Customer training workshop
TRAVEO™ T2G graphics subsystem
Target products

Target product list for this training material:

<table>
<thead>
<tr>
<th>Family Category</th>
<th>Series</th>
<th>Code flash memory size</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRAVEO™ T2G automotive cluster 2D</td>
<td>CYT3DL</td>
<td>Up to 4160 KB</td>
</tr>
<tr>
<td>TRAVEO™ T2G automotive cluster 2D</td>
<td>CYT4DN</td>
<td>Up to 6336 KB</td>
</tr>
</tbody>
</table>
Abstract

› About this training
  – TRAVEO™ T2G graphics subsystem description for beginners
  – This helps engineers to learn the basics of GFX hardware
  – Describes hardware components of TRAVEO™ T2G graphics subsystem with a short description
  – Introduces simple applications and describes how they work (only one example is currently available)
  – No driver/no usage description as of now

Note that this document does not describe detailed features and specifications of the hardware; please refer to the hardware TRM or datasheet for the specific information.
Table of contents

› Session 1: Creating a display image
  - Graphics subsystem overview
  - Basic BLIT example abstraction
  - Making display image
  - Related hardware
  - BLIT data flow (IBO mode)
  - BLIT data flow (LBO mode)
  - Data flow (OTF mode)

› Session 2: Outputting display image
  - Related hardware
  - Shadow registers
  - Outputting display image sequence (IBO/LBO mode)
  - Outputting display image sequence (OTF mode)
Table of contents

› Session 3: Detailed description of making display image
  – Related hardware
  – Detailed sequence BLIT (IBO mode)
  – Detailed sequence BLIT (LBO mode)
  – Detailed sequence BLIT (OTF mode)
  – External line buffer and LBH control

› Appendix
Session 1: Creating a display image
Graphics subsystem overview
Introduction to TRAVEO™ T2G cluster

- Graphics subsystem connects to the CPU subsystem
Components of TRAVEO™ T2G graphics (GFX) subsystem
- 2D GFX core
- Video RAM (VRAM)
- Video I/O
- JPEG decoder (JPEGDEC)
- FPD-link
- MIPI CSI-2
- AXI bus matrix

Interfaces
- RGB
- Low voltage differential signal (LVDS)
- D-PHY lanes

Graphics subsystem overview (1/4)

Hint Bar
Review TRM section 35.3 for additional details
Graphics subsystem should not be powered off while AXI interface between CPU Subsystem and VRAM is not IDLE.
2D GFX core

- **2D GFX core (1)**
  - Realizes raster graphics (copy, blend, etc), and vector graphics (drawing figures, fonts).

- **Bus access (2)**
  - 2D GFX core mainly interacts with the AXI infrastructure to fetch and store image data.

- **Line buffer handshake (LBH) (3)**
  - Interface to video I/O for blit-on-the-fly functionality.

Review TRM section 35.3.1 for additional details.
Video I/O

- Fetches frame buffer (line buffer) and realizes layer blending
- Processes input captured image (1)
- Outputs blended image (2)
- Video I/O mainly interacts with the AXI infrastructure to fetch and store image data.
- Interface to 2D GFX core for blit-on-the-fly functionality.

Review TRM section 35.3.2 for additional details.
VRAM

- Memory where image buffer, commands (for GFX Core), are saved.

- Bus access (1)
  - VRAM can be accessed via AXI bus matrix in graphics subsystem.
  - VRAM has multiple AXI Read ports and multiple AXI write ports, multiple memory banks so that multiple masters can access simultaneously.

Note: See the specific device datasheet to check VRAM capacity.
JPEGDEC

- JPEG image decompression from source to destination buffer in memory

- Bus access
  - JPEGDEC mainly interacts with the AXI infrastructure to fetch JPEG image and store decompressed image data.

Review TRM section 35.3.4 for additional details.
Video out

- Video out
  - Provides FPD-Link/LVDS or RGB display output port functionality

- Supporting mode
  - Dual channel (high-resolution display)
  - Dual screen (two display)
Video out

- Display clone

- Side-by-side display (2 displays next to each other)
Video in

- Video in
  - Allows to connect to an external video source such as a camera
- Interface
  - User can select from RGB or D-PHY lane input interface
  - D-PHY lane
    - The receive interface comprises 4 differential data pins + 1 clock pin.
Bus masters and a bus slave

- Graphics AXI bus masters (highlighted with a red rectangle)
  - 2D GFX core
  - Video I/O
  - JPEGDEC
- Graphics AXI bus slave (highlighted with a blue rectangle)
  - VRAM
- Connection CPUSS AXI bus
  - “GFXSS AXI bus matrix” is connected to “CPUSS AXI bus matrix”.
  - GFX AXI masters can access CPUSS AXI bus slaves (System RAM, Internal/external flash, etc).
  - CPUSS AXI bus masters (CM7s, Ethernet, AXIDMA etc) can access the VRAM.
GFX subsystem clock

