MICRO-BUS

Compiled by DJD.

Appearing every two months, Micro-Bus presents ideas, applications, and programs for the most popular microprocessors; ones that you are unlikely to find in the manufacturers' data books. The most original ideas often come from readers working on their own systems, and payment will be made for any contribution featured.

THIS month's Micro-Bus examines the Acorn Teletext VDU, an unusual memory-mapped VDU with the ability to display Prestel and Teletext pictures. The VDU card can be used with the low-cost System One microcomputer, and several programs are presented to demonstrate the use of the VDU with the System One for graphics.

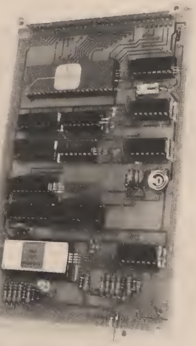

Fig. 1. The Acorn Teletext VDU card

BLOCK DIAGRAM

A block diagram of the Acorn Teletext VDU is shown in Fig. 2. It consists of three main parts: the VDU memory, the controller chip, and the teletext character generator. The VDU displays the contents of the 1K of memory, which is switched between the VDU and the processor for half of each memory cycle; in other words, access to the VDU is transparent so that the processor can read from or write to the screen memory without affecting the display. The VDU is normally configured for a format of 25 lines of 40 characters, thus using 1000 bytes of the 1024 RAM locations. This format includes an extra line over the teletext format, which is 24 × 40. The memory is normally addressed as locations #0400 to #07FF.

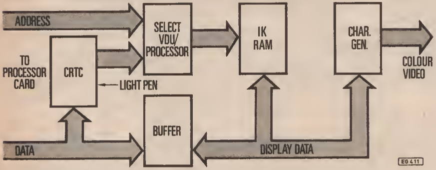

Fig. 2. Block diagram of the teletext VDU card; it interfaces to the computer's address and data lines

SCREEN FORMAT

The format of characters on the screen is controlled by the 6845 CRT controller chip, and the format can, to a certain extent, be reprogrammed simply by altering constants stored in the 6845's registers. The 6845 is addressed as two memory locations; an address register and a data register. In fact the 6845 contains a total of 18 registers; the number loaded into the address register determines which of the other registers is selected as the data register. A drawback of using the CRTC is that it must be programmed before any display will be obtained. The routine of Fig. 3 illustrates how to do this using the 6502 micro. The routine writes the sixteen values from a table after the program to registers #00 to #0F respectively, and sets the display up in 25 × 40 format with a flashing-underline cursor.

```
                    ;       SET UP CRTC REGISTERS
                    ;       ON TELETEXT VDU.
                    ;
                    CRTA   =$0800               ADDRESS
                    CRTB   =$0801               DATA
                    RESTRT =$FF04               IN MONITOR
                    ;
0000                       .=$0980
0980  A0 0F         SETCRT LDY   #$0F           NO. OF REGS
0982  8C 00 08      LOOPS  STY   CRTA           SELECT REG
0985  B9 91 09             LDA   CRTTAB,Y       GET VALUE
0988  8D 01 08             STA   CRTB
098B  88                   DEY
098C  10 F4                BPL   LOOPS
098E  4C 04 FF             JMP   RESTRT         BACK.
                    ;
0991  3F            CRTTAB .BYTE $3F,$28,$33,$05
0995  1E                   .BYTE $1E,$02,$19,$1B
0999  00                   .BYTE $00,$09,$68,$09
099D  04                   .BYTE $04,$00,$07,$FF
                           .END
```

Fig. 3. Program for the 6502 micro sets up the VDU card for a 25 × 40 display, and a flashing-underline cursor

Registers #00 to #09 determine the screen format. For example, to obtain a format of 16 rows of 32 characters set register #01 (number of characters per line) to #20, and register #06 (number of lines) to #10. The display can be turned off completely, without affecting the contents of the display memory, by setting register #06 to #00.

Registers #0C and #0D contain the high and low bytes respectively of the screen start address. The start address can be changed to alter the mapping of memory to the screen; for example, incrementing it by 40, the number of locations per line, will scroll the display.

