

## Chapter 6 - TED Video

---

The TED chip reproduces the picture model of the C16 and Plus/4. It sits on compositor layer 12, between VGA (layer 10) and ANTIC (layer 13). TED's design folds video and audio into one chip; this chapter covers the video half. The audio half is described in Chapter 16.

### 6.1 What TED can show

| Item           | Value   |
|----------------|---|
| Display area   | 320 × 200 pixels  |
| Character grid | 40 × 25 cells of 8 × 8                                      |
| Border         | 32 left, 32 right, 35 top, 37 bottom                        |
| Total frame    | 384 × 272 pixels  |
| Colours        | 121 unique (16 hues × 8 luminances minus duplicated blacks) |
| Modes          | Text, bitmap, multicolour bitmap, extended-colour text      |

In multicolour mode each pixel is two screen pixels wide, so the effective resolution drops to 160 × 200 with four colours per cell instead of two.

### 6.2 BASIC keywords

The BASIC keyword TED introduces every TED subcommand. The recognised forms are:

| Form                          | Effect   |
|-------------------------------|--|
| TED ON                        | Enable the chip (sets TED_V_ENABLE bit 0).   |
| TED OFF                       | Disable the chip.  |
| TED MODE $n$                  | 0 = text, 1 = bitmap, 2 = multicolour bitmap. Sets the BMM and MCM bits and forces DEN on. |
| TED COLOR $bg$ [ , $border$ ] | Set background colour 0 and (optionally) the border colour.                                |
| TED CHAR $addr$               | Set the character/bitmap base register.  |
| TED VIDEO $addr$              | Set the video matrix base register.  |
| TED CLS                       | Fill the selected video matrix with the space character (\$20).                            |
| TED SCROLL $dx, dy$           | Write the X and Y fine-scroll fields of CTRL2 and CTRL1 (each 0-7).                        |

For anything beyond the simplest setup, use POKE and POKE8 on the registers and TED VRAM. This is the normal TED programming style, because the useful picture data is just bytes in the matrix, colour RAM, character set, and bitmap area.

This first program enables TED text mode and writes a coloured banner directly into the video matrix:

```

10 TED ON
20 POKE8 &H000F0F20, &H18           : REM DEN + RSEL
30 POKE8 &H000F0F24, &H08           : REM CSEL
40 POKE8 &H000F0F30, &H06           : REM blue background
50 POKE8 &H000F0F40, &H5E           : REM bright dark-blue border
60 FOR I=0 TO 999
70 POKE8 &H000F3000+I, 32
80 POKE8 &H000F3400+I, &H71
90 NEXT I
100 T$="TED RAINBOW"
110 FOR I=1 TO LEN(T$)
120 A=&H000F3000+12*40+14+I
130 C=&H000F3400+12*40+14+I
140 POKE8 A, ASC(MID$(T$,I,1))
150 POKE8 C, &H10+I
160 NEXT I

```

You should see a TED text screen with a dark-blue border and a multi-coloured message near the centre.

## 6.3 The register block

TED's video registers live in the small block \$F0F20-\$F0F6B. The chip's audio registers sit at \$F0F00-\$F0F05 (Chapter 16).

