



APPLICATION NOTE 4117

## Generating Custom Characters and Graphics by Using the MAX7456's Memory and EV Kit File Formats

*Abstract: The MAX7456 is a single-channel on-screen display (OSD) generator that allows the user to overlay custom-generated graphics over live video. This application note describes the memory organization in the MAX7456 and shows how to use the device to generate custom characters and graphics. The file formats for the MAX7456 evaluation (EV) kit are presented. The article also explains how these files can be manipulated simply by using a spreadsheet.*

### Overview

The [MAX7456](#) single-channel on-screen display (OSD) generator allows the user to overlay custom-generated graphics over live video. The MAX7456 lowers system cost by eliminating the need for an external video driver, sync separator, and video switch. The device serves all national and international markets with 256 user-programmable monochrome characters in 525 and 625 line standards. It easily displays information such as company logo, custom graphics, time, and date with arbitrary fonts and sizes. This application note describes how to generate custom characters and graphics with the MAX7456. The file formats for the MAX7456 evaluation (EV) kit are also presented.

### MAX7456 Memory Organization

A user-defined character set is combined with the input video stream to generate a CVBS plus OSD video output. A maximum of 256, 12 x 18 pixel characters can be stored, but may be reprogrammed dynamically. In 525 line mode, 13 rows x 30 characters are displayed; in 625 line mode, 16 rows x 30 characters are displayed.

The MAX7456 OSD contains two sets of memories: the display memory and the nonvolatile character memory (NVM).

### Display Memory

The display memory (SRAM) stores 480 character addresses that "point to" the characters stored in the NVM character memory. The content of the display memory is user-programmable with the SPI™-compatible serial interface. The display memory address corresponds to a fixed location on a monitor. See **Figure 1** below. Each character requires two bytes. The first byte contains the character number in the character memory (see discussion below); the second byte contains the character attribute status bits, shown in **Figure 2**. Characters are numbered from left to right and top to bottom.

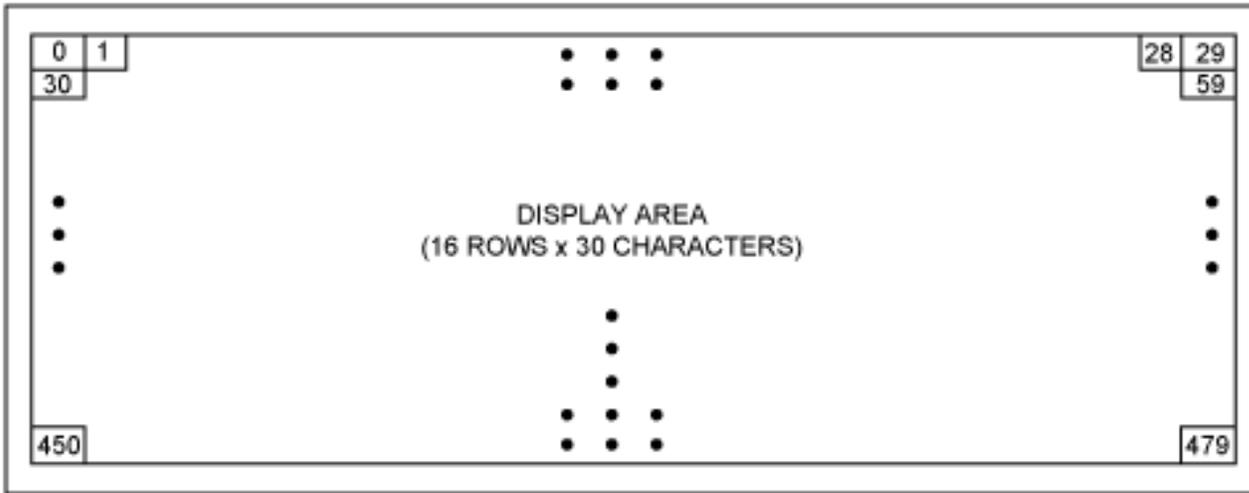


Figure 1. The fixed location of the display memory.

