EFF5 Print character

BANK 1:#E000 to #EFFF

E000 Entry points
E0FE FPT multiplication
E108 FPT division
E126 Store registers ABCD in RAM 00D5-00D8
E133 Restore registers ABCD from RAM 00D5-00D8
E16D Integer addidion
E18D Integer subtraction
E1AC Integer multiplication
E22B Integer division
E3CF Store registers BCDE in RAM 00D5-00D8
E855 FPT power
EDAA FPT addition
EDB4 FPT subtraction

BANK 2:#E000 to #EFFF

E000 Entry points
E030 Screen parameters MODE 0
E045 Screen parameters MODE 1/1A
E05A Screen parameters MODE 2/2A
E06F Screen parameters MODE 3/3A
E084 Screen parameters MODE 4/4A
E099 Screen parameters MODE 5/5A
E0AE Screen parameters MODE 6/6A
E0C3 Initialize screen
E102 Output one character
E13D Output carriage return
E159 Output form feed
E166 Output backspace
E1A9 When end of line reached
E1CB Scrolling
E1FD Initialise screen character area
E21C Change to character MODE
E237 Set text colours
E254 Set colour parameters
E267 Load colours in header/trailer area
E279 Set cursor position
E2CC Ask cursor position and size character screen
E316 Set cursor mode
E330 Update cursor pointers
E344 Flash cursor
E36B Load screen location pointed at by cursor
E38B Get character from line
E3D9 Change MODE
E407 Set up screen RAM area
E545 Load pointers with screen parameters
E59A Start addresses tables screen parameters
E5A6 Initialise memory management routine
E5FC Initialise header
E687 Place cursor on beginning of line
E6A4 Set graphics colours
E6F2 (HL) = (HL) - (DE)

E6FB Compare DE - HL
E701 Add offset to address
E706 Two complement of 16-bits data
E710 Draw a DOT on screen
E71B Draw a line on the screen
E818 Fill a rectanglar area on the screen
E884 Ask colour of point and size of graphics screen
EB46 Calculate number of bytes on extended lines
EBF4 Initialise editor
EC1E Obey edit
EC4B Window up
ECB3 Window down
ECF8 Window right
ED50 Window left
ED88 Cursor up
EDAB Cursor down
EDD2 Cursor left
EDF6 Cursor right
EF4B Insert character in buffer
EFCC Delete character in buffer

BANK 3:#E000 to #EFFF

E000 Entry points
E018 Update input pointer
E024 Encode inputs without line number
E04F Check if character after basiccmd is ':' or car.ret.
E09A Get address encoding instruction and go to it
E2E6 Get char. from line, check if an upper case char.
E2F1 Strings variable types
E81F Set D depending on contents of A
E859 Check statement terminator
E8C5 ASCII table upper case
E8FD ASCII table lower case
E935 Get inputs from keyboard or RS232
E93F Load ASCII value for key pressed in buffer
EA00 Start UTILITY
EA0D Initialise UTILITY
EA62 Error
EA74 Entry from BASIC
EA7D Initialisation parameters
EA8E Table with UTILITY commands
EAB3 D - Display
EADB Handle inputs after a UT command
EB15 Convert ASCII-character to HEX-value
EB26 L - Look
EB41 Set look windows, start look
EB56 Go/Look check for carriage return
EB5D Restart 0 (RST 0)
EC45 Compare DE with HL
EC63 Restore CPU registers
EC7C Calculate DE - HL
EC83 M - Move
ECBA Z - Reset
ED01 Print space
ED06 Scan keyboard, print character

```
++++++++++++++++++++++
+ SFGT  POKE - NOTES +
++++++++++++++++++++++
```

POKE :

768,POS	hor. position for mode 1 - 4
769,0	
768,POS MOD 256	hor. position for mode 5 - 6
769,POS / 256	
770,POS	ver. position
771,STEP	hor. position between two caracters
772,STEP	ver. position between two caracters
773,STEP	hor. step by ver. CR
774,STEP	ver. step by hor. CR
775,MIN MOD 256	hor. min.
776,MIN / 256	
778,MAX MOD 256	hor. max.
779,MAX / 256	
777,MIN	ver. min.
780,MAX	ver. max.
781,BEGIN	matrix begin
782,END	matrix end
783,BEGIN	begin of the string
784,LENGHT	lenght of the string
785,X	x = 0 : display right- up
	x = 1 : display left- up
	x = 2 : display right- down
	x = 3 : display left- down
	+4 : retain hor. position
	+8 : retain ber. position
	+16 : upside- down
	+64 : high speed
	+128 : inkey flag
786,COL	for-& background color in 16 color mode
	number of colorg in 4 color modes
787,DELAY	gives delay / caracter
788,CHAR	number of characters / image
789,HORSTAP	horstap / character
790,TABEL	tabel choise
791,NUMBER	the number of characters to be combined to one character

mikroshop hageland

COMPUTERS en ALLE TOEBEHOREN

Herseltsesteenweg 103 - 3220 AARSCHOT

tel. 016/56 87 70

OPEN : dinsdag - vrijdag 14 u - 20 u.
zaterdag 9 u 30 - 17 u.
GESLOTEN : zondag, maandag, feestdagen.

1.	MEMOCOM MDCR met TOS ,kabel en doos MDCR cassettes,handleiding	17000 BF
2.	DAI pc ,snoeren,manual,demo cassette en abonnement DAInamic	45900 BF
3.	DAI pc ,idem 2 ,MDCR met TOS,kabel,doos MDCR cassettes	61000 BF
4.	DAI pc, idem 2 ,Kleurmonitor met RGB ingang, RGB kabel	64000 BF
5.	DAI pc, idem 2 ,Kleurmonitor,RGB kabel,MDCR met TOS,kabel	74000 BF
6.	DAI pc, idem 2 ,Kleurmonitor,RGB kabel,DAI floppy 2x80K form.	97700 BF
7.	DAI pc, idem 2 ,Kleurmonitor,RGB kabel,DAI floppy 2x160K form.	102900 BF
8.	DAI pc, idem 2 ,Kleurmonitor,RGB kabel,DAI floppy 2x320K form.	112200 BF

DAI floppy 2 x 80K form.	54470 BF
DAI floppy 2 x160K form.	59000 BF
DAI floppy 2 x320K form.	73600 BF

DAI Hardware manuals Deel I+II	1140 BF
DAI Software manuals Deel I+II	2285 BF
DAI pc testmanual	1785 BF
Draagtas voor DAI pc	1130 BF
Firmware manual(ROM listing)	1290 BF
Grafische interface voor DAI pc	5900 BF

Monitors

BARCO : *	42 cm RGB kleurmonitor met RGB en video ingang sound weergave via ingebouwde luidspreker;met RGB kabel	24500 BF
*	42 cm RGB monitor | TV combinatie met afstandsbediening RGB,video en UHF ingang;ingebouwde luidspreker voor sound	32900 BF
INDATA :*	KAGA medium resolution monitor 380 dots,kleur,31 cm geen ingebouwde sound,RGB ingang,met RGB kabel	21900 BF
*	KAGA super high resolution ;630 dots;kleur ; 31 cm RGB ingang ,geen ingebouwde sound ;met RGB kabel	38000 BF
	RGB kabel apart voor monitor met RGB ingang (prise PPT)	990 BF

Printers

: alle printers zijn grafische interface voor DAI pc inbegrepen.

EPSON : *	RX-80 (tractor feed)	80 kar.100 cps	33500 BF
*	RX-80 F|T (friction + tractor feed)	80 kar.100 cps	
*	FX-80 (friction + pinfeed)	80 kar.160 cps	46900 BF
*	MX-100 Type III F|T (friction|tractorfeed)	132 kar.100 cps	54900 BF
STAR : *	STX -80 Thermal printer (frictionfeed)	80 kar. 60 cps	18500 BF
*	GP 510 (friction|tractorfeed) 2K buffer	80 kar.100 cps	30500 BF
*	GP 515 (friction|tractorfeed) 2K buffer	132 kar.100 cps	40900 BF
Brother : *	CE-50 BT Bidirectionele Daisy Wheel schrijfmachine printer en plotter ,2K buffer,proportioneel	14 cps	42900 BF

BTW 19 % inbegrepen Prijswijzingen voorbehouden 15.10.83

REAL CIRCLES

OPMERKINGEN OVER "INCREMENTAL CIRCLE GENERATION"
DOOR F.VAN AMERONGEN.(CF.DAInamic 14)
****

Dit klein programma is inderdaad knap. De hoge snelheid van tekening wordt bekomen door het vermijden van de conventionele sinussen en cosinussen. Maar het gebruikte algorithme is NIET afkomstig van de cirkelvergelijking.Het gaat hier inderdaad om ELLIPSEN met nagenoeg gelijke assen,schuins liggende met een hoek van 45 graad op de x-as. Dit kan iedereen gemakkelijk vaststellen indien K1 in lijn 30 door een hogere waarde (b.v.=30) vervangen wordt.(Indien een "number out of range" bekomen wordt dient de oorspronkelijke waarde van R verkleind te worden). De enige punten van deze ellipsen die ook tot de cirkels behoren zijn 0,R),(R,0), (0,-R) en (-R,0).

Om een echte cirkel te bekomen dient men gebruik te maken van de parametrische vergelijkingen:

$$x = R\cos\theta \qquad y = R\sin\theta$$

,die door differentieren het volgende geven:

$$dx = -y\,d\theta \qquad dy = x\,d\theta$$

Dit geeft aanleiding tot:

$$x_2 \simeq x_1 - y_1\Delta\theta \qquad y_2 \simeq y_1 + x_1\Delta\theta$$

waar $\Delta\theta$..een kleine hoekverschil is, en $\simeq$."ongeveer gelijk tot" betekent.

Het vertrekpunt (x_1 , y_1) voldoet aan de vergelijking

$$x_1^2 + y_1^2 = R^2$$

doch men stelt vast dat punt (x_2, y_2) niet op de cirkel ligt:

$$x_2^2 + y_2^2 = R^2(1+\Delta\theta^2)$$

Men ziet dat elk bekomen punt coordinaten heeft die elk te groot zijn met een factor

$$\sqrt{(1+\Delta\theta^2)}$$

,zodat het iteratie-process een "accumulatiefaktor" van $\left(\sqrt{(1+\Delta\theta^2)}\right)^{n-1}$ zal produceren, waar n het aantal zijden van de benaderende veelhoek voorstelt:

$$n = \frac{2\pi}{\Delta\theta}$$

,en een spiraalkromme i.p.v. een cirkel bekomen zou worden.

Ten einde een cirkel te bekomen dient dus elk nieuw coordinaat door factor

$$\sqrt{(1+\Delta\theta^2)}$$

gedeeld te worden.

Hierbij een programma dat duidelijk het verschil laat zien tussen de "pseudo-cirkels" van F.van Amerongen en echte wiskundige cirkels.

G.Doumont

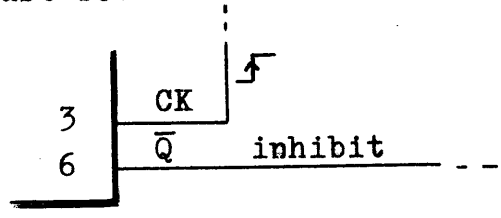
CIRKELS...DIE GEEN (ECHTE) CIRKELS ZIJN

```
5     GOTO 2000
10    PRINT CHR$(12):MODE 0
15    CLEAR 2000:DIM U(190.0),V(190.0)
20    COLORG 0 5 10 15:R=90.0
35    MODE 6:XC=XMAX/2.0:YC=YMAX/2.0
40    DRAW XC,0 XC,YMAX 5:DRAW 0,YC XMAX,YC 5
50    X1=R:Y1=0.0:K1=3.0
55    FOR K1=3.0 TO 31.0 STEP 7.0
60    K=K1/X1:A=SQR(1.0+K*K):B=1.0/A:C=5.0
70    FOR I=0.0 TO (2.0*PI)/K:X2=(X1+K*Y1)*B:Y2=(Y1-K*X1)*B
80    GOSUB 1000:NEXT:WAIT TIME 50
110   FOR I=0.0 TO (2.0*PI)/K:X2=X1+K*Y1:Y2=Y1-K*X2:C=10.0:U(I)=X2:V(I)=Y2
120   GOSUB 1000:NEXT:WAIT TIME 100
122   IF K1=31 GOTO 130
124   C=0.0:DRAW R+XC,YC U(0.0)+XC,V(0.0)+YC C
126   FOR J=1.0 TO (2.0*PI)/K:DRAW U(J)+XC,V(J)+YC U(J-1.0)+XC,V(J-1.0)+YC C
128   NEXT J
130   NEXT K1:WAIT TIME 250:GOTO 10
1000  P=X1:Q=Y1:M=X2:N=Y2:DRAW P+XC,Q+YC M+XC,N+YC C
1100  X1=X2:Y1=Y2:RETURN
2000  PRINT " De cirkel in 't groen is de 'echte wiskundige"
2010  PRINT " cirkel'.In oranje kleur wordt de 'Van Amerongen"
2020  PRINT " cirkel' getekend. Voor elke waarde van K1 tussen"
2030  PRINT " 3 en 31 met step 7 worden beide cirkels op het"
2040  PRINT " scherm voorgesteld en wordt de oranje cirkel daarna"
2050  PRINT " afgeveegd. Op deze manier komt het 'ovaliseren' van"
2060  PRINT " de oranje veelhoeken zeer duidelijk te voorschijn"
2070  WAIT TIME 750:GOTO 10
4000  PRINT CHR$(14);"CIRKELS...DIE GEEN (ECHTE) CIRKELS ZIJN":LIST
4010  PRINT
```

CORRECTIONS DAI SCHEMATICS - 2

Some failures have been found in the DAI schematics. It concerns sheet 4, Timing.