| Address | Name                | Purpose   |
|---------|---------------------|---|
| \$F0F20 | TED_V_CTRL1         | Control 1 (ECM, BMM, DEN, RSEL, YSCROLL).   |
| \$F0F24 | TED_V_CTRL2         | Control 2 (RES, MCM, CSEL, XSCROLL).  |
| \$F0F28 | TED_V_CHAR_BASE     | Character generator base (high nibble) and bitmap base (low nibble), in 1 KB steps. |
| \$F0F2C | TED_V_VIDEO_BASE    | Video matrix base, in 1 KB steps (bits 3-7).  |
| \$F0F30 | TED_V_BG_COLOR0     | Background colour 0.  |
| \$F0F34 | TED_V_BG_COLOR1     | Background colour 1 (multicolour).  |
| \$F0F38 | TED_V_BG_COLOR2     | Background colour 2 (multicolour).  |
| \$F0F3C | TED_V_BG_COLOR3     | Background colour 3 (multicolour).  |
| \$F0F40 | TED_V_BORDER        | Border colour.  |
| \$F0F44 | TED_V_CURSOR_HI     | Cursor position, high byte.   |
| \$F0F48 | TED_V_CURSOR_LO     | Cursor position, low byte.  |
| \$F0F4C | TED_V_CURSOR_CLR    | Cursor colour.  |
| \$F0F50 | TED_V_RASTER_LO     | Current raster line, low byte (read-only).  |
| \$F0F54 | TED_V_RASTER_HI     | Raster bit 8 in bit 0 (read-only).  |
| \$F0F58 | TED_V_ENABLE        | Bit 0 = video enable.   |
| \$F0F5C | TED_V_STATUS        | Bit 0 = VBlank active (read-only).  |
| \$F0F60 | TED_V_RASTER_CMP_LO | Raster compare line, low byte.  |
| \$F0F64 | TED_V_RASTER_CMP_HI | Raster compare line bit 8 in bit 0.   |
| \$F0F68 | TED_V_RASTER_STATUS | Bit 7 = compare pending; write \$80 to clear.                                       |

Every register is a 32-bit word at a 4-byte-aligned address; only the low byte of each is meaningful.

### 6.3.1 CTRL1 bits

| Bit | Name    | Meaning                                      |
|-----|---------|--|
| 6   | ECM     | Extended colour mode.                        |
| 5   | BMM     | Bitmap mode.                                 |
| 4   | DEN     | Display enable. Must be set for any picture. |
| 3   | RSEL    | 0 = 24 rows, 1 = 25 rows.                    |
| 0-2 | YSCROLL | Vertical fine-scroll, 0-7 raster lines.      |

### 6.3.2 CTRL2 bits

| Bit | Name    | Meaning                             |
|-----|---------|-------------------------------------|
| 5   | RES     | Reset.                              |
| 4   | MCM     | Multicolour mode.                   |
| 3   | CSEL    | 0 = 38 columns, 1 = 40 columns.     |
| 0-2 | XSCROLL | Horizontal fine-scroll, 0-7 pixels. |

The picture mode is the combination of BMM, MCM, and ECM:

| BMM | MCM | ECM | Picture   |
|-----|-----|-----|---|
| 0   | 0   | 0   | Standard text.  |
| 0   | 1   | 0   | Multicolour text.   |
| 0   | 0   | 1   | Extended-colour text (8 distinct background colours from the top three bits of the character code). |
| 1   | 0   | 0   | Standard bitmap.  |
| 1   | 1   | 0   | Multicolour bitmap (160 × 200).   |

## 6.4 The colour byte

Every colour register and every byte of colour RAM holds an 8-bit **colour byte** with this layout:

```

bit 7 6 5 4 3 2 1 0
    0 L L L H H H H
      |-luminance| |---hue---|

```

L is the luminance (0-7); H is the hue (0-15). The hue table is:

| Hue | Name  | Hue | Name         |
|-----|-------|-----|--------------|
| 0   | Black | 8   | Orange       |
| 1   | White | 9   | Brown        |
| 2   | Red   | 10  | Yellow-green |
| 3   | Cyan  | 11  | Pink         |

| Hue | Name   | Hue | Name        |
|-----|--------|-----|-------------|
| 4   | Purple | 12  | Blue-green  |
| 5   | Green  | 13  | Light blue  |
| 6   | Blue   | 14  | Dark blue   |
| 7   | Yellow | 15  | Light green |

Hue 0 (black) is the same colour at every luminance, which is why TED has 121 unique colours rather than 128 ( $16 \times 8$ ).

## 6.5 The VRAM region

TED owns 16 KB of private VRAM beginning at \$F3000. This region holds three things, all of which can be relocated through the TED\_V\_VIDEO\_BASE and TED\_V\_CHAR\_BASE registers:

| Block         | Default offset (from \$F3000) | Size   |
|---------------|-------------------------------|--------|
| Video matrix  | \$0000                        | 1024 B |
| Colour RAM    | \$0400                        | 1024 B |
| Character set | \$0800                        | 2048 B |

The video matrix is a  $40 \times 25$  array of character codes. The colour RAM is a  $40 \times 25$  array of colour bytes, one per cell. The character set is 256 entries of 8 bytes each; bit 7 of every byte is the leftmost pixel.

