METEORIC PROGRAMMING

For the ORIC-1

JOHN VANDER REYDEN

CONTENTS

METEORIC PROGRAMMING

For the ORIC-1

METEORIC PROGRAMMING

For the ORIC-1

JOHN VANDER REYDEN

MELBOURNE HOUSE

First Published in the United Kingdom
by Melbourne House

This Remastered Edition
Published in 2022 by
Acorn Books
acornbooks.uk

Publisher's Note

In keeping with our ongoing commitment to provide both literature and software for personal computers, Melbourne House is very proud to be able to publish this book of games and general routines for the Oric-1.

You will find complete program listings, comprehensive structures and useful hints on each program. We have taken care to design the format in such a way that the programs will be easily read, to reduce the possibility of transcription errors, especially with the graphics characters.

After working through this book, I think you will agree that the programs should set the standard by which future programs for the Oric-1 will be judged by.

I know you will enjoy not only the programs themselves, but also the insight you will gain about programming the Oric-1.

Melbourne House are not just publishers - we are dedicated to microcomputer software, and we are always interested in your feedback: If you have an article or program that you think might be of value to other users we want to hear from you.

Until I hear from you, let me just wish you happy programming!

[illegible]

ALFRED MILGROM
PUBLISHER

SPECIAL NOTE

You may have noticed that the listings in the book don't look exactly like the listings you normally see on your screen.

When a program is listed or printed out on the ORIC the keywords are all compressed. The problem with this is that it makes it very hard for anyone to read the program, and harder still if you have to read a part of a line, key it in, and still know exactly where you were in the program line.

To overcome this problem and make it easier for you to key the listings in we have inserted spaces on either side of all colons(:), semi-colons(;), and commas(,) as well as any other places where there could be difficulty in reading the listing. For example, a normal listing would look like this:

```
10 CURMOV6,0,3:PRINTA$;"-";
```

The same line in the book would look like this:

```
10 CURMOV6 , 0 , 3 : PRINT A$ ; "-" ;
```

When you key the programs in you should not key any spaces in other than where they occur within quotation marks.

If you find that you can't enter a whole line go back and check that you have not entered any spaces that should not be there.

Where the number of blank spaces inside of quotation marks is critical or not easily determined we have noted the number of spaces thus: 12#

Notes for Programmers

The following notes are meant to be read by all persons using the programs in this book. Some are just tips from our own experience and some are things that you won't find in the manual.

KEYBOARD SCANNING

As you may know the KEY$ function on the ORIC does not allow for the keys to repeat (i.e. to get KEY$ to recognize a key more than once you must release the key and press it again). This is not very useful for games where you are trying to move and fire your ship (or whatever). You can overcome this problem by peeking a location in ORIC's memory which changes according to the key being held down. The memory location is 520 decimal.

The following table may save you some trouble.

CHARACTER	PEEK (520)	ASCII
	172	8
	188	9
	180	10
	156	11
space	132	32
null	56	error

See also Sound Aid (Play mode) for an advanced method of keyboard scanning.

CONTROL CHARACTERS

Control characters are used to determine the mode of certain operations such as Caps lock, keyclick, double height, etc. The only problem with control characters is that they toggle. This means that if a program uses a control character, and the program is re-run, then the control will toggle to its unwanted state. To overcome this, you can poke a value directly into the memory location which controls these modes. This location is 618 decimal. The most common setting is Cursor-off and keyclick-off. This is achieved by the statement POKE 618,10. Other values may be found by setting the states using control characters then typing PRINT PEEK (618)

USING CURMOV

You cannot use CURMOV without first having set CURSET.

ERROR MESSAGES

If you crash a program and can't see/read the error message just type Paper (n). The screen goes back into text mode and displays the error.

CURSOR POSITION

You may find, when programming, that printing numbers on the screen with the PLOT command can be messy or you may find it difficult to position an INPUT statement. These and other problems can be overcome with the ability to position the PRINT cursor anywhere on the screen and using the PRINT statement.