- IC 37 : It is not a 74LS150, but a 74LS170.
- IC 18 : Some pins are exchanged. The correct lay-out must be:



The wiring to the IC remains as drawn.

Please update your copies accordingly.

Jan Boerrigter - Febr. 1983

E R R A T U M

A bug was reported in the PROGRAMGENERATOR in the last issue of DAINAMIC. Because I sent a wrong programlisting to the editor. Line 140 should be:

```
140 POKE #135,2:CALLM #F800:POKE #135,0:RETURN
```

This will solve the problem of SYNTAX ERROR IN LINE 140 when an INPUT statement is used.

N.P. Looije

CONVERTING A PROGRAM TO 'IMP INT'

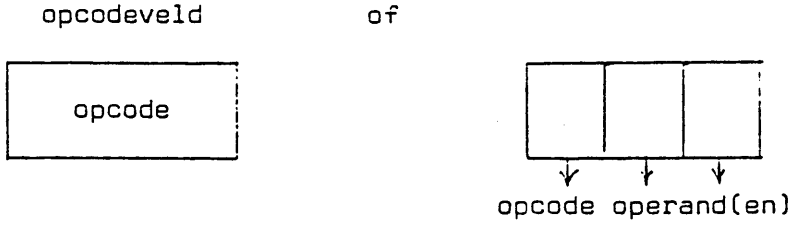
As you all will probably know by now (from Frank Druijff) a program runs faster when all variables and constants are integer. Also I have never met (until now) 1.0 person as if there is also 1.247 person. Though some programs do try to make you believe this. Here is a little method to change all variables AND constants (with .0) to integer. Just type this when your program is in memory and your program will be converted to IMP INT. The 2 lines program search for .0 and substitutes it by spaces.

```
*IMP FPT
*CLEAR (size of program + a little more)
*EDIT
<BREAK> <BREAK>
*IMPINT
*NEW
*65534 FOR A=PEEK(#A2)+PEEK(#A3)*256 TO PEEK(#A4)+PEEK
       (#A5)*256:IF PEEK(A)=46 THEN IF PEEK(A+1)=48
       THEN POKE A,32:POKE A+1,32
*65535 NEXT
*RUN 65534
*NEW
*POKE #135,2
wait a little while and your program
is in IMP INT  inclusive the constants
```

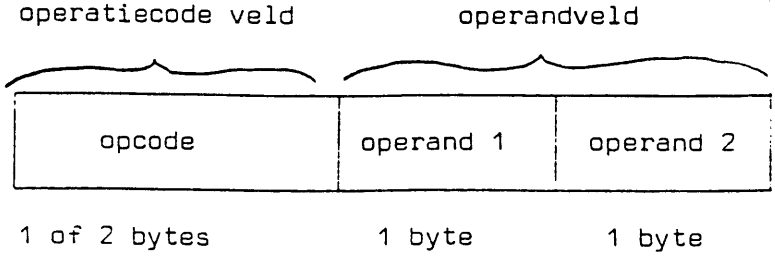
N.P.Looije

Het getal FFFF in hexadecimaal stelt inderdaad het binair adres 1111 1111 1111 1111 voor en komt overeen met het adres van de geheugenplaats 65535 (2^{16}-1).
De benaming van de verschillende adresseringsmethodes is niet dezelfde bij alle constructeurs wat aanleiding kan geven tot verwarring.
De adressering beperkt zich natuurlijk niet alleen tot geheugenplaatsen. Het is eveneens mogelijk bepaalde registers van de microprocessor aan te spreken of te adresseren of eenvoudig de te verwerken data (gegevens) aan te duiden.
Het is meestal hetzelfde register van de microprocessor dat gebruikt wordt als adresseringsregister. Onder adresseringsregister verstaan we het register dat het adres bevat van de te adresseren geheugencel.
Men spreekt over een programmateller (PC : *programcounter*) wanneer het adresseringsregister verwijst naar een instructie die zich in het programmageheugen bevindt.
Betreft het de adressering in het werkgeheugen (data geheugen) dan spreekt men ook van een basisregister (*base pointer*).
In adresseringstechnieken worden eveneens indexregisters gebruikt. De inhoud van het indexregister dient opgeteld te worden bij de inhoud van het basisregister om het eigenlijke (physische) adres te vinden.
In het werkgeheugen is meestal bij de multi-chip processoren, een zone gereserveerd om als stapelgeheugen (*stack*) gebruikt te worden. In het stapelgeheugen kan de microprocessor gegevens onderbrengen die nodig zijn om bijvoorbeeld een programma te kunnen verder zetten na een onderbreking (*interrupt*) of bij het verwerken van subroutines.
Als adresseringsregister van de stack gebruikt men bijvoorbeeld de stapelwijzer (SP : *stack pointer*).
Een instructie bestaat uit een deel dat de aard van de bewerking aangeeft, operatie code veld of kortweg opcode genoemd, meestal gevolgd door een deel dat aangeeft waarop de bewerking betrekking heeft, operandveld genoemd. Dit veld kan meerdere operanden bevatten. De operand kan een geheugen of input/output adres zijn, een register of ook een getal.
In het algemeen bestaat een instructie aldus uit 1, 2, 3 of 4 bytes :

1 byte : voor instructies zonder operand waarbij de operand(en) in het opcodeveld begrepen is (zijn).



2 bytes : voor instructies met 1 byte opcode en 1 byte operand
3 bytes : voor instructies met 1 byte opcode en 2 bytes operandveld
4 bytes : voor instructies met 2 bytes opcodes en 2 bytes operandveld



Als belangrijkste adresseringswijzen onderscheiden we impliciete, directe, indirecte, onmiddellijke, relatieve en geïndexeerde adressering.

waarbij tevens onderlinge combinaties mogelijk zijn zoals bv. geïncexeerd indirect enz.
In hetgeen volgt bespreken we deze methodes met telkens een aanduiding van een instructievoorbeeld voor de vier processor families. In de mate van het mogelijke zullen we de verschillende namen geven die gebruikt worden voor eenzelfde adresseringsmethode. Vermelden we tenslotte dat we ons hier beperken tot de 8 bits processoren. Voor de aanvullende adresseringsmethoden bij de 16 bits processoren verwijzen we naar de studie van de desbetreffende processoren.

1.5.1. Impliciete adressering (*implied*)

Bij de impliciete adressering is de instructielengte 1 byte zodat de volledige informatie voor de instructieverwerking hierin is opgenomen. Hierbij merken we een onderscheid tussen inherente (*implied*) adressering en register adressering. Bij inherente adressering is de operand overbodig, bijgevolg is de bewerking volledig bepaald door de combinatie van acht bits in het opcodeveld.

Voorbeelden

Bij 8085 STC $(37)_{Hex}$ (*set carry*)
De overdrachtsflipflop (*carry flag*) wordt op 1 gezet
(1→CY)

Bij 6800 SEC $(0D)_{Hex}$ (*set carry*)
(1→CY)

Bij Z80 SCF $(37)_{Hex}$ (*set carry flag*)
(1→CY)

Bij 6500 CLC $(18)_{Hex}$ (*clear carry*)
(0→CY)

Bij de registeradressering is de naam van een register (paar) of van meerdere registers opgenomen in de operand(en). Ook hier is de instructielengte slechts 1 byte lang, zodat zowel opcode als operand(en) gevormd worden door een 8 bits woord. Dergelijke adressering is interessant bij instructies met betrekking tot veel gebruikte registers zoals de accumulator(en) of registers van algemeen nut zoals de programmatellers (PC), de stapelwijzer (SP), kladblokregisters, indexregisters enz.
Ook rekenkundige en logische instructies die uiteraard gebeuren met de inhoud van de accumulator, maken meestal gebruik van deze vorm van adressering. Hierbij is de accumulator inherent geadresseerd en wordt beschouwd als een tweede operand.

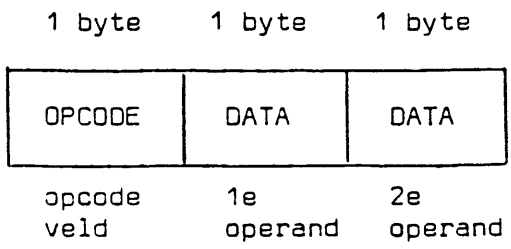
Voorbeelden

Bij 8085	MOV	r_1, r_2	De inhoud van register 2 wordt verplaatst naar register 1. $(r_2) \rightarrow r_1$
	ADD	r	De inhoud van register r wordt opgeteld met de inhoud van de accumulator en het resultaat komt in de accumulator. (A)+(r)→A Het register r vormt de eerste operand en de accumulator de tweede. Deze is niet genoemd in de instructie en is inherent.

PCHL (HL *to stackpointer*)
De inhoud van het registerpaar H(igh) L(ow) wordt in de programmateller gebracht
(HL)→PC

1.5.2. Onmiddellijke adressering (*immediate*)

Bij onmiddellijke adressering zijn de gegevens (*data*) die door de instructie dienen verwerkt te worden, in de opdracht zelf opgenomen. Ze volgen onmiddellijk na de instructiecode als tweede eventueel derde byte. Sommige auteurs spreken alleen van onmiddellijke adressering als de gegevens slechts 8 bits lang zijn (1 operand).
Voor 16 bits gegevens spreken ze van *extended immediate adressing* (uitgebreide onmiddellijke adressering) (2 operanden).



Door deze vorm van adresseren wordt geen geheugenruimte bespaard maar verhoogt de verwerkingssnelheid.

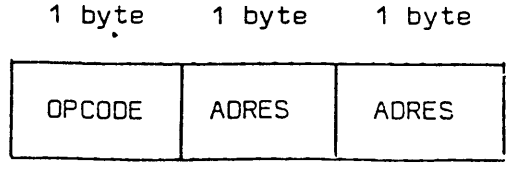
Voorbeelden

Bij 8085 MVI r,d8 (*move immediate*)
Het 8 bits datawoord (d8) wordt in het register gebracht.
d8→r

LXI SP,d16 (*load registerpair immediate*)
Het 16 bits woord (d16) komt in de stack pointer.
d16→SP

1.5.3. Directe adressering (*direct*)

Bij de directe adressering bevat de operand het adres van de te verwerken gegevens. Met 1 byte kunnen 256 plaatsen geadresseerd worden van (00)00 tot (00)FF. De hoogste byte van deze 256 adressen is (00) waardoor men ook spreekt over adressering in pagina nul (*zero page*).
Indien het operand veld bestaat uit 2 bytes kunnen alle 64 K plaatsen geadresseerd worden zodat hiervoor ook de benaming uitgebreide directe adressering (*extended direct*) gebruikt wordt of soms ook absoluut direct.



Merken we ook hier op dat in machinetaal (hex notatie) bij de 8085, de Z80 en de 6500 eerst de laagste byte van het adres geschreven wordt, gevolgd door de hoogste byte, terwijl dit omgekeerd is voor de 6800.