- **Graphics clock from SRSS**
  - CLK_HF10 (Recommended source clock: PLL400#3)
  - Graphics subsystem root clock
  - CLK_HF11 (Recommended source clock: PLL400#4 or PLL200#2)
    - Display engine 0 (Part of Video I/O)
    - FPD-Link 0
  - CLK_HF12 (Recommended source clock: PLL400#4 or PLL200#2)
    - Display engine 1 (Part of Video I/O)
    - FPD-Link 1

Note: See the specific device datasheet to check maximum frequency of the graphics subsystem root clock.
Basic BLIT example abstraction
Basic BLIT example: Graphics terms – Alpha blending

› Alpha blending
- In computer graphics, alpha compositing or alpha blending is the process of combining one image with a background to create the appearance of partial or full transparency. ([https://en.wikipedia.org/wiki/Alpha_compositing](https://en.wikipedia.org/wiki/Alpha_compositing))
- For example, Alpha = 0: full transparency, Alpha = 255: no transparency
- RGBA32 format image has alpha data for each pixel aside from red, blue, and green color data.

![Alpha = 255](image1)
![Alpha = 127](image2)
![Alpha = 0](image3)

BLIT: Block Image Transfer. A method to create an image by filling, copying, and so on.
Basic BLIT example: Graphics terms – Frame buffer/rendering

› Frame buffer
  – A framebuffer (frame buffer, or sometimes framestore) is a portion of random-access memory (RAM) containing a bitmap that drives a video display. It is a memory buffer containing data representing all the pixels in a complete video frame. ([https://en.wikipedia.org/wiki/Framebuffer](https://en.wikipedia.org/wiki/Framebuffer))

› Rendering or Block Image Transfer (BLIT) in TRAVEO™ T2G GFX
  – Rendering or image synthesis is the process of generating a photorealistic or non-photorealistic image from a 2D or 3D model by means of a computer program. ([https://en.wikipedia.org/wiki/Rendering_(computer_graphics)](https://en.wikipedia.org/wiki/Rendering_(computer_graphics)))
Basic BLIT example: Output constant image using BLIT engine

› Example 1 description
  - Output 3 types of moon images on a constant colored background.

› Display
  - 800x480 display (992 x 500 in total)
  - 60 frame per sec
  - RGB24 format

› Frame buffer image
  - RGBA32 format

› BLIT (Block Image Transfer) operation
  - Fill with constant color (1)
  - Copy an image (2)
  - Alpha blend the image (3)
  - Alpha blend the image with alpha mask (4)

› Use three modes of BLIT
  - Image base operation (IBO)
  - Line base operation (LBO)
  - Line base on-the-fly (OTF)

Output result
Basic BLIT example: Source data

› RGBA moon image
  - Alpha value of pixels shown as black area around the moon is “0” (full transparency)
  - Alpha value of pixels of the moon is “255” (no transparency)

› Alpha mask data
  - Only alpha value data
  - Black color describes alpha value = “0” (full transparency)
  - White color describes alpha value = “255” (no transparency)
Basic BLIT example: BLIT operation IBO mode

› Image-based operation (IBO)
  – Image-based operation describes rendering the image, object by object, into a frame buffer until the desired scene is composed.

› Rendering example
  – Fill
  – Copy
  – Blend
  – Blend with mask
Basic BLIT example: BLIT operation LBO mode

› Line-base operating
  - Line-based operation describes rendering slice by slice, into an frame buffer until the desired scene is composed. (slice = 8 lines)

› Rendering Example (6 Slices)
  - Slice 1 (Fill)
  - Slice 2 (Fill)
  - Slice 3 (Fill, Copy, Blend, Mask)
  - Slice 4 (Fill, Copy, Blend, Mask)
  - Slice 5 (Fill)
  - Slice 6 (Fill)
Basic BLIT example: BLIT operation OTF mode

› Line-based on-the-fly operating (OTF)
  - Line-based on-the-fly operation does not make the frame buffer; instead it renders a part of the frame image into a ring buffer called “external line buffer” slice by slice and display image directly be shown on the display.