TED\_V\_VIDEO\_BASE selects the video matrix base in 1 KB steps, through bits 3-7 of its low byte. TED\_V\_CHAR\_BASE packs two selectors into one byte: bits 4-7 choose the character set, and bits 0-3 choose the bitmap base when in bitmap mode. Both selectors are also in 1 KB steps. If a selected base would make the needed data run beyond the 16 KB TED VRAM window, the chip falls back to the default base for that data type.

## 6.6 Text mode data

Standard text mode uses:

- video matrix byte: character code
- colour RAM byte: foreground colour byte
- background colour: TED\_V\_BG\_COLOR0
- character bitmap: eight bytes per character, MSB first

For cell  $(x, y)$ , where  $x$  is 0-39 and  $y$  is 0-24:

```
matrix address = $F3000 + matrixBase + y * 40 + x
colour address = $F3000 + matrixBase + 1024 + y * 40 + x
glyph address  = $F3000 + charBase + charCode * 8 + row
```

When bit 7 of a colour-RAM byte is set, the glyph is drawn in reverse video: set pixels become background and clear pixels become foreground. The displayed colour still uses the low seven bits of the colour byte.

### 6.6.1 Extended-colour text

Extended-colour text sets ECM in TED\_V\_CTRL1. The top two bits of each character code select one of TED\_V\_BG\_COLOR0-3; the low six bits select the glyph. Colour RAM remains the foreground colour.

This program makes four wide background panels using the same blank glyph and four different background registers:

```
10 TED ON
20 POKE8 &H000F0F20, &H58      : REM ECM + DEN + RSEL
30 POKE8 &H000F0F24, &H08      : REM 40 columns
40 POKE8 &H000F0F30, &H72      : REM red
50 POKE8 &H000F0F34, &H75      : REM green
60 POKE8 &H000F0F38, &H76      : REM blue
70 POKE8 &H000F0F3C, &H77      : REM yellow
80 POKE8 &H000F0F40, &H71      : REM white border
90 FOR I=0 TO 999
100 K=1+64*INT((I MOD 40)/10)
110 POKE8 &H000F3000+I, K
120 POKE8 &H000F3400+I, &H71
130 NEXT I
```

The character code is 1, 65, 129, or 193. In extended-colour mode all four codes use glyph 1, while their top two bits select the four background colours.

## 6.6.2 Multicolour text

Multicolour text clears BMM and sets MCM. The character bitmap is read two bits at a time, so each source pixel is two screen pixels wide:

| Bits | Colour source           |
|------|-------------------------|
| 00   | TED_V_BG_COLOR0         |
| 01   | TED_V_BG_COLOR1         |
| 10   | TED_V_BG_COLOR2         |
| 11   | Colour RAM for the cell |

The same addressing as standard text mode is used. The trade-off is horizontal detail: an 8-pixel character row becomes four double-wide colour cells.

This program builds one four-colour glyph and fills the screen with it:

```
10 TED ON
20 POKE8 &H000F0F20, &H18      : REM DEN + RSEL
30 POKE8 &H000F0F24, &H18      : REM MCM + CSEL
40 POKE8 &H000F0F30, &H06      : REM background blue
50 POKE8 &H000F0F34, &H72      : REM multicolour 1 red
60 POKE8 &H000F0F38, &H75      : REM multicolour 2 green
70 POKE8 &H000F0F40, &H71
80 FOR R=0 TO 7
90 POKE8 &H000F3800+2*8+R,&H1B
100 NEXT R
110 FOR I=0 TO 999
120 POKE8 &H000F3000+I,2
130 POKE8 &H000F3400+I,&H77
140 NEXT I
```

The glyph byte \$1B is the bit pattern 00, 01, 10, 11, so each character cell repeats all four multicolour text sources.