Voorbeelden

Bij 8085 STA 2000H (*store accumulator*)
De inhoud van de accumulator wordt in geheugenadres $(2000)_{Hex}$ geplaatst
(*extended*)
(A)→M

1.5.4. Indirecte adressering (*indirect*)

Bij de indirecte adressering wordt in de operand niet het adres zelf maar de plaats (register) aangegeven waar het adres zich bevindt.
Het register(paar) dat het adres bevat kan zich zowel binnen de microprocessor als in het geheugen bevinden.
Bij de 8085 processor bevindt dit registerpaar zich in de processor (HL registerpaar). We spreken dan van register indirecte adressering, bijvoorbeeld :

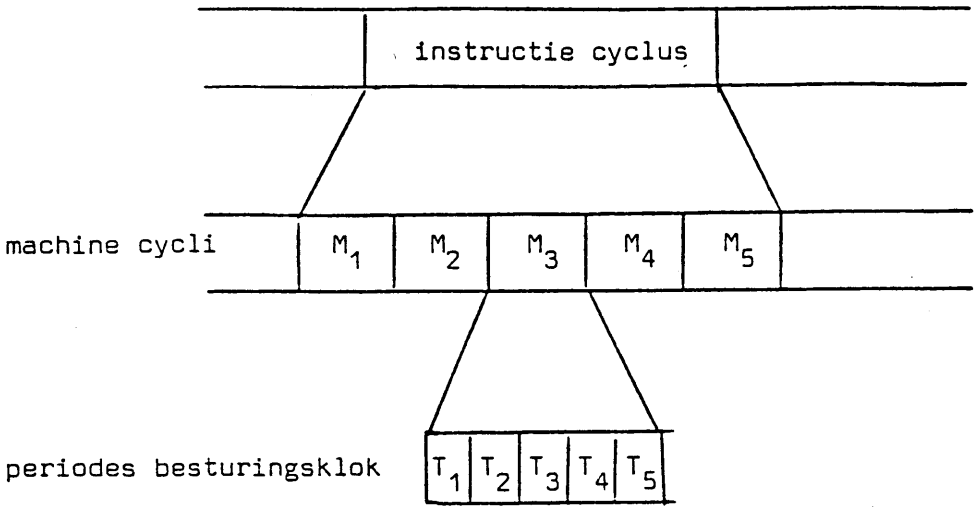
MOV M,A (*move A to M*)
De accumulatorinhoud wordt in het geheugen gebracht met de plaats aangeduid door de inhoud van het registerpaar HL.

Deze instructie is 1 byte lang zodat de eigenschappen dezelfde zijn als deze van de register adressering onder paragraaf 1.5.1.
Het registerpaar HL moet echter ook geladen worden.
Bij de Z80 wordt dit LD r,(HL).
Bij de 6500 bevindt het register zich in het uitwendig geheugen. Voor het bepalen van de geheugenplaats zijn 2 extra bytes nodig.

1.6. Samenstelling van een instructie in fuctie van de tijd

Om de werking van de microprocessor te begrijpen, is het van het grootste belang te weten hoe een programma instructie behandeld wordt.
Een programma instructie of kortweg instructie is ergens opgeborgen in het programmageheugen. De behandeling van de instructie gebeurt in verschillende fasen waarvan de eerste het oproepen van de instructie is.
Onder oproepen (FETCH) verstaan we het feit dat de instructie uit het geheugen in de centrale verwerkingseenheid gebracht wordt voor behandeling.
De eerste byte van een instructie bevat de opcode en wordt steeds in de instructiedecoder van de CPU gebracht.
Bij een multibyte instructie (operand(en) volgt (volgen) nog één of twee oproep bewerkingen (FETCH) waarbij de data in tijdelijke registers wordt opgeslagen. Daarna volgt de uitvoeringsfase (EXECUTE).
Het geheel bestaande uit het oproepen (FETCH) en het uitvoeren (EXECUTE) van een instructie noemen we een instructie cyclus.
Een instructiecyclus is samengesteld uit één of meer machinecycli (oproepen, uitvoeren). Elk van de machine cycli zelf wordt afgehandeld in verschillende periodes van de besturingsklok van de centrale verwerkingseenheid.
Zowel het aantal machinecycli als het aantal klokperiodes is verschillend van microprocessor tot microprocessor.
Specifiek voor de 8080 microprocessor van INTEL is bijvoorbeeld de onderverdeling van een instructiecyclus in 1 tot 5 machinecycli.

Ook de onderverdeling van elke machinecyclus in 3 tot 5 periodes van de besturingsklok, is specifiek voor de 8080 microprocessor.



Naargelang het type instructie kan de behandeling minimum 4 en maximum 18 periodes van de besturingsklok duren. Als we weten dat deze klok een frequentie van bijvoorbeeld 2MHz heeft, dus een periode van 0,5µs dan is de minimum duur van een instructie 2µs en de maximum duur 9µs. Om de totale duur van een programma te schatten, bekomt men meestal een aanvaardbaar resultaat als men een gemiddelde instructieduur van 4µs aanneemt bij een besturingsklok van 2MHz.
Bij het type 8080A1 is de maximum klokfrequentie 3MHz wat overeenkomt met een periode van 330ns. De minimum instructieduur is dus 1,3µ.
Bij het type 8085A is de maximum klokfrequentie 3,125 en bij het type 8085A2 5MHz.
Bij het type 8085 hebben sommige instructies in de eerste machinecyclus (eerste FETCH) 6 klokperiodes in plaats van 5 ; bijvoorbeeld PCHL 6xT in plaats van 5xT.
Andere instructies hebben een machinecyclus minder zoals bijvoorbeeld bij voorwaardelijke sprongopdrachten, wanneer aan de voorwaarde niet voldaan is en niet moet gesprongen worden, bijvoorbeeld JZ (JUMP ON ZERO) 3xM (10xT) bij de sprongvoorwaarde en 2xM (7xT) indien de accumulator ≠ 0.
Bij de 8080 is dit steeds 3xM of 10T.
De minimum toegelaten klokfrequentie is 500KHz, wat overeenkomt met T = 2µs, dit wegens de dynamische technologie.
Bij de 6500 is de minimum klokfrequentie 20KHz met een maximum van 2MHz. De instructietijd varieert van 1µs tot 3µs.
Bij de 6800 is de maximum klokfrequentie 1MHz met een minimum van 100kHz. De instructietijd varieert van 1 tot 6µs. De instructies worden in 2 tot 12 kloktijden behandeld.
Bij de 68A00 is de maximum frequentie 1,5MHz en bij de 68B00 2MHz.
Bij de Z80 is de minimum klokfrequentie 5kHz met een maximum voor het type Z80 van 2,8MHz en voor het type Z80A van 4,5MHz.
De instructietijd varieert van 1 tot 5,75µs. De instructies kunnen tot 4 bytes lang zijn en vereisen tot 23 klokimpulsen. De lage waarde van de klok is te wijten aan het feit dat deze processor van het statisch type is. Enkele inwendige registers zijn echter dynamisch zodat een minimum klok vereist is.
Alhoewel de klokfrequentie van de 6800 en de 6500 lager is dan bij de andere types, kunnen instructies soms sneller verwerkt worden wegens het gebruik van de directe adresseringsmethode (*zero page*) waardoor steeds 1 byte minder nodig is.

HELP!

The commonest theme of your letters was "help!". I will start by giving a list of answers:

For Eddie Spavin:You disable the BREAK key by POKE #5F,#84. This unfortunately disables all the other keys. To avoid that a user's machine code routine to read the key-board is needed. So if writing in BASIC, put the following code at addresses #200-#218. POKE #5F,#84 at the beginning of the program and then when some input is required POKE #5F,#C4 and CALLM #200. The machine code waits for a key to be pressed (but not

```
0200 CD BB D6 CA 00 02 FE FF CA 00 02 CD 95 D6 32 20
0210 02 C9
```

BREAK) and then stores the ASCII code in location #200. As soon as the input is in, POKE #5F,#84 again. A typical subroutine which replaces INPUT A$ with GOSUB 1000 would be:

```
1000 PRINT "?";:A$=""
1010 POKE #5F,#C4:CALLM #200:POKE #5F,#84
1020 CHAR=PEEK(#220):IF CHAR=13 THEN RETURN
1030 A$=A$+CHR$(CHAR):GOTO 1010
```

The disassembly for the code is:

```
0200 CD BB D6   CALL :D6BB   Call GETC routine:returns key in A
0203 CA 00 02   JZ   :0200   If no key (i.e.0) call it again
0206 FE FF      CPI  :FF     Is break key pressed (returns 255)
0208 CA 00 02   JZ   :0200   If so, call GETC again
020B CD 95 D6   CALL :D695   Print character in A
020E 32 30 02   STA  :0220   Store character in #220
0211 C9         RET          Return to BASIC
```

The code is located in the envelope buffer, so if your program uses envelopes you'll have to stick it somewhere else.

Eddies other questions were: how do you change the baud rate on the RS232? Answer:POKE #FF05 with the following hex numbers.

```
#81-110bd    #82-150bd    #84-300bd    #88-1200bd
#90-2400bd   #A0-4800bd   #C0-9600bd (default)
```

Those numbers generate one stop bit. (the norm) If using two stop bits, subtract #80 from each number. Other baud rates would only be available by user-written software. How do you mix text and graphics? Answer:use the DAInamic FGT package to "draw" text. How do you get super-hi-res. Use the MODE 7 and MODE 8 programs in DAInamic 13.

For Dave Fortune:To read the paddle buttons you PEEK #FD00 and then IAND the result with #10 or #20. Here is a short demo.

```
10 A%=PEEK(#FD00) IAND #30
20 A%=A% SHR 4:PRINT A%
30 WAIT TIME 25:GOTO 10
```

This program when run will print a number every half-second 1 if Paddle 2 button pressed, 2 if Paddle 1:3 if both pressed

Dave also asks members for a simple subroutine to print variables with two decimal places only thereafter. Obviously N=INT(N*100)/100 will do the rounding down (../100+0.5 to round up) but the problem is that the DAI converts 0.01 to 0.09 into scientific notation. Can anyone supply a routine to overcome this, please?

HELP!

Jerry Counsell is looking for (a)quick efficient methods of writing graphs to the screen, (b)a method of dumping text and graphics to an Epson printer, (c)the location of the "keyboard-readout", (d)methods of doing very high resolution arithmetic. (e)help with use of the DAI WP, and assembler/loader package.

Well,(a)the best answer would appear to be use of FGT package, in conjunction with a BASIC program. If FGT annotation is used, then (b)any DAI to Epson screen dump program will dump graphics with text, as the text "is" graphics when printed by FGT. (c)By the "keyboard readout", I assume Jeremy means where is the memory mapped location of the keyboard. This is at #FFF1 (Bits 0-6 only) but much more useful is the keyboard buffer which is at #2BA-#2BD, and stores the last four keys pressed. The address of the last key pressed is calculated as one less than the contents of #2BE/#2BF. High resolution arithmetic is easy for large integers, but I assume that Jeremy requires arithmetic on real numbers. I don't have a routine that does this:can any member oblige? Re: the final point, again I know how to use the assembler/loader and the Word Processor, and will oblige by answering specific questions. However on the more general question of "how do you use them?" perhaps someone else could help? Hopefully, I will be able to supply translated program notes at some stage.

Geoff Hawkins has the DAI Disc system and asks how to use the system to open files and print and input from disc. In brief, the answer is to POKE #131,3 and execute normal PRINT statements to write to disc, and POKE #135,3 and use the GETC routine to read from disc. However, if anyone has got some standard disc filing routines written, or indeed any programs written with the DOS in mind, perhaps they could let me have them or send them to Geoff via me (so that I foot the airmail postage cost). Geoff also wants to see some simple machine code so that he can get started and indeed many others of you have written to me in this vein. I am trying to include simple routines where I can, but the problem with a full-blown set of articles on the subject is that most of what I would say is covered much better in texts on the subject. I will repeat that 8080 programming is mainly the same on <u>any</u> 8080 machine, so the best thing for me to do is take up DAI-specific entry-points etc, assuming a general knowledge of 8080 assembler.

Eddie Ryzman, another professional user, asks how the text editor can be used to store and edit non-DAI BASIC programs getting round the DAI syntax checker. Input to the editor is by POKE #131,2 (the input will perhaps be from keyboard, RS232 disk etc.) and then the editor is entered by RST 5 DATA #2A (EF 2A). (BASIC "EDIT" won't work as this clears the edit buffer first). The program lines are removed from the editor by POKE #135,2, but the return area for programs must not be the command line interpreter. POKE #131,3 will therefore have to be used to pass the lines to disk, or a user-defined routine (see the "lower case" program elsewhere in this issue as an example of a user-defined output routine. Eddie is also looking for a machine code WP, that stores text as an addressed block rather than as an array, and is therefore more flexible. I did start work on this, but for one reason and another, never got round to finishing it. Can any member help (they are welcome to the source listings of the bits I have done)

HELP!

