AN210124

Using Effective Texture Mapping with the 3D Graphics Engine in Traveo™ Family S6J3200 Series

Author: Isao Suetake

Associated Part Family: Traveo Family S6J3200 Series

Related Documents: For a complete list, click here.

AN210124 demonstrates the use of texture mapping for a 3D graphics application to maximize the performance of the 3D graphics engine in the Traveo™ family S6J3200 series MCUs.

1 Introduction

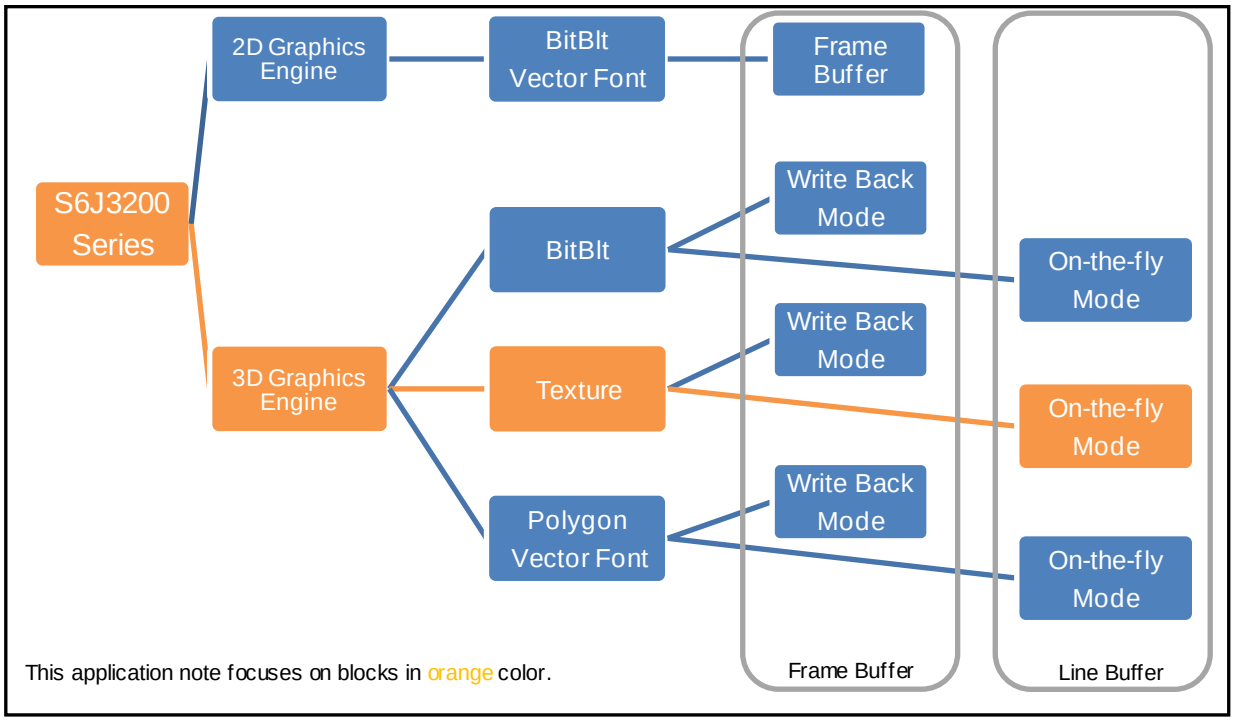
Usually, 3D graphics applications are developed in a PC environment. Texture mapping is a common method of 3D graphics applications. This application note shows you methods to port your texture mapping application to the Traveo family S6J3200 series MCUs that maximize the performance of the 3D graphics engine.

The S6J3200 series includes an ARM® Cortex®-R5 CPU core, 2D and 3D graphics engines, an external memory interface, CAN-FD, memory, and analog and digital peripheral functions. The product lineup of the S6J3200 series includes 216-pin and 208-pin packages and memory size variations. See the S6J3200 series fact sheet, datasheet, and hardware manual for details.