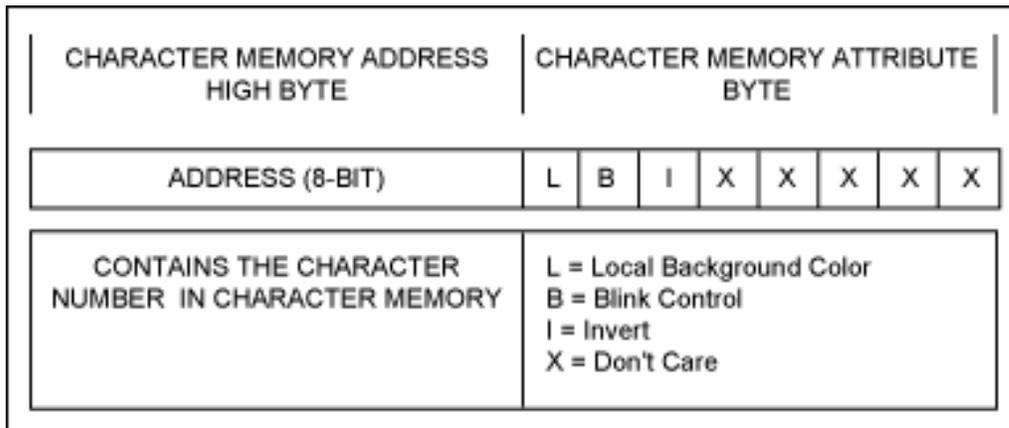


Figure 2. The display memory's character attribute status bits.

## Character Memory

The nonvolatile character memory stores the characters or graphic images to be displayed. The content of the character memory is user-programmable with the SPI-compatible serial interface. Each character consists of 12 horizontal x 18 vertical pixels. Each pixel is represented by two bits:

- 00 = Black, opaque
- 01 = Transparent
- 10 = White, opaque
- 11 = Transparent

There are, consequently,  $12 \times 18 = 216$  pixels per character. One 8-bit byte describes four pixels. Thus each character requires  $216/4 = 54$  bytes of data. (See **Figure 3**.)

To make addressing easier, the memory is organized in blocks of 64 bytes. Each of the first 54 bytes describes a character. The remaining 10 bytes are unused. Therefore the character memory is  $64 \times 256 = 16384$  bytes long.

Consequently, the number contained in each display-memory address location is, in fact, the address of the 64-byte block corresponding to the selected character.

		PIXEL COLUMN NUMBER											CHARACTER MEMORY ADDRESS LOW CMAL[5:0]	
		0	1	2	3	4	5	6	7	8	9	10	11	
PIXEL ROW NUMBER	0	CDMI [7, 6]	CDMI [5, 4]	CDMI [3, 2]	CDMI [1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	0, 1, 2
	1	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	3, 4, 5
	2	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	6, 7, 8
	3	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	9, 10, 11
	4	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	12, 13, 14
	5	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	15, 16, 17
	6	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	18, 19, 20
	7	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	21, 22, 23
	8	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	24, 25, 26
	9	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	27, 28, 29
	10	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	30, 31, 32
	11	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	33, 34, 35
	12	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	36, 37, 38
	13	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	39, 40, 41
	14	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	42, 43, 44
	15	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	45, 46, 47
	16	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	48, 49, 50
	17	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	[7, 6]	[5, 4]	[3, 2]	[1, 0]	51, 52, 53

2-BIT PIXEL DEFINITION:

[x, y]	00 = BLACK
[x, y]	10 = WHITE
[x, y]	X1 = TRANSPARENT (EXTERNAL SYNC MODE) OR GRAY (INTERNAL SYNC MODE)
[x, y]	X = DON'T CARE

Figure 3. Character memory configuration.