John Mitchell has had problems positioning the FGT characters on the screen: this is done by setting the X and Y variables in the BASIC control subroutine that accompanies every FGT program. You must use this subroutine to call FGT:don't just CALLM #300,A$. I have written a short demonstrator which you will find elsewhere in this newsletter. John has also had repeat trouble with some keys on his keyboard. Lack of key debounce is a known bug in BASIC V1.0 but from what John describes, he could have a sticky key. I had a sticky 'I' key at one point (excessive wear!?) and I found that removing the keycap and cleaning the exposed switch helped. John offers a hint for AMD 9511 owners. Some programs run too fast(!) with the maths chip installed, viz tunes etc, so to disable it, POKE #D4,0. Dave Blackadder asks for more info. on the 136 colours promised in the last issue. I am not able to give you anything further yet, but as they say, watch this space.

HINTS & TIPS

An abbrieviated issue this time round because of shortage of space (well, time actually!). Those of you who have sent me useful information will see their contributions published next time, which will hopefully be a couple of months away, and certainly not as long as last time. POKE #131,3 for non-disc users is pretty interesting. It jumps to #2DD-#2DF to direct its output, where a user defined jump can be made. I have stuck a bit of code in #200, the envelope buffer (very useful place for short routines if you aren't using envelopes) as follows:

```
0200 B7 FE 41 DA 0D 02 FE 5B D2 0D 02 C6 20 EF 03 C9
```

and then altered the #2DD-#2DF code to C3 00 02 and a user-defined routine exists. You put the code in with the S command. Now do that, while a BASIC program is in memory and type POKE #131,3 and LIST.

TAILPIECE

I have noticed that a firm called Ikon who sell DCR's for the BBC Micro under the name 'Hobbit' appear to be selling DCR tapes at about £20 a box, several pounds cheaper than any other supplier (unless of course, you know different!) Also a firm called Work Force in Luton sell Epson ribbons at £10 for four. These are without cartridges, so you have to refit them to the old cartridge yourself.

Next time I hope to include more on machine code, including the printer RHJ routines, a selection of translated documentation from published programs, and maybe a couple of printer and pssibly an RGB Monitor review. Keep the stuff coming! Professional users: please pass on your experiences too, especially in the field of hardware expansion/connections. Anyone know where to get 1v composite video out?

Dave Atherton

HIGH SPEED DATA LOADER

(HSDL)

Heeft U ook wel eens het probleem dat U juist een lang programma van band hebt ingelezen en dat de DAI zich dan "ophangt" en U weer opnieuw moet beginnen, of dat U eerst koffie kunt gaan drinken wanneer U een groot programma van zo'n 12K van een audio band moet inlezen en dan terug kerende moet ervaren dat er iets fout ging en men een tweede kop koffie kan gaan drinken of dat op zondagmorgen Uw zoontje U wakker komt maken omdat hij space invader wilt spelen maar het programma er niet in kan krijgen.

Nu Wij wel. Dan hebben Atari, Commodore 64 en Philips P2000 bezitters en nog vele anderen het een stuk eenvoudiger. ROM module met het gewenste programma erin, een of twee commando's en het loopt. Zoiets zou ook wel wat zijn voor de DAI dachten wij. Het idee was daar en na veel praten en knutselen met hard- en software hadden we een doosje in elkaar wat we High Speed Data Loader noemden.

Waarom high speed? Wel, we kunnen data inlezen met een snelheid van iets meer dan 10 kilobyte/sec. Dit betekent dat b.v. SPL niet langer duurt dan één seconde. Dus geen tijd meer voor koffie. De soft/hardware is zo gemaakt dat de HSDL eventueel zonder problemen met Uw DCR's gebruikt kan worden (disc weten we niet omdat we deze niet bezitten).

De HSDL is in staat om basic, MLP en gecombineerde programma's in te lezen. Al naar gelang van wat er in de EPROMS staat.

Wij zijn uitgegaan van een EPROM kaartje waarop 4 EPROM's gaan van de volgende types:

2716, 2732 en 2764 wat ons een maximale programma grootte geeft van resp. 8, 16 en 32 Kbyte.

Dit EPROM kaartje komt met een connector op de HSDL welke via een flatkabel is aangesloten op de DCE connector van de DAI.

Als het is aangesloten, is de rest eenvoudig n.l. zet de DAI aan en type onder basic RDL1 ... 4, return. Wanneer de cursor naar de volgende regel springt is het programma ingelezen. Voor een basic of kombinatie van basic met MLP is de "RUN" commando voldoende en het loopt.

Voor MLP is het iets moeilijker. Ga eerst naar UT, type Z3 en dan Gxxxx de locatie waar het programma start en dit loopt ook. Omdat alle programma's geen 8 resp. 16 K groot zijn hebben we het mogelijk gemaakt om meerdere programma's op een kaartje te zetten.

Iedere EPROM is bereikbaar met het commando RDL 1...4. 1 is de eerste EPROM en 4 is de 4de EPROM.

.../...

Wanneer een programma groter is dan wat in één of meerdere EPROMS past en men roept deze EPROM(S) aan dan krijgt men de foutmelding "NOT AVAILABLE". De ruimte in een EPROM welke niet gebruikt wordt door het aanwezige programma gaat helaas verloren. Alleen selectie van gehele EPROMS is mogelijk.

De high speed data loader is opgebouwd uit de volgende delen:-

1) EPROM kaartje op de X bus waarop een EPROM met besturingssoftware.

2) De data loader waarin de hardware zit, hierop zit een connector waarin het EPROM kaartje gestoken wordt.

3) Het betreffende EPROM kaartje met programma(s).

Er zijn momenteel twee prototypes gebouwd en wij zijn bezig de definitieve print te tekenen.

Bij voldoende belangstelling willen we een kleine productie beginnen en met Dainamic bespreken of het mogelijk is om programma's in PROM of EPROM in hun assortiment op te nemen, eventueel kompleet met printplaatje en connector.

Heeft men eenmaal de basisset dan is alleen het EPROM printje met programma's noodzakelijk. Het belangrijkste is wat gaat dat kosten. We hebben een voorlopige schatting gemaakt en komen op een bedrag dat rond de ƒ 275,-- zal zijn, voor een basisset bestaande uit:-

EPROM met operating software;
high speed data loader;
EPROM kaartje met een demo programma in een EPROM.
Voor eventuele extra's zoals xbus print, flatcable met connectoren, EPROM kaartjes, etc... moeten we nog prijzen vaststellen.

Na een gesprek wat wij hadden met Dianamic hebben we nog een paar opmerkingen en een aantal vragen aan U.

1) Gebruikt U Uw DAI voor een bepaald doel b.v. SPL, INA, Word Processing, VIDITEL of iets anders waarvoor maar 1 of 2 programma's gebruikt worden dan is een HSDL hier uitermate geschikt voor. Eenmaal in EPROM en geen "LOAD" problemen meer.

2) Dainamic levert op één cassette (normaal of DCR) meestal meerdere programma's voor een relatief lage prijs. Dit is met EPROMS niet mogelijk, dus zijn de kosten hiervan hoger dan van welke tape ook. Waarop dan onze vragen:
   a) Welke programma's zou U in EPROM willen zien?
   b) Wilt U de losse EPROMS of kompleet gemonteerd?
   c) Wat wilt U ervoor betalen? Buiten de vaste kosten van de EPROMS, aanmaak etc... komen hierbij de kosten van de programma's zelf (vaste kosten hardware ± ƒ 80,-- 16KByte). Daarom geeft ons een reeële prijs indicatie. Kijkt U maar eens naar de programma's van de speel computers.

Mocht U interesse hebben voor een High Speed Data Loader laat het ons weten. Daarom vragen wij; stuur ons een briefkaartje met al Uw vragen en wensen, dan kunnen wij in één van de komende DAInamics daarop verder ingaan.

H. KOP,
Burg. Smeetsstraat 39,
6151 GM MUNSTERGELEEN NL.

H. RISON,
Luxemburgstr. 17,
6164 BS GELEEN NL.

```
1     REM
2     REM
3     REM [[[[[[[[       M A R Z I A N E T T O     ]]]]]]]]
4     REM
5     REM
6     REM ... A FIRST BASIC (SIC) STEP TO MOVE
7     REM A GREEN ALIEN ON THE SCREEN ...
8     REM
9     REM
10    REM Gianni Uliana
11    REM Via Zuccoli 40
12    REM 00137  ROMA  ITALY
15    COLORG 0 15 0 0
20    DIM A%(15.0)
30    FOR I=0.0 TO 15.0
40    READ A%(I)
50    NEXT
55    PRINT CHR$(12):PRINT :PRINT "MARZIANETTO":PRINT
56    PRINT " USE THE CURSOR KEYS TO MOVE A GREEN ALIEN "
58    WAIT TIME 200
60    MODE 3
70    B%=#BFEB
75    H=H+8.0:IF H=16.0 THEN H=0.0
77    SOUND 1 1 1 2 FREQ(50.0+200.0*H)
80    FOR I=0.0 TO 7.0
90    POKE B%-1,#50
100   POKE B%,A%(I+H)
110   B%=B%-46
120   NEXT
125   SOUND OFF
130   B%=B%+368
140   Z%=B%
150   C=GETC:IF C=0.0 THEN GOTO 150
160   IF C=19.0 AND X=19.0 THEN GOTO 150
170   IF C=18.0 AND X=0.0 THEN GOTO 150
180   IF C=17.0 AND Y=15.0 THEN GOTO 150
190   IF C=16.0 AND Y=0.0 THEN GOTO 150
200   IF C=19.0 THEN X=X+1.0:B%=B%-2
210   IF C=18 THEN X=X-1.0:B%=B%+2
220   IF C=17 THEN Y=Y+1.0:B%=B%-368
230   IF C=16 THEN Y=Y-1.0:B%=B%+368
240   FOR I=0.0 TO 7.0
250   POKE Z%,#0
260   Z%=Z%-46
270   NEXT
280   GOTO 75
500   DATA #00,#18,#3C,#5A,#7E,#42,#C3,#00
510   DATA #18,#18,#3C,#DB,#7E,#42,#42,#42
```

CATALOGUS NEDERLAND

CODE	TITEL	PRIJS (in guldens)
G1	GAMES COLLECTION 1	22,50
G2	GAMES COLLECTION 2	22,50
G3	GAMES COLLECTION 3	22,50
G4	GAMES COLLECTION 4	45,--
G5	GAMES COLLECTION 5	22,50
G6	GAMES COLLECTION 6	42,50
G7	GAMES COLLECTION 7	42,50
G8	GAMES COLLECTION 8	42,50
G9	GAMES COLLECTION 9	42,50
G10	GAMES COLLECTION 10	42,50
G11	GAMES COLLECTION 11	42,50
CTP	CENTIPEDE	35,--
DRI	DRIVER	35,--
SI	SPACE INVADER	45,--
SUI	SUPER INVADER	35,--
DAPA	DAIPANIC	45,--
DN	DAINIBBLE	45,--
ACR	ACROBATES	35,--
CH	SARGON CHESS	85,--
TK1	TOOLKIT 1	55,--
TK2	TOOLKIT 2	55,--
TK3	TOOLKIT 3	55,--
TK4	TOOLKIT 4	55,--
TK5	TOOLKIT 5	55,--
DNA	DNA ASSEMBLY PACK	62,50
SPL	SPL MACRO-ASSEMBLER	62,50
DTP	DAI TINY PASCAL	55,--
DTX	DAINATEXT	115,--
FGT	FAST GRAF TEXT	55,--
FGTA	FGT-APPLICATIONS	55,--
SFGT	SUPER FGT	55,--
GT	GRAPHIC TABLET	55,--
GH	GRAFISCHE HULP	27,50
ML	MAILING LIST	55,--
PE1	PRIMARY EDUCATION 1	55,--
SE1	SECUNDARY EDUCATION 1	55,--
SE2	SECUNDARY EDUCATION 2	55,--
WP	WORDPROCESSOR	55,--
EGT	ENGLISH-GERMAN TRAINER	55,--
JR	MICRO'S-ONDERWIJS	57,50
TT1	TAAL 1	42,50
FB	FAMILIEBUDGET	27,50
W3	WISKUNDE 3	42,50
F1	FYSICA 1	42,50
DD	DAI DEMO + BASIC TUTOR	27,50
T80	TAPE 80-81	47,50
N10	NEWSLETTER 10	27,50
N11	NEWSLETTER 11-12	37,50
N13	NEWSLETTER 13-14-15	37,50
M1	MUSIC COLLECTION 1	17,--
M2	MUSIC COLLECTION 2	17,--
M3	MUSIC COLLECTION 3	17,--

	HARDWARE & PUBLICATIONS	
PCS	DAIpc SCHEMATICS	47,50
BOD	BEST of DAInamic	27,50
SNG	SUPER NOISE GENERATOR	85,--
NC	NEW CHARACTER GENERATOR	55,--
DCE	DCE-INTERFACE-CARDS	140,--
N8	NEWSLETTERS 8-13 (1982)	27,50
	LIDMAATSCHAP	50,--

Op DCR zijn de programma's 8 gulden duurder. De banden kunt U bestellen door het

bedrag over te maken op onderstaande rekening. Bij de medelingen moet U de code van de door U gewenste band(en) vermelden en of U de programma's op AUDIO of DCR

wilt hebben. Als U geld over maakt voor het lidmaatschap wilt U dan vermelden: LIDMAATSCHAP en het jaartal?