2 Target Use Case

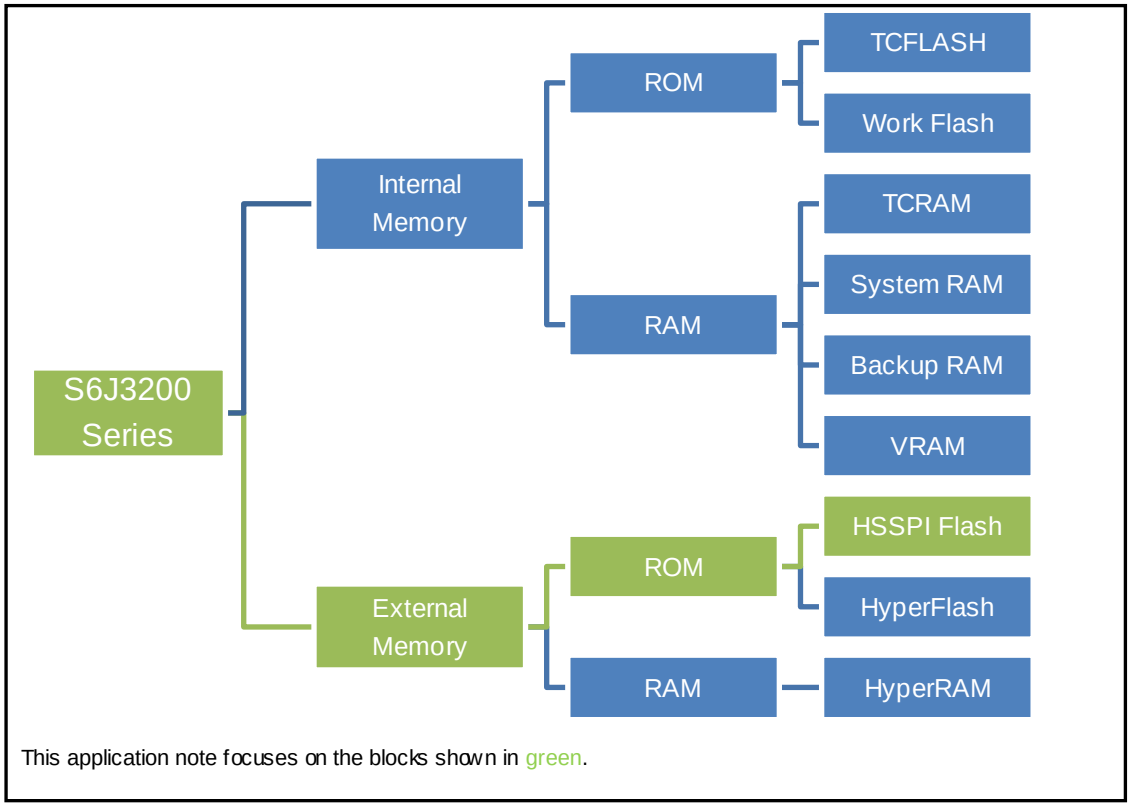
This document describes the use of effective texture mapping with a use case. The use case draws a picture with the on-the-fly rendering mode by the 3D graphics engine, as shown in Figure 1.

Figure 1.Target Drawing Mode



Texture map data is stored in the high-speed serial peripheral interface (HSSPI) of the external memory, as shown in Figure 2.

Figure 2.Target Memory



To maximize the performance of the 3D graphics engine, effective texture mapping is applied.

3 Effective Texture Mapping

This section describes eight methods to maximize the performance of the 3D graphics engine. All methods are recommended for application in the user system. The 3D graphics engine has a line buffer. In the case of the On-The-Fly mode, the 3D graphics engine draws pictures with synchronized LCD timing. The picture data are read from the external HSSPI flash. Removing HSSPI flash access is an efficient method to improve the 3D graphics application performance.

1. Select ETC2 (Ericsson Texture Compression) or ETC2EAC as the texture format.
2. Align texture coordinates to the display coordinates.
3. Divide a polygon into multiple polygons with triangle strips suited for external flash access, and then render polygons from the left side of the LCD screen.
4. Store the vertex data in the video RAM (VRAM).
5. Align the base address of the texture with 8-byte alignment.
6. Do not use multisample anti-aliasing (MSAA).
7. Do not use Z-buffer clear and color buffer clear.
8. Use a texture atlas.

3.1 Select ETC2 for Background and ETC2EAC for Picture

1. Select the ETC2 format for the background picture.
2. Select the ETC2EAC format for a picture that blends with another picture because the ETC2EAC format contains alpha bits.