## 6.7 Bitmap modes

Bitmap mode sets BMM in TED\_V\_CTRL1. The bitmap base comes from the low nibble of TED\_V\_CHAR\_BASE:

```
bitmapBase = (TED_V_CHAR_BASE AND 15) * 1024
```

The bitmap contains 200 rows. For pixel row  $y$ , cell column  $cellX$ , and row inside the cell  $row = y \text{ AND } 7$ , the byte address is:

```
$F3000 + bitmapBase + INT(y / 8) * 320 + row * 40 + cellX
```

That is 8000 bytes for a full 320 x 200 bitmap.

### 6.7.1 High-resolution bitmap

High-resolution bitmap mode clears MCM. Each bitmap bit is one screen pixel. The video matrix byte for the cell supplies the two colours:

| Bitmap bit | Colour source                  |
|------------|--------------------------------|
| 0          | low nibble of the matrix byte  |
| 1          | high nibble of the matrix byte |

This example selects a bitmap at TED VRAM offset \$2000 and draws yellow and blue hatch stripes:

```
10 TED ON
20 POKE8 &H000F0F20, &H38      : REM BMM + DEN + RSEL
30 POKE8 &H000F0F24, &H08      : REM hires, 40 columns
40 POKE8 &H000F0F28, &H28      : REM bitmap base $2000
50 POKE8 &H000F0F40, &H71
60 FOR I=0 TO 999
70 POKE8 &H000F3000+I, &H76      : REM bit 1 yellow, bit 0 blue
80 NEXT I
90 FOR Y=0 TO 199
100 FOR X=0 TO 39
110 A=&H000F5000+INT(Y/8)*320+(Y AND 7)*40+X
120 IF (Y AND 8)=0 THEN POKE8 A,&HAA ELSE POKE8 A,&H55
130 NEXT X
140 NEXT Y
```

### 6.7.2 Multicolour bitmap

Multicolour bitmap mode sets both BMM and MCM. The bitmap is read as two-bit pairs, and each pair draws a double-wide pixel:

| Bits | Colour source                  |
|------|--------------------------------|
| 00   | TED_V_BG_COLOR0                |
| 01   | high nibble of the matrix byte |
| 10   | low nibble of the matrix byte  |
| 11   | colour RAM byte for the cell   |

This example fills the screen with repeating four-colour vertical bars:

```

10 TED ON
20 POKE8 &H000F0F20, &H38           : REM BMM + DEN + RSEL
30 POKE8 &H000F0F24, &H18           : REM MCM + CSEL
40 POKE8 &H000F0F28, &H28           : REM bitmap base $2000
50 POKE8 &H000F0F30, &H06           : REM background blue
60 FOR I=0 TO 999
70 POKE8 &H000F3000+I, &H75         : REM green and blue nibbles
80 POKE8 &H000F3400+I, &H72         : REM bright red
90 NEXT I
100 FOR Y=0 TO 199
110 FOR X=0 TO 39
120 A=&H000F5000+INT(Y/8)*320+(Y AND 7)*40+X
130 POKE8 A, &H1B                   : REM pairs 00,01,10,11
140 NEXT X
150 NEXT Y

```

## 6.8 Fine scrolling and visible area

YSCROLL delays the visible picture by 0-7 raster lines. XSCROLL delays it by 0-7 pixels. Newly exposed pixels at the top or left remain border/background until the scrolled source coordinates enter the display area.

RSEL and CSEL choose the full text area:

| Bit  | Value 0   | Value 1    |
|------|---|------------|
| RSEL | 24 rows, first and last text rows blanked       | 25 rows    |
| CSEL | 38 columns, first and last text columns blanked | 40 columns |

Try this after the text-banner program above:

```

200 FOR S=0 TO 7
210 TED SCROLL S, S
220 FOR D=1 TO 120
230 NEXT D
240 NEXT S
250 TED SCROLL 0, 0

```

The picture slides down and right in eight small steps, then returns to the normal alignment.

## 6.9 The cursor

The cursor is a single character cell. Its position is the 11-bit linear offset stored in TED\_V\_CURSOR\_HI/L0 (row \* 40 + col), and its colour is in TED\_V\_CURSOR\_CLR. The cursor blinks automatically at one cycle every 30 frames. It is drawn as an underline after the rest of the frame has been rendered, so it appears over text and bitmap pixels in its cell.