## Software for the Graphical User Interface (GUI) in the EV Kit

The MAX7456 EV kit is supplied with GUI [software](#) that lets the user control the device and download custom graphics.

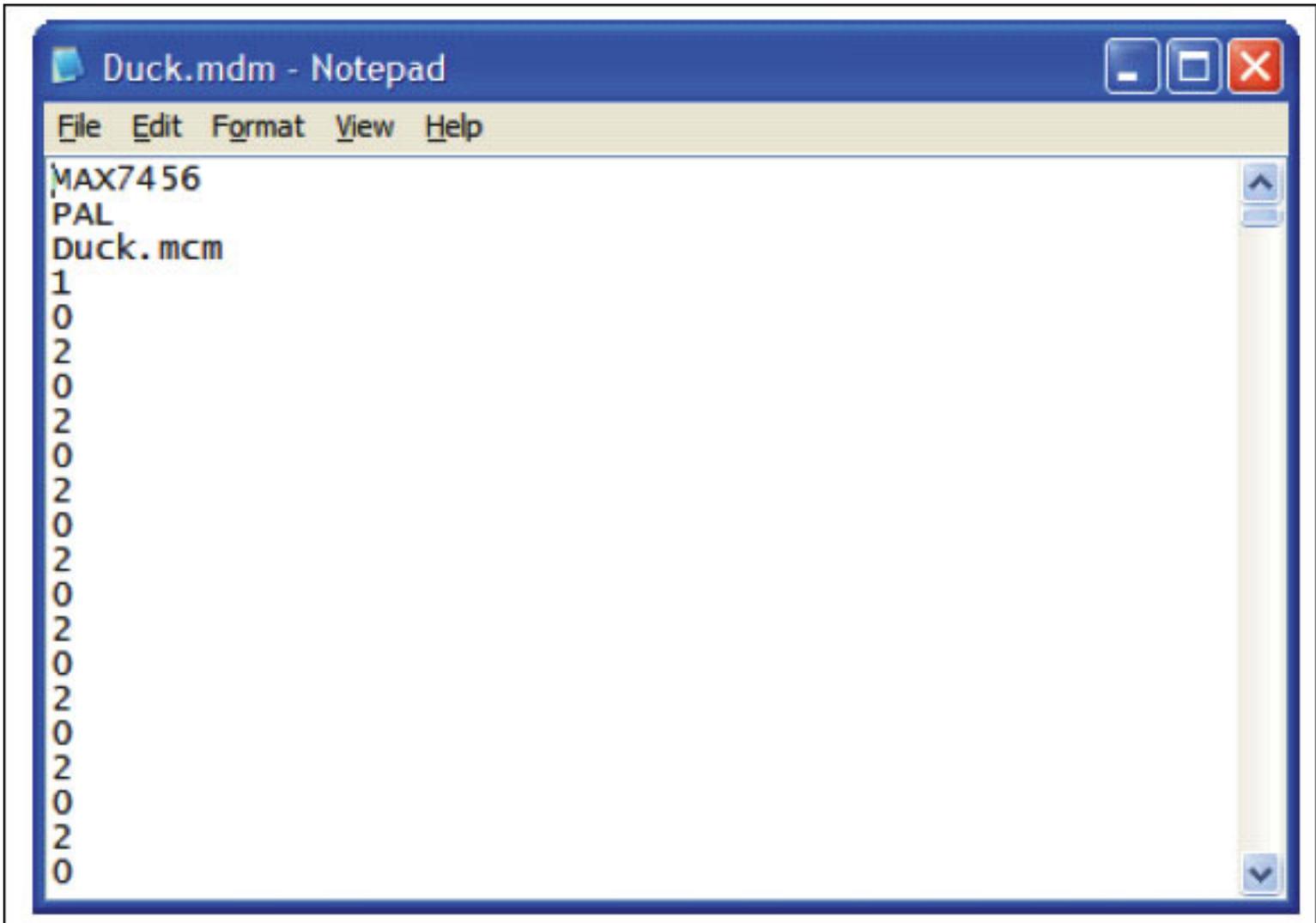
The files associated with the software are organized similar to the MAX7456's memories. The data for the character memory is held in one file, while the data for the display memory is held in another. The file extensions for the files are:

- \*.mdm Display Memory
- \*.mcm Character Memory

These ascii text files can be viewed with any text editor such as the Windows® Notepad. Each line is, therefore, terminated with an ASCII character return/line feed sequence.

