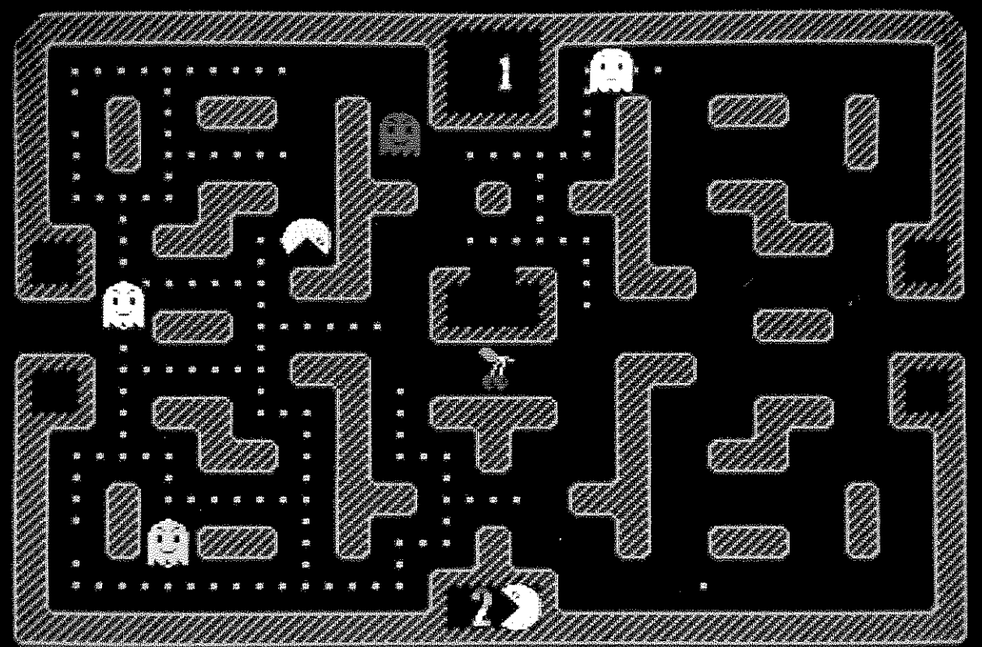
DAI NAMIC

20

tweemaandelijks tijdschrift januari - februari 1984



personal computer users club

een uitgave van dainamic v.z.w.
verantw. uitgever w. hermans, mottaart 20 - 3170 herselt

International

QUE

face

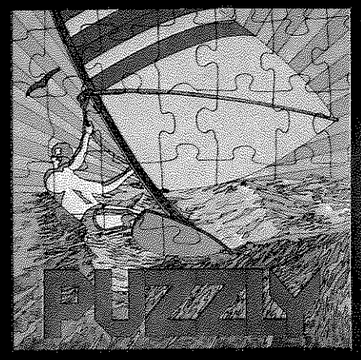
IAL»

S

UEL

DIALOG INFORMATIQUE

PUZZLY



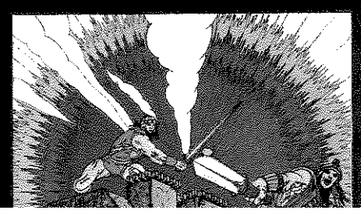
PUZZLY

Vous aimez l'art...
Vour aimez les casses têtes... ...jouez PUZZLY

PUZZLY, le premier puzzle sur micro-ordinateur.
Mais PUZZLY c'est aussi un graphisme étonnant, plusieurs niveaux de jeux, un chronométrage.
Quant aux images, sa vidéothèque ne cesse de s'agrandir.

Ce coffret contient 4 puzzles.

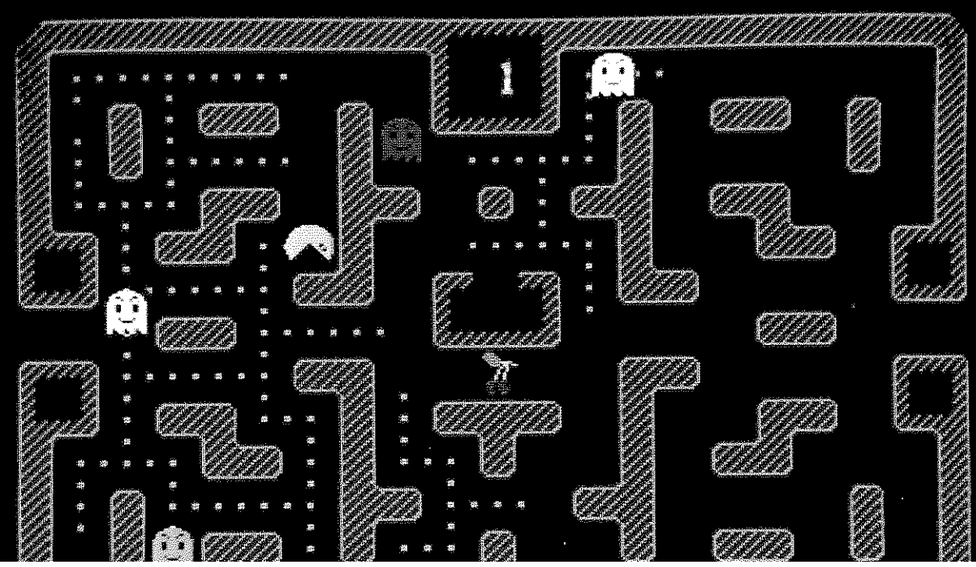
DIALOG INFORMATIQUE



DAI
NAMIC

20

tweemaandelijks tijdschrift januari - februari 1984



4th international DAInamic meeting on saturday 21 april in Tongelsbos, Westerlo.

BOSSTRAAT 2 3180 WESTERLO

FROM 10 H TO 18H

WITH :

PACMAN-CONTEST

1e price : RGB-MONITOR
(offered by INDATA N.V.)

2e price : OKI-MATRIXPRINTER
(offered by MIKROSHOP HAGELAND P.V.B.A.)

3e price : Software packages

4e price : Software packages

5e price : Software packages

Order your PAC-MAN cassette now (with competition-card), train yourself and join the contest on our meeting.

PAC-MAN competition will end at 17 h.

montage:
Il vous faut connecter la carte d'eproms sur le xbus (connecteur interne)de la machine puis effectuer une petite modification materiel qui consiste à relier deux point du circuit imprimée du DAI.

Remarque: Il est possible d'adapter le Ken-dos sur les drives de chez DAI. Une modification devra neamoins etre faite par le constructeur . Le temps d'accés sera alors legerement reduit.

SOMMAIRE DES COMMANDES DU KEN-DOS

VERIFY	Verifie un fichiers et affiche ses parametres
UNLOCK	Deprotege un fichier à l'aide du mot de passe
RESTOR	Reprend un fichier effacé
RENAME	Renomme un fichier
PROTEC	Protege un fichier contre l'ecriture
MANUAL	Acces manuel au secteurs/pistes
LPRINT	Routine d'impression
FORMAT	Formatte une disquette
CREATE	
COMPAC	Copie sur une disquettes sans les fichiers effacés
BACKUP	Fait une copie de la disquette
VOICE	Utilisée avec un processeur vocal (En cours d'etude)
DSAVE	Sauve un fichier utilitaire
DLOAD	Charge " "
CLOSE	Ferme un fichier
CLEAR	Change la protection
WCAS	Copie du disque vers une cassette audio
TIME	Utilise avec l'option Horloge temps réel
TEST	Testes les fonctions du DOS
SWAP	Echange le contenue de fichiers
RCAS	Copie d'une cassette Audio vers une disquette
OPEN	Ouvre un fichier
NAME	Nomme la disquette
LOCK	Protege la disquette avec un mot de passe
KILL	Efface un fichier
INIT	Initialisation du systeme
HELP	Aide à l'utilisation (Mode d'emploi condensé)
DISK	Lecteurs en ligne uniquement
DATE	Entrée de la date (jour/mois/année)

K E N - D O S

(Preliminaire)

Vous trouverez ci-après une documentation sur le KEN-DOS dont la plus grande partie est extrait des notes fournies par le constructeur.

Ayant essayé le systeme par moi meme je vous donnerais donc les premieres (et malheureusement trop courtes) impression d'utilisation. Cet article ne tente donc pas de vous donner le maximum de caracteristiques mais quelques informations éparses.

Tout D'abord une presentation physique avec le 'sujet'. Une carte d'environ 15 Cm x 5 Cm se connecte à l'interieur du DAI C'est elle qui contient le DOS et eventuellement d'autres programmes sur EPROM. Ceci est un aspect trés interessant du systeme. Sur celui que j'ai pu tester le traitement de texte etait résident , et croyez moi c'est vraiment agreable à l'emploi. Une fois la carte montée et une modification mineur apportée au DAI vous pourrez alors replacer le 'capot'. Un cable plat permet de relier le bus DCE aux lecteurs. Ceux-ci contiennent la carte controlleur et l'alimentation(qui par la suite devrait etre un bloc séparé) qui se branche sur le secteur. Les lecteurs DCR eventuels se branchent sur le boitier des disques.

Ensuite quelques mots sur son utilisation. A l'allumage de la machine le message 'KEN-DOS Vx.x' s'affiche en lieu et place du traditionnel 'BASIC V1.x'.Detail amusant la couleur du fond n'est plus le traditionnel gris mais ici une sorte de kaki(en peritel) qui ne fatigue pas du tout la vue. Vous avez alors directement accés aux commandes classiques du Basic ainsi qu'a celles du DOS . Pour vous donner une idée aproximative de la vitesse une image en MODE 6 se charge en environ 2 Secondes. Enfin autre detail,lors de la commande DIR il est possible à tout moment de charger (Ou de charger et de demarrer) le programme pointé sur l'ecran par l'action de l'une des touches curseurs.

Pour finir je vous donnerais quelques idées de prix tels qu'ils sont actuellement pratiquées en Belgique (TTC).

Ken-dos + 1 lecteur 200K-octets	:	6500 Frs
+ 2	:	8500 Frs
+ 1 lecteur 400K-octets	:	7000 Frs
+ 2	:	9500 Frs
+ 1 lecteur 800K-octets	:	8000 Frs
+ 2	:	11000 Frs

Voilà, vous trouverez la plupart des details techniques dans les pages suivantes : bonne lecture !

Cedric DUFOUR

K E N - D O S

(D'près une documentation en Anglais publiée par MIPI)

Le Ken-Dos a été conçu spécialement pour repondre à la demande des utilisateurs du DAI qui desirent une grande capacitée de stockage avec un accés rapide et aisé.

Le controleur du Ken-dos peut supporter 4 lecteurs simple ou double densité, 40 ou 80 pistes ou 4 lecteurs double face au maximum. La capacité maximale du systeme est alors portée à 3,2 Mega-octets. Le systeme minimum quand à lui (1 lecteur simple face double densité) est de 200 K-octets.

Le systeme d'exploitation reside entièrement en memoire morte (EPROM). Les avantages de cette methode sont evidents : La plupart des systemes d'exploitations doivent etre chargèe depuis la disquette , consommant ainsi de large plage de memoire (memoire qui semble toujours insuffisante pour l'utilisateur) et une partie du disque . Le Ken-dos quand à lui est toujours prèsent et n'occupe pas un iota de votre memoire utilisateur. De plus vous ne perdez pas de temps à charger le systeme d'exploitation à l'allumage ou au Reset Par consequent, vos disquettes pourront etre entierement consacrée à vos données et/ou programmes. Si vous n'utilisez qu'un seul lecteur il ne sera pas necessaire de constamment échanger les disquettes non plus que de garder constamment quelques copies du systeme d'exploitation.

Un autre avantage du Ken-dos est que la carte de memoire morte commutable qui le contient a été conçue pour supporter beaucoup plus de memoire que necessaire. De ce fait 80 K-octets de memoire morte sont disponibles. Cette possibilitée vous permet de stocker les programmes qui vous servent le plus (Par ex. Traitement de texte ,Assembleur, ...ect). L'accés à ces programmes se fera très facilement grace à la commande BANK du dos.

Pendant l'elaboration du Ken-dos il a été tenue compte du fait que de nombreux utilisateurs possedait déjà un peripherique pour la sauvegarde.
Une option à donc été prévue dans le systeme pour pouvoir lire des cassettes audio ainsi que des DCR. Dans un futur proche il sera aussi possible de lire les disquettes de chez DAI et de sauver leur contenus sur une disquette du Ken-dos . Cette possibilitée sera operationelle grace à une routine utilitaire speciale

Le Ken-dos complet se compose de :
-Une carte de memoire morte commutable . Sur cette carte il y al place pour un maximum de 6 EPROMs. Chaque EPROM peut accepter, grace à des commutateurs, des EPROMS de 2K, 4K, 8K ou 16K. (Soit en tout 96K) . Meme si vous choisissez d'installer le CP/M bios (qui occupe un support) vous aurez encore largement de la place pour les programmes de votre choix.
-Une carte controlleur (à l'exterieur du DAI)qui peut supporter en tout quatre lecteurs simple ou double densités. Sur demande cette carte peut etre modifié pour implanter des lecteurs 8" ou des micro-lecteurs.
-Une alimentation à découpage qui bien que plus couteuse à l'achat , offre une tres grande sécurité d'emploi. Elle permet d'alimenter la carte controlleur et deux lecteurs
-Un dos sur EPROMs, qui offre 43 commandes à l'utilisateur . Celles-ci peuvent etre utilisés au choix en mode direct ou bien dans un programme.

```
Testen op 1 wel of 2 wel :      IF PEEK(#FD00) IAND #30<>0 THEN .....
          (beide mag !)

Testen op een van de events :   P=PEEK(#FD00)
          (beide mag niet !)    IF (P*2 IAND #20) IXOR (P IAND #20)<>0 THEN ....
```

Een volgende toepassing zit in de WAIT instructie . De omschrijving die het hand boek ervan geeft is echter voor de meeste gebruikers ingewikkeld of misschien zelfs onbegrijpelijk, daar ik hem zelden zie gebruiken ook in gevallen waar dat wel had gekund. Er zijn vier mogelijkheden : WAIT I,J,K ; WAIT MEM I,J,K en nog WAIT I,J en WAIT MEM I,J . De laatste twee zijn eigenlijk identiek aan de eerste twee maar nemen K gelijk aan nul. Vergelijk met de FOR ... TO ... STEP ... waarbij de STEP ook niet opgegeven hoeft te worden ; bij weglaten wordt als defaultwaarde een genomen . De instructies WAIT en WAIT MEM zijn in verwerking gelijk ; het verschil zit in I : bij WAIT is I een real world input port en bij WAIT MEM een geheugenadres. Dit laatste kan via de memory-mapped I/O van de DAI toch weer een extern device zijn. Wat doen eigenlijk J en K ? Welnu de opgegeven byte (I) wordt ge'IXOR'd met K en dan wordt dit resultaat ge'IAND' met J . Als het resultaat nu gelijk is aan J wordt het programma normaal vervolgd . Duidelijk ? ? ? ? Mij niet, toen ik het de eerste keer las. Nog maar eens proberen : We nemen voor het gemak eerst de defaultwaarde nul voor K. Als byte I met nul ge'IXOR'd wordt zal hij niet veranderen. Als we nu met J 'IAND'en zal het resultaat alleen dan gelijk aan J zijn als de bits die bij J een zijn dat ook in het tussenresultaat zijn. De overige bits die in het tussenresultaat misschien een zijn hebben geen invloed op de WAIT test . Hiermee kan dus getest worden of de bepaalde bits in I aanstaan . Dus met WAIT MEM #FD00,#10 wachten we tot eventbutton 1 ingedrukt wordt, vanzelfsprekend wachten we met WAIT MEM #FD00,#20 op het indrukken van eventbutton 2 en met WAIT MEM #FD00,#30 wachten we op beide. Maar nu de 'IXOR' met K. We gaan na wat er met de verschillende bits gebeurt, zie volgend schema :

nr.	I	K	(I IXOR K)	J	(I IXOR K) IAND J
1	0	0	0	0	0
2	0	0	0	1	0
3	0	1	1	0	0
4	0	1	1	1	1
5	1	0	1	0	0
6	1	0	1	1	1
7	1	1	0	0	0
8	1	1	0	1	0

We zien nu dat het resultaat alleen dan gelijk is aan J in de enen in geval vier en zes. Dit is in het geval zes als de te onderzoeken bit in I een is en in J ook terwijl de bit in K dan nul moet zijn. Bij geval vier is de bit in I nul en in J een terwijl de bit in K dan een is.
Nog anders gezegd : als we byte I moeten onderzoeken op bepaalde bits dan moeten we de bits die onderzocht moeten worden een maken in onze J en nul in K als het een moet zijn in I en een in K als het een nul moet zijn. J selecteert om het zo te zeggen de te onderzoeken bits en in K staat of nul of een moeten zijn . Om te wachten totdat eventbutton 1 ingedrukt wordt maar eventbutton 2 niet doen we dus de wachtinstructie WAIT MEM #FD00,#30,#20 en omgekeerd WAIT MEM #FD00,#30,#10.