The CRTC also provides a cursor, which can be flashing or static, and whose shape can be programmed. The type of cursor is determined by registers #0A and #0B. For the normal flashing underline cursor register #0A has the value #68; or cursors can be obtained by altering this register as follows:

Function:	Value:
Cursor off	#1F
Static block	#00
Slow flashing block	#60
Fast flashing block	#40

The position of the cursor on the screen is determined by registers #0E and #0F, which contain the high and low bytes of the cursor address.

The CRTC also provides a light-pen strobe input; a logic-level transition on this input will cause the address of the cell currently being displayed to be stored in the light-pen address registers, #10 and #11.

TELETEXT CONTROLLER

The character generation and display is performed by the SAA5050 Teletext Character Generator, which in addition to being able to display the usual and lower-case character set, has many attractive features such as: double-height characters; the ability to flash characters; display of characters in six colours or white against a background of any colour, black, or white; and display of two different types of graphics. Furthermore, most of these options can be used in combination.

Most memory-mapped VDUs use spare bits in the display memory to select special options for each character; for example, the top bit of the character could be used to determine whether the character is displayed static or flashing. In the teletext VDU there are too many different options to enable them to be specified in this way for each character, so a rather cunning method is used instead. Certain control codes, when present in a line, change the state of subsequent characters on that VDU line. Thus a line will normally be steady, but if a 'flash' code is put on the VDU, all the remaining characters on that line will flash. It is even possible to make one word in a line flash by preceding the word with a 'flash' code, and following it by a 'steady' code.

The Teletext method of selecting special functions has one drawback: the control code will appear on the VDU as a blank cell, corresponding to the background colour; so, for example, it is not possible to display a line of contiguous characters all in different colours; there has to be a blank between each character.

GRAPHICS

To provide the facility for plotting graphs and histograms the character cell is divided into six picture elements, or "pixels", each of which can be either set to a colour, or cleared to the background colour. The state of each cell is determined by one bit of the code, as shown in Fig. 4. For the graphics symbols, bit 5 is always set. To display codes as graphics, rather than characters, they are preceded by a code specifying graphics and the required colour.

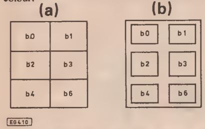

Fig. 4. Diagrams showing how the teletext graphics symbols are constructed for: (a) contiguous graphics, and (b) separated graphics

The usual graphics mode is called 'contiguous graphics', but there is also an option called 'separated graphics' in which the pixels are separated by a thin border; see Fig. 4 (b).

If the whole VDU screen is to be used for graphics the first character of each line should be a 'graphics colour' code; the overall resolution is thus 78 × 75. A routine to clear the display and set it up for white graphics is shown in Fig. 5.

```
                    ;       CLEAR SCREEN FDR GRAPHICS
                    ;       TELETEXT VDU.
                    ;
                    ADDS    =$0023          CELL POINTER
                    RESTRT  =$FF04          IN MONITOR
                    ;
0000                        .=$0E80
0E80  A0 00         CLEARS  LDY   £D
0E82  A9 20                 LDA   £$20      SPACE
0E84  99 00 04      CLEAR   STA   $400,Y    00 256 BYTE
DE87  99 00 D5              STA   $500,Y      BLOCXS.
DE8A  99 00 06              STA   $600,Y
DE8D  99 00 07              STA   $700,Y
DE9D  C8                    INY
0E91  D0 F1                 BNE   CLEAR
DE93  84 23                 STY   ADDS      ZERO
DE95  A9 D4                 LDA   £$04      SET ADDS TO
DE97  85 24                 STA   ADDS+1      SCREEN START
0E99  D8                    CLD
0E9A  A2 18                 LDX   £$18      25 LINES
0E9C  A9 17         SET     LDA   £$17      GRAPHICS WHITE
0E9E  91 23                 STA   (ADDS),Y  START OF LINE
0EA0  A9 28                 LDA   £$28      40 CHARS/LINE
0EA2  18                    CLC
0EA3  65 23                 ADC   ADDS
0EA5  85 23                 STA   ADDS
0EA7  90 02                 BCC   NOCY      CARRY?
0EA9  E6 24                 INC   ADDS+1
0EAB  CA            NOCY    DEX
0EAC  10 EE                 BPL   SET       00 ALL 25 LINES
0EAE  4C 04 FF              JMP   RESTRT    STOP.
                    ;
                            .END
```

Fig. 5. Routine to clear the display, and initialise the display memory for graphics

EXAMPLES

Some examples of displays generated by the Teletext VDU are shown in Fig. 6. The top line demonstrates the use of colour. The default colour at the start of each line is white; to display the word CYAN in the colour cyan it is preceded by an "alpha cyan" code, and it is followed by an "alpha white" code to return to white characters.

