```
1170  PRINT "sans conserver les valeurs"
1180  PRINT "soit n le nombre d'elements"
1190  PRINT "soit X les differentes valeurs du parametre"
1200  PRINT
1210  INPUT "nombre d'elements";N!:PRINT :PRINT "quelles sont les different
es valeurs"
1220  PRINT "appuyez sur RETURN apres chaque donnee"
1230  M!=0.0:V!=0.0:E!=0.0:S!=0.0
1240  FOR I!=1.0 TO N!:INPUT X!
1250  S!=S!+X!:V!=V!+X!*X!:NEXT
1260  M!=S!/N!
1270  PRINT :PRINT "pour la ponderation n presser N"
1280  PRINT "pour la ponderation n-1 presser I"
1290  G!=GETC:IF G!=0.0 THEN 1290
1300  IF G!=ASC("N") THEN 1330
1310  IF G!=ASC("I") THEN 1340
1320  IF G!<>ASC("N") AND G!<>ASC("I") THEN 1270
1330  V!=V!/N!-M!*M!:GOTO 1350
1340  V!=(V!-S!*S!/N!)/(N!-1.0)
1350  E!=SQR(ABS(V!)):PRINT "moyenne ";M!:PRINT SPC(4):PRINT "variance ";V!

1360  PRINT "ecart-type ";E!
1370  RETURN
1380  H=9:PRINT CHR$(12):PRINT "resolution des equations algebriques"
1390  PRINT "du 1er et du 2eme degre"
1400  PRINT :PRINT "formes canoniques:"
1410  PRINT TAB(5):PRINT "B*X+C=0"
1420  PRINT TAB(5):PRINT "A*X^2+B*X+C=0":PRINT
1430  PRINT "si 1er degre A=0"
1440  INPUT "VALEUR DE A ";A!:PRINT SPC(4):INPUT "DE B ";B!:PRINT SPC(4):IN
PUT "DE C ";C!
1450  IF A!=0.0 THEN 1550
1460  D!=B!*B!-4.0*A!*C!:IF D!<0.0 THEN 1510
1470  X1!=(-B!+SQR(D!))/2.0/A!:X2!=(-B!-SQR(D!))/2.0/A!
1480  PRINT :PRINT "deux racines reelles":PRINT "X1=";X1!;SPC(4);"X2=";X2!
1490  IF X1!=X2! THEN PRINT "racine double"
1500  RETURN
1510  D!=ABS(D!):PRINT :PRINT "deux racines complexes conjuguees"
1520  XA!=-B!/2.0/A!:XB!=SQR(D!)/2.0/A!
1530  PRINT "X1=";XA!;"+";XB!;"i";SPC(4);"X2=";XA!;"-";XB!;"i"
1540  RETURN
1550  IF B!=0.0 THEN 1570
1560  PRINT :PRINT "premier degre  X=  ";-C!/B!:RETURN
1570  IF C!=0.0 THEN PRINT :PRINT "equation indeterminee":RETURN
1580  PRINT :PRINT "IMPOSSIBLE":RETURN
```

Guus Knoops has problems with one of his BASIC-roms,
please contact him on 04951/31286 (The Netherlands)
if you have spare roms or another solution.

```
10    REM ***********************************************
20    REM *                                             *
30    REM *      OMLAAGROETSJEN OP EEN KWART CIRKEL     *
40    REM *                                             *
50    REM ***********************************************
60    PRINT F,DT,INT(X*10.0+0.5)/10.0,INT(Y*10.0+0.5)/10.0,INT(10.0*ACOS(COSA)*1
80.0/PI+0.5)/10.0
70    REM BRON/AUTEUR Thijs Berkx (n.a.v. FARADAY jrg.50 nr.6)
80    REM DATUM mei 1982
90    REM OPSLAG BAND nr --  CODE NK01
100   REM
110   REM ************SCHERM-OPMAAK***************************
120   REM
130   MODE 0:PRINT CHR$(12)
140   MODE 6A:COLORG 0 5 3 15
150   DRAW 75,5 275,5 5:DRAW 75,5 75,205 5
160   FOR N%=15 TO 205 STEP 10
170   DRAW 72,N% 75,N% 5:DRAW 71+N%,2 71+N%,5 5
180   NEXT N%
190   REM
200   REM ************INVOER v. PARAMETERS*****************
210   REM
220   INPUT "Wrijvingskoefficient f ";F:PRINT
230   INPUT "Tijdsinterval dT tussen stippen";DT:PRINT
240   REM
250   REM *************BEGINVOORWAARDEN********************
260   REM
270   REM Straal R=200 Äschermeenhedenü
280   REM Valversnelling: g = 10 Äm/(s*s)ü
290   REM Start in (0,200) met beginsnelheid 0 m/s
300   R%=200.0:X=0.0:Y=R%:VX=0.0:VY=0.0
310   REM
320   REM ***************BEREKENING***********************
330   REM
340   COSA=1.0-X/R%:SINA=1.0-Y/R%:V2=VX*VX+VY*VY
350   X1=X
360   REM
370   REM ****************X-RICHTING***********************
380   REM
390   AX=(10.0*SINA+V2/R%)*(COSA-F*SINA)
400   VX=VX+AX*DT
410   X=X+VX*DT
420   IF X<X1 THEN 540:REM Afbreekkonditie
430   REM
440   REM ****************Y-RICHTING*********************
450   REM
460   AY=-10.0+(10.0*SINA+V2/R%)*(SINA+F*COSA)
470   VY=VY+AY*DT
480   Y=Y+VY*DT
490   REM
500   REM ****************PLAATS STIP*****************
510   REM
520   DOT 75+INT(X+0.5),5+INT(Y+0.5) 3
530   GOTO 340:REM Volgend tijdsinterval
540   REM
550   REM ********UITVOER v.GEGEVENS v.EINDPUNT**********
560   REM
570   PRINT CHR$(12)
580   PRINT " f "," dT "," Xeind "," Yeind "," ALFAeind "
590   PRINT F,DT,INT(X*10.0+0.5)/10.0,INT(Y*10.0+0.5)/10.0,INT(10.0*ACOS(COSA)*1
80.0/PI+0.5)/10.0
600   END
```

SHAPES

Dear members,

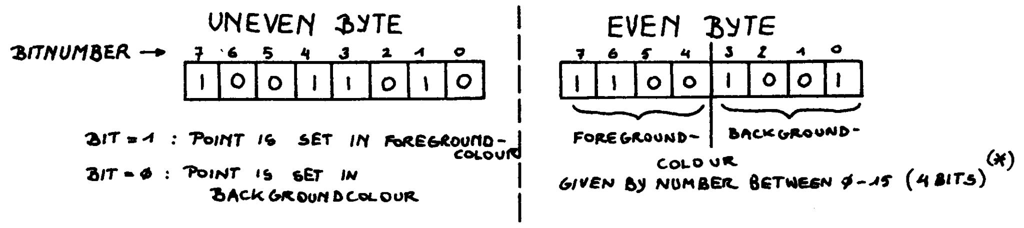
when I bought my DAI computer several years ago, and started to write BASIC programs, I found that even DAI basic is much too slow for fast-moving and sofisticated shapes (in games, for example). Here follows a mlp-routine that can be called from a BASIC program, and that places a shape at a desired position of the screen. The routine is placed in "MLP$" and thus it is protected against the graphic-modes, the basic program or other mlp-routines, as long as no "CLEAR" is executed. The control-byte interval of the mode used by the program has to be poked in "RES", and the beginadress of the table with shape-data must be poked in "TBL" and "TBL+1" (see program).

```
*IMPINT
 1Ø REM Before calling shape-routine :
 2Ø REM poke RES,line-interval of actual mode
 3Ø REM poke TBL,LSB (least significant byte) of the adress
    that points to the beginning of the data-table.
 4Ø REM poke TBL+1,MSB (most significant byte) of the adress
 5Ø REM callm USR,A where A = destination adress on the screen
1ØØ CLEAR 2ØØØ
11Ø MLP$="":FOR X=Ø TO #28:READ A:MLP$=MLP$+CHR$(A):NEXT
12Ø DATA #C5,#23,#23,#56,#23,#5E,#21,#ØØ,#ØØ,#7E,#B7,#CA,#ØØ
13Ø DATA #ØØ,#47,#23,#D5,#7E,#12,#1B,#23,#05,#C2,#ØØ,#ØØ,#D1
14Ø DATA #E5,#21,#ØØ,#ØØ,#EB,#CD,#1A,#DE,#EB,#E1,#C3,#ØØ,#ØØ
15Ø DATA #C1,#C9
16Ø V=VARPTR(MLP$):USR=PEEK(V)+PEEK(V+1)*256+1
17Ø RES=USR+#1C:TBL=USR+#7:AR=USR+#27:AL=USR+#11:AP=USR+#9
18Ø POKE USR+#C,AR IAND #FF:POKE USR+#D,AR SHR 8
19Ø POKE USR+#17,AL IAND #FF:POKE USR+#18,AL SHR 8
2ØØ POKE USR+#25,AP IAND #FF:POKE USR+#26,AP SHR 8
```

As you already know (I hope), on a graphic- screen (mode 1 through 6) a horizontal line is divided in groups of eight points or bits. The combination and colour of these points are stored in two bytes per eight points. The method of storage is not always the same (see article of F. Druijff in Dainamic 13). I'll explain it here very briefly :

In the modes with an uneven number (MODE 1-3-5, the sixteen-colour modes) the first (=uneven) byte contains binary the combination of eight points on the screen that are either in foreground (bit=1) or background (bit=Ø) colour, while the second (=even) byte selects which fore- and back- ground colours are used.

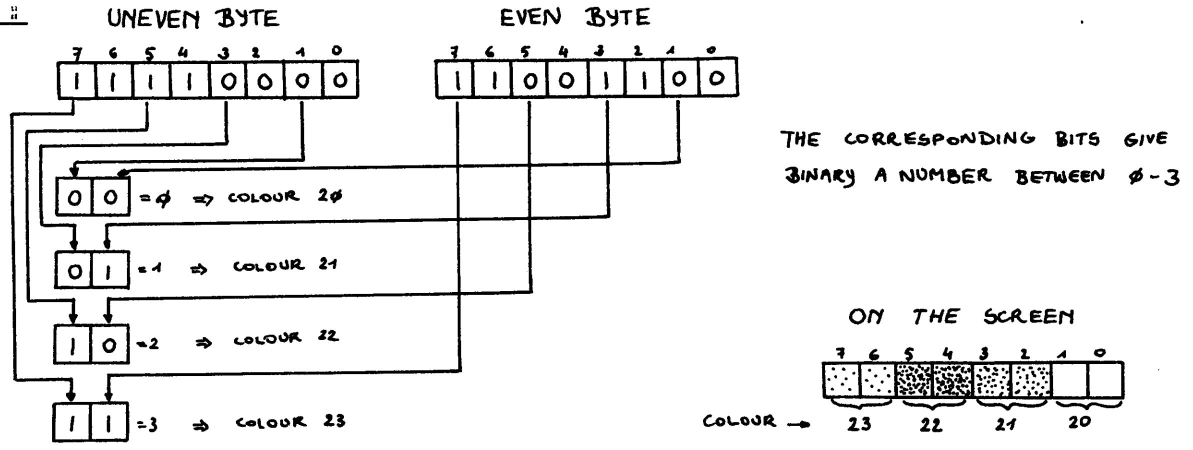
e.g. :


UNEVEN BYTE
BITNUMBER →
7 6 5 4 3 2 1 0
1 0 0 1 1 0 1 0
BIT = 1 : POINT IS SET IN FOREGROUND-COLOUR
BIT = Ø : POINT IS SET IN BACKGROUNDCOLOUR
EVEN BYTE
7 6 5 4 3 2 1 0
1 1 0 0 1 0 0 1
FOREGROUND-
BACKGROUND-
COLOUR
GIVEN BY NUMBER BETWEEN Ø-15 (4 BITS) (*)

In the modes with an even number (MODE 2-4-6, the four-colour modes), the combination of the corresponding bits of two subsequent bytes gives the number of the colour of that bit on the screen (colours 2Ø through 23).

(*) IN THIS EXAMPLE THE FOREGROUNDCOLOUR = COLOUR 12 (BINARY: 1100)
AND THE BACKGROUNDCOLOUR = COLOUR 9 (BINARY: 1001)

e.g. :



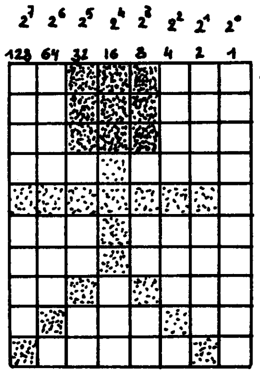
See articles of Louis Gidney (Dainamic 14, page 37-38) and N.P. Looije (Dainamic 12, page 248) or other articles concerning video-screen RAM-setup.

The format of the data, needed for this shape-routine is as follows :
Always start with the number of databytes on the current line, followed by the data itself.
So we get :

n,x1,x2,x3,...xn — The routine places the "n" bytes of data on the screen from left to right, jumps down one line and returns just underneath the first byte of the previous line. This is done until n=Ø ; then the routine returns to the BASIC program.

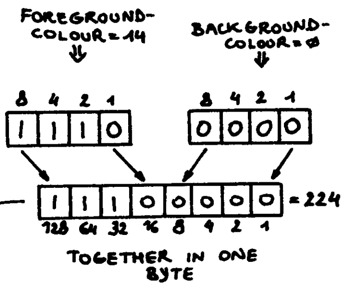
Notice that "A" (for CALLM USR,A) indicates the upper-left corner of the rectangle where the shape is placed in. The calculation of A is fairly simple : with X and Y as the coordinates of the upper-left corner of the shape and R as the line-interval of the control-bytes of the actual mode,
A=#BFEB-(YMAX-Y)*R-(X/8)*2 (NOT X/4!)
and A MOD 8 gives the number of bits you have to rotate the shape to place it on the exact bit position.

Now an example : let's search the shape data of the little man; we want his head in colour 1, the rest of his body in colour 14, the background colour must be Ø and we want to use mode 5 (16-colour mode).



THE SUM OF THE VALUE OF THE BITS IN FOREGROUNDCOLOUR GIVES THE BINARY CONTENTS OF THE UNEVEN BYTE.