We gaan verder met de INOT. Volgens de theorie zou elke een nul en elke nul een moeten worden. Dit is ook zo maar als we niet weten hoe de binaire representatie van getallen in de DAI geschiedt kunnen we toch moeilijkheden verwachten . Begin met POKE 6,#0F en POKE 7,(INOT #F0). Pas op; gebruik 6 en 7 om te POKE'n; andere adressen mogen wel maar 6 en 7 zijn 'normaal' veilig. (5 en 8 beslist niet) De tweede POKE geeft hier een NUMBER OUT OF RANGE. De POKE kan maximaal een byte aan en daar kan maximaal 255 in. De (INOT #F0) is niet zoals we verwachtten #0F

maar #FFFFFF0F omdat er een integer van gemaakt wordt die vier bytes lang is. En de waarde #FFFFFF0F kan echt niet in een byte. We proberen het nogeens nu met de POKE 7,(INOT #FFFFFF0F). Dit lukt en nu naar UT om met bv D 22 te zien wat er nu op 7 staat. Op sommige machines zal er nu F0 staan op andere F2, de eerste is correct de tweede niet. De oorzaak ligt in de AMD 9511 math. chip die INOT fout behandeld. Jan Boerrigter waarschuwde hier reeds voor in de firmware . Als U dus over een AMD 9511 in uw machine beschikt kunt U de INOT op twee manieren gebruiken; correct door eerst POKE #D4,0 te geven of fout maar wel consequent na aanzetten van de math.chip door POKE #D4,123 (of reset). We kijken naar de praktijk en doen met math.chip uit ?INOT 5 en krijgen -6, tenminste als U IMP INT gegeven hebt, anders een foutmelding !!!! De DAI zet de floating point waarde hier niet zelf om in integer. ?INOT 0 levert -1 en ?INOT (-8) geeft 7 . Het komt er dus op neer, dat eerst bij het argument 1 wordt opgeteld en dan het tegengestelde genomen wordt. Nu de math. chip aanschakelen. ?INOT 5 geeft -4, ?INOT 0 geeft 1 en ?INOT (-8) geeft nu 9. Het komt er nu op neer dat eerst het tegengestelde van het argument genomen wordt en dat er dan 1 bij wordt opgeteld. Het verschil tussen math. chip aan of uit is dus altijd 2. Maar pas op: INOT (INOT 6) levert wel altijd 6, math. chip aan of uit maakt nu niet meer uit. De syntaxcontrole is bij INOT overigens ook niet correct. We krijgen geen protest als we ?5 INOT intikken en dat kan knap lastig zijn.

De laatste operatoren die nog niet behandeld werden zijn de SHL en de SHR. Degenen die weleens in assembler hebben gewerkt, zullen de afkortingen wel begrijpen maar voor de anderen : SHL staat voor SHift Left en SHR voor SHift Right. Deze operatoren werken niet op bits maar op groepen bits. In assembler op een groepje van 8 of 9 bits die dan zelfs rond geschoven worden (RLC, RRC, RAL en RAR) en in BASIC op een integer dus vier bytes. Bij G SHL M worden alle bits van G M plaatsen naar links geschoven en bij G SHR N worden al de bits van G N plaatsen naar rechts geschoven. De beide bewerkingen kunt U zich voorstellen als een zogenaamd 'window'; op een lijn staan oneindig veel nullen en ergens op die lijn staat het getal dat u bekijkt door een raam ('window') zodat U maar 32 bits (4 bytes) tegelijk kunt zien. Door de SHL en SHR kunt U nu dit raam verschuiven. Maar pas op de vergelijking is gemaakt om U te laten zien dat er enen verdwijnen en nullen bijkomen, de richting is nu verwarrend. Bij SHL schuift het getal naar links dus lijkt ons raam naar rechts te schuiven; bij SHR is dat natuurlijk analoog . Maar hoe kunnen we SHL en SHR nu handig gebruiken ?
Het meest voor de hand liggende gebruik is het omwerken van bitposities naar getalwaarden. Als we zeggen A=PEEK(#FD00) IAND #30 en dan B=A SHR 4 zal B afhankelijk van de ingedrukte eventbuttons 0,1,2 of 3 zijn . De 0 betekent dat er geen eventbutton is ingedrukt, 1 en 2 betekent dat een van de eventbuttons ingedrukt wordt en 3 dat beide worden ingedrukt. Ook hier is weer een waarschuwing op zijn plaats: A=PEEK(#FD00) IAND #30 SHR 4 zal nooit het gewenste resultaat geven maar wel een getal van drie of kleiner. Ik vermoed overigens dat het altijd 3 is maar kan dat niet nagaan. (wie informeert mij ?) A=(PEEK(#FD00) IAND #30) SHR 4 zal wel correct werken . De oplossing van dit probleem zit in de volgorde van de bewerkingen. De SHR heeft een hogere prioriteit dan de IAND en zal de DAI beginnen met het uitrekenen van #30 SHR 4, dit levert 3 op . Vervolgens zal de inhoud van #FD00 ge'IAND' worden met 3. Bij mij waren bij al de tests bit nul en een steeds een en was het resultaat dus steeds 3 . Voor zover mij bekend worden de bits nul en een op #FD00 nergens voor gebruikt en daarom weet ik ook niet wat ze kan veranderen. NB A=PEEK(#FD00) SHR 4 IAND 3 zal ook goed werken.
Een andere toepassing vinden we in berekeningen. Als de bits van een integer allemaal een naar links worden geschoven zal het getal tweemaal zo groot worden en schuiven we meerdere bits naar links zal het getal net zo vaak verdubbelen als er bits naar links werd geschoven. Getallen worden zo snel te groot, maar hebben we een vermenigvuldiging met 16 nodig is het misschien nuttig te weten dat SHL 4 hetzelfde effect heeft en sneller werkt. Voor vermenigvuldiging van A met 2 kunnen we het best A+A nemen.
Een nog niet besproken toepassing van de bitoperaties ligt bij grafische 'trucs' en die wil ik later nog eens apart behandelen.

Frank H. Druijff

programmeertechnieken

Het onderwerp van deze keer is zoals aangekondigd de verschillende logische operatoren die de DAI-BASIC kent. De functies zullen later ook nog eens aan bod komen. Er zijn er veel die wel wat nadere toelichting behoeven . Weet U soms iets bijzonders te vertellen over een van de operatoren of functies meldt mij dit dan alstublieft. Ik zal het dan weer doorgeven aan alle DAInamic-lezers.

Nu de bit-operators IAND, IOR, IXOR, SHL, SHR en INOT. De laatste is een bijzonder geval omdat hij maar een argument heeft en de andere er twee hebben . In het handboek staan ze echter bij elkaar en dat is vermoedelijk gedaan om alle bitbewerkingen bij elkaar te houden. Wilfried Hermans heeft een programma geschreven dat de bit operators demonstreert; dit programma is gepubliceerd in DAInamic 12. In dit programma kunnen we de bit-patronen goed zien en de werking van de verschillende mogelijkheden beter leren begrijpen. Jammer is alleen dat een echte bitbewerking als INOT er niet bij staat en de MOD wel terwijl dat geen bitbewerking is.

We beginnen met met IAND,IOR en IXOR omdat ze, alhoewel ze verschillende uitkomsten geven, wel dezelfde structuur hebben. De woorden beginnen allemaal met een I om aan te geven dat ze alleen voor integers (beter gezegd vier bytes enen en nullen daar het signbit ook meedoet) gebruikt worden. Ze kunnen wel met floating point getallen gebruikt worden, maar die worden of toch eerst omgezet in integer of als we byte voor byte werken zo veranderd dat de resultaten zelden zinvol zul len zijn.

AND betekent EN, OR betekent OF en XOR betekent eXclusieve OF, alle in de logische betekenis. Hiervoor nog wat uitleg die ik ook bij mijn wiskundelessen gebruik: EN en OF zijn in het dagelijks spraakgebruik niet duidelijk in betekenis. Lees de volgende twee verhaaltjes maar en denk na over de gevolgtrekkingen die we dan kunnen maken.

I) Jan wil naar de bioscoop en laten we zeggen dat dit tien gulden kost . Als hij zonder problemen naar binnen wil zal hij dus een toegangskaartje moeten hebben 'OF' een tientje om dat te kopen. Lees de zin zonder overdreven nadruk op 'OF'. Iedereen er nu mee eens ? Goed Jan naar de bioscoop, hij heeft zelfs twintig gulden bij zich . Hij koopt een kaartje en wordt tot zijn stomme verbazing bij de ingang tegengehouden. Hij heeft toch een kaartje 'EN' een tientje bij zich? Na wat nadenken begrijpen we dat 'OF' hier betekende het een, het ander maar ook beide zijn toegestaan.

II) De volgende dag bezoekt Jan de verjaardag van zijn tante die hem op zeker moment de taartdoos voorhoudt en hem vraagt : "Wil je een vruchtengebakje OF een slagroomgebakje ? " Jan herinnert zich de bioscoop nog en denkt: " Beide mag ook." en grijpt met beide handen een gebakje. En nu is het weer niet goed !!

Het zal U nu hopelijk wel duidelijk zijn dat het spraakgebruik niet erg precies is met 'OF'. Ook voor 'EN' kunnen we voorbeelden geven die duidelijk maken hoe onduidelijk het is. Denk maar eens aan garanties voor auto's; ' 1 jaar of 20000 km' zegt de ene garagist terwijl de andere '1 jaar en 20000 km' zegt. Beide bedoelen vermoedelijk 'max 1 jaar en max 20000 km'; dwz aan beide eisen (auto minder dan 1 jaar oud / kilometrage minder dan 20000) moet voldaan worden om voor garantie in aamerking te komen.
In de programmering kunnen we dergelijke onduidelijkheden niet tolereren. In het handboek staan niet voldoende richtlijnen om er uit te komen. Over de IXOR wordt zelfs helemaal niets verteld. De bit-operaties kunnen echter uit de volgende tabellen begrepen worden.

IAND	0	1
0	0	0
1	0	1

IOR	0	1
0	0	1
1	1	1

IXOR	0	1
0	0	1
1	1	0

INOT	0	1
-	1	0

Onthoudt U de tabellen maar aan de hand van de volgende regels:
IAND --- beide moeten waar (= 1) zijn
IOR ---- minstens een moet waar (= 1) zijn
IXOR --- precies een moet waar (= 1) zijn
om het resultaat waar (= 1) te krijgen. In alle andere gevallen is het onwaar (= 0). Bij INOT verandert simpel elke 1 in een 0 en elke 0 in een 1.
Speelt U met deze kennis maar eens met het programma van Wilfied Hermans en snel zult U de resultaten correct kunnen voorspellen.

De IAND en de IOR bestaan ook in een logische versie, de IXOR (en INOT) vreemd genoeg niet. Ook ontgaat het mij waarom er verschillende woorden voor worden gebruikt.

