Computers do not always have to perform difficult tasks to be useful. Very often it is the boring, repetitive, soul-destroying type of work we make them carry out. Calculating the hexadecimal values of the registers in the 6845 (or 6545) cathode ray tube controller (CRTC) for any given screen format could hardly be called mind-taxing but it is the sort of job that any computer, using this BASIC program, will perform correctly and as often as you like.

programming the 6845

a BASIC description of the CRTC registers

P. Fransen

The value of changing the screen format on your Elektor VDU card (or any other VDU card that uses a 6845 or 6545 CRTC) may not be immediately obvious but once hooked on the technique it is something you are likely to do more and more often. Furthermore this program is interesting and instructive in its own right.

The parameters

The 6845, and all the various details about structure, organisation of the screen format and the signals used, have already been dealt with in Elektor and in other books so we will not bother about that here. Any information required can be found in the literature listed at the end of this article.

The video norms currently in force in Europe use a line frequency of 15625 Hz and an frame frequency of 50 Hz. The time needed to sweep one line on the screen is

1/15625 s = 64 μs, and the time to sweep a complete frame is

1/50 s = 20 ms.

We must now calculate the clock frequency required by the system.

Line synchronisation

Each character is based on a horizontal width of eight screen dots, each of which is scanned in one clock period. Knowing the number of horizontal characters now enables the clock frequency (which we will call f_x) to be calculated. The dot frequency is $1/f_x$ and the character frequency is eight times this value. With a total of 128 horizontal characters the clock frequency is:

$$\frac{128 \times 8}{64} = 16 \text{ MHz.}$$

1

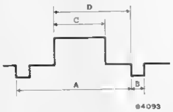

Figure 1. This diagram indicates the relationship between the signals generated by the CRTC and the parameters defined by the user. If period A is the line sync pulse duration, B is the width of that pulse, C is the width of the horizontal display and D defines the horizontal position of the image window. If, on the other hand A, is the frame pulse period, B, C and D are the corresponding vertical parameters.

This is no coincidence, actually, as the figure of 128 characters is chosen because it allows the common, inexpensive 16 MHz crystal to be used.

Working out the character duration gives us:

$$\frac{8 \times 1}{16 \text{ MHz}} = 0.5\ \mu\text{s}.$$

The total number of horizontal characters (minus one) between two horizontal sync pulses forms the contents of register R0. In this example we get:

128 − 1 = 127

or $7F_{HEX}$.

The contents of register R1 indicates the number of characters per line which in most cases will be 80, or 50_{HEX}.

The position of the horizontal sync pulse is determined by the contents of register R2 (see figure 1). This is calculated as follows:

HP = ((TSL − DT − 1.6 × LPB)/2) + DTZ

where DT = the width of the usable window (in μs)

TSL = the line time (in μs)

LPB = breadth of the line sync pulse (in μs), and

HP = the position of the line sync pulse (in μs).

The value of DT is:

80 × 0.5 = 40 μs.

The value of LPB (see R3) is

8 × 0.5 = 4 μs.

Inserting these values into the formula, we get

HP = ((64 − 40 − 1.6 × 4)/2) + 40 = 49 μs.

The factor 1.6 is an optional character to permit the position of the window on the screen to be accurately set.

Register R2 will contain

49/0.5 = 98

which is represented by 62_{HEX}.

Image synchronisation

In order to calculate the image synchronisation the number of screen lines per character must be known. The minimum number is eight, and this is generally used both for text and graphics characters. As the maximum number of character lines is 25, nine screen lines per character line are generally chosen. This gives 24 lines of characters on the screen. Each line then has a duration of

9 × TSL = 9 × 64 = 576 μs,

and sweeping the whole 24 lines takes
$24 \times 576 = 13{,}824\ \mu s$.
This time is generally indicated by VT.
The contents of register 6 will be 24, or 18_{HEX}.
The frame time must be as close as possible to 20 ms. With the line time calculated above we see that
$20{,}000/676 = 34.72$ lines.
Rounded off, this gives 34 lines (24 of which are usable) between successive frame sync pulses. From this we obtain the contents of R4: 34, or 21_{HEX}. As the frame time is only
$34 \times 576 = 19{,}584\ \mu s$
there are still
$20{,}000 - 19{,}584\ \mu s$
needed. A number of extra lines must be swept to bring the total screen time up to 20 ms. The actual number is calculated by dividing the remainder by the line time:
$416/64 = 6.5$
so this is rounded to 6, giving a value of 06_{HEX}.
Calculating the position of the frame sync pulse is similar to that for the line sync:
$VP = VTT - (VT + 1{,}500)/2\ VT$
where VTT is the frame time. In our example:
$34 \times 576 + 6 \times 64 = 19{,}968\ \mu s$.
The contents of R7 can be calculated from VP:
$(19{,}968 - (1500 + 24 \times 576))/2 + 24 \times 576 = 16{,}146\ \mu s$.