Number of data	Bit-combination	Fore/background colour
2	56	16
2	56	16
2	56	16
2	16	224
2	254	224
2	16	224
2	16	224
2	4Ø	224
2	68	224
2	13Ø	224
Ø ← FLAG!		



I hope this routine can help you to achieve a higher animation-speed for your BASIC-program, but don't think you can make a PACMAN in BASIC with this routine (BASIC is still too slow).
An example of fast BASIC-animation using the SHAPE-program (add this to the program listed above) :

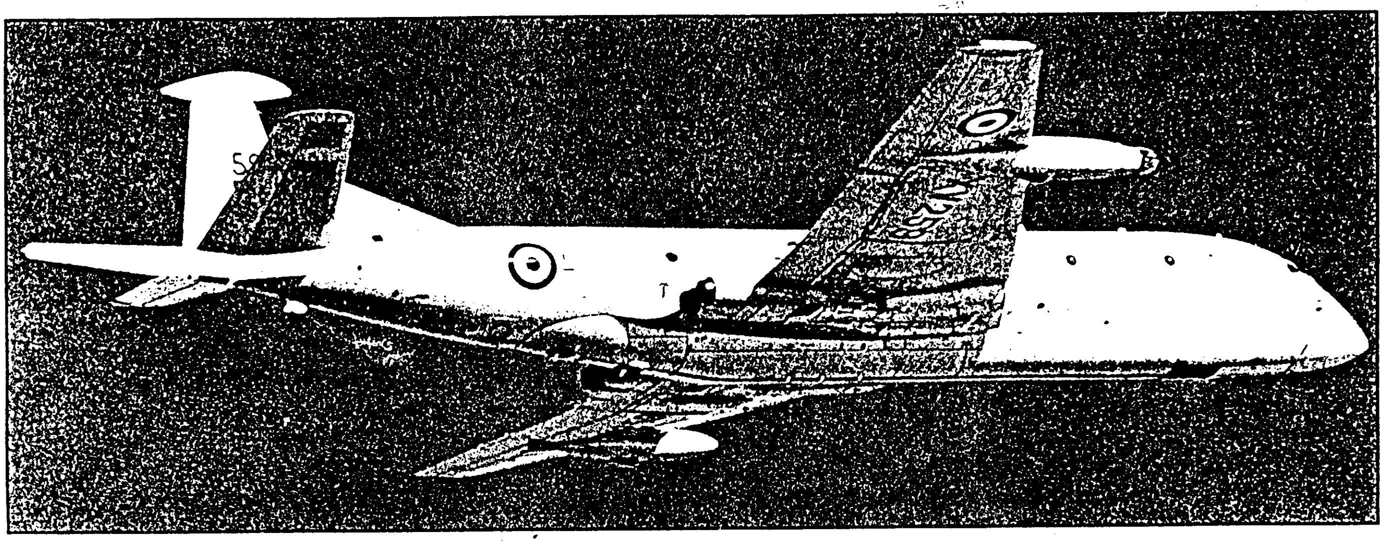
```
500 DIM S$(1),S(1,1):COLORG Ø 5 1Ø 15:MODE5:POKE RES,9Ø
51Ø FOR Y=Ø TO 1:FOR X=1 TO 76:READ Q:S$(Y)=S$(Y)+CHR$(Q):NEXT
52Ø V=VARPTR(S$(Y)):V1=PEEK(V)+PEEK(V+1)*256+1
53Ø S(Y,Ø)=V1 IAND #FF:S(Y,1)=V1 SHR 8:NEXT
54Ø POKE TBL,S(Ø,Ø):POKE TBL+1,S(Ø,1):REM shape 1
55Ø FOR X=#BFED-9Ø2Ø TO #BFED-9Ø6Ø STEP (-1Ø):CALLM USR,X:NEXT
56Ø WAIT TIME 1Ø
57Ø POKE TBL,S(1,Ø):POKE TBL+1,S(1,1):REM shape 2
58Ø FOR X=#BFED-9Ø2Ø TO #BFED-9Ø6Ø STEP (-1Ø):CALLM USR,X:NEXT
59Ø WAIT TIME 9:GOTO 54Ø
6ØØ DATA 4,3,224,224,224,4,15,224,248,224,4,31,224,252,224
61Ø DATA 4,15,224,158,224,4,7,224,158,224,4,3,224,159,224
62Ø DATA 4,1,224,255,224,4,Ø,Ø,255,224,4,1,224,255,224
63Ø DATA 4,3,224,255,224,4,7,224,254,224,4,15,224,254,224
64Ø DATA 4,31,224,252,224,4,15,224,248,224,4,3,224,24Ø,224,Ø
69Ø REM data for second shape :
7ØØ DATA 4,3,224,224,224,4,15,224,248,224,4,31,224,252,224
71Ø DATA 4,63,224,158,224,4,63,224,158,224,4,127,224,159,224
72Ø DATA 4,127,224,255,224,4,Ø,Ø,255,224,4,127,224,255,224
73Ø DATA 4,127,224,255,224,4,63,224,254,224,4,63,224,254,224
74Ø DATA 4,31,224,252,224,4,15,224,248,224,4,3,224,24Ø,224,Ø
```

For further information write : Dirk De Boeck
Hindedreef 15
2Ø7Ø KAPELLEN
(BELGIUM)

Misschien is het zinvol om bij het vierjarig bestaan van DAInamic een kort humorhoekje te voorzien. We zijn ervan overtuigd dat ook in de wereld van de microcomputer vrij veel fijne en diepzinnige humor verscholen ligt. Daarom starten we in dit nummer met twee voorbeelden en hopen dat ze inspirerend werken, zodat we regelmatig gelijkaardige kronkels kunnen afdrukken. Al uw vondsten opsturen naar Bruno Van Rompaey.

- Waar het hart van vol is loopt de mond van over.
  Zegt de leraar tegen zijn leerlingen :
  " Houdt u allemaal in stilte basic"

- Een waarschuwing : wees zuinig met de stringvariabelen in je programma; ze worden erg duur.



The Ultimate Video Game

Andrew Bond reports on how the innovative application of personal computing has produced a low cost solution to the training of operators for Britain's airborne anti-submarine defence effort

Procurement of defence equipment has always been a mystery to the uninitiated, that is the majority of people, who have tended to gain the impression that 'money no object' is the watchword. However true that may have been in the past, the increasing financial constraints imposed by successive defence reviews have created disciplines as tight as any that are encountered in the commercial world. Thus it should perhaps not be surprising, but is nevertheless gratifying, to find that one of the most innovative applications of low cost personal computing that at least this author has recently come across is with the RAF, providing a vital ι ient in the training of Air Electronics operators for the Nimrod maritime reconaissance aircraft.

The Air Electronics School at RAF Finningley near Doncaster has the prime responsibility of training operators for the highly sophisticated sonar and radar systems carried in the imrod, Britain's airborne contribution to Nato's effort to counter the Soviet submarine threat. The immense capital and operational costs of sophisticated systems such as the Nimrod mean that the amount of training performed on 'the real thing' must be kept to an absolute minimum. Indeed, since the game of cat and mouse played out in earnest day in day out in the North Atlantic between the Soviet submarines on the one hand a' ' the RAF and the Royal Navy on t.. other means that operators, once they join their operational units, must be capable of performing up to the very highest standards that would be required in actual hostilities. Paradoxically however, that also means that the opportunity to test their skills is limited since it depends on the Soviets obligingly laying on a submarine at the right place and the right time. Often, a long patrol can pass without the sonar operator having any contact to track and yet once such a contact does appear, he must be capable of performing his complex function, under stress, with faultless efficiency.

Clearly, the solution to such a training problem is simulation and the RAF has long experience of the use of simulated systems both for pilot training and for specialist operator training. The introduction of the Mark 2 version of the Nimrod, incorporating sonar and radar systems which represent a further quantum jump in technological sophistication, has further highlighted the problem of operator training. Working within a tight budget, the Air Electronics and Air Engineer School at Finningley was charged with providing and operating the facilities to train sonar and radar operators for the Nimrod 2.

The traditional approach to simulation of such systems is to provide the trainee with the same or identical equipment to that which he will use in the operational aircraft and then to provide synthetic inputs and responses to provide a complete simulation of the operational system. While providing a very much cheaper solution than training personnel on actual operational systems, this is nonetheless a costly business since a large part of the equipment of a fully operational system is required in the simulator.

Because of the complexity of the new Searchwater radar carried in the Nimrod 2, it was decided that this fully simulated approach was the appropriate route to take for the training of radar operators. The Basic Processed Radar Trainer (BPRT) now operating at Finningley is about as near as it is possible to get on the ground to providing realistic experience of using the system in the air. Such is the realism that operators from Nimrod 1 converting to the new system are being trained at Finningley until a similar installation is completed at their operational base. The Searchwater radar, accurately simulated by the BPRT provides capability vastly superior to its predecessors. The layman's image of radar operators is of men working in the dark, peering at a flickering CRT on which a timebase continuously rotates. Being a computer processed radar, Searchwater by contrast provides the operator with a high luminescence continuous display, viewable in daylight and more akin to

a detailed annotated chart than the blips of the World War II movie.

Having invested the larger part of its budget on radar training however, the Finningley team was faced with the problem of how to provide comparable facilities for would be sonar operators. Again, the popular impression of sonar still has more to do with Noel Coward and 'In Which We Serve' than with the systems currently operated both in airborne and shipborne anti-submarine warfare. Unlike the Asdic of the last war, the majority of modern sonar operates passively, purely as a listener, rather than through the propagation and subsequent detection of an active signal.

In the case of the systems carried by Nimrod and by the Royal Navy's anti-submarine helicopters, the primary tool is the sonar buoy, dropped by the aircraft to detect sonic emissions from submarines and relay them to the aircraft. The basic task of the operator is to analyse the resultant signals to produce data on the type, position, speed and course of the target. To this end he receives inputs simultaneously from a number of sonar buoys dropped in a pattern to provide a multipoint fix on the signal of interest.

In the earlier systems such as that carried in the Nimrod 1, the data was presented to the operator in the form of traces on a strip chart. The ensuing analysis was then a matter of complex calculations on measurements taken manually off the strip chart. With the operator being presented with a number of separate traces from different sonar buoys, signal processing was a highly skilled process in which sleight of hand and experience played perhaps the major role. Experience and familiarity with the signals is still a major element of operating expertise with the new system but the use of the computer and video display has taken much of the mechanical drudgery out of the task, leaving the operator free to devote his skill to the identification and tracking of the target.

Having identified a training need with regard to the sonar operator's task, the requirement was to seek a cost-effective solution. It was apparent that a full simulator would cost in the region of £750 000 and so the Research Branch of RAF Support Command was asked to evaluate the alternative of a microcomputer-based system, a radical departure from traditional operational equipment. Nevertheless, the response of the HQs was positive and the Research Branch, with the aid of RAF Finningley, produced a prototype to conduct a feasibility study and develop training lessons and exercises. The project was handled over two years by a team of four, comprising a computer scientist, a psychologist with specialist knowledge of training technology and two air-electronics personnel.

The eventual solution to the problem of training sonar operators has proved deceptively simple and quite astoundingly low in cost when compared with the more traditional simulator based directly on operational equipment. The heart of the Basic Acoustic Trainer now used at Finningley is the DAI Personal Computer supplied by Data Applications. Mounted in a mock up console replicating that in the aircraft, it interfaces with a dual floppy disk drive, printer, colour monitor and various 'real world' peripherals, in particular the tracker-ball and keypad through which the operator himself interacts with the system. The use of the colour monitor, coupled with the personal computer's high performance colour graphics and separate high speed maths processor enables the system to simulate accurately the display which will face the operator in the aircraft although the simulator presents him with only one display whereas the full system uses two.

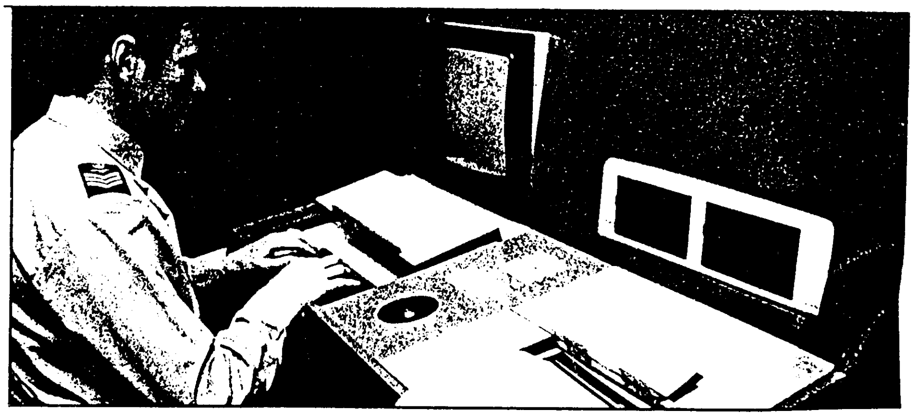
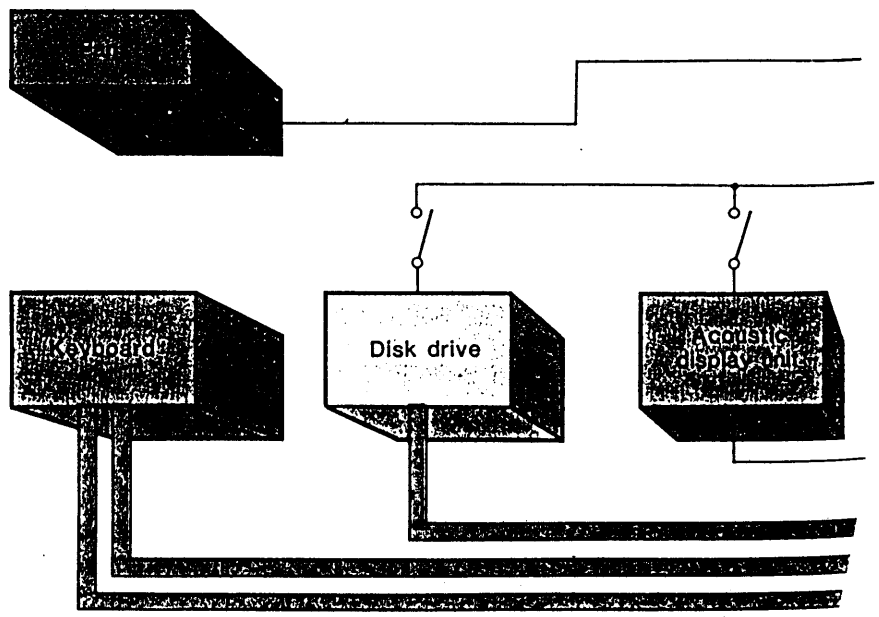