The programming method is as follows:

```
POKE 616, vertical position
PRINT
POKE 617, horizontal position
PRINT anything
```

Using the same principle a downward screen scroll may be achieved as follows:

```
FOR K = 1 to 25
POKE 616, 255
PRINT
NEXT K
```

CSAVE

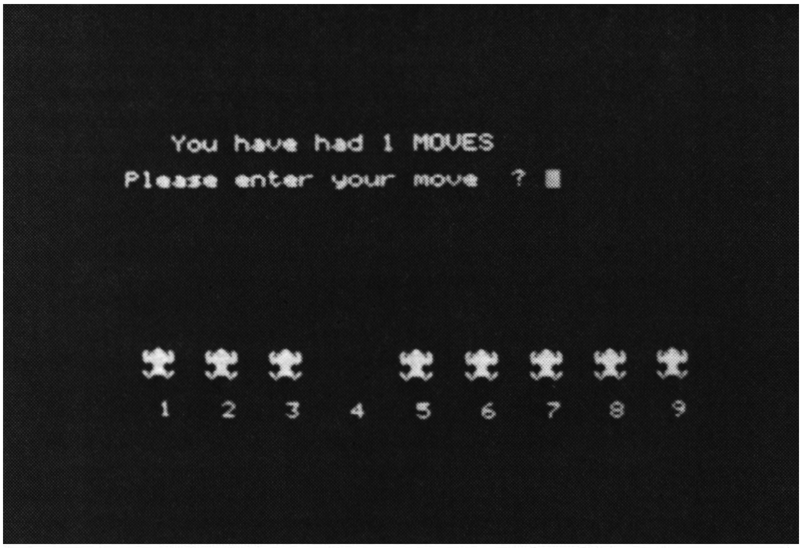
We recommend that you save all the programs before running them. To be on the safe side, and to make sure you don't have to start keying in again, we advise you to use the slow rate. Once you have a "slow" master copy then make a fast one for normal use.

STR$ FUNCTION

When using this function don't forget that the first character in the string will be used for the sense (+, -) of the numerical value. This can also affect the colour used if you print the string.

e.g. STR$ (8)= "STX 8"= 2 characters

Leapfrog



```
   You have had 1 MOVES
Please enter your move  ? █

  1   2   3   4   5   6   7   8   9
```

This game makes a good mind exercise. The idea is simple. Move all the frogs from the left side to the right side and vice-versa by hopping over only one frog at a time. If you succeed, see if you can do it again in less moves.

STRUCTURE

FUNCTION	LINE(s)
Initialization	100 - 170
Display set-up	180 - 220
Prompt for move	240
Test if move valid	250 - 280
Move frogs	300 - 330
Game-end routine	350 - 410
Position-INPUT routine	420 - 450
Create frog characters	430 - 520
Frog character data	460 - 490