GIRO: 4083817
t.n.v.: J.F. van Dunne'
HOFLAAN 70
3062 JJ ROTTERDAM
TEL: (010) 144802

UNIEKE aanbieding voor DAI-gebruikers !!!

Een raspaardje onder de processorkaarten voor definitieve toepassingen:

UNIDATA

Om voor al uw ideeen m.b.t. automatisering van regelingen en/of besturingen een oplossing te vinden is er nu een universele processorkaart ontwikkeld.

Deze eurokaart heeft standaard de volgende eigenschappen:

- **8085 CPU (6.144Mhz)**
- **interne interrupt controller**
- **1/4 K RAM + 2 sockets voor uitbreiding naar 1 1/4 K RAM**
- **2 EPROM-sockets voor 2716 en/of 2732 (dus maximaal 8 K ROM)**
- **programmeerbare timer**
- **universele I/O aansluitbus (DAI-DCE compatible gemaakt)-- 8155**
- **20 mA current-loop (voor direkte aansluiting teletype TTY e.d.)**
- **RS 232 (voor aansluiting modem of bijv. DAI computer)**
- **printbanen voor eigen hardware uitbreiding**
- **e'e'n voedingsspanning (+5V bij current-loop)**
- **uitgebreide nederlandstalige handleiding**

EXTRA leverbaar:

TTY-monitor (1 K in 2716 en SDK-85 compatible) inclusief gegevens belangrijke entry points.

De prijs ? ? ?

Vanwege de aanmaak van proffesionele eurokaarten kan deze processorkaart bij VOLDOENDE belangstelling geleverd worden voor:

UNIDATA-1.....fl 189,-
TTY-monitor...fl 45,- prijzen incl BTW

T.Groeneveld
J.P.Wiersmawei 7
8915 HT leeuwarden
DAI-gebruiker

Indien u belangstelling heeft en deze kaart wilt bestellen, dan dient u schriftelijk kontakt op te nemen met:

2. DE DCE-MICRO-COMPUTER

Het doel van deze cursus is, om ervaring op te doen met het programmeren van een micro-computer.
De DCE-micro-computer is daarbij een hulpmiddel. Op deze computer kunt u n.l. uw programma's invoeren, testen en debuggen.
Het is niet de bedoeling, dat u in deze cursus de werking van elk onderdeel van de DCE-micro-computer leert kennen.
Wilt u er alles van weten, dan verwijzen we naar de literatuur.
In de komende lesen bespreken we alleen datgene wat u van de DCE-micro-computer moet kennen om uw programma's te kunnen invoeren, testen en debuggen.

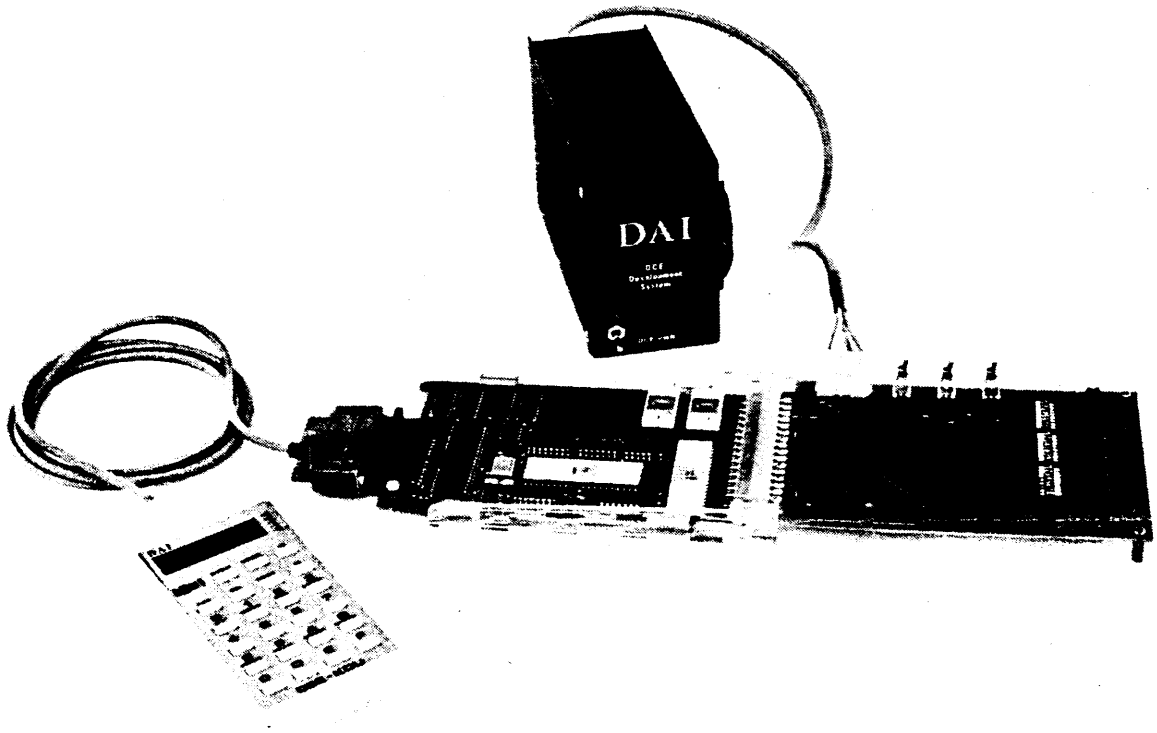


fig.1

De micro-computer uit fig. 1 bestaat, ruimtelijk gezien, uit de volgende delen.
1. De DCE-print (DCE = Digital Control Element).
   Op deze print bevinden zich de 8080, de I/O-modules en het centrale geheugen.
2. De extender (verlengstuk) waarop zich een aantal schakelaars en LED's bevinden om input- en outputsignalen op te wekken of na te bootsen.
3. Het toetsenbord en de display, met behulp waarvan de communicatie tussen de programmeur en de micro-computer verloopt.
4. De voeding.

In deze les gaan we uitgebreid in op de DCE en de extender. Het toetsenbord behandelen we in de volgende les. (Op de voeding gaan we in deze cursus niet in.).

3. BLOKSCHEMA VAN DE DCE

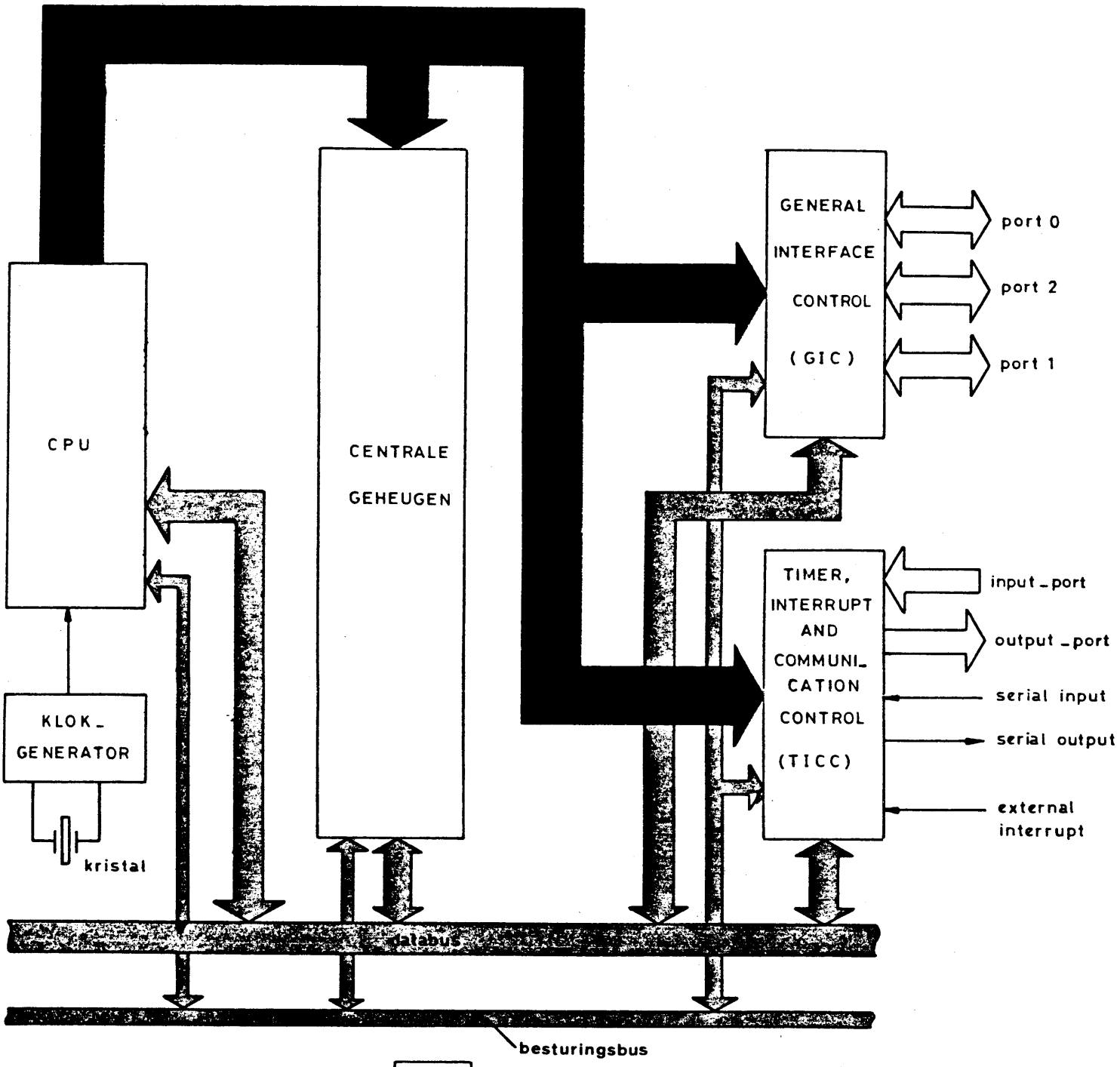


fig.2

In fig. 2 is het blokschema van de DCE weergegeven.
Rondom de micro-processor bevinden zich de volgende eenheden.
1. Het centrale geheugen, bestaande uit RAM's en EPROM's.
2. De klokgenerator.
3. De General Interface Controller (GIC).
4. De Timer, Interrupt and Communications Controller (TICC).

In de volgende paragrafen bespreken we de GIC en de TICC en geven we een overzicht van de bezetting van de geheugencapaciteit.
Wat betreft de werking van de CPU, de klokgenerator en het centrale geheugen, verwijzen we naar de betreffende lessen uit de basiscursus.

4. MEMORY MAPPED I/O

De DCE werkt volgens het systeem van memory mapped I/O. Dit houdt in, datelke in- en outputpoort en elk register dat zich buiten de CPU bevindt, wordt behandeld als een 8-bits geheugenwoord.

Vraag 1: Bij memory mapped I/O zijn poortnummers vervangen door .....

De poortnummers, waarmee in- en outputpoorten werden geadresseerd, worden vervangen door "normale" geheugenadressen.
IN- en OUT-instructies komen ook niet meer voor. Ze worden vervangen door LOAD, STORE en MOVE-instructies.

Hardware-technische gezien moeten wel enkele kunstgrepen worden uitgehaald.
Een I/O-module zal nl. niet meer worden aangestuurd door IN- en OUT-signalen, maar door READ, WRITE en CHIP-SELECT-signalen.

