutettava. Levyaseman laitenumero on tavallisesti 8. Muutos voidaan tehdä väliaikaisesti tai pysyvästi. Pysyvä laitenumeron muuttaminen on paras antaa huoltomiehen tehtäväksi. Väliaikaisen muutoksen voi tehdä M–W (Memory–Write)-komennon avulla (katso 8. luku.)

Kanavanumero

Kanavanumerosta käytetään myös nimeä toisio-osoite. Sen merkitys on eri oheislaitteilla erilainen ja se määrää toimintatavan. Seuraavaan taulukkoon on koottu kanavanumeroiden merkitykset levyaseman kannalta.

0	DOS:in käytössä ohjelmien ja sisällysluettelon lataamiseksi
1	DOS:in käytössä ohjelmien tallentamiseksi
2 – 14	levyketiedoston kirjoittaminen/lukeminen
15	komentokanava
16 – 17	varattu DOS:in sisäiseen käyttöön
>18	ei käytössä

Jokaisella auki olevalla tiedostolla pitää olla oma kanavanumeronsa.

Komentokanavaa käytetään päivityskomentojen (3. luku) lähettämiseen levyasemalle ja levyaseman virheilmoituksen lukemiseen. Kahdeksannessa luvussa käsiteltävät suorasaantikäskyt käyttävät myös komentokanavaa.

Jos tiedosto avataan lukemista varten käyttämällä kanavanumeroa 0, tulos on sama kuin numeroilla 2–14. Jos luetaan sisällysluettelotiedostoa $, tulos on sama kuin käskyllä LOAD"$",8. Jos tiedosto avataan kanavanumerolla 0 kirjoittamista varten, saadaan virheilmoitus ?FILE NOT FOUND.

Jos tiedosto avataan kirjoittamista varten kanavanumerolla 1, tulos on sama kuin numeroilla 2–14. Jos tiedosto avataan lukemista varten, saadaan virheilmoitus ?FILE EXISTS.

Tiedoston tunnusosa

Avauskäskyn tässä osassa ilmoitetaan tiedoston nimi, tyyppi ja toimintatila.

Tunnusosa on seuraavanlainen ohjelma-, peräkkäis- ja käyttäjän tiedostoille:

```
"AS:NIMI,TYYPPI,TILA"
```

Kuva 4.2

AS = asemanumero; 1541-levyasemalla aina 0
NIMI = tiedoston nimi sellaisena kuin se esiintyy sisällysluettelossa. Jos tiedosto on jo olemassa, jokereita saa käyttää.
TYYPPI = PRG, SEQ tai USR. Tyypin voi lyhentää P:llä, S:llä tai U:llä ja sen saa jättää pois, jos tiedosto avataan lukemista varten.
TILA = toimintatila; READ tai WRITE (R tai W). Jos tilaa ei ilmoiteta, oletusarvona on READ.

Kaikki tai osa tunnusosan sisältämästä tiedosta voidaan liittää yhden merkkijonomuuttujan arvoksi.

Esimerkkejä:

```
"0:KORTIT,S,R"

"0:TUOTTEET,SEQ,W"

"0:PELKURI,PRG,R"
```

Kuva 4.3

Jos olemassa oleva tiedosto halutaan korvata kirjoittamalla uudet tiedot vanhojen päälle, asemanumerona pitää olla @ 0.

Jokereita ei saa käyttää tiedoston nimessä, jos tiedosto avataan kirjoittamista varten.

Toimintatilojen READ ja WRITE lisäksi on olemassa tilat APPEND (A) ja MODIFY (M). A-tilaa käyttämällä olemassa olevan tiedoston perään voidaan kirjoittaa lisää tietoa lukematta koko tiedostoa tietokoneen muistiin (6. luku). M-tilan avulla voidaan lukea sellainen tiedosto, jota ei ole suljettu (6. luku).

Esimerkkejä:

```
OPEN 2,8,2,"0:TIEDOSTO,S,R"

OPEN 5,8,5,"0:LEHDET,SEQ,W"

OPEN 5,8,5,"0:LEVYT,S,A"


10 INPUT "TIEDOSTON NIMI";T$
20 OPEN 2,8,2,"@0:"+T$+",S,W"
```

Kuva 4.4

Suhteellisten tiedostojen tunnusosa on seuraavanlainen:

```
"AS:NIMI,L,"+CHR$(PIT)
```

Kuva 4.5

AS	=	asemanumero (aina 0)
NIMI	=	tiedoston nimi, jossa saa olla jokereita, jos tiedosto on ennestään olemassa
L	=	suhteellisen tiedoston tyyppi
PIT	=	tietueen pituus (2–254)

Jos tiedosto on jo olemassa, tunnusosa voidaan esittää lyhyemmin:

```
"AS:NIMI"
```

Kuva 4.6

Suhteellisen tiedoston toimintatilaa ei tarvitse ilmoittaa. Kun tiedosto avataan, siihen voidaan aina kirjoittaa tai sitä voidaan lukea. Suhteellista tiedostoa ei voida korvata käyttämällä asemanumeroa 0.

Tietueen pituutena ei voi olla 42, 58 tai 63, koska ne vastaavat merkkien *, : ja ? ASCII-koodeja ja näillä merkeillä on DOS:ille erikoismerkitys.

Tietueen pituus pitää aina ilmoittaa suhteellista tiedostoa luotaessa. Tämän jälkeen tietueen pituutta ei voi muuttaa.

Esimerkkejä:

```
OPEN 2,8,2,"0:LUETTELO,L,"+CHR$(150)

OPEN 5,8,2,"0:KORTIT,L,"+CHR$(60)

OPEN 5,8,2,"KORTIT"
```

Kuva 4.7

TIEDOSTOON KIRJOITTAMINEN

Kirjoittaminen tapahtuu PRINT#-käskyn avulla.

Syntaksi:

```
PRINT#TN,ML
```

Kuva 4.8

TN = looginen tiedostonumero, joka on annettu OPEN-käskyn yhteydessä
ML = muuttujalista

PRINT#-käskyä voidaan käyttää sekä suorassa toimintatilassa että ohjelmassa.

PRINT#-käskyä ei saa lyhentää ?#, vaan sen oikea lyhenne on P+SHIFT R. Käskysanan (PRINT) ja #-merkin välissä ei saa olla välilyöntiä.

PRINT#-käsky lähettää automaattisesti muuttujan arvon perään merkin CHR$(13) (RETURN), ellei sitä estetä puolipisteellä.

Esimerkkejä:

```
1) PRINT#2,A$      2) PRINT#2,A$;
```

Kuva 4.9

Ensimmäisessä esimerkissä CHR$(13) kirjoitetaan, toisessa esimerkissä sitä ei kirjoiteta.

Jos samalla PRINT#-käskyllä kirjoitetaan useiden muuttujien arvoja, kenttien (muuttujien arvojen) väliin on kirjoitettava selvä erotin, esim. CHR$(13).

Esimerkiksi:

```
PRINT#3,A$;CHR$(13);B$;CHR$(13);C$
```

Kuva 4.10

Erottimella varmistetaan tietojen oikea lukeminen INPUT#-käskyllä.

TIEDOSTON LUKEMINEN

Tiedoston lukeminen suoritetaan joko INPUT#- tai GET#-käskyllä.

Syntaksi:

```
INPUT#TN,ML

GET#TN,ML
```

Kuva 4.11

TN = looginen tiedostonumero, joka on annettu OPEN-käskyn yhteydessä
ML = muuttujalista

INPUT#-käsky voidaan lyhentää I + SHIFT N, mutta GET#-käskyä ei voi lyhentää. Käskysanojen ja #-merkin välissä ei saa olla välilyöntiä. Jos samalla kertaa luetaan arvo usealle muuttujalle, muuttujat erotetaan toisistaan pilkulla.

Esimerkiksi:

```
1) INPUT#3,A$,B$,C$

2) GET#2,A$,B$
```

Kuva 4.12

Lukukäskyissä INPUT# ja GET# esiintyvän muuttujan ja luettavan tiedon on tyypiltään vastattava toisiaan. Merkkijonoa ei voida lukea numeerisen muuttujan arvoksi. GET#-käskyn yhteydessä kannattaa aina käyttää lukumuuttujana merkkijonomuuttujaa.

GET#-käskyllä voidaan lukea yksi merkki kerrallaan ja siitä syystä se on erittäin hidas. GET#-käsky ei lue merkkiä CHR$(0) oikein, vaan sen asemesta luetaan nollajono (ei merkkejä). Siksi GET#-käskyä käytetään yleensä muodossa:

```
GET#2,A$:IF A$="" THEN A$=CHR$(0)
```

Kuva 4.13

INPUT#-käskyllä luettavan merkkijonon pituus voi olla korkeintaan 88 merkkiä. Jos vastaavassa PRINT#-käskyssä ei ole käytetty erotinta CHR$(13), lukemisessa saattaa tapahtua virhe (?STRING TOO LONG). Seuraavassa esimerkissä on tutkittu PRINT#-käskyssä käytettyjen erottimien vaikutusta tietojen lukemiseen.

```
10 A$="1":B$="12":C$="123"
20 OPEN 2,8,2,"0:KOE1,S,W"
30 PRINT#2,A$,B$,C$
40 CLOSE 2
50 OPEN 2,8,2,"0:KOE2,S,W"
60 PRINT#2,A$;B$;C$
70 CLOSE 2
80 OPEN 2,8,2,"0:KOE3,S,W"
90 PRINT#2,A$;CHR$(13);B$;CHR$(13);C$
100 CLOSE 2
110 OPEN 2,8,2,"0:KOE1,S,R"
120 INPUT#2,A$,B$,C$
130 PRINTA$;B$;C$
140 CLOSE2:PRINT
150 OPEN 2,8,2,"0:KOE2,S,R"
160 INPUT#2,A$,B$,C$
170 PRINTA$;B$;C$
180 CLOSE2:PRINT
190 OPEN 2,8,2,"0:KOE3,S,R"
200 INPUT#2,A$,B$,C$
210 PRINTA$;B$;C$
220 CLOSE2
```

Kuva 4.14

Tiedoston lopun selvittäminen

Tilamuuttuja ST, joka on varattu muuttuja, ilmoittaa viimeisimmän I/O-toiminnan tilan. Sarjaväylän tila antaa tietoa siitä, miten kommunikointi oheislaitteiden kanssa sujuu. Seuraavaan taulukkoon on koottu sarjaväylän kannalta tärkeät ST-muuttujan arvot ja niiden merkitykset.

ST	MERKITYS
0	kaikki on kunnossa
1	ajan ylitys kirjoitettaessa
2	ajan ylitys luettaessa
64	EOF, tiedoston loppu

Tavallisesti ST=0, mutta kun tiedoston viimeinen tieto on luettu ST saa arvon 64. Sen avulla tiedoston loppuminen on helppo selvittää. Seuraava esimerkki esittelee ST-muuttujan käyttöä tiedoston loppua haettaessa.

```
10 OPEN 3,8,3,"0:LOPPU,S,W"
20 FOR I=1 TO 10
30 PRINT#3,I
40 NEXT I
50 CLOSE 3
60 OPEN 3,8,3,"0:LOPPU,S,R"
70 INPUT#3,X
80 PRINTX,"ST = ";ST
90 IF ST=0 THEN 70
100 CLOSE 3
```

Kuva 4.15

ST-muuttujan arvo ei muutu näyttöön tulostettaessa. Jos tulostus tapahtuu näytön asemesta kirjoittimelle, ST:n arvo muuttuu. Myös virhekanavan lukeminen INPUT#-käskyllä vaikuttaa ST-muuttujan arvoon.

TIEDOSTON SULKEMINEN

Tiedosto suljetaan CLOSE-käskyllä.

Syntaksi:

```
CLOSE TN
```

Kuva 4.16

TN = looginen tiedostonumero, joka on annettu OPEN-käskyn yhteydessä

Tiedostot on ehdottomasti suljettava, kun niiden käsittely on päättynyt. Muista, että komentokanavan sulkeminen sulkee muutkin tiedostot. Siksi komentokanava on avattava ensimmäisenä ja suljettava viimeisenä.

5. LUKU

OHJELMATIEDOSTOT

Ohjelmatiedosto on tavallisin tiedosto. Ohjelmatiedostoon tallennetaan osa C64:n RAM- tai ROM-muistista. Ohjelmatiedosto koostuu useimmiten BASIC- tai konekielisestä ohjelmasta. Lisäksi ohjelmatiedostoon voidaan tallentaa tekstitietoa, esim. levykkeen sisällysluettelo.

Lataamisen ja tallentamisen ohella ohjelmatiedostoon voidaan myös kirjoittaa ja sitä voidaan lukea. Tämän tiedostotyypin hyödyntämiseksi pitää tietää, miten BASIC-ohjelma on tallennettu tietokoneen muistiin.

OHJELMATIEDOSTON RAKENNE

Ohjelmatiedosto ei ole muodostunut tietueista ja kentistä. Tiedoston kaksi ensimmäistä tavua sisältävät latausosoitteen ja loppuosa muodostuu tavuista, jotka ladataan muistiin.

BASIC-ohjelman latausosoite on $0801. Vähemmän merkitsevä tavu on $01 ja enemmän merkitsevä tavu on $08. Käytän tavuista 10-järjestelmän lukuina tunnuksia LSB ja MSB. Latausosoite saadaan 10-järjestelmään seuraavasti:

latausosoite = 256 * MSB + LSB = 256 * 8 + 1 = 2049

Jonkin konekielisen ohjelman latausosoite voi olla esimerkiksi $3520, joka saadaan 10-järjestelmään seuraavasti:

MSB = 3 * 16 + 5 = 53 LSB = 2 * 16 = 32
latausosoite = 256 * 53 + 32 = 13600

Ohjelman latausosoitteen selvittämiseksi voit käyttää levyaseman mukana tulleella levykkeellä olevaa ohjelmaa LOAD ADDR.

Tutkimme seuraavan pienen ohjelman avulla, miten BASIC-ohjelma on tallennettu tietokoneen muistiin.

```
10 PRINT 5*6
20 END
```

Kuva 5.1

Kirjoita seuraavat komennot suorassa toimintatilassa:

```
FOR X=2048 TO 2065:PRINT X,PEEK(X):NEXT
```

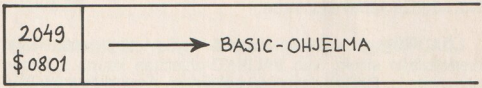
Kuva 5.2

Näyttöön tulostuu seuraavan taulukon kaksi vasemmanpuoleista saraketta.

OSOITE	SISÄLTÖ	TULKINTA
2048	0	1. tavu on aina 0
2049	11	linkkiosoittimen alatavu
2050	8	linkkiosoittimen ylätavu
2051	10	rivinumeron alatavu
2052	0	rivinumeron ylätavu
2053	153	PRINT
2054	32	välilyönti
2055	5	5
2056	42	kertomerkki *
2057	6	6
2058	0	1. rivin loppu
2059	17	linkkiosoittimen alatavu
2060	8	linkkiosoittimen ylätavu
2061	20	rivinumeron alatavu
2062	0	rivinumeron ylätavu
2063	128	END
2064	0	nolla
2065	0	nolla
2065	0	nolla

Muistipaikan 2048 arvo on aina 0. Seuraavat kaksi muistipaikkaa 2049–2050 osoittavat seuraavan ohjelmarivin alkuun: 8 * 256 + 11 = 2059. Muistipaikassa 2059 on seuraavaa riviä edeltävän linkin alatavu. Muistipaikoissa 2051–2052 on rivinumero 0 * 256 + 10 = 10. Muistipaikan 2053 arvo 153 on PRINT-käskysanan koodi. Ensimmäisen rivin lopun osoittaa muistipaikassa 2058 oleva arvo 0. Muistipaikoissa 2059–2060 olevat arvot osoittavat ohjelman loppuun 8 * 256 + 17 = 2065. Huomaa, että muista tiedostotyypeistä poiketen BASIC-ohjelman lopussa on aina kolme nollaa.

Muistiin tallennettu BASIC-ohjelma on kaaviokuvana seuraavanlainen:



Kuva 5.3

OHJELMAN LATAUS

Ohjelman lataus suoritetaan LOAD-käskyllä.

Syntaksi:

```
LOAD "NIMI",LN

tai

LOAD "NIMI",LN,KN
```

Kuva 5.4

NIMI = ladattavan ohjelman nimi
LN = laitenumero, tavallisesti 8
KN = kanavanumero (toisio-osoite), tavallisesti 1

Jos ohjelman nimessä on jokereita, ladataan ensimmäinen sopiva ohjelma. Konekielisiä ohjelmia ladattaessa on käytettävä kanavanumeroa 1. Muuten ohjelma latautuu muistipaikasta 2049 alkaen eikä alueelle, josta se on tallennettu. Kanavanumerona voi olla jokin muukin kokonaisluku 2–14.

Esimerkkejä:

```
LOAD "SUPERMAN",8

LOAD "SORT",8,1

10 T$="PUUHA"
20 LOAD T$,8
```

Kuva 5.5

Sisällysluettelo on myös eräänlainen ohjelmatiedosto. Sen latausosoite on $0401 (1025). Latauskäskyssä LOAD "$",8 ei saa käyttää kanavanumeroa.

LOAD-käskyä voidaan käyttää myös ohjelmassa. Näin menetellään silloin, kun VALIKKO-ohjelman kautta ladataan osaohjelmia, BASIC-ohjelmasta ladataan konekielisiä aliohjelmia tai laajan ohjelman osia linkitetään toisiinsa.

Ohjelman, jossa lataus suoritetaan, pitää olla ohjelmista pisin. Jos ladattava ohjelma on pitempi kuin alkuperäinen ohjelma, muuttujien arvot saattavat tuhoutua. Tämä johtuu siitä, että arvot on tallennettu RAM-muistissa välittömästi ohjelman perään. Kirjoita seuraavat ohjelmat siten, että ensin kirjoitat ja tallennat levykkeelle ohjelman 2 ja sitten kirjoitat ohjelman 1.

```
10 REM OHJELMA 1
20 REM OLEN PITEMPI KUIN OHJ.2
30 X=8
40 LOAD "OHJELMA 2",8


10 REM OHJELMA 2
20 PRINT X
```

Kuva 5.6

Koska jälkimmäinen ohjelma (2) on lyhyempi, tulostettava X:n arvo on oikea.

Jos jälkimmäinen ohjelma on pitempi, voidaan ohjelman loppua ilmaiseviin osoittimiin 45/46 sijoittaa pisintä ohjelmaa vastaavat arvot. Kirjoita seuraava ohjelma:

```
10 REM OHJELMA B
20 REM OLEN PITEMPI KUIN A
30 REM MUTTA TULOSTAN OIKEAN
40 REM Z:N ARVON
50 PRINT Z
```

Kuva 5.7

Kun ohjelma on kirjoitettu, selvität muistipaikkojen 45 ja 46 arvot käskyllä:

```
PRINT PEEK(45),PEEK(46)
```

Kuva 5.8

ja tallennat ohjelman B. Muistipaikkojen arvot ovat 97 ja 8.

Kirjoita ohjelma A, joka lataa ohjelman B.

```
10 REM OHJELMA A
20 POKE 45,97:POKE46,8:CLR
30 Z=24
40 LOAD "B",8
```

Kuva 5.9

Tulostettu Z:n arvo on oikea.

Jos ladattava ohjelma on lyhyempi, muuttujan arvot säilyvät. Poikkeuksena on ensimmäisessä ohjelmassa vakiona esitelty merkkijonomuuttuja, esim. 10 B$="TESTI". Merkkijonomuuttujilla käytetään osoitinta, joka viittaa muuttujan todelliseen arvoon (TESTI). Osoitin viittaa nyt ohjelmatekstiin. Seuraavaa ohjelmaa ladattaessa tämä osoitin ei muutu. Alkuperäisellä paikalla on nyt ladattu ohjelmateksti eikä muuttujan arvoa ole määritelty. Tämä voidaan välttää siirtämällä muuttujan arvona oleva teksti ylemmäksi RAM-muistiin. Kirjoittamalla esim. 10 B$="TESTI"+"" muuttujan arvo saadaan siirtymään merkkijonomuuttujille varatulle alueelle. Sama koskee käyttäjän määrittelemiä funktioita. Funktio on määriteltävä ladatussa ohjelmassa uudestaan.

Ohjelmasta käsin ladattu toinen ohjelma suoritetaan automaattisesti. Jos ladattava ohjelma on konekielinen, on vältettävä joutumista päättymättömään silmukkaan.

```
10 LOAD "APU",8,1
20 PRINT "VALMIS"
```

Kuva 5.10

Tässä esimerkissä konekielinen ohjelma ladataan toistuvasti uudelleen, koska DOS olettaa, että ladattava ohjelma on BASIC-kielinen. Tämä vältetään seuraavasti:

```
10 IF A=0 THEN A=1:LOAD "APU",8,1
```

Kuva 5.11

OHJELMAN TALLENNUS

Ohjelma tallennetaan SAVE-käskyllä.

Syntaksi:

```
SAVE "NIMI",LN
```

Kuva 5.12

NIMI = tallennettavan ohjelman nimi
LN = laitenumero, tavallisesti 8

Ohjelman nimessä ei saa olla jokereita * ja ? eikä kaksoispistettä, lainausmerkkejä tai pilkkua. Nimen pituus saa olla korkeintaan 16 merkkiä.

SAVE-käskyllä tallennetaan tavallisesti BASIC-ohjelmia muistipaikasta 2049 alkaen. Konekielinen ohjelma tai RAM-muistin alue pitää tallentaa konekielimonitorin tai assemblerin avulla. Toinen keino on muuttaa nollasivulla olevien muistipaikkojen 43/44 ja 45/46 sisältöä ennen tallentamista. Muistipaikat 43 ja 44 osoittavat BASIC-ohjelman alun ja muistipaikat 45 ja 46 osoittavat muuttujille varatun alueen alun (BASIC-ohjelman loppuosoite + 1). Jos tallennettava konekielinen ohjelma sijaitsee esim. alueella $C000 (49152) - $C100 (49408), muistipaikkoihin sijoitetaan seuraavat arvot:

```
POKE44,192:POKE43,0:POKE46,193:POKE45,0
```

Kuva 5.13

koska 49152 = 192 * 256 ja 49408 = 193 * 256

Tämän jälkeen annetaan käsky

```
SAVE "RUTIINI",8
```

Kuva 5.14

Tallennuksen jälkeen osoittimien arvot pitää palauttaa ennalleen.

Vaihtoehtoisesti voidaan käyttää ROM-rutiineja. Jos alueen alkuosoite on ALKU ja loppuosoite on LOPPU, alue voidaan tallentaa seuraavasti:

```
SYS57812"NIMI",8:POKE193,AL:POKE194,AH:P
OKE174,LL:POKE175,LH:SYS62954
```

Kuva 5.15

AH = INT (ALKU/256) AL = ALKU – 256 * AH
LH = INT (LOPPU/256) LL = LOPPU – 256 * LH

Näytön (1024–2024) voi tallentaa seuraavasti:

```
SYS57812"SCREEN",8:POKE193,0:POKE194,4:P
OKE174,232:POKE175,7:SYS62954
```

Kuva 5.16

Seuraavan ohjelman avulla voit tallentaa myös konekielisen ohjelman. (Ohjelma on kirjan levykkeellä nimellä SAVEML).

```
1000 FOR I=1 TO LEN(N$):A$=MID$(N$,I,1):
POKE 827+I,ASC(A$):NEXT I
1010 POKE 183,LEN(N$):POKE 187,60:POKE 1
88,3
1020 AH=INT(AO/256):AL=AO-256*AH:POKE 16
8,AL:POKE 169,AH
1030 LH=INT((LO+1)/256):LL=LO+1-256*LH:P
OKE 781,LL:POKE 782,LH
1040 POKE 780,168:POKE 184,8:POKE 185,8:
```

```
POKE 186,8
1050 SYS 65496
1060 RETURN

READY.
```

Kuva 5.17

Muuttujat:

N$ = ohjelman nimi
AO = tallennettavan alueen alkuosoite
LO = tallennettavan alueen loppuosoite
AL = alkuosoitteen alatavu
AH = alkuosoitteen ylätavu
LL = loppuosoitteen alatavu
LH = loppuosoitteen ylätavu

Ohjelman nimi tallennetaan kasettipuskuriin muistipaikasta 828 ($033C) alkaen. Sinne sijoitetaan nimen jokaisen kirjaimen ASCII-koodi. Muistipaikat ja niihin sijoitettavat arvot ovat

183 = nimen pituus
187 = sen osoitteen alatavu, jossa nimi sijaitsee
188 = sen osoitteen ylätavu, jossa nimi sijaitsee
168 = alkuosoitteen alatavu
169 = alkuosoitteen ylätavu
184 = looginen tiedostonumero
185 = laitenumero
186 = kanavanumero (toisio-osoite)
781 = osoitteen loppuosoite + 1 alatavu
782 = osoitteen loppuosoite + 1 ylätavu
780 = osoite, jossa alkuosoite sijaitsee nollasivulla, tässä tapauksessa 168

Ohjelma on kirjoitettu aliohjelmaksi. Ennen aliohjelmaan siirtymistä tulee ilmoittaa ohjelman nimi N$, alkuosoite AO ja loppuosoite LO.

SAVE-käskyä voidaan käyttää myös ohjelmassa, vaikka hyöty ei ole suuri. SAVE tallentaa ohjelman, jossa sitä on käytetty.

Esimerkiksi:

```
10 N$="LOISTOPELI"
20 SAVE N$,8
```

Kuva 5.18

Jos BASIC-ohjelmasta käsin tallennetaan jokin konekielinen ohjelma tai RAM-muistin alue, on ensin tallennettava muistipaikkojen 43/44 ja 45/46 sisältämät arvot turvallisiin paikkoihin. Sitten muistipaikkoihin 43/44 sijoitetaan tallennettavan ohjelman alkuosoite (ala/ylä) ja muistipaikkoihin 45/46 loppuosoite+1. Kun osoitteet on sijoitettu, kirjoitetaan käsky SAVE''NIMI'',8. Välittömästi tämän jälkeen palautetaan alkuperäiset arvot muistipaikkoihin 43/44 ja 45/46.

Ohjelman latauksen tai tallennuksen voi tarkistaa VERIFY-käskyllä.

Syntaksi:

```
VERIFY "NIMI",LN,KN
```

Kuva 5.19

NIMI = tarkistettavan ohjelman nimi
LN = laitenumero, tavallisesti 8
KN = kanavanumero (toisio-osoite), tavallisesti 1.

Ohjelman nimessä saa käyttää jokereita.

Konekielistä ohjelmaa tarkistettaessa kanavanumero on muistettava, muuten levykkeellä olevaa versiota verrataan muistipaikasta 2049 alkavan alueen sisältöön ja seurauksena on todennäköisesti virheilmoitus ?VERIFY ERROR.

Usein ohjelman nimi unohtuu. Sen voi silloin ladata suoraan sisällysluettelosta viemällä kohdistimen ko. ohjelman riville, kirjoittamalla nimen eteen LOAD, viemällä kohdistimen oikealle nimen yli ja kirjoittamalla sinne ,8: tai ,8,1. Tätä lataustoimintoa voidaan helpottaa, kun ohjelman tallentaminen suoritetaan toisella tavalla. Samalla menetelmällä voidaan ohjelman sisällysluettelossa näkyvään nimeen lisätä tietoa ohjelmasta, esim. latausosoite. Katsotaanpa!

Tallenna ohjelma siten, että lisäät ohjelman nimen jälkeen seuraavat merkit ennen toista lainausmerkkiä, pilkkua ja laitenumeroa 8:

SHIFT + välil. Commodore + D 8 SHIFT + @

Sisällysluettelossa ohjelman nimessä on tavalliseen tapaan lainausmerkit ja pilkku, 8 sekä kaksoispiste lainausmerkkien ulkopuolella. Lataamisen voi suorittaa sisällysluettelosta kirjoittamalla ohjelman nimen eteen LOAD ja painamalla RETURN. Jos kysymyksessä on konekielinen ohjelma, nimen jälkeen pitää kirjoittaa merkit

SHIFT + välil. Commodore + D 8 Comm. + D 1

Sisällysluettelossa näkyy nyt ohjelman nimen kohdalla "NIMI",8,1. Samoilla näppäilyillä voidaan esimerkiksi RENAME-komennon yhteydessä korvata pilkku ja kaksoispiste. Ohjelman nimen perään voidaan sisällysluetteloon lisätä esim. viesti siten, että nimen ja viestin väliin näppäillään SHIFT+välilyönti.

KORVAAVA TALLENNUS

Korvaavalla tallentamisella tarkoitetaan sitä, että ohjelma tallennetaan nimellä, joka on ennestään levykkeen sisällysluettelossa.

Syntaksi:

```
SAVE "@0:NIMI",LN,KN
```

Kuva 5.20

NIMI = ohjelman nimi
LN = laitenumero, tavallisesti 8
KN = kanavanumero (toisio-osoite), tavallisesti 1

Huolehdi siitä, että levykkeellä on tarpeeksi tilaa ohjelman uutta versiota varten. Vanhaa ohjelmaa ei poisteta levykkeeltä ennen kuin uusi on tallennettu. Käskyn suoritus keskeytyy, jos levykkeellä ei ole riittävästi tilaa.

Jos tiedosto ei ole suljettu (sisällysluettelossa *PRG), korvaavaa SAVE-käskyä ei voida käyttää. Tällaiset tiedostot pitää poistaa VALIDATE-komennolla.

Tämä käsky ei aina toimi kunnolla. Joskus tallennettava ohjelma korvaa väärän ohjelman, joskus ohjelma ei korvaudu täysin vaan sekoittuu osittain johonkin toiseen ohjelmaan. Jos SAVE"@0:"-käskyä käytetään monta kertaa peräkkäin samalla levykkeellä, useat ohjelmat korvautuvat yhdellä ja samalla

ohjelmalla. Asiantuntijat väittävät, että ROM-listauksissa ei ole virheitä, ja kuitenkin virheitä esiintyy. Joillakin käyttäjillä käsky toimii aina moitteettomasti. Mielestäni jokainen voi itse päättää ottaako riskin vai ei. Jos epäilee käskyn toimivuutta, on paras tuhota vanha versio SCRATCH-komennolla ja käyttää sitten normaalia SAVE-käskyä.

OHJELMATIEDOSTON KIRJOITTAMINEN JA LUKEMINEN

Ohjelmatiedoston voi avata kirjoittamista tai lukemista varten. Tällaisen ohjelmatiedoston avulla voidaan luoda uusi ohjelma tai muokata entistä ohjelmaa.

Syntaksi:

```
OPEN TN,LN,KN,"Ø:NIMI,PRG,TILA"
```

Kuva 5.21

TN = looginen tiedostonumero (1–127)
LN = laitenumero (8–11), tavallisesti 8
KN = kanavanumero (toisio-osoite) (2–14)
NIMI = tiedoston nimi, jossa saa olla korkeintaan 16 merkkiä
PRG = ohjelmatiedoston tyypin tunnus, joka voidaan lyhentää kirjaimella P
TILA = toimintatila, joka on tavallisesti READ tai WRITE, lyhenteet R tai W

Tiedoston tunnusosan "NIMI,PRG,TILA" voi sijoittaa osittain tai kokonaan merkkijonomuuttujan arvoksi. Jos tilaa ei ilmoiteta, sen oletusarvona on READ. READ-tilassa ohjelman nimessä saa käyttää jokereita. WRITE-tilassa jokereiden käyttö on kielletty.

Tilaa APPEND (A) käytetään silloin, kun tiedoston perään halutaan kirjoittaa lisää tietoa. Näin esim. BASIC-ohjelman perään voidaan liittää konekielinen ohjelma. Tätä tilaa käsitellään seuraavassa luvussa.

Tilaa MODIFY (M) käytetään sellaisen tiedoston lukemiseen, jota ei ole suljettu kunnolla. Näin osa tiedostosta voidaan mahdollisesti pelastaa. Kaikkea ei voida pelastaa, koska viimeistä tietolohkoa ei ole kirjoitettu levykkeelle. Tätäkin tilaa käsitellään seuraavassa luvussa tarkemmin.

Samanaikaisesti auki olevilla tiedostoilla pitää olla eri kanavanumerot ja loogiset tiedostonumerot. Ohjelmatiedostoja voi olla kerralla auki korkeintaan kolme.

Tiedoston käsittelyn (avaamisen, kirjoittamisen/lukemisen ja sulkemisen) yhteydessä kannattaa lukea levyaseman virheilmoitus. Komentokanava on paras avata ensimmäiseksi ja sulkea viimeiseksi, koska komentokanavan sulkeminen sulkee myös muut tiedostot.

Olemassa olevan tiedoston voi korvata samannimisellä tiedostolla joko poistamalla (SCRATCH) vanhan tiedoston ja kirjoittamalla uuden, tai käyttämällä avauskäskyssä muotoa

```
OPEN 3,8,3,"@0:NIMI,P,W"
```

Kuva 5.22

Seuraavan BASIC-ohjelman avulla voidaan levykkeelle luoda konekielinen ohjelma. Tämä ohjelma on kirjan levykkeellä nimellä LUOML.

```
100 REM LUOML
110 :
120 REM AVATAAN OHJELMATIEDOSTO
130 OPEN 15,8,15,"I0":CLOSE 15
140 OPEN 2,8,2,"NIMI,P,W"
150 :
160 REM KIRJOITETAAN ALKUOSOITE
170 PRINT#2,CHR$(AL)CHR$(AH);
180 :
190 REM LUETAAN TIEDOT/KIRJOITETA
200 REM N = DATOJEN LKM
210 FOR X=1 TO N
220 READ TAVU
230 PRINT#2,CHR$(TAVU);
240 NEXT
250 :
260 REM SULJETAAN TIEDOSTO
270 CLOSE 2
280 :
290 END
300 :
310 REM DATAT
320 DATA ...
```

Kuva 5.23

Rivillä 170 kirjoitetaan ohjelman alkuosoite järjestyksessä alatavu AL, ylätavu AH. Jos alkuosoite on AO, AL ja AH saadaan seuraavasti:

AH = INT(AO/256) AL = AO – 256 * AH

Huomaa rivin 170 lopussa oleva puolipiste. Riveillä 210–240 tavut luetaan yksitellen ja kirjoitetaan levykkeelle. Tavujen lukumäärä on N ja ne on sijoitettu 10-järjestelmän lukuina riviltä 320 alkaville DATA-riveille.

Levylle luotu ohjelma ladataan muistiin tavalliseen tapaan käskyllä LOAD "NIMI",8,1.

Levykkeen sisällysluettelo on myös eräänlainen ohjelmatiedosto ja sen nimi on $. Sisällysluettelotiedosto voidaan avata vain lukemista varten. Avaaminen suoritetaan käyttämällä kanavanumerona tiedonsiirtokanavia 2–14 tai DOS:in ohjelman lataamiseen varaamaa kanavaa 0. Seuraavassa ohjelmassa on käytetty kanavaa 0, ja sisällysluettelo tulostuu samalla tavalla kuin normaaleilla komennoilla LOAD"$",8:LIST. Tämä ohjelma on kirjan levykkeellä nimellä SISLU 1.

```
1000 REM SISLU 1
1010 PRINT"□";
1020 OPEN 1,8,0,"$0"
1030 GET#1,A$,A$
1040 GET#1,A$,A$
1050 GET#1,A$,B$
1060 A$=A$+CHR$(0):B$=B$+CHR$(0)
1070 BL=ASC(A$)+ASC(B$)*256
1080 IF ST<>0 THEN 1150
1090 PRINTBL;
1100 GET#1,A$:IFA$=""THEN1120
1110 PRINTA$;:GOTO1100
1120 PRINT:GETA$:IFA$=""THEN1040
1130 GETA$:IFA$=""THEN1130
1140 GOTO1040
1150 CLOSE1:PRINT"PAINA JOTAKIN";
1160 GETA$:IFA$=""THEN1160
1170 END
```

Kuva 5.24

Ohjelman tulostus voidaan keskeyttää painamalla jotain näppäintä ja tulostus saadaan jatkumaan painamalla näppäintä

uudestaan. Jos haluat sisällysluettelosta määrätyntyyppiset ohjelmat, muutat avauskäskyä. Kaikki ohjelmatiedostot saadaan, kun avaaminen suoritetaan seuraavasti:

```
OPEN 1,8,0,"$0:*=P"
```

Kuva 5.25

Kaikki S:llä alkavat ohjelmat saadaan seuraavasti:

```
OPEN 1,8,0,"$0:S*"
```

Kuva 5.26

Seuraava ohjelma tulostaa sisällysluettelosta vain ohjelmien nimet. Ohjelma on kirjan levykkeellä nimellä SISLU 2.

```
100 REM SISLU 2
110 PRINT"";
120 OPEN 1,8,0,"$0"
130 GET#1,A$,A$
140 IF A=2 THEN A=0:PRINT
150 PRINTTAB(20*A);:FOR Y=1 TO 4:GET#1,A
$:NEXT
160 IF ST<>0 THEN CLOSE 1:END
170 GET#1,A$:IF A$="" THEN A=A+1:GOTO 14
0
180 IF A$=CHR$(34) THEN B=NOTB:GOTO 170
190 IF B THEN PRINT A$;
200 GOTO 170
```

Kuva 5.27

Esitetyt ohjelmat on helppo sijoittaa aliohjelmiksi omiin ohjelmiin. Jos ohjelmaa kirjoitettaessa levykkeen sisällysluettelo halutaan näkyviin, toimitaan esimerkiksi seuraavasti, jotta sisällysluettelo ei tule kirjoitettavan ohjelman päälle.

```
POKE44,PEEK(46)+1   muuta BASICin alku
LOAD "$",8          lataa sis.luettelo
LIST                sis.luettelo näytölle
POKE 44,8           muuta osoitin
```

Kuva 5.28

Ennen LOAD- ja LIST-käskyjä BASIC-ohjelman alkuun osoittava osoitin siirretään osoittamaan BASIC-ohjelman loppuun. Lopussa osoitin palautetaan ennalleen. Muista painaa jokaisen rivin jälkeen RETURN-näppäintä.