The second line illustrates the use of a different background colour. The background colour can be set to the current colour with the "new background" code; since the default colour at the start of a line is white, a "new background" code at the start of a line will give a white background. It is followed by an "alpha blue" code to cause the subsequent characters to appear in blue, which is dark against the white background. The background is returned to black after the text with a "black background" code.

Fig. 6. Examples of the types of display possible with the teletext VDU

The double-height characters are obtained by putting the same characters on two successive lines, preceded in each case by "double height"codes. The VDU automatically displays the top halves of the letters on the first line and the lower halves on the second line.

The first graphics example shows two lines of contiguous graphics, preceded by "graphics white" and "graphics cyan" codes respectively. In the second graphics example the lines are the same, but are also preceded by "separated graphics" codes.

GRAPH PLOTTING

The following two programs illustrate how the Acorn Teletext VDU can be used for simple graph-plotting. The routines were devised by *Peter Mayne* of London, who uses the VDU with an Acorn System One microcomputer.

```
                    ;       PLOT POINT AT (XC,YC)
                    ;       ON TELETEXT VDU.
                    ;
                    XC      =$0020          X COORDINATE
                    YC      =$0021          Y COORDINATE
                    ADDS    =$0023          CELL ADDRESS
                    ;
0000                        *=$D200
0200  D8            PLOT    CLD
0201  E6 21                 INC   YC        CALCULATE WHICH
0203  A9 D7                 LDA   £$07        LINE...
D205  85 24                 STA   ADDS+1
D207  A9 C1                 LDA   £$C1
D209  85 23                 STA   ADDS      BOTTOM LEFT
020B  A4 21                 LDY   YC
D20D  A2 02         YCALC   LDX   £2        DIVIDE BY 3
020F  86 21         LOOP    STX   YC
0211  88                    DEY
0212  F0 10                 BEQ   XCALC     OONE Y?
0214  CA                    DEX
0215  10 F8                 BPL   LOOP
0217  A5 23                 LDA   ADDS      MOVE UP BY
0219  38                    SEC               DNE LINE.
021A  E9 28                 SBC   £40       CHARS/LINE
021C  85 23                 STA   ADDS
021E  B0 ED                 BCS   YCALC
0220  C6 24                 DEC   ADDS+1    CARRY
0222  90 E9                 BCC   YCALC     I.E. ALWAYS
0224  18            XCALC   CLC
0225  66 20                 ROR   XC        DIVIDE BY 2
0227  90 2F                 BCC   EVEN
0229  A5 21                 LDA   YC        Y REMAINDER
022B  00 04                 BNE   NZERA
022D  A9 02                 LDA   £$02      PIXEL MASX
D22F  00 0A                 BNE   CONT      I.E. ALWAYS
D231  C9 01         NZERA   CMP   £$01
D233  00 04                 BNE   NONEA
0235  A9 08                 LDA   £$08      PIXEL MASX
0237  00 02                 BNE   CONT      I.E. ALWAYS
0239  A9 40         NONEA   LDA   £$40      PIXEL MASX
023B  85 21         CONT    STA   YC        PLOT PDINT..
023D  A5 23                 LDA   ADDS      GET ADDRESS
D23F  18                    CLC               OF LINE.
D240  65 20                 ADC   XC        ADD COLUMN
0242  85 23                 STA   ADDS        NUMBER.
D244  90 02                 BCC   OONE
0246  E6 24                 INC   ADDS+1    CARRY
0248  A0 00         OONE    LDY   £D
024A  A5 21                 LDA   YC        GET MASK
024C  31 23                 AND   (ADDS),Y
024E  00 07                 BNE   RETN
0250  B1 23                 LDA   (ADDS),Y
0252  18                    CLC
D253  65 21                 ADC   YC        ADD IN MASK
0255  91 23                 STA   (ADDS),Y
0257  60            RETN    RTS             RETURN.
D258  A5 21         EVEN    LDA   YC        NO REMAINDER
025A  D0 04                 BNE   NZERB
D25C  A9 01                 LDA   £$01      PIXEL MASK
025E  00 DB                 BNE   CONT      I.E. ALWAYS
D26D  C9 01         NZERB   CMP   £$01
0262  00 04                 BNE   NONEB
0264  A9 04                 LDA   £$04      PIXEL MASK
0266  00 D3                 BNE   CONT      I.E. ALWAYS
0268  A9 10         NONEB   LDA   £$10      PIXEL MASX
026A  00 CF                 BNE   CONT      I.E. ALWAYS
                    ;
                            .END
```