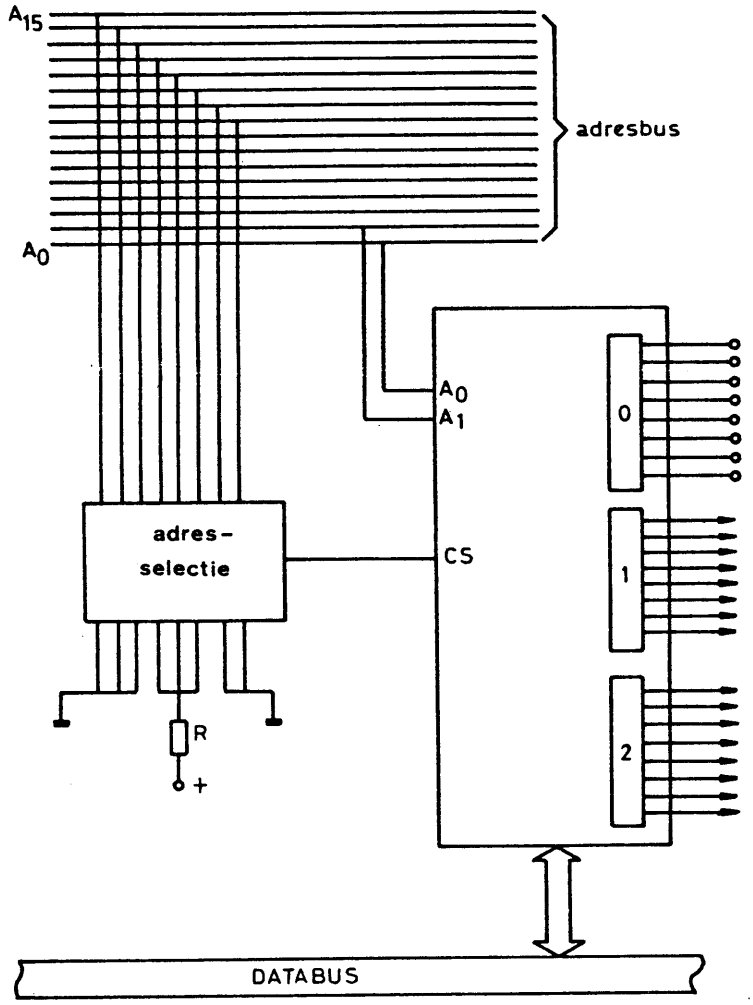


fig.3

4 Antw.1: geheugenadressen. 20-01

In fig. 3 is weergegeven hoe we een I/O-module als geheugenbouwsteen kunnen opnemen.
De I/O-module in fig. 3 bestaat uit 2 output-poorten en 1 input-poort. Voor de selectie van deze poorten zijn 3 ingangen aanwezig: 2 adres-ingangen en 1 chip-select (CS)-ingang. Wanneer we nl. de chip (de I/O-module) hebben geselecteerd, moeten we alleen nog 1 van de poorten binnen die chip adresseren. Dit kan m.b.v. 2 bits. Hiermee zijn immers 4 combinaties mogelijk, nl. $00_2 = 0$; $01_2 = 1$; $10_2 = 2$ en $11_2 = 3$.
De chip selecteren we, door b.v. de 8 hoogste adresbits m.b.v. een comparator te vergelijken met een vooraf ingestelde waarde.

Vraag 2: In fig. 3 worden de 8 hoogste adresbits vergeleken met $00111100_2/00011100_2$.

In fig. 3 is dit de waarde $00011100_2 = 1C_{16}$.

Vraag 3: De adressen van de 3 poorten in fig. 3 zijn: ……$_{16}$, ……$_{16}$ en ……$_{16}$.

De adressen van de 3 poorten in fig. 3 zijn dus $1C00_{16}$, $1C01_{16}$ en $1C02_{16}$.

We zien nu meteen een groot nadeel opdoemen van memory mapped I/O.
De I/O-module in fig. 3 neemt nl. een veel groter adresbereik in beslag dan de 3 adressen die voor de poorten nodig zijn.
Immers, bij alle adressen die beginnen met 1C.... wordt de I/O-module aangesproken. We mogen dus stellen, dat de adressen 1C04 t/m $1CFF_{16}$ niet meer voor andere doeleinden gebruikt mogen worden.
In de meeste gevallen is dit niet zo'n groot bezwaar, omdat de volle 64k-geheugenruimte toch niet nodig is.
Is het benutten van alle geheugencapaciteit wel nodig, dan moeten we werken met grotere comparators. In fig. 3 zullen we dan de adresbits A_7 t/m A_2 ook moeten vergelijken met een vooraf ingestelde waarde.

Een groot voordeel van memory mapped I/O is, dat bij een input- en output-operatie de bestemming resp. de bron niet meer persé de accumulator behoeft te zijn.
Vullen we b.v. registerpaar H,L met $1C01_{16}$, dan kunnen we de inhoud van register D direct naar output-poort 1 brengen met de instructie MOV M,D.

20-01 Antw.2: 00011100_2. Antw.3: $1C00_{16}$; $1C01_{16}$ en $1C02_{16}$.

5. DE GIC

De DCE beschikt over 2 I/O-modules, de GIC en de TICC.
De GIC (General Interface Controller) wordt gevormd door de 8255 van Intel.
Deze IC beschikt over 3 poorten, welke onafhankelijk van elkaar als input-poort of als output-poort kunnen fungeren.

M.b.v. een 8-bits woord, dat we naar de z.g. mode-controller van de GIC zenden, bepalen we in welke mode (input of output) we elk van de 3 poorten schakelen.
M.a.w. de mode van de 3 poorten is programmeerbaar.
(Een betere aanduiding voor de GIC zou dus eigenlijk zijn: Programmable Peripheral Interface (PPI)).

In fig. 4 is een blokschema van de GIC weergegeven.
De mode-controller van de GIC heeft, net als de 3 poorten, een adres.
De adressen zijn in fig. 4 tussen haakjes weergegeven.

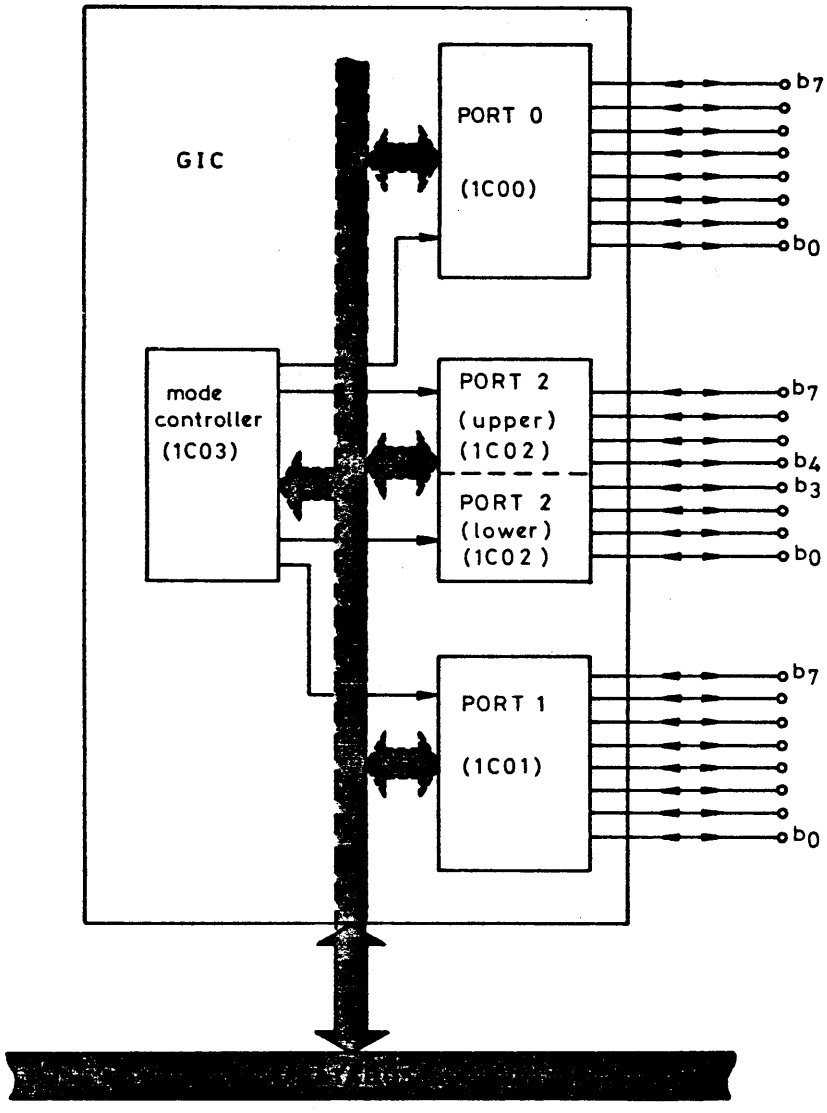


fig.4

Vraag 4: Het adres van de mode controller is .....$_{16}$.

We kunnen de mode controller dus beschouwen als een geheugenwoord met adres $1C03_{16}$.
De mode controller decodeert het 8-bits codewoord dat hij ontvangt, en schakelt dan de GIC-poorten in de gewenste mode.
We merken nog op dat port 2, wat betreft de mode, in 2 delen kan worden gesplitst, nl. port 2 upper, waartoe b_4 t/m b_7 behoren en port 2 lower, gevormd door b_0 t/m b_3.

In fig. 5 is weergegeven hoe het codewoord, waarmee we de modes van de 3 poorten aangeven, is opgebouwd.

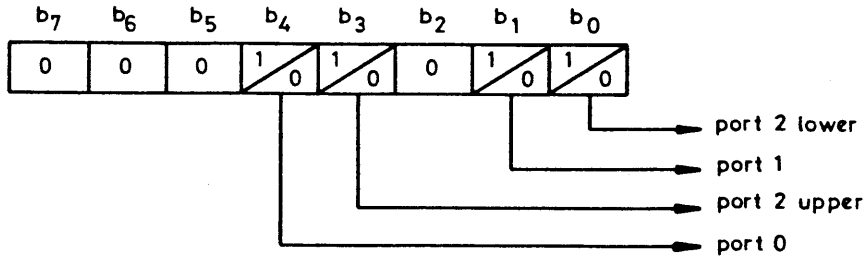


fig.5

We zien, dat de bits b_7, b_6, b_5 en b_2 niet worden gebruikt. Deze bits zijn altijd 0.
Met b_0 geven we de mode van port 2 (lower) aan. Is $b_0 = 0$, dan werkt port 2 (lower) als output-poort. Is $b_0 = 1$, dan werkt port 2 (lower) als input-poort.
Op dezelfde manier geven we met b_1, b_3 en b_4 de modes van resp. port 1, port 2 (upper) en port 0 aan.
Een 0 komt dus overeen met output; een 1 komt overeen met input.

Voorbeeld 1:

Gevraagd:
Welk codewoord moeten we naar adres $1C03_{16}$ sturen om de poorten van de GIC in de volgende modes te schakelen ?
port 0 = input-poort
port 2 (upper en lower) = input-poort
port 1 = output-poort

Oplossing:
b_0 moet 1 worden, omdat port 2 (lower) als input-poort moet werken.
b_1 moet 0 worden, omdat port 1 als output-poort moet werken.
b_3 en b_4 moeten beide ook 1 worden, omdat port 2 (upper) en port 0 als input-poort moeten werken.
Het code-woord wordt dus $00011001_2 = 19_{16}$.
M.b.v. de 2 nevenstaande instructies kunnen we dit woord naar de mode controller zenden.

```
MVI  A,19
STA  1C03
```


20-01 Antw.4: $1C03_{16}$.

Voorbeeld 2:

Gegeven:
De inhoud van de accumulator is 13_{16}.

Gevraagd:
In welke modes komen de poorten na uitvoering van de instructie STA 1C03 ?

Oplossing:
Met de instructie STA 1C03 sturen we de inhoud van de accumulator naar adres $1C03_{16}$, d.w.z. naar de mode controller.
Deze ontvangt dan het woord $13_{16} = 00010011_2$. b_0 en b_1 zijn 1, zodat port 2 (lower) en port 1 in de input mode komen.
$b_3 = 0$, zodat port 2 (upper) in de output mode komt.
$b_4 = 1$, zodat port 0 in de input mode komt.