Fig 1



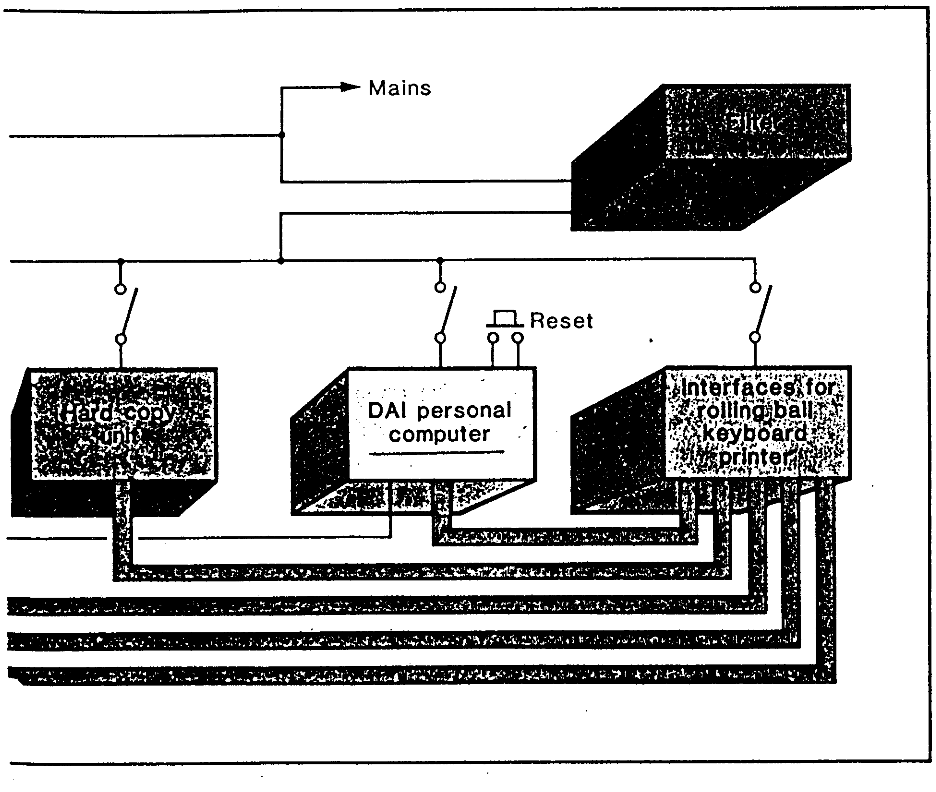
The actual display is in essence simply a real time version of the strip chart output of the earlier system. What makes the operator's task so much more complex than it might appear at first sight is that the signal received and relayed by the sonar buoy, comprises the emissions from all the vessels in the area, be they submarines or surface ships, naval or military, friendly or hostile. Moreover those signals are themselves masked to a greater or lesser extent by general background noise. It is the operator's task to discern from this mass of incoherent information the vital data which will enable the Nimrod to perform its role. It is a somewhat reassuring measure of the capabilities of the human brain that this task, while made easier and more effective by the use of computer processing, is nonetheless still performed better by the human operator than by any practicable computer-based pattern recognition system.

The purpose of the simulator then is to train the operator to recognise from the displayed data every one of the many signals and combinations of signals with which he is ever likely to be faced.

This is achieved by loading into the system via the disk drives data which either originates from actual operational sorties or which has been prepared by the instructor to highlight specific problems.The signal emitted by a vessel, be it a surface ship or a submarine, is a combination of frequencies bearing a constant relationship to each other. From careful analysis of this signal it is possible to determine, for example, the number of blades on its propeller and its shaft speed together with much more revealing data such as auxiliary machinery running at multiples of the basic shaft speed. From this signal, bearing in mind again that it may have to be discerned against a background of signals from other vessels and of general noise, the operator can determine sufficient information to ascertain, by comparison with data on both friendly and alien vessels, its exact type.

Further information obtained from the numerous buoys deployed enables the operator to determine position, course and speed and this allows him to track it and, if necessary, direct the aircraft in an attack. Using the personal computer based simulator, an instructor can therefore present the operator with a range of realistic situations varying from the relatively simple to the most complex that he will encounter in real conditions.

Setting up the Basic Acoustic Trainer, the keyboard is hidden during training exercises



Training of operators for this highly skilled and demanding task is by no means cheap. Indeed the cost of delivering a fully operational radar operator to a Nimrod 2 is in excess of £70 000 at 1981 prices. It is thus not surprising that the RAF has devoted considerable effort to ensuring that as few as possible drop out along the way. To this end it is interesting to note that it relies heavily on psychologists both to assist in the selection of candidates and in their subsequent training. The team responsible for the development of the Basic Acoustic Trainer has thus been a multidisciplinary one drawing on operational experience, training theory and practice and computer systems expertise. System and software engineering for the project has been undertaken by the

Research Branch of RAF Support Command based at RAF Brampton in Cambridgeshire, with support from Data Applications.

Any doubts that a trainer based on a low cost personal computer, albeit one of the more powerful available and designed to interface to a wide range of 10 devices, would not produce a realistic environment have been dispelled by the enthusiastic response the system has received both from instructors, themselves experienced operators, and from personnel who have had access to the system when converting from Nimrod 1.

Data Applications has now completed delivery of the ten systems ordered by MoD for RAF Finningley but is hopeful that that may not be the end of the story. On the one hand there is the possibility of a requirement for further systems at operational RAF bases while the Royal Navy, which operates a very similar system in its anti submarine helicopters, is actively considering its adoption. ☐

The author thanks the RAF and in particular staff at RAF Finningley for their generous help in the preparation of this article.

Data Applications Ltd is at Cirencester, Glos. Tel: 0285 61828

NEWSLETTER 17 EDITORIAL
(from DAInamic 17, p215)

Dear Members,

It was about three years ago that the first publication of DAInamic appeared, a sober stencilled sheet distributed via the firm DAI. It announced that a users' club had been founded for the DAI personal computer. The formation was occasioned by the lack of information on this revolutionary machine, and the need to make contact with fellow users in the neighbourhood so as to learn together. The founding of DAInamic met with great approval abroad, especially from Holland in the early months, and such enthusiastic response was a delightful surprise for our nucleus of members. The users of that time will certainly still remember the difficulties to be overcome to get possession of a DAI computer. Many of us had to settle for an 8K black & white version without sound. Around that time DAI suffered a great disappointment when they could not deliver their machines on time for the TELEAC Course and so lost a fine chance of widespread promotion. However all the computers in production for that were soon bought up and the TELEAC affair quickly forgotten. But alas, DAI's manufacturing capability was inadequate to satisfy the huge demand. In France, Britain, Germany and Italy interest in the computer was growing and DAInamic was getting swamped with questions in many tongues. It was therefore time to depart from the one-language issues and thus the various translation services came into being. The anxiety and uncertainty caused by the bankruptcy of the DAI Company was later followed by contentment when a healthy take-over occurred: INDATA was the new name. New people and new policies. Meanwhile it had become clear that the change was even more important for the home market. The number of Belgian members is now more than 450. In this short history we must mention a few names: J C Camby who as a true diplomat dealt with the impatient purchasers and those still waiting; Frank Druijff who quickly applied to join the Belgian nucleus; Freddy De Raedt who looked after programs like FGT and Assembler and answered members wanting to know more about machine language; Hans Wegman who put up his marker in DAI-land with the development of MDCR; Jan Boerrigter who with his colleagues unravelled the DAI hardware secrets for everybody and produced the Firmware Manual. Bruno Van Rompaey took the teaching profession in hand and founded diDAIsoft. There are so many co-workers and correspondents both near and far we cannot name for lack of space. All have helped to ensure that the DAIpc still has its place in the turbulent computer market and has a healthy future ahead of it. We thank you for many pleasurable contacts.

Until the next time,

Wilfried Hermans

VIDEOTEX IN BELGIUM.
(Synopsis of DAInamic 17, page 218)

The article describes the farcical situation existing prior to April 1983 in Belgium, allegedly as a result of the State monopoly of modems for connecting videotex equipment to telephone lines. Purchasers of modern videotex terminals with automatic dialling and in-built modems still had to rent the official modem even though they had no wish to use it. To make matters worse the official one was huge, old-fashioned and lacked the auto-dialling facility. They were reputed to have been consigned to a cupboard while only the complete new terminal was connected to the phone line. That appeared to take care of the legal niceties, but the official rent was so high that often it exceeded the rent of the complete modern videotex terminal. In April last the State relinquished its monopoly on modems, so that there would be no hindrance to technical developments! The Belgian videotex users have two prestel services, one run by Editel in Brussels and the other provided by Bell Telephone in Antwerp. Their current complaint is that telephone charges for calls to the videotex computers are too high, and compare unfavourably with the costs levied in other countries including the UK.

PROGRAMMING TECHNIQUES

(from DAInamic 17, page 224)

The problem for discussion this time is on attributing to a variable a value which itself depends on the value of another variable. This is often solved in the following way:-

```
150  IF A=5 THEN P=3: GOTO 200
160  IF A=6 THEN P=5: GOTO 200
170  IF A=7 THEN P=7: GOTO 200
180  IF A=8 THEN P=9: GOTO 200
190  P=0
200  .......
```

This is clear and is the best method when there are many possibilities for A and P or when the value has to be obtained by a simple calculation. I will give a number of worked examples of better solutions which result in shorter and sometimes faster programs. Suppose we want numbers from 100 to 300 checked to see if they are divisible by prime numbers less than 20. Input the following program after an IMP FPT.

```
  5  WAIT TIME 1: POKE #1BE,#FF: POKE #1BF,#FF
 10  FOR I=100.0 TO 300.0
 20  IF I/2.0=INT(I/2.0) GOTO 110
 30  IF I/3.0=INT(I/3.0) GOTO 110
 40  IF I/5.0=INT(I/5.0) GOTO 110
 50  IF I/7.0=INT(I/7.0) GOTO 110
 60  IF I/11.0=INT(I/11.0) GOTO 110
 70  IF I/13.0=INT(I/13.0) GOTO 110
 80  IF I/17.0=INT(I/17.0) GOTO 110
 90  IF I/19.0=INT(I/19.0) GOTO 110
100  PRINT I
110  NEXT
195  A=PEEK(#1BE): B=PEEK(#1BF): ?(#FFFF-A-B*256.0)/50.0;" SEC"
```

Lines 5 and 195 measure the running time. On my machine it took 12.44 (6.46) seconds. The time in the brackets is with the maths chip. We can see that for half the numbers the jump in line 20 will be needed. Thus the order of testing is logical. If we put lines 20 to 90 inclusive in reverse order the running time will be increased to 21.74 (11.12) seconds. Naturally one can write the program better. After an IMP INT type in:

```
  5  WAIT TIME 1: POKE #1BE,#FF: POKE #1BF,#FF
 10  FOR I=100 TO 300
 20  IF I/19*19=I GOTO 110
 30  IF I/17*17=I GOTO 110
 40  IF I/13*13=I GOTO 110
 50  IF I/11*11=I GOTO 110
 60  IF I/7*7=I GOTO 110
 70  IF I/5*5=I GOTO 110
 80  IF I/3*3=I GOTO 110
 90  IF I/2*2=I GOTO 110
100  PRINT I
110  NEXT
195  A=PEEK(#1BE): B=PEEK(#1BF): PRINT (#FFFF-A-B*256)/50.0
```

The running time now is 9.58 (6.56) seconds, the gain coming from working in integers. The lines 20 to 90 are still in reverse order; if they are again reversed the time becomes 5.88 (4.4). Combining line 20 with 30 and line 40 with 50 will save a bit more, achieving 5.82 (3.74) seconds. But it can still be better:-

```
 20  IF I/2*2=I THEN NEXT: GOTO 195
 30  IF I/3*3=I THEN NEXT
 40  IF I/5*5=I THEN NEXT
 50  IF I/7*7=I THEN NEXT
 60  IF I/11*11=I THEN NEXT
 70  IF I/13*13=I THEN NEXT
 80  IF I/17*17=I THEN NEXT
 90  IF I/19*19=I THEN NEXT
100  PRINT I: NEXT
```

Lines 5, 10 and 195 are as before. It is a less attractive construction because after a FOR in line 10 there 9 NEXTs. To keep the program portable each NEXT should be followed by a GOTO 195 but that would only increase the typing time, not the running time of 5.7 (3.62) seconds. Now change line 100 to read PRINT I;: NEXT The added semicolon is not much of a change but the time now becomes 4.96 (2.88). Although the maths chip has been shown to speed up running time by 30% to 50% thoughtful programming can sometimes achieve 75%. Consider now some variations on the original problem:-

First: A can be 2, 3, 4, 5, 6, 7 or 8 and in the same order. P must be 12, 15, 18, 21, 24, 27 or 30. There is an obvious mathematical link between A and P such that as A increases by 1, P increases by 3. P can therefore be obtained by multiplying A by 3 and adding 6.

```
     OLD                     NEW
150  IF A=2 THEN P=12
160  IF A=3 THEN P=15
170  IF A=4 THEN P=18        150  P=A*3+6
180  IF A=5 THEN P=21
190  IF A=6 THEN P=24
200  IF A=7 THEN P=27
210  IF A=8 THEN P=30
```

There is a difference but in practice it will rarely be a problem: 'old' has IFs so P changed conditionally but with 'new' P always changes.

Second: The same values as previously but in addition P must be 9 if A is less than 2 and 33 if A is greater than 8.

```
     OLD                          NEW
150  IF A<2 THEN P=9
160  IF A=2 THEN P=12
170  IF A=3 THEN P=15             150  P=A*3+6
...  .......                      160  IF A<2 THEN P=9
220  IF A=8 THEN P=30             170  IF A>8 THEN P=33
230  IF A>8 THEN P=33
```

The drawback of the 'new' method is the possibility of no output from line 150. If in many cases A is less than 2 it would be better to exchange lines 150 and 160 and put a GOTO 180 after the P=9.