### Organization of the Display Memory File

A segment of an .mdm display memory file is shown in **Figure 4** below.



```
MAX7456
PAL
Duck.mcm
1
0
2
0
2
0
2
0
2
0
2
0
2
0
2
0
```

Figure 4. Display memory organization.

In Figure 4, Line 1 gives the device type: the MAX7456. Line 2 gives the video standard: PAL or NTSC. Line 3 shows the filename of the associated character file: Duck.mcm. Lines 4 and following are line pairs that give the

character in each position on the screen, as shown in Figure 1 above.

Therefore, where  $x$  is the character position in Figure 1, the lines describing the character are given by:

$$n = 4 + 2 \times x \text{ character number}$$
$$n = 5 + 2 \times x \text{ character attributes}$$

The character attributes are show in **Figure 5**:

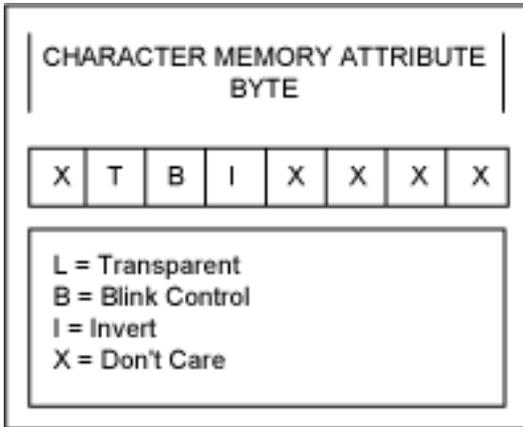


Figure 5. The character attributes for character positions shown in Figure 1.

In this example:

character 0 in the display uses character 1 from the character memory.  
character 0 is not blinking, inverted, or transparent  
character 1 in the display uses character 2 from the character memory.  
character 1 is not blinking, inverted, or transparent  
character 2 in the display uses character 2 from the character memory.  
character 2 is not blinking, inverted, or transparent  
etc.

Note that the length of an .mdm file depends on the video standard. If PAL is selected, the file will have 480 line pairs; the file will have 390 line pairs in NTSC.

## Organization of the Character Memory File

A segment of an \*.mcm file is shown below in **Figure 6**.