6. LUKU

PERÄKKÄISTIEDOSTOT

Tiedostojen käsittely laajentaa levyaseman käyttömahdollisuuksia ohjelmien lataamisen ja tallentamisen ulkopuolelle. Peräkkäistiedosto on yksinkertaisin ja helppokäyttöisin tiedostotyyppi. Nämä ominaisuudet ovat varsinkin aloittelijalle tärkeitä. Peräkkäistiedostot käyttävät levyketilaa tehokkaasti. Kukin tiedosto on juuri niin laaja kuin siihen tallennetut tiedot. Tiedot kirjoitetaan peräkkäistiedostoon peräkkäin. Tämä ominaisuus supistaa niiden käyttömahdollisuutta.

JOHDANTO

Tiedosto jaetaan tietueisiin ja edelleen kenttiin. Kenttä voidaan jakaa yksittäisiin merkkeihin. Käytettävissä oleva muistitila rajoittaa tietueiden lukumäärää ja kenttien lukumäärää tietueessa. Kentän pituutta ei periaatteessa rajoita mikään, mutta tehokkaan ja nopean työskentelyn kannalta kentän ei tule olla yli 88 merkkiä, joka on INPUT#-käskyllä luettavan merkkijonon suurin mahdollinen pituus. Merkkijonossa voi muuten olla 256 merkkiä.

Seuraava kuva esittää tiedoston yksinkertaista rakennetta:

TIEDOSTO									
TIETUE 1			TIETUE 2			TIETUE 3			
KENTTÄ 1	KENTTÄ 2	KENTTÄ 3	KENTTÄ 1	KENTTÄ 2	KENTTÄ 3	KENTTÄ 1	KENTTÄ 2	KENTTÄ 3	

Kuva 6.1

Yksinkertaisin postitustiedosto muodostuu tietueista, joissa on kaksi kenttää: nimi ja osoite.

TIETUE 1	TIETUE 2	TIETUE 3	TIETUE 4	...
kenttä 1	kenttä 1	kenttä 1	kenttä 1	...
kenttä 2	kenttä 2	kenttä 2	kenttä 2	...

Jos tähän kuviteltuun tiedostoon kirjoitetaan nimiä, tiedoston alkupää näyttää yksinkertaistettuna seuraavalta:

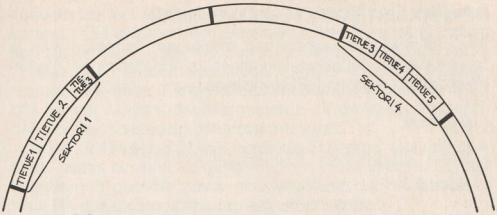
MATTI * HML * LIISA * HKI * PEKKA * TRE * JUSSI-PEKKA * HKI * . . .

Kertomerkki * kuvaa merkkiä CHR$ (13) (RETURN), joka tässä tapauksessa toimii erottimena sekä kenttien että tietueiden välillä.

Halutessamme muuttaa tietueen tietoja tai poistaa tietueen, meidän pitää ensin lukea sitä edeltävät tietueet. Kun korjaus tai poisto on suoritettu, koko tietueet pitää lukea ennen tiedoston uudelleenkirjoittamista. Tämä asettaa rajoituksia peräkkäistiedostojen käytölle, sillä varsinaisen ohjelman lisäksi C64:n muistissa ei voida pitää kovin suuria tiedostoja. Koko tiedostoa ei tietenkään tarvitse kerralla lukea muistiin, vaan esim. tietueetta poistettaessa tietue, jota ei poisteta, voidaan välittömästi kirjoittaa takaisin levykkeelle. Näin muodostuva uusi tiedosto avataan kirjoittamista varten.

Jos käytettävissä on yksi levyasema, tiedoston koko ei voi olla yli 85 K, sillä molempien versioiden tulee mahtua samalle levykkeelle.

Levykkeelle peräkkäistiedosto tallennetaan sektoreittain sektorin alusta alkaen. Yhdeksännessä luvussa käsitellään yksityiskohtaisesti tiedostojen tallennusta levykkeelle. Tiedosto näyttää yksinkertaisesti esitettynä seuraavalta:



Kuva 6.2

Peräkkäistiedostoja kannattaa käyttää, jos tallennettavia tietoja on suhteellisen vähän. Jos tiedosto vaatii vähän ja harvoin suoritettavia päivitystoimenpiteitä, se voi olla kooltaan suurempikin, mutta ylärajana on 85 K.

Peräkkäistiedostoja ei kannata käyttää, jos tietueita pitää usein muuttaa tai poistaa, tai jos niitä pitää voida järjestää esim. kenttien mukaan.

PERÄKKÄISTIEDOSTOJEN KÄSITTELY

Peräkkäistiedoston käsittelyssä on aina seuraavat vaiheet:

1) tiedoston avaaminen;

2) tiedoston päivittäminen (tietojen lukeminen, kirjoittaminen tai muuttaminen);

3) tiedoston sulkeminen.

Tiedoston yksityiskohtainen rakenne on suunniteltava ennen edellisiä toimenpiteitä. Ainakin seuraavat asiat on syytä selvittää:

1) Käyttäjän mahdollisuudet tiedoston päivittämiseksi

2) Tietueiden määrä (kiinteä tai käyttäjän määräämä)

3) Tietueen pituus

4) Kenttien lukumäärä

5) Kenttien nimet

6) Kentän tietotyyppi (numeerinen tai aakkosnumeerinen)

7) Tiedoston vaatima tila (kohdat 2–3)

PERÄKKÄISTIEDOSTON AVAAMINEN

Peräkkäistiedostoa avattaessa ilmoitetaan tiedoston nimi ja toimintatila eli tiedoston lukeminen tai kirjoittaminen. Peräkkäistiedosto avataan OPEN-käskyllä.

Syntaksi:

```
OPEN TN,LN,KN,"0:NIMI,SEQ,TILA"
```

Kuva 6.3

TN = looginen tiedostonumero (1–127)
LN = laitenumero (8–11), tavallisesti 8
KN = kanavanumero (2–14)
NIMI = tiedoston nimi
SEQ = tiedoston tyyppi, joka voi olla pelkkä S
TILA = toimintatila, WRITE tai READ (W tai R)

Nimi, tyyppi ja tila voidaan asettaa merkkijonomuuttujan sisällöksi. Tiedoston tyyppi ja toimintatila voidaan jäättää pois. Jos tyyppi jätetään pois, DOS tarkistaa sen sisällysluettelosta. Jos tila jätetään pois, oletusarvona on READ.

W- ja R-tilojen lisäksi on olemassa tilat APPEND (A) ja MODIFY (M). A-tilaa käyttämällä olemassa olevan tiedoston perään voidaan lisätä tietueita edellisiä tietueita lukematta. M-tilan avulla voidaan lukea sellainen tiedosto, jota ei ole kunnolla suljettu.

Tiedoston nimessä saa käyttää jokereita (* ja ?) vain, jos kysymyksessä on tiedoston lukeminen (READ).

Seuraavassa on joitakin esimerkkejä avauskäskystä.

```
1) 10 OPEN 2,8,2,"0:TIED1,SEQ,WRITE"

2) 10 OPEN 1,8,3,"0:PELIT,S,R"

3) 10 N$="KIRJAT"
   20 T$="0:"+N$+",S,W"
   30 OPEN 3,8,3,T$
```

Kuva 6.4

Kolmannessa esimerkissä on kaikki tiedoston tunnistamiseen tarvittava tieto koottu muuttujan T$ sisällöksi.

Ohjelmoijan tulee huolehtia virheilmoituksen lukemisesta tiedoston avaamisen jälkeen. Näin voidaan varmistaa, että

tiedosto on auki eivätkä kirjoitettavat tiedot huku. Komentokanava pitää avata ensimmäisenä ja sulkea viimeisenä, koska sen sulkeminen sulkee myös muut avoinna olevat tiedostot.

Avaamisen jälkeen tiedostoon viitataan loogista tiedostonumeroa käyttäen. Kahdella avoinna olevalla tiedostolla ei voi olla samaa loogista tiedostonumeroa. Myös kanavanumeroiden (toisio-osoitteiden) pitää olla erisuuret.

KORKEINTAAN KOLME peräkkäistiedostoa saa olla samalla kertaa avoinna. Jokainen avoinna oleva tiedosto käyttää levyaseman puolella omaa muistialuettaan eli ns. puskuria. Neljästä puskurista kolme on tiedostoja varten.

KIRJOITTAMINEN PERÄKKÄISTIEDOSTOON

Tiedot kirjoitetaan PRINT#-käskyllä, joka voidaan lyhentää P + SHIFT R (EI ?#).

Syntaksi:

```
PRINT#TN,ML
```

Kuva 6.5

TN = looginen tiedostonumero (sama, jota on käytetty tiedostoa avattaessa)
ML = muuttujalista

Muuttujalistassa ML voi olla yksi tai useampia muuttujia. Koska jokainen muuttuja edustaa yhden kentän sisältöä ja kentät pitää luettaessa erottaa toisistaan, on huolehdittava kenttien (ja tietueiden) välisen erottimen kirjoittamisesta. Erottimena kannattaa käyttää merkkiä CHR$ (13) (RETURN). Erottimen vaikutusta tietojen lukemiseen selvitellään kohdassa peräkkäistiedoston lukeminen.

Tietoja kirjoitettaessa voidaan menetellä kahdella tavalla. Tieto voidaan kirjoittaa levykkeelle välittömästi sen jälkeen, kun se on annettu näppäimistöltä tai kaikki tiedot voidaan koota taulukkoon, joka kirjoitetaan kerralla levykkeelle.

Seuraavassa esimerkissä tiedot kirjoitetaan levykkeelle heti näppäinsyötön jälkeen. Jos tietueiden lukumäärää ei tarkisteta esim. laskurimuuttujan tai toistorakenteen avulla, kirjoittamisen lopettamisesta on ilmoitettava jonkin erikoismerkin tai -sanan avulla.

```
100 REM PRINT# - 1
110 PRINT"":OPEN 15,8,15
120 OPEN 2,8,2,"0:KIRJAT,S,W"
130 INPUT#15,VN,VI$,VU,VS
140 IF VN<>0 THEN PRINT VN;VI$;VU;VS:CLO
SE 2:CLOSE 15:STOP
150 PRINT"KIRJAILIJAN NIMI (*=LOPETUS)"
160 INPUT K$
170 IF K$="*" THEN 220
180 INPUT "KIRJOITA TEOS ";T$
190 PRINT#2,K$
200 PRINT#2,T$
210 GOTO 150
220 CLOSE 2:CLOSE 15:END
```

Kuva 6.6

Esimerkissä muodostetaan tiedosto, jonka tietueissa on kaksi kenttää: kirjailija ja teos. Kenttien ja tietueiden erottimena on CHR$ (13), joka tässä kirjoittuu automaattisesti. Jos molemmat kentät kirjoitetaan samalla PRINT#-käskyllä, on rivi 200 poistettava ja rivi 190 kirjoitettava muotoon

```
190 PRINT#2,K$;CHR$(13);T$
```

Kuva 6.7

Erottimena oleva pilkku (,) on sijoitettava lainausmerkkeihin

```
190 PRINT#2,K$","T$
```

tai

```
190 PRINT#2,K$;",";T$
```

Kuva 6.8

Yleensä on edullisempaa koota tiedot taulukkoon ennen levylle kirjoittamista. Näin kannattaa menetellä erikoisesti silloin, kun tietueiden pitää olla levykkeellä määrätyssä järjestyksessä (esim. aakkosjärjestyksessä). Tässä menetelmässä kaikkien tietueiden pitää kerralla mahtua tietokoneen muistiin. Taulukkoa käytettäessä tietueiden lukumäärän pitää olla tiedossa.

Seuraavassa esimerkissä kirjoitetaan 10 tietuetta FOR . . . NEXT-rakennetta käyttäen.

```
100 REM PRINT# - 2
110 PRINT"":OPEN 15,8,15
120 FOR N=1 TO 10
130 INPUT "KIRJAILIJAN NIMI ";K$(N)
140 INPUT "KIRJOITA TEOS ";T$(N)
150 NEXT N
160 OPEN 2,8,2,"0:KIRJAT,S,W"
170 INPUT#15,VN,VI$,VU,VS
180 IF VN>19 THEN PRINT VN;VI$;VU;VS:CLO
SE 2:CLOSE 15:STOP
190 FOR N=1 TO 10
200 PRINT#2,K$(N)
210 PRINT#2,T$(N)
220 NEXT N
230 CLOSE 2:CLOSE 15:END
```

Kuva 6.9

Edellisessä esimerkissä voidaan käyttää myös kaksiulotteista taulukkoa K$ (10,2).

PERÄKKÄISTIEDOSTON LUKEMINEN

Tiedoston sisältö voidaan lukea joko GET#- tai INPUT#-käskyllä. GET#-käskyä ei voida kirjoittaa lyhennettynä. INPUT# lyhennetään I + SHIFT N.

Syntaksi:

```
INPUT#TN,ML


GET#TN,ML
```

Kuva 6.10

TN = looginen tiedostonumero (sama, jota on käytetty tiedostoa avattaessa)
ML = muuttujalista

Muuttujalistassa ML voi olla yksi tai useampia muuttujia. Jos muuttujia on useita, ne erotetaan toisistaan pilkulla.

Peräkkäistiedostoja luettaessa käytetään yleensä INPUT#-käskyä, koska näin voidaan kerralla lukea enemmän. GET#-käskyllä luetaan vain yksi merkki kerrallaan.

Joskus on kuitenkin pakko käyttää GET#-käskyä. Jos kentän pituus on yli 88 merkkiä, INPUT#-käskyä ei voida käyttää, koska tällöin luettavan merkkijonon pituus voi olla korkeintaan 88 merkkiä. Jos merkkijono sisältää erikoismerkkejä, kuten pilkun, kaksoispisteen tai nollajonon, käytetään GET#-käskyä. Jos tiedoston tarkkaa sisältöä ei tunneta, sen tutkiminen on paras suorittaa lukemalla tiedostoa merkki kerrallaan GET#-käskyllä.

Tiedoston lukeminen aloitetaan aina tiedoston alusta. Jos vahingossa luet tarvitsemasi tiedon ohi, et voi palata takaisin, vaan tiedosto on suljettava ja lukeminen aloitettava alusta.

Seuraavassa luetaan kirjailijatiedostoa, kunnes oikea kirjailija löytyy. Ohjelmassa käytetään INPUT#-käskyä. Ohjelma tulostaa kirjailijan kaikki teokset.

```
100 REM INPUT# - 1
110 PRINT"":OPEN 15,8,15
120 OPEN 2,8,2,"0:KIRJAT,S,R"
130 INPUT#15,VN,VI$,VU,VS
140 IF VN<>0 THEN PRINT VN;VI$;VU;VS:CLO
SE 2:CLOSE 15:STOP
150 INPUT"KIRJAILIJA (* = LOPPU) ";KN$
160 IF KN$="*" THEN 240
170 INPUT#2,K$,T$
180 IF KN$=K$ THEN FL=1:PRINT"TEOS ON "T
$
190 IF ST=0 THEN 170
200 IF FL=0 THEN PRINT"KIRJAILIJA EI OLE
 TIEDOSTOSSA"
210 PRINT:PRINT"HAENKO MUITA (K/E)?"
220 GETR$:IF R$="" THEN 220
230 IF R$="K" THEN FL=0:CLOSE 2:GOTO 120
240 CLOSE 2:CLOSE 15:END
```

Kuva 6.11

Tätä tapaa on käytettävä silloin, kun koko tiedosto ei mahdu kerralla tietokoneen muistiin.

Muuttuja FL toimii lippuna siten, että sen arvo on 1, jos annetun kirjailijan teos on tiedostossa, muuten sen arvo on 0. Rivillä 190 testataan tiedoston loppu tilamuuttujan (ST) avulla. Rivi 190 voidaan kirjoittaa myös

```
190 IF ST<>64 THEN 170
```

Kuva 6.12

Tiedosto voidaan lukea myös kokonaan tietokoneen muistiin ja sitten etsitään tarvittava tietue. Jos tietueiden lukumäärä tiedetään, lukeminen voidaan suorittaa FOR . . . NEXT-toistorakenteen avulla, muuten on käytettävä laskuria. Seuraavassa on käytetty FOR . . . NEXT-rakennetta.

```
100 REM INPUT# - 2
110 PRINT"":DIM K$(50),T$(50)
120 OPEN 15,8,15
130 OPEN 3,8,3,"0:KIRJAT,S,R"
140 INPUT#15,VN,VI$,VU,VS
150 IF VN<>0 THEN PRINT VN;VI$;VU;VS:CLO
SE 3:CLOSE 15:STOP
160 INPUT"KUINKA MONTA TIETUETTA ";T
170 IF T<1 OR T>50 THEN 160
180 FOR N=1 TO T
190 INPUT#3,K$(N),T$(N)
200 NEXT:CLOSE 3
210 INPUT"KIRJAILIJA :";KN$
220 FOR N=1 TO T
230 IF K$(N)=KN$ THEN FL=1:PRINT "TEOS O
N: "T$(N)
240 NEXT N
250 IF FL=0 THEN PRINT"KIRJAILIJAA EI OL
E TIEDOSTOSSA !"
260 CLOSE 15:END
```

Kuva 6.13

Kummastakin kentästä on omat taulukot, joissa suurin mahdollinen indeksi on 50. Koko tiedosto luetaan tietokoneen muistiin riveillä 180–200.

Tietueiden lukumäärä voidaan sijoittaa myös varsinaiseen tiedostoon ennen tietueita. Tiedostosta luetaan silloin ensin tietueiden lukumäärä. Sen avulla suoritetaan taulukoiden esit-

tely (DIM) ja tietueet luetaan taulukkoon. Tiedoston alussa voi olla myös muuta tiedoston käsittelyssä tarvittavaa tietoa, esim. kenttien nimet. Tietueiden lukumäärää ja muuta tietoa varten voidaan myös käyttää erillistä tiedostoa.
Koska GET#-käskyllä luetaan merkki kerrallaan, yksittäiset merkit pitää liittää merkkijonoksi (esim. yhden kentän sisältö) käyttämällä sijoitusta A$ = A$ + R$, jossa R$ on luettava merkki ja A$ lopullinen merkkijono. Tämä sijoitus aiheuttaa osajonojen tarpeettoman kasaantumisen muistiin. Muistin täyttyessä tietokone joutuu suorittamaan ns. muistisiivouksen (garbage collection), jolloin se poistaa tarpeettomat merkkijonot muistista. Jos tietokone lukkiutuu (kohdistin katoaa) eikä ota vastaan komentoja, se todennäköisesti "siivoaa". Tämä toiminta vie joskus aikaa.

Seuraavassa esimerkissä tiedosto luetaan GET#-käskyn avulla.

```
100 REM GET#
110 PRINT"":DIM K$(50),T$(50):N=0:R$=CHR
$(13)
120 OPEN 15,8,15
130 OPEN 2,8,2,"0:KIRJAT,S,R"
140 INPUT#15,VN,VI$,VU,VS
150 IF VN<>0 THEN PRINT VN;VI$;VU;VS:CLO
SE 2:CLOSE 15:STOP
160 N=N+1:IF N>50 THEN 260
170 GET#2,N$:IF N$="" THEN N$=CHR$(0)
180 IF N$=R$ THEN 210
190 K$(N)=K$(N)+N$
200 GOTO 170
210 GET#2,T$:IF T$="" THEN T$=CHR$(0)
220 IF T$=R$ THEN 250
230 T$(N)=T$(N)+T$
240 GOTO 210
250 IF ST=0 THEN 160
260 CLOSE 2:CLOSE 15
270 INPUT"KIRJAILIJA (* = LOPPU)";KN$
280 IF KN$="*" THEN END
290 FOR I=1 TO N
300 IF KN$=K$(I) THEN FL=1:PRINT"TEOS :"
T$(I):PRINT
310 NEXT I
```

```
320 IF FL=0 THEN PRINT"KIRJAILIJA EI OLE
 TIEDOSTOSSA !":PRINT
330 FL=0:GOTO 270
```

Kuva 6.14

Laskurina on käytetty muuttujaa N. Ensimmäinen kenttä luetaan riveillä 170–200. Luettava merkki on N$, joka liitetään muuttujaan K$ (N), kunnes sen arvo on R$ = CHR$(13) (jolloin kenttä loppuu). Toisen kentän T$(N) sisältö luetaan riveillä 210–240. Tiedoston loppu testataan rivillä 250.

Seuraavien esimerkkien avulla voit tutkia erottimien vaikutusta tietojen lukemiseen.

```
100 REM KIRJOITA TESTIT
110 OPEN 15,8,15:R$=CHR$(13)
120 A$="ALKU":B$="AAMU":C$="ILTA"
130 A=123:B=456:C=789
135 :
140 N$="TESTI":GOSUB500
150 PRINT#2,A$;R$;B$;R$;C$;R$;A;R$;B;R$;
C
160 CLOSE 2
165 :
170 N$="TESTI1":GOSUB500
180 PRINT#2,A$;B$;C$;A;B;C
190 CLOSE 2
195 :
200 N$="TESTI2":GOSUB500
210 PRINT#2,A$","B$","C$","A","B","C
220 CLOSE 2
225 :
230 N$="TESTI3":GOSUB500
240 PRINT#2,A$,B$,C$,A,B,C
250 CLOSE 2:CLOSE 15:END
255 :
500 OPEN 2,8,2,"0:"+N$+",S,W"
510 INPUT#15,VN,VI$,VU,VS
520 IF VN<>0 THEN PRINT VN;VI$;VU;VS:CLO
SE 2:CLOSE 15:STOP
530 RETURN
```

```
100 REM
110 OPEN 15,8,15:PRINT""
120 INPUT" TESTIN NIMI ";N$
130 OPEN 2,8,2,"0:TESTI,S,R"
140 GOSUB 300
150 INPUT#2,A$,B$,C$,A,B,C
160 PRINT "<"A$"><"B$"><"C$"><"A"><"B"><
"C">"
170 CLOSE 2
180 OPEN 2,8,2,"0:"+N$+",S,R"
190 INPUT#2,A$,B$,C$,A,B,C
200 PRINT "<"A$"><"B$"><"C$"><"A"><"B"><
"C">"
210 CLOSE 2
220 CLOSE 15:END
300 INPUT#15,VN,VI$,VU,VS
310 IF VN<>0 THEN PRINT VN;VI$;VU;VS:CLO
SE 2:CLOSE 15:STOP
320 RETURN
```

Kuva 6.15

Ensimmäinen ohjelma muodostaa neljä tiedostoa, joihin samat kentät kirjoitetaan erilaisia erottimia käyttäen. TESTI-tiedosto toimii vertailutiedostona ja siinä on erottimena CHR$(13). TESTI-tiedoston lukeminen antaa oikeat kentät. Jälkimmäisen ohjelman avulla edellä luodut tiedostot luetaan ja tulostetaan näyttöön. Rivillä 120 pyydetään testitiedoston nimi. Riveillä 150–170 luetaan vertailutiedosto ja sen 6 kenttää tulostetaan näyttöön. Kentän oikea muoto nähdään merkkien <ja> avulla. Riveillä 180–210 luetaan käyttäjän pyytämä tiedosto (TESTI1, TESTI2 tai TESTI3) ja sen kentät tulostetaan TESTI-tiedoston kenttien alapuolelle. Kenttiä vertailtaessa huomataan, että vain tiedosto TESTI2 toimii oikein. Jos rivillä 190 kunkin muuttujan arvo luetaan omalla ohjelmarivillään, tulos on väärä. Kokeilepa! Syynä on se, että INPUT#-käsky lukee vain CHR$(13)-merkkiin asti. Virheiden välttämiseksi kenttien ja tietueiden välisenä erottimena kannattaa käyttää merkkiä CHR$(13).

PERÄKKÄISTIEDOSTON SULKEMINEN

Tiedosto pitää sulkea, kun lukeminen tai kirjoittaminen on päättynyt. Tiedosto suljetaan CLOSE-käskyllä.

Syntaksi:

```
CLOSE TN
```

Kuva 6.16

TN = looginen tiedostonumero (sama, jota on käytetty tiedostoa avattaessa)

Jos kirjoittamista varten avattu tiedosto jää sulkematta, osa tiedoista hukkuu. Sulkematon tiedosto voidaan pelastaa kirjan levykkeellä olevalla ohjelmalla (SULKEMATON), mutta siitä huolimatta osa tiedoista katoaa.

TIETUEIDEN LISÄÄMINEN **APPEND**-TILAN AVULLA

Peräkkäistiedoston loppuun voidaan lisätä tietueita käyttämällä tiedoston avauskäskyssä toimintatilaa APPEND, joka lyhennetään A. Tiedosto avataan seuraavasti:

Syntaksi:

```
OPEN TN,LN,KN,"0:NIMI,SEQ,A"
```

Kuva 6.17

TN = looginen tiedostonumero (1–127)
LN = laitenumero (8–11), tavallisesti 8
KN = kanavanumero (toisio-osoite) (2–14)
NIMI = tiedoston nimi
SEQ = peräkkäistiedoston tunnus, joka voi olla myös S
A = lisäystilan tunnus

Ennen A-tilan käyttämistä kannattaa tarkistaa, että lisättävät tiedot mahtuvat levykkeelle. Jos tila loppuu kesken, lisäystoiminta keskeytyy ja tiedosto jää auki. Kaikki vanhatkin tiedot katoavat, jollet käytä kirjan levykkeellä olevaa SULKEMATON -nimistä ohjelmaa.

Katsotaanpa esimerkkiä. Ohjelman alussa luodaan tiedosto, jossa on viisi tietuetta. Tämän jälkeen tiedostoon lisätään kolme tietuetta. Lopuksi koko tiedosto luetaan ja tulostetaan näyttöön.

```
100 REM APPEND
110 OPEN 15,8,15
120 OPEN 2,8,2,"0:MORE,S,W"
130 GOSUB 300
140 FOR I=65 TO 69
150 PRINT#2,CHR$(I)
160 NEXT:CLOSE 2
165 :
170 OPEN 2,8,2,"0:MORE,S,A"
180 GOSUB 300
190 FOR I=70 TO 72
200 PRINT#2,CHR$(I)
210 NEXT:CLOSE 2
215 :
220 OPEN 2,8,2,"0:MORE,S,R"
230 GOSUB 300
240 FOR I=1 TO 8
250 INPUT#2,A$
260 PRINT A$;
270 NEXT:CLOSE 2:CLOSE 15:END
300 INPUT#15,VN,VI$,VU,VS
310 IF VN<>0 THEN PRINTVN;VI$;VU;VS:CLOS
E 2:CLOSE 15:STOP
320 RETURN
```

Kuva 6.18

Näyttöön tulostuu ABCDEFGH. Lisäys on onnistunut.

Ohjelmatiedostoissa APPEND-tilaa voidaan käyttää konekielisen ohjelman lisäämiseksi BASIC-ohjelman perään.

UUDEN TIETUEEN SIJOITTAMINEN

Kun tietueet ovat määrätyssä järjestyksessä, esim. nimet aakkosten mukaan tai luvut suurimmasta pienimpään, uusi nimi tai luku on sijoitettava tarkasti oikeaan paikkaan. Tällöin ei yleensä kannata lukea kaikkia vanhoja tietueita muistiin ja suorittaa lajittelua uudestaan uuden tietueen ollessa mukana.

Yksinkertaisinta on etsiä uuden tietueen paikka, siirtää entisiä tietueita jonossa tai taulukossa ylöspäin ja sijoittaa uusi tietue syntyneeseen väliin. Oletetaan, että kussakin tietueessa on yksi numeerinen kenttä. Jos lukuja on esim. 500 ja ne on tallennettu yksiulotteiseen taulukkoon pienimmästä suurimpaan, etsiminen kannattaa aloittaa taulukon lopusta, koska

sijoitettavaa lukua suurempi vanha luku voidaan samalla siirtää taulukossa ylöspäin.

Ensimmäisessä esimerkissä luvun paikan etsiminen aloitetaan taulukon alusta. Sen jälkeen suoritetaan taulukon siirtäminen, ja uusi luku sijoitetaan kohdalleen.

```
100 DIM A(600):MAX=500
110 FOR I=1 TO MAX:A(I)=I:NEXT
120 INPUT"ANNA UUSI LUKU ";UL
130 GOSUB 200:REM HAE PAIKKA
140 GOSUB 300:REM SIJOITA
150 END
200 FOR I=1 TO MAX
210 IF A(I)>=UL THEN PA=I:I=MAX
220 NEXT I
230 MAX=MAX+1:RETURN
300 FOR I=MAX TO PA+1 STEP -1
310 A(I)=A(I-1)
320 NEXT I
330 A(PA)=UL
340 RETURN
```

Kuva 6.19

Etsiminen suoritetaan aliohjelmassa 200–230. Kun uuden luvun (UL) paikka on löytynyt, silmukkamuuttujan (I) arvoksi sijoitetaan loppuarvo MAX niin, että etsintä keskeytyy. Muuttuja PA ilmoittaa uuden luvun (UL) paikan taulukossa. Sijoittaminen suoritetaan aliohjelmassa 300-340. Taulukon luvut siirretään ylöspäin ja rivillä 330 uusi luku (UL) sijoitetaan oikeaan paikkaan. Uuden luvun (UL) suuruudesta riippuen sijoittaminen kestää 3–5 sekuntia.

Edellistä esimerkkiä muutetaan niin, että haku ja sijoitus suoritetaan samalla kertaa.

```
100 DIM A(600):MAX=500
110 FOR I=1 TO MAX:A(I)=I:NEXT
120 INPUT"ANNA UUSI LUKU ";UL
130 GOSUB 200: REM HAE & SIJOITA
150 END
200 FOR I=MAX TO 1 STEP -1
210 IF UL>=A(I) THEN PA=I:I=1:GOTO 230
220 A(I+1)=A(I)
230 NEXT I
```

```
240 A(PA+1)=UL
250 MAX=MAX+1
260 RETURN
```

Kuva 6.20

Hakeminen aloitetaan ylhäältä. Kun uuden luvun paikka on löytynyt, I:n arvoksi muutetaan loppuarvo I riviltä 210. Jos uusi luku (UL) on vertailulukua pienempi, vertailtavaa lukua siirretään taulukossa ylöspäin (rivi 220). Rivillä 240 uusi luku (UL) sijoitetaan oikealle paikalle. Jälkimmäinen menetelmä on nopeampi, varsinkin kun taulukon koko kasvaa.

Hakuaikaa lyhentämällä sijoitusta voitaisiin nopeuttaa. Syventymättä sen enempää erilaisiin hakumenetelmiin tutustumme lyhyesti ns. puolitushakuun. Edellisissä esimerkeissä joudutaan suorittamaan keskimäärin 250 vertailua oikean paikan löytämiseksi. Binaarihakua eli puolitushakua käyttämällä selviämme yhdeksällä vertailulla.

Oletetaan, että taulukossa on luvut 1–500 ja sijoitettava luku on 378. Merkitään UL = 378 ja edetään vaiheittain.

1) Suurin 2:n potenssi, joka on pienempi kuin 500, on 256 = 2^8. Merkitään PA = 256. Ensimmäiseksi lukua UL verrataan taulukon alkioon A(PA).

2 Koska A(PA) = 256 < 378, PA:n arvoon lisätään kakkosen potenssi 2^7 = 128. PA:n uusi arvo on 256 + 128 = 384. Nyt UL:n arvoa verrataan taulukon alkioon A(PA) = 384.

3 Koska A(PA) = 384 > 378, PA:n arvosta vähennetään kakkosen potenssi 2^6 = 64. PA:n uusi arvo on 384 – 64 = 320. Nyt UL:n arvoa verrataan taulukon alkioon A(PA) = 320.

4) Koska A(PA) = 320 < 378, PA:n arvoon lisätään kakkosen potenssi 2^5 = 32. PA:n uusi arvo on 320 + 32 = 354. Nyt UL:n arvoa verrataan taulukon alkioon A(PA) = 354.

5) Koska A(PA) = 354 < 378, PA:n arvoon lisätään kakkosen potenssi 2^4 = 16. PA:n uusi arvo on 354 + 16 = 370. Nyt UL:n arvoa verrataan tulukon alkioon A(PA) = 370.

6) Koska A(PA) = 370. < 378, PA:n arvoon lisätään kakkosen potenssi $2^3 = 8$. PA:n uusi arvo on 370 + 8 = 378. Nyt UL:n arvoa verrataan taulukon alkioon A(PA) = 378.

7) Löytyi!!

Kerrataan lyhyesti. Lukua verrataan ensin siihen lukuun, jonka paikan ilmoittaa suurin kakkosen potenssi, joka on pienempi tai yhtä suuri kuin alkioiden kokonaismäärä. Jos ko. paikassa oleva luku on suurempi, vähennetään paikan ilmoittavasta luvusta 2:n seuraavaksi pienempi potenssi. Jos luku taas on pienempi, lisätään 2:n potenssi. Näin saadussa uudessa paikassa olevaa lukua verrataan sijoitettavaan lukuun. Sen mukaan onko luku suurempi vai pienempi kuin sijoitettava luku vähennetään tai lisätään paikan järjestyslukuun 2:n potenssi. Näin jatketaan, kunnes oikea paikka on löytynyt.

Vaikka taulukkomme sisältääkin kokonaisluvut 1–500, menetelmä soveltuu kaikkiin järjestettyihin taulukoihin. Vertailujen kokonaismäärä on korkeintaan

INT(LOG(MAX)/LOG(2)) + 1

jossa MAX = alkioiden lukumäärä ja LOG luonnollinen logaritmi.

Sijoitusohjelma kirjoitetaan nyt käyttämällä edellä esitettyä binaarihakua.

```
100 DIM A(600):MAX=500
110 FOR I=1 TO MAX:A(I)=I:NEXT
120 INPUT"ANNA UUSI LUKU ";UL
125 TI$="000000"
130 GOSUB 200:REM HAE PAIKKA
140 GOSUB 300:REM SIJOITA
145 PRINT TI$
150 END
200 PO=INT(LOG(MAX)/LOG(2))
210 PA=2^PO
220 FOR KP=PO-1 TO 0 STEP -1
230 PA=PA+2^KP*((UL<A(PA))-(UL>A(PA)))
240 IF PA>MAX THEN PA=MAX
250 NEXT KP
260 IF A(PA)<UL THEN PA=PA+1
270 MAX=MAX+1:RETURN
```

```
300 FOR I=MAX TO PA+1 STEP -1
310 A(I)=A(I-1)
320 NEXT I
330 A(PA)=UL
340 RETURN
```

Kuva 6.21

PO ilmoittaa suurimman mahdollisen eksponentin, tässä tapauksessa PO = 8. PA ilmoittaa ensimmäisen luvun paikan taulukossa. Silmukkamuuttuja KP saa arvot 7–0 (kakkosen tutkittavat eksponentit). Rivillä 230 PA:n arvo kasvaa tai vähenee sen mukaan onko UL > A(PA) vai UL < A(PA). Jos UL > A(PA), loogisen vertailun tulos on –1, muuten 0. Näin PA = PA + 2^{KP*} (+1) tai PA = PA + 2^{KP*} (–1).

Edellä esitetyt menetelmät ovat käyttökelpoisia vain silloin, kun koko tiedosto mahtuu kerralla tietokoneen muistiin.

Jos tiedosto on suuri, uusi tietue sijoitetaan siten, että vanhasta tiedostosta luetaan tietue kerrallaan ja avainkenttää verrataan sijoitettavaan tietueeseen. Jos uuden tietueen paikka ei ole vertailukohdassa, vanha tietue kirjoitetaan uuteen, kirjoittamista varten avattuun tiedostoon. Näin jatketaan, kunnes sijoitettava tietue voidaan kirjoittaa uuteen tiedostoon. Tämän jälkeen vanhan tiedoston loput tietueet kirjoitetaan uuteen tiedostoon. Vanha tiedosto voidaan nyt tuhota ja uudelle tiedostolle annetaan vanhan tiedoston nimi.