Third: The case where A increases regularly but there is no simple mathematical link between A and P; a calculation is thus difficult or impossible.

```
     OLD                          NEW
150  IF A=3 THEN P=7
160  IF A=4 THEN P=4               10  DIM P(9)
170  IF A=5 THEN P=15              20  FOR I=3 TO 9: READ P(I): NEXT
180  IF A=6 THEN P=31
190  IF A=7 THEN P=76             150  P=P(A)
200  IF A=8 THEN P=45
210  IF A=9 THEN P=29             900  DATA 7,4,15,31,76,45,29
```

The 'new' method slows the program somewhat in the beginning but amply compensates later. If the 'old' was extended by say 20 lines the 'new' would have at most one extra line.

Fourth: The case where P regularly increases and A behaves irregularly. Here too an array would be appropriate.

```
     OLD                          NEW
150  IF A=3 THEN P=2
160  IF A=5 THEN P=3               10  DIM A(8)
170  IF A=9 THEN P=4               20  FOR I=2 TO 8: READ A(I): NEXT
180  IF A=12 THEN P=5             150  FOR I=2 TO 8: IF A(I)=A GOTO 160: NEXT
190  IF A=33 THEN P=6
200  IF A=42 THEN P=7
210  IF A=57 THEN P=8             160  P=I
```

Fifth: When there is no logical relationship either between A and P or in the values which are attributed to them. This could be when, for example, A is the ASCII code of a key while P is the action to be executed in a program. An array or arrays can be used; either a 2-dimensional array where As and Ps are next each other, or two separate single arrays. The latter is perhaps less elegant but works faster.

```
     OLD                          NEW
10   IF A=16 THEN P=7              10  DIM A(10),P(10)
20   IF A=17 THEN P=8              20  FOR I=1 TO 10: READ A(I),P(I): NEXT
30   IF A=18 THEN P=15
40   IF A=19 THEN P=16             50  FOR I=1 TO 10
50   IF A=65 THEN P=0              60  IF A=A(I) THEN P=P(I): GOTO 80
60   IF A=66 THEN P=2
70   IF A=74 THEN P=-1             70  NEXT
```

```
 80  IF A=78 THEN P=99        80  .......
 90  IF A=83 THEN P=100      900  DATA 16,7,17,8,18,15,19,16,65,0
100  IF A=9 THEN P=5         910  DATA 66,2,74,-1,78,99,83,100,9,5
```

Lines 50, 60 and 70 of the 'new' could be replaced by one line thus:

```
50   FOR I=1 TO 10: P=P(I): IF A=A(I) GOTO 60: NEXT
```

There are some points to look out for in order to avoid snags in the 'new' method: the new line 60 ends with a GOTO which helps to improve the speed but could jeopardise the FOR-NEXT loop if there is also a NEXT from an outer loop. This can be overcome by using NEXT I instead of just NEXT. Nico P Looije has assisted me in the task of speeding up the original program.

Frank H Druijff

8080 CASSETTE ROUTINES SDK-85.

(from DAInamic 17, page 232)

LETTER from Mr van Ool, Electronics Tutor, Almelo, Netherlands.

Dear Sir,

Herewith a complete source-listing of the promised program that makes it possible for all microcomputers which use the 8080, 8085 or Z80 microprocessor to communicate with the DAI pc in machine language via the audio cassette recorder.
The complete program contains the write and read routines which, via the original DAI pc interface, can record a machine language program on cassette (CASSrc) and read one in from cassette (CASSrd), as long as the micro system has been provided with the same interface. When the addresses where the program is located are used for other purposes by the other system, it is naturally possible to move everything. This is not difficult for DAI pc users who can handle the DNA assembler.

The benefit of the program is realised in teaching situations where the DAI pc is used for developing machine language programs with the DNA assembler (or the SPL macro assembler). The object files generated, with for example the #P. command, can be recorded on cassette and from there read in to the microsystem via the read routine. This saves the user the tedium of inputting the hex codes. The undersigned thinks DAIpc users with a technical leaning will especially find this gratifying.

Should there in the future be any interest in a CHECK program for testing a recorded program, I would be pleased to hear from you. You received previously the write program, before the read program was available; that may now be destroyed as the one with this has a few modifications. I am looking forward with pleasure to the insertion in DAInamic.

Friendly greetings from a northern neighbour,

J.J.H. van Ool.

INDATA NEWS

(from DAInamic 17, page 249)

New version of DAI masterDOS for existing floppy drives.

An new addition to the DAI MasterDOS now makes it possible to read and write directly to sectors and tracks. The addition was the result of general demand and gives the opportunity of making a real data base.

The syntax is as follows:-

RREC File name.ext Sector Memory position (Hex)
WREC File name.ext Sector Memory position (Hex)

Example: RREC TEST.BAS 1 5000: reads the first sector of the file "TEST.BAS" and puts the information at #5000.
All names and values can be variables so that these values can be used for programming.

Price: 500 Belgian francs for a disc and instructions.

SERVICE MANUAL

A service manual for the DAI Personal Computer has been published recently. The manual gives a very comprehensive description of the hardware functions and contains timing diagrams, memory map, and descriptions of processor, RAM, ROM, and video; in fact, all that a professional user needs to understand the workings of his machine. There are 16 pages giving the complete schematics of the computer.

Price: 1500 Belgian francs.

USING AZERTY USER

(from DAInamic 17, page 250)

1 Put the cassette in the DCR and connect up to the computer. When the DCR has stopped you can start working with the AZERTY keyboard.
2 Pressing Reset without the cassette in the DCR will reconfigure your keyboard to QWERTY again.
3 If you have already made use of the USER cassette you can still get back to AZERTY without trouble, from BASIC, by typing in CALLM #2F0. Take care to get the M; it is the 5th key from the left on the bottom row on a QWERTY board. Pay attention to what appears on the screen.
4 Never try G2F0 in the Utility mode; your computer will stop. If you are already in BASIC with an AZERTY keyboard then AZERTY will be effective for all programs both in BASIC and Utility.
5 The program is not to be used with other programs located below the Heap, as for example FGT. Should you require an adaptation for working such programs with an AZERTY keyboard you may get in touch with me; say which program is involved and give details like start and end addresses, entry point, version number, etc. If you wish, give a telephone number too but remember that I can only ring back during weekends.
6 You can make a back-up copy of the program as follows:-

```
* REW              <ret>  Rewind a new cassette.
* REW              <ret>  Rewind the USER cassette.
> UT               <ret>  Go to Utility mode
> R                <ret>  Read-in the USER file.
>                         Put new cassette in DCR
> W2F0 37E USER    <ret>  Copy the USER file.
> B                       Return to BASIC.
*
```

??? Queries, problems and suggestions may be directed to:-
Jos Schepens,
Sint Jorisgilde 53,
B-9330 DENDERMONDE, Belgium.
I can be reached by telephone on 052/21 67 43, but only on Saturdays and Sundays between 1400 and 2000.

```
2      DIM A$(12.0)
5      DATA CAPRICORNE,VERSEAU,POISSON,BELIER,TAUREAU,GEMEAU,CANCER,LION,VIE
RGE,BALANCE,SCORPION,SAGITTAIRE
7      FOR I!=1.0 TO 12.0:READ A$(I!):NEXT I!
10     MODE 0:PRINT CHR$(12):COLORT 12 5 0 0:CURSOR 19,23:PRINT "- SIGNES DU
 ZODIAQUE -":CURSOR 19,22:PRINT "====================":PRINT
20     PRINT "Christian Poels - 8/4/1981 - Ref.: Le BASIC par la pratique"
25     PRINT "                               (J.P. Lamoitier).":PRINT
30     PRINT "Quelle est votre date de naissance ? JJ/MM/AAAA":CURSOR 37,16:
PRINT "../../....":CURSOR 37,16
40     GOSUB 1000
41     J!=RU:GOSUB 1000:J!=J!*10.0+RU:PRINT "/";:GOSUB 1000:M!=RU:GOSUB 1000
:M!=M!*10.0+RU:PRINT "/";:GOSUB 1000
45     A!=RU
50     GOSUB 1000:A!=A!*10.0+RU:GOSUB 1000:A!=A!*10.0+RU
60     GOSUB 1000:A!=A!*10.0+RU
70     I!=M!:L!=20.0
80     ON M! GOTO 300,300,250,300,250,250,200,100,100,100,200,100
100    L!=L!+1.0
200    L!=L!+1.0
250    L!=L!+1.0
300    IF J!<L! THEN 320
310    I!=I!+1.0
320    IF I!<=12.0 THEN 340
330    I!=1.0
340    PRINT CHR$(12):PRINT "Vous etes ";A$(I!);"."
350    PRINT :PRINT "Voulez-vous recommencer (O/N) ?";
360    RE!=GETC:IF RE!=79.0 THEN 10
370    IF RE!<>78.0 THEN 360
380    END
1000   RE=GETC:IF RE<48.0 OR RE>57.0 THEN 1000
1010   RU=RE-48:RE$=MID$(STR$(RU),1,1):PRINT RE$;:RETURN
```

mikroshop hageland

COMPUTERS en ALLE TOEBEHOREN

SPECIALE AANBIEDINGEN TO EINDE '84

KLEUREN MONITORS met RGB ingang + gratis RGB-kaart (waarde 2380)

- BARCO 42 cm (nieuw model)	28.950
- TAXAN KAGA I (380 dots)	26.500
- ROLAND (High resolution)	35.000

PRINTERS + gratis grafische interface (waarde 6000)

- EPSON RX80 T	29.900
- EPSON RX80 F/T	33.500
- EPSON FX80 F/T	43.500
- STAR GEMINI 10X F/T	26.500
- STAR DELTA 10 F/T	40.900

PLOTTERS ROLAND seriele + parallel interface standard

- DXY-101 (1 pen)	39.500
- DXY-800 (8 pen)	49.500

MDCR MEMOCOM

DCR+TOS+KABEL+1 DOOS CASSETTES	16.000

COPAM PC401 I.B.M. compatibel systeem

- 8088 processor 4.77 Mhz. / 8087 optioneel
- 128K RAM / uitbreidbaar tot 256K
- 8 I.B.M. compatibele slots
- 2 360K drives ingebouwd
- seriele , parallel en klokkaart ingebouwd
- MS-DOS 2.11 operating system
- monochroom en RGB sturing ingebouwd
- monochrome monitor (I.B.M. alike)
- toetsenbord (I.B.M. compatibel)

STUNTPRIJS : 154.000

Alle vermelde prijzen in Belgische franken , B.T.W.(19%) incl.

Herseltsesteenweg 103 - 3220 Aarschot — tel. 016/56 87 70

BTW Nr. 424 175 357 GB Nr. 230-0282619-78

D-BASIC part 3

DBASIC EXTENSIONS

1. Purpose of this article

In previous articles and in the DBASIC manual, I mentioned that an extension can be used to add new commands and/or statements to the existing instruction set of DBASIC.
Using an example, I will explain how such an extension can be programmed in assembly language. Some knowledge of 8080 assembly language programming and the DAI operating system is desired for understanding this explanation.

2. The example : direct input/output

A usefull extension of DBASIC could be a direct input/output facility : i.e. writing a part of memory directly to tape or disk, or reading a saved part of memory direcly from tape or disk (cfr. R and W commands in utilities).
These commands are supported on some systems as DLOAD and DSAVE (ex KENDOS). I propose the following syntax-rules (items in square brackets are optional) :

```
     DSAVE LOWADDRESS,HIGHADDRESS[;FILENAME]
     DLOAD[ OFFSET][;FILENAME]
```

3. Table driven syntax

To link these commands to DBASIC you have to provide a table specifying the syntax,runaddresses etc...
Listing 1 (page 326), a SPL macro assembler source listing of a program to create the DBASIC extension DIO (Direct Input Output), you will find the table-layout.
In this table non-documented items are just length-bytes. The maximum length is 0fh. All the other items are described below.

-extension name : is used for error-reporting and the $DELETE command.
-extension id : is a number between 0h and 0ffh which is needed for compilation. I advice you to number your own extensions descending from 0efh on to avoid conflicts with standard DBASIC extensions.
ex. extension id. of $SYSTEM is 0ffh
$DCR is 0h
-relocation table : is used in $EXTEND and points to a table with all the addresses to be relocated. In order to be completely relocatable a machine language program should only contain 2-byte word memory-references (ex. avoid the use of LOW and HIGH operators in MACRO 80). After loading and relocation of the extension the relocation table will not be kept on line.
-separators : is a set of 8 punctuation marks needed during encoding and listing of the commands. Any argument is always preceeded by one of these separators.
-command string : identifies the command. Only the 1st character of the command string may be non-alphanumeric (ex. $ in $EXTEND).
-encode control : it's binairy form is ccXX 1111 (X stands for don't care).

with	1111	number of possible arguments+2
	cc=X1	statement valid in program
	cc=1X	command valid in direct command mode

-execution address : offset to the start-address of the command's execution code.

```
-argument syntax :  it's binairy form is tttt sssf
with      sss       the number of the separator which preceeds the argument
                    (from 0 to 7).
          f=1       the argument preceded by separator sss is obligatory.
          f=0       the argument preceded by separator sss is optional.
          tttt=0000 the argument is a floating point expression.
          tttt=0001 the argument is an integer expression.
          tttt=0010 the argument is a string expression.
          tttt=0011 the argument is a variable reference.
          tttt=0111 the argument is an array reference (cfr. LOADA).
          tttt=1011 the  argument  is  a  group  of  variable  references
                    separated by ',' (cfr. READ).
          tttt=1111 the  rgument is a group of array-references  separated
                    by ','.
```

In our example the encode control of DSAVE is 0c5h, thus DSAVE can be used as direct command or as statement in a program. The length of the info is 5, 2 bytes for the run-address and 3 bytes of argument syntax description :

argument syntax 17h : an integer expression preceeded by separator 3 (a blank) has to be supplied.
argument syntax 11h : an integer expression preceeded by separator 0 (a ',') has to be supplied.
argument syntax 22h : a string expression preceeded by separator 1 (a ';') is optional.