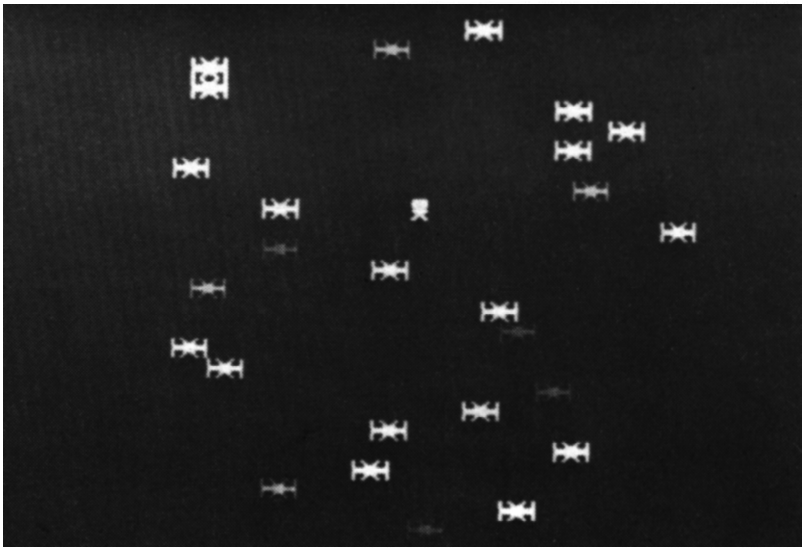
VARIABLES

A1$	= Left set of frogs
A2$	= Right set of frogs
CN	= Move-counter
B1$	= Frog being moved
B2$	= Frog being replaced
k$	= Input from player

```
100 PAPER0 : INK2
110 GOSUB 430                                                    4#
120 A1$=CHR$(2)+"AB " : A1$=A1$+A1$+A1$+A1$+"    "
130 A1$=A1$+CHR$(3)+"AB "+CHR$(3)+"AB "+CHR$(3)+"AB
    "+CHR$(3)+"AB"
140 A2$=CHR$(2)+"CD " : A2$=A2$+A2$+A2$+A2$+"    "
150 A2$=A2$+CHR$(3)+"CD "+CHR$(3)+"CD "+CHR$(3)+"CD          1#
    "+CHR$(3)+"CD"
160 B1$=A1$ : B2$=A2$
170 CN=0
180 CLS : PLOT3 , 23 , "1   2   3   4   5   6   7   8
    9"                                                           3#
190 PLOT1 , 20 , B1$ : PLOT1 , 21 , B2$
200 IFLEFT$(B1$ , 15)=RIGHT$(A1$ , 15)ANDRIGHT$(B1$ ,
    15)=LEFT$(A1$ , 15)THENGOTO350
210 PLOT1 , 8 , "   You have had"+STR$(CN)+" moves"
220 PLOT1 , 10 , "Please enter your move       "
230 GOSUB420
240 INPUT K$
250 IF LEN(K$)<>2THENGOTO210
260 FR=4*(ASC(LEFT$(K$ , 1))-49)+1
270 TR=4*(ASC(RIGHT$(K$ , 1))-49)+1
280 IF(MID$(B1$ , TR ,
    1)<>" "ORABS(TR-FR)>8)THENGOTO210
290 CN=CN+1
300 B1$=LEFT$(B1$ , TR-1)+MID$(B1$ , FR ,
    3)+MID$(B1$ , TR+3)
310 B2$=LEFT$(B2$ , TR-1)+MID$(B2$ , FR , 3)+MID$(B2$
    , TR+3)                                                      3#
320 B1$=LEFT$(B1$ , FR-1)+"   "+MID$(B1$ , FR+3)
330 B2$=LEFT$(B2$ , FR-1)+"   "+MID$(B2$ , FR+3)
340 GOTO190
350 PLOT3 , 8 , "You did it in"+STR$(CN)+" MOVES"
360 PLOT1 , 10 , " Do you want another go?        "
370 GOSUB420
380 GETK$
390 IF K$="Y"THENGOTO160
400 IF K$<>"N"THENGOTO360
410 CLS : END
420 POKE618 , 10 : POKE616 , 10 : PRINT : POKE617 ,
    26 : POKE618 , 3 : RETURN
430 FORI=46600TO46631
440 READX : POKEI , X
450 NEXTI
460 DATA1 , 19 , 53 , 39 , 63 , 63 , 23 , 3
470 DATA32 , 50 , 43 , 57 , 63 , 63 , 58 , 48
480 DATA3 , 3 , 39 , 55 , 20 , 8 , 0 , 0
490 DATA48 , 48 , 57 , 59 , 10 , 4 , 0 , 0
500 PLOT1 , 20 , CHR$(9)
510 PLOT0 , 21 , CHR$(9)
520 RETURN
```

Where the number of blank spaces inside of quotation marks is critical or not easily determined we have noted the number of spaces thus: 12#.

Asteroids in Space



You are travelling through space when you encounter a space storm. You must avoid the asteroids by moving left and right using the "←" and "→" keys. After some time you end up in the heart of the storm and you must negotiate your way between much larger asteroids. Eventually you will crash, when you do you will be given a survival rating.

HOW THE PROGRAM WORKS

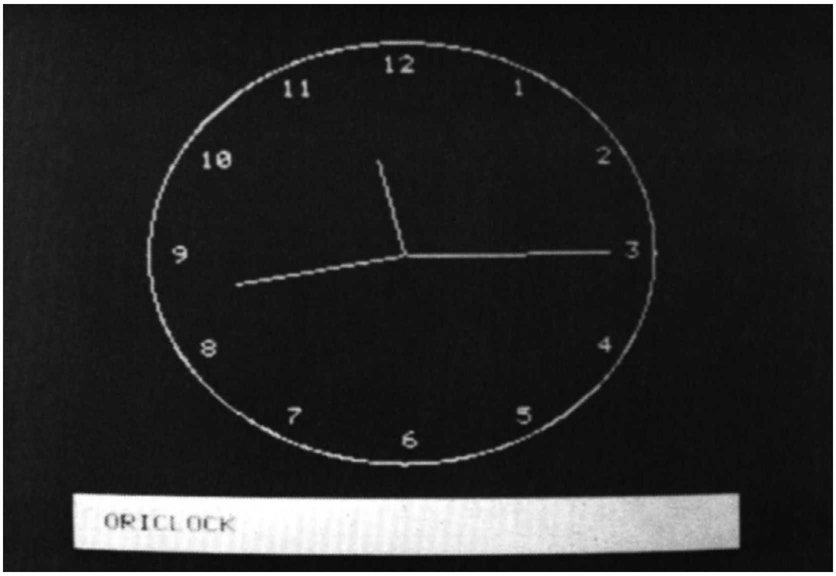
This program uses the PRINT function to scroll the screen upwards. When the display is scrolled, your ship will move with everything else, so it is overwritten with spaces then printed again in the correct position. In this way your ship stays on the same line while everything else moves. The creatures and space ships appear randomly at the bottom of the screen. The screen is scrolled two lines each cycle so that there are not too many obstacles on the screen at any one time.