De toepassing van deze bitbewerkers is meestal te vinden in input / output handelingen bij gespecialiseerde programma's . Een eenvoudige toepassing die bijna elke spelletjesfanaat kent is het controleren op de eventknoppen. Als de eventbutton rechts wordt ingedrukt, wordt bit 4 op adres #FD00 1 en bit 5 wordt 1 als de linker eventbutton wordt ingedrukt. Ze worden vanzelfsprekend beide 1 als ook beide knoppen worden ingedrukt. Als alleen bit 4 aanstaat ziet #FD00 er dus in bitnotatie als volgt uit : 00010000 (=#10) en bij bit 5 00100000 (=#20) en bij 4 en 5 00110000 (=#30). Pas op ; niet 'normaal' tellen van links naar rechts met 1,2,3,.... maar juist van rechts naar links met 0,1,2,.... . Maar de andere bits van #FD00 kunnen ook 1 zijn als gevolg van de zogenaamde memory mapped I/O , die de DAI kent . We kunnen dus niet simpel doen: IF PEEK(#FD00)=#10 THEN ....., omdat dat alleen maar goed gaat als de andere bits van #FD00 nul zijn. We zullen de te onderzoeken bit ('s) eerst moeten uitselecteren. Dit kan op vele manieren, ik zal er enkele noemen; probeert U er zelf eens andere te vinden.
Bij PEEK(#FD00) IAND #10 krijgen we alleen bit 4 in het resultaat en als dat dan nul is wordt de rechterknop niet ingedrukt, is hij niet nul en dus 16 (=#10) dan is de knop wel ingedrukt. Met IAND #20 krijgen we bit 5 en kunnen we zien of de linkereventbutton is ingedrukt en met #30 testen we beide tegelijk. In het laatste geval testen we met ongelijk nul alleen op het indrukken van linkerevent of rechterevent met 'of' in de juiste logische betekenis , dus ook beide ingedrukt houden heeft effect. Willen we echter meer specifiek op de events testen bv. een wel maar ander niet of precies een van de eventbuttons ingedrukt of beide tegelijk ingedrukt kunnen we met behulp van de net gegeven mogelijkheden iets doen :

```
Testen op 1 (2 onbepaald) :      IF PEEK(#FD00) IAND #10=#10 THEN .....

Testen op 2 (1 onbepaald) :      IF PEEK(#FD00) IAND #20=#20 THEN .....

Testen op 1 niet (2 onbepaald): IF PEEK(#FD00) IAND #10=0 THEN .....

Testen op 2 niet (1 onbepaald): IF PEEK(#FD00) IAND #20=0 THEN .....

Testen op 1 wel en 2 niet :      IF PEEK(#FD00) IAND #30=#10 THEN .....

Testen op 2 wel en 1 niet :      IF PEEK(#FD00) IAND #30=#20 THEN .....

Testen op 1 wel en 2 wel :       IF PEEK(#FD00) IAND #30=#30 THEN .....

Testen op 1 niet en 2 niet :     IF PEEK(#FD00) IAND #30=0 THEN .....
```

```
CALL #F7C0,C$   F7C0  FE 22      CPI     22      ENTRY FOR CALLED COMMANDS
                F7C2  C0         RNZ             PARAMETER MUST BE STRING
                F7C3  C5         PUSH    B       SAVE BASIC POINTER
                F7C4  5E         MOV     E,M     GET ADDR. OF THE
                F7C5  23         INX     H       STRINGVARIABLE WITH
                F7C6  56         MOV     D,M     THE DOS COMMAND
                F7C7  EB         XCHG
                F7C8  7E         MOV     A,M     GET LENGTH OF STRING
                F7C9  32 34 01   STA     0134    SET IT AS INPUT-LENGTH
                F7CC  23         INX     H
                F7CD  22 32 01   SHLD    0132    COMMAND-STRING-ADDR.
                F7D0  3E 01      MVI     A,01    SET INPUTSWITCH FOR
                F7D2  32 35 01   STA     0135    INPUT FROM STRING
                F7D5  21 00 00   LXI     H,0000
                F7D8  22 71 02   SHLD    0271    DISABLE COMMAND EXIT
                F7DB  39         DAD     SP
                F7DC  22 73 02   SHLD    0273    SAVE CURRENT STACK-P.
                F7DF  0E 00      MVI     C,00    RESET CURSOR FOR INPUT
                F7E1  CD 79 F0   CALL    F079    EXECUTE DOS-COMMAND
-------------------------------------------------------------------------------
EXTENDED EXIT   F7E4  D1         POP     D       RETURN FROM DOS-COMMAND
                F7E5  3E F8      MVI     A,F8
                F7E7  F9         SPHL             RESTORE STACKPOINTER
                F7E8  BC         CMP     H       IF IN COMMAND-MODE,
                F7E9  CA 51 F0   JZ      F051    THEN NORMAL DOS-EXIT
                F7EC  2A 73 02   LHLD    0273    RESTORE STACKPOINTER IF
                F7EF  F9         SPHL            CALLED COMMAND FROM BASIC
                F7F0  AF         XRA     A
                F7F1  32 34 01   STA     0134    RESET INPUT FROM
                F7F4  32 35 01   STA     0135    STRING-VARIABLE
                F7F7  C1         POP     B       RESTORE BASIC-POINTER
                F7F8  3E A0      MVI     A,A0    IF THE COMMAND WAS
                F7FA  BB         CMP     E       NOT A LOAD, THEN RETURN
                F7FB  C0         RNZ             TO THE BASIC PROGRAM
                F7FC  C3 B7 F7   JMP     F7B7    IF LOAD, SO EXECUTE THE
                F7FF  FF         DATA    FF      RUN COMMAND

-------------------------------------------------------------------------------
----------------------    END  OF  PROM    ------------------------------------
-------------------------------------------------------------------------------
```

```
-------------------------------------------------------------------------------
*******            D I S K   O R G A N I S A T I O N            *******
-------------------------------------------------------------------------------

HARDWARE: TEAC FC-50 CONTROLLER WITH TEAK FD-50F DRIVE

5.25 INCH, DOUBLE SIDED FLOPPIES WITH  80 TRACKS / SIDE

96 TPI, SINGLE DENSITY RECORDING TECHNIQUE (FM)

TYPE   NUMBER OF SECTORS   SECTORLENGTH  TRACK-CAPACITY
  0           16               128           2048
  1            9               256           2304
  2            5               512           2560

PROGRAM-DISK ORGANISATION:

SIDE TRACK   CONTENT
  0    00    DIRECTORY
  1    00    DIRECTORY BACKUP
  0    01    >
    .         >
    .          >
  0    4F      >
                >  PROGRAMS AND DATA
  1    01      >
    .          >
    .           >
  1    4F      >


DIRECTORY LAYOUT: 4 BLOCKS AT 32 BYTES PER SECTOR
                  THE FIRST BLOCK IS FOR DISK-IDENTIFICATION
                  63 BLOCKS FOR PROGRAM OR DATA IDENTIFICATION
```



```
-------------------------------------------------------------------------------
*******            D I S K   D I R E C T O R Y                  *******
-------------------------------------------------------------------------------

LOCATION        USAGE

00              0C CHARACTER FOR CLEAR SCREEN
01 - 17         23 CHARACTER FOR DISK-IDENTIFICATION
18 - 1F         FREE

00 - 01         42 20 (='B ') DIRECTORY-BLOCK FOR A BASIC-PROGRAM
02 - 17         22 CHARACTERS FOR PROGRAM-IDENTIFICATION
18 - 19         SIZE OF HEAP
1A - 1B         SIZE OF PROGRAM
1C - 1D         SIZE OF VARIABLE-TABLE
1E              START TRACK  ( 01 = TRK 1  SIDE 0 )
1F              END TRACK    ( 81 = TRK 1  SIDE 1 )

00 - 01         4D 20 (='M ') DIRECTORY-BLOCK FOR MEMORY-SAVE
02 - 17         22 CHARACTERS FOR DATA-IDENTIFICATION
18 - 19         FREE
1A - 1B         FROM LOCATION  (SAME AS W IN UTILITY-MODE)
1C - 1D         TO LOCATION
1E              START TRACK  ( 01 = TRK 1  SIDE 0 )
1F              END TRACK    ( 81 = TRK 1  SIDE 1 )

00 - 1F         E5 E5 E5 E5   FREE ENTRY

-------------------------------------------------------------------------------
*******            M E M O R Y   A L L O C A T I O N            *******
-------------------------------------------------------------------------------

LOCATION        USAGE

0038 - 004D     READ/WRITE ROUTINE DURING DISK-I/O
                ORIGINAL CONTENTS WILL BE SAVED AND REPLACED
013E - 01BD     DIRECTORY-INPUT-BUFFER, SAME LOCATION AS
                THE DAI INPUT-BUFFER OF 128 BYTES
0248 - 025F     SAVED LOCATIONS #38 - #4F DURING DISK-I/O
                BETWEEN I/O USABLE AS ENVELOPE STORAGE.
                DISK-I/O WILL DESTROY CURRENT ENVELOPE.
-------------------------------------------------------------------------------
0260 - 0270     DISK WORK AREA
   0260         DEVICE SELECTION
   0261         CURRENT STATUS (COMMAND MODE)
   0262         CURRENT COMMAND
   0263         CURRENT DATA IN/OUT
   0264         CURRENT TRACK
   0265         CURRENT SECTOR
   0266         TRACK PRESET
   0267         SECTOR PRESET
0268 - 0269     BUFFER-POINTER AT START OF READ
026A - 026B     BUFFER-POINTER AFTER READ
026C - 026D     BUFFER-POINTER AT START OF WRITE
026E - 026F     BUFFER-POINTER AFTER WRITE
   0270         I/O-STATUS AFTER READ OR WRITE
-------------------------------------------------------------------------------
0271 - 0272     SAVED STACK POINTER DURING DOS-COMMAND
0273 - 0274     SAVED STACK POINTER DURING DOS-BASIC-CALLS
F000 - F7FF     DOS-ROM

-------------------------------------------------------------------------------
*******            U S A B L E   E N T R Y P O I N T S          *******
-------------------------------------------------------------------------------

   F0F0    X    DESELECT DISK-DRIVE AND MOTOR OFF
   F100    X    PUT DISK-STATUS      TO   #261
   F10F    X    PUT DATA    REGISTER TO   #263
   F11E    X    PUT TRACK   REGISTER TO   #264
   F12D    X    PUT SECTOR  REGISTER TO   #265
   F13C    X    DISK MOTOR ON, NO CHECKS
   F147    X    SET DEVICE  REGISTER FROM #260  (SELECT)
   F153    X    SET DATA    REGISTER FROM #263
   F15F    X    SET SECTOR  REGISTER FROM #267
   F16B    X    SET TRACK   REGISTER FROM #266
   F177    X    SET COMMAND REGISTER FROM #262
   F190         INIT DCE-BUS TO MODE 0, A-IN,B-OUT,C-IN
   F227    X    READ FROM DISK
   F26C    X    WRITE TO DISK
   F334         RESTORE DISK TO TRACK 00 WITH CHECK
   F34A         SEEK TO TRACK, A=DEVICE BITS,D=TARGET TRACK
   F7C0    X    COMMAND VIA BASIC: C$="LOAD 23"+CHR$(13):CALLM #F7C0,C$
                |
                +--> ALL REGISTER ARE SAVED AND RESTORED

-------------------------------------------------------------------------------
```

```
                F664  EB          XCHG
                F665  2A A1 02    LHLD   02A1    START OF VARIABLE TABLE
                F668  CD 1A DE    CALL   DE1A    SUBSTRACT TO GET SIZE
                F66B  D1          POP    D
                F66C  CD 64 F5    CALL   F564    PUTADR FOR REL. PRG. SIZE
                F66F  E5          PUSH   H       SAVE BUFFERPOINTER (DIR)
                F670  2A 9B 02    LHLD   029B
                F673  EB          XCHG
                F674  2A A3 02    LHLD   02A3    END OF VARIABLE TABLE
                F677  CD 1A DE    CALL   DE1A    SUBSTRACT TO GET PRG.SIZE
                F67A  D1          POP    D       GET BUFFERPOINTER (DIR)
                F67B  CD 64 F5    CALL   F564    PUTADR REL. PRG.SIZE+TABLE
                F67E  13          INX    D
                F67F  CD 14 F7    CALL   F714    NTRK CALL VIA BUFFERSET
                F682  2A 9B 02    LHLD   029B    BUFFERADDR=PRG.START
                F685  1C          INR    E       INCR. ENDTRACKNUMBER
          +---> F686  D5          PUSH   D       AND SAVE START AND END
          |     F687  22 6C 02    SHLD   026C    SET BUFFER FOR DATA
          |     F68A  CD 42 F5    CALL   F542    SIDESEL
          |     F68D  3E A0       MVI    A,A0    WRITE COMMAND
          |     F68F  32 62 02    STA    0262    TO COMMAND AREA
          |     F692  CD 6C F2    CALL   F26C    MULTISECTOR-WRITE
          |     F695  2A 6E 02    LHLD   026E    GET CURRENT BUFFERPOSITION
          |     F698  D1          POP    D
          |     F699  14          INR    D       NEXT TRACK
          |     F69A  7A          MOV    A,D
          |     F69B  BB          CMP    E       LAST TRACK REACHED ?
          |  +--  F69C  CA A9 F6  JZ     F6A9    YES JUMP
          |  |   F69F  FE 50       CPI    50      END OF SIDE 0 ?
          +--|-- F6A1  C2 86 F6    JNZ    F686    NO, JUMP
          |  |   F6A4  16 81       MVI    D,81    YES, CHANGE TO SIDE 1
          +--|-- F6A6  C3 86 F6    JMP    F686    AND CONTINUE SAVE
          |  +-> F6A9  CD 11 F4    CALL   F411    GLOBAL I/O-TEST
          |     F6AC  C3 EE F3    JMP    F3EE    RESTORE AND DISK-OFF
          |     F6AF  FF
----------|-----------------------------------------------------------
@W        |     F6B0  F5          PUSH   PSW     SAVE MEMORY
          |     F6B1  C5          PUSH   B
          |     F6B2  D5          PUSH   D
          |     F6B3  E5          PUSH   H
          |     F6B4  CD BF F5    CALL   F5BF    FIND FREE SPACE IN DIR
          |     F6B7  36 4D       MVI    M,4D    FLAG FOR MEMORY-SAVE M
          |     F6B9  E5          PUSH   H       RETAIN DIR-POINTER
          |     F6BA  CD 5D F3    CALL   F35D    HEXIN FOR FROM-VALUE
          |     F6BD  EB          XCHG
          |     F6BE  22 6C 02    SHLD   026C    SAVE IT
          |     F6C1  CD 5D F3    CALL   F35D    HEXIN FOR TO-VALUE
          +-----+ F6C4  EB          XCHG
          |     F6C5  22 6E 02    SHLD   026E    SAVE IT
          |     F6C8  E1          POP    H       GET DIR-POINTER BACK
          |     F6C9  0D          DCR    C       DECREMENT INPUT-STRING
          |     F6CA  1E 19       MVI    E,19    GET MAX. 25
          |  +-> F6CC  23          INX    H       CHARACTER OF
          |  |   F6CD  CD E0 DD    CALL   DDE0    DOCUMENTATION-TEXT
          |  |   F6D0  FE 0D       CPI    0D      UNTIL CR AND
          |  +-- F6D2  CA DF F6    JZ     F6DF    STORE IT IN
          |  |   F6D5  77          MOV    M,A     THE DIR-BUFFER
          |  |   F6D6  0C          INR    C
          |  |   F6D7  1D          DCR    E
          |  +-- F6D8  C2 CC F6    JNZ    F6CC
          |  +-- F6DB  C3 E5 F6    JMP    F6E5    ALL TEXTSPACE USED
          |  |  +-> F6DE  23          INX    H
          +--|--|-> F6DF  36 20       MVI    M,20    PAD REMAINDER WITH BLANKS
          |  |   F6E1  1D          DCR    E
          |  +-- F6E2  C2 DE F6    JNZ    F6DE
          |  +-> F6E5  EB          XCHG
          |     F6E6  2A 6C 02    LHLD   026C    GET "FROM" VALUE
          |     F6E9  CD 64 F5    CALL   F564    PUTADR
          |     F6EC  EB          XCHG
          |     F6ED  2A 6E 02    LHLD   026E    GET "TO" VALUE
          |     F6F0  CD 64 F5    CALL   F564    PUTADR
          |     F6F3  E5          PUSH   H       SAVE CURRENT DIR-POINTER
          |     F6F4  2A 6C 02    LHLD   026C    GET "FROM" VALUE AGAIN
          |     F6F7  EB          XCHG
          |     F6F8  CD 1A DE    CALL   DE1A    SUB FOR NUMBER OF BYTES
          |     F6FB  EB          XCHG           TO BE SAVED
          |     F6FC  13          INX    D       NUMBER OF BYTES IN D
          |     F6FD  2A 6C 02    LHLD   026C    SAVE THE "FROM" VALUE
          |     F700  22 6A 02    SHLD   026A    TO THE FREE LOCATION
          |     F703  21 3E 01    LXI    H,013E  REPLACE OUTPUTBUFFER-ADDR
          |     F706  22 6C 02    SHLD   026C    BY THE DIR-ADDR
          |     F709  E1          POP    H       GET CURRENT DIR-POINTER
          |     F70A  CD 6A F5    CALL   F56A    NTRK
          |     F70D  1C          INR    E       ENDTRACKNUMBER+1
          |     F70E  2A 6A 02    LHLD   026A    GET SAVED "FROM" VALUE
          +---------- F711  C3 86 F6    JMP    F686    CONTINUE AS BY @SAVE
```



```
----------------------------------------------------------------------
NTRK+BUFFER     F714  E5          PUSH   H       NTRK VIA BUFFERSET
                F715  21 3E 01    LXI    H,013E  SET DIR-BUFFER ADDR.
                F718  22 6C 02    SHLD   026C    IN DISK-AREA
                F71B  E1          POP    H
                F71C  CD 6A F5    CALL   F56A    NTRK
                F71F  C9          RET
----------------------------------------------------------------------
@TN             F720  3A 40 00    LDA    0040    TAPE ON
                F723  E6 CF       ANI      CF
                F725  32 40 00    STA    0040
                F728  C9          RET
----------------------------------------------------------------------
@TF             F729  3A 40 00    LDA    0040    TAPE OFF
                F72C  F6 30       ORI      30
                F72E  32 40 00    STA    0040
                F731  C9          RET
----------------------------------------------------------------------
@Z              F732  21 78 D5    LXI    H,D578  RESET DOS
                F735  22 6E 00    SHLD   006E
                F738  CD 90 F1    CALL   F190
                F73B  E1          POP    H
                F73C  C3 0C C8    JMP    C80C
----------------------------------------------------------------------
@C              F73F  3E 5F       MVI    A,5F    RESET CURSOR
                F741  32 75 00    STA    0075
                F744  C9          RET
----------------------------------------------------------------------
@E              F745  3E 02       MVI    A,02    READ FROM EDITBUFFER
                F747  32 35 01    STA    0135    =POKE #135,#2
                F74A  C9          RET
----------------------------------------------------------------------
@M              F74B  2A 9F 02    LHLD   029F    PREPARE FOR MERGE OF
                F74E  22 9B 02    SHLD   029B    BASIC PROGRAMS
                F751  21 00 01    LXI    H,0100
                F754  22 9D 02    SHLD   029D
                F757  C3 B8 DE    JMP    DEB8
----------------------------------------------------------------------
@P              F75A  21 EC 02    LXI    H,02EC  RESET HEAP-POINTER
                F75D  22 9B 02    SHLD   029B    =POKE #29B,#EC
                F760  C9          RET             POKE #29C,#02
----------------------------------------------------------------------
@BS             F761  2A EC 03    LHLD   03EC    SAVE HEAP-POINTERS
                F764  22 45 00    SHLD   0045
                F767  06 0A       MVI    B,0A
                F769  21 9B 02    LXI    H,029B
                F76C  11 47 00    LXI    D,0047
                F76F  C3 AE F1    JMP    F1AE
----------------------------------------------------------------------
@BL             F772  2A 45 00    LHLD   0045    RESTORE HEAP-POINTERS
                F775  22 EC 03    SHLD   03EC
                F778  06 0A       MVI    B,0A
                F77A  11 9B 02    LXI    D,029B
                F77D  21 47 00    LXI    H,0047
                F780  C3 AE F1    JMP    F1AE
----------------------------------------------------------------------
@R              F783  3E D2       MVI    A,D2    RUN XXX WITHOUT CLEAR OF
                F785  32 D2 00    STA    00D2    VARIABLES
                F788  CD 24 C0    CALL   C024    GET AND DECODE
                F78B  21 3E 01    LXI    H,013E  LINENUMBER AS INTEGER
                F78E  E7          RST    4       FOR THE GOTO-COMMAND
                F78F  0F          DATA   0F
                F790  23          INX    H
                F791  36 89       MVI    M,89    INSERT GOTO-CODE
                F793  21 00 00    LXI    H,0000
                F796  22 42 01    SHLD   0142    SET COMMAND-END
                F799  01 3F 01    LXI    B,013F  LOAD COMMAND-ADDR.
             +-- F79C  C3 A6 F7    JMP    F7A6    CONTINUE AS FOR @0
-------------|--------------------------------------------------------
@0           |  F79F  9A FF 00 00       BASIC MODE 0 COMMAND
             |  F7A3  01 9F F7    LXI    B,F79F  COMMAND-ADDR. FOR MODE 0
             +-> F7A6  3A 40 00    LDA    0040    CHECK IF IN BASIC-MODE
                 F7A9  E6 C0       ANI      C0    IT IS ?
                 F7AB  C2 8D F0    JNZ    F08D    NO, --> SYNTAX-ERROR
                 F7AE  21 00 F0    LXI    H,F000  RE-ENABLE
                 F7B1  22 6E 00    SHLD   006E    DOS-COMMANDS
                 F7B4  C3 92 C8    JMP    C892    EXECUTE BASIC COMMAND
----------------------------------------------------------------------
RUN              F7B7  01 BD F7    LXI    B,F7BD  COMMAND-ADDR. OF RUN
                 F7BA  C3 92 C8    JMP    C892    EXECUTE BASIC COMMAND
                 F7BD  87 00 00           BASIC RUN COMMAND
----------------------------------------------------------------------
```

```
                F512  CD 11 F4  CALL   F411   CHECK I/O
                F515  14        INR    D
                F516  7A        MOV    A,D    NEXT TRACK
                F517  BB        CMP    E
                F518  CA EE F3  JZ     F3EE   STOP IF LAST TRACK
                F51B  FE 50     CPI      50   END OF SIDE 0 ?
             +--- F51D  C2 FF F4  JNZ    F4FF   CONT. INPUT AT NEXT TRACK
             |  F520  3E 81     MVI    A,81
             |  F522  57        MOV    D,A    CONT. INPUT AT
             +--- F523  C3 FF F4  JMP    F4FF   TRACK 1 / SIDE 1
-------------------------------------------------------------------------------
MEMORY-READ     F526  FE 4D     CPI      4D   MEMORY-READ ?
                F528  C2 F8 F3  JNZ    F3F8   EXIT IF NOT
                F52B  7B        MOV    A,E
                F52C  C6 1A     ADI      1A
                F52E  5F        MOV    E,A    GET MEMORY-POINTERS
                F52F  EB        XCHG           FROM DIRECTORY-TABLE
                F530  5E        MOV    E,M
                F531  23        INX    H
                F532  56        MOV    D,M
                F533  23        INX    H
                F534  23        INX    H
                F535  23        INX    H
                F536  EB        XCHG
                F537  22 68 02  SHLD   0268   INPUT BUFFER = MEMORY-
                F53A  EB        XCHG           START-LOCATION
                F53B  56        MOV    D,M    START TRACK IN REG. D
                F53C  23        INX    H
                F53D  5E        MOV    E,M    END TRACK IN REG. E
                F53E  1C        INR    E
                F53F  C3 02 F5  JMP    F502   CONTINUE AT NEXT-TRACK
-------------------------------------------------------------------------------
SIDESEL         F542  7A        MOV    A,D
                F543  E6 80     ANI      80   WITCH SIDE ?
             +--- F545  CA 51 F5  JZ     F551   SIDE 0
             |  F548  7A        MOV    A,D    SIDE 1
             |  F549  E6 7F     ANI      7F   MASK FOR CORRECT TRACKNR.
             |  F54B  57        MOV    D,A
             |  F54C  3E 94     MVI    A,94   SELECT SIDE 1
             |  F54E  C3 53 F5  JMP    F553
             +-> F551  3E 84     MVI    A,84   SELECT SIDE 0
                 F553  CD 4A F3  CALL   F34A   SEEK
                 F556  3E 01     MVI    A,01
                 F558  32 67 02  STA    0267   SET STARTSECTOR = 1
                 F55B  3E 05     MVI    A,05
                 F55D  32 65 02  STA    0265   SET ENDSECTOR = 5
                 F560  CD 5F F1  CALL   F15F   SET SECTOR-REGISTER IN FDC