The DLOAD command has two argument syntax bytes :

argument syntax 16h : an integer expression preceeded by separator 3 (a blank) is optional.
argument syntax 22h : a string expression preceeded by separator 1 (a ';') is optional.

If you understand this you will agree with me that, using the same separators, the command HOME has no argument syntax byte and that the command ERASE ARRAY1,ARRAY2,... will have one argument syntax byte : 0f7h.

The code

The runtime code usualy can be seen as a sequention of 2 parts :
part 1 : evaluate the arguments.
part 2 : do some processing using the evaluated arguments as parameters.

For evaluation of the arguments you need the addresses of standard DBASIC routines. The 2 routines needed in DIO are :

REXI2 : evaluate a 2 byte integer expression in hl.
REXSR : evaluate a string expression (hl points to the string).

A list of the most important DBASIC routines with a description of the entry-conditions and the produced output will be available soon.
Note that in evaluating optional arguments a test is done on a 0-byte in the textbuffer. This is because for a non-supplied optional argument a 0-byte is encoded in the textbuffer.

Extension controls

Five pointers in the DBASIC system ram are reserved to control encoding, listing and evaluation of extended commands.
These five controls are :

USCMTB : is a pointer to the first extension-root (=start of 1st table)
A next extension is linked to the previous extension through the next table pointer (=relocation table pointer).

SEPTAB : is a pointer to the separator table during encoding.

ROTSAV : is used in error-handling. If ROTSAV=0h an error will be considered to be generated in a DBASIC command, else the error will be considered to be a specific extension error and ROTSAV points to the extension root. Thus if we want explicit extension errors instead of for instance a 'NUMBER OUT OF RANGE' error, the first thing we have to do is to set ROTSAV equal to our extension root (DIOROT in our example).
Then we would get error messages of the form :

```
DIO ERROR nnn  or
DIO 'special error message' (see ERRREP)
```

ERRREP : is a pointer to a special extension-error-reporting-routine. This pointer has to be supplied during the execution of the extended commands.
Assume we want 2 special error messages in DIO :

```
DIO DSAVE ERROR  (ERR=1)  and
DIO DLOAD ERROR  (ERR=2)
```

To print the special error messages we have to supply DIOERR to ERRREP. As you can see this is done during execution of the extension's auto-recovery (USCREC).

USCREC : is a jump to the extension's auto-recovery routine.
If an error occures during execution of an extended command, you may have to restore some system data or anything else that has been changed by the extended command. USCREC allows you to do it. In our example this auto-recovery feature is only used to convert the error codes and to enable special error reporting.

Another extension : HOME

Listing 2 (page 337) shows how the HOME command (Apple 2) can be implemented. As you can see the code of this extension is very simple since no arguments have to be evaluated and no error reporting has to be done.

I hope you will have enough information to be able to experiment with DBASIC extensions.

Willy Coremans

```
;
        TITL     'DIO : DIRECT INPUT/OUTPUT'
;
;note : Assemble with offset (ex. A1000).
;       Pass this offset to the write macro (write OFFSET).
;       To write the extension, execute WRITE+OFFSET.
;
OFFSET  SET      1000H          ;offset for Assemble
;
TRUE    SET      0FFFFH
FALSE   SET      0H
;
ERRORR  SET      TRUE           ;special error reporting
;
DIOID   SET      0EFH           ;extension id
;
;---system ram---
;
POROM   SET      40H            ;duplicate of 0FD06H
PORO    SET      0FD06H         ;discrete output port
;
ROPEN   SET      2CEH           ;open file for read
RBLK    SET      2D1H           ;read block
RCLOSE  SET      2D4H           ;close file after read
;
;---dbasic system ram---
;
ERRBYT  SET      5H             ;error code
USCREC  SET      33H            ;extension's auto recovery
USCMTB  SET      0C8H           ;root of first extension
SEPTAB  SET      0CAH           ;separator table
ROTSAV  SET      0CCH           ;saved extension root
ERRREP  SET      0CEH           ;special error reporting
;
;---dbasic call's---
;
REXI2   SET      1F47H          ;run 2-byte int. ex. in hl
REXSR   SET      1FAFH          ;run $-ex. in hl
;
;---rom call's---
;
DADA    SET      0DE30H
PMSG    SET      0DAD4H
;
;---data definition of macro's---
;
DATA    SET      TRUE
CODE    SET      FALSE
        write    OFFSET
;
        ORG      0H
;
;---command table---
;
DIOROT  DB       3H
```

```
                DB      'DIO'           ;=extension name
                DB      0BH
                DB      DIOID           ;=extension id.
                DW      RELTBL          ;=relocation table/next table
                DB      ',;#  ./='      ;=separators
                DB      5H
                DB      'DSAVE'         ;=command name
                DB      0C5H            ;=encode control
        REL010  DW      RDSAVE          ;=run-address
                DB      17H             ;=argument syntax
                DB      11H             ;=argument syntax
                DB      22H             ;=argument syntax
                DB      5H
                DB      'DLOAD'         ;=command name
                DB      0C4H            ;=encode control
        REL020  DW      RDLOAD          ;=run-address
                DB      16H             ;=argument syntax
                DB      22H             ;=argument syntax
                DB      0H              ;=end table
        ;
        ;---runtime-code---
        ;
        ;---direct save---
        ;
        RDSAVE  IF      ERRORR=TRUE
        RLE010  LXI H   DIOROT          ;enable ext. error reporting
                SHLD    ROTSAV
        RLE020  LXI H   DSVERR          ;set ext. auto recovery
                SHLD    USCREC+1H
        ;
                ELSE
                LXI H   0H
                SHLD    ROTSAV          ;disable ext. error reporting
                ENDIF
        ;
                LDA     POROM
                PUSH PSW                ;save current bank
        RL0010  CALL    RDSAV1          ;direct save
                POP PSW
                ORA A
        RL0020  JMP     BANKRS          ;restore bank
        ;
        RDSAV1  CALL    REXI2           ;get low address
                PUSH H
                CALL    REXI2           ;get high address
                PUSH H
                LXI H   0H              ;default is no file-name
        RL0030  CALL    REXSRS          ;get optional file-name
        RL0040  CALL    BANK3           ;switch to bank 3
                JMP     0EEF0H          ;write file
        ;
        BANK3   LDA     POROM
                ANI     3FH
                ORI     0C0H
        BANKRS  STA     POROM
                STA     PORO
                RET

```

```
SPL V1.1  PAGE 3       DIO : DIRECT INPUT/OUTPUT

;---direct load---
;
RDLOAD  IF       ERRORR=TRUE
RLE030  LXI H    DIOROT            ;enable ext. error reporting
        SHLD     ROTSAV
RLE040  LXI H    DLDERR            ;set ext. auto-recovery
        SHLD     USCREC+1H
        LXI H    0H
;
        ELSE
        LXI H    0H                ;default is no offset
        SHLD     ROTSAV            ;disable ext. error reporting
        ENDIF
;
RL0050  CALL     REXI2S            ;get optional offset
        XCHG                       ;in de
        LXI H    0H                ;default is read without name
RL0060  CALL     REXSRS
        PUSH B                     ;save txtbuf-pointer
        PUSH D                     ;save offset
        LXI B    3100H             ;file type '1'
        PUSH H                     ;no display while read
        LHLD     100H
        MOV A,H
        ORA L
        POP H
RL0065  JNZ      DLDPGM
        MVI C    0FFH              ;if dir. cmd. display
DLDPGM  CALL     ROPEN             ;open file
RL0070  LXI H    DUMPSA+1H
RL0080  LXI D    DUMPSE
        CALL     RBLK              ;dump start-address
DUMPSA  LXI H    0H                ;get start-address
DUMPSE  POP D
        DAD D                      ;add offset
        LXI D    0F900H
        CALL     RBLK              ;direct load
        CALL     RCLOSE            ;close file
        POP B
        ORA A
        RET
;
REXI2S  LDAX B
        INX B
        ORA A
        RZ
        DCX B
        JMP      REXI2
;
REXSRS  LDAX B
        INX B
        ORA A
        RZ
        DCX B
        JMP      REXSR
;
        IF       ERRORR=TRUE
;
```

```
;---extension's auto recovery---
;
DSVERR  MVI A    1H             ;convert error to 1
DIOSER  STA      ERRBYT
RLE050  LXI H    DIOERR         ;set special error reproting
        SHLD     ERRREP
        RET
;
DLDERR  MVI A    2H             ;convert error to 2
RLE060  JMP      DIOSER
;
;---special error reporting---
;
DIOERR  LDA      ERRBYT         ;report the error
        ADD A
RLE070  LXI H    DIOETB
        CALL     DADA
        MOV A,M
        INX H
        MOV H,M
        MOV L,A
        JMP      PMSG
;
;---extension's error-message table---
;
DIOETB  SET      $-2H
RELE10  DW       MDSVER
RELE20  DW       MDLDER
;
;---error messages---
;
MDSVER  DB       'D'            ;D
        strin    0CD23H         ;SAVE
        mess     0DC15H         ; ERROR
        DB       0H
MDLDER  DB       'D'            ;D
        strin    0CD1BH         ;LOAD
        mess     0DC15H         ; ERROR
        DB       0H
;
        ENDIF
;
;---relocation table---
;
RELTBL  DW       REL010
        DW       REL020
;
        IF       ERRORR=TRUE
        DW       RELE10
        DW       RELE20
        ENDIF
;
        DW       RL0010+1H
        DW       RL0020+1H
        DW       RL0030+1H
        DW       RL0040+1H
        DW       RL0050+1H
        DW       RL0060+1H
```

```
SPL V1.1  PAGE 5        DIO : DIRECT INPUT/OUTPUT

                DW      RL0065+1H
                DW      RL0070+1H
                DW      RL0080+1H
        ;
                IF      ERRORR=TRUE
                DW      RLE010+1H
                DW      RLE020+1H
                DW      RLE030+1H
                DW      RLE040+1H
                DW      RLE050+1H
                DW      RLE060+1H
                DW      RLE070+1H
                ENDIF
        ;
                DW      0H
        ;
        WRITEN  SET     $
        ;
        DATA    SET     FALSE
        CODE    SET     TRUE
        WRITE   SET     $
                write   OFFSET
        ;
                END
        ;
        ;---write an extension ---
        ;
        write   MACRO   OFF
                IF      DATA=TRUE
        WOPEN   SET     2C5H
        WBLK    SET     2C8H
        WCLOSE  SET     2CBH
                ENDIF
                IF      CODE=TRUE
                MVI A   '$'
                LXI H   OFF
                CALL    WOPEN
                LXI D   2H
                LXI H   DUM+OFF
                CALL    WBLK
                LXI H   OFF
                LXI D   WRITEN
                CALL    WBLK
                JMP     WCLOSE
        DUM     DW      0H
                ENDIF
                MEND
        ;
        strin   MACRO   PTR
        PTR     SET     PTR-4000H
                DB      PTR)8H
                DB      PTR&0FFH
                MEND
        ;
        mess    MACRO   PNTR
                DB      PNTR)8H
                DB      PNTR&0FFH
                MEND
```