This puts the cursor on row 12, column 20, in bright yellow:

```

10 P=12*40+20
20 POKE8 &H000F0F44, INT(P/256)
30 POKE8 &H000F0F48, P AND 255
40 POKE8 &H000F0F4C, &H77

```

## 6.10 Raster compare and status

---

TED\_V\_RASTER\_LO and bit 0 of TED\_V\_RASTER\_HI report the current scanline as a 9-bit value. The rendered TED frame is 0-271. Compare values outside the rendered frame are stored, but do not set the pending latch during the current visible frame.

Writing the same 9-bit value into the compare registers arms the raster compare. When the raster reaches the line, TED\_V\_RASTER\_STATUS bit 7 is set and the chip raises its interrupt line. The CPU clears the latch by writing 1 to bit 7 of TED\_V\_RASTER\_STATUS.

TED\_V\_STATUS bit 0 is the VBlank flag. Reading TED\_V\_STATUS acknowledges that flag. TED\_V\_RASTER\_STATUS is separate and remains pending until acknowledged with a write of 128.

This short polling example waits for scanline 100, changes the background to bright red, then acknowledges the compare:

```
10 TED ON
20 POKE8 &H000F0F20, &H18
30 POKE8 &H000F0F24, &H08
40 POKE8 &H000F0F30, &H76
50 POKE8 &H000F0F60, 100
60 POKE8 &H000F0F64, 0
70 IF (PEEK8(&H000F0F68) AND 128)=0 THEN 70
80 POKE8 &H000F0F30, &H72
90 POKE8 &H000F0F68, 128
```

Raster interrupts are how the C16 and Plus/4 split the screen into multiple regions with different scroll, colour, or character-set choices. The same trick works here.

## 6.11 Copper-driven TED bands

---

The VideoChip copper can also write TED registers on selected scanlines. Use SETBASE with \$F0F20, then MOVE the TED register index. The register index is (register address - \$F0F20) / 4.

For example, TED\_V\_BORDER is \$F0F40, so its copper index is 8. This hand-packed list changes the TED border from red to green on scanline 80. The copper word format is documented in Chapter 4.

```
10 TED ON
20 POKE8 &H000F0F40, &H72
30 L=&H00003000
40 POKE32 L+0, &H8003C3C8 : REM SETBASE $F0F20
50 POKE32 L+4, &H00050000 : REM WAIT y=80, x=0
60 POKE32 L+8, &H40080000 : REM MOVE border index
70 POKE32 L+12, &H00000075 : REM green
80 POKE32 L+16, &HC00000000 : REM END
90 POKE32 &H000F0010, L : REM COPPER_PTR
100 POKE32 &H000F000C, 1 : REM COPPER_CTRL
```

You should see the top border in bright red and the lower border in bright green.

## 6.12 Setup order, side effects, and limits

---

The shortest TED program from machine language:

1. Write a non-zero value to TED\_V\_ENABLE (or use TED ON).

2. Set the picture mode in TED\_V\_CTRL1 and TED\_V\_CTRL2 (or use TED\_MODE).
3. Choose where the video matrix and character set live with TED\_V\_VIDEO\_BASE and TED\_V\_CHAR\_BASE.
4. Fill the video matrix and colour RAM in the regions you chose.
5. To animate, poll TED\_V\_STATUS bit 0 for the vertical retrace, or arm a raster compare.

Important side effects and limits:

- Only the low byte of each TED video register is meaningful.
- TED\_V\_STATUS clears its VBlank bit when read.
- TED\_V\_RASTER\_STATUS is sticky until bit 7 is written back as 1.
- TED\_V\_RASTER\_LO and TED\_V\_RASTER\_HI are read-only.
- TED\_V\_CTRL2 bit 5 is stored as RES, but it does not clear TED VRAM.
- Base registers select offsets inside the 16 KB TED VRAM window; an out-of-range selection falls back to the default base.
- Standard and extended-colour text use one-pixel horizontal detail. Multicolour text and multicolour bitmap use double-wide colour pixels.
- High-resolution bitmap mode uses only the two four-bit matrix nibbles for each cell. Multicolour bitmap adds the full colour-RAM byte as the fourth colour source.

The next chapter covers ANTIC and GTIA, the Atari 8-bit picture chips, which live on layer 13.