                 F563  C9        RET
-------------------------------------------------------------------------------
PUTADR          F564  EB        XCHG           AT ENTRY: DE=BUFFERPOINTER
                F565  23        INX    H                 HL=VALUE
                F566  73        MOV    M,E    PUT ADDR. DE TO DIR-BUFFER
                F567  23        INX    H      POINTED BY HL
                F568  72        MOV    M,D    AT RET. : DE=VALUE
                F569  C9        RET                      HL=BUFFERP. + 2
-------------------------------------------------------------------------------
NTRK            F56A  E5        PUSH   H      DEFINE NUMBER OF TRACKS
                F56B  06 01     MVI    B,01   NUMBER OF TRACKS=1
                F56D  21 00 0A  LXI    H,0A00 NUMBER OF BYTES / TRACK
          +---> F570  CD 14 DE  CALL   DE14   COMP. HL-DE, MORE SPACE ?
          |  +-- F573  D2 7E F5  JNC    F57E   NO
          |  |  F576  7C        MOV    A,H    YES, NEED MORE
          |  |  F577  C6 0A     ADI      0A   INCREMENT SPACE BY #A00
          |  |  F579  67        MOV    H,A
          |  |  F57A  04        INR    B      AND TRACKNUMBER BY 1
          +--|-- F57B  C3 70 F5  JMP    F570   CHECK AGAIN
             +-> F57E  3A 66 02  LDA    0266   GET CURRENT TRACK USED
                 F581  3C        INR    A      = STARTTRACK-1
                 F582  FE 50     CPI      50   END OF SIDE 0 ?
             +-- F584  C2 89 F5  JNZ    F589   NO, SO JUMP
             |  F587  3E 81     MVI    A,81   LOAD VALUE FOR SIDE 1
             +-> F589  E1        POP    H      GET BUFFERPOINTER
                 F58A  23        INX    H      SET START-TRACK-NUMBER
                 F58B  77        MOV    M,A    IN DIR-BUFFER AND IN
                 F58C  57        MOV    D,A    REG. D FOR WRITE-LOOP
             +-- F58D  C3 9C F5  JMP    F59C
          +--|-> F590  05        DCR    B
       +--|--|-- F591  CA A4 F5  JZ     F5A4   COMPUTE ENDTRACK-NUMBER
       |  |  |  F594  3C        INR    A      BY INCR. TRACKNUMBER IN A
       |  |  |  F595  FE 50     CPI      50   UNTIL NUMBER IN B=0 AND
       |  |  +-- F597  C2 9C F5  JNZ    F59C   TAKE CARE FOR SIDE CHANGE
       |  |  |  F59A  3E 81     MVI    A,81   NOW ON SIDE 1
       |  |  +-> F59C  FE D0     CPI      D0   IF VALUE D0 IS REACHED
    +--|--|----- F59E  CA F7 F5  JZ     F5F7   THEN THE DISK IS FULL
    |  |  +----- F5A1  C3 90 F5  JMP    F590
    |  +-------> F5A4  23        INX    H      SET ENDTRACK IN DIR-BUFFER
    |           F5A5  77        MOV    M,A    AND IN REG E FOR
    |           F5A6  5F        MOV    E,A    THE WRITE-LOOP
```



```
                F5A7  3A 60 02  LDA    0260
                F5AA  E6 EF     ANI      EF   SET DEVICE TO SIDE 0
                F5AC  32 60 02  STA    0260
                F5AF  CD 47 F1  CALL   F147
                F5B2  3A 67 02  LDA    0267   SET DIR-SECTORNUMBER FOR
                F5B5  32 65 02  STA    0265   THE UPDATE
                F5B8  CD 6C F2  CALL   F26C   UPDATE DIRECTORY (SIDE 0)
                F5BB  CD 11 F4  CALL   F411   CHECK I/O
                F5BE  C9        RET
-------------------------------------------------------------------------------
FREESPACE       F5BF  CD 00 F1  CALL   F100   GET STATUS
                F5C2  3A 61 02  LDA    0261
                F5C5  E6 40     ANI      40   CHECK IF WRITEPROTECTED
                F5C7  C2 1D F4  JNZ    F41D   JUMP IF PROTECTED
                F5CA  16 01     MVI    D,01   SET SECTOR=1
                F5CC  CD AF F3  CALL   F3AF   AND READ DIRECTORY
                F5CF  1E 04     MVI    E,04   SEARCH THE 4 BLOCKS
                F5D1  21 3E 01  LXI    H,013E
          +---> F5D4  7E        MOV    A,M    FOR A FREE SPACE
          |  |  F5D5  FE E5     CPI      E5   E5=FREE SPACE
       +--|--|- F5D7  CA 04 F6  JZ     F604   FREE SPACE FOUND
       |  |  |  F5DA  7D        MOV    A,L
       |  |  |  F5DB  C6 1F     ADI      1F   POSIT FOR NEXT DIR ENTRY
       |  |  |  F5DD  6F        MOV    L,A
       |  |  |  F5DE  7E        MOV    A,M
       |  |  |  F5DF  32 66 02  STA    0266   RETAIN MAX USED SECTORS
       |  |  |  F5E2  1D        DCR    E
       |  |  +- F5E3  CA EA F5  JZ     F5EA   THIS BLOCK IS FULL
       |  |     F5E6  23        INX    H      POSIT TO NEXT ENTRY
       |  +----|F5E7  C3 D4 F5  JMP    F5D4   CHECK AGAIN
       |        +-> F5EA  3A 67 02  LDA    0267
       |           F5ED  3C        INR    A      SET FOR NEXT SECTOR
       |           F5EE  FE 11     CPI      11
       |        +-- F5F0  CA F7 F5  JZ     F5F7   JUMP IF NO MORE AVAILLABLE
       |        |  F5F3  57        MOV    D,A    CONTINUE SEARCH
       |        |  F5F4  C3 CC F5  JMP    F5CC   IN NEXT SECTOR
-------|--------|-------------------------------------------------------------
       +--------+-> F5F7  CD 55 DD  CALL   DD55   SEND NEW LINE
                   F5FA  21 2A F6  LXI    H,F62A SETUP AND DISPLAY
                   F5FD  CD D4 DA  CALL   DAD4   DISK FULL MESSAGE
                   F600  E1        POP    H      SIMULATE RETURN
                   F601  C3 B4 F2  JMP    F2B4   EXIT VIA ABORT & DISK OFF
-------------------------------------------------------------------------------
                +-------> F604  E5        PUSH   H      DIRECTORY BACKUP
                          F605  3A 60 02  LDA    0260
                          F608  F6 14     ORI      14   SELECT SIDE 1
                          F60A  32 60 02  STA    0260
                          F60D  21 3E 01  LXI    H,013E SET OUTPUT-BUFFER
                          F610  22 6C 02  SHLD   026C   WITH CURRENT DIRECTORY
                          F613  00        NOP            SECTOR
                          F614  CD 47 F1  CALL   F147   SELECT DEVICE
                          F617  3E A0     MVI    A,A0   WRITE CURRENT
                          F619  32 62 02  STA    0262   DIRECTORY-SECTOR
                          F61C  3A 67 02  LDA    0267   ON SAME PLACE
                          F61F  32 65 02  STA    0265   BUT ON SIDE 1
                          F622  CD 6C F2  CALL   F26C   OF THE DISK
                          F625  CD 11 F4  CALL   F411   CHECK I/O
                          F628  E1        POP    H
                          F629  C9        RET
-------------------------------------------------------------------------------
                          F62A  4449534B2046554C4C0D00
                                 D I S K   F U L L
-------------------------------------------------------------------------------
@SAVE                     F635  F5        PUSH   PSW
                          F636  C5        PUSH   B
                          F637  D5        PUSH   D
                          F638  E5        PUSH   H
                          F639  CD BF F5  CALL   F5BF   FIND FREE SPACE IN DIR
                          F63C  36 42     MVI    M,42   SET BASIC-FLAG
                          F63E  1E 17     MVI    E,17   GET 23
                +---------> F640  23        INX    H      CHARACTERS
                |         F641  CD E0 DD  CALL   DDE0   TO ENCODE
                |         F644  FE 0D     CPI      0D   CR ?
                |  +------ F646  CA 53 F6  JZ     F653   YES, END OF INPUT-TEXT
                |  |      F649  77        MOV    M,A    NO , STORE CHAR. IN
                |  |      F64A  0C        INR    C      DIR-BUFFER
                |  |      F64B  1D        DCR    E      NEXT CHAR
                +--|------ F64C  C2 40 F6  JNZ    F640
                   |  +--- F64F  C3 59 F6  JMP    F659
                   |  |  +> F652  23        INX    H
                   +--|--|> F653  36 20     MVI    M,20   PAD REMAINDER WHITH BLANKS
                      |  |  F655  1D        DCR    E
                      |  +- F656  C2 52 F6  JNZ    F652
                      +----> F659  EB        XCHG
                             F65A  2A 9D 02  LHLD   029D
                             F65D  CD 64 F5  CALL   F564   PUTADR FOR HEAP-SIZE
                             F660  E5        PUSH   H
                             F661  2A 9B 02  LHLD   029B   START OF HEAP
```

```
READ-DIR        F3AF  CD CA F2  CALL   F2CA  @DON
                F3B2  3A 60 02  LDA    0260
                F3B5  E6 80     ANI      80  IS DEVICE ON ?
                F3B7  CA EA F3  JZ     F3EA  NO, SO POP+POPRET
                F3BA  CD 34 F3  CALL   F334  RESTORE
                F3BD  3E 84     MVI    A,84
                F3BF  32 60 02  STA    0260  SELECT DEVICE
                F3C2  CD 47 F1  CALL   F147
                F3C5  21 3E 01  LXI    H,013E  INPUTBUFFER
                F3C8  22 68 02  SHLD   0268
                F3CB  7A        MOV    A,D
                F3CC  32 67 02  STA    0267  SET STARTSECTOR=ENDSECTOR
                F3CF  32 65 02  STA    0265
                F3D2  CD 5F F1  CALL   F15F  SET SECTOR-REGISTER
                F3D5  3E 80     MVI    A,80
                F3D7  32 62 02  STA    0262  SET READ-DATA-COMMAND AND
                F3DA  CD 27 F2  CALL   F227  EXECUTE IT
                F3DD  3E 80     MVI    A,80
                F3DF  32 60 02  STA    0260
                F3E2  CD 47 F1  CALL   F147  DESELECT DRIVE
                F3E5  CD 11 F4  CALL   F411  CHECK I/O
                F3E8  C9        RET          RETURN TO CALLER IF NO ERRORS
-----------------------------------------------------------------------------
                F3E9  E1        POP    H
                F3EA  E1        POP    H     POPRET FROM DIFERENT LEVELS
                F3EB  C3 4D C1  JMP    C14D
-----------------------------------------------------------------------------
                F3EE  CD F0 F0  CALL   F0F0  DESELECT DRIVE
                F3F1  CD 34 F3  CALL   F334  RESTORE
                F3F4  C3 B4 F2  JMP    F2B4  @DOFF
-----------------------------------------------------------------------------
                F3F7  E1        POP    H
                F3F8  E1        POP    H     MULTI STACK-LEVEL
                F3F9  D1        POP    D
                F3FA  C1        POP    B     SYNTAX-ERROR-EXIT
                F3FB  F1        POP    PSW
                F3FC  E1        POP    H
                F3FD  C3 75 F0  JMP    F075
-----------------------------------------------------------------------------
HEX-TABLE       F400  00303132333435363738394142434445 46    HEX-TABLE
                        0 1 2 3 4 5 6 7 8 9 A B C D E F
-----------------------------------------------------------------------------
CHECK I/O       F411  F5        PUSH   PSW
                F412  C5        PUSH   B
                F413  D5        PUSH   D
                F414  E5        PUSH   H
                F415  3A 70 02  LDA    0270  GET I/O-STATUS
                F418  E6 5C     ANI      5C
                F41A  CA 4D C1  JZ     C14D  POPRET IF NO ERRORS
                F41D  CD 55 DD  CALL   DD55  SETUP ERROR-MSG
                F420  21 28 F3  LXI    H,F328
                F423  CD D4 DA  CALL   DAD4  DISPLAY IT
                F426  C3 B4 F2  JMP    F2B4  @DOFF
-----------------------------------------------------------------------------
@DIR            F429  F5        PUSH   PSW   DISPLAY DIRECTORY
                F42A  C5        PUSH   B     LOCATED ON TRACK 00
                F42B  D5        PUSH   D
                F42C  E5        PUSH   H
        +-------> F42D  16 01     MVI    D,01  BEGIN WITH SECTOR 1
        | +-----> F42F  CD AF F3  CALL   F3AF  READ DIRECTORY SECTOR
        | |       F432  1E 04     MVI    E,04
        | |       F434  21 3E 01  LXI    H,013E
        | |       F437  7E        MOV    A,M
        | |       F438  FE E5     CPI      E5  UNUSED DIR-ENTRY ?
        +-|------ F43A  CA 2D F4  JZ     F42D  YES, SO RESHOW DIR
          |   +-> F43D  E5        PUSH   H
          |   |   F43E  CD FF DA  CALL   DAFF  PMSGR
          |   |   F441  19 F3     DATA   F319  DISPLAY BLANK HEADER
          |   |   F443  21 00 F4  LXI    H,F400 ENTRY TO HEXTABLE
          |   |   F446  6A        MOV    L,D
          |   |   F447  7E        MOV    A,M
          |   |   F448  CD 60 DD  CALL   DD60  OUTC SECTORNUMBER 0-F
          |   |   F44B  3E 34     MVI    A,34
          |   |   F44D  93        SUB    E
          |   |   F44E  CD 60 DD  CALL   DD60  OUTC PROGRAMNUMBER 0-3
          |   |   F451  3E 20     MVI    A,20
          |   |   F453  CD 60 DD  CALL   DD60  OUTC SPACE
          |   |   F456  3E 18     MVI    A,18
          |   |   F458  E1        POP    H
          |   |   F459  CD 44 DB  CALL   DB44  PSTRM DISPLAY DIR-TEXTLINE
          |   |   F45C  1D        DCR    E
          |   +-- F45D  CA 6A F4  JZ     F46A  NEXT LINE OR ENTRY-PROCESS
          |       F460  7D        MOV    A,L
          |       F461  C6 08     ADI      08  POSIT FOR NEXT TEXTLINE
          |       F463  6F        MOV    L,A
          |       F464  7E        MOV    A,M   GET NEXT ENTRY-BYTE
          |       F465  FE E5     CPI      E5  UNUSED DIR-ENTRY ?
          +------ F467  C2 3D F4  JNZ    F43D  CHECK NEXT ENTRY
```



```
ENTRY     | | +---> F46A  CD FF DA  CALL   DAFF  DISPLAY "PGM="
          | |       F46D  22 F3     DATA   22F3
          | |       F46F  EF        RST    5
          | |       F470  0C        DATA   0C    POSIT CURSOR
          | |       F471  2C        INR    L
          | |       F472  E5        PUSH   H
          | |       F473  3E 20     MVI    A,20
          | |       F475  CD 1F DD  CALL   DD1F  INLINX
          | |       F478  C1        POP    B
          | |       F479  DA B4 F2  JC     F2B4  @DOFF, IF BREAK PRESSED
          | |       F47C  CD D2 DD  CALL   DDD2  IGNB
          | |       F47F  FE 0D     CPI      0D  RETURN KEY ONLY ?
          | +-----  F481  C2 9B F4  JNZ    F49B  NO, SO DECODE INPUT
          |         F484  3A 67 02  LDA    0267  YES, SO DISPLAY NEXT
          |         F487  3C        INR    A           DIRECTORY SECTOR
          |         F488  FE 11     CPI      11  END OF DIRECTORY ?
        +-|-|-----  F48A  CA 2D F4  JZ     F42D  YES, DIR RESTART
          |         F48D  57        MOV    D,A   NEXT SECTOR IN D
          |         F48E  06 06     MVI    B,06
          |   +-> F490  3E 08     MVI    A,08
          |   |   F492  EF        RST    5     CLEAR "PGM=" USING
          |   |   F493  03        DATA   03    6 BACKSPACES
          |   |   F494  05        DCR    B
          |   +-- F495  C2 90 F4  JNZ    F490
        +-|-----  F498  C3 2F F4  JMP    F42F  GOTO READ NEXT
----------|-----------------------------------------------------------------
DECODE      +---> F49B  21 10 F4  LXI    H,F410 TOP OF HEXTABLE
            +-> F49E  BE        CMP    M
          +-|-- F49F  CA A9 F4  JZ     F4A9  DEFINE SECTOR ( 0 TO F )
          | |   F4A2  2D        DCR    L
          | +-- F4A3  C2 9E F4  JNZ    F49E  COMPARE NEXT CHARACTER
          |     F4A6  C3 B4 F2  JMP    F2B4  INCORRECT SECTORDEFINITION
          +---> F4A9  55        MOV    D,L
                F4AA  CD AF F3  CALL   F3AF  READ THIS SECTOR FROM TRK0
                F4AD  0C        INR    C
                F4AE  CD D2 DD  CALL   DDD2  IGNB
                F4B1  21 04 F4  LXI    H,F404 HEXTAB OFFSET 4
          +---> F4B4  BE        CMP    M
          | +-- F4B5  CA BF F4  JZ     F4BF  SELECTED PROGRAM = 0,1,2,3
          | |   F4B8  2D        DCR    L     CONTINUE SEARCH UNTIL
          +-|-- F4B9  C2 B4 F4  JNZ    F4B4  ADDR. = F400
            |   F4BC  C3 B4 F2  JMP    F2B4  @DOFF, IF INCORRECT PRO.ID
            +-> F4BF  11 3E 01  LXI    D,013E START OF DIR-BUFFER
          +---> F4C2  2D        DCR    L
          | +-- F4C3  CA CD F4  JZ     F4CD  PROG.ID FOUND
          | |   F4C6  7B        MOV    A,E
          | |   F4C7  C6 20     ADI      20  POINT TO NEXT
          | |   F4C9  5F        MOV    E,A   DIRECTORY ENTRY
          +-|-- F4CA  C3 C2 F4  JMP    F4C2
------------|-----------------------------------------------------------------
FOUND       +-> F4CD  1A        LDAX   D
                F4CE  FE 42     CPI      42  IT IS A BASIC PROGRAM ?
                F4D0  C2 26 F5  JNZ    F526  NO JUMP TO MEMORY-READ
                F4D3  7B        MOV    A,E
                F4D4  C6 18     ADI      18
                F4D6  5F        MOV    E,A   GET BASIC-PROGRAM PARAM.
                F4D7  D5        PUSH   D     FROM DIRECTORY-TABLE
                F4D8  EB        XCHG
                F4D9  5E        MOV    E,M
                F4DA  23        INX    H
                F4DB  56        MOV    D,M
                F4DC  EB        XCHG
                F4DD  22 9D 02  SHLD   029D  SET SIZE OF HEAP
                F4E0  EB        XCHG
                F4E1  2A 9B 02  LHLD   029B  GET START OF HEAP
                F4E4  19        DAD    D
                F4E5  D1        POP    D
                F4E6  13        INX    D
                F4E7  13        INX    D
                F4E8  22 9F 02  SHLD   029F  SET BEGIN OF BASIC-PGM
                F4EB  CD 93 F3  CALL   F393  CALL ADDR
                F4EE  22 A1 02  SHLD   02A1  SET BEGIN OF VARIABLE-TAB.
                F4F1  CD 93 F3  CALL   F393  CALL ADDR
                F4F4  22 A3 02  SHLD   02A3  SET END OF VARIABLE-TABLE
                F4F7  EB        XCHG
                F4F8  56        MOV    D,M   START-TRACK IN REG. D
                F4F9  23        INX    H
                F4FA  5E        MOV    E,M   END-TRACK IN REG. E
                F4FB  1C        INR    E
                F4FC  2A 9B 02  LHLD   029B  SET START OF HEAP AS
            +-> F4FF  22 68 02  SHLD   0268  INPUT-BUFFER ADDR.
NEXT-TRACK  |   F502  D5        PUSH   D     SAVE TRACK-INFO
            |   F503  CD 42 F5  CALL   F542  SIDESEL
            |   F506  3E 80     MVI    A,80
            |   F508  32 62 02  STA    0262  SET READ-COMMAND
            |   F50B  CD 27 F2  CALL   F227  MULTISECTOR-READ
            |   F50E  2A 6A 02  LHLD   026A  GET CURRENT MEM-POINTER
            |   F511  D1        POP    D     GET TRACK-INFO
```

```
MULTI WRITE     F26C  F3          DI              MULTI-SECTOR WRITE
                F26D  F5          PUSH    PSW
                F26E  E5          PUSH    H
                F26F  D5          PUSH    D
                F270  C5          PUSH    B
                F271  CD B7 F1    CALL    F1B7    PROLOGE
                F274  21 0F F2    LXI     H,F20F  WRITE-LOOP LOCATION
                F277  CD AC F1    CALL    F1AC    MOVE IT TO WORK-AREA
           +->  F27A  2A 6C 02    LHLD    026C    OUTPUT-BUFFER-ADDR.
           |    F27D  EB          XCHG            TO DE
           |    F27E  21 03 FE    LXI     H,FE03  SET DCE-BUS TO MODE 2
           |    F281  36 A1       MVI     M,A1    IN/OUT  IN OUT
           |    F283  2B          DCX     H         A     C1  B
           |    F284  2B          DCX     H
           |    F285  36 8E       MVI     M,8E    SELECT PORT
           |    F287  3A 02 FF    LDA     FF02    CLEAR PENDING INTERRUPT
           |    F28A  3A 62 02    LDA     0262    SETUP
           |    F28D  32 00 FE    STA     FE00    WRITE COMMAND
           |    F290  36 88       MVI     M,88    SELECT FDC
           |    F292  2B          DCX     H
           |    F293  C3 3B 00    JMP     003B    GOTO OUTPUT PROCESS
           |    F296  F3          DI              STOP IT
           |    F297  EB          XCHG
           |    F298  22 6C 02    SHLD    026C    SAVE CURRENT
           |    F29B  22 6E 02    SHLD    026E    BUFFERPOINTERS
           |    F29E  CD CD F1    CALL    F1CD    I/O-CHECK
           |    F2A1  CA E2 F1    JZ      F1E2    EPILOGE
           |    F2A4  3C          INR     A       SET FOR NEXT SECTOR
           |    F2A5  32 67 02    STA     0267
           |    F2A8  06 0A       MVI     B,0A
           |    F2AA  CD 83 F1    CALL    F183    SECTORNUMBER TO FDC
           +--  F2AD  C3 7A F2    JMP     F27A    PROCESS NEXT SECTOR
----------------------------------------------------------------------
@DF             F2B0  F5          PUSH    PSW     DISK OFF
                F2B1  C5          PUSH    B
                F2B2  D5          PUSH    D
                F2B3  E5          PUSH    H
      +-------> F2B4  3E D0       MVI     A,D0
      |         F2B6  32 62 02    STA     0262    FORCE DEVICE ABORT
      |         F2B9  CD 77 F1    CALL    F177
      |         F2BC  CD F0 F0    CALL    F0F0    DEVICE & MOTOR OFF
      |         F2BF  3A 60 02    LDA     0260
      |         F2C2  E6 7F       ANI       7F    FLAG DEVICE BYTE
      |         F2C4  32 60 02    STA     0260
      |         F2C7  C3 25 F0    JMP     F025    MULTILEVEL-EXIT
------|---------------------------------------------------------------
@DN   |         F2CA  F5          PUSH    PSW     DISK ON
      |         F2CB  C5          PUSH    B
      |         F2CC  D5          PUSH    D
      |         F2CD  E5          PUSH    H
      |         F2CE  CD 90 F1    CALL    F190    INIT DCE-BUS
      |         F2D1  CD 00 F1    CALL    F100    GET STATUS
      |         F2D4  3A 02 FE    LDA     FE02    CHECK IF POWER ON DISK-UNIT
      |         F2D7  E6 01       ANI       01
      |    +----- F2D9  CA F2 F2    JZ      F2F2    --> IF POWER IS OFF
      |    |    F2DC  CD 3C F1    CALL    F13C    SET MOTOR ON
      |    |    F2DF  06 04       MVI     B,04    TRY 4 TIMES
      |    | +-> F2E1  CD 00 F1    CALL    F100    TO GET THE DRIVE READY
      |    | |   F2E4  3A 61 02    LDA     0261
      |    | |   F2E7  B7          ORA     A
      |  +-|-|-- F2E8  F2 FE F2    JP      F2FE    --> AT READY
      |  | | |   F2EB  CD 41 DE    CALL    DE41    WAIT 745MS
      |  | | |   F2EE  05          DCR     B
      |  | | +-- F2EF  C2 E1 F2    JNZ     F2E1    --> TRY AGAIN
      |  | +---> F2F2  21 09 F3    LXI     H,F309
      |  |       F2F5  CD 55 DD    CALL    DD55    DISPLAY 'DISK NOT READY'
      |  |       F2F8  CD D4 DA    CALL    DAD4
      +--|------ F2FB  C3 B4 F2    JMP     F2B4    TERMINATE VIA @DOFF
         +-----> F2FE  3A 60 02    LDA     0260
                 F301  F6 80       ORI       80    FLAG DEVICE READY
                 F303  32 60 02    STA     0260
                 F306  C3 4D C1    JMP     C14D    POPRET END COMMAND
                 F309  4449534B204E4F54205245414459 0D00
                        D I S K   N O T   R E A D Y
                 F319  0D20202020202020 00          LINEHEADER FOR DIR-DISPLAY
                 F322  2050474D3D00
                        P G M =
                 F328  4449534B20455252 4F520D00
                        D I S K   E R R O R