Fig. 7. Routine for the 6502 micro to plot a point at specified coordinates on the teletext VDU

The PLOT routine, in Fig. 7, plots a point, or pixel, on the display at the coordinates specified in the locations XC and YC. The coordinates (0,0) correspond to the bottom left-hand corner of the display, and (77,74) to the top right. Note that XC and YC are modified by the PLOT routine.

The program works by finding, in ADDS, the address of the location that contains the required pixel, and determines a number to add to that location to set the required pixel. Since there are three pixels per cell in the Y direction, the routine must divide the value of YC by three to find which line contains the required location.

The GRAPH program, Fig. 8, demonstrates how the PLOT routine can be used to plot a graph of an equation. As it stands, the program plots $Y = \frac{1}{2}X + 16$ for values of X from 0 to 63. The program also draws axes, and finally labels the graph with the equation; see Fig. 9. A delay is included to slow down the plotting.

```
                   ;       PLOT A GRAPH
                   ;
                   XC      =$0020             X COORDINATE
                   YC      =$0021             Y COORDINATE
                   XVAL    =$0040
                   PLOT    =$0200             PLOT POINT
                   RESTRT  =$FF04             IN MONITOR
                   WAIT    =$FECO             DELAY
                   ;
0000                        .=$0300
0300  A2 00        GRAPH   LDX   £0           FOR EACH
0302  86 20        DRAW    STX   XC             VALUE OF X..
0304  A9 00                LDA   £0
0306  85 21                STA   YC
0308  86 40                STX   XVAL
030A  20 00 02             JSR   PLOT         X AXIS
030D  A6 40                LDX   XVAL
030F  86 21                STX   YC
0311  A9 00                LOA   £0
0313  85 20                STA   XC
0315  86 40                STX   XVAL
0317  20 00 02             JSR   PLOT         Y AXIS
031A  A6 40                LDX   XVAL
031C  86 20                STX   XC
031E  8A                   TXA                GET X VALUE
031F  4A                   LSR   A            DIVIDE BY 2
0320  18                   CLC
0321  69 10                ADC   £16          ADO 16
0323  85 21                STA   YC           GIVES Y
0325  86 40                STX   XVAL
0327  20 00 02             JSR   PLOT         PLOT POINT
032A  A6 40                LOX   XVAL
032C  A0 20                LDY   £$20         DELAY
032E  20 CD FE     SLOW    JSR   WAIT
0331  88                   DEY
0332  10 FA                BPL   SLOW
0334  E8                   INX                NEXT X..
0335  E0 40                CPX   £$40         ALL OONE?
0337  D0 C9                BNE   DRAW
0339  A2 58        AXES    LDX   £'X          LABEL AXES
033B  8E 01 04             STX   $401
033E  E8                   INX                GET 'Y
033F  8E E3 07             STX   $7E3
0342  A2 07                LDX   £7           PUT TITLE
0344  BD 50 03     TITLE   LDA   CHARS,X
0347  9D 21 06             STA   $621,X
034A  CA                   OEX
034B  10 F7                BPL   TITLE
034D  4C 04 FF             JMP   RESTRT       STOP.
                   ;
0350  07           CHARS   .BYTE $7           ALPHA WHITE
0351  59 30 5C             .BYTE 'Y=½'
0354  58 2B 31             .BYTE 'X+1'
0357  36                   .BYTE '6'
                           .ENO
```

Fig. 8. Program for the 6502 micro to plot a graph on the teletext VDU

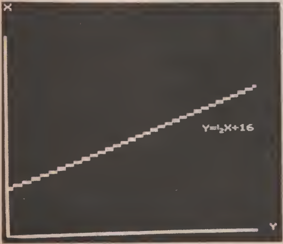

Fig. 9. Graph produced by the program of Fig. 8

Before running the program, the routines of Fig. 3 and Fig. 5 should be loaded and executed at #0980 and #0E80 respectively to initialise the CRTC and clear the screen for graphics.