Oletetaan, että levykkeellä on tiedosto VANHA, johon pitää sijoittaa yksi uusi tietue. Tietueessa on kolme kenttää. Ohjelma voi olla

```
100 REM SIJOITUS
110 OPEN 15,8,15:R$=CHR$(13)
120 OPEN 2,8,2,"0:VANHA,S,R"
130 GOSUB 500
140 OPEN 3,8,3,"0:UUSI,S,W"
150 GOSUB 500
160 PRINT" UUSI TIETUE ?"
170 FOR I=1 TO 3
180 INPUT U$(I)
190 NEXT I:FL=0
200 INPUT#2,A$,B$,C$
210 S=ST:IF A$>U$(1) AND FL=0 THEN FOR I
=1 TO 3:PRINT#3,U$(I):NEXT:FL=1
220 PRINT#3,A$;R$;B$;R$;C$
```

```
230 IF S<>64 THEN 200
240 IF FL=0 THEN FOR I=1 TO 3:PRINT#3,U$
(I):NEXT
250 CLOSE 2:CLOSE 3
260 PRINT#15,"S0:VANHA"
270 PRINT#15,"R0:VANHA=UUSI"
280 CLOSE 15:END
500 INPUT#15,VN,VI$,VU,VS
510 IF VN<>0 THEN PRINT VN;VI$;VU;VS:CLO
SE 2:CLOSE 3:CLOSE 15:STOP
520 RETURN
```

Kuva 6.22

Tässä esimerkissä tietueen avainkenttänä toimii 1. kenttä. Tietueet on järjestetty 1. kentän mukaan esimerkiksi aakkosjärjestykseen. Vanhaa tiedostoa luetaan rivillä 200. Kun paikka löytyy eikä uutta tietuetta ole vielä kirjoitettu, se kirjoitetaan rivillä 210. Muuttujan FL arvo asetetaan ykköseksi, kun uusi tietue kirjoitetaan. Rivillä 230 testataan vanhan tiedoston loppu. Jos uutta tietuetta ei ole kirjoitettu ennen vanhan tiedoston loppua, se kirjoitetaan rivillä 240. Tiedosto VANHA tuhotaan rivillä 260 ja tiedoston UUSI nimi muutetaan rivillä 270.

Tätä menetelmää käytettäessä tiedoston koko ei saa olla yli 85 K, koska molempien tiedostojen pitää mahtua samalle levykkeelle. Jos käytettävissä on kaksi levyasemaa, kumpikin tiedosto voi olla omalla levykkeellään, jolloin tiedoston koko saa olla noin 170 K.

Merkkijonoja käsiteltäessä pitää aina muistaa muistinsiivous (garbage collection). On olemassa staattisia (paikallaan olevia) ja dynaamisia (ohjelman kuluessa luotavia) muuttujia. Vain dynaamiset muuttujat aiheuttavat kasaantumista. Siksi merkkijonoja ei kannata siirrellä esim. lajiteltaessa, vaan käytetään ns. indeksitaulukkoa ja siirretään kuhunkin jonoon kuuluvaa indeksiä. Merkkijonotaulukko pysyy entisessä järjestyksessä ja indeksitaulukosta nähdään merkkijonon paikka lajittelun jälkeen. On muistettava, että muistinsiivouksen aiheuttama viive on likimain verrannollinen muuttujien lukumäärän neliöön. Jos siis muuttujien lukumäärä kymmenkertaistuu, viive kasvaa satakertaiseksi.

Siivoustilanteelta vältytään, jos tietokone pakotetaan säännöllisin väliajoin puhdistamaan merkkijonoille varattu muistialue. Tämä saadaan aikaan sijoituksella X = FRE(0). Merkkijonojen käyttöä pitäisi myös välttää silloin, kun se ei ole

ehdottomasti tarpeen. Arvoja, jotka voidaan ilmoittaa numeerisen muuttujan avulla, ei pidä sijoittaa merkkijonomuuttujaan.

TIEDOSTON KORVAAMINEN

Olemassa oleva tiedosto voidaan korvata uudella päivitetyllä versiolla kahdella eri tavalla.

1) Ensin kirjoitetaan uusi tiedosto eri nimellä, sitten tuhotaan vanha tiedosto ja uusi tiedosto nimetään uudestaan.

2) Tiedosto avataan kirjoittamista varten käyttämällä avauskäskyssä ns. korvausmerkkiä @.

Ensimmäinen menetelmä on aina varma. Tiedosto luetaan kokonaan tai tietue kerrallaan tietokoneen muistiin, tietueet päivitetään ja korjatut tietueet kirjoitetaan uuteen tiedostoon. Sitten vanha tiedosto tuhotaan ja uuden tiedoston nimi muutetaan. Ohjelman pääosat ovat:

```
100 LUETAAN VANHA TIEDOSTO
 .
 .
300 MUUTETAAN TIETUEET
 .
 .
500 KIRJOITETAAN UUSI TIEDOSTO
 .
 .
700 PRINT#15,"S0:VANHA"
710 PRINT#15,"R0:VANHA=UUSI"
```

Kuva 6.23

Toisessa menetelmässä vanha tiedosto korvataan suoraan uudella, samannimisellä tiedostolla. Tiedosto avataan kirjoittamista varten seuraavasti:

```
OPEN 2,8,2,"@0:NIMI,S,W"
```

Kuva 6.24

Jos tiedosto ei ole levykkeellä ennestään, se luodaan.

Tätä tapaa käytettäessä pitää muistaa, että vanhaa tiedostoa ei poisteta levykkeeltä ennen kuin uusi on kirjoitettu. Varsinaisesta päällekirjoittamisesta ei siis ole kysymys. Siksi levykkeellä pitää olla tarpeeksi tilaa molempia versioita varten. Jos tila loppuu kesken, toimitus keskeytyy, uusi tiedosto on epätäydellinen ja vanha on kadonnut. Vanha tiedosto on vielä levykkeellä mutta ei näy sisällysluettelossa, koska ura-sektorilinkit on muutettu uutta tiedostoa vastaaviksi.

Korvattavan tiedoston pitää olla kunnolla suljettu (ei *-merkkiä sisällysluettelossa). Sulkemattomat tiedostot on poistettava VALIDATE-komennolla.

Tiedostoa avattaessa käyttäjälle voidaan antaa mahdollisuus olemassa olevan tiedoston korvaamiseen esim. seuraavasti:

```
10 INPUT "ANNA TIEDOSTON NIMI ";T$
20 T$=T$+",S,W":V$="E"
30 INPUT "KORVATAANKO VANHA (K/E) ";V$
40 IF V$="K" THEN T$="@0:"+T$
50 OPEN 3,8,3,T$
```

Kuva 6.25

MODIFY-tila

Jos tiedostoa ei suljeta kirjoittamisen jälkeen, sisällysluettelossa on tyypin tunnuksen edessä *-merkki. Tiedoston käyttämien lohkojen lukumäärä on 0. Tämä ei tarkoita, että tiedostosta ei ole kirjoitettu mitään. Tiedostoa ei kirjoiteta levykkeelle tavujen jatkuvana jonona vaan puskureittain. Puskuri on määrätty alue levyaseman RAM-muistissa ja sen pituus on 256 tavua. Kun tiedosto suljetaan, levykkeelle kirjoitetaan puskurissa jäljellä olevat tavut. Näin ainakin osa tiedoista – puskureissa olevat – hukkuu, jos tiedostoa ei suljeta kirjoittamisen jälkeen. Koko tiedosto katoaa, kun siitä ei ehditä kirjoittaa levykkeelle mitään (pienet tiedostot).

Sulkematonta tiedostoa ei voi lukea READ-tilassa, vaan on käytettävä erikoista MODIFY- eli M-tilaa. Tällöin saadaan luettua ne tiedoston tiedot, jotka on ehditty kirjoittaa levykkeelle.

Kokeilemme M-tilaa muodostamalla peräkkäistiedoston, jota ei suljeta.

```
10 REM M-TILA.1
20 DIM A$(200):OPEN 15,8,15,"I"
30 OPEN 3,8,3,"0:M-TILA,S,W"
40 FOR I=1 TO 200
50 A$(I)="DATA"+STR$(I)
60 PRINT#3,A$(I)
70 NEXT
80 END
```

Kuva 6.26

Ohjelmassa kirjoitetaan M-TILA-tiedostoon 200 tietuetta: DATA 1, DATA 2 jne. Kun tietueet on kirjoitettu riveillä 40–70, ohjelma loppuu ilman tiedoston sulkemista (myöskään komentokanavaa 15 ei suljeta). Levyaseman merkkivalo jää palamaan merkiksi siitä, että tiedosto on vielä auki. Kun levyke otetaan asemasta pois, valo sammuu.

Sisällysluettelosta nähdään, että levyllä olevien vapaiden lohkojen määrä on pienentynyt, vaikka sulkemattoman tiedoston käyttöön on merkitty 0 lohkoa. Määrällisesti tämä pienentyminen riippuu urasta, jolle kirjoittaminen tapahtuu. Kun tiedoston kirjoittaminen aloitetaan, merkitään aloitusuran loput lohkot varatuiksi BAM:iin. Jos tiedosto alkaa uuden uran alusta, uran kaikki lohkot merkitään varatuiksi. Oikeata BAM-versiota säilytetään levyaseman RAM-muistissa, josta se siirretään levykkeelle, kun tiedosto suljetaan. Kun tiedostoa ei suljeta, BAM on virheellinen: lohkoja on varattu liikaa. Tämä pitää paikkansa vain peräkkäistiedostoihin nähden. Sulkemattomien ohjelmatiedostojen lohkoja ei merkitä varatuiksi. Jos ohjelmatiedosto jostain syystä jää sulkematta, levykkeelle ei saa tallentaa mitään, mikäli osa tiedostosta halutaan pelastaa. Lohkot pitää ensin varata ohjelmallisesti. Sulkematonta suhteellista tiedostoa ei voi olla olemassa.

Yritetäänpä nyt lukea tiedosto M-TILA käyttämällä M-tilaa.

```
10 REM M-TILA.2
20 DIMA$(200):N=0:OPEN 15,8,15,"I"
30 OPEN 3,8,3,"0:M-TILA,S,M"
40 N=N+1
50 INPUT#3,A$(N)
60 PRINT A$(N)
70 IF ST<>0 THEN PRINT ST:GOTO 90
80 GET R$:IF R$="" THEN 40
90 CLOSE 3:CLOSE 15:END
```

Kuva 6.27

Tiedostoa luetaan tietue kerrallaan käyttämällä INPUT#-käskyä. Tietueen tulostamisen jälkeen tutkitaan tilamuuttujan (ST) avulla onko kaikki kunnossa. Tulostusta jatketaan kunnes ST<>0. Tulostus voidaan haluttaessa keskeyttää (rivi 80).

Tiedoston alkupää tulostuu oikein. Viimeinen omassa laitteessani tulostunut tietue oli DATA 181, jonka jälkeen ST sai arvon 66 ja ohjelma keskeytyi. Viimeinen oikea tietue on siis DATA 181. Jos tiedoston loppumerkki olisi löytynyt, ST:n arvo olisi ollut 64.

Jos tiedoston rakennetta ei tunneta, edellisessä ohjelmassa kannattaa käyttää GET#-käskyä.

Yksinkertaisin tiedoston pelastamisohjelma on seuraavanlainen. Pelastetun tiedoston nimi on KUTILA.

```
10 REM M-TILA.3
20 DIMA$(200):N=0:OPEN 15,8,15,"I"
30 OPEN 3,8,3,"0:M-TILA,S,M"
40 OPEN 2,8,2,"0:KUTILA,S,W"
50 N=N+1
60 INPUT#3,A$(N):S=ST
70 PRINT#2,A$(N)
80 IF S=0 THEN 50
90 CLOSE 3:CLOSE 2:CLOSE 15:END
```

Kuva 6.28

Luettaessa sulkematonta tiedostoa M-tilassa voidaan törmätä virheilmoitukseen ILLEGAL TRACK OR SECTOR.

Ohjelmatiedostoa ei voida pelastaa edellisellä ohjelmalla. Ennen pelastamista tiedoston käyttämien urien vapaat lohkot pitää merkitä BAM:iin varatuiksi. Sitten pelastaminen sujuu edellisen kaltaisella ohjelmalla. Tiedostoa pitää vain lukea GET#-käskyllä. Voit yrittää ohjelmatiedostojen pelastamista kirjan levykkeellä olevilla ohjelmilla VARAA ja SULKEMATON.

OHJELMATIEDOSTO PERÄKKÄISTIEDOSTOKSI

Ohjelmatiedosto voidaan tallentaa levykkeelle peräkkäistiedostoksi tai käyttäjän tiedostoksi (USER) tulostamalla ohjelmalistaus näytön asemesta suoraan levykkeelle (samalla tavalla kuin kirjoittimelle).

Puhdas BASIC-ohjelma saadaan peräkkäistiedostoksi seuraavilla komennoilla:

```
OPEN 2,8,2,"0:NIMI,S,W"
CMD 2
LIST
```

Kuva 6.29

Kun levyasema lopettaa pyörimisen, kirjoita

```
PRINT#2
CLOSE 2
```

Kuva 6.30

Pienillä ohjelmilla kirjoittaminen alkaa vasta CLOSE-käskyn kirjoittamisen jälkeen.

Tiedoston tunnuksena on nyt sisällysluettelossa SEQ, mutta ohjelma ei ole alkuperäisessä muodossaan vaan ASCII-tyyppisenä tekstitiedostona.

POKE-ohjelmointia käyttäen saamme edellisen avulla yksinkertaisen tekstinkäsittelyohjelman. Kirjoita tekstisi käyttämällä rivinumeroita. Halutessasi käyttää pilkkuja aloitat jokaisen rivin lainausmerkillä.

```
10 "1541 levyasema on joskus kiva laite,
20 "joskus sitten taas ei.
```

Kuva 6.31

Kun teksti on valmis, kirjoitat

```
POKE 22,35
```

Kuva 6.32

Nyt voit tallentaa tekstin peräkkäistiedostoksi edellä esitetyllä tavalla tai voit tulostaa sen kirjoittimelle.

Ohjelmatiedosto saadaan levykkeelle käyttäjäntiedostoksi, kun OPEN-käskyssä tiedoston tyyppi S korvataan tyypillä U.

Mitä hyötyä on BASIC-ohjelmasta, joka on tallennettu levykkeelle peräkkäistiedostoksi? Sitähän ei voida ladata eikä suorittaa tavalliseen tapaan. Jos tällainen peräkkäistiedosto saadaan muutetuksi ohjelmatiedostoksi ja liitetyksi BASIC-ohjelmaan, usein käytetyistä aliohjelmista voidaan muodostaa kirjasto. Yksi mahdollisuus on toimia siten, että peräkkäistiedostosta luetaan ohjelmarivi kerrallaan ja käytetään C64:n itse-editointiominaisuuksia. Itse-editointi tarkoittaa sitä, että ohjelma

suoritettaessa luo itse uusia ohjelmarivejä tai käskyjä.

Kun olet kirjoittanut pääohjelman, johon haluat lisätä sopivan aliohjelman (levykkeellä peräkkäistiedostona), lisäät pääohjelman alkuun seuraavat rivit:

```
0 OPEN 2,8,2,"NIMI"
1 POKE 152,1:PRINT"[illegible]"
2 GET#2,A$:PRINT A$;:IF A$<>CHR$(13) THE
N 2
3 PRINT"GOTO 1[illegible]";:POKE 631,13:POKE 632,1
3:POKE 198,2:END
```

Kuva 6.33

Kirjoita riville 0 haluamasi aliohjelman nimi ja suorita ohjelma. Sulje avattu tiedosto suorituksen päätyttyä käskyllä CLOSE 2 ja poista rivit 0–3. Mahdollisista virheilmoituksista ei tarvitse välittää. Joitakin ehtoja pitää kuitenkin asettaa. Liitettävän aliohjelman on oltava levykkeellä peräkkäistiedostona, yhdelläkään rivillä ei saa olla yli 80 merkkiä, eikä aliohjelmassa saa olla samoja rivinumeroita kuin pääohjelmassa. Jos samoja rivinumeroita löytyy, aliohjelman rivi korvaa pääohjelman rivin.

Riviltä 0 voi jättää pois tyypin S ja tilan R, koska ne ovat oletusarvot. Jos aliohjelma on tallennettu USER-tiedostoksi, rivillä 0 voi olla pelkkä aliohjelman nimi tai nimi, tyyppi U ja tila R. Tyyppinä ei voi olla S.

Aliohjelmaa luetaan merkki kerrallaan, kunnes rivin päättymismerkki CHR$(13) tulee vastaan. Kukin aliohjelman rivi tulostetaan näyttöön toiselta riviltä alkaen. Rivin jälkeen kirjoitetaan käsky GOTO 1, ja kohdistin viedään näytön vasempaan ylänurkkaan. Näppäimistöpuskuriin (631–640) sijoitetut kaksi RETURN-merkkiä (CHR$(13)) aiheuttavat aliohjelmasta luetun rivin ja GOTO 1 -rivin kuittaamisen. Näppäilyjen lukumäärä sijoitetaan muistipaikkaan 198. Tämä editointi aikaansaa kanavan sulkeutumisen C64:n puolella (levyaseman puolella kanava on edelleen auki) ja siksi rivillä 1 muistipaikkaan 152 sijoitetaan arvo 1. Tämä muistipaikka kertoo kulloinkin avoinna olevien tiedostojen lukumäärän.

Edellä olevalla menetelmällä voidaan myös pelastaa ohjelma, joka on vahingossa poistettu C64:n muistista NEW-käskyllä tai SYS 64738 -komennolla. NEW-käsky ei poista ohjelmaa muistista vaan siirtää osoittimet siten, että muisti on ikään kuin tyhjä. Ohjelma voidaan pelastaa seuraavasti:

1) Kirjoita NEW-käskyn (tai SYS 64738) jälkeen seuraavat käskyt

```
POKE 2050,1:SYS 42291
```

Kuva 6.34

2) Ohjelma voidaan nyt listata, mutta ei suorittaa, koska osoittimet ovat sekaisin.

3) Listaus ohjataan levykkeelle ja ohjelma tallennetaan peräkkäistiedostoksi.

4) Tiedosto muutetaan jälleen ohjelmatiedostoksi edellä esitetyllä liittämisohjelmalla

Vahingossa poistetussa ohjelmassa ei saa olla rivejä 0 – 3. Muut ehdot ovat samat kuin edellä. Jos pelastetussa ohjelmassa on rivinumerot 0 – 3, nämä rivit on kirjoitettava uudestaan pelastamisen jälkeen.

Ohjelmatiedosto saadaan peräkkäistiedostoksi myös SAVE-käskyllä, kun kirjoitetaan seuraavasti:

```
SAVE"NIMI,S,W",LN
```

Kuva 6.35

Laitenumero on LN.

Vaikka sisällysluettelossa tiedoston tyyppinä on SEQ, se on rakenteeltaan täysin samanlainen kuin ohjelmatiedosto ja voidaan ladata LOAD-käskyllä.

```
LOAD"NIMI,S,R",LN
```

Kuva 6.36

ja suorittaa RUN-käskyllä.

Samalla tavalla ohjelmatiedosto saadaan käyttäjän tiedostoksi. SAVE- ja LOAD-käskyissä on tiedoston tyypiksi vain merkittävä U tai USER.

7. LUKU

SUHTEELLISET TIEDOSTOT

Suhteelliset tiedostot ovat rakenteeltaan hieman monimutkaisempia kuin peräkkäistiedostot ja niiden käyttökin on vaikeampaa. Ne ovat kuitenkin erittäin nopeita ja hyödyllisiä.

Suhteellisen tiedoston jokaista tietuetta voidaan käsitellä (lukea/kirjoittaa) muihin tietueisiin koskematta. Suhteelliset tiedostot sopivat sen vuoksi erikoisen hyvin sellaisiin tehtäviin, joissa tietoja joudutaan usein päivittämään. Koska koko tiedoston ei tarvitse olla kerralla tietokoneen muistissa, suhteellisilla tiedostoilla voidaan hoitaa suuriakin tietomääriä. Kuhunkin tietueeseen päästään nopeasti käsiksi, kun vain tiedetään tietueen numero.

Suhteellisten tiedostojen haittapuolena on se, että jokainen tietue on samanpituinen, vaikkei kaikkea tilaa tarvittaisikaan. Jos tietueen pituus on suuri ja tietueeseen kuuluvan tiedon pituus pieni, levyketilaa kuluu suotta. Jos tietueen pituus määrätään tiedostoa luotaessa liian pieneksi, törmätään varmasti tietueeseen, joka ei mahdu annettuihin rajoihin. Huolellisella ennakkosuunnittelulla tällaiset ongelmat voidaan varmasti välttää.

Suhteellisilla tiedostoilla hoidettavissa sovelluksissa tarvitaan usein apuna myös peräkkäistiedostoja. Niihin tallennetaan tietuenumerot sovelluksen vaatimassa järjestyksessä. Tällaisia tiedostoja kutsutaan indeksitiedostoiksi. On kuitenkin muistettava, että samalla kertaa voi olla avoinna vain yksi suhteellinen tiedosto ja yksi peräkkäistiedosto.

SUHTEELLISEN TIEDOSTON RAKENNE

Suhteellinen tiedosto muodostuu tietueista, jotka jaetaan kenttiin. Tietueen pituus ilmoitetaan tiedostoa luotaessa eikä sitä sen jälkeen voi muuttaa. Peräkkäistiedostojen lukeminen ja kirjoittaminen voi tapahtua kenttä kerrallaan, mutta suhteellisten tiedostojen kirjoittaminen on syytä suorittaa tietue kerrallaan. Yhteen tietueeseen voi kuulua useita kenttiä. Ainoana rajoituksena on tietueen maksimipituus, joka on 254 merkkiä. Levyketilan turha käyttö aiheutuu siitä, että tietueen kenttien sisältämä tieto vaihtelee. Määrätyn tietuepituuden perusteella DOS löytää jokaisen tietueen nopeasti. Ohjelman käyttäjän pitää vain ilmoittaa tietueen pituus tiedostoa luodessaan ja DOS pitää huolta lopusta. Yhdeksännessä luvussa käsitellään yksityiskohtaisesti suhteellisen tiedoston rakennetta levykkeellä.

Tietueen suunnittelussa on valittavana kaksi vaihtoehtoa:

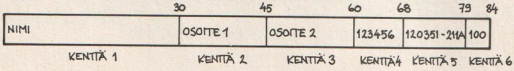
- kentän pituus on kussakin tietueessa sama;
- kentän pituus vaihtelee kentän sisällön mukaan.

Tarkastellaan esimerkkinä yksinkertaista jäsenrekisteriä, jonka yhdessä tietueessa on yhden jäsenen kaikki tiedot. Tietue voi muodostua kuudesta kentästä.

KENTTÄ	SISÄLTÖ	PITUUS	MERKIT
1	Nimi	30	1 – 30
2	Osoite 1	15	31 – 45
3	Osoite 2	15	46 – 60
4	Puhelin	8	61 – 68
5	Sos.tunnus	11	69 – 79
6	Jäs.maksu	5	80 – 84

Jokaisen kentän pituuteen on laskettava myös kenttien välinen erotin CHR$ (13).

Tietueen pituus on siis 84 merkkiä. Jos kentän pituus pidetään vakiona, tietue on seuraavanlainen:



Kuva 7.1

Jos kentän sisältö on pienempi kuin kentän pituus, kentän loppu pitää täyttää välilyönneillä. Kentän sijainti pysyy tietueen sisällä aina samana.

Parempi tapa on pitää kentän pituus vaihtelevana. Tällöin kenttää ei täytetä välilyönneillä ja kenttien välissä on vain erotin. Näin menetellen tietueen pituus tulee yleensä pienemmäksi. Edellisessä esimerkissä pituudeksi voidaan asettaa 70. Kentän sijainti tietueen sisällä vaihtelee kentän pituudesta riippuen. Tietueen loppuun jäävä tyhjä tila täytetään nollilla. Tietue näyttää tällaiselta:

NIMI	OSOITE1	OSOITE2	123456	120351-211A	100	(LOPPU NOLLIA)

Kuva 7.2

Kentän pituus kannattaa pitää vakiona, jos tallennettavat tiedot ovat vakiomittaiset, jos yksittäisen kentän sisältö pitää nopeasti selvittää, tai jos nopeus on tärkeämpi tekijä kuin levyketilan turha käyttö. Jos kentän sisältö vaihtelee, kentän pituus kannattaa pitää vaihtelevana.

Kentän pituuden määräämiseksi ei voida esittää mitään sääntöjä. Jos kentän pituus on vakio, pituuden määrää pisin kenttään tuleva tieto. Tämä voidaan selvittää huolellisella ennakkosuunnittelulla. Tietueen pituus on kenttien pituuksien summa.

Jos kentän pituus vaihtelee, kannattaa arvioida lyhimmät ja pisimmät kentän pituudet. Oikea kentän pituus on likimain näiden arvojen keskiarvo.

Jotta tietueen lukeminen kävisi nopeasti INPUT#-käskyllä, kentän pituus ei saa ylittää 88 merkkiä.

SUHTEELLISEN TIEDOSTON AVAUS

Suhteellisen tiedoston käytössä voidaan erottaa seuraavat osavaiheet:

1) Komentokanavan avaaminen

2) Suhteellisen tiedoston avaaminen

3) Tietueosoittimen asettaminen P-komennolla oikean tietueen alkuun

4) Tietueen kirjoittaminen (PRINT#)

5) Tietueen lukeminen (GET# tai INPUT#)

6) Suhteellisen tiedoston ja komentokanavan sulkeminen

Komentokanavan avaaminen on tuttu asia. Kerrataan se kuitenkin lyhyesti.

Syntaksi:

```
OPEN TN,LN,KN
```

Esimerkiksi:

```
OPEN 15,8,15
```

Kuva 7.3

TN = looginen tiedostonumero (1–127)
LN = laitenumero (8–11), tavallisesti 8
KN = kanavanumero (toisio-osoite), aina 15

Komentokanavan yhteydessä tiedostonumerona on yleensä 15. Näin on helppo muistaa, mikä tiedostonumero liittyy komentokanavaan, koska komentokanava on aina 15.
Komentokanavaa avattaessa voidaan samalla lähettää komento:

```
OPEN 15,8,15,"I0"
```

Kuva 7.4

Suhteellinen tiedosto avataan OPEN-käskyllä seuraavasti:

Syntaksi:

```
OPEN TN,LN,KN,"NIMI,L,"+CHR$(PI)
```

Kuva 7.5

TN = looginen tiedostonumero (1–127)
LN = laitenumero (8–11), tavallisesti 8
KN = kanavanumero (toisio-osoite) (2–14)
NIMI = tiedoston nimi
L = tiedostotyypin tunnus
PI = tietueen pituus (2–254, ei kuitenkaan 42, 58 tai 63)

Tätä avauskäskyn muotoa on käytettävä, kun tiedosto avataan ensimmäisen kerran, eli kun tiedosto luodaan. Muotoa voidaan käyttää myös olemassa olevan tiedoston avaamiseen. Jos tiedosto on jo olemassa, se voidaan avata lukemista/kirjoittamista varten hieman lyhyemmin.

```
OPEN TN,LN,KN,"NIMI"
```

Kuva 7.6

Olemassa olevan tiedoston nimessä saa käyttää jokereita (* ja ?). Tiedoston tunnus L ja tietueen pituus PI voidaan jättää pois, jos tiedosto on jo olemassa.

Tietueen pituutena ei saa käyttää arvoja 42, 58 ja 63, joilla on erikoismerkitys DOS:ille. Ne vastaavat merkkien *, : ja ? ASCII-koodeja. Jos joku niistä on PI:n arvona, saadaan virheilmoitus SYNTAX ERROR. Tietueen pituutta ei voida muuttaa sen jälkeen, kun tiedosto on luotu. Jos olemassa olevaa suhteellista tiedostoa avattaessa ilmoitetaan uusi tietueen pituus, DOS hylkää avauspyynnön. Ainoa mahdollisuus on avata kokonaan uusi tiedosto uudella tietueen pituudella ja kopioida vanha tiedosto tietue kerrallaan tähän uuteen tiedostoon.

Suhteellinen tiedosto avataan aina sekä lukemista että kirjoittamista varten. Siksi avauskäskyssä ei ole tilaparametria R tai W.

Kiinnitä erikoisesti huomiota avauskäskyssä olevan L-tunnuksen jäljessä olevaan pilkkuun. Sitä ei saa jättää pois.

Kun avaat suhteellisen tiedoston, voit avata muita tiedostoja seuraavasti:

1 peräkkäistiedosto
1 peräkkäistiedosto ja 1 suorasaantitiedosto (-kanava)
2 suorasaantitiedostoa (-kanavaa)

Esimerkkejä:

```
1) OPEN 2,8,2,"POSTI,L,"+CHR$(60)

2) OPEN 3,8,6,"POSTI"

3) 10 T$="REKISTERI":PI=65
   20 OPEN 3,8,4,T$+",L,"+CHR$(PI)
```

Kuva. 7.7

Ensimmäisessä esimerkissä on ensi kerran avattu POSTI-niminen suhteellinen tiedosto, jossa tietueen pituus on 60. Toisessa esimerkissä sama tiedosto on avattu myöhemmin lukemista tai kirjoittamista varten. Kolmannessa esimerkissä on rivillä 10 sijoitettu tiedoston nimi REKISTERI muuttujan T$ arvoksi ja tietueen pituus 65 muuttujan PI arvoksi. Rivin 20 avauskäskyssä on käytetty näitä muuttujia todellisten arvojen asemesta.

P-KOMENNON KÄYTTÖ

Ennen kuin tietue voidaan lukea tai kirjoittaa, tietueosoitin on siirrettävä oikean tietueen kohdalle. Osoitin siirretään P-komennon avulla.

Syntaksi:

```
PRINT#TN,"P"+CHR$(KN)+CHR$(TA)+CHR$(TY)+
CHR$(TP)
```

Kuva 7.8

TN = looginen tiedostonumero, jota on käytetty komentokanavaa avattaessa
KN = kanavanumero (toisio-osoite), jota on käytetty suhteellista tiedostoa avattaessa
TA = tietueen numeron alatavu
TY = tietueen numeron ylätavu
TP = tavun paikka (1 – tietueen pituus)

TP ilmoittaa sen merkin paikan tietueessa, josta lukeminen/kirjoittaminen aloitetaan. Tämä täytyy AINA ilmoittaa! Jos TP jätetään pois, DOS saattaa olettaa, että CHR$(13) on tavun paikka, ja lukeminen/kirjoittaminen alkaa 13. merkistä.

Tietueen järjestysnumero on ilmoitettava muodossa alatavu/ylätavu (TA/TY). Jos tietueen numero on TNU, TA ja TY saadaan seuraavasti:

TY = INT (TNU/256)
TA = TNU – 256 * TY

Esimerkkejä:

1) Tietueen numero TNU = 198
TY = INT(198/256) = INT(0.7734) = 0

TA = 198 – 256 * TY = 198 – 256 * 0 = 198

2) Tietueen numero TNU = 445
TY = INT(445/256) = INT(1.7383) = 1
TA = 445 – 256 * TY = 445 – 256 * 1 = 189

Jos tietueen numero TNU <256, niin TY = 0 ja TA = TNU.
P-komento on suorasaantikomento, joten se on lähetettävä komentokanavan kautta. Siksi komentokanavan pitää olla auki ennen kuin sijoituskomentoa (P) käytetään. Komentokanava on syytä avata aina ensimmäisenä ja sulkea viimeisenä.

Esimerkkejä:

```
1) PRINT#15,"P"+CHR$(2)+CHR$(115)+CHR$(Ø
   )+CHR$(1)

2) PRINT#15,"P"+CHR$(4)+CHR$(25)+CHR$(2)
   +CHR$(1Ø)
```

Kuva 7.9

Ensimmäisessä esimerkissä tietueen järjestysnumero on 115 (115 + 256 * 0) ja osoitin on tietueen 1. merkissä. Toisessa esimerkissä tietueen järjestysnumero on 537 (25 + 256 * 2) ja osoitin on tietueen 10. merkissä.
Tietueita voi olla korkeintaan 65535. Lisäksi käytettävissä oleva levyketila rajoittaa tietueiden lukumäärää siten, että jos tietueen pituus on PI, tietueita voi olla korkeintaan

INT(167132/PI).

Eräät Commodore-käyttäjät ovat kertoneet, että P-komento toimii epäluotettavasti, kun tietueita luetaan/kirjoitetaan satunnaisessa järjestyksessä. Virhe voidaan välttää, kun tietueosoitin siirretään kahdesti samaan paikkaan ja ohjelmoidaan pieni viive ennen PRINT#-käskyn suorittamista. Näin DOS saa tarpeeksi aikaa tietueen etsimiseen.

TIETUEEN KIRJOITTAMINEN

Kun tietueosoitin on siirretty halutun tietueen kohdalle, tietue voidaan kirjoittaa PRINT#-käskyllä.

Syntaksi:

```
PRINT#TN,ML
```

Kuva 7.10

TN = looginen tiedostonumero, jota on käytetty suhteellista tiedostoa avattaessa
ML = muuttujalista (kirjoitettava tieto)

Muuttujalistassa voi olla yksi tai useampia muuttujia. Jos muuttujia on useita, niiden väliin pitää kirjoittaa erotin, esim. CHR$(13).

Koko tietue kannattaa kirjoittaa yhdellä PRINT#-käskyllä. Eri kenttiä vastaavat merkkijonot liitetään yhdeksi merkkijonoksi, joka kirjoitetaan tiedostoon. Silloin kirjoittaminen tapahtuu PRINT#-käskyllä tietueosoittimen sijaintikohdasta tietueen loppuun. Tietueen alkuosaa ei voida kirjoittaa uudestaan siten, että loppuosa pysyy ennallaan. Alkuosan jälkeen tietueen loppu kirjoitetaan täyteen CHR$(0) merkkejä ja loppuosassa olevat tiedot hukkuvat.

Merkkijonon pituus on hyvä tarkistaa ennen tiedostoon kirjoittamista. Jos merkkijonon pituus on suurempi kuin annettu tietueen pituus, kaikkia merkkejä ei kirjoiteta, vaan jono katkaistaan tietueen pituuden mittaiseksi. Levyasema antaa tällöin virheilmoituksen 51 OVERFLOW IN RECORD.

Jos tietueen pituus on avauskäskyssä ilmoitettua pienempi, käyttäjän ei tarvitse lisätä tietueen loppuun CHR$ (0)-merkkejä, siitä DOS pitää huolen.

Koko tiedosto kannattaa kirjoittaa jo tiedostoa luotaessa täyteen CHR$(0)-merkkejä. Tämä tehdään sijoittamalla tietue-osoitin P-komennolla viimeisen tietueen alkuun ja kirjoittamalla viimeisen tietueen sisällöksi esim. CHR$(255). Näin koko tiedosto saadaan kerralla täyteen tyhjiä tietueita. Seuraavassa esimerkissä sadan tietueen tiedosto kirjoitetaan täyteen tyhjiä tietueita.

```
100 REM SUHT.TIETUEEN LUOMINEN
110 TL=100
120 TY=INT(TL/256)
130 TA=TL-256*TY
140 OPEN 15,8,15
150 OPEN 2,8,2,"SUHTKOE1,L,"+CHR$(50)
160 INPUT#15,VN,VI$,VU,VS
```

```
170 IF VN=0 OR VN=50 THEN 190
180 PRINT VN;VI$;VU;VS:CLOSE2:CLOSE15:ST
OP
190 PRINT#15,"P"+CHR$(2)+CHR$(TA)+CHR$(T
Y)+CHR$(1)
200 PRINT#2,"VIIMEINEN TIETUE"
210 INPUT#15,VN,VI$,VU,VS
220 CLOSE 2:CLOSE 15:END
```

Kuva 7.11

Tietueen pituus on tässä tapauksessa 50 merkkiä. Muistathan, että mukaan on laskettava kenttien väliset erottimet ja tietueen lopussa oleva erotin. Ohjelman suoritus kestää noin 17 sekuntia. Viimeiseen tietueeseen kirjoitetaan ilmoitus "VIIMEINEN TIETUE". Koska 100. tietuetta kirjoitettaessa joudutaan tiedoston lopun yli, levyasema antaa virheilmoituksen 50 RECORD NOT PRESENT. Siitä ei tässä tapauksessa tarvitse välittää. Rivillä 210 on sama virheilmoitus merkkivalon sammuttamiseksi.