Notes

- The compression ratio of the ETC2EAC texture format is 25 percent. Texture data can be read with high efficiency from the external HSSPI flash.
- The ETC2 format does not have alpha bits. However, the data size is half that of ETC2EAC. The size of the external HSSPI flash can be saved.
- In contrast to ETC2 and ETC2EAC, the SN format is not recommended for reading data from the external HSSPI flash because the data is not stored in a continuous address (SN format is a lossless compression type, in which an offset address is read first, and then texture data is read).

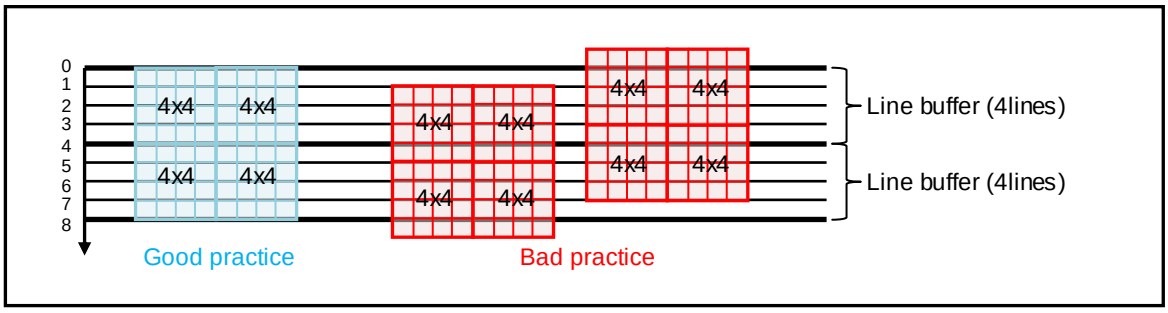
3.2 Align Texture Coordinates to Display Coordinates

Align the coordinates between the line buffer of the 3D graphics engine and the compressed texture data, as shown in Figure 3.

Notes

- The 3D graphics engine draws the texture with a line buffer, which contains four scanning lines. ETC2EAC and ETC2 data is compressed texture data for each 4x4 pixel.
- External HSSPI flash access occurs twice if each coordinate is not aligned because two blocks of ETC2EAC compressed data are needed.

Figure 3. Position of Display and Texture Coordinates



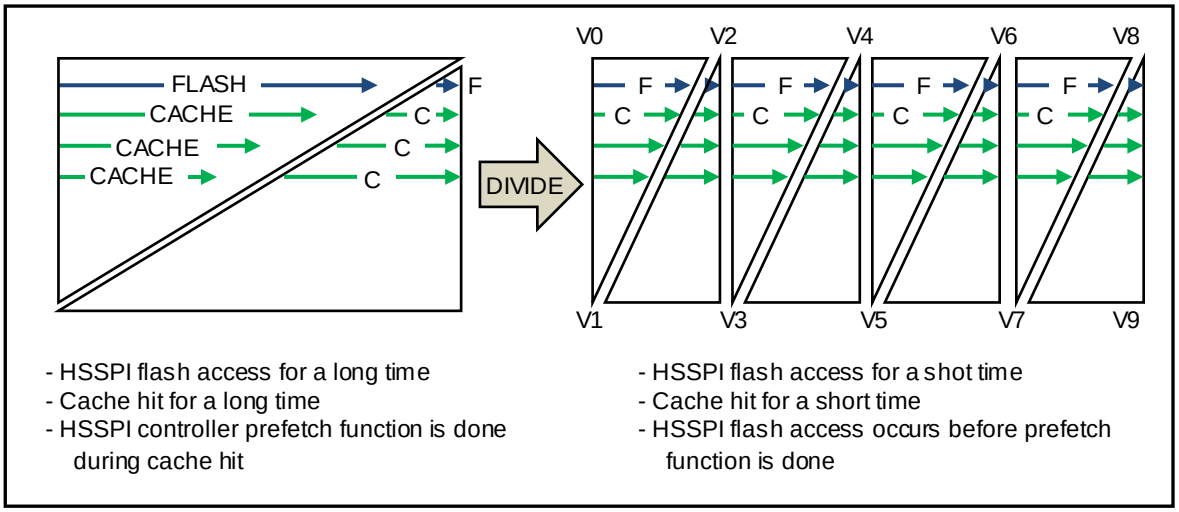
3.3 Divide a Polygon into Multiple Polygons

If the texture width is larger than 64 pixels, divide a polygon into multiple polygons equally with triangle strips suited for external flash access and render polygons from the left-hand side of the LCD screen as shown in Figure 4. If the Graphic HSSPI clock (G_SCLK0) is 66 MHz, the polygon width should be ETC2:200pixel, ETC2EAC:100pixel or less. Use this method if methods described in sections 3.1 and 3.2 are used.