MEMORYMAP MODE 1/2

64	BFEF	BFEE	BFED	BFEC	BFEB	BFEA	BFE9	BFE8	BFE7	BFE6	BFE5	BFE4	BFE3	BFE2	BFE1	BFE0	BFDF	BFDE	BFDD	BFDC	BFDB	BFDA	BFD9	BFD8
63	BFD7	BFD6	BFD5	BFD4	BFD3	BFD2	BFD1	BFD0	BFCF	BFCE	BFCD	BFCC	BFCB	BFCA	BFC9	BFC8	BFC7	BFC6	BFC5	BFC4	BFC3	BFC2	BFC1	BFC0
62	BFBF	BFBE	BFBD	BFBC	BFBB	BFBA	BFB9	BFB8	BFB7	BFB6	BFB5	BFB4	BFB3	BFB2	BFB1	BFB0	BFAF	BFAE	BFAD	BFAC	BFAB	BFAA	BFA9	BFA8
61	BFA7	BFA6	BFA5	BFA4	BFA3	BFA2	BFA1	BFA0	BF9F	BF9E	BF9D	BF9C	BF9B	BF9A	BF99	BF98	BF97	BF96	BF95	BF94	BF93	BF92	BF91	BF90
60	BF8F	BF8E	BF8D	BF8C	BF8B	BF8A	BF89	BF88	BF87	BF86	BF85	BF84	BF83	BF82	BF81	BF80	BF7F	BF7E	BF7D	BF7C	BF7B	BF7A	BF79	BF78
59	BF77	BF76	BF75	BF74	BF73	BF72	BF71	BF70	BF6F	BF6E	BF6D	BF6C	BF6B	BF6A	BF69	BF68	BF67	BF66	BF65	BF64	BF63	BF62	BF61	BF60
58	BF5F	BF5E	BF5D	BF5C	BF5B	BF5A	BF59	BF58	BF57	BF56	BF55	BF54	BF53	BF52	BF51	BF50	BF4F	BF4E	BF4D	BF4C	BF4B	BF4A	BF49	BF48
57	BF47	BF46	BF45	BF44	BF43	BF42	BF41	BF40	BF3F	BF3E	BF3D	BF3C	BF3B	BF3A	BF39	BF38	BF37	BF36	BF35	BF34	BF33	BF32	BF31	BF30
56	BF2F	BF2E	BF2D	BF2C	BF2B	BF2A	BF29	BF28	BF27	BF26	BF25	BF24	BF23	BF22	BF21	BF20	BF1F	BF1E	BF1D	BF1C	BF1B	BF1A	BF19	BF18
55	BF17	BF16	BF15	BF14	BF13	BF12	BF11	BF10	BF0F	BF0E	BF0D	BF0C	BF0B	BF0A	BF09	BF08	BF07	BF06	BF05	BF04	BF03	BF02	BF01	BF00
54	BEFF	BEFE	BEFD	BEFC	BEFB	BEFA	BEF9	BEF8	BEF7	BEF6	BEF5	BEF4	BEF3	BEF2	BEF1	BEF0	BEEF	BEEE	BEED	BEEC	BEEB	BEEA	BEE9	BEE8
53	BEE7	BEE6	BEE5	BEE4	BEE3	BEE2	BEE1	BEE0	BEDF	BEDE	BEDD	BEDC	BEDB	BEDA	BED9	BED8	BED7	BED6	BED5	BED4	BED3	BED2	BED1	BED0
52	BECF	BECE	BECD	BECC	BECB	BECA	BEC9	BEC8	BEC7	BEC6	BEC5	BEC4	BEC3	BEC2	BEC1	BEC0	BEBF	BEBE	BEBD	BEBC	BEBB	BEBA	BEB9	BEB8
51	BEB7	BEB6	BEB5	BEB4	BEB3	BEB2	BEB1	BEB0	BEAF	BEAE	BEAD	BEAC	BEAB	BEAA	BEA9	BEA8	BEA7	BEA6	BEA5	BEA4	BEA3	BEA2	BEA1	BEA0
50	BE9F	BE9E	BE9D	BE9C	BE9B	BE9A	BE99	BE98	BE97	BE96	BE95	BE94	BE93	BE92	BE91	BE90	BE8F	BE8E	BE8D	BE8C	BE8B	BE8A	BE89	BE88
49	BE87	BE86	BE85	BE84	BE83	BE82	BE81	BE80	BE7F	BE7E	BE7D	BE7C	BE7B	BE7A	BE79	BE78	BE77	BE76	BE75	BE74	BE73	BE72	BE71	BE70
48	BE6F	BE6E	BE6D	BE6C	BE6B	BE6A	BE69	BE68	BE67	BE66	BE65	BE64	BE63	BE62	BE61	BE60	BE5F	BE5E	BE5D	BE5C	BE5B	BE5A	BE59	BE58
47	BE57	BE56	BE55	BE54	BE53	BE52	BE51	BE50	BE4F	BE4E	BE4D	BE4C	BE4B	BE4A	BE49	BE48	BE47	BE46	BE45	BE44	BE43	BE42	BE41	BE40
46	BE3F	BE3E	BE3D	BE3C	BE3B	BE3A	BE39	BE38	BE37	BE36	BE35	BE34	BE33	BE32	BE31	BE30	BE2F	BE2E	BE2D	BE2C	BE2B	BE2A	BE29	BE28
45	BE27	BE26	BE25	BE24	BE23	BE22	BE21	BE20	BE1F	BE1E	BE1D	BE1C	BE1B	BE1A	BE19	BE18	BE17	BE16	BE15	BE14	BE13	BE12	BE11	BE10
44	BE0F	BE0E	BE0D	BE0C	BE0B	BE0A	BE09	BE08	BE07	BE06	BE05	BE04	BE03	BE02	BE01	BE00	BDFF	BDFE	BDFD	BDFC	BDFB	BDFA	BDF9	BDF8
43	BDF7	BDF6	BDF5	BDF4	BDF3	BDF2	BDF1	BDF0	BDEF	BDEE	BDED	BDEC	BDEB	BDEA	BDE9	BDE8	BDE7	BDE6	BDE5	BDE4	BDE3	BDE2	BDE1	BDE0
42	BDDF	BDDE	BDDD	BDDC	BDDB	BDDA	BDD9	BDD8	BDD7	BDD6	BDD5	BDD4	BDD3	BDD2	BDD1	BDD0	BDCF	BDCE	BDCD	BDCC	BDCB	BDCA	BDC9	BDC8
41	BDC7	BDC6	BDC5	BDC4	BDC3	BDC2	BDC1	BDC0	BDBF	BDBE	BDBD	BDBC	BDBB	BDBA	BDB9	BDB8	BDB7	BDB6	BDB5	BDB4	BDB3	BDB2	BDB1	BDB0
40	BDAF	BDAE	BDAD	BDAC	BDAB	BDAA	BDA9	BDA8	BDA7	BDA6	BDA5	BDA4	BDA3	BDA2	BDA1	BDA0	BD9F	BD9E	BD9D	BD9C	BD9B	BD9A	BD99	BD98
39	BD97	BD96	BD95	BD94	BD93	BD92	BD91	BD90	BD8F	BD8E	BD8D	BD8C	BD8B	BD8A	BD89	BD88	BD87	BD86	BD85	BD84	BD83	BD82	BD81	BD80
38	BD7F	BD7E	BD7D	BD7C	BD7B	BD7A	BD79	BD78	BD77	BD76	BD75	BD74	BD73	BD72	BD71	BD70	BD6F	BD6E	BD6D	BD6C	BD6B	BD6A	BD69	BD68
37	BD67	BD66	BD65	BD64	BD63	BD62	BD61	BD60	BD5F	BD5E	BD5D	BD5C	BD5B	BD5A	BD59	BD58	BD57	BD56	BD55	BD54	BD53	BD52	BD51	BD50
36	BD4F	BD4E	BD4D	BD4C	BD4B	BD4A	BD49	BD48	BD47	BD46	BD45	BD44	BD43	BD42	BD41	BD40	BD3F	BD3E	BD3D	BD3C	BD3B	BD3A	BD39	BD38
35	BD37	BD36	BD35	BD34	BD33	BD32	BD31	BD30	BD2F	BD2E	BD2D	BD2C	BD2B	BD2A	BD29	BD28	BD27	BD26	BD25	BD24	BD23	BD22	BD21	BD20
34	BD1F	BD1E	BD1D	BD1C	BD1B	BD1A	BD19	BD18	BD17	BD16	BD15	BD14	BD13	BD12	BD11	BD10	BD0F	BD0E	BD0D	BD0C	BD0B	BD0A	BD09	BD08
33	BD07	BD06	BD05	BD04	BD03	BD02	BD01	BD00	BCFF	BCFE	BCFD	BCFC	BCFB	BCFA	BCF9	BCF8	BCF7	BCF6	BCF5	BCF4	BCF3	BCF2	BCF1	BCF0
32	BCEF	BCEE	BCED	BCEC	BCEB	BCEA	BCE9	BCE8	BCE7	BCE6	BCE5	BCE4	BCE3	BCE2	BCE1	BCE0	BCDF	BCDE	BCDD	BCDC	BCDB	BCDA	BCD9	BCD8
31	BCD7	BCD6	BCD5	BCD4	BCD3	BCD2	BCD1	BCD0	BCCF	BCCE	BCCD	BCCC	BCCB	BCCA	BCC9	BCC8	BCC7	BCC6	BCC5	BCC4	BCC3	BCC2	BCC1	BCC0
30	BCBF	BCBE	BCBD	BCBC	BCBB	BCBA	BCB9	BCB8	BCB7	BCB6	BCB5	BCB4	BCB3	BCB2	BCB1	BCB0	BCAF	BCAE	BCAD	BCAC	BCAB	BCAA	BCA9	BCA8
29	BCA7	BCA6	BCA5	BCA4	BCA3	BCA2	BCA1	BCA0	BC9F	BC9E	BC9D	BC9C	BC9B	BC9A	BC99	BC98	BC97	BC96	BC95	BC94	BC93	BC92	BC91	BC90
28	BC8F	BC8E	BC8D	BC8C	BC8B	BC8A	BC89	BC88	BC87	BC86	BC85	BC84	BC83	BC82	BC81	BC80	BC7F	BC7E	BC7D	BC7C	BC7B	BC7A	BC79	BC78
27	BC77	BC76	BC75	BC74	BC73	BC72	BC71	BC70	BC6F	BC6E	BC6D	BC6C	BC6B	BC6A	BC69	BC68	BC67	BC66	BC65	BC64	BC63	BC62	BC61	BC60
26	BC5F	BC5E	BC5D	BC5C	BC5B	BC5A	BC59	BC58	BC57	BC56	BC55	BC54	BC53	BC52	BC51	BC50	BC4F	BC4E	BC4D	BC4C	BC4B	BC4A	BC49	BC48
25	BC47	BC46	BC45	BC44	BC43	BC42	BC41	BC40	BC3F	BC3E	BC3D	BC3C	BC3B	BC3A	BC39	BC38	BC37	BC36	BC35	BC34	BC33	BC32	BC31	BC30
24	BC2F	BC2E	BC2D	BC2C	BC2B	BC2A	BC29	BC28	BC27	BC26	BC25	BC24	BC23	BC22	BC21	BC20	BC1F	BC1E	BC1D	BC1C	BC1B	BC1A	BC19	BC18
23	BC17	BC16	BC15	BC14	BC13	BC12	BC11	BC10	BC0F	BC0E	BC0D	BC0C	BC0B	BC0A	BC09	BC08	BC07	BC06	BC05	BC04	BC03	BC02	BC01	BC00
22	BBFF	BBFE	BBFD	BBFC	BBFB	BBFA	BBF9	BBF8	BBF7	BBF6	BBF5	BBF4	BBF3	BBF2	BBF1	BBF0	BBEF	BBEE	BBED	BBEC	BBEB	BBEA	BBE9	BBE8
21	BBE7	BBE6	BBE5	BBE4	BBE3	BBE2	BBE1	BBE0	BBDF	BBDE	BBDD	BBDC	BBDB	BBDA	BBD9	BBD8	BBD7	BBD6	BBD5	BBD4	BBD3	BBD2	BBD1	BBD0
20	BBCF	BBCE	BBCD	BBCC	BBCB	BBCA	BBC9	BBC8	BBC7	BBC6	BBC5	BBC4	BBC3	BBC2	BBC1	BBC0	BBBF	BBBE	BBBD	BBBC	BBBB	BBBA	BBB9	BBB8
19	BBB7	BBB6	BBB5	BBB4	BBB3	BBB2	BBB1	BBB0	BBAF	BBAE	BBAD	BBAC	BBAB	BBAA	BBA9	BBA8	BBA7	BBA6	BBA5	BBA4	BBA3	BBA2	BBA1	BBA0
18	BB9F	BB9E	BB9D	BB9C	BB9B	BB9A	BB99	BB98	BB97	BB96	BB95	BB94	BB93	BB92	BB91	BB90	BB8F	BB8E	BB8D	BB8C	BB8B	BB8A	BB89	BB88
17	BB87	BB86	BB85	BB84	BB83	BB82	BB81	BB80	BB7F	BB7E	BB7D	BB7C	BB7B	BB7A	BB79	BB78	BB77	BB76	BB75	BB74	BB73	BB72	BB71	BB70
16	BB6F	BB6E	BB6D	BB6C	BB6B	BB6A	BB69	BB68	BB67	BB66	BB65	BB64	BB63	BB62	BB61	BB60	BB5F	BB5E	BB5D	BB5C	BB5B	BB5A	BB59	BB58
15	BB57	BB56	BB55	BB54	BB53	BB52	BB51	BB50	BB4F	BB4E	BB4D	BB4C	BB4B	BB4A	BB49	BB48	BB47	BB46	BB45	BB44	BB43	BB42	BB41	BB40
14	BB3F	BB3E	BB3D	BB3C	BB3B	BB3A	BB39	BB38	BB37	BB36	BB35	BB34	BB33	BB32	BB31	BB30	BB2F	BB2E	BB2D	BB2C	BB2B	BB2A	BB29	BB28
13	BB27	BB26	BB25	BB24	BB23	BB22	BB21	BB20	BB1F	BB1E	BB1D	BB1C	BB1B	BB1A	BB19	BB18	BB17	BB16	BB15	BB14	BB13	BB12	BB11	BB10
12	BB0F	BB0E	BB0D	BB0C	BB0B	BB0A	BB09	BB08	BB07	BB06	BB05	BB04	BB03	BB02	BB01	BB00	BAFF	BAFE	BAFD	BAFC	BAFB	BAFA	BAF9	BAF8
11	BAF7	BAF6	BAF5	BAF4	BAF3	BAF2	BAF1	BAF0	BAEF	BAEE	BAED	BAEC	BAEB	BAEA	BAE9	BAE8	BAE7	BAE6	BAE5	BAE4	BAE3	BAE2	BAE1	BAE0
10	BADF	BADE	BADD	BADC	BADB	BADA	BAD9	BAD8	BAD7	BAD6	BAD5	BAD4	BAD3	BAD2	BAD1	BAD0	BACF	BACE	BACD	BACC	BACB	BACA	BAC9	BAC8
9	BAC7	BAC6	BAC5	BAC4	BAC3	BAC2	BAC1	BAC0	BABF	BABE	BABD	BABC	BABB	BABA	BAB9	BAB8	BAB7	BAB6	BAB5	BAB4	BAB3	BAB2	BAB1	BAB0
8	BAAF	BAAE	BAAD	BAAC	BAAB	BAAA	BAA9	BAA8	BAA7	BAA6	BAA5	BAA4	BAA3	BAA2	BAA1	BAA0	BA9F	BA9E	BA9D	BA9C	BA9B	BA9A	BA99	BA98
7	BA97	BA96	BA95	BA94	BA93	BA92	BA91	BA90	BA8F	BA8E	BA8D	BA8C	BA8B	BA8A	BA89	BA88	BA87	BA86	BA85	BA84	BA83	BA82	BA81	BA80
6	BA7F	BA7E	BA7D	BA7C	BA7B	BA7A	BA79	BA78	BA77	BA76	BA75	BA74	BA73	BA72	BA71	BA70	BA6F	BA6E	BA6D	BA6C	BA6B	BA6A	BA69	BA68
5	BA67	BA66	BA65	BA64	BA63	BA62	BA61	BA60	BA5F	BA5E	BA5D	BA5C	BA5B	BA5A	BA59	BA58	BA57	BA56	BA55	BA54	BA53	BA52	BA51	BA50
4	BA4F	BA4E	BA4D	BA4C	BA4B	BA4A	BA49	BA48	BA47	BA46	BA45	BA44	BA43	BA42	BA41	BA40	BA3F	BA3E	BA3D	BA3C	BA3B	BA3A	BA39	BA38
3	BA37	BA36	BA35	BA34	BA33	BA32	BA31	BA30	BA2F	BA2E	BA2D	BA2C	BA2B	BA2A	BA29	BA28	BA27	BA26	BA25	BA24	BA23	BA22	BA21	BA20
2	BA1F	BA1E	BA1D	BA1C	BA1B	BA1A	BA19	BA18	BA17	BA16	BA15	BA14	BA13	BA12	BA11	BA10	BA0F	BA0E	BA0D	BA0C	BA0B	BA0A	BA09	BA08
1	BA07	BA06	BA05	BA04	BA03	BA02	BA01	BA00	B9FF	B9FE	B9FD	B9FC	B9FB	B9FA	B9F9	B9F8	B9F7	B9F6	B9F5	B9F4	B9F3	B9F2	B9F1	B9F0
0	B9EF	B9EE	B9ED	B9EC	B9EB	B9EA	B9E9	B9E8	B9E7	B9E6	B9E5	B9E4	B9E3	B9E2	B9E1	B9E0	B9DF	B9DE	B9DD	B9DC	B9DB	B9DA	B9D9	B9D8