Opmerking:

Behalve de "normale" input- en output-mode is m.b.v. de 8255 ook handshake I/O en bi-directionele I/O mogelijk. Deze modes worden bepaald door de bits b_7, b_6, b_5 en b_3.
In deze cursus maken we van deze mogelijkheden echter geen gebruik, zodat deze bits in onze gevallen altijd 0 zijn.

SAMENVATTING 1

1. De DCE (Digital Controller Element) bestaat uit:
   a. een 8080 micro-processor.
   b. RAM's en EPROM's.
   c. 2 I/O-modules, de GIC en de TICC.

2. De DCE werkt volgens memory mapped I/O.
   Dit houdt in, dat elk register buiten de CPU als geheugenwoord wordt beschouwd.

3. GIC betekent General Interface Controller.
   De GIC beschikt over 3 I/O-poorten die m.b.v. een codewoord onafhankelijk als input- of outputpoort geschakeld kunnen worden.

MICRO FAST GRAPHICS

```
PAGE 01     MICRO FAST GRAFFICS  *** DAINAMIC ***

002
003
004                 ************************************************
005                 * THIS UTILITY PROVIDES A FAST DRAWING FACILITY  *
006                 * APPLICABLE FOR EACH MODE EXEPT MODE 0.         *
007                 * IT IS TAILERED TO BE USED TOGETHER WITH BASIC. *
008                 * THIS MEANS THAT PARAMETERS ARE PASSED VIA BA-  *
009                 * SICS SYMBOL TABLE. YOU HAVE TO SPECIFY 4 VARIA-*
010                 * BLES I.E. - X (ABSIS OF THE GRAFFICS SCREEN)   *
011                 *           - Y (ORDINATE OF THE GRAFFICS SCREEN)*
012                 *           - C (DESIRED COLOR )               *
013                 *           - E (ENTRY OF A TABLE WITH THE NE-  *
014                 *           CESSARY INFO TO DRAW A PICTURE)    *
015                 *               SEE FUTHER.                    *
016                 *  X,Y SPECIFY THE STARTPOSITON OF THE PICTURE ON *
017                 *  THE SCREEN                                  *
018                 * BEFORE CALLING THE PROCEDURE ONE MUST PASS THE  *
019                 * NECESSARY INFORMATION REFFERING TO THE MENTIONED*
020                 * VARIABLES.                                     *
021                 * THIS CAN BE DONE IN THIS WAY (I = INTEGER)     *
022                 * I=VARPTR(X)+2:POKE#2F0,I MOD 256:POKE#2F1,I/256 *
023                 * I=VARPTR(Y)+3:POKE#2F2,I MOD 256:POKE#2F3,I/256 *
024                 * I=VARPTR(C)+3:POKE#2F4,I MOD 256:POKE#2F5,I/256 *
025                 * I=VARPTR(E)+2:POKE#2F6,I MOD 256:POKE#2F6,I/256 *
026                 * PICT=#300, TO ACTIVATE PROCEDURE BY CALLM PICT .*
027                 * THE USER MUST CREATE HIS SPECIFIC TABEL CONTAI- *
028                 * NING ALL NECESSARY INFORMATION TO DRAW THE RE-  *
029                 * QUIRED PICTURE (S).                            *
030                 * THE TABLE CONSISTS OF ONE OR MORE ENTRIES, ONE  *
031                 * FOR EACH PICTURE. EACH ENTRY CONSISTS OF SEVERAL*
032                 * ELEMENTS(ONE FOR EACH DOT,DRAW OF FILL FUNCTION)*
033                 * EACH ELEMENT CONSISTS OF 5 BYTES: M,x1,y1,x2,y2 *
034                 * THE LAST ELEMENT HOWEVER CONSISTS OF ONLY ONE   *
035                 * BYTE (#FF) DENOTING THE END OF THE ENTRY.       *
036                 * - M = OPERATOR: #1E/DOT, 21/DRAW, 24/FILL       *
037                 * - x1,x2 ABCIS IN A 256x256 MATRIXS (IF MODE 6)  *
038                 * - y1,y2 ORDINATES IN THE SAME MATRIX            *
039                 *   x AND y ARE OFFSETS OF X AND Y                *
040                 *   NOTE THAT x2,y2 ARE DON'T CARE IN CASE OF THE *
041                 *   DOT OPERATION.                                *
042                 ************************************************rt
043
044
045                 *
046                 * SYMBOL TABLE #2F0...2F8
047                 *
048                         ORG    :2F0
049                                          **** USER DEFINABLE
050 02F0 0000       X       DBL    0         +VARPTR(X-COORD)+2
051 02F2 0000       Y       DBL    0         +VARPTR(Y-COORD)+3
052 02F4 0000       COLOR   DBL    0         +VARPTR(COLOR)+3
053 02F6 0000       ENTRY   DBL    0         +VARPTR(ENTRY)+2
054                                          **** SYSTEM TEMPORARY STORAG
```

```
055 02F8 00          KLEUR   DATA  0
056                  *
057                  * START PROGRAM
058                  *
059                          ORG   :300
060 0300 E5          START   PUSH  H          PUSH ALL
061 0301 D5                  PUSH  D
062 0302 C5                  PUSH  B
063 0303 F5                  PUSH  PSW
064                  *
065                  * GET ENTRY
066                  *
067 0304 2AF602              LHLD  ENTRY      LOAD ADDRESS OF
068 0307 56                  MOV   D,M        TABLE ENTRY (2 BYTES )
069 0308 23                  INX   H
070 0309 5E                  MOV   E,M
071                  *
072                  * GET COLOR
073                  *
074 030A 2AF402              LHLD  COLOR      GET COLOR
075 030D 7E                  MOV   A,M
076 030E 32F802              STA   KLEUR      STORE TEMPORARY
077 0311 EB                  XCHG
078                  *
079                  * PICK UP OPERATOR (FILL,DRAW,DOT)
080                  *
081 0312 7E          MFGTO   MOV   A,M        FIRST ELEMENT OF ENTRY
082 0313 23                  INX   H          INDICATES FILL, DOT, DRAW
083 0314 FEFF                CPI   :FF
084 0316 CA5903              JZ    END
085 0319 324D03              STA   LABEL+1    FILL IN OPERATOR
086                  *
087                  * PICK UP X1,Y1,X2,Y2
088                  *
089 031C 0603                MVI   B,3        REPEAT 2 TIMES FOLLOWING
090                                           LOOP
091 031E 05          MFGT1   DCR   B
092 031F CA4103              JZ    MFGT2
093                  *
094                  * CALCULATE X1 = X + x1 AND X2 = X + x2
095                  *
096 0322 5E                  MOV   E,M        LOAD x
097 0323 23                  INX   H
098 0324 1600                MVI   D,0
099 0326 E5                  PUSH  H
100 0327 D5                  PUSH  D
101 0328 2AF002              LHLD  X          LOAD CONTENTS OF X
102 032B 56                  MOV   D,M        MS BYTE
103 032C 23                  INX   H
104 032D 5E                  MOV   E,M        LS BYTE
105 032E E1                  POP   H
106 032F 19                  DAD   D          CALCULATE X + x
107 0330 EB                  XCHG
```

```
108 0331 E1                     POP   H          CURRENT TABEL POINTER
109 0332 D5                     PUSH  D          PUSH X1 OR X2 ON STACK
110                     *
111                     * CALCULATE Y1 = y1 + Y OR Y2 = y2 + Y
112                     *
113 0333 4E                     MOV   C,M        GET y1 OR y2
114 0334 23                     INX   H
115 0335 E5                     PUSH  H
116 0336 2AF202                 LHLD  Y          LOAD CONTENTS OF Y
117 0339 7E                     MOV   A,M        (ONLY ONE BYTE)
118 033A 81                     ADD   C          CALCULATE Y+y
119 033B 4F                     MOV   C,A
120 033C E1                     POP   H
121 033D C5                     PUSH  B          PUSH Y1 OR Y2 ON STACK
122 033E C31E03                 JMP   MFGT1
123                     *
124                     * PERFORM OPERATION
125                     *
126 0341 E3             MFGT2   XTHL             FETCH Y2
127 0342 45                     MOV   B,L
128 0343 E1                     POP   H
129 0344 D1                     POP   D          FETCH X2
130 0345 E3                     XTHL             FETCH Y1
131 0346 4D                     MOV   C,L
132 0347 E1                     POP   H
133 0348 E3                     XTHL             FETCH X1
134 0349 3AF802                 LDA   KLEUR      FETCH COLOR
135 034C EF             LABEL   RST   5          ACTIVATE DRAW, FILL OR DOT
136 034D 21                     DATA  :21        1E:DOT;21:DRAW;24:FILL
137 034E E1                     POP   H          FETCH TABELPOINTER
138 034F DA5503                 JC    ERROR      CHECK FOR ERROR
139 0352 C31203                 JMP   MFGTO
140 0355 3E45           ERROR   MVI   A,'E'      IN CASE AN ERROR IS DE-
141 0357 EF                     RST   5          TECTED, A 'E' IS PRINTED
142 0358 03                     DATA  3
143 0359 F1             END     POP   PSW        POP ALL
144 035A C1                     POP   B
145 035B D1                     POP   D
146 035C E1                     POP   H
147 035D C9                     RET
148 035E                        END
```

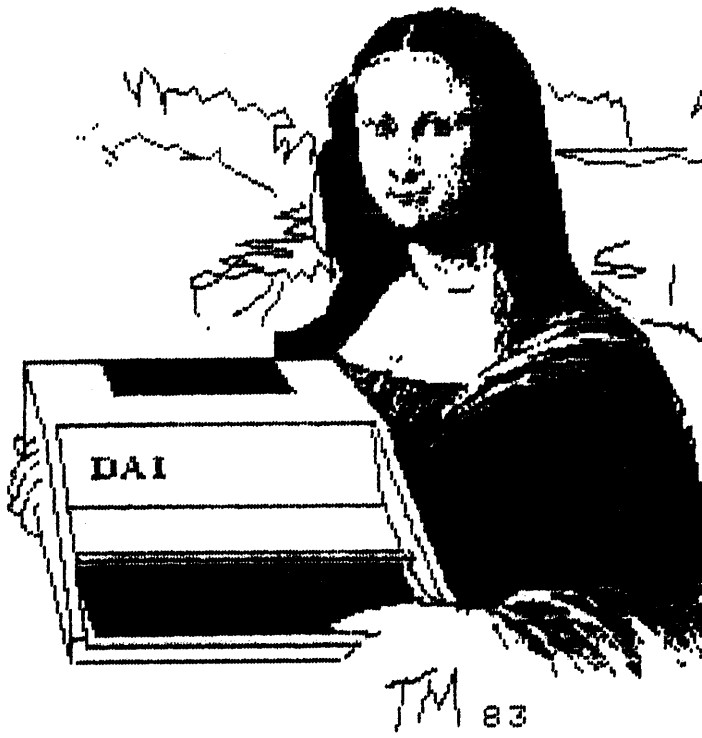
```
****************************
* S Y M B O L   T A B L E *
****************************

COLOR  02F4   END    0359   ENTRY  02F6   ERROR  0355
KLEUR  02F8   LABEL  034C   MFGTO  0312   MFGT1  031E
MFGT2  0341   START  0300   X      02F0   Y      02F2
```

```
0300 E5 D5 C5 F5 2A F6 02 56 23 5E 2A F4 02 7E 32 F8
0310 02 EB 7E 23 FE FF CA 59 03 32 4D 03 06 03 05 CA
0320 41 03 5E 23 16 00 E5 D5 2A F0 02 56 23 5E E1 19
0330 EB E1 D5 4E 23 E5 2A F2 02 7E 81 4F E1 C5 C3 1E
0340 03 E3 45 E1 D1 E3 4D E1 E3 3A F8 02 EF 21 E1 DA
0350 55 03 C3 12 03 3E 45 EF 03 F1 C1 D1 E1 C9
```