Table 1.

```
100 REM *** CONSTANTS ***
105 DIM R(15)
110 R(3)=8
120 K$="REGISTER"
130 L$="MICROSECONDS"
150 REM ********** R0 **********
160 PRINT "HORIZONTAL LINE LENGTH (CHAR.): "
170 INPUT A0
180 R(0)=A0-1
190 TC=64/A0
200 FX=8/TC
210 PRINT "FREQUENCY = ";FX;" MHZ"
220 PRINT "CRYSTAL FREQUENCY (MHZ): "
230 INPUT FX
240 TC=1/(FX/8)
250 LPB=R(3)*TC
260 TSL=A0*TC
300 REM ********** R1 **********
310 PRINT "NUMBER OF CHARACTERS PER LINE: "
320 INPUT R(1)
330 DT=R(1)*TC
400 REM ********** R2 **********
410 HP=DT+(TSL-1.5*LPB-DT)/2
420 R(2)=HP/TC
500 REM ********** R3 **********
600 REM ********** R4 **********
610 PRINT "NUMBER OF SCAN LINES: "
620 INPUT A
623 IF A<8 THEN PRINT "MINIMUM 8 SCAN LINES !":GOTO 610
625 PRINT "NUMBER OF CHARACTER LINES: "
630 INPUT B
640 TR=(A)*TSL
650 VT=(B+1)*TR
660 IF VT<=20000 THEN 680
665 PRINT
670 PRINT " IMPOSSIBLE! "
675 PRINT "FEWER CHARACTER OR SCAN LINES, PLEASE. "
677 GOTO 600
680 Y=INT(20000/TR)
690 R(4)=Y-1
700 REM ********** R5 **********
710 R(5)=INT((20000-Y*TR)/TSL)
800 REM ********** R6 **********
810 R(6)=B
815 VD=R(6)*TR
900 REM ********** R7 **********
910 R(7)=INT((((TR*Y+TSL*R(5))-(1500+B*TR))/2+B*TR)/TR)
915 VP=R(7)*TR
1000 REM ********** R8 **********
1010 R(8)=0
1100 REM ********** R9 **********
1110 R(9)=A-1
1200 REM ********** R10 & R11 **********
1202 REM UNDERLINE CURSOR
1204 IF A=8 THEN R(11)=A :R(10)=64+A :GOTO 1300
1206 R(10)=73 :R(11)=9
1300 REM ********** R12, R13, R14 & R15 **********
1310 R(12)=0
1320 R(13)=0
1330 R(14)=0
1340 R(15)=0
1350 PRINT :PRINT
1352 PRINT "SCREEN FORMAT = ";R(1);" X ";B
1354 PRINT: PRINT
1700 FOR Q=0 TO 15
1710 PRINT K$;" R";Q;
1720 PRINT TAB(20);" = ";
1727 Z2=R(Q)
1730 GOSUB 2000
1740 PRINT
1750 NEXT Q
1760 PRINT :PRINT:
1800 PRINT " CLOCK PERIOD                ";TC;L$
1810 PRINT " LINE SYNC. PULSE WIDTH      ";LPB;L$
1815 PRINT " LINE SYNC. PULSE PERIOD     ";TSL;L$
1830 PRINT " HORIZONTAL DISPLAY TIME     ";DT;L$
1840 PRINT " HORIZONTAL POSITION         ";HP;L$
1850 PRINT " CHARACTER LINE PERIOD       ";TR;L$
1855 VE=Y*TR+R(5)*TSL
1860 PRINT " RASTER SYNC. PERIOD         ";VE;L$
1865 PRINT " VERTICAL DISPLAY TIME       ";VD;L$
1867 PRINT " VERTICAL POSITION           ";VP;L$
1990 END
2000 REM ********** DEC TO HEX **********
2010 PRINT "$";
2020 FOR Z=1 TO 0 STEP -1
2030 Z1=INT(Z2/16^Z)
2040 Z2=Z2-Z1*16^Z
2050 Z1=Z1+48
2060 IF Z1>57 THEN Z1=Z1+7
2070 PRINT CHR$(Z1);
2080 NEXT Z:RETURN
```

programming the 6845

Table 1. Using this short BASIC program it is a very simple matter to calculate the appropriate hexadecimal addresses to insert into the 6845 registers for any given screen format.