Seuraava esimerkki havainnollistaa kenttien liittämistä tietueeksi ja tietueen kirjoittamista yhdellä PRINT#-käskyllä.

```
100 REM TIETUEEN KIRJOITTAMINEN
110 TL=100:KE=4:R$=CHR$(13)
120 OPEN 15,8,15
130 OPEN 2,8,2,"KOETUS,L,"+CHR$(60)
140 INPUT#15,VN,VI$,VU,VS
150 IF VN<>0 AND VN<>50 THEN PRINT VN;VI
$;VU;VS:CLOSE 2:CLOSE 15:STOP
160 INPUT "TIETUEEN NO ";NU
170 IF NU<1 OR NU>TL THEN PRINT"!!":GOTO
 160
180 T$="":FOR I=1 TO KE
190 PRINT"KENTTA "I;:INPUT K$(I)
200 T$=T$+K$(I)+R$
210 NEXT I
220 IF LEN(T$)>60 THEN PRINT "LIIAN PITK
A TIETUE !":GOTO 180
230 TY=INT(NU/256):TA=NU-256*TY
240 PRINT#15,"P"+CHR$(2)+CHR$(TA)+CHR$(T
Y)+CHR$(1)
250 PRINT#2,T$
```

```
260 INPUT#15,VN,VI$,VU,VS
270 CLOSE 2:CLOSE 15:END
```

Kuva 7.12

Rivillä 110 on annettu tietueiden lukumäärä TL, kenttien lukumäärä KE ja erotin R$. Kun komentokanava ja suhteellinen tiedosto on avattu, tietueen numero NU pyydetään rivillä 160. Rivillä 170 tarkistetaan, että NU on sallituissa rajoissa. Riveillä 180–210 annetaan kenttien sisällöt. Kenttien sisällöt ja erottimet kootaan rivillä 200 muuttujan T$ arvoksi. Rivillä 220 tarkistetaan kirjoitetun tietueen pituus. Tietuenumeron ala- ja ylätavut lasketaan rivillä 230, ja rivillä 240 tietueosoitin siirretään oikean tietueen alkuun. Tietueen sisältö kirjoitetaan rivillä 250.

Kun olemassa olevaan tiedostoon lisätään tietueita, joudutaan usein tiedoston lopun yli. Virheilmoituksesta RECORD NOT PRESENT ei tarvitse välittää. Se voidaan tulkita ilmoitukseksi siitä, että tiedostoa laajennetaan. Tämä ilmoitus esiintyy vain silloin, kun levykkeeltä joudutaan DOS:in toimesta varaamaan uusi lohko tiedostoa varten. Jos esimerkiksi luettava tietue on jo kirjoitettu ja DOS antaa edellä olevan virheilmoituksen, olet ehkä unohtanut sulkea tiedoston kirjoittamisen jälkeen ja osa tietueista on kadonnut.

Levykkeellä olevaa suhteellista tiedostoa ei voida korvata @-toiminnalla kuten ohjelma- ja peräkkäistiedostoja. Jos suhteellinen tiedosto halutaan korvata uudella, samannimisellä versiolla, vanha tiedosto on tuhottava ennen uuden luomista. Jos avauskäskyyn lisätään @-merkki, DOS ei reagoi vaan avaa tiedoston tavalliseen tapaan.

SUHTEELLISEN TIEDOSTON LUKEMINEN

Suhteellisen tiedoston lukeminen suoritetaan INPUT#- tai GET#-käskyllä. Ennen lukemista tietueosoitin on siirrettävä P-komennolla oikean tietueen kohdalle. Yleensä käytetään INPUT#-käskyä, koska sen avulla voidaan kerralla lukea enemmän. GET#-käsky lukee vain yhden merkin.

Syntaksi:

```
INPUT#TN,ML

GET#TN,ML
```

Kuva 7.13

TN = looginen tiedostonumero, jota on käytetty suhteellista tiedostoa avattaessa
ML = muuttujalista

Jos samalla käskyllä luetaan arvoja useille muuttujille, muuttujat on erotettava toisistaan pilkuilla. GET#-käskyä käytetään yleensä viallisen tiedoston tai sellaisten merkkien lukemiseen, joita ei voida kunnolla lukea INPUT#-käskyllä. Tällaisia ovat esimerkiksi pilkku ja nollajono.

Jos kentät on erotettu toisistaan CHR$(13)-merkillä, ne voidaan lukea yhdellä INPUT#-käskyllä, tai jokainen kenttä voidaan lukea omalla INPUT#-käskyllään (ks. esimerkki). Jos kenttien erottimena on pilkku, kaikki kentät luetaan yhdellä INPUT#-käskyllä.

DOS ei anna virheilmoitusta, jos luet tietueen lopun ohi. Tietueen loppu pitää selvittää ST-tilamuuttujan arvon avulla. Tietueen lopussa ST = 64.

Jos tietueessa on CHR$(0)-merkki, merkkijonomuuttujan arvoksi tulee tyhjä jono (ei merkkejä). Tämä pitää aina tarkistaa seuraavasti:

```
GET#2,A$:IF A$="" THEN A$=CHR$(0)

tai

GET#2,A$:A$=A$+CHR$(0)
```

Kuva 7.14

Seuraavassa esimerkissä luetaan edellisen esimerkin PRINT#-käskyllä kirjoitettu tietue.

```
100 REM TIETUEEN LUKEMINEN
110 TL=100:KE=4
120 OPEN 15,8,15
130 OPEN 2,8,2,"KOETUS,L,"+CHR$(60)
140 INPUT#15,VN,VI$,VU,VS
```

```
150 IF VN<>0 AND VN<>50 THEN PRINT VN;VI
$;VU;VS:CLOSE 2:CLOSE 15:STOP
160 INPUT "TIETUEEN NO ";NU
170 IF NU<1 OR NU>TL THEN PRINT"TT":GOTO
 160
180 TY=INT(NU/256):TA=NU-256*TY
190 PRINT#15,"P"+CHR$(2)+CHR$(TA)+CHR$(T
Y)+CHR$(1)
200 FOR I=1 TO KE
210 INPUT#2,K$(I)
220 PRINT "KENTTA"I;K$(I)
230 NEXT I
240 CLOSE 2:CLOSE 15:END
```

Kuva 7.15

Tietue luetaan riveillä 200–230 INPUT#-käskyllä siten, että lukeminen suoritetaan kenttä kerrallaan. Lukeminen voidaan suorittaa myös yhdellä INPUT#-käskyllä:

```
200 INPUT#2,A$,B$,C$,D$
```

Kuva 7.16

Rivit 210–230 on nyt poistettava. Ohjelmaan pitää vielä lisätä kenttien tulostaminen näyttöön.

Nyt suoritetaan sama tehtävä GET#-käskyllä. Ensimmäisen INPUT#-esimerkin rivit kirjoitetaan rivistä 200 eteenpäin seuraavasti:

```
200 FOR I=1 TO KE
210 GET#2,A$:IF A$="" THEN A$=CHR$(0)
220 IF A$=CHR$(13) THEN 240
230 K$(I)=K$(I)+A$:GOTO 210
240 PRINT"KENTTA"I;K$(I)
250 NEXT I
260 CLOSE 2:CLOSE 15:END
```

Kuva 7.17

Jos kenttien lukumäärä ei ole tiedossa, lukeminen voidaan suorittaa tekemällä ohjelmaan seuraavat muutokset ja lisäykset:

```
200 I=I+1
215 IF ST=64 THEN 260
250 GOTO 200
```

Kuva 7.18

SUHTEELLISEN TIEDOSTON SULKEMINEN

Kun tietueet on kirjoitettu tai luettu, tiedosto suljetaan. Sulkeminen suoritetaan CLOSE-käskyllä.

Syntaksi:

```
CLOSE TN
```

Kuva 7.19

TN = looginen tiedostonumero, jota on käytetty suhteellista tiedostoa avattaessa

Sulkeminen on tärkeää varsinkin kirjoittamisen jälkeen, sillä viimeksi kirjoitetut tietueet saattavat kadota. Sulkematonta suhteellista tiedostoa ei voida muodostaa, koska työskentely tapahtuu aina luku/kirjoitustilassa. Jos haluat lukea juuri kirjoittamasi tietueet, on varminta sulkea tiedosto ja avata se uudelleen lukemista varten.

INDEKSOIDUT SUHTEELLISET TIEDOSTOT

Suhteelliset tiedostot ja peräkkäistiedostot yhdistetään tässä tiedostotyypissä siten, että varsinaiset tiedot (tietueet) tallennetaan suhteelliseen tiedostoon ja peräkkäistiedostossa pidetään ns. indeksit, jotka ilmoittavat tietueiden numerot jonkin kentän mukaisessa järjestyksessä. Jos henkilötietueen ensimmäisessä kentässä on esimerkiksi henkilön sukunimi, tämä kenttä voi olla avainkenttänä, jonka mukaan tietuenumerot järjestetään indeksitiedostoon.

TIETUE N:O	1. KENTTÄ	INDEKSI
1	Lahtinen	2
2	Aaltonen	3
3	Jokinen	5
4	Peltonen	1
5	Koskinen	4

Indeksitiedosto ilmoittaa oikean aakkosjärjestyksen (2–3–5–1–4). Koska Aaltonen on aakkosjärjestyksessä ensimmäisenä, on Aaltosta vastaavan tietueen numero indeksitiedoston ensimmäinen alkio.

Indeksitiedostoja voi olla useita. Edellisessä tiedostossa voi toisena avainkenttänä olla henkilön syntymäaika.

Indeksitiedostoa luotaessa menetellään seuraavasti:

- Valitaan tietueen kenttä, jonka mukaan lajittelu suoritetaan (avainkenttä).
- Käydään suhteellinen tiedosto läpi ja poimitaan valittu kenttä ja tietueen numero väliaikaiseen merkkijonotaulukkoon.
- Lajitellaan väliaikainen taulukko.
- Muodostetaan indeksitiedosto siten, että lajitellusta väliaikaisesta taulukosta poimitaan tietuenumerot ja kirjoitetaan ne indeksitiedostoon.

Indeksitiedoston luomisohjelma voi olla seuraavanlainen. Ohjelma on kirjan levykkeellä nimellä LUO INDEKSI.1.

```
100 REM INDEKSITIEDOSTON LUOMINEN
110 DIM K$(TL):TY$="   "
120 OPEN 15,8,15,"I0"
130 OPEN 2,8,2,"PERUS,L,"+CHR$(60)
140 FOR I=1 TO TL
150 TY=INT(I/256):TA=I-256*TY
160 PRINT#15,"P"+CHR$(2)+CHR$(TA)+CHR$(T
Y)+CHR$(1)
170 INPUT#2,A1$,A2$,A3$,A4$
180 K$(I)=A2$+RIGHT$(TY$+STR$(I),3)
190 NEXT I:CLOSE 2
195 :
200 REM TAULUKON K$() LAJITTELU
210 Z=2^(INT(LOG(TL-1)/LOG(2))+1)
220 Z=Z/2:Y=TL-Z-1:W=1:IF Z<1 THEN 300
230 Q=W
240 G=Q+Z:IF K$(Q)>K$(G) THEN 270
250 W=W+1:IF W>Y THEN 220
260 GOTO 230
270 W$=K$(Q):K$(Q)=K$(G):K$(G)=W$
280 Q=Q-Z:IF Q<0 THEN 250
290 GOTO 240
295 :
```

```
300 OPEN 3,8,3,"PERUS.IND,S,W"
310 FOR I=1 TO TL
320 K=VAL(RIGHT$(K$(I),3))
330 PRINT#3,K
340 NEXT I
350 CLOSE 3:CLOSE 15:END
```

Kuva 7.20

Muuttujan TL paikalle pitää sijoittaa tietueiden lukumäärä. Avainkentäksi on valittu tietueen toinen kenttä A2$.

Ohjelmassa on käytetty Shell-Metznerin lajittelualgoritmia. Se on edullinen, kun tietueita on paljon.

Indeksitiedosto voidaan myös luoda ensin tietuenumeroiden pohjalta ja sitten järjestetään indeksitaulukko. Näin on menetelty kirjan levykkeellä olevassa ohjelmassa LUO INDEKSI.2, jossa lajittelualgoritmina on yksinkertainen kuplalajittelu.

Uuden tietueen lisääminen suhteelliseen tiedostoon ei ole vaikeaa. Vaikeampi tehtävä on indeksitiedostojen saattaminen ajan tasalle. Yksinkertaisin (ja hitain) tapa on verrata uuden tietueen avainkenttää entisiin avainkenttiin. Kun oikea paikka löytyy, uusi tietuenumero sijoitetaan indeksitiedostoon. Kuudennessa luvussa esitetty binaarihaku on nopeampi menetelmä.

Seuraava binaarihaulla (puolitushaulla) toteutettu aliohjelma hakee oikean paikan ja sijoittaa uuden tietuenumeron. Aliohjelmaan tultaessa uusi tietue on suhteellisen tiedoston lopussa. Muuttujia käytetään seuraavasti:

TL	=	tietueiden lukumäärä ennen uuden tietueen sijoittamista
U$	=	uuden tietueen avainkenttä
V$	=	vanhan tietueen avainkenttä (johon verrataan)
IN( )	=	indeksitaulukko
H	=	paikka indeksitaulukossa
TN	=	verrattavan tietueen tietuenumero
UN	=	uuden tietueen numero
TY	=	tietuenumeron ylätavu
TA	=	tietuenumeron alatavu

```
1100 REM ALIOHJELMA HAKU & SIJOITUS
1110 N=INT(LOG(TL)/LOG(2))
1120 H=2^N
1125 :
1130 FOR P=N-1 TO 0 STEP -1
1140 TN=IN(H):GOSUB 2000
1150 H=H+2^P*((U$<V$)-(U$>V$))
1160 IF H>TL THEN H=TL
1170 NEXT P:TL=TL+1
1175 :
1180 TN=IN(H):GOSUB 2000
1190 IF V$<U$ THEN H=H+1
1195 :
1200 FOR I=TL TO H+1 STEP -1
1210 IN(I)=IN(I-1)
1220 NEXT I
1230 IN(H)=UN
1240 RETURN
1245 :
2000 TY=INT(TN/256):TA=TN-256*TY
2010 PRINT#15,"P"+CHR$(2)+CHR$(TA)+CHR$(
TY)+CHR$(1)
2020 INPUT#2,V$,B$,C$
2030 RETURN
```

Kuva 7.21

Oikean paikan hakeminen suoritetaan riveillä 1130–1190. Tietuenumeroiden siirtäminen ja uuden tietuenumeron sijoittaminen suoritetaan riveillä 1200–1230. Tietue (3 kenttää) luetaan riviltä 2000 alkavassa aliohjelmassa.

Indeksitiedostojen käyttö suhteellisten tiedostojen yhteydessä vaatii huomattavasti enemmän etukäteissuunnittelua kuin pelkkien suhteellisten tiedostojen käyttö. Suunnittelussa on hyvä ottaa huomioon ainakin seuraavat seikat:

- Suhteellisen tiedoston koko: mahtuvatko kaikki tietueet samalle levykkeelle vai pitääkö indeksitiedostot sijoittaa eri levykkeelle?
- Indeksitiedostojen lukumäärä
- Erikoistapausten huomioiminen: tiedostossa ei ole tietueita, virheilmoituksen lukeminen ja ilmoittaminen.
- Tietueen poistaminen tiedostosta ja tietueen korjaaminen sekä näiden toimenpiteiden aiheuttamat muutokset indeksitiedostoihin.

8. LUKU

SUORASAANTIKÄSKYT

Tähän mennessä olemme työskennelleet tiedostojen kanssa, jotka muodostuvat läheisesti yhteenkuuluvista tiedoista. Käyttäessämme peräkkäis- ja suhteellisia tiedostoja olemme avanneet tiedoston nimeen liittyvän kanavan 1541-levyasemaan ja siten päässeet tietoihin käsiksi. Tietojen varsinaisesta käsittelystä on huolehtinut levyaseman käyttöjärjestelmä DOS.

MITEN 1541 TYÖSKENTELEE?

Commodore 64 ja 1541-levyasema ovat välittäneet tietoja seuraavanlaisten tehtävien suorittamisessa:

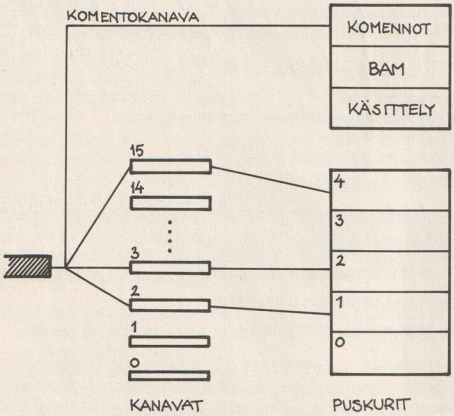
- ohjelmien tallentaminen
- ohjelmien lataaminen
- tiedoston avaaminen lukemista tai kirjoittamista varten
- tietojen lukeminen ja kirjoittaminen
- tiedoston sulkeminen
- virheilmoituksen lukeminen
- komentojen välittäminen

Näitä tehtäviä varten 1541-levyasemalla on sisäisiä kanavia (väyliä), joiden välityksellä tiedot kirjoitetaan levykkeelle tai luetaan levykkeeltä. Avatessamme tiedoston käskyllä OPEN TN,LN,KN ilmoitamme kolme numeroa, parametria. Näistä ensimmäinen, looginen tiedostonumero (TN), on sen kanavan numero, jota tietokone käyttää yhteydenpidossaan oheislait-

teen kanssa. Kirjoittaessamme tiedostoon PRINT#-käskyllä käyttämme sen yhteydessä vain loogista tiedostonumeroa. Tietokone välittää muut parametrit tälle kanavalle (tiedostonumerolle), joten niitä ei tarvitse toistaa PRINT#-käskyssä. Kanavanumero (toisio-osoite) KN saa aikaan kanavan avautumisen sarjaväylän toisessa päässä, ja levyasema avaa numerolla KN kanavan tietojen välittämiseksi tietokoneen kanssa.

Koska useita kanavia (tiedostoja) voi olla auki, levyaseman pitää kyetä järjestämään jokaiselle levykkeelle pyrkijälle oikea kanava.

Jotta suoraan levykkeelle tapahtuvien luku/kirjoitustapahtumien lukumäärä pysyisi pienenä, levyasemassa on kahdeksan tietopuskuria. Jokainen puskuri voi sisältää 256 tavua. DOS huolehtii puskureista ja osoittaa avatulle kanavalle puskurin, jota se voi käyttää. Kahdeksasta puskurista vain viisi on kanavien käytössä. 1541 tarvitsee tiedostojen käsittelyä varten kaksi puskuria (esim. BAM:ia varten yhden) ja lisäksi yhden puskurin tietoväylää pitkin tulevien komentojen tunnistamiseen.



Kuva 8.1

Seuraavaksi tarkastellaan, mitä tapahtuu kun ohjelma ladataan levykkeeltä tietokoneen muistiin.

Ladattaessa käytetyn kanavanumeron 0 perusteella 1541 tietää, että kysymyksessä on ohjelman lataaminen ja asettaa kanavan käyttöön yhden tietopuskurin. Komentopuskurista saadaan ohjelman nimi ja lataus käynnistyy. Ohjelma etsitään sisällysluettelosta ja luetaan ensimmäisen ohjelman käyttämän lohkon osoite. Ohjelma siirretään lohko lohkolta tietopuskuriin ja sieltä edelleen komentokanavaa pitkin tietokoneeseen.

Ohjelmaa tallennettaessa menetellään päinvastoin. Tietopuskuri täytetään ensin Commodore 64:n lähettämillä tiedoilla ja sitten puskurin sisältö kirjoitetaan levykkeelle.

SUORASAANNIN VALMISTELUT

Kuudennessa ja seitsemännessä luvussa esitetyt tiedostorakenteet eivät aina riitä ratkaisemaan tiedostojenkäsittelyn erikoisongelmia. 1541 mahdollistaa esitettyjen tiedostorakenteiden kiinteän muodon ohittamisen. Voimme siis ohjelmallisesti toimia samoin kuin DOS, kun se kirjoittaa yksittäisiä tietoja levykkeen sektoreille.

Suorasaantikäskyt välittyvät DOS:ille komentokanavan kautta. Näiden käskyjen ymmärtämiseksi tulee muistaa, että 1541:ssä on toiminnallisesti neljä osaa, joiden yhteistyö mahdollistaa levyketyöskentelyn:

- levyke
- 1541:n puskurit
- 1541:n DOS
- 1541:n kanavat

Pystyäksemme ohjaamaan 1541:n tavallisesti automaattisesti tapahtuvia toimintoja, meidän pitää osata seuraavat osatehtävät:

1. Komentokanavan avaaminen
2. Luku/kirjoituskanavan avaaminen
3. Vapaan puskurin osoittaminen avatulle kanavalle
4. Levykkeen määrätyn lohkon lukeminen puskuriin
5. Työskentely puskurissa
6. Puskurin sisällön kirjoittaminen levykkeen määrätylle lohkolle
7. Levykkeen BAM:in muokkaaminen

Näiden osatoimintojen avulla voimme käsitellä levykkeen tietoja ilman DOS:in tukea. DOS itsekin työskentelee näiden toimintojen avulla. Jokaista tehtävää varten DOS:illa on oma komentonsa, eräille useitakin.

Suorasaantikäskyt tai -komennot voidaan jakaa seuraaviin ryhmiin:

Levykekomennot

BLOCK-READ	lukee tietolohkon 1541:n RAM-muistiin (B-R tai U1).
BLOCK-WRITE	kirjoittaa tietolohkon 1541:n RAM-muistista levykkeelle (B-W tai U2).
BLOCK-EXECUTE	lukee tietolohkon levykkeeltä puskuriin ja suorittaa lohkon sisällön konekieliohjelmana (B-E).

BAM-komennot

BLOCK-ALLOCATE	merkitsee ilmoitetun sektorin BAM:iin käytetyksi (B-A).
BLOCK-FREE	merkitsee ilmoitetun sektorin BAM:iin vapaaksi (B-F).

Puskurikomennot

BUFFER-POINTER	siirtää osoittimen määrättyyn tavuun (B-P) puskurissa.

Muistikomennot

MEMORY-READ	lukee tavun 1541:n RAM- tai ROM-muistista (M-R).
MEMORY-WRITE	kirjoittaa tavun 1541:n RAM- tai ROM-muistiin (M-W).
MEMORY-EXECUTE	suorittaa 1541:n RAM- tai ROM-muistiin tallennetun konekielisen ohjelman (6502) (M-E).

Käyttäjän komennot

USER0	palauttaa USER JUMP -vektorin alkuarvon (U0).
USER1	sama kuin BLOCK-READ (U1)
USER2	sama kuin BLOCK-WRITE (U2)
USER3-USER9	aiheuttavat hypyn määrättyyn osoitteeseen 1541:n RAM-muistissa (U3–U9).

Edellä esitetyt käskyt täydentävät useimmiten toisiaan. Levykkeen lohko voidaan lukea U1-käskyllä, lohkon sisältöä voidaan tutkia B–P- tai M–R-käskyillä ja lohkon sisältö voidaan kirjoittaa takaisin levykkeelle U2-käskyllä. Tavallisimmat ja useimmin käytetyt käskyt ovat U1, B–P ja U2. Suorasaantikäskyjä käytettäessä kirjoitetaan aina käskyn lyhennetty muoto, joka yllä olevassa taulukossa on merkitty sulkeisiin selityksen jälkeen. Pitempää muotoa saa käyttää kaikista muista käskyistä paitsi M–R, M–W ja M–E.

Aiemmin esitetyistä seitsemästä osatehtävästä neljä viimeistä voidaan suorittaa näiden käskyjen avulla. Ensimmäinen tehtävä, komentokanavan avaaminen, on entuudestaan tuttu. Tehtävät 2 ja 3 voidaan suorittaa samalla kertaa, kun avataan ns. suorasaantikanava tai -tiedosto.

Syntaksi:

```
OPEN TN,LN,KN,"#PN"
```

Kuva 8.2

TN = looginen tiedostonumero (1–127)

LN = laitenumero (8–11), tavallisesti 8
KN = kanavanumero (2–14)
PN = puskurin numero (0–4)

Jos puskurin numero jätetään pois, DOS huolehtii puskurin valinnasta. Puskurin numero (PN) kannattaa tavallisesti jättää pois, koska emme tiedä, onko ilmoittamamme puskuri vapaa. Jos ilmoitettu puskuri on jo käytössä, saadaan virheilmoitus 70 NO CHANNEL 0 0.

Avaamalla suorasaantikanavan muodostamme yhteyden C64:n ja tietopuskurin (luku/kirjoituspuskurin) välille. Tieto kirjoitetaan puskuriin käskyllä PRINT#TN ja luetaan joko käskyllä INPUT#TN tai GET#TN. Käskyjen yhteydessä EI siis käytetä kanavanumeroa KN. Kanavanumeroon viitataan vain suorasaantikäskyn yhteydessä, esimerkiksi U1. Loogisen tiedostonumeron ei tarvitse olla sama kuin kanavanumero. Samaa numeroa käytetään vain muistamisen helpottamiseksi.

Ennen kuin siirrymme tarkastelemaan varsinaisia käskyjä, kertaamme eri urilla olevien sektoreiden lukumäärät:

URAT	SEKTORIT	SEKTOREITA YHT.
1 - 17	0 - 20	21
18 - 24	0 - 18	19
25 - 30	0 - 17	18
31 - 50	0 - 16	17

BLOCK-READ TAI USER1

Tämä käsky siirtää määrätyn uran ja sektorin sisällön 1541:n RAM-muistin jollekin alueelle ns. puskuriin. Käsky lyhennetään joko B-R tai U1. Koska B-R ei aina toimi oikein, suositellaan muotoa U1.

Syntaksi:

```
PRINT#TN,"U1";KN;AS;U;S

tai

PRINT#TN,"U1:"KN;AS;U;S

tai

PRINT#TN,"U1:KN,AS,U,S"
```

Kuva 8.3

TN = komentokanavaa avattaessa käytetty looginen tiedostonumero (1-127)
KN = suorasaantikanavaa avattaessa käytetty kanavanumero (toisio-osoite) (2-14)
AS = asemanumero, aina 0
U = ura (1-35)
S = sektori (0-uran yläraja)

B–R-käskyn syntaksi on samanlainen.

Kun annetun uran määrätty sektori on siirretty puskuriin, puskuriosoitin osoittaa tavuun 255. Tavut 0–255 voidaan nyt lukea 0-tavusta alkaen. Jos lukeminen halutaan aloittaa jostakin muusta tavusta, puskuriosoitin on ensin siirrettävä ko. tavuun B–P-käskyllä. Lukeminen suoritetaan GET#-käskyllä tavu kerrallaan. Koska INPUT#-käsky ei lue nollajonoja " " eikä ohjausmerkkejä oikein, sitä ei käytetä. Kun tavu on luettu, tarkistetaan onko se nollajono. Nollajono pitää muuntaa merkiksi CHR$(0), koska ASC-funktio ei toimi nollajonoilla.

U1-käskyn vaihtoehtoisista muodoista valitaan kulloinkin sopivin. Jos uran ja sektorin numero ilmoitetaan muuttujan avulla, käytetään muotoa 1 tai 2.

Katsotaan esimerkkejä. Levykkeen sisällysluettelon ensimmäinen lohko sijaitsee uralla 18 sektorilla 1. Se luetaan puskuriin ja sisältö tulostetaan näyttöön.

```
100 REM U1
110 OPEN 15,8,15
120 PRINT#15,"I0"
130 OPEN 1,8,2,"#"
140 PRINT#15,"U1";2;0;18;1
150 FOR I=0 TO 255
160 GET#1,A$
170 IF A$="" THEN A$=CHR$(0)
180 A=ASC(A$)
190 IF A<31 OR A>96 THEN A$="."
200 PRINT I,A,A$
210 NEXT
220 CLOSE 2
230 INPUT#15,VN,VI$,VU,VS
240 CLOSE 15
250 END
```

Kuva 8.4

110	komentokanavan avaaminen
120	levykkeen initialisointi
130	suorasaantikanavan avaaminen
140	uran 18 sektorin 1 sisällön lukeminen kanavalle 2 varattuun puskuriin
150	256 tavun lukeminen
160	tavun siirtäminen kanavan 2 puskurialueelta C64:n muistiin

170	nollajonon muuntaminen
180	tavun ASCII-arvo
190	epätavallinen ASCII-arvo: merkiksi "."
200	tavun numeron, ASCII-arvon ja merkin tulostaminen
210	silmukan päättyminen
220	suorasaantikanavan sulkeminen
230	virheilmoituksen lukeminen
240	komentokanavan sulkeminen
250	loppu

GET#-käskyn yhteydessä rivillä 160 oleva numero 2 viittaa suorasaantikanavan loogiseen itedostonumeroon eikä kanavanumeroon. Jos siis avaaminen suoritetaan esimerkiksi käskyllä OPEN 1,8,2,"#", rivillä 160 on GET#1,A$. Jos merkin ASCII-arvoa ei tarkasteta rivillä 190, merkkien väri muuttuu kesken tulostuksen ja kohdistin siirtyy viimeisten tavujen kohdalla yhtäkkiä vasempaan ylänurkkaan.

Jos virheilmoitusta ei lueta rivillä 230, punainen merkkivalo jää palamaan. Tämä johtuu siitä, että DOS yrittää kirjoittaa BAM:in takaisin levykkeelle suorasaantikanavan sulkemisen jälkeen.

Rivit 170 ja 180 voidaan yhdistää merkitsemällä A = ACS(A$+CHR$(0)).

Lisäesimerkkejä U1-käskystä on muiden käskyjen yhteydessä ja luvun lopussa.

Muuttamalla edellisen ohjelman riviä 140 voimme tutkia B–R-käskvn luonnetta ja omituisuutta.

```
140 PRINT#15,"B-R";2;0;18;1
```

Kuva 8.5

Kun ohjelma suoritetaan uudestaan, tulostus on erilainen. Ensimmäisen tavun (0-tavu) arvon pitäisi olla 18 mutta se on 4, joka on toisen tavun (1. tavu) arvo. Ensimmäistä tavua ei päästä lukemaan, ellei osoitinta siirretä 0-tavuun. Lisää ohjelmaan seuraava rivi:

```
145 PRINT#15,"B-P";2;0
```

Kuva 8.6

Nyt myös ensimmäinen tavu (tavu 0) saadaan luetuksi, mutta tulostus ei ole vieläkään samanlainen kuin alkuperäisellä ohjelmalla. Tavut 1–18 ovat samat, mutta niitä seuraavat tavut eivät ole oikein, vaan tavujen 1–18 arvot toistuvat.

Poikkeavuus johtuu siitä, että B-R-käskyä käytettäessä DOS merkitsee ensimmäisen tavun (tavu 0) osoittimeksi. Viimeinen luettava tavu on se, minkä ensimmäisen tavun sisältö ilmoittaa, eli tässä tapauksessa 18. Lohkon ensimmäinen tavu ilmoittaa uran, jolla sisällysluettelo jatkuu, ja se on 18 (sektori 4). 19. tavu voidaan lukea osoitinta siirtämällä.

Ongelmat jatkuvat luettaessa lohkon sisältöä B-R-käskyllä sen jälkeen, kun se on kirjoitettu uudestaan B-W-käskyllä. Tästä lisää tuonnempana.

BUFFER-POINTER

Tämän käskyn avulla päästään puskurin yksittäiseen tavuun. Mikä tahansa puskurin tavuista 0–255 voidaan lukea tai kirjoittaa. Käsky lyhennetään B-P.

Syntaksi:

```
PRINT#TN,"B-P";KN;TA

tai

PRINT#TN,"B-P:"KN;TA

tai

PRINT#TN,"B-P:KN,TA"
```

Kuva 8.7

TN = komentokanavaa avattaessa käytetty looginen tiedostonumero (1–127)
KN = suorasaantikanavaa avattaessa käytetty kanavanumero (toisio-osoite) (2–14)
TA = tavu (0–255)

Seuraava esimerkki lukee ja tulostaa uran 18 sektorin 0 viidennen tavun.

```
100 REM B-P.1
110 OPEN 15,8,15
120 PRINT#15,"I0"
130 OPEN 2,8,2,"#"
140 PRINT#15,"U1";2;0;18;0
150 PRINT#15,"B-P";2;5
160 GET#2,A$:A=ASC(A$+CHR$(0))
170 PRINT"VIIDES TAVU ON " A
180 CLOSE 2: CLOSE 15
```

Kuva 8.8

Toisessa esimerkissä luetaan levykkeen nimi ja ID-numero. Levykkeen nimi sijaitsee uralla 18 sektorilla 0 alkaen tavusta 144. Nimessä on korkeintaan 16 merkkiä. Jos nimi on lyhyempi kuin 16 merkkiä, sen perään lisätään tarvittava määrä merkkejä SHIFT + välilyönti (CHR$(160)). Nimen jäljessä seuraa kaksi CHR$(160)-merkkiä ja kaksimerkkinen ID-numero. Kun silmukkamuuttujan ylärajaksi asetetaan 23, tulostuu myös levykkeen formaatin tunnus 2A.

```
100 REM B-P.2
110 OPEN 15,8,15
120 PRINT#15,"I0"
130 OPEN 2,8,2,"#"
140 PRINT#15,"U1";2;0;18;0
150 PRINT#15,"B-P";2;144
160 FOR I=1 TO 20
170 GET#2,A$
180 PRINT A$;:NEXT
190 CLOSE 2: CLOSE 15
```

Kuva 8.9

BLOCK-WRITE TAI USER2

Tällä käskyllä puskurin sisältö kirjoitetaan ilmoitetulle uralle/sektorille. Käsky lyhennetään B-W tai U2. Koska B-W ei toimi kunnolla, suositellaan muotoa U2.

Syntaksi:

```
PRINT#TN,"U2";KN;AS;U;S
```

tai

```
PRINT#TN,"U2:"KN;AS;U;S
```

tai

```
PRINT#TN,"U2:KN,AS,U,S"
```

Kuva 8.10

TN = komentokanavaa avattaessa käytetty looginen tiedostonumero (1–127)
KN = suorasaantikanavaa avattaessa käytetty kanavanumero (toisio-osoite) (2–14)
AS = asemanumero, aina 0
U = ura (1–35)
S = sektori (0-uran yläraja)

B–W-käskyn syntaksi on samanlainen.

Kun U2-käsky suoritetaan, puskurin koko sisältö kirjoitetaan annetulle uralle/sektorille. Puskuriosoittimen arvoa ei oteta huomioon käskyä suoritettaessa.

Ensimmäisessä esimerkissä kirjoitetaan teksti "KOKEILUA" uralle 1 sektorille 0 alkaen ensimmäisestä tavusta.

```
100 REM U2.1
120 OPEN 15,8,15
130 OPEN 2,8,2,"#"
140 PRINT#15,"U1";2;0;1;0
150 A$="KOKEILUA"
160 PRINT#2,A$
170 PRINT#15,"U2";2;0;1;0
180 CLOSE 2
190 PRINT#15,"I0":CLOSE15
```

Kuva 8.11

Rivillä 140 ko. ura/sektori luetaan puskuriin. Näin sektorin sisältö muuttuu vain alussa olevan tekstin osalta. Rivillä 160 teksti kirjoitetaan puskuriin ja rivillä 170 puskurin sisältö siirretään uralle 1 sektorille 0. Ohjelmaa on paras ensin kokeilla jollakin vähemmän tärkeällä levykkeellä.

Levykkeen nimi voidaan muuttaa seuraavan ohjelman avulla.

```
100 REM MUUTA NIMI
110 FOR I=1TO16:SP$=SP$+CHR$(160):NEXT
120 OPEN 15,8,15,"I0"
130 GOSUB500
140 OPEN 1,8,2,"#"
150 PRINT#15,"U1";2;0;18;0
160 GOSUB500
170 PRINT#15,"B-P";2;144
180 FOR I=1 TO 16
190 GET#1,A$:IF A$="" THEN A$=CHR$(0)
200 IF A$=CHR$(160) THEN 220
210 VN$=VN$+A$:GOTO 190
220 PRINT"⌧ VANHA NIMI ON ⌧"VN$"⌧"
230 INPUT" ANNA UUSI NIMI ";UN$
240 IFLEN(UN$)<>0ANDLEN(UN$)<17THEN 260
250 GOTO 230
260 INPUT"⌧ OLETKO VARMA (K/E) K";V$
270 IF V$<>"K" THEN 230
280 UN$=LEFT$(UN$+SP$,16)
290 PRINT#15,"B-P";2;144
300 PRINT#1,UN$;
310 PRINT#15,"U2";2;0;18;0
320 GOSUB 500:CLOSE 2
330 GOSUB 500:PRINT#15,"I0"
340 GOSUB 500:CLOSE 15
350 PRINT"⌧ VALMIS !"
360 END
500 INPUT#15,VN,VI$,VU,VS
510 IF VN=0 THEN RETURN
520 CLOSE2:INPUT#15,VN,VI$,VU,VS
530 CLOSE 15
```

Kuva 8.12

110	merkkijonomuuttujan SP$ arvoksi sijoitetaan 16 merkkiä CHR$ (160)
120	levyaseman initialisointi
140	suorasaantikanavan avaaminen
150	18. uran 0. sektorin tiedot luetaan puskuriin
170	puskuriosoitin siirretään 144. tavuun
180–220	levykkeen vanhan nimen lukeminen
230–270	käyttäjältä pyydetään uusi nimi
280	nimen perään lisätään tarvittava määrä merkkejä CHR$ (160)

290	puskuriosoitin siirretään 144. tavuun
300	uusi nimi kirjoitetaan puskuriin
310	puskurin sisältö kirjoitetaan levykkeelle
500–530	aliohjelma virheilmoituksen lukemiseksi

Kun lohko kirjoitetaan levykkeelle B-W-käskyllä, tavut 1–255 tulevat oikein, mutta puskuriosoittimen viimeisin paikka sijoitetaan 0-tavuun. Kun lohko sitten luetaan B-R-käskyllä, 0-tavun ASCII-arvo ilmoittaa kuinka monta tavua lohkosta luetaan (katso B-R). Lohkon ensimmäinen tavu (0-tavu) ilmoittaa uran, jolla tiedosto jatkuu levykkeellä. Näin tämä linkkiosoitin saattaa tuhoutua.