IMPROVING THE PROGRAM!

The program has been kept deliberately simple to enable you to improve it. Firstly, you can create more stages using the user defined character set on the Oric. The test for the second stage is in line 270. Secondly, more sound could be included, if only to slow the program up somewhat. You can slow the program by changing the length of the notes in your sounds. Thirdly, you could include the option to fire at the objects on the screen, adding their values to your score, and so on!

```
100 REM * ASTEROIDS *
110 REM
120 PAPER0 : INK7 : CLS
130 PRINT CHR$(17)
140 PLOT 15 , 2 , "FIND THE GAP...."
150 PLOT 15 , 3 , "ASTEROIDS IN SPACE"
160 FOR J=35 TO 38
170 FOR I=0 TO 7
180 READ K
190 POKE 46080+I+(J*8) , K
200 PLOT P , X , "&"
210 NEXT I
220 NEXT J
230 A$="#"
240 N=0
250 T=1
260 P=19 : X=10
270 R=INT(RND(1)*35)+3
280 PLOT R-1 , 26 , CHR$(INT(RND(1)*7)+1)
290 PLOT R , 26 , A$
300 PLOT P , X-2 , " "
310 PLOT P , X , "&" : PLOT P-1 , X , CHR$(2) :
    PLOT P+1 , X , CHR$(0)
320 N=N+T
330 IF N=100 THEN A$="$%"
340 IF N=104 THEN T=2
350 PLOT P , X-1 , " "
360 PLOT P , X , "&"
370 IF SCRN(P , X+2)<>32 THEN 450
380 IF SCRN(P+T , X+2)<>32 THEN 450
390 IF SCRN(P-T , X+2)<>32 THEN 450
400 X$=KEY$
410 IF X$="W" THEN PLOT P , X , " " :
    P=P-T : IF P<3 THEN P=37
420 IF X$="E" THEN PLOT P , X , " " : P=P+T :
    IF P>37 THEN P=3
430 PRINT : PRINT
440 GOTO 270
450 EXPLODE
460 PRINT"your survival rating ";N
470 PRINT CHR$(17)
480 PRINT"do you want another game , Y/N" :
    INPUT A$ : IF A$="Y"THEN RUN
490 DATA 45 , 45 , 30 , 61 , 51 , 30 , 33 , 63
500 DATA 36 , 34 , 35 , 63 , 63 , 35 , 34 , 36
510 DATA 9 , 17 , 49 , 63 , 63 , 49 , 17 , 9
520 DATA 31 , 21 , 31 , 4 , 31 , 4 , 10 , 17
530 END
```

Oriclock



This program is a simulation of a REAL TIME CLOCK. The clock has alarm facilities and ticks away for as long as you like to leave it. The alarm is bound to wake you (by driving you mad). If you find it is still not loud enough then increase the volume if you wish.

The clock is not extremely accurate as it uses the computer's "wait" instructions. You can adjust this wait statement to tune the clock's accuracy though it will always tend to vary in cycle time. The clock will stop on alarm, this is unavoidable.

The Oriclock is not meant to replace your clock but is a demonstration program with visual appeal and purpose. If anyone knows how to (if it is possible) get access to the frames controller (sends picture "frames" to video output) we would like to know as this will provide a stable oscillator to use to control the clock's accuracy.

HOW TO RUN THE PROGRAM

Type RUN followed by (ENTER) to start the program after having loaded it. You will need to enter the HOUR (12 or less), MINUTE and SECONDS at which you want the clock to start. The program checks the validity of your input to prevent you from entering wrong data. You will also be asked if you wish to set the alarm, to do so you enter the values you require in the same format as above.

ORICLOCK STRUCTURE:

INITIALIZATION:

```
1.  Display clock-face.
2.  Input hour, minute and seconds.
3.  If alarm set;
4.  Set alarm hours, minutes and seconds.
5.  Initialize starting time.
```

THE MAIN LOOP:

```
H. Determine hour hand position
M. Determine minute hand position
S. Determine second hand position
   Draw second, minute hour hands
   Display clock face
   If alarm time matches real time then alarm rings
   Wait 1 second
   Re-initialize time
   Draw over second hand
   If same minute then goto S:
   Draw over minute hand
   If same hour then goto M
   Draw over hour hand
   Goto H
```

HOW THE PROGRAM WORKS:

Sin and Cos functions are used to draw the clock hands. The second hand is moved every second, the minute hand every minute and the hour hand every 12 minutes.