----------------------------------------------------------------------
```



```
RESTORE         F334  F5          PUSH    PSW
                F335  3E 03       MVI     A,03
           +--> F337  32 62 02    STA     0262    SET RESTORE COMMAND AND
           |    F33A  CD 77 F1    CALL    F177    EXECUTE IT
           | +-> F33D  CD 00 F1    CALL    F100    GET STATUS
           | |   F340  3A 61 02    LDA     0261
           | |   F343  E6 19       ANI       19    AND CHECK IT
           | +-- F345  C2 3D F3    JNZ     F33D    --> BUSY
           |     F348  F1          POP     PSW
           |     F349  C9          RET             RETURN IF RESTORE COMPLETED
-----------|----------------------------------------------------------
SEEK       |     F34A  32 60 02    STA     0260
           |     F34D  CD 47 F1    CALL    F147    SELECT DEVICE
           |     F350  7A          MOV     A,D     TARGET TRACK IS IN D
           |     F351  32 63 02    STA     0263
           |     F354  CD 53 F1    CALL    F153    SET TARGET TRACK
           |     F357  3E 13       MVI     A,13    SET SEEK COMMAND
           |     F359  F5          PUSH    PSW
           +---- F35A  C3 37 F3    JMP     F337    EXECUTE SEEK
----------------------------------------------------------------------
HEXIN           F35D  11 00 00    LXI     D,0000
      +-------> F360  21 10 F4    LXI     H,F410  TOP OF HEX-TABLE
      |         F363  CD E0 DD    CALL    DDE0    GET NEXT INPUT-CHARACTER
      |         F366  0C          INR     C       INCR. CHARACTERPOSITION
      |         F367  FE 20       CPI       20
      |         F369  C8          RZ              END IF SPACE OR CR
      |         F36A  FE 0D       CPI       0D
      |         F36C  C8          RZ              RESULT IS IN HL
      |         F36D  EB          XCHG
      |         F36E  29          DAD     H
      |         F36F  29          DAD     H       MULTIPLY CURRENT HEX-VALUE
      |         F370  29          DAD     H       BY 16
      |         F371  29          DAD     H
      |         F372  EB          XCHG
      |      +-> F373  BE          CMP     M       IF CURRENT CHAR. = TABLEVAL
      |      +-|-- F374  CA 7E F3    JZ      F37E    THEN JUMP
      |      | |   F377  2D          DCR     L       SELECT NEXT TABLEVALUE
      |   +--|-|-- F378  CA 85 F3    JZ      F385    UNTIL ZERO
      |   |  | +-- F37B  C3 73 F3    JMP     F373
      |   |  +---> F37E  7D          MOV     A,L
      |   |        F37F  3D          DCR     A       HEX-DIGIT = TABLEVALUE-1
      |   |        F380  B3          ORA     E
      |   |        F381  5F          MOV     E,A     INSERT AT LEAST SIG. POS.
      |  .+-|----- F382  C3 60 F3    JMP     F360    NEXT CHARACTER
      |     +----> F385  E1          POP     H
      |            F386  E1          POP     H
      |            F387  CD 55 DD    CALL    DD55      DISPLAY CR
      |            F38A  21 23 DC    LXI     H,DC23    AND MSG
      |            F38D  CD D4 DA    CALL    DAD4      SYNTAX-ERROR
      |            F390  C3 B4 F2    JMP     F2B4      END VIA @DOFF
----------------------------------------------------------------------
ADDR            F393  D5          PUSH    D       BUFFERPOINTER IN DE, SAVE
                F394  EB          XCHG
                F395  5E          MOV     E,M
                F396  23          INX     H       GET RELATIVE PROGR. ADDR.
                F397  56          MOV     D,M
                F398  2A 9B 02    LHLD    029B    GET STARTADDR.
                F39B  19          DAD     D       ADD TO GET THE ABS. ADDR.
                F39C  D1          POP     D       OF BASIC PROGR. IN HL
                F39D  13          INX     D
                F39E  13          INX     D       POSIT BUFFERPOINTER
                F39F  C9          RET
----------------------------------------------------------------------
@LOAD           F3A0  F5          PUSH    PSW
                F3A1  C5          PUSH    B
                F3A2  D5          PUSH    D
                F3A3  E5          PUSH    H
                F3A4  CD D2 DD    CALL    DDD2    GET SELECTION-NUMBER
                F3A7  FE 0D       CPI       0D    CR ONLY ?
                F3A9  C2 9B F4    JNZ     F49B    --> NO , GOTO DECODE
                F3AC  C3 F8 F3    JMP     F3F8    POPRET VIA SYNTAX-ERROR
----------------------------------------------------------------------
```

```
SET DATA REG.   F153  F3          DI            TRANSM. DATA TO FDC
                F154  F5          PUSH  PSW
                F155  E5          PUSH  H
                F156  C5          PUSH  B
                F157  06 08       MVI   B,08     SELECT DATA REGISTER
                F159  3A 63 02    LDA   0263     DATA FROM #263
                F15C  C3 F8 F0    JMP   F0F8     WRITE TO FDC AND TERMINATE
-------------------------------------------------------------------------
SET SECTOR REG. F15F  F3          DI
                F160  F5          PUSH  PSW
                F161  E5          PUSH  H
                F162  C5          PUSH  B
                F163  06 0A       MVI   B,0A     SELECT SECTOR REGISTER
                F165  3A 67 02    LDA   0267     SECTOR FROM #267
                F168  C3 F8 F0    JMP   F0F8     WRITE TO FDC AND TERMINATE
-------------------------------------------------------------------------
SET TRACK REG.  F16B  F3          DI
                F16C  F5          PUSH  PSW
                F16D  E5          PUSH  H
                F16E  C5          PUSH  B
                F16F  06 0C       MVI   B,0C     SELECT TRACK REGISTER
                F171  3A 66 02    LDA   0266     TRACK FROM #266
                F174  C3 F8 F0    JMP   F0F8     WRITE TO FDC AND TERMINATE
-------------------------------------------------------------------------
SET COMMAND     F177  F3          DI            FDC-COMMANDS
                F178  F5          PUSH  PSW
                F179  E5          PUSH  H
                F17A  C5          PUSH  B
                F17B  06 0E       MVI   B,0E     SELECT COMMAND REGISTER
                F17D  3A 62 02    LDA   0262     COMMAND FROM #262
                F180  C3 F8 F0    JMP   F0F8     WRITE TO FDC AND TERMINATE
-------------------------------------------------------------------------
WRITE TO FDC    F183  21 03 FE    LXI   H,FE03   SET DCE BUS TO MODE A1 B0
                F186  36 A1       MVI   M,A1     OUT  IN OUT OUT
                F188  2B          DCX   H          A   C1  C2  B
                F189  2B          DCX   H        SET REG. SELECTION TO
                F18A  70          MOV   M,B      PORT B (#FE01)
                F18B  32 00 FE    STA   FE00     DATA TO PORT A (#FE00)
                F18E  00          NOP
                F18F  00          NOP
INIT DCE        F190  21 03 FE    LXI   H,FE03   SET DCE-BUS TO MODE 0
                F193  36 99       MVI   M,99     IN IN IN OUT
                F195  2B          DCX   H        A  C1 C2  B
                F196  2B          DCX   H
                F197  36 01       MVI   M,01     SET DISK ENABLED
                F199  C9          RET
-------------------------------------------------------------------------
READ FROM FDC   F19A  21 03 FE    LXI   H,FE03   SET DCE-BUS TO MODE 2
                F19D  36 C1       MVI   M,C1     IN/OUT  IN OUT
                F19F  2B          DCX   H          A     C1  B
                F1A0  2B          DCX   H        SET REG. SELECTION TO
                F1A1  70          MOV   M,B      PORT B (#FE01)
                F1A2  78          MOV   A,B      AND GENERATE A STROBE
                F1A3  F6 40       ORI     40
                F1A5  77          MOV   M,A      SET READ STROBE
                F1A6  3A 00 FE    LDA   FE00     FETCH DATA FROM PORT A
                F1A9  36 01       MVI   M,01     RESET STROBE
                F1AB  C9          RET
-------------------------------------------------------------------------
MOVE            F1AC  06 18       MVI   B,18     MOVE 24 BYTES
          +->   F1AE  7E          MOV   A,M      FROM H,L TO D,E
          |     F1AF  12          STAX  D
          |     F1B0  23          INX   H        USED TO MOVE DISK-I/O
          |     F1B1  13          INX   D        ROUTINE TO INTERRUPT AREA
          |     F1B2  05          DCR   B        AND SAVE THE AREA.
          +--   F1B3  C2 AE F1    JNZ   F1AE
                F1B6  C9          RET
-------------------------------------------------------------------------
I/O PROLOGE     F1B7  3E 80       MVI   A,80     MASK INTERRUPT
                F1B9  32 F8 FF    STA   FFF8
                F1BC  21 38 00    LXI   H,0038   SAVE INTERRUPT AREA
                F1BF  11 48 02    LXI   D,0248   INTO ENVELOPE AREA
                F1C2  CD AC F1    CALL  F1AC     MOVE THE 24 BYTES
                F1C5  AF          XRA   A
                F1C6  32 70 02    STA   0270     CLEAR I/O STATUS
                F1C9  11 38 00    LXI   D,0038   D,E NEW AREA FOR DISK-I/O
                F1CC  C9          RET
-------------------------------------------------------------------------
I/O-CHECK       F1CD  06 0E       MVI   B,0E     SELECT STATUS REGISTER
                F1CF  CD 9A F1    CALL  F19A     READ STATUS
                F1D2  47          MOV   B,A
                F1D3  3A 70 02    LDA   0270     CUMULATE BY OR
                F1D6  B0          ORA   B        INTO #270 FOR ALL
                F1D7  32 70 02    STA   0270     MULTIPLE I/O
                F1DA  3A 67 02    LDA   0267     LOAD CURRENT SECTOR
                F1DD  21 65 02    LXI   H,0265   AND LAST SECTOR
                F1E0  BE          CMP   M        ALL SECTORS PROCESSED ?
                F1E1  C9          RET            DECISION IN MAIN PART
```



```
-------------------------------------------------------------------------
I/O EPILOGE     F1E2  21 48 02    LXI   H,0248
                F1E5  11 38 00    LXI   D,0038
                F1E8  CD AC F1    CALL  F1AC     MOVE BACK ORIG. INT. RAM
                F1EB  3A 5F 00    LDA   005F
                F1EE  32 F8 FF    STA   FFF8     RESTORE ORIG. INT. MASK
                F1F1  C1          POP   B
                F1F2  D1          POP   D
                F1F3  E1          POP   H
                F1F4  F1          POP   PSW
                F1F5  FB          EI
                F1F6  C9          RET
-------------------------------------------------------------------------
*               //    EXECUTION-ADDR. OF READ-LOOP
READ-LOOP       38    F1F7  7E          MOV   A,M      GET 1 BYTE FROM FDC
                39    F1F8  12          STAX  D        PUT IT INTO RAM
                3A    F1F9  13          INX   D        NEXT RAM LOCATION
                3B    F1FA  C1          POP   B        SIMULATE RETURN
                3C    F1FB  FB          EI             READY FOR NEXT BYTE
                3D    F1FC  00          NOP            >
                3E    F1FD  00          NOP             >
                3F    F1FE  00          NOP              >
          +->   40    F1FF  00          NOP               >  AWAITING INTERRUPT
          |     41    F200  00          NOP               >  FROM FDC
          |     42    F201  00          NOP              >
          |     43    F202  00          NOP             >
          |     44    F203  00          NOP             >
          |     45    F204  3A 02 FE    LDA   FE02     CHECK IF FDC=BUSY
          |     48    F207  E6 01       ANI     01
          +--   4A    F209  C2 40 00    JNZ   0040     STILL BUSY, WAIT AGAIN
                4D    F20C  C3 52 F2    JMP   F252     EXIT AT END OF READ
-------------------------------------------------------------------------
*               //    EXECUTION-ADDR. OF WRITE-LOOP
WRITE-LOOP      38    F20F  70          MOV   M,B      WRITE 1 BYTE TO FDC
                39    F210  13          INX   D        NEXT BYTE TO BE WRITTEN
                3A    F211  C1          POP   B        SIMULATE RETURN
                3B    F212  1A          LDAX  D        LOAD NEXT BYTE
                3C    F213  FB          EI             READY FOR NEXT WRITE
                3D    F214  47          MOV   B,A      BYTE IS IN B
                3E    F215  00          NOP            >
                3F    F216  00          NOP             >
          +->   40    F217  00          NOP              >
          |     41    F218  00          NOP               > AWAITING INT.
          |     42    F219  00          NOP              >
          |     43    F21A  00          NOP             >
          |     44    F21B  00          NOP             >
          |     45    F21C  3A 02 FE    LDA   FE02     CHECK IF FDC=BUSY
          |     48    F21F  E6 01       ANI     01
          +--   4A    F221  C2 40 00    JNZ   0040     STILL BUSY, WAIT AGAIN
                4D    F224  C3 96 F2    JMP   F296     EXIT AT END OF WRITE
-------------------------------------------------------------------------
MULTI READ      F227  F3          DI            MULTI-SECTOR READ
                F228  F5          PUSH  PSW
                F229  E5          PUSH  H
                F22A  D5          PUSH  D
                F22B  C5          PUSH  B
                F22C  CD B7 F1    CALL  F1B7     PROLOGE
                F22F  21 F7 F1    LXI   H,F1F7   READ-LOOP LOCATION
                F232  CD AC F1    CALL  F1AC     MOVE IT TO WORK-AREA
          +->   F235  2A 68 02    LHLD  0268     INPUT-BUFFER-ADDR
          |     F238  EB          XCHG           TO DE
          |     F239  21 03 FE    LXI   H,FE03   SET DCE-BUS TO MODE 2
          |     F23C  36 C1       MVI   M,C1     IN/OUT  IN OUT
          |     F23E  2B          DCX   H          A     C1  B
          |     F23F  2B          DCX   H
          |     F240  36 AE       MVI   M,AE     SELECT PORT
          |     F242  3A 02 FF    LDA   FF02     CLEAR PENDING INTERRUPT
          |     F245  3A 62 02    LDA   0262     SETUP
          |     F248  32 00 FE    STA   FE00     READ COMMAND
          |     F24B  36 E8       MVI   M,E8     SELECT FDC
          |     F24D  2B          DCX   H
          |     F24E  FB          EI             READY FOR FDC
          |     F24F  C3 40 00    JMP   0040     GOTO INPUT PROCESS
          |     F252  F3          DI             STOP IT
          |     F253  EB          XCHG
          |     F254  22 68 02    SHLD  0268     SAVE CURRENT
          |     F257  22 6A 02    SHLD  026A     BUFFERPOINTERS
          |     F25A  CD CD F1    CALL  F1CD     I/O-CHECK
          |     F25D  CA E2 F1    JZ    F1E2     EPILOGE
          |     F260  3C          INR   A        SET FOR NEXT SECTOR
          |     F261  32 67 02    STA   0267
          |     F264  06 0A       MVI   B,0A
          |     F266  CD 83 F1    CALL  F183     SECTORNUMBER TO FDC
          +--   F269  C3 35 F2    JMP   F235     PROCESS NEXT SECTOR
-------------------------------------------------------------------------
```

```
*******  D A I  -  S W I S S  -  D O S     BY  A. MEYSTRE  1983  *******
*******          STARTED VIA UTILITY MODE BY SETTING            *******
*******                    V6 TO F000                           *******
```

```
KEYBOARD-LOOP   F000  F3          DI
                F001  21 09 F0    LXI    H,F009   SET NEW RETURN-ADDR
                F004  E5          PUSH   H        AND CONTINUE
                F005  E5          PUSH   H        WITH STANDARD
                F006  C3 78 D5    JMP    D578     KEYBOARD-PROCESSING
```

```
DOS             F009  F3          DI              ENTRY AFTER
                F00A  C5          PUSH   B        KEYBOARD PROCESSING
                F00B  D5          PUSH   D
                F00C  E5          PUSH   H
                F00D  F5          PUSH   PSW
                F00E  3E 09       MVI    A,09     TEST THE 4 BYTES
                F010  21 BA 02    LXI    H,02BA   OF THE INPUTBUFFER
                F013  16 04       MVI    D,04
          +->   F015  BE          CMP    M        TAB-KEY PRESSED ?
        +-|--   F016  CA 2C F0    JZ     F02C     YES IT WAS -->
        | |     F019  23          INX    H
        | |     F01A  15          DCR    D        CHECK NEXT BYTE
        | +--   F01B  C2 15 F0    JNZ    F015
    +---|--->   F01E  F1          POP    PSW      NO TAB PRESSED
    |   |       F01F  E1          POP    H
    |   |       F020  D1          POP    D
    |   |       F021  C1          POP    B
    |   |       F022  E1          POP    H        RETURN TO
    |   |       F023  FB          EI              NORMAL WORK
    |   |       F024  C9          RET
----|---|-------------------------------------------------------------
EXIT|   |       F025  2A 71 02    LHLD   0271     FROM ANY STACK LEVEL
    |   |       F028  00          NOP
    |   |       F029  C3 E4 F7    JMP    F7E4     TO EXTENDED EXIT
----|---|-------------------------------------------------------------
    |   +--->   F02C  36 00       MVI    M,00     CLEAR THE TAB-CHARACTER
    |           F02E  21 00 00    LXI    H,0000
    |           F031  39          DAD    SP       SAVE CURRENT STACKPOINTER
    |           F032  22 71 02    SHLD   0271
    |           F035  00          NOP
    |           F036  21 78 D5    LXI    H,D578   INHIBIT FURTHER DOS
    |           F039  22 6E 00    SHLD   006E     COMMANDS
    |           F03C  FB          EI
    |           F03D  3E 40       MVI    A,40     DISPLAY THE DOS-PROMT
    |           F03F  CD 1A DD    CALL   DD1A     AND GET THE INPUT
    |           F042  DA 54 F0    JC     F054     BREAK IS PRESSED -->
    |           F045  CD D2 DD    CALL   DDD2     TEST COMMAND
    |           F048  FE 0D       CPI      0D     JUST ENTER ?
    |     +---- F04A  C2 79 F0    JNZ    F079     NO
    |     |     F04D  3E 0C       MVI    A,0C     IF ENTER ONLY, THEN
    |     |     F04F  EF          RST    5        CLEAR SCREEN
CMD-      |     F050  03          DATA   03       AND TERMINATE
EXIT -|---|---> F051  CD 5E DD    CALL   DD5E     NEW LINE
    |     |     F054  F3          DI              -----> STOP INTERRUPT
    |     |     F055  3A 40 00    LDA    0040     CHECK CURRENT TASK
    |     |     F058  E6 C0       ANI      C0
    |     |     F05A  3E 3E       MVI    A,3E     > CURSOR IF IN UTILITY
    |     | +-- F05C  C2 61 F0    JNZ    F061
    |     | |   F05F  3E 2A       MVI    A,2A     _ CURSOR IF IN BASIC
    |     | +-> F061  CD 60 DD    CALL   DD60
    |     |     F064  21 B9 02    LXI    H,02B9   KEYBOARD-BUFFER-ADDR.
    |     |     F067  16 05       MVI    D,05
    |     | +-> F069  36 00       MVI    M,00     CLEAR KEYBOARD-BUFFER
    |     | |   F06B  23          INX    H
    |     | |   F06C  15          DCR    D        ALL 4 POSITIONS
    |     | +-- F06D  C2 69 F0    JNZ    F069
    |     |     F070  21 00 F0    LXI    H,F000   ENABLE DOS-COMMANDS
    |     |     F073  22 6E 00    SHLD   006E
    |     +---- F076  C3 1E F0    JMP    F01E     DOS EXIT
CMD FROM -+---> F079  21 99 F0    LXI    H,F099   COMMAND-TABLE
PROGRAM-CALL    F07C  1E 01       MVI    E,01
                F07E  CD 34 CA    CALL   CA34     COMMAND SEARCH
          +--   F081  D2 8D F0    JNC    F08D     INVALID --->
          |     F084  5E          MOV    E,M
          |     F085  23          INX    H        D,E=COMMAND-ADDR.
          |     F086  56          MOV    D,M
          |     F087  21 25 F0    LXI    H,F025   COMMAND EXIT
          |     F08A  E5          PUSH   H        SAVE IT
          |     F08B  EB          XCHG
          |     F08C  E9          PCHL            EXECUTE COMMAND
          +->   F08D  CD 55 DD    CALL   DD55
                F090  21 23 DC    LXI    H,DC23   DISPLAY SYNTAX-ERROR
                F093  CD D4 DA    CALL   DAD4
                F096  C3 25 F0    JMP    F025     COMMAND EXIT
```



```
COMMAND-TABLE   F099  03 44 49 52 29 F4       DIR     F429  DIRECTORY
                F09F  04 4C 4F 41 44 A0 F3    LOAD    F3A0  LOAD PGM/MEM
                F0A6  04 53 41 56 45 35 F6    SAVE    F635  SAVE PGM
                F0AD  01 57 B0 F6             W       F6B0  SAVE MEM
                F0B1  02 44 4E CA F2          DN      F2CA  DISK ON
                F0B6  02 44 46 B0 F2          DF      F2B0  DISK OFF
                F0BB  01 5A 32 F7             Z       F732  RESET DOS
                F0BF  01 55 E3 02             U       02E3  USER COMMAND
                F0C3  02 54 4E 20 F7          TN      F720  TAPE ON
                F0C8  02 54 46 29 F7          TF      F729  TAPE OFF
                F0CD  02 42 53 61 F7          BS      F761  SAVE BASIC PT
                F0D2  02 42 4C 72 F7          BL      F772  LOAD BASIC PT
                F0D7  01 45 45 F7             E       F745  POKE #135,2
                F0DB  01 4D 4B F7             M       F74B  MERGE SETUP
                F0DF  01 50 5A F7             P       F75A  #29B=#2EC
                F0E3  01 43 3F F7             C       F73F  RESET CURSOR
                F0E7  01 52 83 F7             R       F783  RUN NNN
                F0EB  01 30 A3 F7             0       F7A3  MODE 0
END OF TABLE    F0EF  00
```

```
DRIVE OFF       F0F0  F3          DI              SET DRIVE AND MOTOR OFF
                F0F1  F5          PUSH   PSW
                F0F2  E5          PUSH   H
                F0F3  C5          PUSH   B
                F0F4  06 06       MVI    B,06     SELECT DEVICE REGISTER
                F0F6  3E 00       MVI    A,00     SET DRIVE OFF
                F0F8  CD 83 F1    CALL   F183     WRITE TO FDC
TERMINATE I/O   F0FB  C1          POP    B
                F0FC  E1          POP    H
                F0FD  F1          POP    PSW
                F0FE  FB          EI
                F0FF  C9          RET
```

```
GET STATUS      F100  F3          DI
                F101  F5          PUSH   PSW
                F102  E5          PUSH   H
                F103  C5          PUSH   B
                F104  06 0E       MVI    B,0E     SELECT STATUS REGISTER
                F106  CD 9A F1    CALL   F19A     READ FROM FDC
                F109  32 61 02    STA    0261     SET STATUSBYTE TO #261
                F10C  C3 FB F0    JMP    F0FB
```

```
GET DATA REG.   F10F  F3          DI
                F110  F5          PUSH   PSW
                F111  E5          PUSH   H
                F112  C5          PUSH   B
                F113  06 08       MVI    B,08     SELECT DATA REGISTER
                F115  CD 9A F1    CALL   F19A     READ FROM FDC
                F118  32 63 02    STA    0263     DATAREGISTERBYTE TO #263
                F11B  C3 FB F0    JMP    F0FB
```