MEMORY-READ

Tällä käskyllä voidaan lukea mikä tahansa 1541:n ROM- tai RAM-muistin alue. Käskyn yhteydessä on ilmoitettava muistipaikka, josta lukeminen alkaa. Kerralla voidaan lukea korkeintaan 255 tavua. Käsky lyhennetään M-R.

Syntaksi:

```
PRINT#TN,"M-R"CHR$(AL)CHR$(YL)CHR$(LKM)
```

tai

```
PRINT#TN,"M-R:"CHR$(AL)CHR$(YL)CHR$(LKM)
```

Kuva 8.13

TN	=	komentokanavaa avattaessa käytetty looginen tiedostonumero (1–127)
AL	=	muistipaikan alatavu (0–255)
YL	=	muistipaikan ylätavu (0–255)
LKM	=	tavujen lukumäärä (1–255)

Viimeinen parametri CHR$(LKM) voidaan jättää pois. Tällöin oletusarvona on LKM = 1.

Jos luettavan muistialueen alkupaikka 10-järjestelmässä on AP, niin

YL = INT(AP/256) ja AL = AP – 256 * YL

Liitteessä 1 olevasta 1541:n muistikartasta nähdään, että levykkeellä olevien vapaiden lohkojen (sektoreiden) lukumäärä

löytyy muistipaikoista $2FA ja $2FC eli 762 ja 764. Koska 762 = 250 + 256 * 2 ja 764 = 252 + 256 * 2, saadaan vapaiden lohkojen lukumäärä selville seuraavalla ohjelmalla.

```
100 OPEN 15,8,15,"I0"
110 PRINT#15,"M-R"CHR$(250)CHR$(2)
120 GET#15,A$:A=ASC(A$+CHR$(0))
130 PRINT#15,"M-R"CHR$(252)CHR$(2)
140 GET#15,B$:B=ASC(B$+CHR$(0))
150 PRINT A+256*B "LOHKOA VAPAANA"
160 CLOSE 1
```

Kuva 8.14

Tässä ohjelmassa on M-R-käskystä jätetty parametri CHR$(LKM) pois, koska molemmilla kerroilla luetaan yksi tavu.

Jos levyltä luetaan ennen M-R-käskyä jokin lohko U1-käskyllä, on suorasaantikanavaa avattaessa ilmoitettava käytettävän puskurin numero. 1541:n puskurialueet ovat seuraavassa taulukossa.

PUSKURI	MUISTIALUE	HUOM!
	$0000 – $00FF	ei käytössä, nollasivu
	$0100 – $01FF	ei käytössä, pino
	$0200 – $02FF	ei käytössä, komento-puskuri
0	$0300 – $03FF	OPEN 5,8,5,"#0"
1	$0400 – $04FF	OPEN 5,8,5,"#1"
2	$0500 – $05FF	OPEN 5,8,5,"#2"
3	$0600 – $06FF	OPEN 5,8,5,"#3"
4	$0700 – $07FF	ei käytössä, BAM

Tavut luetaan GET#-käskyllä komentokanavaa eikä suorasaantikanavaa pitkin. GET#-käskyn yhteydessä käytetään komentokanavaa avattaessa ilmoitettua loogista tiedostonumeroa. Levykkeen nimi luetaan tällä menetelmällä seuraavasti:

```
100 OPEN 15,8,15,"I0"
110 OPEN 2,8,2,"#1"
120 PRINT#15,"U1";2;0;18;0
130 PRINT#15,"M-R"CHR$(144)CHR$(4)CHR$(16)
140 FOR I=1 TO 16
```

```
150 GET#15,A$
160 IF A$=CHR$(160) THEN 200
170 A$=A$+CHR$(0)
180 PRINT A$;
190 NEXT I
200 CLOSE 2:CLOSE15
```

Kuva 8.15

Koska BAM sijoitetaan puskuriin 4 ($0700 - $077) ja levykkeen nimi on BAM:in 144:stä tavusta eteenpäin, levykkeen nimi saadaan myös lukemalla (nimen pituus) 16 tavua muistipaikasta $0790 = 1936 = 144 + 7 * 256 eteenpäin. Seuraavassa esimerkissä nimi luetaan INPUT#-käskyllä.

```
100 OPEN 15,8,15,"I0"
130 PRINT#15,"M-R"CHR$(144)CHR$(7)CHR$(16)
150 INPUT#15,A$
180 PRINT A$
200 CLOSE15
```

Kuva 8.16

Kirjan levykkeellä olevan ohjelman TUTKI DRIVE avulla voit tutkia 1541:n RAM- tai ROM-muistia mielesi mukaan.

MEMORY-WRITE

Tällä käskyllä kirjoitetaan tietoja DOS:in RAM-muistiin: nollasivulle, pinoon ja puskurialueille. Tiedot lähetetään komentokanavan kautta. Käsky lyhennetään M–W.

Syntaksi:

```
PRINT#TN,"M-W"CHR$(AL)CHR$(YL)CHR$(LKM)C
HR$(TA)
```

tai

```
PRINT#TN,"M-W:"CHR$(AL)CHR$(YL)CHR$(LKM)
CHR$(TA)
```

Kuva 8.17

TN = komentokanavaa avattaessa käytetty looginen tiedostonumero (1–127)
AL = muistipaikan alatavu (0–255)
YL = muistipaikan ylätavu (0–28)
LKM = tavujen lukumäärä (1–34)
TA = tavut

Tavujen lukumäärä voi teoriassa olla 255, mutta koska syöttöpuskurin pituus on vain 40, voidaan kerralla lähettää korkeintaan 34 tavua.

Tämän käskyn avulla voidaan monia levyaseman piirteitä muuttaa, koska useat DOS:in parametrit sijaitsevat RAM-muistissa. Voimme esimerkiksi muuttaa levyaseman laitenumeron tai ennen virheilmoituksen antamista suoritettavien lukuyritysten lukumäärän ($6A, oletusarvo 5).

Muistipaikassa $77 (119) olevaa arvoa $28 = $20 + $08 = 32 + 8 = 40 muuttamalla voidaan muuttaa vastaanottavan laitteen (listener address) laitenumero. Muuttamalla muistipaikassa $78 (120) olevaa arvoa $48 = $40 + $08 = 64 + 8 = 72 saadaan vastaavasti lähettävän laitteen (talker address) laitenumero muutetuksi. Laitenumerot ovat tavallisesti samat, mutta voivat olla erisuuretkin. Levyaseman laitenumeroksi saadaan 9, kun muistipaikkan 119 sijoitetaan arvo 32 + 9 = 41 ja muistipaikkaan 120 arvo 64 + 9 = 73. Tämä tapahtuu seuraavilla käskyillä.

```
OPEN 15,8,15
PRINT#15,"M-W"CHR$(119)CHR$(0)CHR$(2)CHR
$(41)CHR$(73)
CLOSE 15
```

Kuva 8.18

Jos levyasemia on kaksi, kytketään levyasema, jonka laitenumero on 8, muuttamisen ajaksi verkosta. Kertaamme edelliset käskyt vielä yleisessä muodossa. LN on uusi laitenumero.

```
OPEN 15,8,15
PRINT#15,"M-W"CHR$(119)CHR$(0)CHR$(2)CHR
$(32+LN)CHR$(64+LN)
CLOSE 15
```

Kuva 8.19

Eräitä pelejä ladattaessa levyasemasta kuuluu pientä kolinaa, joka johtuu siitä, että luku- ja kirjoituspää yrittää lukea virheellistä sektoria. Sektorilla oleva virhe on yleensä osa ohjelman suojausta. Kolina voidaan useimmiten poistaa lukuyritysten määrää muuttamalla (muistipaikka $6A = 106). Tämä käy seuraavalla M-W-käskyllä:

```
OPEN 15,8,15
PRINT#15,"M-W"CHR$(106)CHR$(0)CHR$(1)CHR
$(133)
CLOSE 15
```

Kuva 8.20

Edelliset toimenpiteet on helpompi suorittaa erikoisen käyttäjän tiedoston (USER), ns. &-tiedoston, avulla.

MEMORY-EXECUTE

Tämän käskyn avulla voidaan suorittaa jokin levyaseman ROM-rutiineista tai 1541:n RAM-muistiin kirjoitettu konekielinen ohjelma. Käsky lyhennetään M-E.

Syntaksi:

```
PRINT#TN,"M-E"CHR$(AL)CHR$(YL)
```

tai

```
PRINT#TN,"M-E:"CHR$(AL)CHR$(YL)
```

Kuva 8.21

TN = komentokanavaa avattaessa käytetty looginen tiedostonumero (1–127)
AL = aloitusosoitteen alatavu
YL = aloitusosoitteen ylätavu

Jos kysymyksessä ei ole ROM-rutiini, konekielinen ohjelma on ensin kirjoitettava M-W-käskyllä. Konekoodin viimeisenä komentona pitää olla RTS ($60 = 96).

Seuraava ohjelma suorittaa muistipaikasta $0110 = 272 = 16 + 256 * 1 alkavan konekielisen ohjelman.

```
OPEN 15,8,15
PRINT#15,"M-E"CHR$(16)CHR$(1)
CLOSE 15
```

Kuva 8.22

Tämän käskyn käyttöä rajoittaa käytettävissä oleva RAM-muistin määrä. Puskureita on vähän ja niistä on aina joku käytössä. Näin ollen konekoodin päälle saattaa tulla muuta tietoa. Jos koodia on vähän, eikä suhteellisia tiedostoja käytetä, koodi voidaan sijoittaa ko. tiedoston käyttöön varatuille muistipaikoille, esim. $00B5–$00BA, $00BB–$00C0 tai $00C7–$00CC (ks. muistikartta). ROM-rutiineja tutkimalla voi myös löytyä erikoisia asioita.

```
10 OPEN 1,8,15
20 PRINT#1,"M-E"CHR$(70)CHR$(242)
30 GET#1,A$:PRINT A$;:IF ST<>64 THEN 30
40 CLOSE 1
```

Kuva 8.23

BLOCK-ALLOCATE

Tämän käskyn avulla voidaan määrättyä uraa ja sektoria vastaava lohko merkitä varatuksi BAM:iin. Lohko varataan asettamalla sitä vastaava bitti nollaksi (katso 9. luku). Käsky lyhennetään B–A.

Syntaksi:

```
PRINT#TN,"B-A";AS;U;S

tai

PRINT#TN,"B-A:";AS;U;S
```

Kuva 8.24

TN = komentokanavaa avattaessa käytetty looginen tiedostonumero (1–127)
AS = asemanumero, aina 0
U = ura (1–35)
S = sektori (0-uran yläraja)

B-A-käskyllä varattuun lohkoon ei voida kirjoittaa esimerkiksi tavallisella SAVE-käskyllä, mutta lohkoon voidaan kirjoittaa U2 (BLOCK WRITE)-käskyllä.

Lohko merkitään varatuksi vain levyaseman BAM-puskuriin eikä levykkeelle. Ennen käskyn käyttöä levyke on initialisoitava I-komennolla. Muutettu BAM kirjoitetaan levykkeelle vasta sitten, kun luku/kirjoituskanava suljetaan tai ohjelma tallennetaan levykkeelle. Katsotaan esimerkkiä. Käytä ohjelmaa suorittaessasi varmuuden vuoksi jotain vähemmän tärkeätä levykettä.

```
10 OPEN 1,8,15,"I0"
20 PRINT#1,"B-A";0:1;0
30 INPUT#1,VN,VI$,VU,VS
40 PRINT VN;VI$;VU;VS
50 CLOSE 1
```

Kuva 8.25

Jos lohko on entuudestaan varattu, saadaan virheilmoitus NO BLOCK VU VS. Tällä kertaa VU ja VS ilmoittavat seuraavan vapaan ura/sektorin (kasvavassa ura/sektori-järjestyksessä). Jos levyke on täynnä, saadaan virheilmoitus 67 ILLEGAL TRACK OR SECTOR 36 1 (ei 72 DISK FULL). Virheilmoitukset ovat selityksineen liitteessä 2. Miksi entuudestaan varattu lohko pitää varata uudestaan? Syynä on se, että tällä tekniikalla luotiin satunnaistiedostoja (random access files) ennen suhteellisten tiedostojen aikaa. Sen vuoksi saadaan myös ilmoitus seuraavasta vapaasta urasta/sektorista.

Varattu lohko pysyy varattuna, kunnes BAM päivitetään VALIDATE-komennolla. Jos ura/sektori kuuluu johonkin ohjelmaan tai tiedostoon, sitä ei merkitä vapaaksi (vastaavan lohkon bitti 1). Sisällysluettelon (ura 18) kanssa pitää olla varovainen, jotta sen vapaita lohkoja ei varata. VALIDATE-komento ei vapauta 18. uran lohkoja.

Seuraavan ohjelman avulla voidaan kaikki vapaana olevat ohkot merkitä varatuiksi BAM:iin.

```
10 REM VARAA KAIKKI
20 OPEN 15,8,15,"I0"
30 INPUT#15,VN,VI$,VU,VS
40 IF VN<>0 THEN 190
50 OPEN 2,8,2,"#"
60 U=1:S=0
```

```
70 PRINT#15,"B-A";0;U;S
80 INPUT#15,VN,VI$,VU,VS
90 IF VN=0 THEN 70
100 IF VN<>65 THEN 200
110 VA=VA+1
120 PRINT U,S,VA
130 U=VU
140 IF U=0 THEN 170
150 IF U=18 THEN U=19:S=0:GOTO 70
160 S=VS:GOTO 70
170 CLOSE 2
180 INPUT#15,VN,VI$,VU,VS
190 CLOSE 15:END
200 PRINT:PRINT VN;VI$;VU;VS
210 CLOSE 2
220 INPUT#15,VN,VI$,VU,VS
230 CLOSE 15:END
```

Kuva 8.26

20	levyaseman initialisointi
30	virheilmoituksen lukeminen
40	jos virheen numero ei ole 0, ohjelman suoritus lopetetaan
50	suorasaantikanavan avaaminen
60	U = uran numero, S = sektorin numero
70	B–A -komento
80	virheilmoituksen lukeminen
90	jos VN = 0, niin ura/sektori ei ollut varattu
100	jos VN < > 65, niin jokin on vialla
110	VA = laskuri, joka ilmoittaa varattujen sektoreiden lukumäärän
120	tulostetaan ura, sektori ja laskurin arvo
130	U:n arvoksi asetetaan seuraava vapaa ura
140	jos U = 0, kaikki on jo varattu
150	sisällysluetteloa ei varata
160	S:n arvoksi asetetaan seuraava vapaa sektori
170	suljetaan suorasaantikanava
200–230	virheen käsittely

B–A-käsky ei aina toimi oikein. Jos lohko on ennestään vapaa, kaikki on kunnossa ja B–A toimii poikkeuksetta oikein. Mutta jos lohko on ennestään varattu, kaikki ko. uran lohkot

saatetaan merkitä varatuiksi. Lisäksi virheilmoitus NO BLOCK voi ilmoittaa seuraavaksi vapaaksi uraksi/sektoriksi 18. uran sektorin, vaikka sisällysluetteloura on suojattu. Käskyä käytettäessä pitää olla varovainen. On ehkä paras varata lohko suoraan ilman B-A-käskyä ja asettaa lohkoa vastaava bitti uralla 18/0 nollaksi. Näin on tehty seuraavassa ohjelmassa.

```
150 BP=2+(U-1)*4:BP=BP+1
160 IF U>7 THEN BP=BP+1:S=S-7:GOTO 160
170 OPEN 15,8,15
180 OPEN 2,8,2,"#"
190 PRINT#15,"U1";2;0;18;0
200 PRINT#15,"B-P";2;BP
210 GET#2,BY$:IF BY$=""THEN BY$=CHR$(0)
220 PRINT#15,"B-P";2;BP
230 BY=ASC(BY$)
240 BY=BY OR 2^(7-S)
250 PRINT#2,ASC(BY);
260 PRINT#15,"U2";2;0;18;0
270 CLOSE 2:CLOSE 15
```

Kuva 8.27

Edellinen ohjelma sopii hyvin käytettäväksi aliohjelmana. Ennen aliohjelmaan siirtymistä pitää ilmoittaa ura (U) ja sektori S. Muuttuja BP ilmoittaa puskuriosoittimen arvon.

BLOCK-FREE

Tämän käskyn avulla voidaan määrättyä uraa ja sektoria vastaava lohko merkitä vapaaksi BAM:iin. Lohko on vapaa, kun sitä vastaava bitti on 1. Käsky lyhennetään B-F.

Syntaksi:

```
PRINT#TN,"B-F";AS;U;S
```

tai

```
PRINT#TN,"B-F:";AS;U;S
```

Kuva 8.28

TN = komentokanavaa avattaessa käytetty looginen tiedostonumero (1-127)

AS = asemanumero, aina 0
U = ura (1–35)
S = sektori (0-uran yläraja)

Lohkoa vastaava bitti muutetaan vain BAM-puskurissa eikä levykkeellä. Siksi levyke on initialisoitava (I) ennen tätä käskyä. Näin BAM-puskuriin saadaan levykkeen BAM. Muutettu BAM kirjoitetaan levykkeelle vasta sitten, kun jokin luku/kirjoituskanava suljetaan tai ohjelma tallennetaan. Seuraava esimerkki-ohjelma vapauttaa 4. uran 8. sektoria vastaavan lohkon.

```
10 OPEN 15,8,15,"I0"
20 OPEN 2,8,2,"#"
30 PRINT#15,"B-F";0;4;8
40 CLOSE 2:CLOSE 15
```

Kuva 8.29

Suorasaantikanava pitää sulkea ja avata, jotta ohjelma toimii oikein. Käytä jotain harjoituslevykettä.

BLOCK-EXECUTE

Levykkeen jonkin lohkon sisältö luetaan tämän käskyn avulla puskuriin ja puskurin sisältö suoritetaan konekieliohjelmana. Käsky lyhennetään B–E.

Syntaksi:

```
PRINT#TN,"B-E";KN;AS;U;S
```

tai

```
PRINT#TN,"B-E:";KN;AS;U;S
```

tai

```
PRINT#TN,"B-E:KN,AS,U,S"
```

Kuva 8.30

TN = komentokanavaa avattaessa käytetty looginen tiedostonumero (1–127)
KN = suorasaantikanavaa avattaessa käytetty kanavanumero (toisio-osoite) (2–14)

AS = asemanumero, aina 0
U = ura (1–35)
S = sektori (0-uran yläraja)

Käskyn käyttäminen on sallittua silloin, kun lohkon sisältämät tavut muodostavat konekieliohjelman, joka voidaan suorittaa 1541:ssä. Konekieliohjelman viimeisen komennon pitää olla RTS ($60 = 96), muuten DOS saattaa "romahtaa".

Suorasaantikanavaa avattaessa ilmoitetaan yleensä puskurin numero, jos koodi on kirjoitettu määrättyä puskuria varten eikä ole siirrettävissä. B–E-käsky voidaan korvata käskyillä U1 ja M–E.

```
10 OPEN 15,8,15
20 OPEN 2,8,2,"#2"
30 PRINT#15,"B-E";2;0;7;2
40 CLOSE 2:CLOSE 15
```

Kuva 8.31

Ohjelma lukee 7. uran 2. sektorin sisällön puskuriin 2 ja suorittaa sen siellä konekieliohjelmana. Ohjelmaa ei nyt kannata suorittaa, ellei levykkeen 7. uran 2. sektorilla ole jotain konekielistä ohjelmaa.

USER-käskyt

Näiden käskyjen avulla voidaan käyttää standardeja DOS-rutiineja tai 1541:n RAM-muistiin sijoitettuja konekielisiä ohjelmia. Käskyt voidaan jakaa kolmeen ryhmään. U1- ja U2-käskyt korvaavat käskyt B–R ja B–W. U0, U9 ja U: ovat eräänlaisia levyaseman päivityskäskyjä. U0 asettaa nollasivulla olevan USER JUMP -vektorin oletusarvon, U9 ja U: saavat aikaan resetin, eli levyasema palaa kytkennän jälkeiseen tilaan. Käskyt U3–U8 kulkevat käsi kädessä MEMORY-käskyjen kanssa. Nämä käskyt aiheuttavat hyppytaulukossa olevan osoitteen perusteella hypyn johonkin määrättyyn osoitteeseen ja suorittavat siellä olevan konekielisen ohjelman.

KÄSKY		TOIMINTA
U0		asettaa USER JUMP -vektorin
U1	(UA)	korvaa käskyn B–R ($CD5F)
U2	(UB)	korvaa käskyn B–W ($DC97)
U3	(UC)	hyppy osoitteeseen $0500

U5	(UE)	hyppy osoitteeseen $0506
U6	(UF)	hyppy osoitteeseen $0509
U7	(UG)	hyppy osoitteeseen $050C
U8	(UH)	hyppy osoitteeseen $050F
U9	(UI)	hyppy osoitteeseen $FFFA (reset)
U:	(UJ)	reset ($EAA0)

Syntaksi:

```
PRINT#TN,"UX"
```

Kuva 8.32

TN = komentokanavaa avattaessa käytetty looginen tiedostonumero
X = luku 0–9 tai :.

Taulukossa sulkeisiin merkityt vaihtoehtoiset muodot eivät useimmiten toimi, joten niitä ei kannata käyttää. U:-käsky vastaa virran kytkemistä pois ja takaisin. U9 toimii lähes samalla tavalla mutta ilman diagnostiikkaa (merkkivalo ei välähdä). Jokaisen vektorin pituus on kolme tavua. Jos halutaan hypätä jollekin toiselle alueelle, sijoitetaan JMP ($4C) ja osoite (ala/ylä).

Käsky UI-asettaa levyaseman VIC-20:n nopeudelle. Käskyn UI+ pitäisi asettaa levyasema C64:n nopeudelle, mutta minun levyasemani (tai C64) ainakin lukkiutuu.

Kokeile seuraavaa ohjelmaa.

```
10 OPEN 15,8,15
20 PRINT#15,"U:"
30 FORT=1TO1000:NEXT
40 GET#15,X$:PRINTX$;:IF ST<>64 THEN 40
50 CLOSE 15

READY.
```

Kuva 8.33

Rivillä 20 voit lisäksi kokeilla käskyjä U9, UJ, U; ja UI. Kokeilujeni perusteella UI ei toimi, muilla tulostus on 72,CBM DOS V2.6 1541,00,00. Eri muotoja kokeiltaessa RAM voi sekoittua niin pahasti, että levyasemasta on välillä kytkettävä virta pois. Ohjelman rivillä 30 oleva viive odottaa poiskytkeytymistä. Päällekytkeytymisilmoitus luetaan rivillä 40. Jos rivin 30

viive jätetään pois, ilmoitusta ei anneta.

Seuraavassa esimerkissä sijoitetaan M-W-käskyllä muistipaikasta $0500 alkaen käsky JMP $F1DA. Tämä suoritetaan komennolla U3.

```
10 OPEN 1,8,15
20 PRINT#1,"M-W"CHR$(0)CHR$(5)CHR$(3)CHR
$(76)CHR$(218)CHR$(241)
30 PRINT#1,"U3"
40 GET#1,A$:PRINTA$;:IF ST<>64 THEN 40
50 CLOSE 1
```

Kuva 8.34

Ohjelma tulostaa virheilmoituksen 67,ILLEGAL TRACK OR SECTOR,00,00

LISÄÄ TEORIAA

Tässä osassa tarkastellaan hieman yksityiskohtaisemmin levyaseman toimintaperiaatteita eli miten DOS:in kanssa keskustellaan.

Kaikki edellä esitetyt komennot lähetetään komentokanavan kautta levyaseman RAM-muistin komentopuskuriin ($0200–$0229). Annetun komennon tulkitsee 6502 Interface Processor (IP), joka jakaa komennon pienempiin osatehtäviin eli töihin (jobs). Kullakin tehtävällä tai työllä on oma koodinsa, joka sijoitetaan ns. työjonoon (job queue). Työjono sijaitsee alueella $0000–$0005. IP asettaa myös RAM-muistissa kutakin työtä varten tarvittavat alkuarvot. Floppy Disk Controller (FDC) käy työjonon läpi 10 ms:n välein ja tarkistaa, onko jonossa suoritettavia töitä. Jos töitä löytyy, FDC suorittaa ne.

FDC voi suorittaa seuraavat työt:

TYÖ	KOODI		
READ	$80	(128)	– lukee lohkon
WRITE	$90	(144)	– kirjoittaa lohkon
VERIFY	$A0	(160)	– tarkistaa lohkon
SEEK	$B0	(176)	– etsii määrätyn lohkon
BUMP	$C0	(192)	– siirtää luku- ja kirjoituspään uralle 1
JUMP	$D0	(208)	– hyppy konekielirutiiniin
EXECUTE	$E0	(224)	– suorittaa rutiinin

Kun FDC on suorittanut löytämänsä työn virheettömästi tai virheellisesti, se sijoittaa jonossa olleen työkoodin paikalle koodin, joka ilmoittaa työn tilan. Näin FDC ilmoittaa IP:lle, että työ on valmis tai että jokin virhe on tapahtunut. IP muuntaa FDC:n antaman virhekoodin ja sijoittaa sen virhepuskuriin ($02D6–$02F8) siinä muodossa, jonka käyttäjä saa luettuaan virhekanavan.

FDC:n KOODI	IP:n KOODI	VIRHEILMOITUS
$01 (1)	0	OK
$02 (2)	20	READ ERROR
$03 (3)	21	READ ERROR
$04 (4)	22	READ ERROR
$05 (5)	23	READ ERROR
$07 (7)	25	WRITE ERROR
$08 (8)	26	WRITE PROTECT ON
$09 (9)	27	READ ERROR
$0A (10)	28	WRITE ERROR
$0B (11)	29	READ ERROR

Virheilmoitukset on esitetty selityksineen tarkemmin liitteessä 2.

Kullekin työjonossa ($0000–$0005) olevalle työlle varataan puskurialue. Lisäksi on ilmoitettava ura/sektori, jota lukeminen/kirjoittaminen koskee. BUMP- ja JUMP-töissä ei tarvita uraa eikä sektoria. Ura ja sektori ilmoitetaan omassa taulukossaan $0006–$0011. Työjonossa oleva työ, puskurialue ja ura-sektoritaulukko vastaavat toisiaan seuraavasti:

TYÖJONO	PUSKURI	URA-SEKTORI
$0000	0 ($0300 –$03FF)	$06 – $07
$0001	1 ($0400 – $04FF)	$08 – $09
$0002	2 ($0500 – $05FF)	$0A – $0B
$0003	3 ($0600 – $06FF)	$0C – $0D
$0004	4 ($0700 – $07FF)	$0E – $0F
$0005	5 ei RAM	$10 – $11

Jos esimerkiksi kirjoitamme (U2) lohkon 12/2 (ura/sektori) levykkeelle käyttämällä puskuria 0, tapahtuu IP:n tulkittua komennon seuraavaa. FDC löytää työn koodin $90 (144) työjonon paikasta $0000, uran (12) paikasta $06 ja sektorin (2) paikasta $07. Kun työ on suoritettu, FDC asettaa työjonon paikkaan $0000 ilmoituksen siitä, että työ on valmis. Jos kaikki

sujuu hyvin, paikkaan $0000 sijoitetaan arvo $01 (1). Virheen tapahtuessa sinne sijoitetaan vastaava virhekoodi. IP muuntaa ilmoituksen selväkieliseksi, ja käyttäjä voi lukea ilmoituksen INPUT#- tai GET#-käskyllä.

Tarkastellaan asiaa vielä esimerkin avulla. Luetaan 18. uran 0-sektorilta levykkeen ID, joka löytyy lohkon tavuista 162–163 ($A2–$A3).

```
100 OPEN 1,8,15,"I0"
120 REM HAE (SEEK)
130 U=18:S=0
140 TY=176
150 GOSUB 300
160 IF V<>1 THEN 250
170 REM LUE (READ)
180 TY=128
190 GOSUB 300
200 IF V<>1 THEN 250
210 PRINT#1,"M-R"CHR$(162)CHR$(3)CHR$(2)
220 FOR I=1 TO 2
230 GET#1,A$:A$=A$+CHR$(0)
240 PRINTA$;:NEXT
250 CLOSE 15:END
300 Y=0
310 PRINT#1,"M-W"CHR$(6)CHR$(0)CHR$(2)CH
R$(U)CHR$(S)
320 PRINT#1,"M-W"CHR$(0)CHR$(0)CHR$(1)CH
R$(TY)
330 Y=Y+1
340 PRINT#1,"M-R"CHR$(0)CHR$(0)
350 GET#1,V$:V=ASC(V$+CHR$(0))
360 IF Y=500 THEN RETURN
370 IF V>127 THEN 330
380 RETURN
```

Kuva 8.35

100	komentokanavan avaaminen ja initialisointi
130–150	ilmoitetaan ura, sektori ja työn SEEK-koodi sekä siirrytään aliohjelmaan suorittamaan työ
160	virheen tarkistus: jos kaikki ei ole kunnossa, keskeytetään
180–190	ilmoitetaan työn READ-koodi ja siirrytään tekemään työ

200	virheen tarkistus
210–240	luetaan ja tulostetaan levyaseman puskurista (0) kaksi merkkiä alkaen tavusta 162 (ID)
250	suljetaan komentokanava ja lopetetaan

Aliohjelma:

300	aikamuuttuja nollaksi
310	kirjoitetaan ura ja sektori muistipaikkoihin $06 ja $07
320	kirjoitetaan työn koodi muistipaikkaan $00
330–380	FDC suorittaa työn. Lukemalla muistipaikka $00, saadaan selville onko työ tehty

Huomaa, että työn koodi on aina suurempi kuin 127 (bitti 7 on 1) ja virheen koodi pienempi kuin 128 (bitti 7 on 0). Tehtyäsi työn muista korvata sen koodi virheen koodilla. Jos koodi > 127, työ ei ole valmis ja riviltä 370 siirrytään riville 330.

Kun käytetään työjonoa ohjelmoijan pitää itse selvittää työn tila. FDC:lle annetaan aikaa työn tekemiseen (aikamuuttuja Y). On erittäin tärkeää, että työn paikka jonossa, puskuri ja ura/sektoritaulukko vastaavat toisiaan ja ovat varmasti oikein. Jos ilmoitettu ura ja sektori ylittävät sallitut rajat, luku/kirjoituspää saattaa lukkiutua. Jos työn koodi on väärä, FDC yrittää tehdä olematonta työtä, minkä seurauksena levyasema lukkiutuu.

JUMP ($D0) suorittaa RAM:iin sijoitetun konekielisen ohjelman. Kun FDC suorittaa työtä EXECUTE ($E0), sitä ei voi keskeyttää ennen kuin työ on valmis. Lisäksi ohjelmoijan pitää konekielisen ohjelman päätyttyä itse korvata työkoodi virhekoodilla $01. Muussa tapauksessa FDC suorittaa ohjelman uudestaan.

KÄYTTÄJÄN &-TIEDOSTOT

Näillä tarkoitetaan erityisellä tavalla muotoiltuja käyttäjän (USR) tiedostoja, jotka levyasema lataa ja suorittaa automaattisesti, kun ohjelman nimi lähetetään komentokanavaa pitkin levyasemalle.

Kaikki ohjelman suorittamiseksi tarvittava tieto kirjoitetaan tiedostoon, joka avataan kirjoittamista varten seuraavasti:

```
OPEN TN,LN,KN,"Ø:&:NIMI,U,W"
```

Kuva 8.36

missä:

TN = looginen tiedostonumero (1–127)
LN = laitenumero (8–11), tavallisesti 8
KN = kanavanumero (toisio-osoite (2–14)
& = tiedoston tunnus
NIMI = tiedoston nimi
U = tiedoston tyyppi, voi olla myös USR
W = tila, voi olla myös WRITE

Nimen edessä pitää olla &:. Myös sisällysluettelossa tiedoston nimen edessä on &:. Koska tiedosto on käyttäjän määrittelemä, sen tyyppinä on sisällysluettelossa USR.

Levykkeellä tiedoston rakenne on seuraava:

TAVU	SISÄLTÖ
0,1	seuraava ohjelman käyttämä ura ja sektori (jos käytössä)
2	alkuosoitteen alatavu
3	alkuosoitteen ylätavu
4	ko. sektorilla olevan ohjelman pituus tavuina
5-N	konekielinen ohjelma
N+1	tarkistussumma

Jos ohjelman pituus on suurempi kuin 250 tavua, pitää samat tiedot antaa myös seuraavalla sektorilla.

Tarkistussumma pitää olla jokaisen 256 tavun jälkeen sekä tavussa 3 ilmoitetun ohjelman pituuden jälkeen. Ohjelmoijan pitää itse ilmoittaa tarkistussumma. Se saadaan laskemalla ensin yhteen tavut 2-N ja sitten yhteen saadun summan ala- ja ylätavu. Jos tavut 2-N ovat esimerkiksi 0, 3, 5, 169, 197, 133, 106 ja 96, tarkistussumma on 199, sillä 0 + 3 + 5 + 169 + 197 + 133 + 106 + 96 = 709 = 197 + 2 * 256. Ala- ja ylätavun summa on 197+2 = 199.

Kun tiedosto on valmis, se voidaan suorittaa komennolla.

```
OPEN 15,8,15:PRINT#15,"&:NIMI"
```

Kuva 8.37

Tehdään esimerkiksi sellainen &-tiedosto, joka muuttaa levyaseman laitenumeron 9:ksi. Kuten muistamme, muistipaikkaan $77 (listener address) pitää sijoittaa arvo $09 + $20 ja muistipaikkaan $78 (talker address) arvo $09 + $40.

```
10 REM &-TIEDOSTO
20 OPEN 2,8,2,"0:&:LAITE9,U,W"
30 FOR X=1 TO 13:READ Y:PRINT#2,CHR$(Y);
:NEXT
40 CLOSE 2
100 DATA 0,3     :REM ALKUOSOITE $0300
110 DATA 9       :REM OHJELMAN PITUUS
120 DATA 169,41  :REM LDA #$29
130 DATA 133,119:REM STA $77
140 DATA 169,73  :REM LDA #$49
150 DATA 133,120:REM STA $78
160 DATA 96      :REM RTS
170 DATA 45      :REM TARKISTUSSUMMA
```

Kuva 8.38

20	tiedoston avaaminen kirjoittamista varten
30	luetaan tavut ja kirjoitetaan ne tiedostoon
40	suljetaan tiedosto
100–170	tavut

Kun suoritat ohjelman RUN-käskyllä, levylle muodostuu tiedosto &:LAITE9, jonka tyyppi on USR.
Kun haluat muuttaa levyaseman laitenumeron, kirjoita

```
OPEN 15,8,15:PRINT#15,"&:LAITE9":CLOSE15
```

Kuva 8.39

&-tiedostojen käyttöä rajoittaa levyaseman RAM-muistin pieni koko, mutta siitä huolimatta ne ovat hyödyllisiä, koska pelkästään joidenkin muistipaikkojen sisältöä muuttamalla voidaan säädellä DOS:in käyttäytymistä.

9. LUKU

LEVYKKEEN SISÄINEN RAKENNE

Tässä luvussa käsitellään levykkeen sisäistä loogista rakennetta. Tarkoitus on selvittää sektorin rakenne, eri urille ja sektoreille tallennetut tiedot, BAM ja sisällysluettelo sekä eri tiedostotyyppien tallennus.

Yhdistämällä edellisessä luvussa käsitellyt suorasaantikäskyt ja tässä luvussa ilmoitetut rakenneseikat voit luoda useita hyödyllisiä apuohjelmia, jotka helpottavat ja yksinkertaistavat levyasematyöskentelyä.

URAT JA SEKTORIT

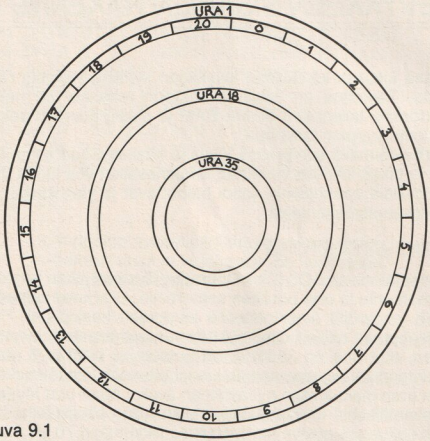
Formatoinnissa DOS (Disk Operating System) jakaa levykkeen pinnan uriin ja urat edelleen sektoreihin. Urat ovat samankeskisiä ympyröitä (keskipisteenä levykkeen keskipiste). Tiedot tallennetaan näille urille. 1541-levyasema formatoi levykkeen siten, että uria on 35. Urat on numeroitu 1–35 ja 1. ura on uloimpana. Levyaseman luku- ja kirjoituspää löytää ensimmäisen uran mekaanisen pysäyttimen avulla. Muut urat löydetään ensimmäisestä urasta lasketun etäisyyden perusteella. Luku- ja kirjoituspää pystyy periaatteessa liikkumaan 70 eri etäisyydelle ensimmäisestä urasta. Puolikasurille ei kuitenkaan voida tehokkaasti tallentaa tietoa luku- ja kirjoituspään fyysisen leveyden takia (noin 0,4 mm). Jos uralle 12,5 kirjoitetaan tietoa, pyyhitään samalla urat 12 ja 13 tyhjiksi. Ohjaamalla askelmoottoria M–W-käskyllä voidaan siirtyä jopa uralle 37.