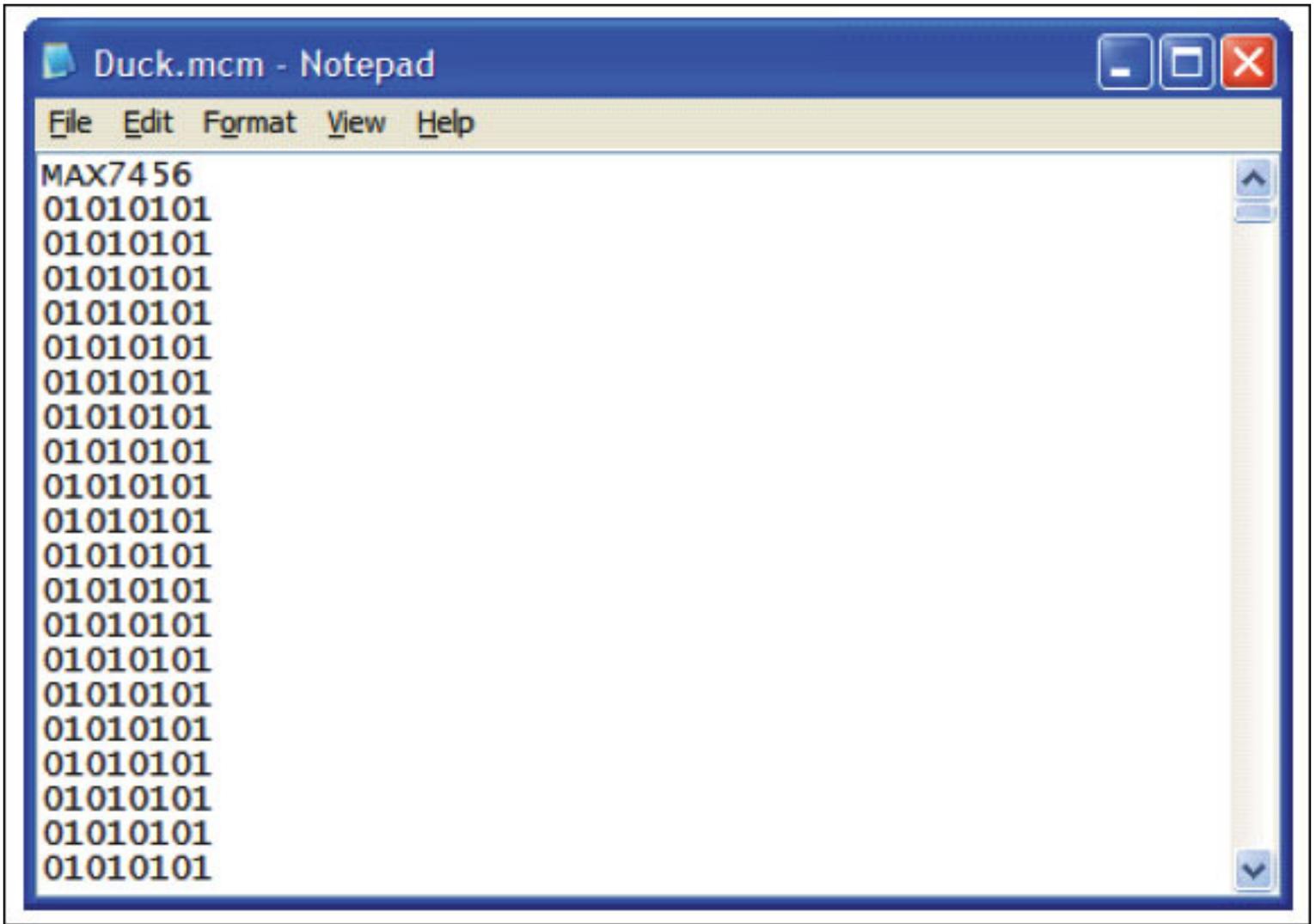


Figure 6. Organization of the display memory.

Line 1 gives the device type. Lines 2 and following define the pixels as shown above in Figure 3. In this example, each pixel of character 0 is set to "01," i.e., transparent.

### Tip for the Character Memory

It is not essential that each bit of character 0 be set to "01." When the device powers up, however, the display memory will by default reset to point to character 0. Hence, on power-up, the input video will be passed through with no erroneous overlays. After correctly initializing the device, normal characters can be overlaid as normal.

### Why a Pixel Appears as Gray or Transparent

If a pixel is defined in the character memory as transparent ("01" or "11" in the memory), then it can appear transparent or gray depending on several other conditions.

There are three conditions that affect a pixel:

1. Character level, defined by the attribute bit for each character set in the display memory.
2. Global level, defined in the background bit of the Video Mode Register (bit 7, Reg 01).
3. State of the external sync detector.

A bit can also be transparent, depending on the state of input video sync detection. The Truth Table is shown in

**Table 1**, which assumes that the pixel is defined as transparent at the pixel level. If not transparent, then the pixel will be displayed as black or white, independent of any other settings. Of course, it is also assumed that the OSD display is set to on.