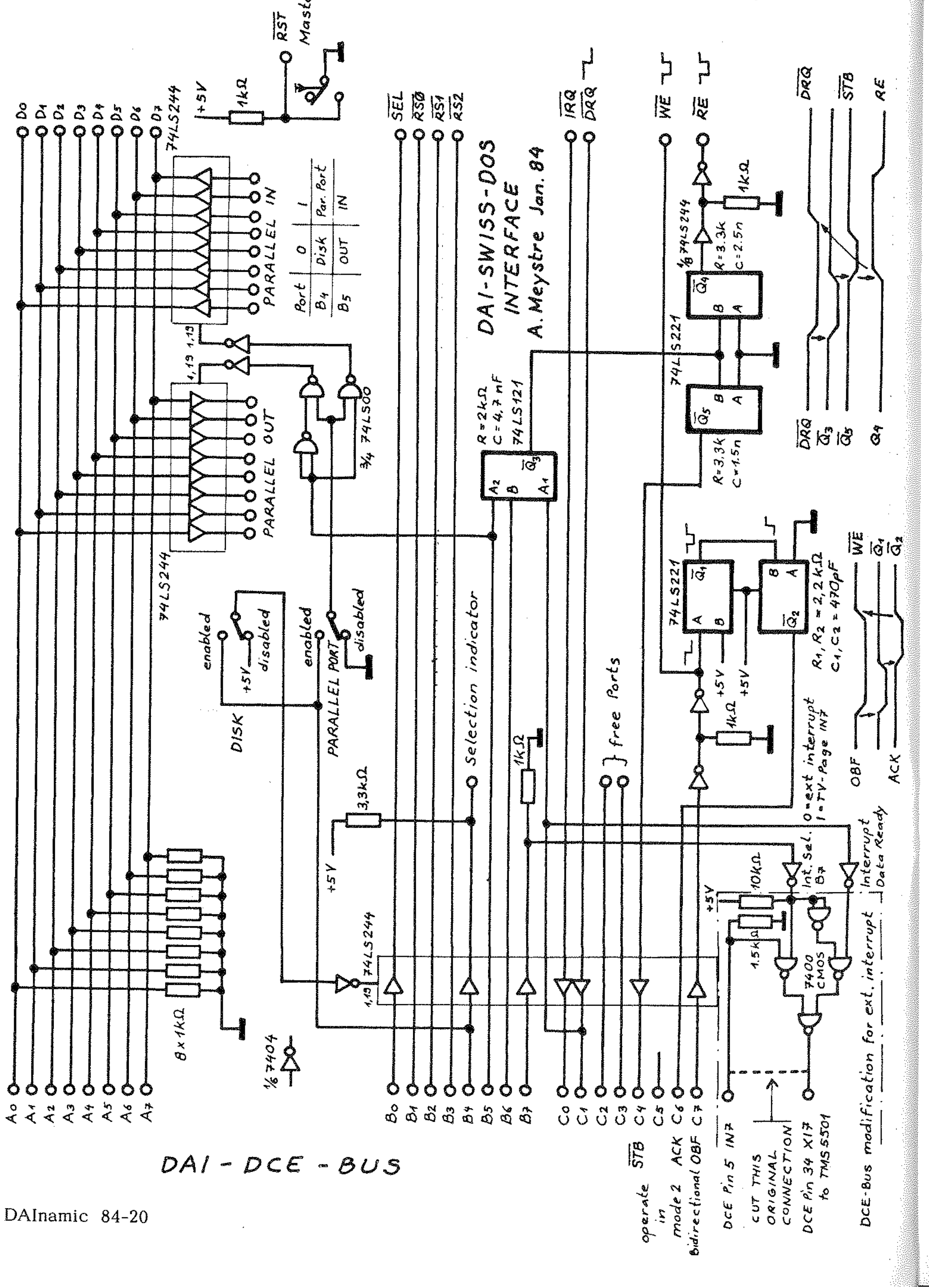
```
GET TRACK REG.  F11E  F3          DI
                F11F  F5          PUSH   PSW
                F120  E5          PUSH   H
                F121  C5          PUSH   B
                F122  06 0C       MVI    B,0C     SELECT TRACK REGISTER
                F124  CD 9A F1    CALL   F19A     READ FROM FDC
                F127  32 64 02    STA    0264     CURRENT TRACK TO #264
                F12A  C3 FB F0    JMP    F0FB
```

```
GET SECTOR REG. F12D  F3          DI
                F12E  F5          PUSH   PSW
                F12F  E5          PUSH   H
                F130  C5          PUSH   B
                F131  06 0A       MVI    B,0A     SELECT SECTOR REGISTER
                F133  CD 9A F1    CALL   F19A     READ FROM FDC
                F136  32 65 02    STA    0265     CURRENT SECTOR TO #265
                F139  C3 FB F0    JMP    F0FB
```

```
DRIVE ON        F13C  F3          DI              DRIVE ON, NO CHECKS
                F13D  F5          PUSH   PSW
                F13E  E5          PUSH   H
                F13F  C5          PUSH   B
                F140  06 06       MVI    B,06     SELECT DEVICE REGISTER
                F142  3E 80       MVI    A,80     SET DRIVE ON
                F144  C3 F8 F0    JMP    F0F8     WRITE TO FDC AND TERMINATE
```

```
SET DEVICE REG. F147  F3          DI              SELECT THE DRIVE
                F148  F5          PUSH   PSW
                F149  E5          PUSH   H
                F14A  C5          PUSH   B
                F14B  06 06       MVI    B,06     SELECT DEVICE REGISTER
                F14D  3A 60 02    LDA    0260     DEVICE ATTR. FROM #260
                F150  C3 F8 F0    JMP    F0F8     WRITE TO FDC AND TERMINATE
```