LOOK-OUT

```
1       MODE 0
2       PRINT CHR$(12)
3       CURSOR 23,14:PRINT "Look out!"
4       CURSOR 23,13:PRINT "========="
5       CURSOR 32,10:PRINT "Gios A"
6       IF GETC=0 THEN 6
7       POKE #75,32
9       P1%=15.0:P2%=0.0:N2%=5.0:N3%=0.0:AAA%=0.0:AA2%=-1.0:AA3%=0.0
10      A3%=1.0:A2%=51.0:AA%=0.0
20      C0%=0.0:C1%=7.0:C2%=8.0:C3%=13.0
30      ENVELOPE 0 15,5;5,5;
40      ENVELOPE 1 10,30;5,30;
50      MODE 4A
51      PRINT CHR$(12)
52      CURSOR 23,3:PRINT "Look out !"
53      CURSOR 10,2:PRINT "Kaart"
60      COLORG 0 0 0 0
61      RESTORE
65      IF P2%<86 GOTO 71
66      FOR N1%=0.0 TO 15.0:READ X1%,Y1%,X2%,Y2%:NEXT:AA2%=0.0
67      IF P2%=86.0 THEN AA3%=AA3%+1.0:IF AA3%=1.0 THEN P1%=25.0:N2%=N2%+1.0
71      FOR N1%=0.0 TO 22.0:READ X1%,Y1%,X2%,Y2%:DRAW X1%,Y1% X2%,Y2% 22:NEXT
75      CURSOR 10,1:PRINT P2%+1.0
76      FOR N%=1.0 TO N2%:CURSOR 42+N%*2,1:PRINT "X":NEXT
77      FOR N1%=0.0 TO 10.0
78      NN1%=INT(RND(43.0)):NN2%=118.0+INT(RND(XMAX-118.0)):NN3%=87.0+INT(RND(104.0-87.0)
79      NN4%=20.0+INT(RND(YMAX-20.0)):NN5%=INT(RND(XMAX))
80      DOT NN1%,NN4% 22
81      DOT NN2%,NN4% 22
82      DOT NN5%,NN3% 22
83      NEXT
85      FOR N1%=0.0 TO P1%
86      X%=45.0+INT(RND(115.0-45.0)):Y%=20.0+INT(RND(84.0-20.0))
87      DOT X%,Y% 22
88      NEXT
89      SOUND 0 0 15 0 FREQ(800.0):WAIT TIME 5
90      COLORG C0% C1% C2% C3%
91      SOUND OFF
92      DRAW 85,48 85,54 0:WAIT TIME 10
93      DRAW 85,48 85,54 22:WAIT TIME 10
97      IF GETC=0 GOTO 92
98      NOISE 1 15
99      A2%=A2%+AA2%
100     A1%=A3%+1.0:IF SCRN(A2%,A1%)=C2% GOTO 1000
110     DOT A2%,A1% C3%
120     G%=GETC:IF G%>15.0 AND G%<20.0 THEN ON G%-15 GOTO 120,200,300,399
130     DOT A2%,A3% C0%
140     A3%=A1%:AA%=AA%+1.0
150     GOTO 100
199     A2%=A2%+1.0
200     A1%=A3%-1.0:IF SCRN(A2%,A1%)=C2% GOTO 1000
210     DOT A2%,A1% C3%
220     G%=GETC:IF G%>15.0 AND G%<20.0 THEN ON G%-15 GOTO 99,220,299,400
230     DOT A2%,A3% C0%
240     A3%=A1%:AA%=AA%+1.0
250     GOTO 200
299     A3%=A3%-1.0
300     A4%=A2%-1.0:IF SCRN(A4%,A3%)=C2% GOTO 1000
310     DOT A4%,A3% C3%
320     G%=GETC:IF G%>15.0 AND G%<20.0 THEN ON G%-15 GOTO 99,200,320,400
330     DOT A2%,A3% C0%
340     A2%=A4%:AA%=AA%+1.0
350     GOTO 300
399     A3%=A3%+1.0
400     A4%=A2%+1.0:IF SCRN(A4%,A3%)=C2% GOTO 1000
410     DOT A4%,A3% C3%
420     G%=GETC:IF G%>15.0 AND G%<20.0 THEN ON G%-15 GOTO 99,199,300,420
```

```
430    DOT A2%,A3% C0%
440    A2%=A4%:AA%=AA%+1.0
450    GOTO 400
1000   IF A2%=84.0 AND A3%>48.0 AND A3%>54.0 GOTO 2000
1010   NOISE 0 15
1020   WAIT TIME 30
1030   SOUND OFF :AAA%=AAA%+AA%
1040   N2%=N2%-1.0:IF N2%>0.0 GOTO 10
1050   GOTO 3000
2000   P1%=P1%+5.0:P2%=P2%+1.0
2010   SOUND OFF
2020   FOR N1%=1.0 TO P2%
2030   COLORG 0 0 14 0
2040   SOUND 1 0 15 0 FREQ(2000.0):WAIT TIME 5:COLORG C0% C1% C2% C3%
2050   SOUND 1 0 15 2 FREQ(600.0):WAIT TIME 20
2060   NEXT:SOUND OFF :AAA%=AAA%+AA%
2070   GOTO 10
3000   PRINT CHR$(12)
3010   MODE 0:PRINT :PRINT :PRINT :PRINT :PRINT "                         LOOK OUT!"
3020   PRINT :PRINT :PRINT "                UW SCORE :";AAA%;:PRINT " ";P2%+1.0
3030   IF AAA%>BB% THEN BB%=AAA%
3040   PRINT :PRINT :PRINT "            De hoogste score";BB%
3050   PRINT :PRINT :PRINT "         VOOR EEN NIEW SPEL DRUK EEN TOETS"
3060   IF GETC=0.0 GOTO 3060
3070   WAIT TIME 5
3080   IF GETC=0 GOTO 3080
3090   GOTO 9
5000   DATA 45,20,45,84,45,84,115,84,115,84,115,20,115,20,55,20
5010   DATA 55,20,55,74,55,74,105,74,105,74,105,30,105,30,65,30
5020   DATA 65,30,65,64,65,64,95,64,95,64,95,40,95,40,75,40
5030   DATA 75,40,75,54,75,54,85,54,85,54,85,48,85,48,82,48
5040   DATA 45,20,25,0,45,84,25,104,115,84,135,104,115,20,135,0
5050   DATA 0,50,45,50,115,50,159,50,0,0,159,0
5060   DATA 45,20,45,84,45,84,115,84,115,84,115,20,115,20,55,20
5070   DATA 55,20,55,74,65,84,65,30,75,20,75,54,75,54,105,54
5080   DATA 65,64,95,64,75,74,105,74,75,40,95,40,82,48,85,48
5090   DATA 85,30,105,30,85,48,85,54,95,40,95,47,105,30,105,74
```

```
100    REM *** UPPER TO LOWER CASE : DEMO ****************
110    REM *** GESCHREVEN DOOR : DE BONT CORNEEL *********
120    REM *** (NAAR HET PROGRAMMA VAN J.BOERRIGTER ******
130    REM *** UIT NEWSLETTER 16 : PAGINA 174 ) **********
140    REM ********************************************
200    CLEAR 5000:POKE #29B,#FF:POKE #29C,5:PRINT CHR$(12)
210    FOR X=#400 TO #43F:CURSOR 20,20:PRINT #43F-X;" ";
220    READ A:POKE X,A:NEXT:PRINT CHR$(12);
230    PRINT " ZIEHIER DATA IN UPPER CASE.":LIST 310-400
240    CALLM #400
250    PRINT " ZIEHIER GETRANSFORMEERDE DATA.":LIST 310-400
300    REM *** UPPER TO LOWER CASE MLP
310    DATA #F5,#C5,#D5,#E5,#2A,#A1,#02,#EB,#2A,#9F,#02,#06
320    DATA #00,#CD,#14,#DE,#D2,#3B,#04,#CA,#3B,#04,#4E,#23
330    DATA #23,#23,#7E,#FE,#A2,#CA,#26,#04,#2B,#2B,#09,#C3
340    DATA #0D,#04,#23,#4E,#0C,#23,#0D,#CA,#0D,#04,#7E,#CD
350    DATA #02,#DE,#D2,#29,#04,#C6,#20,#77,#C3,#29,#04,#E1
360    DATA #D1,#C1,#F1,#C9,#00,#00,#00,#00,#00,#00,#00,#00
370    DATA DIT PROGRAMMA LAADT EEN KORTE MLP-ROUTINE IN RAM
380    DATA BIJ EEN RUN ZAL DEZE ROUTINE ALLE KARAKTERS,AAN-
390    DATA WEZIG IN DATA-LINES OMVORMEN VAN UPPER CASE NAAR
400    DATA LOWER CASE.zie ter demo deze run..
```

PRINT ROUTINES IN THE DAI

The DAI has in its firmware several very useful routines for printing of strings and numbers. These routines can easily be used in your own machine language programs.
This articles describes several of these print routines. For more information is referred to the 'DAI firmware manual'.

In the examples given, it is assumed that the string to be printed is in memory, and starts at address XXXX.
As an example, always the string "TEST" will be used.

1. PRINT A STRING:

1.1. This routine is at address #DB32.
On entry, HL must contain the stringaddress.

The format of the string must be as follows:

- A length byte.
- The string in ASCII.

Program example:

```
XXXX  04 - 54.45.53.54   ('TEST' in ASCII)

      LXI H,:XXXX    Get stringaddr in HL
      CALL  :DB32    Print 'TEST'
```

On exit, HL points after the string. All other registers are preserved.

1.2. An alternative routine can be found on address #DB44.
On entry, HL points to the string. Its length must be in A.

Program example:

```
XXXX  54.45.53.54        ('TEST' in ASCII)

      LXI H,:XXXX   Get stringaddr in HL
      MVI A,:04     Length in A
      CALL  :DB44   Print 'TEST'
```

The exit conditions are identical to routine 1.1.

2. PRINT A MESSAGE:

2.1. This routine can be found on address #DAD4. It is a subroutine with additional possibilities. It can be used for printing of strings, which in itself, refer to other strings.

On entry, HL must point to the string. On exit, HL points after the string. All other registers are preserved.

2.1.1. Format of a simple string:

- String bytes in ASCII. All bytes must be between #01 and #7F.
- 00 (end of string).

Program example:

```
XXXX  54.45.53.54 - 00    ('TEST' in ASCII)

      LXI H,:XXXX   HL points to string
      CALL  :DAD4   Print 'TEST'
```

2.1.2. Format of a message with internal reference to other submessages:

- The first byte must be >= #80. This indicates the presence of a subreference message.
- If of this first byte, bit 14=1, then the lower bits 0-13 must be added to #C000 to find the address of the message. This message must again end with 00.
- If bit 14 of the first byte is 0, then the address found by adding bits 0-13 to #C000 is the address of a string, consisting of a length byte + characters in ASCII.

Program example:

```
DD0A  8D.1B    'LOAD'
      DB.F3    'ING'
      DC.15    ' ERROR'
      20
      00

      LXI H,:DD0A   Address message
      CALL  :DAD4   Print 'LOADING ERROR'

8D1B: Bit 15=1: Subreference message.
      Bit 14=0: Points to string with address
                C000 + 0D1B = CD1B:
                04 - 4C.4F.41.44 ('LOAD').

DBF3: Bit 15=1: Subreference message.
      Bit 14=1: Points to message with address
                C000 + 1BF3 = DBF3:
                49.4E.47 - 00 ('ING').

DC15: Bit 15=1: Subreference message.
      Bit 14=1: Points to message with address
                C000 + 1C15 = DC15:
                20.45.52.52.4F.52 - 00 (' ERROR').

20  : Bit 15=0: Simple string byte.
00  :           End of message.
```

Several other examples can be found in the messages on the addresses #DB6F - #DD19.

2.2. Another routine to print messages can be found on the address #DAFF.
It print messages in exactly the same way as the routine on #DAD4, but the routine is 'called' in a different way.

Program example:

```
XXXX        start of message (format see 2.1).

CALL  :DAFF     Print message with address
DBL   :XXXX     given as datablock.
```

This datablock address is taken from stack, the stack-pointer is updated to after the datablock, and the message is printed.
On exit, all registers are preserved.

2.3. A special form of routine 2.2 can be found on address #CEE4. This one is used if an error occurs during working in a switched ROM-bank.
Before printing the error message with routine 2.2, the ROM bank 0 is selected.

3. SEVERAL USEFUL PRINT ROUTINES:

3.1. Routines which can be used always:

3.1.1. #CE68: Print an expression, followed by a space.

3.1.2. #CE6B: Print a space.

3.1.3. #CE70: Print a comma.

3.1.4. #CE75: Print a string between spaces.
Before and after the string, a space is printed.

```
Program example: CALL  :CE75
                 DBL   :XXXX
```

3.1.5. #DB0D: Cursor to next field. To be used as 'tab' to get cursor to the next field. The field positions are 0,12,24,36,48.

3.1.6. #DB2A: Cursor to column 8.

3.1.7. #DD5E: Print a carriage return.

3.1.8. #DD60: Print a character, which is in A and in ASCII format.

3.2. To be used only in BASIC with a CALLM-instruction:

3.2.1. #0EFBD-#0EFE0: Several useful LIST routines. This routines can only be used if the m.l.routine is called from a BASIC program with CALLM, because they are in ROM bank 0.
The numbers to be printed must be in the math. accumulator.

One program example:

```
LXI H,:0010  '0010' is decimal 16
CALL  :EB46  Number into MACC
CALL  :EFBD  Convert MACC from binary to ASCII
             and print result in decimal: '16'
```

3.3. To be used only in programs running under the Utility monitor (in ROM-bank 3):

3.3.1. #ED18: Print an double-byte number. The number must be in HL.

3.3.2. #ED1D: Print a single-byte number, which is in A.