**Table 1. Character Attributes for a Gray Pixel**

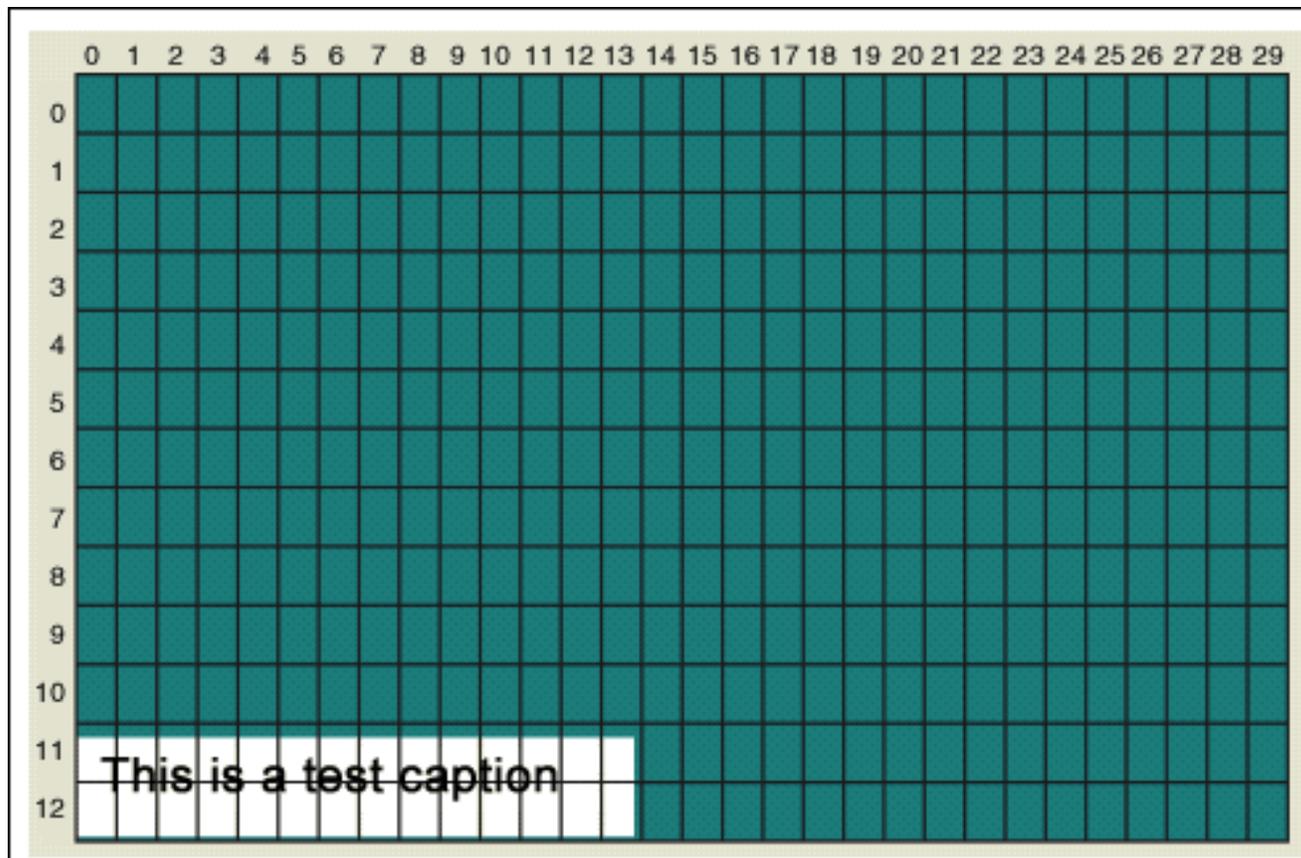
External Sync Detect (1 = Yes, 0 = No)	Global Background Mode	Character Level Attribute	Pixel Appearance
0	X	X	Gray
1	0	0	
1	0	1	
1	1	X	Gray

If the pixel appears as gray, then the brightness can be set on a row-by-row basis using registers RB0 to RB15 in address locations 90H to 9FH.