This value is divided by the line time
16,146/576 = 28.03
giving 28 when rounded, or $1C_{HEX}$. Register 8 will almost invariably contain zero as we do not want to have an interlaced frame. The contents of register 9 is simply the number of screen lines per character line.

Table 2.

```
RUN
HORIZONTAL LINE LENGTH (CHAR.):
? 128

FREQUENCY =  16  MHZ

CRYSTAL FREQUENCY (MHZ):
? 16

NUMBER OF CHARACTERS PER LINE:
? 80

NUMBER OF SCAN LINES:
? 9

NUMBER OF CHARACTER LINES:
? 24


SCREEN FORMAT =  80  X  24


REGISTER R 0        = $7F
REGISTER R 1        = $50
REGISTER R 2        = $62
REGISTER R 3        = $08
REGISTER R 4        = $21
REGISTER R 5        = $06
REGISTER R 6        = $18
REGISTER R 7        = $1C
REGISTER R 8        = $00
REGISTER R 9        = $08
REGISTER R 10       = $49
REGISTER R 11       = $09
REGISTER R 12       = $00
REGISTER R 13       = $00
REGISTER R 14       = $00
REGISTER R 15       = $00


 CLOCK PERIOD                .5 MICROSECONDS

 LINE SYNC. PULSE WIDTH      4 MICROSECONDS

 LINE SYNC. PULSE PERIOD     64 MICROSECONDS

 HORIZONTAL DISPLAY TIME     40 MICROSECONDS

 HORIZONTAL POSITION         49 MICROSECONDS

 CHARACTER LINE PERIOD       576 MICROSECONDS

 RASTER SYNC. PERIOD         19968 MICROSECONDS

 VERTICAL DISPLAY TIME       13824 MICROSECONDS

 VERTICAL POSITION           16128 MICROSECONDS

OK
```

Table 2. When the four user-defined parameters have been loaded the contents of the CRTC registers are output in this way.

The cursor

The program dealt with in this article does not permit a very flexible programming of the cursor. This can be improved by including a few BASIC lines to add a choice of options as we will now see. Registers 10 and 11 define the upper and lower limits (the size, in other words) of the cursor respectively. Bits 5 and 6 of register 10 determine whether the cursor is present at all and if so whether it flashes or simply lights. As an example, assume we want a non-flashing cursor which has the form of a single underline. The register 10 configuration needed is given by the value 48_{HEX} (more details of this are given in Paperware 3). As the lower limit of the cursor will be the last line swept (for any given character line), register 11 must contain 08_{HEX}. Unlike what we have dealt with up to now, registers 12...17 do not lend themselves to individual calculations so we will have to be content simply to initialise them.

A few examples

Programming the 6845 is made easier in any system with the aid of the program shown in table 1. Given four parameters (the number of characters between two line sync pulses [horizontal total], which gives the ideal crystal frequency that should be used, the number of characters used per line, the number of screen lines per character line and the number of character lines on the screen) it returns the hexadecimal contents of all the 6845 registers concerned. An example of this result is shown in table 2. All the parameters can also be stated in decimal base.
Having let the program work out all these results the next question is what to do with them. If you are not using the Elektor VDU card and its software you will have to study your system's software to find out how to access the 6845 initialisation routine. In the Elektor system (detailed in Paperware 3) this initialisation procedure carries out two operations: one (routine MOVCRT) to change the look-up table containing the RAM and ROM parameters (CRT timing table) and the other to transfer the RAM parameters to the CRTC (routine CRTINT). This latter routine is the one we are interested in. Before starting it (by means of DISK!GO F36C, for example) the data calculated by the BASIC program of table 1 must be saved from address $EFDC_{HEX}$ (61404 decimal) onwards. As is often the case, changing the screen format demands a total erasure so execute the RESET routine ($F33Ø_{HEX}$) immediately and this simply calls the CRTINT routine needed to program the CRTC.

References:
Elektor Paperware 3 and 4
Motorola 8-bit Micropressors Manual
Synertek Data Book