Notes

- The 3D graphics engine includes a texture cache. When the first line of data is read, the second, third, and fourth lines of data are stored in the cache because the ETC2 and ETC2EAC format compresses the texture data of each group of 4x4 pixels. The second, third, and fourth lines of data are hit from the cache during a read. Therefore, the extra data read from the external HSSPI flash does not occur.
- If the texture width is large, the cache hit continues. The data read from the external HSSPI flash is stopped. The data read from the external HSSPI flash is continued if the texture width is small.
- The HSSPI controller has a prefetch function. The prefetch function works effectively if the texture data width is ETC2:200pixel, ETC2EAC:100pixel or less (if the Graphic HSSPI clock (G_SCLK0) is 66 MHz).

Figure 4. Divide into Polygons for Efficient HSSPI Flash Access



Note: Without the Align Texture Coordinates step, the polygon width should be 256 pixels or less for a working cache function.

3.4 Store Vertex Data in VRAM

Vertex data is stored in VRAM to prevent discontinuous reading from the external HSSPI flash.

Note

The 3D graphics engine reads the vertex data and the texture data separately. The extra overhead of the external HSSPI flash access is added because of the discontinuous reading addresses that result when alternately reading the vertex data and the texture data.

3.5 Align Base Address of Texture with 8-Byte Alignment

The texture base address is located in the external HSSPI flash with 8-byte alignment.

Note

The 3D graphics engine is implemented on the AXI bus. The AXI bus is 64 bits wide; therefore, 8-byte access is most efficient.

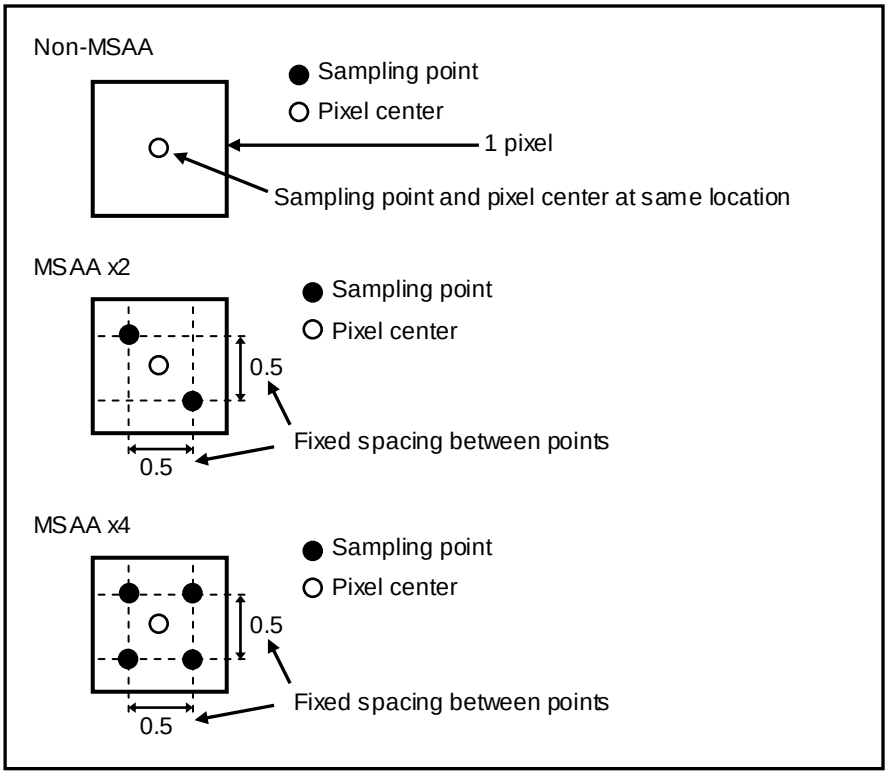
3.6 Do Not Use MSAA

MSAA is disabled to prevent reading data with the extra sampling point (see Figure 5).