## Converting the Pixel Color in a Character Memory File

By understanding the structure of the EV kit files, one can easily convert pixels to black, white, or transparent color.

The character memory file is easily converted by a program like Excel. An example of this is shown below in **Figure 7**. The caption looks like this when loaded into the EV kit software.



*Figure 7. The color of a pixel can be altered by using Excel to change the character memory.*

The caption lettering is black on a white background. There is a simple way to convert this color scheme.

1. Load the .mcm file into Excel. As a text file, the .mcm file can be imported by simply opening it. Remember that the lines must be treated as text, otherwise leading zeros will be removed. Therefore, set the column data format to text during the import.

2. To preserve formatting, it is best to copy the column of values to a new spreadsheet.
3. Now chop the lines into 4 x 2 characters by using the Excel MID function.
4. Following this, translate the two character values.
5. Rebuild the lines.
6. Now replace the column in the original file with the assembled column, and save it.
7. As proof of what is achieved, load the new .mdm file into the EV kit software.

In this example, all black characters convert to white and all white characters to transparent. Thus:

"00" becomes "10"—black converts to white  
 "10" becomes "01"—white converts to transparent  
 "01" is unchanged—transparent is unchanged

This color conversion is done simply in Excel. The results are returned to the original file to preserve formatting.

The Excel formulae are simple and shown below. Column A is the original data.

Column B = MID(Ax, 1, 2) Selects two characters starting at character 1  
 Column C = MID(Ax, 3, 2) Selects two characters starting at character 3  
 Column D = MID(Ax, 5, 2) Selects two characters starting at character 5  
 Column E = MID(Ax, 7, 2) Selects two characters starting at character 7  
 Column F = IF(Bx="00","10",IF(Bx="10","01",Bx))  
 Column G = IF(Cx="00","10",IF(Cx="10","01",Cx))  
 Column H = IF(Dx="00","10",IF(Dx="10","01",Dx))  
 Column I = IF(Ex="00","10",IF(Ex="10","01",Ex))  
 Replaces a "00" with a "10" or a "10" with a "01"  
 Column J = CONCATENATE(F2,G2,H2,I2) Rebuilds the new word

Column J is copied to the original file and saved to preserve formatting (**Figure 8**).

	A	B	C	D	E	F	G	H	I	J
103	10101010	10	10	10	10	01	01	01	01	01010101
104	00000000	00	00	00	00	10	10	10	10	10101010
105	10101010	10	10	10	10	01	01	01	01	01010101
106	10101010	10	10	10	10	01	01	01	01	01010101

Figure 8. Formulae above are used to convert the pixel color in Excel.

Once loaded in the EV kit software, the new file appears as in **Figure 9**.