CONSIDERATIONS:

Due to the fact that the clock is only using the "wait" instruction for timing, the clock is not very accurate. Therefore to get any reasonable accuracy. It would be necessary to divide into the clock timer chip within ORIC.
Line 1160 would then become:

```
1160 IF INT (TIMER/FREQ) < = 0 THEN 1160
```

where freq = 50 or 60 i.e. frequency of mains

```
   5 REM * ORICLOCK *
  10 CLS : PAPER1 : INK0 : N=1 : T=30
  20 GOTO 3000
1000 REM CLOCK MOVE
1005 PAPER1 : INK0
1020 IF HI=60 THEN HI=0
1040 H=HI*C
1050 HX=40*SIN(H) : HY=-40*COS(H)
1060 IF MI=60 THEN MI=0 : HT=HT+1
1065 IF HT=24 THEN HT=0
1070 M=MI*C
1080 MX=60*SIN(M) : MY=-60*COS(M)
1090 A=SI*C
1100 SX=72*SIN(A) : SY=-72*COS(A)
1110 REM DRAW HANDS
1120 CURSET 119 , 96 , 3 : DRAW SX , SY , 1
1130 CURSET 119 , 96 , 3 : DRAW MX , MY , 1
1140 CURSET 119 , 96 , 3 : DRAW HX , HY , 1
1155 IF(AL=1 AND HT=AH AND MI=AM AND SI=AS)THEN 2000
1157 N=7-N
1158 PLAY1 , 0 , 1 , 100 : MUSIC1 , 4 , N , 5 : PLAY0
     , 0 , 0 , 0
1160 WAIT(T)
1210 CURSET 119 , 96 , 3 : SI=SI+1 : DRAW SX , SY , 2
1220 IF SI=60 THEN SI=0 ELSE 1090
1230 CURSET 119 , 96 , 3 : MI=MI+1 : DRAW MX , MY , 2
1240 IF (MI-INT(MI/12)*12)=0 THEN HI=HI+1 ELSE 1060
1250 CURSET 119 , 96 , 3 : DRAW HX , HY , 2
1260 GOTO 1020
2000 PING
2010 A0$=KEY$
2020 IF A0$=" " THEN 1210
2025 WAIT 100
2030 GOTO 2000
3000 REM SET CLOCK
3030 P1=3.14159 : C=P1/30
3040 HIRES : CURSET 119 , 96 , 3 : CIRCLE 90 , 1
3050 FOR I=1 TO 12
3060 Q=I/6*PI
3070 CURSET 118+80*SIN(Q) , 92-80*COS(Q) , 3
3080 IF I<10 THEN CHAR 48+I , 0 , 1 : GOTO 3120
3090 IF I=10 THEN CHAR 49 , 0 , 1 : CURMOV 6 , 0 , 3 :
     CHAR 48 , 0 , 1 : GOTO 3120
3100 IF I=11 THEN CHAR 49 , 0 , 1 : CURMOV 6 , 0 , 3 :
     CHAR 49 , 0 , 1 : GOTO 3120
3110 CURMOV-5 , 0 , 3 : CHAR 49 , 0 , 1 : CURMOV 6 ,
     0 , 3 : CHAR 50 , 0 , 1
3120 NEXT I
4000 INPUT"HOUR.";HT
4010 IF HT<0 OR HT>24 THEN 4000
4020 INPUT"MINUTES";MI
4030 IF  MI<0 OR MI>60 THEN 4020
4040 INPUT"SECONDS";SI
```

```
4050 IF SI<0 OR SI>60 THEN 4040
4060 IF HT>12 THEN HT=HT-12 ELSE HI=HT
4070 HI=HI*5+INT(MI/12)
4080 CLS : PRINT"DO YOU WANT TO SET THE ALARM." :
     PRINT"(Y OR N)"
4090 INPUT A$
4100 IF A$<>"Y" AND A$<>"N" THEN 4090
4110 IF A$="N" THEN AL=0 : GOTO 4190
4120 AL=1 : CLS
4130 INPUT"ALARM HOUR.";AH
4140 IF AH<0 OR AH>24 THEN 4130
4150 INPUT"ALARM MINUTES.";AM
4160 IF AM<0 OR AM>60 THEN 4150
4170 INPUT"ALARM SECONDS.";AS
4180 IF AS<0 OR AS>60 THEN 4170
4190 REM DISPLAY TIME
4200 PRINT : PRINT
4210 PRNT"ORICLOCK."
4220 GOTO 1000
4230 END
```