3.3.3. #ED2F: Print a string. HL points to the string. The format is: <string bytes in ASCII> - 00.

3.3.4. #ED3A: Print a carriage return.

3.3.5. #EEB4: Print a character. The character must be in register C and in ASCII-format.

3.4. Routines for printing numbers, which are in the math. accumulator:

3.4.1. #DB4A: Print a number in the MACC in hex format.

3.4.2. #DB53: Print a number in the MACC in integer format.

3.4.3. #DB59: Print a number in the MACC in floating point format.

(C) - Jan Boerrigter - Aug. 1984

---

cont. from p. 330

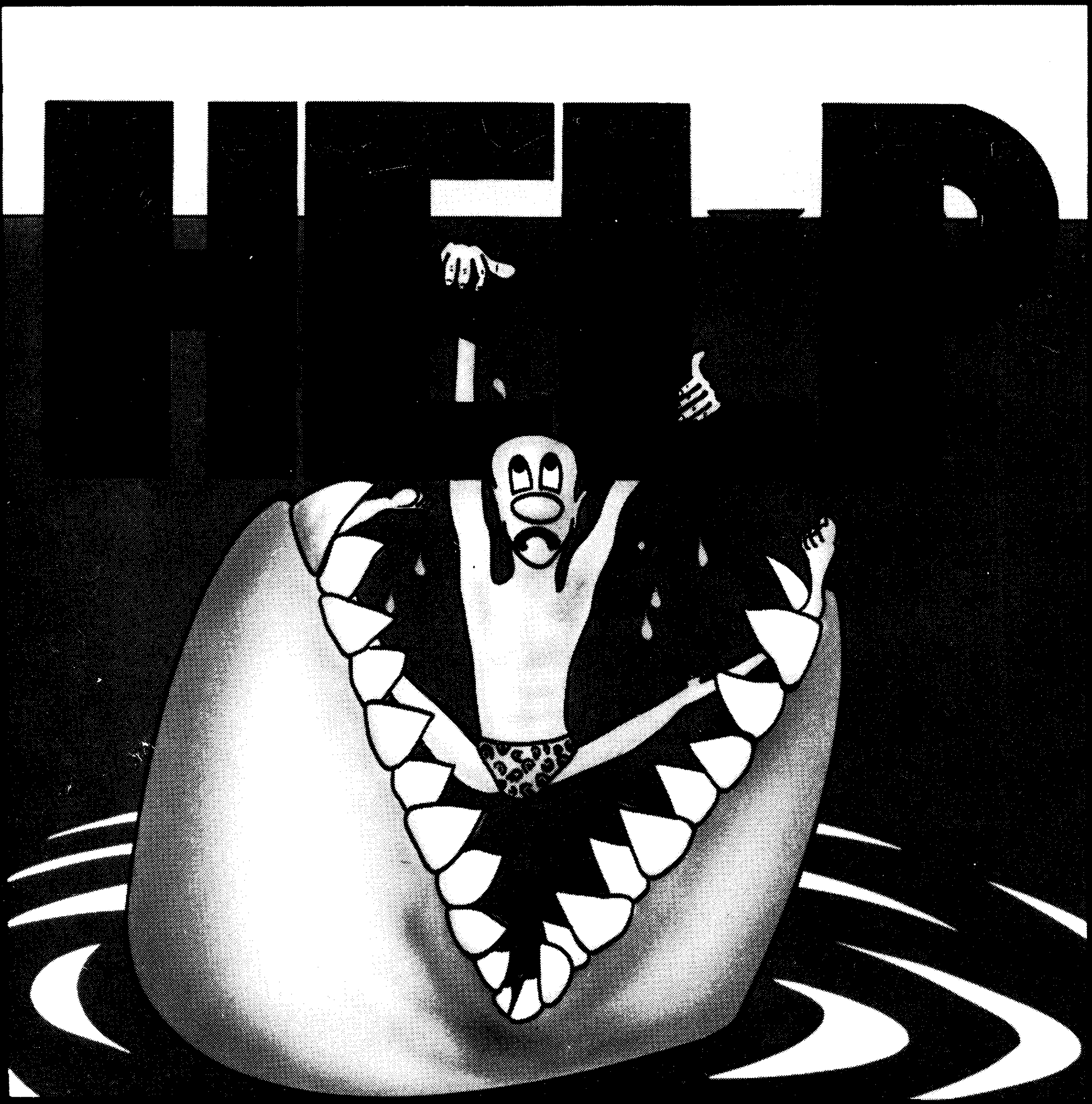
(D.BASIC.3)

```
;
        TITL    'HOME : HOME EXTENSION'
;
        ORG     0H
HOMROT  DB      4H
        DB      'HOME'
        DB      0BH
        DB      0EEH
        DW      RELTBL
        DB      ',;#  ./='
        DB      4H
        DB      'HOME'
        DB      0C2H
REL010  DW      RHOME
        DB      0H
;
RHOME   MVI A   0CH
        RST 5
        DB      3H
        ORA A
        RET
;
RELTBL  DW      REL010
        DW      0H
;
        END
;
```

00 NOP	10 ---	20 ---	30 ---	40 MOV B,B
01 LXI B addr	11 LXI D addr	21 LXI H addr	31 LXI SP addr	41 MOV B,C
02 STAX B	12 STAX D	22 SHLD addr	32 STA addr	42 MOV B,D
03 INX B	13 INX D	23 INX H	33 INX SP	43 MOV B,E
04 INR B	14 INR D	24 INR H	34 INR M	44 MOV B,H
05 DCR B	15 DCR D	25 DCR H	35 DCR M	45 MOV B,L
06 MVI B data	16 MVI D data	26 MVI H data	36 MVI M data	46 MOV B,M
07 RLC	17 RAL	27 DAA	37 STC	47 MOV B,A
08 ---	18 ---	28 ---	38 ---	48 MOV C,B
09 DAD B	19 DAD D	29 DAD H	39 DAD SP	49 MOV C,C
0A LDAX B	1A LDAX D	2A LHLD addr	3A LDA addr	4A MOV C,D
0B DCX B	1B DCX D	2B DCX H	3B DCX SP	4B MOV C,E
0C INR C	1C INR E	2C INR L	3C INR A	4C MOV C,H
0D DCR C	1D DCR E	2D DCR L	3D DCR A	4D MOV C,L
0E MVI C data	1E MVI E data	2E MVI L data	3E MVI A data	4E MOV C,M
0F RRC	1F RAR	2F CMA	3F CMC	4F MOV C,A
50 MOV D,B	60 MOV H,B	70 MOV M,B	80 ADD B	90 SUB B
51 MOV D,C	61 MOV H,C	71 MOV M,C	81 ADD C	91 SUB C
52 MOV D,D	62 MOV H,D	72 MOV M,D	82 ADD D	92 SUB D
53 MOV D,E	63 MOV H,E	73 MOV M,E	83 ADD E	93 SUB E
54 MOV D,H	64 MOV H,H	74 MOV M,H	84 ADD H	94 SUB H
55 MOV D,L	65 MOV H,L	75 MOV M,L	85 ADD L	95 SUB L
56 MOV D,M	66 MOV H,M	76 HLT	86 ADD M	96 SUB M
57 MOV D,A	67 MOV H,A	77 MOV M,A	87 ADD A	97 SUB A
58 MOV E,B	68 MOV L,B	78 MOV A,B	88 ADC B	98 SBB B
59 MOV E,C	69 MOV L,C	79 MOV A,C	89 ADC C	99 SBB C
5A MOV E,D	6A MOV L,D	7A MOV A,D	8A ADC D	9A SBB D
5B MOV E,E	6B MOV L,E	7B MOV A,E	8B ADC E	9B SBB E
5C MOV E,H	6C MOV L,H	7C MOV A,H	8C ADC H	9C SBB H
5D MOV E,L	6D MOV L,L	7D MOV A,L	8D ADC L	9D SBB L
5E MOV E,M	6E MOV L,M	7E MOV A,M	8E ADC M	9E SBB M
5F MOV E,A	6F MOV L,A	7F MOV A,A	8F ADC A	9F SBB A
A0 ANA B	B0 ORA B	C0 RNZ	D0 RNC	E0 RPO
A1 ANA C	B1 ORA C	C1 POP B	D1 POP D	E1 POP H
A2 ANA D	B2 ORA D	C2 JNZ addr	D2 JNC addr	E2 JPO addr
A3 ANA E	B3 ORA E	C3 JMP addr	D3 OUT port	E3 XTHL
A4 ANA H	B4 ORA H	C4 CNZ addr	D4 CNC addr	E4 CPO addr
A5 ANA L	B5 ORA L	C5 PUSH B	D5 PUSH D	E5 PUSH H
A6 ANA M	B6 ORA M	C6 ADI data	D6 SUI data	E6 ANI data
A7 ANA A	B7 ORA A	C7 RST 0	D7 RST 2	E7 RST 4
A8 XRA B	B8 CMP B	C8 RZ	D8 RC	E8 RPE
A9 XRA C	B9 CMP C	C9 RET	D9 ---	E9 PCHL
AA XRA D	BA CMP D	CA JZ addr	DA JC addr	EA JPE addr
AB XRA E	BB CMP E	CB ---	DB IN port	EB XCHG
AC XRA H	BC CMP H	CC CZ addr	DC CC addr	EC CPE addr
AD XRA L	BD CMP L	CD CALL addr	DD ---	ED ---
AE XRA M	BE CMP M	CE ACI data	DE SBI data	EE XRI data
AF XRA A	BF CMP A	CF RST 1	DF RST 3	EF RST 5
F0 RP	F8 RM			
F1 POP PSW	F9 SPHL			
F2 JP addr	FA JM addr			
F3 DI	FB EI			
F4 CP addr	FC CM addr			
F5 PUSH PSW	FD ---			
F6 ORI data	FE CPI data			
F7 RST 6	FF RST 7			

	Ø	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Ø	NOP	---	---	---	MOV B,B	MOV D,B	MOV H,B	MOV M,B	ADD B	SUB B	ANA B	ORA B	RNZ	RNC	RPO	RP
1	LXI B bb	LXI D bb	LXI H bb	LXI SP bb	MOV B,C	MOV D,C	MOV H,C	MOV M,C	ADD C	SUB C	ANA C	ORA C	POP B	POP D	POP H	POP PSW
2	STAX B	STAX D	SHLD bb	STA bb	MOV B,D	MOV D,D	MOV H,D	MOV M,D	ADD D	SUB D	ANA D	ORA D	JNZ bb	JNC bb	JPO bb	JP bb
3	INX B	INX D	INX H	INX SP	MOV B,E	MOV D,E	MOV H,E	MOV M,E	ADD E	SUB E	ANA E	ORA E	JMP bb	OUT p	XTHL	DI
4	INR B	INR D	INR H	INR M	MOV B,H	MOV D,H	MOV H,H	MOV M,H	ADD H	SUB H	ANA H	ORA H	CNZ bb	CNC bb	CPO bb	CP bb
5	DCR B	DCR D	DCR H	DCR M	MOV B,L	MOV D,L	MOV H,L	MOV M,L	ADD L	SUB L	ANA L	ORA L	PUSH B	PUSH D	PUSH H	PUSH PSW
6	MVI B b	MVI D b	MVI H b	MVI M b	MOV B,M	MOV D,M	MOV H,M	HLT	ADD M	SUB M	ANA M	ORA M	ADI b	SUI b	ANI b	ORI b
7	RLC	RAL	DAA	STC	MOV B,A	MOV D,A	MOV H,A	MOV M,A	ADD A	SUB A	ANA A	ORA A	RST Ø	RST 2	RST 4	RST 6
8	---	---	---	---	MOV C,B	MOV E,B	MOV L,B	MOV A,B	ADC B	SBB B	XRA B	CMP B	RZ	RC	RPE	RM
9	DAD B	DAD D	DAD H	DAD SP	MOV C,C	MOV E,C	MOV L,C	MOV A,C	ADC C	SBB C	XRA C	CMP C	RET	---	PCHL	SPHL
A	LDAX B	LDAX D	LHLD bb	LDA bb	MOV C,D	MOV E,D	MOV L,D	MOV A,D	ADC D	SBB D	XRA D	CMP D	JZ bb	JC bb	JPE bb	JM bb
B	DCX B	DCX D	DCX H	DCX SP	MOV C,E	MOV E,E	MOV L,E	MOV A,E	ADC E	SBB E	XRA E	CMP E	---	IN p	XCHG	EI
C	INR C	INR E	INR L	INR A	MOV C,H	MOV E,H	MOV L,H	MOV A,H	ADC H	SBB H	XRA H	CMP H	CZ bb	CC bb	CPE bb	CM bb
D	DCR C	DCR E	DCR L	DCR A	MOV C,L	MOV E,L	MOV L,L	MOV A,L	ADC L	SBB L	XRA L	CMP L	CALL bb	---	---	---
E	MVI C b	MVI E b	MVI L b	MVI A b	MOV C,M	MOV E,M	MOV L,M	MOV A,M	ADC M	SBB M	XRA M	CMP M	ACI b	SBI b	XRI b	CPI b
F	RRC	RAR	CMA	CMC	MOV C,A	MOV E,A	MOV L,A	MOV A,A	ADC A	SBB A	XRA A	CMP A	RST 1	RST 3	RST 5	RST 7

DIALOG INFORMATIQUE



HELP

Quel temps aujourd'hui ! Tu as entendu la météo ? ... «Soleil radieux sur la côte ouest pour la journée» ...
Cela promet !, la mer est déjà noire de monde. Enfin... esperons que les coucous publicitaires ne nous poserons pas de problème.
Avec tous ces baigneurs, viléplanchistes, et autres bateaux de plaisance; je sens que les problèmes ne vont pas tarder : «ILS» vont être attirés comme des abeilles sur un pot de miel.
Bon... je crois que le plein est fait: monttons dans l'helicoptère sauver ces bronzés des dents tranchantes des «SQAULES»

HELP (S.O.S. HELI)

Start your helicopter and fly above the sea to save the drowning persons. Get your heli in position and let down your ladder, they will climb on board... during your S.O.S.operations, look out for collision with other aircrafts and drop your bombs to kill the hungry sharks !!
The more lifes you can save, the more points you score... but don't take too many persons on board, your cargo is limited !