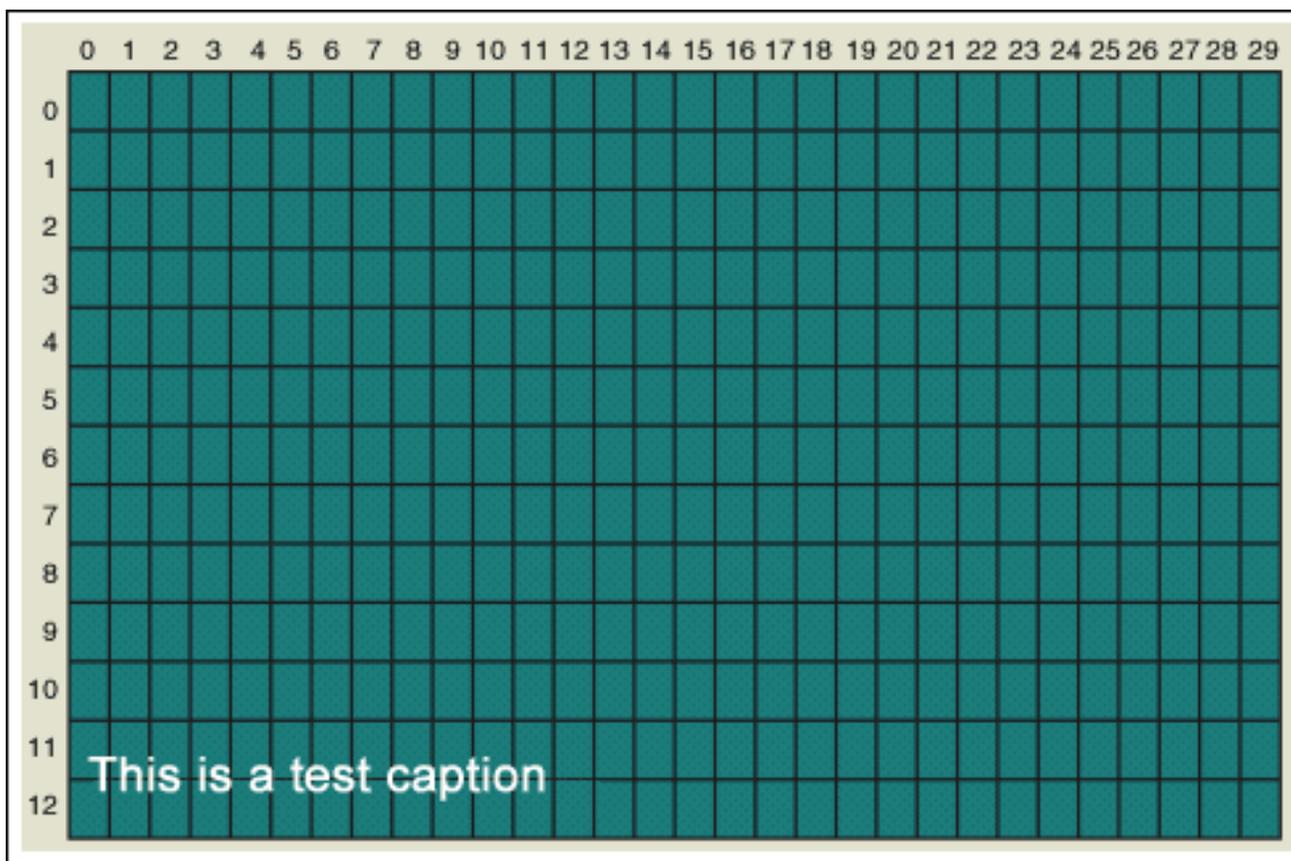


Figure 9. Illustration of how the altered pixel colors appear in Excel.

The Excel spreadsheet presented above and relevant files for the EV kit are available for download in [AN4117.zip](#).

SPI is a trademark of Motorola, Inc.  
Windows is a registered trademark of Microsoft Corp.

---

Application Note 4117: [www.maxim-ic.com/an4117](http://www.maxim-ic.com/an4117)

#### More Information

For technical support: [www.maxim-ic.com/support](http://www.maxim-ic.com/support)

For samples: [www.maxim-ic.com/samples](http://www.maxim-ic.com/samples)

Other questions and comments: [www.maxim-ic.com/contact](http://www.maxim-ic.com/contact)

---

#### Keep Me Informed

Preview new application notes in your areas of interest as soon as they are published. Subscribe to [EE-Mail - Application Notes](#) for weekly updates.

---

#### Related Parts

MAX7456: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)

MAX7456EVKIT: [QuickView](#) -- [Full \(PDF\) Data Sheet](#)

AN4117, AN 4117, APP4117, Appnote4117, Appnote 4117

Copyright © by Maxim Integrated Products

Additional legal notices: [www.maxim-ic.com/legal](http://www.maxim-ic.com/legal)