Jokainen ura on jaettu 17 tai useampaan sektoriin. Kuhunkin sektoriin voidaan tallentaa 256 tavua tietoa. Sektoreiden lukumäärä urilla riippuu uran järjestysnumerosta seuraavan taulukon mukaisesti:

URAT	SEKTORIT	SEKTORIT YHT.	TAVUJA/URA
1 – 17	0 – 20	21	5376
18 – 24	0 – 18	19	4864
25 – 30	0 – 17	18	4608
31 – 35	0 – 16	17	4352

Sektroreita on yhteensä 683. Näin levykkeelle mahtuu kaiken kaikkiaan 683 * 256 tavua = 174848 tavua. Koska sisällysluettelo varaa 18. uran, jää ohjelmille ja tiedostoille 664 sektoria = 664 * 256 tavua = 169984 tavua.

Kaaviomainen levykkeen rakenne on seuraava:



Kuva 9.1

SEKTORIN RAKENNE

Jokaisessa sektorissa on kaksi lohkoa:

- otsikkolohko, joka toimii sektorin tunnistimena;
- tietolohko, jolle tallennetaan varsinaiset tiedot eli 256 tavua.

Kumpaakin lohkoa edeltää erikoinen tahdistusmerkki, joka muodostuu kymmenestä tai useammasta (tavallisesti 40) ykkösbitistä. Kahden peräkkäisen sektorin välissä on ns. sektoriväli (inter-sector gap), jonka pituus on 4–12 tavua.

						⟼ UUSI SEKTORI	
TAHDISTUS-MERKKI	OTSIKKO-LOHKO	OTSIKKO-VÄLI	TAHDISTUS-MERKKI	TIETOLOHKO	SEKTORI-VÄLI	TAHDISTUS-MERKKI	OTSIKKO-LOHKO

Kuva 9.2

DOS tunnistaa uran ja sektorin otsikkolohkon avulla. Otsikkolohkosta voidaan erottaa tahdistusmerkki, kahdeksan tavua tunnistamiseen käytettyä tietoa ja otsikkoväli. Otsikkolohko kirjoitetaan levykkeelle vain formatoinnin yhteydessä, ei koskaan myöhemmin.

TAHDISTUS-MERKKI	OTSIKKOLOHKON TUNNUS	TARKISTUS-SUMMA	SEKTORI	URA	10 TAVU 2	10 TAVU 1	$0F	$0F	OTSIKKO-VÄLI

Kuva 9.3

Tahdistusmerkki: 10 tai useampia ykkösbittejä, joista DOS (tai levyohjain) huomaa, että tulossa on otsikkolohko tai tietolohko.

Otsikkolohkon tunnusmerkki (ID): tavallisesti $08. Sen avulla DOS erottaa otsikko- ja tietolohkon toisistaan. Jos tässä kohden on jokin muu arvo, saadaan sektoria luettaessa virheilmoitus 20 READ ERROR.

Otsikkolohkon tarkistussumma: Tarkistusmerkin avulla DOS varmistaa, että otsikkolohko on luettu oikein.

Sektorin numero: Kyseisen sektorin numero.

Uran numero: Kyseisen uran numero, josta DOS tietää, että luku- ja kirjoituspää on oikean uran kohdalla.

Tunnusmerkki 2 (ID2): tämä on toinen NEW-formatointikäskyn yhteydessä annettavista ID-merkeistä. Jos formatointikäsky on NEW:OHJELMAT,OH, tunnusmerkki 2 on H.

Tunnusmerkki 1 (ID1): Tämä on ensimmäinen formatointikäskyn yhteydessä annettavista ID-merkeistä. Edellisen käskyn mukaan se on O. Huomaa, että ID-merkit ovat otsikkolohkossa järjestyksessä HO, vaikka ne NEW-käskyssä ovat OH.

$0F-tavut: välilyöntejä (OFF-tavut)

Otsikkoväli: Se muodostuu kahdeksasta tavusta $55. Nämä tavut antavat DOS:ille aikaa valmistautua seuraavana olevan tietolohkon lukemiseen. DOS ei lue näitä tavuja.

Katsotaan vielä miten otsikkolohkon tarkistussumma laske-

taan. Tarkistussumma muodostetaan uranumeron, sektorinumeron ja molempien tunnusnumeroiden avulla käyttämällä loogista operaatiota EOR. Kun kahta bittiä verrataan EOR-operaatiolla, tulos on joku seuraavista:

```
0 EOR 0 = 0        0 EOR 1 = 1
1 EOR 1 = 0        1 EOR 0 = 1
```

Jos uranumero on 22, sektorinumero 8 ja ID = ZX, tarkistussumma saadaan näin:

```
22 = $16 = 00010110
 8 = $08 = 00001000
 ------------------
     EOR = 00011110
 Z = $5A = 01011010
 ------------------
     EOR = 01000100
 X = $58 = 01011000
 ------------------
     EOR = 00011100  = $1C = 28
```

Otsikkolohkon tarkistussumma on siis $1C = 28.

Kun otsikkolohko luetaan, tarkistussumma lasketaan uudestaan. Saatua summaa verrataan levykkeellä olevaan tarkistussummaan. Jos ne eivät täsmää, DOS antaa virheilmoituksen 27 READ ERROR.

Tietolohkoon tallennetaan 256 tavua tietoa. Tietolohkosta voidaan erottaa tahdistusmerkki, tietolohkon tunnusmerkki (ID), tietotavut, tietolohkon tarkistussumma, kaksi välitavua ja sektoriväli. Tietolohko kirjoitetaan aina kokonaisuudessaan levykkeelle (vrt. otsikkolohko).

TAHDISTUS-MERKKI	TIETOLOHKON TUNNUS	TIETOTAVUT	TARKISTUS-SUMMA	$00	$00	SEKTORI-VÄLI

Kuva 9.4

Tahdistusmerkki: 10 tai useampia ykkösbittejä, joista DOS huomaa, että tulossa on otsikko- tai tietolohko.

Tietolohkon tunnusmerkki (ID): Tämän tavun arvo on tavallisesti $07. Sen avulla DOS tietää, että kysymyksessä on tietolohko eikä otsikkolohko ($08). Jos merkki ei ole $07, DOS

antaa virheilmoituksen 22 READ ERROR.

256 tietotavua: Ne muodostavat varsinaisen sektorilla olevan tiedon. Tavallisen tiedoston kahta ensimmäistä tavua käytetään seuraavan uran/sektorin osoittamiseen. Tiedoille jää siis vain 254 tavua.

Tietolohkon tarkistussumma: Tämän tarkistusmerkin avulla DOS varmistaa, että tavut on luettu oikein. Se muodostetaan vertaamalla kaikkia 256 tavua keskenään EOR-operaatiolla. Kun tietolohko luetaan, tarkistussumma lasketaan uudestaan ja sitä verrataan levykkeellä olevaan tarkistussummaan. Jos summat eivät ole samat, DOS antaa virheilmoituksen 23 READ ERROR.

$00-tavut: välitavuja (OFF-tavut)

Sektoriväli: Näiden tavujen tarkoituksena on antaa DOS:ille aikaa valmistautua seuraavana tulevan otsikkolohkon lukemiseen. Välin pituus vaihtelee eri urien ja eri levyasemien kesken.

LOHKOKARTTA ELI **BAM**

Lohkokartta eli BAM (Block Availability Map) sijaitsee 18. uralla sektorilla 0. Sinne DOS merkitsee tietoa sisältävät lohkot (sektorit) ja vapaat lohkot.

Ensimmäisessä luvussa tutkimme levykkeen sisällysluetteloa siinä muodossa, jossa se saadaan näyttöön käskyillä LOAD"$",8 ja LIST. Samassa yhteydessä selvitimme kunkin tulostetun tiedon merkityksen. Nyt tutkimme, miten osa näistä tiedoista löytyy BAM:ista ja seuraavassa kohdassa tarkastelemme sisällysluettelon loppuosaa.

Kaikki sektoreiden sisältöjen tulostukset on suoritettu kirjan levykkeellä olevalla ohjelmalla TUTKI U&S. Ohjelman avulla voit tulostaa samat kuvat kirjoittimella tai näyttöön.

Seuraava kuva esittää kirjan levykkeen BAM:ia.

```
URA : 18   SEKTORI : 0

00: 12 01 41 00 15 FF FF 1F  ..A.....
08: 15 FF FF 1F 15 FF FF 1F  ........
10: 15 FF FF 1F 15 FF FF 1F  ........
18: 15 FF FF 1F 15 FF FF 1F  ........
20: 15 FF FF 1F 15 FF FF 1F  ........
28: 15 FF FF 1F 15 FF FF 1F  ........
30: 15 FF FF 1F 15 FF FF 1F  ........
38: 15 FF FF 1F 05 00 03 1C  ........
40: 00 00 00 00 00 00 00 00  ........
```

```
48: 0F 6C FF 07 00 00 00 00  ........
50: 00 00 00 00 13 FF FF 07  ........
58: 13 FF FF 07 13 FF FF 07  ........
60: 13 FF FF 07 12 FF FF 03  ........
68: 12 FF FF 03 12 FF FF 03  ........
70: 12 FF FF 03 12 FF FF 03  ........
78: 12 FF FF 03 11 FF FF 01  ........
80: 11 FF FF 01 11 FF FF 01  ........
88: 11 FF FF 01 11 FF FF 01  ........
90: 4B 49 52 4A 41 4C 45 56  KIRJALEV
98: 59 4B 45 A0 A0 A0 A0 A0  YKE.....
A0: A0 A0 41 53 A0 32 41 A0  ..AS.2A.
A8: A0 A0 A0 00 00 00 00 00  ........
B0: 00 00 00 00 00 00 00 00  ........
B8: 00 00 00 00 00 00 00 00  ........
C0: 00 00 00 00 00 00 00 00  ........
C8: 00 00 00 00 00 00 00 00  ........
D0: 00 00 00 00 00 00 00 00  ........
D8: 00 00 00 00 00 00 00 00  ........
E0: 00 00 00 00 00 00 00 00  ........
E8: 00 00 00 00 00 00 00 00  ........
F0: 00 00 00 00 00 00 00 00  ........
F8: 00 00 00 00 00 00 00 00  ........
```

Kuva 9.5

BAM:in 256 tavua on järjestetty kahdeksan tavun riveihin, joita on kaikkiaan 32. Äärimmäisenä oikealla on kunkin rivin ensimmäisen tavun järjestysnumero heksadesimaalilukuna. Jokaisella rivillä on ensin kahdeksan tavun sisältö heksadesimaalilukuna ja sitten selväkielisinä niiden tavujen sisällöt, joita vastaavat ASCII-koodit 32–96. Muut tavut on merkitty pisteinä.

Katsotaanpa, mitä tietoa BAM tarjoaa. Tavun sisältö on merkitty sekä heksadesimaali- että desimaalilukuna (sulkeissa).

TAVUT	SISÄLTÖ JA MERKITYS		
0	$12	(18),	vars. sis.luettelon ura
1	$01	(01),	sisällysluettelon ensimmäinen sektori.
2	$41	(65),	A-kirjain osoittaa 1541-formaatin

3	$00 (00),	ei käytössä
4 - 143		varsinainen BAM
144 - 159		levykkeen nimi ja tarvittaessa SHIFT + välilyöntejä
160 - 161	$A0 (160),	SHIFT + välilyönti
162 - 163		levykkeen ID
164	$A0 (160),	SHIFT + välilyönti
165 - 166	$32,$41 (50,65)	DOS-versio 2 ja formaatin tunnus A
167 - 170	$A0 (160)	SHIFT + välilyönti
171 - 255	?	ei käytössä

Tavuista 0 ja 1 nähdään, että varsinainen sisällysluettelo alkaa 18. uran ensimmäiseltä sektorilta. Näitä tavuja kutsutaan osoittimiksi.

Kolmas tavu (tavu 2) on tärkeä. Jos tämän tavun arvoksi sijoitetaan jokin muu kuin $41 (A), saadaan virheilmoitus DOS MISMATCH. Levykkeen voi suojata kirjoittamiselta sijoittamalla tähän arvoksi esimerkiksi $45 (E).

Seuraavat 140 tavua sisältävät BAM:in (lohkokartan). Kullekin uralle on varattu neljä tavua siten, että ensimmäisen uran käytössä on tavut 4–7, toisen 8–11, kolmannen 12–15 jne. Katsotaan esimerkkinä uraa 15, jonka vapaat (ja varatut) lohkot on ilmoitettu tavuissa $3C–$3F (60–63).

```
38: 15 FF FF 1F 05 00 03 1C  ........
```

Kuva 9.6

Ensimmäinen tavu ilmoittaa uralla vapaana olevien lohkojen määrän. Uralla on siis $05 = 5 vapaata lohkoa. Laskemalla yhteen uria vastaavien tavuryhmien ensimmäiset tavut, saadaan levykkeellä vapaana olevien lohkojen määrä. Summaksi tulee 583 lohkoa. Tästä luvusta pitää vähentää 18. uran vapaat lohkot, joita on 15. Käyttäjällä on siis 583 - 15 = 568 lohkoa. Palataan kuitenkin uraan 15. Seuraavat kolme tavua ilmoittavat mitkä lohkot on käytössä ja mitkä vapaita. Tämä nähdään kirjoittamalla tavujen sisällöt binaarilukuina:

TAVU	$3D	$3E	$3F
ARVO	$00	$03	$1C
BIN.	00000000	00000011	00011100
LOHKO	76543210	111111 54321098	21111 09876

Vapaat lohkot on taulukon kolmannella rivillä merkitty ykkösellä (1) ja käytössä olevat nollalla (0). Viimeiseltä riviltä nähdään lohkojen järjestys. Vapaina ovat lohkot 8, 9, 18, 19, ja 20. Kolmas tavu on aina vajaa, koska uralla voi olla korkeintaan 21 sektoria (urat 1–17).

Levykkeen nimi on aina 16 merkin pituinen; nimen perään lisätään tarvittaessa merkkejä CHR$(160), SHIFT + välilyönti. Levykkeen nimi esiintyy levykkeellä vain tässä kohdassa. Sisällysluettelon tulostaminen näyttöön voidaan estää sijoittamalla nimen kuuden ensimmäisen tavun arvoiksi heksadesimaaliluvut 14 14 14 00 00 00.

Levykkeen ID (tavut 162–163) sijoitetaan alustuksessa jokaisen sektorin otsikkolohkoon. Kun levykkeen ID muutetaan, muutos tapahtuu vain tavuissa 162–163 eikä otsikkolohkoissa. Kysymyksessä on ns. kosmeettinen muutos. DOS käyttää otsikkolohkoissa olevaa ID:tä levykkeen tunnistamiseen.

Formaatin tunnus A sijaitsee kahdessa eri paikassa, tavussa 2 ja 166. DOS käyttää formaatin tunnistamiseen 2. tavun sisältöä.

Tavut 170–255 sisältävät useimmiten arvon $00, mutta sinne voidaan myös sijoittaa esimerkiksi jokin viesti.

Seuraavaksi tarkastelemme varsinaisen sisällysluettelon ensimmäistä sektoria: 18. ura, 1. sektori.

SISÄLLYSLUETTELO

Varsinainen sisällysluettelo alkaa aina 18. uran ensimmäiseltä sektorilta. Tässä on kirjan levykkeen vastaava sektori.

```
URA : 18    SEKTORI : 1

00: 12 04 82 11 00 42 41 43  .....BAC
08: 4B 55 50 A0 A0 A0 A0 A0  KUP.....
10: A0 A0 A0 A0 A0 00 00 00  ........
18: 00 00 00 00 00 00 09 00  ........
20: 00 00 82 11 01 46 49 49  .....FII
28: 4C 49 A0 A0 A0 A0 A0 A0  LI......
30: A0 A0 A0 A0 A0 00 00 00  ........
38: 00 00 00 00 00 00 08 00  ........
40: 00 00 82 11 07 54 55 54  .....TUT
48: 4B 49 20 55 26 53 A0 A0  KI U&S..
50: A0 A0 A0 A0 A0 00 00 00  ........
58: 00 00 00 00 00 00 05 00  ........
60: 00 00 82 13 00 42 41 4D  .....BAM
```

```
68: 42 41 4D A0 A0 A0 A0 A0  BAM.....
70: A0 A0 A0 A0 A0 00 00 00  ........
78: 00 00 00 00 00 00 06 00  ........
80: 00 00 82 13 03 53 55 50  .....SUP
88: 45 52 44 49 52 A0 A0 A0  ERDIR...
90: A0 A0 A0 A0 A0 00 00 00  ........
98: 00 00 00 00 00 00 08 00  ........
A0: 00 00 82 13 07 4D 55 55  .....MUU
A8: 54 41 20 4E 49 4D 49 20  TA NIMI
B0: 26 20 49 44 A0 00 00 00  & ID....
B8: 00 00 00 00 00 00 04 00  ........
C0: 00 00 82 13 09 54 41 52  .....TAR
C8: 4B 41 53 54 41 20 49 44  KASTA ID
D0: A0 A0 A0 A0 A0 00 00 00  ........
D8: 00 00 00 00 00 00 04 00  ........
E0: 00 00 82 10 00 54 41 52  .....TAR
E8: 4B 41 53 54 41 20 4C 45  KASTA LE
F0: 56 59 4B 45 A0 00 00 00  VYKE....
F8: 00 00 00 00 00 00 03 00  ........
```

Kuva 9.7

Tavut sisältöineen ovat seuraavat:

TAVU	SISÄLTÖ JA MERKITYS	
0	$12 (18),	uraosoitin, sisällysluettelon seuraava ura
1	$04 (04),	sektoriosoitin, sisällysluettelon seuraava sektori
2 - 31		1. tiedoston tiedot
32 - 33	$00	ei käytössä
34 - 63		2. tiedoston tiedot
64 - 65	$00	ei käytössä
66 - 95		3. tiedoston tiedot
96 - 97	$00	ei käytössä
98 - 127		4. tiedoston tiedot
128 - 129	$00	ei käytössä
130 - 159		5. tiedoston tiedot
160 - 161	$00	ei käytössä
162 - 191		6. tiedoston tiedot
192 - 193	$00	ei käytössä
194 - 223		7. tiedoston tiedot
224 - 225	$00	ei käytössä
226 - 255		8. tiedoston tiedot

Yhdelle sektorille mahtuu kahdeksan tiedoston tiedot. Koska 18. uralla on 19 sektoria, joista yhden varaa BAM, voidaan levykkeelle tallentaa korkeintaan 8 * 18 = 144 eri tiedostoa. Ensimmäiset kaksi tavua ilmoittavat, että sisällysluettelo jatkuu 18. uran 4. sektorilla. Koska kutakin tiedostoa varten on 30 tavua, on muidenkin kuin 1. tiedoston edellä kaksi käyttämätöntä tavua. Ennen kuin tutkimme tarkemmin yhden tiedoston tietoja, katsomme miten sisällysluettelo jatkuu. Seuraavaksi siis 18. uran 4. sektori:

```
URA : 18    SEKTORI : 4

00: 12 07 82 10 01 54 41 52  .....TAR
08: 4B 41 53 54 41 20 55 52  KASTA UR
10: 41 A0 A0 A0 A0 00 00 00  A.......
18: 00 00 00 00 00 00 03 00  ........
20: 00 00 82 10 03 54 55 54  .....TUT
28: 4B 49 20 44 52 49 56 45  KI DRIVE
30: A0 A0 A0 A0 A0 00 00 00  ........
38: 00 00 00 00 00 00 03 00  ........
```

Kuva 9.8

Tässä on osa sektorin sisällöstä. Tavuista 0 ja 1 nähdään, että sisällysluettelo jatkuu 18. uran ($12) 7. sektorilla ($07).

```
URA : 18    SEKTORI : 7

00: 00 FF 82 14 07 53 49 53  .....SIS
08: 4C 55 20 31 A0 A0 A0 A0  LU 1....
10: A0 A0 A0 A0 A0 00 00 00  ........
18: 00 00 00 00 00 00 02 00  ........
20: 00 00 82 14 08 53 49 53  .....SIS
28: 4C 55 20 32 A0 A0 A0 A0  LU 2....
30: A0 A0 A0 A0 A0 00 00 00  ........
38: 00 00 00 00 00 00 01 00  ........
40: 00 00 82 14 09 4C 55 4F  .....LUO
48: 20 49 4E 44 45 4B 53 49   INDEKSI
50: 2E 31 A0 A0 A0 00 00 00  .1......
58: 00 00 00 00 00 00 03 00  ........
60: 00 00 82 14 0F 4C 55 4F  .....LUO
68: 20 49 4E 44 45 4B 53 49   INDEKSI
70: 2E 32 A0 A0 A0 00 00 00  .2......
78: 00 00 00 00 00 00 02 00  ........
```

```
80: 00 00 84 0F 00 52 45 4C  .....REL
88: 2E 54 45 53 54 49 A0 A0  .TESTI..
90: A0 A0 A0 A0 A0 0F 0A 32  .......2
98: 00 00 00 00 00 00 0B 00  ........
A0: 00 00 81 0F 07 53 45 51  .....SEQ
A8: 2E 54 45 53 54 49 A0 A0  .TESTI..
B0: A0 A0 A0 A0 A0 00 00 00  ........
B8: 00 00 00 00 00 00 02 00  ........
C0: 00 00 00 00 00 00 00 00  ........
C8: 00 00 00 00 00 00 00 00  ........
D0: 00 00 00 00 00 00 00 00  ........
D8: 00 00 00 00 00 00 00 00  ........
E0: 00 00 00 00 00 00 00 00  ........
E8: 00 00 00 00 00 00 00 00  ........
F0: 00 00 00 00 00 00 00 00  ........
F8: 00 00 00 00 00 00 00 00  ........
```

Kuva 9.9

Huomaat, että sektorit täytetään kolmen välein, ei peräkkäin. Siirtyminen tapahtuu tällä tavalla mahdollisimman nopeasti ja pienellä luku- ja kirjoituspään siirtämisellä.

Sektorin ensimmäisen tavun sisältä on $00 ja toisen $FF. Nämä eivät enää osoita seuraavaa sektoria, vaan ilmoittavat, että tässä on sisällysluettelon viimeinen sektori. Jos ensimmäisen tavun arvoksi sijoitetaan $12 ja toisen arvoksi $01, sisällysluettelo saadaan ikuiseen silmukkaan. Ohjelmat latautuvat kunnolla, mutta sisällysluettelo ei saada näkyviin.

Sisällysluettelo käyttää sektorit seuraavassa järjestyksessä:

0 (BAM), 1, 4, 7, 10, 13, 16, 2, 5, 8, 11, 14, 17, 3, 6, 9, 12, 15, ja 18

Katsotaanpa tarkemmin esimerkiksi sisällysluettelon 1. ohjelman BACKUP-tietoja.

```
URA : 18    SEKTORI : 1

00: 12 04 [82] 11 00 42 41 43  .....BAC
08: 4B 55 50 A0 A0 A0 A0 A0  KUP.....
10: A0 A0 A0 A0 A0 00 00 00  ........
18: 00 00 00 00 00 00 09 00  ........
```

Kuva 9.10

Ensimmäinen tiedoston kannalta tärkeä tavu on kolmas tavu (tavu 2), joka ilmoittaa tiedoston tyypin. Tässä tapauksessa tavun arvo on $82, mikä tarkoittaa, että kysymyksessä on ohjelmatiedosto. Commodoren käyttämät tiedostotyypit on koottu liitteeseèn 3.

```
URA : 18    SEKTORI : 1

00: 12 04 82 11 00 42 41 43   .....BAC
08: 4B 55 50 A0 A0 A0 A0 A0   KUP.....
10: A0 A0 A0 A0 A0 00 00 00   ........
18: 00 00 00 00 00 00 09 00   ........
```

Kuva 9.11

Seuraavat kaksi tavua (tavut 3–4) ilmoittavat tiedoston alun levykkeellä. Tässä tapauksessa tiedoston käyttämä ensimmäinen sektori on 17. uran ($11) sektori 0 ($00).

```
URA : 18    SEKTORI : 1

00: 12 04 82 11 00 42 41 43   .....BAC
08: 4B 55 50 A0 A0 A0 A0 A0   KUP.....
10: A0 A0 A0 A0 A0 00 00 00   ........
18: 00 00 00 00 00 00 09 00   ........
```

Kuva 9.12

Sitten seuraa tiedoston nimi eli tavut 5–20. Nimen pituudelle varataan aina 16 tavua. Jos nimi on lyhyempi, loppuun lisätään merkkejä CHR$(160), SHIFT + välilyönti ($A0).

```
URA : 18    SEKTORI : 1

00: 12 04 82 11 00 42 41 43   .....BAC
08: 4B 55 50 A0 A0 A0 A0 A0   KUP.....
10: A0 A0 A0 A0 A0 00 00 00   ........
18: 00 00 00 00 00 00 09 00   ........
```

Kuva 9.13

Seuraavat kolme tavua käytetään vain kun kysymyksessä on suhteellinen tiedosto. Tavut 21 ($15) ja 22 ($16) ilmoittavat ensimmäisen sivusektorin ja tavu 23 ($17) antaa suhteellisen tiedoston tietueen pituuden. Tavuja 24–27 ($18-$1B) ei käytetä

ja niiden arvona on aina $00.

```
URA : 18    SEKTORI : 1

00: 12 04 82 11 00 42 41 43   .....BAC
08: 4B 55 50 A0 A0 A0 A0 A0   KUP.....
10: A0 A0 A0 A0 A0 00 00 00   ........
18: 00 00 00 00 00 00 09 00   ........
```

Kuva 9.14

DOS käyttää tavuja 28 ja 29 ($1C ja $1D) väliaikaisesti silloin, kun ohjelma tallennetaan tai tiedosto avataan kirjoittamista varten käyttäen korvaustoimintaa ☺. Näihin tavuihin sijoitetaan korvaavan tiedoston ensimmäinen ura ja sektori. Kun korvaustoimitus on ohi, tavujen sisältö nollataan.

```
URA : 18    SEKTORI : 1

00: 12 04 82 11 00 42 41 43   .....BAC
08: 4B 55 50 A0 A0 A0 A0 A0   KUP.....
10: A0 A0 A0 A0 A0 00 00 00   ........
18: 00 00 00 00 00 00 09 00   ........
```

Kuva 9.15

Viimeiset kaksi tavua ilmoittavat tiedoston koon. Tavu 30 ($1E) ilmoittaa alatavun ja 31 ($1F) ylätavun. Tiedoston koko on tässä siis 9 + 256 * 0 = 9 lohkoa.

OHJELMATIEDOSTON RAKENNE

Ohjelmatiedoston käyttämät sektorit voivat olla rakenteeltaan kolmenlaisia. Ensimmäinen sektori on aina seuraavanlainen:

SEURAAVA URA	SEURAAVA SEKTORI	LATAUSOSOITE ALATAVU	LATAUSOSOITE YLÄTAVU	OHJELMAN ENSIMMÄISET 252 TAVUA

Kuva 9.16

Ensimmäinen tavu (tavu 0) ilmoittaa ohjelmatiedoston seuraavan lohkon uran ja toinen tavu (tavu 1) sektorin. Tavut 2 ja 3 ilmoittavat latausosoitteen: tavu 2 alatavun ja tavu 3 ylätavun. BASIC-ohjelmalla näiden tavujen sisällöt ovat $01 ja $08. BASIC-ohjelma alkaa siis muistipaikasta 1 + 256 * 8 = 2049.

Kun ohjelma ladataan käskyllä LOAD "NIMI", 8, näiden tavujen sisältöä ei tarkisteta. Tavut otetaan huomioon vain ladattaessa käskyllä LOAD "NIMI", 8,1. Tavut 4–255 sisältävät varsinaisen ohjelman 252 ensimmäistä tavua.

Seuraavat sektorit viimeistä lukuun ottamatta ovat rakenteeltaan seuraavanlaiset:

SEURAAVA URA	SEURAAVA SEKTORI	254 TAVUA OHJELMAA

Kuva 9.17

Ensimmäinen tavu ilmoittaa seuraavan uran ja toinen tavu ilmoittaa seuraavan sektorin. Loput 254 tavua ovat varsinaista ohjelmaa.

Viimeinen sektori näyttää seuraavalta:

$00	VIIMEISEN TAVUN OSOITTIN	OHJELMAN VIIMEISET TAVUT	ROSKAA

Kuva 9.18

Ensimmäisen tavun arvo on $00. Siitä DOS tietää, että kysymyksessä on viimeinen sektori. Toisen tavun sisältö ilmoittaa viimeisen ohjelmaan kuuluvan tavun. Yleensä viimeinen sektori on vain osittain täytetty. Viimeisen tavun jälkeen tulevat tavut ovat "roskaa".

Tarkastellaanpa kirjan levykkeellä olevaa ohjelmaa TUTKI U&S, joka löytyy sisällysluettelon 18. uran 1. sektorista.

```
URA : 18    SEKTORI : 1

40: 00 00 82 11 07 54 55 54  .....TUT
48: 4B 49 20 55 26 53 A0 A0  KI U&S..
50: A0 A0 A0 A0 A0 00 00 00  ........
58: 00 00 00 00 00 00 05 00  ........
```

Kuva 9.19

Tästä nähdään, että ohjelma alkaa 17. uran ($11) sektorista 7 ($07). Ohjelman pituus on 5 lohkoa. Ura 17 – sektori 7 näyttää seuraavalta:

```
URA : 17   SEKTORI : 7

00: 11 11 01 08 15 08 64 00  ........
08: 8F 20 55 52 41 20 24 20  . URA $
10: 53 45 4B 54 4F 52 49 00  SEKTORI.
18: 37 08 6E 00 97 35 33 32  7....532
20: 38 31 2C 30 3A 97 35 33  81,0:.53
28: 32 38 30 2C 30 3A 97 36  280,0:.6
30: 34 36 2C 37 3A 4C 4E B2  46,7:LN.
38: 33 00 5A 08 78 00 99 22  3.Z....."
40: 93 22 3B A3 31 32 29 22  ."";.12)"
48: 2A 2A 20 55 52 41 20 26  ** URA &
50: 20 53 45 4B 54 4F 52 49   SEKTORI
58: 20 2A 2A 22 00 89 08 82   **"....
60: 00 85 22 11 11 20 12 4B  .."..  .K
68: 92 55 56 41 52 55 55 54  .UVARUUT
70: 55 20 56 41 49 20 12 50  U VAI .P
78: 92 52 49 4E 54 54 45 52  .RINTTER
80: 49 20 20 4B 9D 9D 9D 22  I  K..."
88: 3B 52 24 00 B1 08 8C 00  ;R$.....
90: 8B 20 52 24 B3 B1 22 4B  . R$.."K
98: 22 20 AF 20 52 24 B3 B1  " . R$..
A0: 22 50 22 20 A7 20 99 22  "P" . ."
A8: 91 91 91 22 3B 3A 89 20  ...";:.
B0: 31 33 30 00 C5 08 96 00  130.....
B8: 8B 20 52 24 B2 22 50 22  . R$."P"
C0: 20 A7 20 4C 4E B2 34 00   . LN.4.
C8: F6 08 A0 00 99 22 11 11  .....".
D0: 20 41 53 45 54 41 20 4C   ASETA L
D8: 45 56 59 4B 45 20 41 53  EVYKE AS
E0: 45 4D 41 41 4E 20 4A 41  EMAAN JA
E8: 20 50 41 49 4E 41 20 12   PAINA .
F0: 52 45 54 55 52 4E 92 22  RETURN."
F8: 00 0F 09 AA 00 A1 52 24  ......R$
```

Kuva 9.20

Kahdesta ensimmäisestä tavusta nähdään, että ohjelma jatkuu 17. uran sektorissa 17. Seuraavat kaksi tavua ilmoittavat latausosoitteen $0801. Entä sitten? Tämä on BASIC-ohjelmaa. Seuraavat kaksi tavua ilmoittavat seuraavan ohjelmarivin alun tietokoneen muistissa $0815. Sitten tulee ensimmäisen rivin rivinumero 100 + 0 * 256 = 100. Seuraavan tavun sisältö $8F = 143 on REM-käskysanan koodi, sitten tulee välilyönti $20 = 32.

Ja ohjelma jatkuu.
Ohjelmaa seurataan ura- ja sektoriosoittimien avulla.

```
URA : 17    SEKTORI : 17

00: 11 09 3A 8B 52 24 B3 B1  ..:.R$..
08: C7 28 31 33 29 20 A7 20  .(13) .
10: 31 37 30 00 35 09 B4 00  170.5...
18: 48 24 B2 22 30 31 32 33  H$."0123

URA : 17    SEKTORI : 9

00: 11 13 20 3A 22 53 3A 98  .. :"S:.
08: 34 00 12 0A 0E 01 81 20  4......
10: 4B B2 30 20 A4 20 33 31  K.0 . 31
18: 00 36 0A 18 01 46 B2 31  .6...F.1

URA : 17    SEKTORI : 19

00: 10 07 4C B2 56 41 AB 48  ..L.VA.H
08: 25 AC 31 36 00 2E 0B 5E  %.16...^
10: 01 48 54 24 B2 CA 28 48  .HT$..(H
18: 24 2C 48 25 AA 31 2C 31  $,H%.1,1
```

Kuva 9.21

Seuraava sektori on viimeinen (5 lohkoa).

```
URA : 16    SEKTORI : 7

00: 00 E9 0C C2 01 8B 20 52  ......  R
08: 24 B2 22 4C 22 20 A7 20  $."L" .
10: A0 20 32 3A A0 20 31 35  . 2:. 15
18: 3A 80 00 24 0C CC 01 8B  :..$....
20: 20 52 24 B3 B1 22 53 22   R$.."S"
28: 20 A7 20 34 33 30 00 3C   . 430.<
30: 0C D6 01 55 B2 53 55 3A  ...U.SU:
38: 53 B2 53 53 3A 53 55 B2  S.SS:SU.
40: 30 3A 53 53 B2 30 00 56  0:SS.0.V
48: 0C E0 01 8B 20 55 B1 30  .... U.0
50: 20 AF 20 55 B3 33 36 20   . U.36
58: A7 20 99 3A 89 32 33 30  . .:.230
60: 00 8A 0C EA 01 8B 20 55  ......  U
68: B2 30 20 A7 20 99 3A 99  .0 . .:.
```

```
70: 20 22 45 44 45 4C 4C 49   "EDELLI
78: 4E 45 4E 20 4F 4C 49 20  NEN OLI
80: 56 49 49 4D 45 49 4E 45  VIIMEINE
88: 4E 20 21 22 3A 99 3A 89  N !":.:.
90: 20 32 31 30 00 B6 0C F4   210....
98: 01 99 3A 99 22 20 45 44  ..:." ED
A0: 45 4C 4C 49 4E 45 4E 20  ELLINEN
A8: 45 49 20 4B 55 55 4C 55  EI KUULU
B0: 4E 55 54 20 54 49 45 44  NUT TIED
B8: 4F 53 54 4F 4F 4E 11 22  OSTOON."
C0: 00 DD 0C FE 01 99 22 20  ......"
C8: 56 41 4C 49 54 53 45 3A  VALITSE:
D0: 20 20 12 55 92 55 53 49    .U.USI
D8: 20 20 12 4C 92 4F 50 50    .L.OPP
E0: 55 22 3A 89 34 33 30 00  U":.430.
E8: 00 00 22 20 A7 20 53 55  .." . SU
F0: B2 30 3A 53 53 B2 30 3A  .0:SS.0:
F8: 99 3A 89 32 31 30 00 10  .:.210..
```

Kuva 9.22

Ensimmäisen tavun sisällöstä $00 nähdään, että kysymyksessä on todella viimeinen sivu. Toinen tavu ilmoittaa viimeisen ohjelmatiedostoon kuuluvan tavun. Viimeinen tavu on $E9 = 233. Sen jälkeen tulevat tavut ovat roskaa eivätkä kuulu tähän ohjelmatiedostoon. Huomaa, että BASIC-ohjelman kolmen viimeisen tavun arvo on $00. Sektorin lopussa oleva roska aiheutuu siitä, että DOS ei nollaa puskurin sisältöä, kun tietokone lähettää uutta tietoa. Ohjelman käyttämän viimeisen sektorin lopussa voi sen vuoksi olla edellisen luku- ja kirjoitustapahtuman jälkeen puskuriin jäänyttä tietoa. Sisällysluettelo ja suhteelliset tiedostot muodostavat poikkeuksen. Sisällysluettelon vajaa sektori täytetään aina nollilla $00. Sisällysluettelon viimeinen sektori on aina merkitty täydeksi; sen kaksi ensimmäistä tavua ovat $00 ja $FF.

PERÄKKÄISTIEDOSTON RAKENNE

Peräkkäistiedostojen käyttämien sektoreiden rakenne on viimeistä lukuun ottamatta seuraava:

SEURAAVA URA	SEURAAVA SEKTORI	254 TAVUA TIEDOSTOSTA

Kuva 9.23

0. tavu ilmoittaa seuraavan uran ja 1. tavu vastaavasti uran sektorin. Loput 254 tavua ovat varsinaiset tietotavut.