TEAK FC-50 CONTROLLER



3. ORGANIZATION

Fig. 3.1 below shows the FC-50 System Block Diagram.

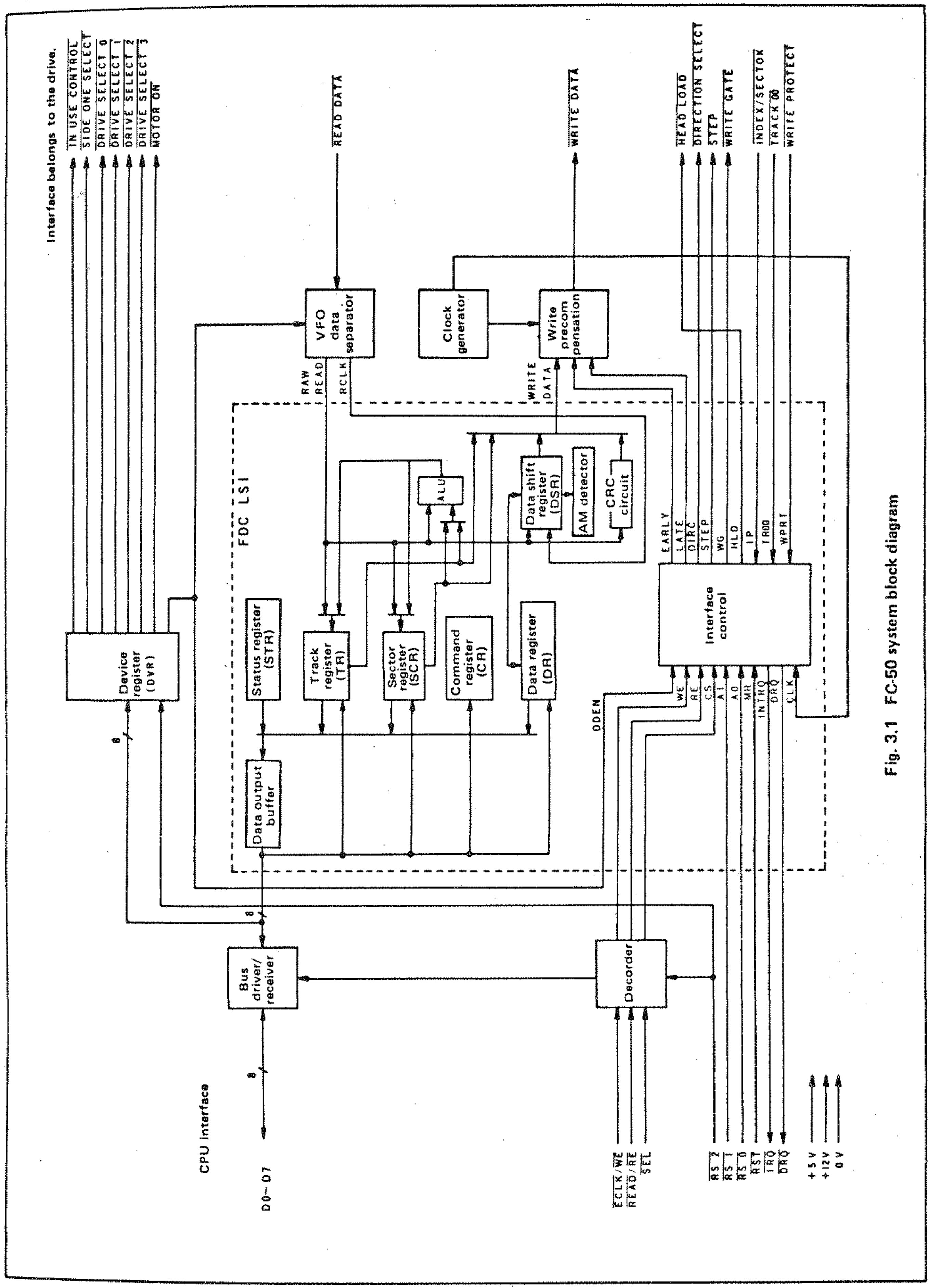


Fig. 3.1 FC-50 system block diagram

more commands with the DAI-SWISS-DOS:

@BS	save BASIC-pointer to 0047 in memory
@BL	load BASIC-pointer after crash, may be your lucky !?
@P	reset BASIC-pointer to origin 29B = 2EC then use CLEAR MMM to shift your program
@R NNN	run from line NNN without clear of variables, usefull for BASIC 1.0
@U	User-exit by jumping to 2E3 where normaly you will find a ret-command change C9 00 00 by C3 and your own routine address and come back via ret-command. All registers are saved and restored by DOS.
@Z	finally this command will terminate any DOS activity.

All commands can be used in BASIC 1.0 or 1.1 and in Utility mode and exepted DIR they can all be called from a BASIC-program:

```
C$ ="W65E0 BFFF MY GRAPHIC IN MODE 6"+CHR$(13):CALLM #F7C0
```

When the so called command is a LOAD, your current program may be overloaded thats why a RUN is automaticaly generated after this LOAD.
By this methode you can run an automatic program show.

One of the musts was to run old software beginning at #300 and to save screen-pictures from any graphic-mode. For this reason the workareas of the DOS are located in the 128 bytes DAI I/O-buffer and in the upper part of ENVELOPE 1. So you can not play music during disk read or write. After LOAD of a program you can use ENVELOPE 1 without restrictions.
The location of the DOS-ROM is F000 - F7FF. Programs and memory-data are read or written without buffering directly to the final RAM-locations This methode save space and time. On this way the most old programs can be SAVEed and LOADed by disk without changes.
The loading or saveing time for a program or memory-file is about the same as from tape, but measured in seconds instead of minutes. A 40K program need only 7 seconds, a MODE 6 picture 5 seconds if the disk motor is remaining on. Getting the drive ready by DN (motor on) will need 1 second extra.
The disks can be organised in all formats supported by the TEAK FC-50 controller (IBM-format).
the directory display at loading time looks like this:

```
          TEST DISK FGT   23.04.83      this is the disk-header

          01 B DIR-LISTING
          02 B FGT-DEMO 1
          03 B FGT-DEMO 1.1
          10 M FGT SMALL CHARACTERS
          11 M FGT LARGE CHARACTERS
          12 M DEMO + FGT 29B-183F
          13 M SCREEN 1 MODE 6A
          20 M SCREEN 2 MODE 6A
          21 B SHORT TEST BASIC ONLY
          22 B POKE ACTION DAInamic 8  PGM=
label ......**
data type .....*
B = BASIC
M = Memory
documentation ...**********************
entry for the program selection by label ....**
```

if you type just return, you get the next directory block displayed DIR will terminate without loading at BREAK. Any label can be entered at any time. DIR is working in all text-screen-modes and in UT.

How to initialize the system ? After power on or reset go to utility-mode by typing

UT then RETURN-key

then change the keyboard-vector V6

V6 D578-F000 and RETURN-key twice

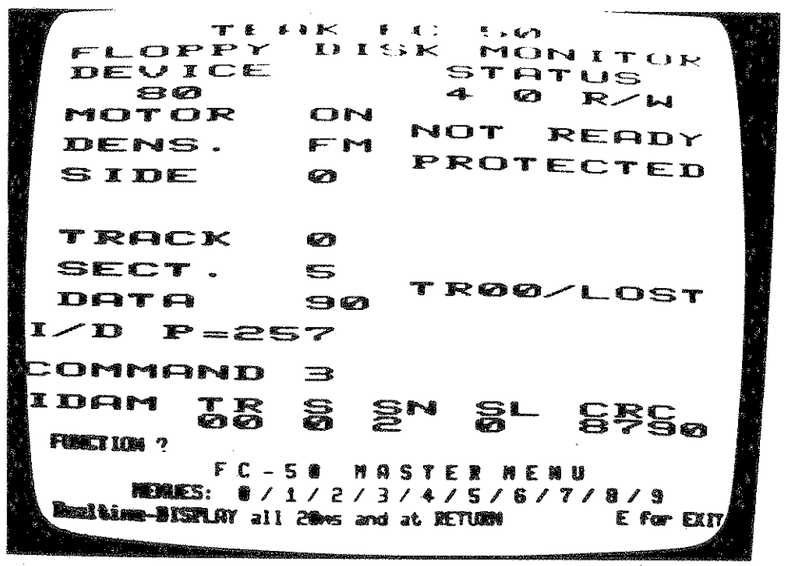
then you can return to BASIC by typing B

From now the DOS is ready for execution of each command.
As you can see the often used timer-vector V7 is not used by the DOS but during the disk I/O the V7 interrupt is used for the data transfer. This will stop your clock counter for somes seconds.
Some of the DOS-commands are usefull without disk. The used methode of intercepting the TAB-key can be used to create your own command extension.
There was not enought space in the ROM to create a secure DELETE command, but d'ont worry about this. The best way to never lost important data is to create many backup copies. The valid copy is the last one found in the directory.
Using BASIC-programs delete,compress,copy or any else is possible.
The follwing DOS-software is ready now:

MONITOR	Disk manipulations and hardware-test
FORMATER	formate, initialize directory and check disk
SCREEN-SHOW	save and load MODE 6 screens (demo)
COPY	disk copy with 1 drive (one to one for backup)

The DOS-command are supporting one drive only, but under program-control up to 4 drives can be accessed.
The DAI-SWISS-DOS works with BASIC 1.0 and BASIC 1.1
I am using my DOS since 1 year without any crash or datalost.