Note

The extra HSSPI flash access occurs for reading the data from the previous set or next set of compressed data if MSAA is enabled.

Figure 5. MSAA Functions



3.7 Do Not Use Z-Buffer Clear and Color Buffer Clear

Z-buffer clear and color buffer clear are not needed if on-the-fly mode is used.

Note

The 3D graphics engine cannot access the external HSSPI flash during a Z-buffer clear and color buffer clear operation. Reading the next texture data from the external HSSPI flash is possible if these clear buffer operations are not in progress.

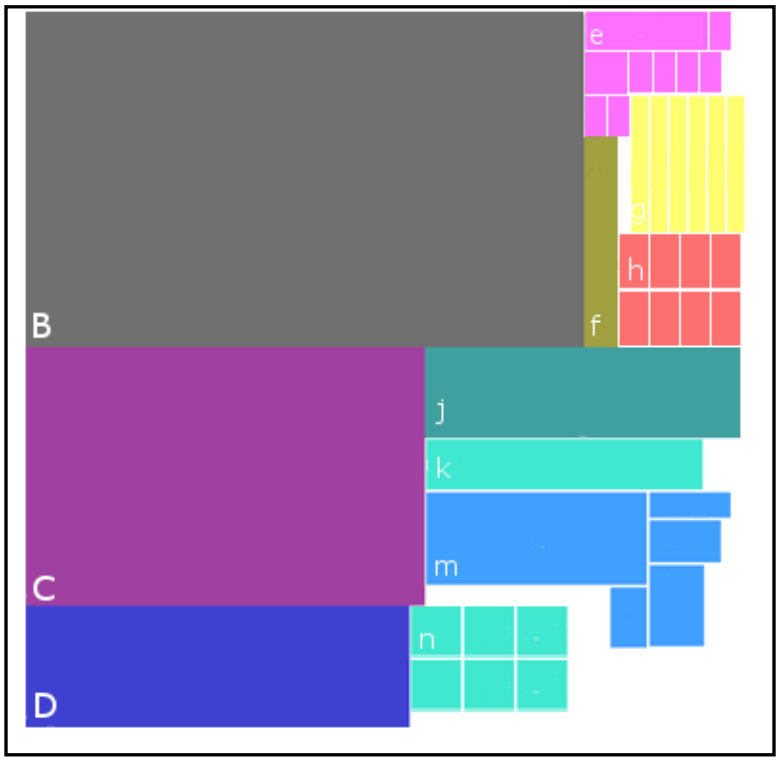
3.8 Use Texture Atlas

- The texture atlas is a large texture that includes multiple textures, as shown in Figure 6. The texture can be selected by the texture coordinates.
- The RegTexture command and BindTexture command are called once at the beginning of each scene.
  - If all the texture cannot be stored in one texture atlas, it can be stored in two separate texture atlases. In this case, the scene is divided into the first half and the second half. Two texture atlases are registered with the RegTexture command. The BindTexture command is issued for each texture atlas.
  - The ETC2 format and ETC2EAC formats cannot be mixed in one texture atlas. These texture atlases must be separated.

Note

The RegTexture command and BindTexture command are issued after all the texture is drawn.

Figure 6. Example of Texture Atlas



4 Summary

Cypress provides graphics drivers and sample software to help you get started using the Traveo S6J3200 series.

5 Related Documents

- S6J3200_NP708-00002 - S6J3200 Series 32-Bit Microcontroller with Graphics Traveo MCU Family Fact Sheet
- S6J3200_DS708-00003 - S6J3200 Series 32-Bit Microcontroller Traveo Family Datasheet
- S6J3200_MN708-00005 - S6J3200 Series 32-Bit Microcontroller Traveo Family Hardware Manual

Document History

Document Title: AN210124 – Using Effective Texture Mapping with the 3D Graphics Engine in Traveo™ Family S6J3200 Series

Document Number: 002-10124

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5174295	ISSU	05/11/2016	New application note
*A	5823023	AESATMP9	07/18/2017	Updated logo and copyright.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709