Viimeisen peräkkäistiedoston käyttämän sektorin tuntee tavusta 0, jonka sisältö on aina $00. Toinen tavu ilmoittaa, mikä sektorin tavuista on tiedoston viimeinen tavu. Tiedoston viimeisen tavun jälkeen sektorilla olevat tavut sisältävät roskaa.

$00	VIIMEISEN TAVUN OSOITIN	TIEDOSTON LOPUT TAVUT	ROSKAA

Kuva 9.24

Kirjan levykkeeltä löytyy yksi peräkkäistiedosto SEQ. TESTI, jonka tiedot ovat sisällysluettelossa 18. uran 7. sektorilla

```
URA : 18    SEKTORI : 7

A0: 00 00 81 0F 07 53 45 51  .....SEQ
A8: 2E 54 45 53 54 49 A0 A0  .TESTI..
B0: A0 A0 A0 A0 A0 00 00 00  ........
B8: 00 00 00 00 00 00 02 00  ........
```

Kuva 9.25

Kolmas tavu ($A2) ilmoittaa tiedoston tyypin, joka on peräkkäistiedostoilla $81 = 149. Seuraavat kaksi tavua ilmoittavat tiedoston ensimmäisen uran ja sektorin, eli 15 ($0F) ja 7 ($07). Sitten saadaan tiedoston nimi sekä SHIFT + välilyöntejä ($A0). Viimeiset kaksi tavua ilmoittavat tiedoston koon, eli 2 + 256 * 0 = 2 lohkoa.

Katsotaanpa sitten varsinaista tiedostoa. Se alkaa siis 15. uran 7. sektorilta.

```
URA : 15    SEKTORI : 7

00: 0F 11 53 45 51 2E 54 49  ..SEQ.TI
08: 45 54 55 45 20 31 0D 53  ETUE 1.S
10: 45 51 2E 54 49 45 54 55  EQ.TIETU
18: 45 20 32 0D 53 45 51 2E  E 2.SEQ.
20: 54 49 45 54 55 45 20 33  TIETUE 3
28: 0D 53 45 51 2E 54 49 45  .SEQ.TIE
30: 54 55 45 20 34 0D 53 45  TUE 4.SE
38: 51 2E 54 49 45 54 55 45  Q.TIETUE
40: 20 35 0D 53 45 51 2E 54   5.SEQ.T
```

```
48: 49 45 54 55 45 20 36 0D  IETUE 6.
50: 53 45 51 2E 54 49 45 54  SEQ.TIET
58: 55 45 20 37 0D 53 45 51  UE 7.SEQ
60: 2E 54 49 45 54 55 45 20  .TIETUE
68: 38 0D 53 45 51 2E 54 49  8.SEQ.TI
70: 45 54 55 45 20 39 0D 53  ETUE 9.S
78: 45 51 2E 54 49 45 54 55  EQ.TIETU
80: 45 20 31 30 0D 53 45 51  E 10.SEQ
88: 2E 54 49 45 54 55 45 20  .TIETUE
90: 31 31 0D 53 45 51 2E 54  11.SEQ.T
98: 49 45 54 55 45 20 31 32  IETUE 12
A0: 0D 53 45 51 2E 54 49 45  .SEQ.TIE
A8: 54 55 45 20 31 33 0D 53  TUE 13.S
B0: 45 51 2E 54 49 45 54 55  EQ.TIETU
B8: 45 20 31 34 0D 53 45 51  E 14.SEQ
C0: 2E 54 49 45 54 55 45 20  .TIETUE
C8: 31 35 0D 53 45 51 2E 54  15.SEQ.T
D0: 49 45 54 55 45 20 31 36  IETUE 16
D8: 0D 53 45 51 2E 54 49 45  .SEQ.TIE
E0: 54 55 45 20 31 37 0D 53  TUE 17.S
E8: 45 51 2E 54 49 45 54 55  EQ.TIETU
F0: 45 20 31 38 0D 53 45 51  E 18.SEQ
F8: 2E 54 49 45 54 55 45 20  .TIETUE
```

Kuva 9.26

Ensimmäiset kaksi tavua ilmoittavat seuraavan tiedostolohkon: 15. ura 17. sektori. Lopputavut ovat varsinaista tiedostoa. Siirrytään seuraavalle sektorille.

```
URA : 15    SEKTORI : 17

00: 00 9E 31 39 0D 53 45 51  ..19.SEQ
08: 2E 54 49 45 54 55 45 20  .TIETUE
10: 32 30 0D 53 45 51 2E 54  20.SEQ.T
18: 49 45 54 55 45 20 32 31  IETUE 21
20: 0D 53 45 51 2E 54 49 45  .SEQ.TIE
28: 54 55 45 20 32 32 0D 53  TUE 22.S
30: 45 51 2E 54 49 45 54 55  EQ.TIETU
38: 45 20 32 33 0D 53 45 51  E 23.SEQ
40: 2E 54 49 45 54 55 45 20  .TIETUE
48: 32 34 0D 53 45 51 2E 54  24.SEQ.T
50: 49 45 54 55 45 20 32 35  IETUE 25
58: 0D 53 45 51 2E 54 49 45  .SEQ.TIE
```

```
60: 54 55 45 20 32 36 0D 53   TUE 26.S
68: 45 51 2E 54 49 45 54 55   EQ.TIETU
70: 45 20 32 37 0D 53 45 51   E 27.SEQ
78: 2E 54 49 45 54 55 45 20   .TIETUE
80: 32 38 0D 53 45 51 2E 54   28.SEQ.T
88: 49 45 54 55 45 20 32 39   IETUE 29
90: 0D 53 45 51 2E 54 49 45   .SEQ.TIE
98: 54 55 45 20 33 30 0D 00   TUE 30..
A0: 00 00 01 0F 07 53 45 51   .....SEQ
A8: 2E 54 45 53 54 49 A0 A0   .TESTI..
B0: A0 A0 A0 A0 A0 00 00 00   ........
B8: 00 00 00 00 00 00 00 00   ........
C0: 00 00 00 00 00 00 00 00   ........
C8: 00 00 00 00 00 00 00 00   ........
D0: 00 00 00 00 00 00 00 00   ........
D8: 00 00 00 00 00 00 00 00   ........
E0: 00 00 00 00 00 00 00 00   ........
E8: 00 00 00 00 00 00 00 00   ........
F0: 00 00 00 00 00 00 00 00   ........
F8: 00 00 00 00 00 00 00 00   ........
```

Kuva 9.27

Ensimmäisen tavun sisältö on $00, joten se on viimeinen sektori. Toinen tavu ilmoittaa viimeisen tiedostoon kuuluvan tavun: $9E. Sektorin loppu $9F–$FF on roskaa, joskin se on nyt selvästi luettavissa. Usein peräkkäistiedoston viimeinen sektori täydennetään levykkeen sisällysluetteloon kuuluvilla tavuilla.

SUHTEELLISTEN TIEDOSTOJEN RAKENNE

Levykkeellä olevat suhteellisen tiedoston sektorit ovat rakenteeltaan kahdenlaisia. Varsinaisen tiedoston käyttämät sektorit ovat samanlaisia kuin peräkkäistiedostoilla. Niiden lisäksi suhteelliset tiedostot tarvitsevat ns. sivusektoreita.

Sivusektoreita on aina vähintään yksi ja korkeintaan kuusi. Kukin sivusektori sisältää kaksi erilaista ura-sektoriosoittimista muodostunutta joukkoa. Toisen muodostavat sivusektoreiden urat ja sektorit, toinen muodostuu tietolohkojen urista ja sektoreista. Jokainen sivusektori on rakenteeltaan seuraavanlainen:

SEURAAVA URA	SEURAAVA SEKTORI	SIVUSEKTORIN NUMERO	TIETUEEN PITUUS	6 SIVUSEKTORIN URA/SEKTORI-OSOITTIMET	120 TIETOLOHKON URA/SEKTORI-OSOITTIMET

Kuva 9.28

TAVU	MERKITYS
0	seuraavan sivusektorin ura
1	seuraavan sivusektorin sektori
2	sivusektorin järjestysnumero
3	tietueen pituus
4–15	sivusektoreiden urat ja sektorit
	4–5 1. sivusektorin ura ja sektori
	6–7 2. sivusektorin ura ja sektori
	8–9 3. sivusektorin ura ja sektori
	10–11 4. sivusektorin ura ja sektori
	12–13 5. sivusektorin ura ja sektori
	14–15 6. sivusektorin ura ja sektori
16–225	120 tietolohkon urat ja sektorit
	16–17 1. tietolohkon ura ja sektori
	18–19 2. tietolohkon ura ja sektori
	...
	...
	254–255 120. tietolohkon ura ja sektori

Ensimmäisellä sivusektorilla on siis tietolohkojen 1–120 urat ja sektorit, toisella sivusektorilla tietolohkojen 121–240 urat ja sektorit jne. Sivusektoreiden urille ja sektoreille varatut tavut ovat kaikissa sivusektoreissa samat.

Seuraavaksi katsotaan levykkeellä olevan suhteellisen tiedoston sisällysluettelotietoja, jotka löytyvät 18. uran 7. sektorilta.

```
URA : 18    SEKTORI : 7

80: 00 00 84 0F 00 52 45 4C  .....REL
88: 2E 54 45 53 54 49 A0 A0  .TESTI..
90: A0 A0 A0 A0 A0 0F 0A 32  .......2
98: 00 00 00 00 00 00 0B 00  ........
```

Kuva 9.29

Kolmas tavu ($82) ilmoittaa tiedoston tyypin. Koska kysymyksessä on suhteellinen tiedosto, se on $84 = 132. Seuraavat kaksi tavua ilmoittavat varsinaisen tiedoston ensimmäisen tietolohkon uran ja sektorin. Seuraavat kuusitoista tavua on varattu tiedoston nimelle. Seuraavat kaksi tavua (tavut $95 ja $96) ilmoittavat ensimmäisen sivusektorin. Se on nyt uralla 15 ($0F) sektorilla ($0A). Seuraava tavu ilmoittaa tietueen pituuden $32 = 50. Kuuden seuraavan tavun arvo on $00. Seuraavat kaksi tavua ilmoittavat tiedoston pituuden lohkoina: 11 ($0B) lohkoa. Mukaan on laskettu myös sivusektorit.

Näiden tietojen avulla voidaan laskea tiedostossa olevien tietueiden lukumäärä. Sivusektoreiden lukumäärä saadaan jakamalla lohkojen määrä 11 120:lla ja pyöristämällä osamäärä ylöspäin lähimpään kokonaislukuun:

11/120 = 0.092

Sivusektoreita on siis 1 ja varsinaisia tietolohkoja 11 – 1 = 10. Sitten lasketaan tavujen lukumäärä. Kussakin lohkossa on 254 varsinaista tietotavua, joten niitä on yhteensä 10 * 254 = 2540. Jakamalla tavujen lukumäärä tietueen pituudella saadaan tietueiden lukumäärä:

2540/50 = 50.8

Tietueita on siis 50.
Katsotaan ensimmäistä sivusektoria (15. ura 10. sektori).

```
URA : 15    SEKTORI : 10

00: 00 23 00 32 0F 0A 00 00  .#.2....
08: 00 00 00 00 00 00 00 00  ........
10: 0F 00 0F 0B 0F 01 0F 0C  ........
18: 0F 02 0F 0D 0F 03 0F 0E  ........
20: 0F 05 0F 0F 00 00 00 00  ........
28: 00 00 00 00 00 00 00 00  ........
30: 00 00 00 00 00 00 00 00  ........
38: 00 00 00 00 00 00 00 00  ........
40: 00 00 00 00 00 00 00 00  ........
48: 00 00 00 00 00 00 00 00  ........
50: 00 00 00 00 00 00 00 00  ........
58: 00 00 00 00 00 00 00 00  ........
60: 00 00 00 00 00 00 00 00  ........
68: 00 00 00 00 00 00 00 00  ........
```

```
70: 00 00 00 00 00 00 00 00  ........
78: 00 00 00 00 00 00 00 00  ........
80: 00 00 00 00 00 00 00 00  ........
88: 00 00 00 00 00 00 00 00  ........
90: 00 00 00 00 00 00 00 00  ........
98: 00 00 00 00 00 00 00 00  ........
A0: 00 00 00 00 00 00 00 00  ........
A8: 00 00 00 00 00 00 00 00  ........
B0: 00 00 00 00 00 00 00 00  ........
B8: 00 00 00 00 00 00 00 00  ........
C0: 00 00 00 00 00 00 00 00  ........
C8: 00 00 00 00 00 00 00 00  ........
D0: 00 00 00 00 00 00 00 00  ........
D8: 00 00 00 00 00 00 00 00  ........
E0: 00 00 00 00 00 00 00 00  ........
E8: 00 00 00 00 00 00 00 00  ........
F0: 00 00 00 00 00 00 00 00  ........
F8: 00 00 00 00 00 00 00 00  ........
```

Kuva 9.30

Tässä tiedostossa ensimmäinen sivusektori on samalla viimeinen, koska ensimmäisen tavun arvo on $00. Toinen tavu ilmoittaa viimeisen tähän tietosektoriin kuuluvan tavun. Koska kunkin sivusektorin tavut 16–255 ilmoittavat tietolohkojen urat ja sektorit, pitää viimeisen tavun $23 lisäksi katsoa tavua $22. Nämä tavut ($22 ja $23) ilmoittavat varsinaisen tiedoston viimeisen tietolohkon uran ja sektorin. Sektorin loppuosa on täytetty arvoilla $00 (ei siis roskia). Sektorin kolmannen tavun arvo on $00. Se ilmoittaa sivusektorin järjestysnumeron. Sivusektorit numeroidaan 0–5. Neljäs tavu ilmoittaa tietueen pituuden. Seuraavat kuusi tavuparia ilmoittavat sivusektoreiden urat ja sektorit. Ensimmäinen tavupari osoittaa tutkimamme sivusektorin. Koska se on samalla ainoa, seuraavien viiden tavuparin tavujen arvot ovat $00. Sitten seuraavat tietolohkojen urat ja sektorit, kaikkiaan 10 paria. Tämän suhteellisen tiedoston tietolohkot ovat seuraavilla urilla/sektoreilla: 15/0, 15/11, 15/1, 15/12, 15/2, 15/13, 15/3, 15/14, 15/4 ja 15/15.

Katsotaan varsinaisen tiedoston ensimmäistä tietolohkoa.

```
URA : 15    SEKTORI : 0

00: 0F 0B 54 49 45 54 55 45  ..TIETUE
08: 20 31 0D 00 00 00 00 00   1......
10: 00 00 00 00 00 00 00 00  ........
18: 00 00 00 00 00 00 00 00  ........
20: 00 00 00 00 00 00 00 00  ........
28: 00 00 00 00 00 00 00 00  ........
30: 00 00 00 00 54 49 45 54  ....TIET
38: 55 45 20 32 0D 00 00 00  UE 2....
40: 00 00 00 00 00 00 00 00  ........
48: 00 00 00 00 00 00 00 00  ........
50: 00 00 00 00 00 00 00 00  ........
58: 00 00 00 00 00 00 00 00  ........
60: 00 00 00 00 00 00 54 49  ......TI
68: 45 54 55 45 20 33 0D 00  ETUE 3..
70: 00 00 00 00 00 00 00 00  ........
78: 00 00 00 00 00 00 00 00  ........
80: 00 00 00 00 00 00 00 00  ........
88: 00 00 00 00 00 00 00 00  ........
90: 00 00 00 00 00 00 00 00  ........
98: 54 49 45 54 55 45 20 34  TIETUE 4
A0: 0D 00 00 00 00 00 00 00  ........
A8: 00 00 00 00 00 00 00 00  ........
B0: 00 00 00 00 00 00 00 00  ........
B8: 00 00 00 00 00 00 00 00  ........
C0: 00 00 00 00 00 00 00 00  ........
C8: 00 00 54 49 45 54 55 45  ..TIETUE
D0: 20 35 0D 00 00 00 00 00   5......
D8: 00 00 00 00 00 00 00 00  ........
E0: 00 00 00 00 00 00 00 00  ........
E8: 00 00 00 00 00 00 00 00  ........
F0: 00 00 00 00 00 00 00 00  ........
F8: 00 00 00 00 54 49 45 54  ....TIET
```

Kuva 9.31

Ensimmäinen tavupari on tuttu. Tavut ilmoittavat seuraavan uran ja sektorin, 15/11. Sitten tulee ensimmäinen tietue. Tietueen loppu on täytetty nollilla, joten tietueen pituudeksi saadaan vakiona 50. Sektorin loppuosa on muodostunut seuraavista tietueista. Katsotaan seuraavaa tietolohkoa.

```
URA : 15   SEKTORI : 11

00: 0F 01 55 45 20 36 0D 00  ..UE 6..
08: 00 00 00 00 00 00 00 00  ........
10: 00 00 00 00 00 00 00 00  ........
18: 00 00 00 00 00 00 00 00  ........
20: 00 00 00 00 00 00 00 00  ........
28: 00 00 00 00 00 00 00 00  ........
30: 54 49 45 54 55 45 20 37  TIETUE 7
38: 0D 00 00 00 00 00 00 00  ........
40: 00 00 00 00 00 00 00 00  ........
48: 00 00 00 00 00 00 00 00  ........
50: 00 00 00 00 00 00 00 00  ........
58: 00 00 00 00 00 00 00 00  ........
60: 00 00 54 49 45 54 55 45  ..TIETUE
68: 20 38 0D 00 00 00 00 00   8......
70: 00 00 00 00 00 00 00 00  ........
78: 00 00 00 00 00 00 00 00  ........
80: 00 00 00 00 00 00 00 00  ........
88: 00 00 00 00 00 00 00 00  ........
90: 00 00 00 00 54 49 45 54  ....TIET
98: 55 45 20 39 0D 00 00 00  UE 9....
A0: 00 00 00 00 00 00 00 00  ........
A8: 00 00 00 00 00 00 00 00  ........
B0: 00 00 00 00 00 00 00 00  ........
B8: 00 00 00 00 00 00 00 00  ........
C0: 00 00 00 00 00 00 54 49  ......TI
C8: 45 54 55 45 20 31 30 0D  ETUE 10.
D0: 00 00 00 00 00 00 00 00  ........
D8: 00 00 00 00 00 00 00 00  ........
E0: 00 00 00 00 00 00 00 00  ........
E8: 00 00 00 00 00 00 00 00  ........
F0: 00 00 00 00 00 00 00 00  ........
F8: FF 00 00 00 00 00 00 00  ........
```

Kuva 9.32

Seuraava tietolohko on 15. uran 1. sektorilla, kuten kahdesta ensimmäisestä tavusta voidaan todeta. Sitten tulevat tietueet. Tiedostoon on kirjoitettu vasta 10 tietuetta, mutta kaikki muutkin on varattu, sillä tiedoston avaamisen jälkeen viimeinen tietue kirjoitettiin sijoittamalla esimerkiksi CHR$(255) eli $FF. Näin koko tiedosto tulee kerralla varatuksi ja samalla se saadaan pakattua paremmin. Yhdennentoista tietueen ensim-

mäinen tavu on $FF ja muut ovat nollia. Kaikki loput tietueet ovat samanlaiset. Seuraavan tietolohkon 15/1 alkuosa näyttää seuraavalta:

```
URA : 15   SEKTORI : 1

00: 0F 0C 00 00 00 00 00 00  ........
08: 00 00 00 00 00 00 00 00  ........
10: 00 00 00 00 00 00 00 00  ........
18: 00 00 00 00 00 00 00 00  ........
20: 00 00 00 00 00 00 00 00  ........
28: 00 00 00 00 FF 00 00 00  ........
30: 00 00 00 00 00 00 00 00  ........
38: 00 00 00 00 00 00 00 00  ........
40: 00 00 00 00 00 00 00 00  ........
48: 00 00 00 00 00 00 00 00  ........
50: 00 00 00 00 00 00 00 00  ........
58: 00 00 00 00 00 00 FF 00  ........
60: 00 00 00 00 00 00 00 00  ........
68: 00 00 00 00 00 00 00 00  ........
70: 00 00 00 00 00 00 00 00  ........
78: 00 00 00 00 00 00 00 00  ........
```

Kuva 9.33

Siirrytään viimeiseen tietolohkoon 15/15.

```
URA : 15   SEKTORI : 15

00: 00 D7 00 00 00 00 00 00  ........
08: 00 00 00 00 00 00 00 00  ........
10: FF 00 00 00 00 00 00 00  ........
18: 00 00 00 00 00 00 00 00  ........
20: 00 00 00 00 00 00 00 00  ........
28: 00 00 00 00 00 00 00 00  ........
30: 00 00 00 00 00 00 00 00  ........
38: 00 00 00 00 00 00 00 00  ........
40: 00 00 FF 00 00 00 00 00  ........
48: 00 00 00 00 00 00 00 00  ........
50: 00 00 00 00 00 00 00 00  ........
58: 00 00 00 00 00 00 00 00  ........
60: 00 00 00 00 00 00 00 00  ........
68: 00 00 00 00 00 00 00 00  ........
70: 00 00 00 00 FF 00 00 00  ........
```

```
78: 00 00 00 00 00 00 00 00  ........
80: 00 00 00 00 00 00 00 00  ........
88: 00 00 00 00 00 00 00 00  ........
90: 00 00 00 00 00 00 00 00  ........
98: 00 00 00 00 00 00 00 00  ........
A0: 00 00 00 00 00 00 FF 00  ........
A8: 00 00 00 00 00 00 00 00  ........
B0: 00 00 00 00 00 00 00 00  ........
B8: 00 00 00 00 00 00 00 00  ........
C0: 00 00 00 00 00 00 00 00  ........
C8: 00 00 00 00 00 00 00 00  ........
D0: 00 00 00 00 00 00 00 00  ........
D8: FF 00 00 00 00 00 00 00  ........
E0: 00 00 00 00 00 00 00 00  ........
E8: 00 00 00 00 00 00 00 00  ........
F0: 00 00 00 00 00 00 00 00  ........
F8: 00 00 00 00 00 00 00 00  ........
```

Kuva 9.34

Tämä on viimeinen tietolohko, koska ensimmäisen tavun arvo on $00. Toinen tavu ilmoittaa viimeisen tähän tiedostoon kuuluvan tietueen viimeisen tavun: $D7. Kuten huomaat, DOS on askeleen edellä, sillä seuraavakin tietue (51.) on jo merkitty. Tavun $D8 arvo on $FF.

MUIDEN TIEDOSTOTYYPPIEN RAKENNE

Ohjelmoija määrää käyttäjän tiedoston USER ($83) muodon. Se voi olla rakenteeltaan joko ohjelma- tai peräkkäistiedoston kaltainen, jos se on luotu DOS:in avulla. Jos on käytetty suorasaantikäskyjä (&-tiedostot), se on rakenteeltaan edellisistä poikkeava.

DEL-tiedoston tyyppinä sisällysluettelossa on $80. DEL on lyhenne sanasta deleted (poistettu), mutta kysymyksessä ei ole tuhottu (scratched) tiedosto, vaan Commodoren dokumentoimaton tiedostotyyppi. Rakenteeltaan se voi olla ohjelma- tai peräkkäistiedoston kaltainen. DEL-tiedostoa (tunnus $80) ei kuitenkaan luoda OPEN-käskyllä. Se voidaan luoda vain muuttamalla jonkin olemassa olevan tiedoston tyypin tunnusta sisällysluettelossa. Jos DEL-tiedosto on rakenteeltaan ohjelmatiedoston kaltainen, se voidaan ladata käskyillä LOAD "NIMI, DEL, R",8 tai LOAD "NIMI, DEL, R" 8,1. Jos se on rakenteeltaan peräkkäistiedoston kaltainen, se voidaan avata

lukemista varten käskyllä OPEN 5,8,5, "NIMI, DEL, R".

Lukitut tiedostot (sisällysluettelossa esim. PRG< tai REL<) voidaan muodostaa vain muuttamalla tiedoston tunnusta sisällysluettelossa (18. ura). Kirjan levykkeellä olevalla ohjelmalla voit lukita (ja avata) haluamasi tiedoston. Lukittua tiedostoa ei voida tuhota, mutta se voidaan kopioida tai nimetä uudelleen.

LIITE 1

1541-MUISTIKARTTA

1541-levyasema sisältää yhden 6502-mikrosuorittimen, kaksi 6522-piiriä, jotka huolehtivat I/O-toiminnoista, 2 K:n RAM-muistin sekä 15,8 K:n levynkäyttöjärjestelmän (DOS), joka sijaitsee ROM-muistissa.

6502-mikrosuoritin toimii kahdessa tilassa: levyohjaintilassa (Floppy Disk Controller) ja tiedostonhallintatilassa (File Manager).

Seuraavassa on ensin esitetty muistikartta yleisesti ja sitten RAM-muistin tärkeimmät muistipaikat.

$0000 – $0011	SIVU 0:	työjono, osoittimet
$0012 – $00FF		vakiot, työalue
$0100 – $01B9	SIVU 1:	pinot
$01BA – $01FF		työalue, ylivuotopuskuri
$0200 – $0229	SIVU 2:	komentopuskuri
$022A – $02FF		6502:n työalue
$0300 – $03FF	SIVU 3:	tietopuskuri 0 väliaikaisten tietojen tallennus
$0400 – $04FF	SIVU 4:	tietopuskuri 1 väliaikaisten tietojen tallennus
$0500 – $05FF	SIVU 5:	tietopuskuri 2 väliaikaisten tietojen tallennus

$0600 – $06FF	SIVU 6:	tietopuskuri 3 väliaikaisten tietojen tallennus
$0700 – $07FF	SIVU 7:	tietopuskuri 4 varattu BAM:ille
$1800 – $180F	6522-piiri: I/O	
$1C00 – $1C0F	6522-piiri: levyohjain	
$C100 – $F258	ROM: tiedostonhallintarutiinit	
$F259 – $FFFF	ROM: levyohjainrutiinit	

Seuraavaan on koottu tärkeimmät RAM-muistin osoitteet, kuvaus muistipaikan sisällöstä sekä muistipaikan sisältö kytkettäessä levyasemaan virta, jos sisältö on erisuuri kuin $00. Sisältö on merkitty sulkeisiin kuvauksen jälkeen.

OSOITE	SISÄLTÖ
$0000	Työjono: työ 0, käytössä puskuri 0
$0001	työ 1, käytössä puskuri 1
$0002	työ 2, käytössä puskuri 2
$0003	työ 3, käytössä puskuri 3
$0004	työ 4, käytössä puskuri 4
$0005	työ 5, käytössä puskuri 5
	Otsikkotaulu: $0006 – $0011
$0006 – 07	ura/sektori työlle 0
$0008 – 09	ura/sektori työlle 1
$000A – 0B	ura/sektori työlle 2
$000C – 0D	ura/sektori työlle 3
$000E – 0F	ura/sektori työlle 4
$0010 – 11	ura/sektori työlle 5
$0012 – 13	levykkeen ID, joka annetaan levykettä alustettaessa, $12 = 1. ID-merkki, $13 = 2. ID-merkki
$0014 – 15	ei käytössä; levykkeen ID asemalle 1
$0016 – 1A	OTSIKKOLOHKO $16 = 1. ID-merkki $17 = 2. ID-merkki $18 = ura $19 = sektori $1A = tarkistussumma

$001B	ei käytössä (levyohjaimen työ)
$001C	lippu: muutos kirjoituksenestossa (01)
$001D	ei käytössä (lippu asemalle 1) (01)
$001E	kirjoituksenestokytkimen tila (01)
$001F	ei käytössä (tila asemalle 1)
$0020	levyaseman 0 tila bitti 4: moottorin sammutus (1=K), (0=E) bitti 5: moottori käynnissä (1=K, 0=E) bitti 6: luku- ja kirjoituspää liikkuu (1=K, 0=E) bitti 7: levyasema valmis? (1=E, 0=K)
$0021	ei käytössä (levyaseman 1 tila)
$0022	luku- ja kirjoituspään alla oleva ura
$0023	ei käytössä (ura asemalle 1)
$0024 – 2D	tallennustaulukko muunnoksille BIN – > GCR
$002E – 2F	osoittimien väliaikainen tallennus
$0030 – 31	osoitin: aktiivinen puskuri
$0032 – 33	osoitin: otsikkotaulun aktiiviset arvot ($32 = ura, $33 = sektori)
$0034	osoitin: viimeinen muunnettu merkki (GCR)
$0035	ei käytössä (dekoodausvirhe, GCR)
$0036	tavulaskuri muunnokselle GCR/BIN
$0037	ei käytössä (bittilaskuri)
$0038	tietolohkon tunnusmerkki ($07)
$0039	otsikkolohkon tunnusmerkki ($08)
$003A	tarkistussumman tallennus (tieto- tai otsikkolohko)
$003B	ei suoraan käytössä
$003C	ei suoraan käytössä
$003D	aseman numero, 1541:lle aina $00
$003E	aktiivinen asema ($FF, jos ei aktiivinen)
$003F	edellisen työn paikka jonossa (0 – 5)
$0040	sektori – sisäinen tallennus (tavulaskuri muunnokselle GCR/BIN)
$0041	seuraavan työn paikka jonossa (0 – 5)
$0042	seuraava ura, jolle luku- ja kirjoituspää siirretään
$0043	sektorilaskuri, jota käytetään alustuksen yhteydessä
$0044	väliaikainen työalue
$0045	työn koodin väliaikainen tallennus
$0046	ei käytössä

$0047	tietolohkon ID-merkki, alussa arvo on $07. Arvoa muuttamalla tietolohko voidaan kirjoittaa eri ID-merkillä. Arvon pitää kuitenkin olla pienempi kuin $10.
$0048	ajastin: luku- ja kirjoituspään liikkeen kiihtyvyys
$0049	pino-osoittimen väliaikainen tallennus ($39)
$004A	seuraavalle uralle siirtymiseen tarvittavien askelten lukumäärä. Arvot 0 – 127 siirtävät ulkoreunaa kohti. Arvot, jotka ovat yli 128, siirtävät keskustaa kohti.
$004B	väliaikainen tallennus
$004C	viimeksi luettu sektori
$004D	seuraava sektori
$004E	osoitin (ylätavu): seuraava puskuri, jossa on BIN-muotoon muutettavia GCR-tavuja.
$004F	osoitin (alatavu): seuraavaksi muunnettavan tavun sijainti puskurissa.
$0050	lippu: muoto, johon aktiivisessa puskurissa oleva tieto on jäänyt (0=BIN, 1=GCR)
$0051	alustettavana olevan uran numero ($FF)
$0052 – 55	binaaritaulukko: 4 binaarimuodossa olevaa tavua, jotka muunnetaan tai on muunnettu GCR-muotoon
$0056 – 5D	GCR-taulukko: viisi GCR-muodossa olevaa tavua, jotka muunnetaan tai on muunnettu binaarimuotoon
$005E	luku- ja kirjoituspään kiihdyttämiseen/hidastamiseen käytettävien askelten lukumäärä ($04)
$005F	kiihdytyskerroin/hidastuskerroin ($04)
$0060	kiihdyttämiseen/hidastamiseen tarvittavien askelten jäljellä oleva määrä
$0061	jäljellä olevien askelten lukumäärä ennen kuin luku- ja kirjoituspään nopeus on suurimmillaan
$0062 – 63	osoitin: luku- ja kirjoituspään siirtorutiini ($FA05)
$0064	luku- ja kirjoituspään pienin tarvittava askelmäärä ($C8)
$0065 – 66	osoitin: NMI-rutiinin alku ($EB2E)
$0067	lippu: NMI käynnissä tai ei
$0068	lippu: levykkeen automaattinen initialisointi havaittaessa ID MISMATCH (1 = E, 0 = K)

$0069	sektorin numeron lisäys käytettäessä SEQ-tiedostoja, alkuarvo 10 ($0A)
$006A	lukemisyritysten lukumäärä havaittaessa virhe ($05)
$006B – 6C	osoitin: hyppytaulukon alku (FFE6)
$006D – 6E	osoitin: bittikartan alku ($0400)
$006F	väliaikainen työalue ($6F)
$0070	väliaikainen työalue
$0071	väliaikainen työalue
$0072	väliaikainen työalue ($FF)
$0073	väliaikainen työalue
$0074	väliaikainen työalue
$0075 – 76	epäsuora osoitinmuuttuja ($0100)
$0077	vastaanottajan osoite (laitenumero + $20) ($28)
$0078	lähettäjän osoite (laitenumero + $40) ($48)
$0079	lippu: aktiivinen vastaanottaja
$007A	lippu: aktiivinen lähettäjä
$007C	lippu: ATN-keskeytys
$007D	lippu: 6502 ATN-tilassa
$007E	viimeinen käsitelty ohjelma
$007F	levyaseman numero (1541:llä aina $00)
$0080	käsiteltävä ura
$0081	käsiteltävä sektori
$0082	looginen indeksi (kanavanumero)
$0083	toisio-osoite
$0084	alkuperäinen toisio-osioite ($6F)
$0085	käsiteltävä tietotavu ($3F)
$0086 – 8A	väliaikaiset tulokset
$008B – 8E	tulos, kerto- ja jakolaskurutiinit
$008F – 93	jäännös, kerto- ja jakolaskurutiinit
$0094 – 95	osoitin: sisällysluettelopuskuri ($0204)
$0099 – 9A	osoitin: seuraava tavu puskurissa 0 ($0300)
$009B – 9C	osoitin: seuraava tavu puskurissa 1 ($0400)
$009D – 9E	osoitin: seuraava tavu puskurissa 2 ($0500)
$009F – A0	osoitin: seuraava tavu puskurissa 3 ($0600)
$00A1 – A2	osoitin: seuraava tavu puskurissa 4 ($0700)
$00A3 – A4	osoitin: seuraava tavu komentopuskurissa ($0200)
$00A5 – A6	osoitin: seuraava tavu virhepuskurissa ($02D6)
$00A7 – AD	taulukko: puskureihin liitetyt ei-aktiiviset kanavanumerot

$00BB – C0	taulukko: tietuenumeroiden ylätavut
$00C1 – C6	taulukko: seuraavat tietuenumerot
$00C7 – CC	taulukko: tietueen koko
$00CD – D2	taulukko: sivusektorit
$00D4	osoitin: tietueen alku
$00D5	sivusektorin numero
$00D6	indeksi: sivusektorin tietolohko
$00D7	osoitin: tietueessa
$00D8 – DC	sisällysluettelon sektorit
$00DD – E1	indeksi: sisälllysluettelo
$00E2 – E6	oletusarvo, asemanumero ($00)
$00E7 – EB	tiedoston tyyppi (jokerit, @, suljettu
$00EC – F1	tiedoston tyyppi (kanava)
$00F2 – F7	kanavan tila
$00F9	tehtävä työ (puskurin numero)
$00FA – FE	viimeksi käytetty taulukko
$00FF – 100	lippu: ei asemaa (asemat 0 ja 1)
$0101 – 02	DOS-versio (uralta 18 sektorilta 0)
$0103	ei käytössä
$0104 – FF	pinoalue
$0200 – 29	komentopuskuri: tietokoneelta lähetettyjen komentojen tallennuspaikka
$022A	komennon koodi
$022B – 2D	taulukko: looginen indeksi (LINDX) Taulukko ilmoittaa kunkin kanavan (toisio-osoitteen) tilan, $22B = kanava 0, $22C = kanava 1 jne. Mahdolliset tila-arvot ovat: $FF = ei aktiivinen $81 = avattu kirjoittamista varten $41 = luku/kirjoitus $01 = avattu lukemista varten
$022E – 43	viimeksi luettu/kirjoitettu tavu kullekin kanavalle
$0244 – 49	osoitin: kanavan viimeinen merkki osoittaa viimeksi luetun/kirjoitetun merkin kanavan puskurissa
$024A	aktiivisen tiedoston tyyppi
$024B	merkkijonon pituus
$024C	väliaikainen kanavanumero (toisio-osoite)
$024E	työalue parhaan sektorin löytämiseksi
$024F – 50	puskurin varaus
$0251 – 52	lippu: BAM sekaisin (asemat 0 ja 1)
$0253	lippu: tiedosto löytynyt sisällysluettelosta
$0254	lippu: sisällysluettelon tulostus

$0255	lippu: odottaa komentoa
$0257	viimeksi käytetty puskuri
$0258	tietueen pituus
$0259	sivusektorin ura
$025A	sivusektorin sektori
$025B – 5F	viimeinen työ (puskurit)
$0260 – 65	sisällysluettelon sektori (puskurit)
$0266 – 6B	tiedoston indeksi sisällysluettelossa (puskurit)
$026C	virheilmoitus
$026D	virhevalo
$026E	viimeisen ohjelman asema
$026F	viimeisen ohjelman sektori
$0270	kirjoita LINDX
$0271	lue LINDX
$0272 – 73	lohkojen lukumäärä (väliaikainen)
$0274	komentojonon pituus
$0275	tulkittava merkki
$027A – 7F	osoitintaulukko: tiedostojen nimet
$0280 – 84	ensimmäinen tiedosto-osoitin (ura)
$0285 – 89	toinen tiedosto-osoitin (sektori)
$028A	lippu: jokereita
$028C	etsittyjen asemien lukumäärä
$028D	lippu: aseman etsiminen
$028E	asema, jossa viimeksi kirjoitus/avausvirhe: käytetään oletusasemana
$028F	lippu; löytynyt sisällysluettelosta
$0290	sisällysluettelon sektori
$0291	ensimmäisen saatavan tiedoston sektori
$0292	ensimmäisen saatavan tiedoston indeksi
$0293	viimeisellä lohkolla 0
$0294	käsiteltävä puskuri-indeksi
$0295	laskuri: tiedostot
$0296	lippu: jokereihin sopiva nimi
$0297	aktiivinen toimintatila (R tai W)
$0298	lippu: työn palautus
$0299	osoitin: virheen lukeminen uudestaan
$029A	urien kokonaismäärä
$029B – 9C	osoitin: BAM:in viimeinen päivitys
$029D – A0	BAM:in kuvan sektori (asemat 0 ja 1)
$02A1 – B0	BAM:in kuvat
$02B1 – D4	sisällysluettelon puskuri
$02D5 – F8	virheilmoitusten puskuri

$02F9	lippu: älä kirjoita BAM:ia. Asetetaan nollaksi ennen jokaista komentoa ja jokaisen komennon jälkeen
$02FA – FB	vapaiden lohkojen lukumäärä, alatavu asemille 0 ja 1
$02FC – FD	vapaiden lohkojen lukumäärä, ylätavu asemille 0 ja 1
$02FE – FF	askelmoottorin jakso
$0300 – FF	puskuri 0
$0400 – FF	puskuri 1
$0500 – FF	puskuri 2
$0600 – FF	puskuri 3
$0700 – FF	puskuri 4 (vain BAM:ille)

LIITE 2

VIRHEILMOITUKSET

Virheilmoituksen yleinen muoto on:

VN, VI$, VU, VS

VN = virheen numero
VI$ = virheen yleiskuvaus
VU = ura, jolla virhe on havaittu
VS = sektori, jolla virhe on havaittu

Esimerkiksi:

```
34 SYNTAX ERROR 0  0
```

Kuva L2.1

Seuraavaan on koottu kaikki virheilmoitukset, niiden selitykset ja vastaavasti virheiden mahdolliset aiheuttajat. Näitä virheilmoituksia ei saa sekoittaa C64:n antamiin virheilmoituksiin, jotka löydät tietokoneen käsikirjasta. Edelleen kannattaa muistaa, että FDC (Floppy Disk Controller, levyohjain) tarkistaa ja ilmoittaa luku/kirjoitusvirheet määrätyssä järjestyksessä. Lukuvirheet ilmoitetaan järjestyksessä 21, 20, 27, 29, 20, 22, 23, 0 ja kirjoitusvirheet järjestyksessä 73, 29, 26, 25, 0.