January 1984 A. Meystre HB9BIG
Andlauerstr. 10
CH-4132 MUTTENZ
SWITZERLAND



CATALOG

CODE	TITLE	AUDIO	DCR
G1	Games collection 1	400 Bfr	550
G2	Games collection 2	400	550
G3	Games collection 3	400	550
G4	Games collection 4	800	950
G5	Games collection 5	400	550
G6	Games collection 6	750	900
G7	Games collection 7	750	900
G8	Games collection 8	750	900
G9	Games collection 9	750	900
G10	Games collection 10	750	900
G11	Games collection 11	750	900
G12	Games collection 12	750	900
DNA	DNA assembly pack	1100	1250
FGT	fast graph text	1000	1150
FGTA	FGT applications	1000	1150
TK1	Toolkit 1	1000	1150
TK2	Toolkit 2	1000	1150
TK3	Toolkit 3	1000	1150
TK4	Toolkit 4	1000	1150
TK5	Toolkit 5	1000	1150
PE1	Primary Education 1	1000	1150
MF	Math'fun	1000	1150
SE1	Secundary Education 1	1000	1150
SE2	Secundary Education 2	1000	1150
W3	Mathematics 3	1000	1150
BB	Bits & Bytes	750	900
ML	Mailing List	1000	1150
GT	Graphic Tablet	1000	1150
M1	Music collection 1	300	450
M2	Music collection 2	300	450
M3	Music collection 3	300	450
DTP	DAI Tiny Pascal	1000	1150
EGT	English-German trainer	1000	1150
DD	DAI DEMO + Basic tutor	500	650
CH	Sargon Chess	1500	1650
SI	Space Invaders	800	950
T80	Tape 80-81	850	1000
N10	Newsletter 10	500	650
N11	Newsletter 11-12	650	800
N13	Newsletter 13-14-15	650	800
CTP	Centipede	600	750
DRI	Driver	600	750
SUI	Super Invader	600	750
DAPA	DAI PANIC	800	950
JR	MICRO'S-Onderwijs	990	1100
SPL	SPL Macro-assembler	1100	1250
TT1	Taal-tape 1	750	900

CODE	TITLE	AUDIO	DCR
F1	Fysica 1	750	900
FB	Familiebudget	500	650
GH	Grafische Hulp	500	650
ACR	Acrobates	600	750
CG	Character Generator	1750	1900
FWP	Fast Word Processor	2000	2150
PM	PAC-MAN	1100	1250
DL	DAYLAXIANS	2100	2100
PZ	PUZZLY	2100	2100
DU	DUEL	2100	2100
CL	C.L.I.O.	3000	3000

HARDWARE & PUBLICATIONS

PCS	DAIpc SCHEMATICS	850
BOD	BEST of DAInamic (80-81)	500
SNG	Super Noise Generator	1500
DCE	DCE-interface cards	2500
N82	Newsletters 9 - 13	500
GP	Gestuctureerd Programmeren	1100

This developement was much harder than I first think, but compared with the DAI-DOS I get much more for the money with a greather ease of use. As hardware I chose the

TEAK FC-50 controller and the TEAK FC-50F drive

So I get 400K of disk storage on one floppy for about SFr 1700.- .
The housing and the power supply is not included in this price.
It is not possible to use the DAI-power-supply because you will need 2 Amperes at 5 Volts and 2 Amperes at 12 Volts for the disk-unit.
The disk-system is connected via the DCE-bus. To get a real external interrupt input, the DCE-bus must be slitly modified, but this changes are easy. The slow clockrate of the DAI make it not possible to use the double recording density (MFM) of the drive. If you want to use MFM and shifting so your diskcapacity to 800K on 1 diskette, you will need a DMA-attachement, or may be instead of the slow interrupt way of the TMS5501-chip use the HOLD input of the 8080-processor, but I have not tested this yet. Anyway my DOS-system is working fine since 1 year with single recording density (FM,400K).
The interface from the DCE-bus to the FC-50 controller is built with 9 low cost TTL-chips. After the implementation of this DOS it is still possible to use port A of the DCE-bus alternatively with the disk.
The selection function is done by software or hardware. The DOS-software consist of disk-access subroutines for all controller functions callable from BASIC or machine language programs and the following DOS-direct-commands initiated by the useless TAB-key. Keying TAB, the DOS-prompt @ is displayed indicating DOS-command-state reached. Every DOS-command is terminated by the RETURN-key.

Start DOS in UT-mode by setting V6 D578-F000

@DN disk motor on
@DF disk motor off
@SAVE XXXXXXXXXXXXXXXXXXXXX
save a BASIC-program to disk with 22 character for documentation
@W29B A8FF XXXXXXXXXXXXXXXXXXXXX
save memory locations 29B - A8FF XXXXXX = documentation text
this could be a BASIC-program with the machine code part
all saved and loaded in once.
or saving a picture in MODE 6A
@W63B8 BFFF
the pointer addresses needed to do this are found in UT-mode
with D290 2AF
The so saved programs and memory-parts can be retrieved by
@DIR displaying 4 entries of the directory and then waiting for
a program selection by numerical label or just RETURN to continue.
You can do this even in MODE 6A. Or if you know the label just use
@LOAD LL
where LL is the numeric label.

now some non disk commands:

@ clear screen
@0 set screen to MODE 0
@C reset cursor to standard character
@TN tape on
@TF tape off

or as mergeing help: first CLEAR MMMMM for enough memory space
then EDIT first part and BREAK, BREAK
@M prepare for merge command
@LOAD LL the second part.
@E and RETURN key twice to get the first part back and merged to second part, but you must use different linenumbers.

libr	83/353	Bits & Bytes / Char. generator / Math' fun	—
libr	83/14	Catalog	—
libr	83/81	Catalog	—
libr	83/142	Catalog 1.4.83	—
libr	83/253	Catalog	—
libr	83/333	Catalog NL	v.d. Dunne
libr	83/249	DAI service manual	INDATA
libr	83/149	DAI-nibble	v. Randen
libr	83/151	DDT : description DAInamic debugging tool	Coremans
libr	83/289	DDT : Erratus	—
libr	83/53	DIDAISOFT : Catalog	v. Rompaey
libr	83/283	Games collection 12	—
libr	83/346	Programs T. Mikulic	Mikulic
libr	83/317	SFGT : Poke notes	Hermans
libr	83/150	SFGT : specifications	Lambrecht
libr	83/217	Software contributions	Druijff
libr	83/152	Toolkit 5	—
pasc			
pasc	83/157	Hardcopy source DTP	Mariatte
periph			
periph	83/297	Centronics 739 printer	Atherton
periph	83/72	DCE bus : Universal interface card	Uytterhoeven
periph	83/380	EPROM-programmer	Beuckelaers
periph	83/305	EPROM-programmer (English)	Knoops/Atherton
periph	83/368	Epson MX-80/100 : Use Graftrax + mode III	de Dauw
periph	83/6	Epson : Bit image graphics	Epson USA
periph	83/330	High speed data loader	Kop/Rison
periph	83/56	KEN-DOS floppy standard	Gooswit
periph	83/374	KEN-DOS information	Gooswit
periph	83/34	MDCR : Review on use	Atherton
periph	83/399	MDCR : Tape direction indication	Siccardo
periph	83/294	Printers : Specs Star DP-510/515; Epson RX-80/FX-80	—
periph	83/334	Unidata : processor card	Groeneveld
progr			
progr	83/219	16 colour characters	Mariatte
progr	83/303	COLORG : programming techniques (English)	Druijff/Atherton
progr	83/413	Deleting parts of programs	Vandebergh
progr	83/257	Demo ON ERROR GOTO	De Bont
progr	83/33	Duplication of program lines	Di Martino
progr	83/328	Entry to editor	Atherton
progr	83/285	Evaluation «Star hunt»	Druijff
progr	83/39	Hints and tips	—
progr	83/224	How to assign values to variables	Druijff
progr	83/16	How to make a program	Druijff
progr	83/154	How to make fast running programs	Druijff
progr	83/15	IF-THEN-ELSE statement	Boerrigter
progr	83/359	Mathematical operators	Druijff
progr	83/363	Printer on/off	Druijff
progr	83/102	Programming in machine language	Druijff
progr	83/138	Run (line nr) in BASIC V1.0	Gruiters
progr	83/153	Timing problems in time measurements	v. Lieshout
read			
read	83/24	Magazine news	Schepens
rem			
rem	83/96	8080 mnemonics with BASIC equivalent	Atherton
rem	83/27	CP/M	Digital Research

rem	83/354	Didaisoft info	—
teach			
teach	83/335	Course DCE-microcomputer - 1	Dirksen Opleidingen
teach	83/408	Course DCE-microcomputer - 2	Dirksen Opleidingen
teach	83/270	Course microprocessors - 1	Beuckelaers
teach	83/322	Course microprocessors - 2	Beuckelaers
teach	83/400	Course microprocessors - 3	Beuckelaers
teach	83/364	Lamps & Switches : Connections	de Bont
tool			
tool	83/250	AZERTY keyboard routine	Schepens
tool	83/188	Basicode 2	v. Lieshout
tool	83/234	Bootstrap for screen files	De Winter
tool	83/208	Cassette adjustment test	Mariatte
tool	83/232	Cassette routines SDK-85 interface	v. Ool
tool	83/220	Cassette tape lister	Assink
tool	83/327	Changing Baud-rate	Atherton
tool	83/321	Conversion programs to IMP INT	Looije
tool	83/174	Conversion upper to lower case	Boerrigter
tool	83/112	DAI as terminal on telephone lines	Gruiters
tool	83/327	Disable Break	Atherton
tool	83/179	Flashing in 4 colour	Harte
tool	83/202	Integers : Print USING alignment	Bonne
tool	83/343	Micro Fast Graphics	Tjoews
tool	83/178	Negative cursor	de Bont
tool	83/327	Paddles : How to read event buttons	Atherton
tool	83/180	Printer spooler	Lelie
tool	83/247	Program generator	Looije
tool	83/321	Program generator : erratum	Looije
tool	83/184	Read/write in Utility	Boerrigter
tool	83/167	Screen continuation lines	Boerrigter
tool	83/290	Screen continuation lines (French)	Boerrigter/Dufour
tool	83/168	Screen to buffer / buffer to screen	Hermans
tool	83/256	Screen to buffer / buffer to screen	De Bont
tool	83/394	Seikosha GP100 : New characters	de Jong
tool	83/370	Sort in machine language	Poels
tool	83/208	Use of GOTO X / RUN X	v. Dunne
tool	83/200	Use of direct commands in programs	Dufour
tool	83/388	Video RAM table generator	Pennisi
use			
use	83/42	SPL Assembler	Sphynx
ut			
ut	83/162	Sorting demo	Maertens
ut	83/252	Speed comparison BASIC-PLM	Uliana
viewd			
viewd	83/218	Videotex in Belgium	RTT Belgium

tool	82/43	DBL : Bootstrap loader V1.*	de Raedt
tool	82/283	DBL : Bootstrap loader V2.*	de Raedt
tool	82/222	DNA : Alternative printer routine	Menier
tool	82/282	DNA : modif. altern. printer routine	Menier
tool	82/160	DNA : modifications	de Raedt
tool	82/288	FGT : check on ASCII-values used	—
tool	82/274	Hardcopy mode 0	Leuenberger
tool	82/38	Lichtpen routine	Berkcx
tool	82/118	MDCR tape controller	Div.
tool	82/149	ML programs in a REM-statement	Berkx
tool	82/322	Mode 7	Looije
tool	82/261	Mode 7/8	Looije
tool	82/320	Mode 8	Looije
tool	82/139	Mode 8 : 240 x 528 resolution	Sip
tool	82/145	One textline in mode 5/6	Hermans
tool	82/108	Oscilloscope	Sip
tool	82/83	Printer : alternative user routine	Boerrigter
tool	82/20	RANDOM; how random is it	Hermans
tool	82/226	Variable cursor	Druijff
tool	82/328	Video RAM : demo control + mode bytes	Looije
tool	82/161	Weather satellite pictures	Bakker
use			
use	82/206	3D-representation	—
use	82/310	DAI-panic	—
use	82/218	Microbeam : Static calculations	Dal
use	82/121	Rubic's cube	Druijff
use	82/312	SPU : System program unit	—
use	82/31	Secundary Education 2	v. Rompaey
ut			
ut	82/176	DAI time	Bonduelle
ut	82/86	Flight simulator	Meystre
ut	82/318	Mixing of phrases	Moens
ut	82/195	Rohrschach tests	de Bont
viewd			
viewd	82/154	Interactive Videotex	RTT Belgium
viewd	82/62	Viditel	v.d. Hijden

DAInamic 1983

CODE	PAGE	SUBJECT	AUTHOR
calc			
calc	83/244	Mathematic programs	Duluins
disk			
disk	83/46	DAI floppies	Baptiste
disk	83/45	FGT-DISK-PEEK-POKE-2	Couwberghs
disk	83/328	File handling	Atherton
disk	83/249	New DAI master DOS	INDATA
disk	83/176	Random access	Couwberghs
firmw			
firmw	83/37	32 Display modes and 136 colours	Gidney
firmw	83/59	Audio cassette interface - 1	Boerrigter

firmw	83/130	Audio cassette interface - 2	Boerrigter
firmw	83/201	BASIC function CALLM n.m.	Dufour
firmw	83/28	BASIC function TAB	Boerrigter
firmw	83/298	BASIC monitor - part 1	Boerrigter
firmw	83/376	BASIC monitor - part 2	Boerrigter
firmw	83/172	DAI BASIC : extension ON ERROR GOTO	Looije
firmw	83/107	DAI BASIC : extension restart routines	Looije
firmw	83/291	DAI BASIC : exension restart routines (French)	Looije/Dufour
firmw	83/28	DCE bus : Initialisation	Boerrigter
firmw	83/228	Editor story	Boerrigter
firmw	83/135	Error messages : codes	Dufour
firmw	83/231	Firmware manual : corrections	Boerrigter
firmw	83/379	Firmware manual : corrections - 2	Boerrigter
firmw	83/254	Paddles : explanation and modification	Dufour
firmw	83/311	ROM Index	Hards
firmw	83/329	Use of OTSW # 131	Atherton
firmw	83/308	Video RAM : screen control bytes (English)	Looije/Atherton
game			
game	83/101	Character invasion	Hermans
game	83/372	Labyrinth game	de Bont
game	83/387	Maze game	Dierckx
game	83/285	Star hunt	v. Espen
graph			
graph	83/169	2D Transformations	v. Amerongen
graph	83/158	3D Plot : Rotating bugalow	Vingerling
graph	83/64	3D colour pictures	Roelants
graph	83/128	3D drawing : Hyperboloide	Roelants
graph	83/137	Bloc in reserve	Druijff
graph	83/412	Christmas night	Moeys
graph	83/319	Circle drawing	Doumont
graph	83/278	Copy variations	Vingerling
graph	83/134	Cube : rotating + translating in space	Dufour
graph	83/161	FGT : Demo on use	Atherton
graph	83/171	FIAT logo	Uliana
graph	83/175	Graftext	v. Amerongen
graph	83/40	Greetings from Hawai	De Bont
graph	83/137	Hour glass	Druijff
graph	83/19	Incremental circle generation	v. Amerongen
graph	83/186	Large text	Groeneveld
graph	83/332	Marzianetto	Uliana
graph	83/201	Mountains	De Brouwere/Dufour
graph	83/199	Notte di fuoco in mode 1	Uliana
graph	83/70	Paint : fill a shape with a colour	Druijff
graph	83/248	Shadeshape	Di Ciris
graph	83/276	Shape design	Hermans
graph	83/137	Symmetric fan	v. Randen
hardw			
hardw	83/20	A/D conversion	Kopp
hardw	83/407	Adapter DAI/RGB - Monochrome monitor	Mariatte
hardw	83/126	DAI Character generator	Cassebaum
hardw	83/320	DAI schematics : errata - 2	Boerrigter
hardw	83/86	Defectuous +5V power supply	Verberkt
hardw	83/207	Power supply and overvoltage protection	Corswandt
hardw	83/164	Video hardware modifications	Hospers/Doornenbal
libr			
libr	83/5	Acrobates	v. Rijsselberg