NINE PROBLEMS REVISITED

One of the problems posed in the April Micro-Bus asked for a way to reverse the bits in a byte, on the 6800 micro, in under 10 cycles. A solution in hardware was proposed, but *John Diamond* of Coventry has solved the problem using only software by providing a look-up table in which each byte gives the reverse of that byte's position in the table. The six-byte routine to reverse a byte, shown in Fig. 10, then takes only 9 cycles. It uses self-modifying code to avoid having to use the index register, thus saving several cycles. A further cycle can be saved by using zero-page memory.

The listing of Fig. 10 also includes an initialisation routine to generate the 256-byte look-up table. Other functions could be implemented simply by providing a different table.

Another of the problems was to write a programme to find the highest prime factor of a number using a rudimentary machine-code called MINIL. The solution was given without explanation, and readers were asked to provide one. *J. Rennie* of Somerset wrote:

"Your challenge to provide an explanation of the MINIL program . . . proved to be quite irresistable, and I found the task most interesting. The program is based on a fairly simple algorithm (see flowchart of Fig. 11). Let N denote the number whose highest prime factor is to be found. Thus $N = (P_o \times P_1 \times \ldots \times P_n)$, where P_o to P_n are the n prime factors of N. The algorithm finds the highest factor of N, working down from N – 1, which is $(P_o \times P_1 \times \ldots \times P_{n-1})$. This new number is then denoted N, and the process is repeated to give the new factor $(P_o \times P_1 \times \ldots \times P_{n-2})$. This is repeated until only P_o is left. No more factors can be found and P_o, the highest prime factor of the original number, can be output.

"In the original MINIL program registers D and A hold the current values of N and X respectively, while steps 4–10 find the factors of N. The two loops NOT and NEW are identical to loops 1 and 2 of the flowchart."

A similar explanation of the program was received from *Doug Letts* of London. Readers interested in the MINIL language may like to try writing programs which find the greatest common divisor of two numbers, the integer part of a number's square root, and the factorial of a number. But be warned: one of these problems is impossible, and the other two are not simple!

```
                   * REVERSE ACC. A INTO ACC. B
                   *
E0E3               CONTRL  EQU      £E0E3
0200               TABLE   EQU      £0200
                   * MAIN PROGRAM
0100                       ORG      £0100
0100  B7 01 05     START   STA  A   START+5  5  CYCLES
0103  F6 02 00             LDA  B   TABLE    4  CYCLES
                   * END OF REVERSING ROUTINE
                   *
                   * INITIALISATION ROUTINE
                   *
0180                       ORG      £0180
0180  CE 02 00             LDX      £TABLE
0183  4F                   CLR  A
0184  36           LOOP    PSH  A
0185  C6 80                LDA  B   ££80        THESE SIX BYTES
0187  48           REVRSE  ASL  A               REVERSE A INTO B
0188  56                   ROR  B               IN 66 CYCLES
0189  24 FC                BCC      REVRSE
018B  E7 00                STA  B   .0, X
018D  32                   PUL  A
018E  08                   INX
018F  4C                   INC  A
0190  26 F2                BNE      LOOP
0192  7E E0 E3             JMP      CONTRL
                           END
```

Fig. 10. Routine to reverse the bits in a byte on the 6800 micro, together with a program to generate the table used by the routine

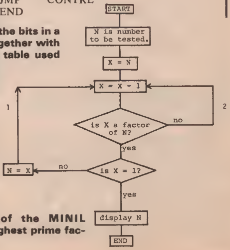

Fig. 11. Flowchart of the MINIL program to find the highest prime factor of a number