00, OK, 00, 00
Kaikki on kunnossa.

01, FILES SCRATCHED, NN, 00
Tämä ilmoitus annetaan SCRATCH-komennon suorittamisen jälkeen. NN ilmoittaa levykkeeltä poistettujen ohjelmien lukumäärän, joka esimerkiksi jokereita käytettäessä voi olla suurempi kuin 1. Merkkivalo ei pala, koska kysymyksessä ei ole varsinainen virhe.

20, READ ERROR, VU, VS
Levyohjain ei ole löytänyt otsikkolohkoa. Kun jokin määrätty sektori on luettava, levyohjain etsii ensin sektorin tahdistinmerkin ja lukee sen jälkeen tulevan tavun, otsikkolohkon tunnistimen. Tunnistinta verrataan arvoon $08. Jos arvot eivät ole samat, yritetään uudestaan. Jos oikea tunnistin ei löydy 90 yrityksen jälkeen, annetaan virheilmoitus. Useimmiten levyke on viallinen. Vika voi aiheutua siitä, että levyke on asemassa, kun virta kytketään tai katkaistaan. Se voi myös olla osa suojausta.

21, READ ERROR, VU, VS
Levyohjain ei ole löytänyt sektorin otsikkolohkon tahdistinmerkkiä määrätyn ajan (20 ms) kuluessa. Syynä voi olla alustamaton levyke, levykkeen puuttuminen asemasta tai luku- ja kirjoituspään kohdistusvirhe. Tämä voi myös olla osa suojausta.

22, READ ERROR, VU, VS
Levyohjain on löytänyt sektorin otsikkolohkon, mutta tietolohkon ensimmäinen tavu, tunnistin, ei ole oikea eli $07 (muistipaikka $0047). Tämä voi aiheutua luku- ja kirjoituspään kohdistusvirheestä tai virheellisestä lohkon kirjoittamisesta. Se voi olla osa suojausta.

23, READ ERROR, VU, VS
Virhe tietolohkon tarkistussummassa. Levyohjain on löytänyt otsikkolohkon ja lukenut tietolohkon tiedot. Tietolohkon 256 tavusta laskettu tarkistussumma ei ole kuitenkaan täsmännyt oikean tarkistussumman kanssa, joka on tallennettu levykkeelle tietojen tallennuksen yhteydessä. Yksi tai useat tietotavut ovat virheelliset. Tiedostoja voidaan yrittää pelastaa toiselle levykkeelle. Virhe saattaa aiheutua levy-

aseman kytkemisestä tai sammuttamisesta levykkeen ollessa asemassa. Se voi myös olla osa suojausta.

24, READ ERROR, VU, VS
Tavun dekoodausvirhe. Levyohjain on lukenut levykkeeltä koodatun tavun, jolle ei löydy heksadesimaalivastinetta. 1541 ei tarkista tämäntyyppistä virhettä.

25, WRITE ERROR, VU, VS
Kysymyksessä on oikeastaan VERIFY ERROR. Kun tietolohko on kirjoitettu, se luetaan takaisin ja tavuja verrataan puskurissa oleviin tavuihin. Virheilmoitus annetaan, jos tavut eivät täsmää. Virhe johtuu yleensä viallisesta levykkeestä. Korjausta voidaan yrittää antamalla virheen aiheuttanut käsky uudestaan. Jos tämä ei auta, sektori kannattaa merkitä B–A-käskyllä varatuksi.

26, WRITE PROTECT ON, VU, VS
Levykkeessä on kirjoituksenestotarra. Tiedostoa ei kirjoiteta. Tarra pitää poistaa. Levyke ei ole vahingoittunut.

27, READ ERROR, VU, VS
Virhe otsikkolohkon tarkistussummassa. Luettuaan otsikkolohkon levyohjain on dekoodannut otsikkolohkon tavut. Tämän jälkeen laskettu tarkistussumma ei ole täsmännyt otsikkolohkon oikean tarkistussumman kanssa. Virhe voi aiheutua levyaseman kytkemisestä tai sammuttamisesta levykkeen ollessa asemassa. Se voi joskus olla myös osa suojausta.

28, WRITE ERROR, VU, VS
Liian pitkä tietolohko. Uuden tietolohkon kirjoittamisen jälkeen levyohjain ei ole löytänyt seuraavan lohkon alussa olevaa tahdistinmerkkiä. 1541 ei tarkista tämäntyyppistä virhettä.

29, DISK ID MISMATCH, VU, VS
Otsikkolohkon ID-merkit eivät täsmää levyaseman RAM-

muistiin (paikat $12 ja $13) tallennettujen ID-merkkien kanssa. Levykettä ei ole initialisoitu tai sektorin otsikkolohko on viallinen. Syynä voi olla myös luku- ja kirjoituspään kohdistusvirhe. Sitä käytetään usein suojauksen osana.

30, SYNTAX ERROR, 00, 00
Yleinen kirjoitusvirhe. DOS ei pysty tulkitsemaan komentokanavan kautta lähetettyä komentoa. Komento pitää tarkistaa ja korjata.

31, SYNTAX ERROR, 00, 00
Laiton komento. DOS ei tunnista komentokanavan kautta lähetettyä komentoa. Esim. BACKUP ja DUPLICATE ovat sellaisia Commodoren kaksoislevyasemissa käytettyjä komentoja, joita 1541:n DOS ei tunnista.

32, SYNTAX ERROR, 00, 00
Liian pitkä komentojono. Komentokanavan kautta lähetetyssä käskyssä saa olla yhteensä korkeintaan 58 merkkiä. Käskyjonoa pitää lyhentää.

33, SYNTAX ERROR, 00, 00
Laiton tiedoston nimi. Jokereita on käytetty väärin OPEN- tai SAVE-käskyjen yhteydessä. Jokerit pitää poistaa.

34, SYNTAX ERROR, 00, 00
Tiedoston nimi puuttuu. Komennosta puuttuu yhden tai useamman tiedoston nimi. Käskysanan jälkeen puuttuu esim. : .

39, FILE NOT FOUND, 00, 00
Harvinainen virhe, joka esiintyy vain käytettäessä &-tiedostoja. Käskyssä mainittua tiedostoa ei löydy levykkeen sisällysluettelosta.

50, RECORD NOT PRESENT, 00, 00
Tietue puuttuu. P-komentoa on yritetty käyttää suhteelli-

sen tiedoston tietueeseen, jota ei ole. Ilmoituksesta ei tarvitse välittää silloin, kun suhteelliseen tiedostoon kirjoitetaan uusia tietueita (DOS laajentaa tiedoston pituutta automaattisesti niin, että uusi tietue mahtuu). Jos tietueen pitäisi olla tiedostossa, ohjelma on syytä tarkistaa virheiden havaitsemiseksi. Virhe voidaan välttää luomalla koko tiedosto kerralla (ks. 7. luku).

51, OVERFLOW IN RECORD, 00, 00
Suhteelliseen tiedostoon kirjoitetun tietueen pituus on liian suuri. Ylimääräisiä merkkejä ei kirjoiteta järjestyksessä seuraavan tietueen suojelemiseksi.

52, FILE TOO LARGE, 00, 00
P-komennossa ilmoitettu tietueen numero on liian suuri. Levykkeen tila ei riitä tiedostolle, jos tietue lisätään.

60, WRITE FILE OPEN, 00, 00
Tiedosto, jota ei ole suljettu kunnolla, on yritetty avata lukemista varten. Tällaisen tiedoston tyypin edessä sisällysluettelossa on *-merkki, esim. *SEQ. Tiedosto ei sulkeudu esimerkiksi silloin, kun levyke poistetaan asemasta ennen avatun tiedoston sulkemista. Tällaisen tiedoston tiedot voidaan lukea käyttämällä OPEN-käskyn yhteydessä toimintatilaa M.

61, FILE NOT OPEN, 00, 00
Yritetään käsitellä tiedostoa, jota ei ole avattu. Virheilmoitus on harvinainen. Tavallisesti DOS hylkää tiedostonkäsittelypyynnön.

62, FILE NOT FOUND, 00, 00
OPEN-, LOAD-, COPY- tai RENAME-käskyn yhteydessä käytettyä tiedostonimeä ei löydy levykkeen sisällysluettelosta. Virheilmoitusta ei anneta silloin, kun SCRATCH-komennon yhteydessä olevaa tiedostonimeä ei löydy. Tällöin DOS ilmoittaa 01, FILES SCRATCHED, 00, 00.

63, FILE EXISTS, 00, 00
OPEN-, SAVE-, COPY- tai RENAME-käskyn yhteydessä käytetty tiedostonimi on jo levykkeen sisällysluettelossa.

64, FILE TYPE MISMATCH, 00, 00
Tiedostonkäsittelykäskyn yhteydessä ilmoitetun tiedoston tyyppi ei täsmää tiedoston sisällysluettelossa olevan tyypin kanssa.

65, NO BLOCK, VU, VS
Tämä virheilmoitus esiintyy vain B–A (BLOCK-ALLOCATE)-käskyn yhteydessä. Ilmoitus annetaan, jos yritetään varata lohkoa, joka on jo merkitty BAM:iin varatuksi. VU ja VS ovat seuraavan vapaan lohkon uran ja sektorin järjestysluvut.

Jos vapaita lohkoja ei enää ole, VU = VS = 0. B–A-käsky ei toimi kunnolla 1541-levyasemalla. Jos lohko on jo varattu, kaikki uran lohkot saatetaan merkitä varatuiksi. Voi käydä myös niin, että VU:n arvoksi ilmoitetaan 18 (!). Katso kahdeksatta lukua.

66, ILLEGAL TRACK OR SECTOR, VU, VS
On yritetty kirjoittaa/lukea lohkoa, jonka ura- tai sektoriluku ei ole sallituissa rajoissa. Ilmoitus voi esiintyä käytettäessä M-tilaa sulkemattoman tiedoston lukemiseksi.

67, ILLEGAL TRACK OR SECTOR, VU, VS
DOS on yrittänyt käsitellä kiellettyä uraa/sektoria. Tiedoston ura-sektorilinkitys voi olla virheellinen ja osoittaa kiellettyyn uraan/sektoriin.

70, NO CHANNEL, 00, 00
Vaadittu suorasaantikanava ei ole käytettävissä. OPEN-käskyssä ilmoitettua tiedostoa ei avata. Vikana on yleensä se, että tiedosto yritetään avata silloin, kun sallittu määrä tiedostoja on jo auki.

71, DIRECTORY ERROR, VU, VS
Levyaseman RAM-muistissa oleva vapaiden lohkojen määrä ei täsmää BAM:in kanssa. Levyke pitää initialisoida.

72, DISK FULL, 00, 00
Levykkeen tila on käytetty kokonaan tai levykkeen sisällysluettelo on täynnä. Sisällysluetteloon mahtuu korkeintaan 144 tiedostoa. 1541 sulkee tiedoston, jos levy täyttyy SAVE-käskyä suoritettaessa, mutta tiedosto on epätäydellinen. Jos levy täyttyy tiedostoon kirjoitettaessa, tiedosto ei sulkeudu kunnolla.

73, CBM DOS V2.6 1541, 00, 00
Ilmoitus saadaan, jos virhekanava luetaan heti levyaseman kytkemisen jälkeen. Virheilmoituksena tämä esiintyy kirjoitettaessa levykkeelle, joka on formatoitu levyasemalla, jonka DOS-versio ei ole V2.6, esim. CBM 3040.

74, DRIVE NOT READY, 00, 00
Levyasemassa ei ole levykettä tai luukku ei ole kunnolla kiinni. Ilmoitus saattaa esiintyä myös suorasaantitiedostoja käytettäessä. Jos se esiintyy, kun suorasaantitiedosto avataan OPEN-käskyllä, käskyn yhteydessä on ilmoitettava halutun puskurin numero.

LIITE 3

TAULUKKOJA

COMMODOREN TIEDOSTOTYYPIT

Tähän taulukkoon on koottu kaikki Commodoren tiedostotyypit. Tiedostosta ilmoitetaan tyyppi heksadesimaalilukuna ja desimaalilukuna, nimi ja tyypin tunnus levykkeen sisällysluettelossa.

HEX	ASCII	TYYPPI	SIS.L.
$00	0	tuhottu tiedosto	ei
$00	0	sulkematon DEL-tiedosto	ei
$01	1	sulkematon peräkkäistiedosto	*SEQ
$02	2	sulkematon ohjelmatiedosto	*PRG
$03	3	sulkematon käyttäjän tiedosto	*USR
$04	4	sulkematon suhteellinen tiedosto	ei
$80	128	DEL-tiedosto	DEL
$81	129	peräkkäistiedosto	SEQ
$82	130	ohjelmatiedosto	PRG
$83	131	käyttäjän tiedosto	USR
$84	132	suhteellinen tiedosto	REL
$A0	160	DEL-tiedosto (@-korvaus)	DEL
$A1	161	peräkkäistiedosto (@)	SEQ
$A2	162	ohjelmatiedosto (@)	PRG
$A3	163	käyttäjän tiedosto (@)	USR
$A4	164	suhteellinen tiedosto (@)	ei
$C0	192	lukittu DEL-tiedosto	DEL<
$C1	193	lukittu peräkkäistiedosto	SEQ<

$C2	194	lukittu ohjelmatiedosto	PRG<
$C3	195	lukittu käyttäjän tiedosto	USR<
$C4	196	lukittu suhteellinen tiedosto	REL<

MUUNNOSTAULUKKO

HEKSADESIMAALI/ DESIMAALI

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Kuva L3.1

URILLE MAHTUVIEN SEKTOREIDEN LUKUMÄÄRÄ

URAT	SEKTORIT	SEKTOREIDEN LUKUMÄÄRÄ
1 – 17	0 – 20	21
17 – 24	0 – 18	19
25 – 30	0 – 17	18
31 – 35	0 – 16	17

SISÄLLYSLUETTELON SEKTOREIDEN SISÄLTÖ

Tiedoston n:o	1	2	3	4	5	6	7	8
Tyyppi	02	34	66	98	130	162	194	226

1. ura	03	35	67	99	131	163	195	227
1. sektori	04	36	68	100	132	164	196	228
Nimen alku	05	37	69	101	133	165	197	229
SS:n ura	21	53	85	117	149	181	213	245
SS:n sektori	22	54	86	118	150	182	214	246
Tietueen pituus	23	55	87	119	151	183	215	247
Tiedoston pituus	30	62	94	126	158	190	222	254
ala/ylä	31	63	95	127	159	191	223	255

Tavun järjestysnumero on ilmoitettu desimaalilukuna. SS = sivusektori.

LIITE 4

DOS 5.1

Levyaseman mukana tulleella levykkeellä VIC-1541 TEST/DEMO on lyhyt konekielinen ohjelma DOS 5.1, johon tutustumme nyt. Kysymyksessä ei ole uusi levyaseman käyttöjärjestelmä vaan apuohjelma, joka sisältää useita levyaseman käyttöä helpottavia ja nopeuttavia käskyjä.

DOS 5.1 latautuu alueelle $CC00–$CF60 eikä ole tavallisen BASIC-ohjelman tiellä. On kuitenkin mahdollista, että DOS 5.1 ei toimi joidenkin konekielisten ohjelmien kanssa.

Ennen kuin tutustumme itse ohjelmaan ja sen tarjoamiin käskyihin, seuraa lyhyt yhteenveto ohjelman eduista:

- Levyasemakäskyjä on helpompi käyttää suorassa toimintatilassa.
- Lainausmerkkejä ei tarvita.
- Komentokanavaa ei tarvitse avata.
- Virheilmoitukset voidaan lukea suoraan.
- Uusia käskyjä voi käyttää myös omissa BASIC-ohjelmissa.
- Levykkeen sisällysluettelo voidaan lukea muistissa olevaa ohjelmaa häiritsemättä.

DOS 5.1 -OHJELMAN LATAAMINEN

Voit ladata ohjelman samalla levykkeellä olevan latausohjelman (C–64 WEDGE) avulla tai suoraan. Latausohjelmaa käyttäessäsi toimit seuraavasti:

```
LOAD "C-64 WEDGE",8
```

Kuva L4.1

Kun latausohjelma on latautunut, suoritat sen komennolla RUN. Hetken kuluttua näyttöön tulostuu ilmoitus DOS MANAGER V5.1/071382 ja tekijän nimi, BOB FAIRBAIRN. Ohjelma on nyt valmis käyttöön.

Jos lataat ohjelman ilman latausohjelmaa, kirjoitat seuraavan latauskäskyn:

```
LOAD "DOS 5.1",8,1
```

Kuva L4.2

Kun ohjelma on latautunut, kirjoitat komennon SYS 52224 ja painat RETURN-näppäintä. Uudet käskyt ovat nyt käytettävissä.

LEVYKETIETOJEN PÄIVITTÄMISKÄSKYT

Kaikissa seuraavissa käskyissä voidaan merkin > asemesta käyttää merkkiä @ . Merkeillä > ja @ on erikoinen merkitys DOS 5.1 -ohjelmalle. Ne toimivat komentosanoina. DOS 5.1 tarkistaa näiden merkkien jäljessä tulevat merkit, ja jos ne muodostavat sallitun DOS 5.1 -komennon, komento suoritetaan. Merkkiä > voidaan silti käyttää myös sen tavallisessa merkityksessä (suurempi kuin).

Seuraavassa on käytetty päivityskäskyjen lyhennettyä muotoa. Käskyjen syntaksit voit tarkistaa kolmannesta luvusta.

Levykkeen alustaminen (tavallinen)

```
>N0:NIMI,ID
```

Kuva L4.3

Levykkeen alustaminen (nopea)

```
>N0:NIMI
```

Kuva L4.4

Tiedoston tuhoaminen

```
>S0:NIMI
```

Kuva L4.5

Tiedoston nimen muuttaminen

```
>R0:UUSI NIMI=0:VANHA NIMI
```

Kuva L4.6

Tiedoston kopioiminen

```
>C0:FILE1=0:FILE2
```

Kuva L4.7

Levykkeen initialisointi

```
>I0
```

Kuva L4.8

Validate

```
>V0
```

Kuva L4.9

Levyaseman reset

```
>UJ
```

Kuva L4.10

Virheilmoituksen lukeminen

```
>
```

Kuva L4.11

DOS 5.1 -ohjelman lopettaminen

```
>Q
```

Kuva L4.12

Lopettamisen jälkeen ohjelmaan päästään takaisin komennolla SYS 52224, ellei virta ole ollut välillä kytkettynä pois tietokoneesta.

Kaikista edellisistä käskyistä saa jättää levyaseman numeron 0 pois. Tiedostojen nimissä voidaan käyttää jokereita * ja ?. Seuraavassa joitakin esimerkkejä:

```
1) >S0:K*
2) >S0:*=S
3) >S0:?A*
```

Kuva L4.13

Ensimmäisessä esimerkissä tuhotaan kaikki tiedostot, joiden nimi alkaa K-kirjaimella. Toisessa esimerkissä tuhotaan kaikki peräkkäistiedostot ja kolmannessa esimerkissä kaikki tiedostot, joiden nimen toinen kirjain on A.

OHJELMIEN TALLENNUS JA LATAUS

Seuraavaan on koottu ohjelmien lataukseen ja tallennukseen sekä sisällysluettelon tutkimiseen käytettävät käskyt.

BASIC-ohjelman lataus

```
/NIMI
```

Kuva L4.14

BASIC-ohjelman lataus ja suoritus

```
^NIMI
```

Kuva L4.15

Konekielisen ohjelman lataus

```
%NIMI
```

Kuva L4.16

BASIC-ohjelman tallennus

```
←NIMI
```

Kuva L4.17

BASIC-ohjelman tallennus + kirjaimen lisääminen

```
←NIMI[kirjain]
```

Kuva L4.18

Sisällysluettelon tulostaminen näyttöön

```
>$0
```

Kuva L4.19

Varsinkin sisällysluetteloa luettaessa on hyödyllistä käyttää jokereita.

```
1)  >$0:PA*
2)  >$0:?SU*
3)  >$0:*=P
4)  >$0:$$
```

Kuva L4.20

Ensimmäinen esimerkki tulostaa sisällysluettelosta ne ohjelmat, joiden nimet alkavat kirjaimilla PA. Toinen esimerkki tulostaa ne ohjelmat, joiden nimien toisena ja kolmantena kirjaimena ovat S ja U. Kolmannessa esimerkissä tulostetaan sisällysluettelosta kaikki ohjelmatiedostot, ja viimeinen esimerkki tulostaa sisällysluettelon ensimmäisen ja viimeisen rivin (nimi + ID ja vapaat lohkot). Viimeisessä esimerkissä voi kaksoispisteen jälkeen olla mikä tahansa merkkijono, joka ei esiinny ohjelman nimenä.

Ohjelma voidaan tallentaa myös niin, että nimen perään liitetään jokin kirjain. Kirjain tulee nimen 16. merkiksi. Varsinaisen nimen ja kirjaimen väliin tulee automaattisesti mahdolliset välilyönnit CHR$(160). Kun tällä tavalla tallennettu ohjelma ladataan, nimeen ei tarvitse kirjoittaa lisättyä merkkiä. Ohjelmaa tuhottaessa kirjaimen pitää olla oikealla paikallaan.

KÄSKYJEN KÄYTTÖ OHJELMASSA

DOS 5.1 -käskyt voidaan myös lisätä BASIC-ohjelmaan, jos DOS 5.1 on ladattu muistiin ennen BASIC-ohjelman suorittamista.

Kun käskyjä käytetään ohjelmassa, on edellä esitettyjen seikkojen lisäksi huomioitava seuraavat lisäykset:

- Komentosana tai ohjelman nimi on kirjoitettava lainausmerkkeihin.
- Käsky pitää kirjoittaa sellaisenaan merkin > tai @ jälkeen. Käskyä ei voi liittää merkkijonomuuttujan arvoksi.

Kokeile seuraavia ohjelmia.

```
10 PRINT "PAINA RETURN, NIIN FORMATOIN L
LEVYN"
20 GETA$:IF A$<>CHR$(13) THEN 20
30 >"N0:UUSI LEVY,L1"
40 PRINT "LEVYKE ON FORMATOITU !"
```

Kuva L4.21

Ensimmäistä ohjelmaa EI voida kirjoittaa näin:

```
10 PRINT "PAINA RETURN, NIIN FORMATOIN L
LEVYN"
20 GETA$:IF A$<>CHR$(13) THEN 20
25 N$="N0:UUSI LEVY,L1"
30 >N$
40 PRINT "LEVYKE ON FORMATOITU !"
```

Kuva L4.22

Tässä komento on liitetty merkkijonon N$ arvoksi.

DOS 5.1 -OHJELMAN KOPIOIMINEN

Jos käytettävissäsi ei ole konekielimonitoria tai kopiointiohjelmaa, voit kopioida DOS 5.1 -ohjelman toiselle levykkeelle seuraavasti:

- Lataa DOS 5.1 tietokoneen muistiin.
- Kirjoita seuraavat käskyt:

```
POKE 44,204:POKE 43,0
POKE 46,207:POKE 45,89
```

Kuva L4.23

- Laita asemaan levyke, jolle haluat tallentaa ohjelman.
- Tallenna DOS 5.1 käskyllä SAVE "DOS 5.1",8.
- Kun ohjelma on tallentunut, kirjoita NEW ja paina RETURN-näppäintä.
- Jos käytät latausohjelmaa, tallenna se tavalliseen tapaan uudelle levykkeellesi.

Tietokonelehdissä on julkaistu ohjelmia, joilla tätä apuohjelmaa voidaan laajentaa entisestään, esim. COMPUTE! 1984/marraskuu.

LIITE 5

LEVYKKEEN OHJELMAT

Kaikki levykkeen ohjelmat ovat BASIC-ohjelmia. Ohjelmissa BACKUP ja FIILI on konekielisiä osia kirjoitus/lukutapahtumien nopeuttamiseksi.

Ohjelmat ladataan komennolla LOAD "NIMI",8 ja suoritetaan komennolla RUN. Seuraavassa on lyhyt kuvaus jokaisesta ohjelmasta.

BACKUP

Ladattuasi ohjelman laita kopioitava levyke asemaan ja anna RUN-komento. Ohjelma kopioi levykkeen korkeintaan kolmessa osassa. Kun ensimmäinen osa on luettu, sinua pyydetään vaihtamaan asemaan uusi levyke (se levyke, jolle kopioit), jonka pitää olla formatoitu. Kun tiedot on kirjoitettu, asemaan on tarvittaessa vaihdettava alkuperäinen levyke (old diskette). Näin jatketaan, kunnes koko levyke on kopioitu.

Kopioinnin jälkeen voit ottaa samasta levykkeestä toisen kopion antamalla kysymykseen "another diskette?" vastaukseksi "y".

FIILI

Tällä ohjelmalla voit kopioida kerralla korkeintaan 20 ohjelma- ja peräkkäistiedostoa.

Kun olet laittanut alkuperäisen levykkeen asemaan ja painanut <RETURN>, tulostuvat sisällysluettelon tiedostot yksi kerrallaan näyttöön. Jos haluat kopioida tiedoston, paina "K". Kun kaikki tiedostot on käyty läpi tai kun on valittu kopioita-

vaksi 20 tiedostoa, kopiointi voidaan aloittaa painamalla "1". Kun painat "2", voit vaihtaa toisen levykkeen ja valita uudestaan. Kun halutut tiedostot on luettu, laita asemaan se levyke, jolle tiedostot kirjoitetaan.

TUTKI U&S

Valitaan ensin kuvaruutu (K) tai printteri (P). Tämän jälkeen annettaan ura ja sektori, joita halutaan tutkia, esimerkiksi 18,0. Kun puolet sektorin sisällöstä on tulostettu, tulostus pysähtyy. Painamalla <RETURN>, pääset eteenpäin. Kun koko sektori on tulostettu, voit valita seuraavan tiedostoon kuuluvan sektorin (S), uuden sektorin (U) tai lopettaa ohjelman (L).

BAMBAM

Ohjelma tulostaa näyttöön levykkeen lohkokartan eli BAM:in, levykkeen nimen, ID-numeron ja vapaiden lohkojen lukumäärän. Vapaa lohko on merkitty keltaisella ympyrällä ja varattu lohko sinisellä vinoneliöllä. Kun kartta on valmis, näytön oikeaan alanurkkaan tulostuu P?. Painamalla "P"-näppäintä saat kopioiduksi näytön kirjoittimelle. Muiden näppäinten painaminen lopettaa ohjelman.

SUPERDIR

Tämän ohjelman avulla saat levykkeen sisällysluettelon hieman laajennettuna. Näyttöön tulostuu samalla kertaa 8 tiedoston tiedot. Painamalla <RETURN> saat seuraavat 8 tiedostoa.

Kustakin tiedostosta ilmoitetaan nimi, käytettyjen lohkojen lukumäärä, käytettyjen tavujen lukumäärä, ensimmäinen ura/sektori ja tyyppi. Lopuksi ilmoitetaan levykkeen vapaa tila lohkoina ja tavuina sekä sisällysluetteloon mahtuvien tiedostojen lukumäärä.

MUUTA NIMI & ID

Ohjelman avulla voit muuttaa levykkeen nimen ja ID-numeron (kosmeettinen muutos).

TARKASTA ID

Ohjelman avulla voit verrata levykkeen varsinaista ID-numeroa

ja kunkin sektorin otsikkolohkossa olevaa ID-numeroa toisiinsa.

TARKASTA LEVYKE

Ohjelma tutkii levykkeen urat ja ilmoittaa mahdolliset virheet.

TARKASTA URA

Ohjelma tutkii halutun uran kaikki sektorit ja antaa ilmoituksen jokaisen sektorin kunnosta.

TUTKI DRIVE

Tämän ohjelman avulla voit tutkia levyaseman RAM/ROM-muistia. Tulostus saadaan joko kuvaruutuun (K) tai printterille (P). Alku- ja loppuosoite annetaan 10-järjestelmän lukuina, esim. 1000, 1200.

PELASTA

Ohjelma pelastaa vahingossa tuhotun tiedoston, jonka nimi ja tyyppi pitää ilmoittaa.

LUKITSE

Tällä ohjelmalla voit lukita tai avata tiedoston. Jos tiedosto ei ole lukittu, se lukitaan. Lukittu tiedosto avataan. Lukitun tiedoston tunnuksena on merkki <, esim. PRG<.

VARAA

Ohjelmalla varataan kaikki sulkemattoman tiedoston lohkot. Tiedostosta pitää ilmoittaa nimi ja tyyppi.

SULKEMATON

Ohjelman avulla voit yrittää pelastaa P-, U- tai S-tyyppisen sulkemattoman tiedoston. Tiedoston nimi ja tyyppi on ilmoitettava. Lisäksi pitää ilmoittaa pelastetun tiedoston nimi ja tyyppi.

SAVEML	katso luku 5
LUOML	katso luku 5
SISLU 1	katso luku 5
SISLU 2	katso luku 5
LUO INDEKSI.1	katso luku 7
LUO INDEKSI.2	katso luku 7

HAKEMISTO

–A–

alustaminen 9, 31
APPEND 50, 85
asemanumero 50
AT-merkki (@) 15, 66, 92
autostart 17
avainkenttä 111, 112

–B–

B–A 118, 132
B–E 118, 136
B–F 118, 135
B–P 118, 123
B–R 118, 120
B–W 118, 124
BAM 34, 149
BAM-puskuri 116, 128
binaarihaku 88
Block Availability Map 34, 149
BLOCK–ALLOCATE 118, 132
BLOCK–EXECUTE 118, 136
BLOCK–FREE 118, 135
BLOCK–READ 118, 120
BLOCK–WRITE 118, 124
BUFFER–POINTER 118, 123
BUMP 139

–C–

C-64 WEDGE 193
carbage collection 91
CLOSE 22, 24, 56, 84, 111
COPY 38

–D–

DEL 46, 171
Disk Operating System 21
DOS 21
DOS 5.1 193

–E–

EOF 55
EOR 148
erotin 45
EXECUTE 139

–F–

FDC 139, 181
File Manager 173
Floppy Disk Controller 139, 173, 181

–G–

GCR 175
GET# 22, 53, 79, 108

-H-

haku ja sijoitus 87

-I-

ID-numero 10, 32, 174
indeksitiedosto 99, 111
INITIALIZE 34
INPUT# 22, 53, 79, 108
inter-sector gap 147
Interface Processor 139
IP 139

-J-

job queue 139
jobs 139
jokerit 41, 196, 197
JUMP 139

-K-

kanavanumero 22, 49
kenttä 73, 100
kentän pituus 73, 100
komentokanava 21
komentopuskuri 117, 173
korvaava tallettaminen 66
kuplalajittelu 112
käyttäjän tiedosto 14, 142
käyttöjärjestelmä 21

-L-

laitenumero 22, 48
latausosoite 57, 157
levykkeen formaatti 13
levyohjaintila 173
linkkiosoittimen alatavu 58
linkkiosoittimen ylätavu 58
listener address 130, 144
LOAD 16, 59
lohko 146
lohkokartta 149
looginen tiedostonumero 22
lähettävä laite 130

-M-

M-E 118, 131
M-R 118, 127
M-W 118, 129
MEMORY-EXECUTE 118, 131
MEMORY-READ 118, 127
MEMORY-WRITE 118, 129
MODIFY 50, 93

-N-

NEW 9, 31
NEW (nopea) 32, 34
nollasivu 173
näppäimistöpuskuri 97

-O-

ohjelmatiedosto 14, 57
OPEN 47, 76, 102
otsikkolohko 147, 174
otsikkolohkon tarkistus-summa 147
otsikkolohkon tunnus-merkki 147
otsikkoväli 147

-P-

P-komento 104
peräkkäistiedosto 14, 73

pino 173
PRG 14, 46, 189
PRG < 190
PRINT# 22, 52, 77, 105
puskuri 93, 117, 180
päivityskomento 31

-R-

READ 50, 139
REL 14, 46, 189
REL< 190
RENAME 35
rivinumeron alatavu 58
rivinumeron ylätavu 58
ROM-rutiini GET# 29
ROM-rutiini INPUT# 30
roskien keruu 91

-S-

satunnaistiedosto 46, 133
SAVE 14, 62
SCRATCH 36
SEEK 139
sektori 146
sektoriosoitin 153
sektoriväli 147
SEQ 14, 46, 189
SEQ< 189
Shell-Metznerin lajittelu 112
simuloitu GET# 29
simuloitu INPUT# 30
sisällysluettelo 11, 69, 152
sivusektori 164
suhteellinen tiedosto 14, 99
suorasaantikäsky 115
syöttöpuskuri 130

-T-

tahdistusmerkki 147
talker address 130, 144
tarkistussumma 143, 147, 148
tiedosto 45
tiedoston koko 13
tiedoston korvaaminen 92
tiedoston nimeäminen uudelleen 31, 35
tiedoston nimi 13, 50
tiedoston tuhoaminen 31, 36
tiedoston tunnusosa 49
tiedoston tyyppi 13, 46
tiedostonhallintatila 173
tietolohko 148
tietolohkon tarkistussumma 148
tietolohkon tunnusmerkki 148
tietopuskuri 173
tietue 73
tietueen pituus 51, 102
tietueen sijoitus 86
tietueosoitin 104
tilamuuttuja ST 55, 80
toimintatila 50, 76
toisio-osoite 49
työjono 139, 173, 174
työn koodi 139

-U-

U0 119, 137
U1 119, 120, 137
U2 119, 124, 137
U3 - U9 119, 138
U: 138
UA - UJ 138
ura 145
uraosoitin 153
USER0 119
USER1 119, 120
USER2 119, 124
USER3 - USER9 119

USR 14, 171, 189
USR< 190

-V-

VALIDATE 40
vastaanottava laite 130
VERIFY 17, 139
VI $ 26, 181
virheilmoitus 26, 181
virhekoodi 140
virhepuskuri 140
VN 26, 181
VS 26, 181
VU 26, 181

-W-

WRITE 50, 139

Reijo Lundahl

1541-LEVYASEMA

Commodore 64

Kirja sisältää kaiken käyttäjän tarvitseman tiedon 1541-kalvolevyasemasta. Selvän rakenteen, useiden esimerkkien ja mukana seuraavan levykkeen vuoksi kirja sopii aloitteleville Commodore-tietokoneiden käyttäjille. Kirjasta on apua myös edistyneemmille mikroharrastajille, koska sitä tehtäessä on pyritty oikaisemaan aiemmin esiintyneitä virheellisiä tietoja 1541-kalvolevyasemasta. Lisäksi kirjassa on paljon aikaisemmin julkaisematonta tietoa.

61

ISBN 951-35-3707-7