video graphics

TOMISLAV MIKULIC
DAKICEV TRG 3/5
YU - 41000 ZAGREB
YUGOSLAVIA
tel. (041) 535 490



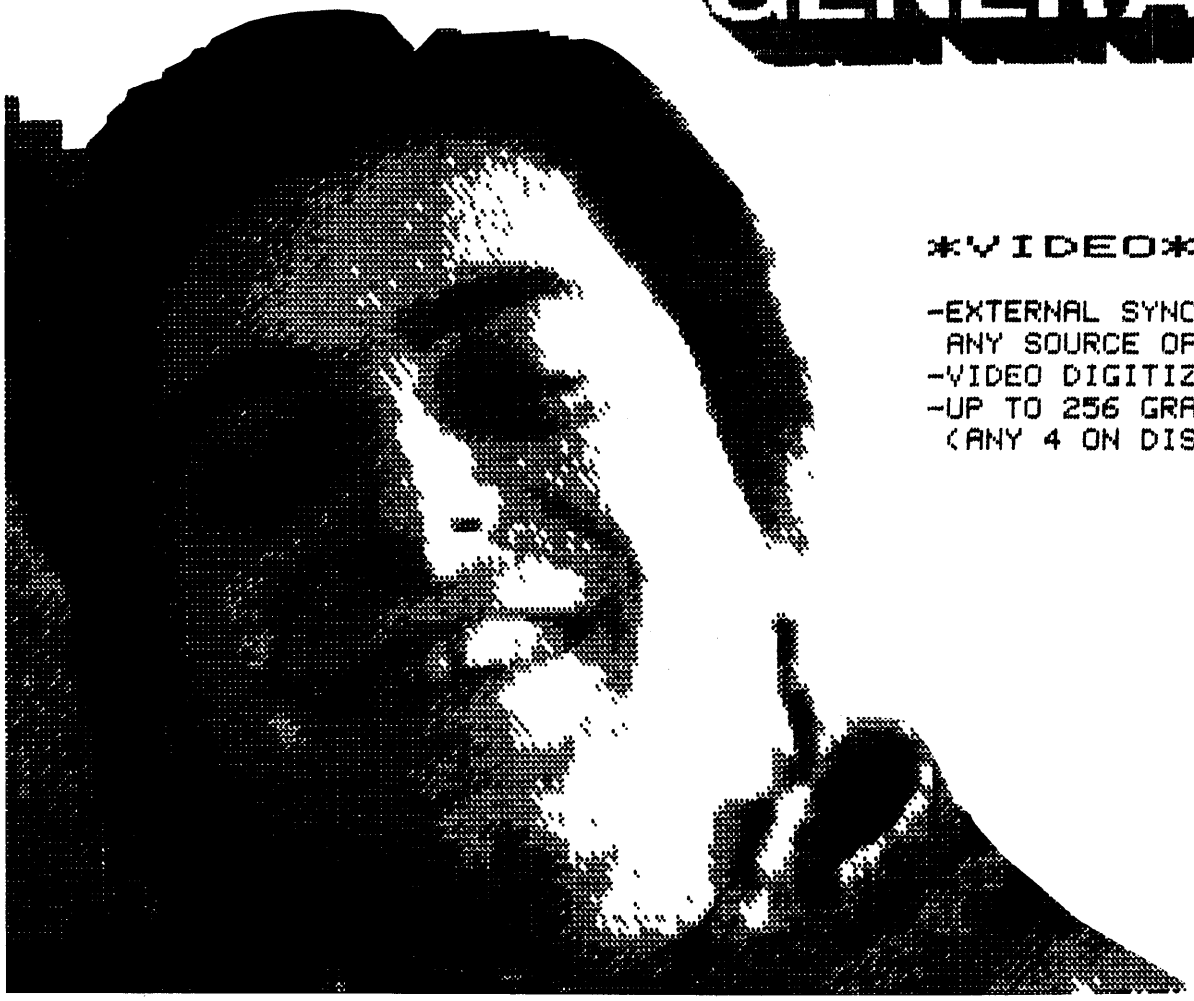
GRAPHICS TABLET

-SCREEN MENU OVERLAY
-HAND DRAWING
-ZOOM, SHRINK
-SPECIAL EFFECTS
-FILL AREA
-GEOMETRIC SHAPES
-USER BRUSHES, etc.

CHARACTER GENERATOR

-FULL GRAPHICS OVERLAY
-ANY STYLE FONTS
-HELVETICA IN VARIOUS SIZES
-USER DEFINED SYMBOLS
-VARIOUS SHADOWS AND EDGES

CHARACTER GENERATOR



VIDEO

-EXTERNAL SYNCHRONIZATION WITH
 ANY SOURCE OF COMPOSITE VIDEO
-VIDEO DIGITIZER
-UP TO 256 GRAY LEVELS
 (ANY 4 ON DISPLAY)

IF YOU ARE SERIOUS ABOUT DAI VIDEO GRAPHICS CONTACT ME ON ABOVE ADDRESS

8080 instructionset

INSTRUCTION	FUNCTION		HEX
MOVE GROUP		reg	A B C D E H L M
MOV A, reg	(A)←(reg)		7F 78 79 7A 7B 7C 7D 7E
MOV B, reg	(B)←(reg)		47 40 41 42 43 44 45 46
MOV C, reg	(C)←(reg)		4F 48 49 4A 4B 4C 4D 4E
MOV D, reg	(D)←(reg)		57 50 51 52 53 54 55 56
MOV E, reg	(E)←(reg)		5F 58 59 5A 5B 5C 5D 5E
MOV H, reg	(H)←(reg)		67 60 61 62 63 64 65 66
MOV L, reg	(L)←(reg)		6F 68 69 6A 6B 6C 6D 6E
MOV M, reg	(M)←(reg)		77 70 71 72 73 74 75 --
ACCUMULATOR GROUP			
ADD reg	(A)←(A)+(reg)	*	87 80 81 82 83 84 85 86
ADC reg	(A)←(A)+(reg)+(CY)	*	8F 88 89 8A 8B 8C 8D 8E
SUB reg	(A)←(A)-(reg)	*	97 90 91 92 93 94 95 96
SBB reg	(A)←(A)-(reg)-(CY)	*	9F 98 98 9A 9B 9C 9D 9E
ANA reg	(A)←(A)∧(reg)	*	A7 A0 A1 A2 A3 A4 A5 A6
XRA reg	(A)←(A)∀(reg)	*	AF A8 A9 AA AB AC AD AE
ORA reg	(A)←(A)∨(reg)	*	B7 B0 B1 B2 B3 B4 B5 B6
CMP reg	(A)-(reg)	*	BF B8 B9 BA BB BC BD BE
INCREMENT/DECREMENT REGISTER			
INR reg	(reg)←(reg)+1	**	3C 04 0C 14 1C 24 2C 34
DCR reg	(reg)←(reg)-1	**	3D 05 0D 15 1D 25 2D 35
REGISTER PAIR GROUP		rp	B D H SP PSW
INX rp	(rp)←(rp)+1		03 13 23 33 --
DCX rp	(rp)←(rp)-1		0B 1B 2B 3B --
LDAX rp	(A)←((rp))		0A 1A -- -- --
STAX rp	((rp))←(A)		02 12 -- -- --
DAD rp	(H,L)←(H,L)+(rp) ***		09 19 29 39 --
PUSH rp	((SP)-1)←(rh),((SP)-2)←(rl), (SP)←(SP)-2		C5 D5 E5 -- F5
POP rp	(rl)←((SP)),(rh)←((SP)+1), (SP)←(SP)+2		C1 D1 E1 -- F1 *
DIRECT ADDRESS GROUP			
LDA addr	(A)←(addr)		3A al ah
STA addr	(addr)←(A)		32 al ah
LHLD addr	(L)←(addr),(H)←(addr+1)		2A al ah
SHLD addr	(addr)←(L),(addr+1)←(H)		22 al ah
IMMEDIATE GROUP			
MVI A, data	(A)←data		3E dd
MVI B, data	(B)←data		06 dd
MVI C, data	(C)←data		0E dd
MVI D, data	(D)←data		16 dd
MVI E, data	(E)←data		1E dd
MVI H, data	(H)←data		26 dd
MVI L, data	(L)←data		2E dd
MVI M, data	(M)←data		36 dd
ADI data	(A)←(A)+data	*	C6 dd
ACI data	(A)←(A)+data+(CY)	*	CE dd
SUI data	(A)←(A)-data	*	D6 dd
SBI data	(A)←(A)-data-(CY)	*	DE dd
ANI data	(A)←(A)∧data	*	E6 dd
XRI data	(A)←(A)∀data	*	EE dd
ORI data	(A)←(A)∨data	*	F6 dd
CPI data	(A)-data	*	FE dd
LXI B, addr	(B)←ah,(C)←al		01 al ah
LXI D, addr	(D)←ah,(E)←al		11 al ah
LXI H, addr	(H)←ah,(L)←al		21 al ah
LXI SP,addr	(SP_h)←ah,(SP_l)←al		31 al ah

INSTRUCTION	FUNCTION		HEX
JUMP GROUP			
JMP addr	(PC)←addr		C3 al ah
JNZ addr	If Z=0, (PC)←addr		C2 al ah
JZ addr	If Z=1, (PC)←addr		CA al ah
JNC addr	If CY=0,(PC)←addr		D2 al ah
JC addr	If CY=1,(PC)←addr		DA al ah
JPO addr	If P=0, (PC)←addr		E2 al ah
JPE addr	If P=1, (PC)←addr		EA al ah
JP addr	If S=0, (PC)←addr		F2 al ah
JM addr	If S=1, (PC)←addr		FA al ah
PCHL	(PC_h)←(H), (PC_l)←(L)		E9
CALL GROUP			
CALL addr	(TOS)←(PC),(PC)←addr		CD al ah
CNZ addr	If Z=0, (TOS)←(PC),(PC)←addr		C4 al ah
CZ addr	If Z=1, (TOS)←(PC),(PC)←addr		CC al ah
CNC addr	If CY=0,(TOS)←(PC),(PC)←addr		D4 al ah
CC addr	If CY=1,(TOS)←(PC),(PC)←addr		DC al ah
CPO addr	If P=0, (TOS)←(PC),(PC)←addr		E4 al ah
CPE addr	If P=1, (TOS)←(PC),(PC)←addr		EC al ah
CP addr	If S=0, (TOS)←(PC),(PC)←addr		F4 al ah
CM addr	If S=1, (TOS)←(PC),(PC)←addr		FC al ah
	N.B. (TOS)←(PC) designates the following:- ((SP)-1)←(PC_h),((SP)-2)←(PC_l),(SP)←(SP)-2		
RETURN GROUP			
RET	(PC)←(TOS)		C9
RNZ	If Z=0, (PC)←(TOS)		C0
RZ	If Z=1, (PC)←(TOS)		C8
RNC	If CY=0,(PC)←(TOS)		D0
RC	If CY=1,(PC)←(TOS)		D8
RPO	If P=0, (PC)←(TOS)		E0
RPE	If P=1, (PC)←(TOS)		E8
RP	If S=0, (PC)←(TOS)		F0
RM	If S=1, (PC)←(TOS)		F8
	N.B. (PC)←(TOS) designates the following:- (PC_l)←((SP)),(PC_h)←((SP)+1),(SP)←(SP)+2		
RESTART GROUP			
RST 0	(TOS)←(PC),(PC)←00_{16}		C7
RST 1	(TOS)←(PC),(PC)←08_{16}		CF
RST 2	(TOS)←(PC),(PC)←10_{16}		D7
RST 3	(TOS)←(PC),(PC)←18_{16}		DF
RST 4	(TOS)←(PC),(PC)←20_{16}		E7
RST 5	(TOS)←(PC),(PC)←28_{16}		EF
RST 6	(TOS)←(PC),(PC)←30_{16}		F7
RST 7	(TOS)←(PC),(PC)←38_{16}		FF
ROTATE/CONTROL/SPECIAL GROUP			
RLC	(A_{n+1})←(A_n),(A_0)←(A_7),(CY)←(A_7)	***	07
RRC	(A_n)←(A_{n+1}),(A_7)←(A_0),(CY)←(A_0)	***	0F
RAL	(A_{n+1})←(A_n),(A_0)←(CY),(CY)←(A_7)	***	17
RAR	(A_n)←(A_{n+1}),(A_7)←(CY),(CY)←(A_0)	***	1F
NOP	No operation		00
HLT	Processor stopped until interrupt or reset		76
DI	Interrupts disabled		F3
EI	Interrupts enabled after next instruction		FB
XTHL	(L)←((SP)),(H)←((SP)+1)		E3
SPHL	(SP_h)←(H),(SP_l)←(L)		F9
XCHG	(H)←(D),(L)←(E)		EB
DAA	Decimal adjust accumulator	*	27
CMA	(A)←$\overline{(A)}$		2F
STC	(CY)←1	***	37
CMC	(CY)←$\overline{(CY)}$	***	3F
OUT port) IN port)	Not used in DCE Systems		D3 port DB port

DAI video graphics

SEND YOUR DRAWINGS TO DAINAMIC
EDITOR