› Rendering example (6 slices)
  - Slice 1 (Fill)
  - Slice 2 (Fill)
  - Slice 3 (Fill, copy, blend, mask)
  - Slice 4 (Fill, copy, blend, mask)
  - Slice 5 (Fill)
  - Slice 6 (Fill)
### Basic BLIT example: Summary of BLIT modes

<table>
<thead>
<tr>
<th></th>
<th>IBO mode</th>
<th>LBO mode</th>
<th>OTF mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traveo I backward compatibility</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Performance</td>
<td>Low</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Memory bandwidth</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Buffer size</td>
<td>High</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Supporting features</td>
<td><strong>Maximum features</strong></td>
<td>Restricted features</td>
<td>Restricted features Restricted usage</td>
</tr>
</tbody>
</table>

**Hint Bar**

Review appendix page 135 - 141 to know what feature/usage are restricted in LBO/OTF mode
Making display image
(Rendering frame buffer)
Related hardware
BLIT data flow related hardware: Overview

› Hardware described in this section
  – 2D GFX core (BLIT)
  – VRAM

› Here, we learn the manner in which frame image is rendered by BLIT

Review TRM chapter 35.3.1 - 35.3.3 for additional details
Related hardware: 2D GFX Core - BLIT engine

- **BLIT engine**
  - Fetches image data, stores processed image data via “AXI Bus Matrix” (e.g. rotation)
  - Connected with “Video I/O” via LBH (Line-based handshake) for blit-on-the-fly functionality
  - Outputs status signal to synchronize “Command sequencer”
  - Raster graphics: Fill, copy, blend, scale, rotate, etc.
  - **Modes**
    - Image based operation (IBO) mode
    - Line based operation (LBO) mode
    - Line based operation on-the-fly (OTF) mode
Related hardware: 2D GFX core - BLIT engine (IBO mode)

- **Fetch engines: SRC, MASK, DST**
  - **SRC (1)**
    - **Input**
      - Pixel image fetched via AXI bus matrix
    - **Output**
      - Pixel image output by MASK
      - Pixel image output by DST
  - **MASK (2)**
    - **Input**
      - Pixel image fetched via AXI bus matrix
    - **Output**
      - Pixel image to SRC for Alpha channel, Planar color formats, warping coordinate buffer
      - Pixel image into Pixel processing pipeline
  - **DST (3)**
    - **Input**
      - Pixel image fetched via AXI bus matrix
    - **Output**
      - Pixel image to "blitblend0" for blend operation
      - Pixel image to MASK for planar color formats
      - Pixel image to Pixel processing pipeline

- **blitblend0 (pixel processing pipeline) (4)**
  - Blending of two images with same dimension

- **Store engine (5)**
  - Write back to memory in different formats (color depth, compression)
Related hardware: 2D GFX core - BLIT engine (LBO/OTF mode)

- Fetch engines: SRC0, SRC1, SRC2 (1)
  - These work in parallel.
  - Input
    - Pixel image fetched via AXI bus matrix
    - Alpha mask data from internal line buffer
  - Output
    - Pixel image to pixel processing pipeline
- blitblend0, blitblend1, blitblend2 (2)
  - Input
    - Pixel image from pipeline
    - Destination pixel data from internal line buffer
- Line control block (3)
  - Input
    - Three pixel pipelines
  - Output
    - 1 pixel output pipeline (connected to store)
- Internal line buffer (4)
  - Contains RGBA buffer and MASK buffer.
  - This is called I-line buffer in this document
- LBH control (line buffer handshake control) (5)
  - Interconnects the GFX2D store unit with the VIDEOIO fetch units for OTF operation.
Related hardware: VRAM (1/3)

- Simultaneous access
  (an example)
  - 5 AXI read slave
  - 3 AXI write slave
  - 8 logical banks
  - Round robin with per port priority settings

Note: These numbers (5 AXI Read Slave 3 AXI Write Slave, 8 logical banks) may depend on TVII device. See the specific device datasheet to check the numbers.
BLIT data flow (IBO mode)
BLIT data flow

- Assume we have a moon image and the alpha mask data in VRAM
- The frame buffer is blank in this moment.

: RGBA moon image (1)

Complete frame image

: Image data flow
Data flow in IBO mode: Fill

- Fill
  - BLIT in 2D GFX core works
  - “SRC” generates constant color stream. (1)
  - “store engine” stores the constant color data into the frame buffer. (2)
  - Data passes through ROP, CLUT, MATRIX, GPSCALEL, blitblend0.
Data flow in IBO mode: Copy

- **Copy**
  - BLIT in 2D GFX core works
  - “SRC” fetches the “moon” image. (1)
  - “store engine” stores the fetched image into the frame buffer. (2)
  - Data passes through ROP, CLUT, MATRIX, GPSCALE, blitblend0.
Data flow in IBO mode: Blend

- **Blend**
  - “SRC” fetches the “moon” image. (1)
  - “DST” fetches the image located on the destination position. (2)
  - The two images are blended by “blitblend0”. (3)
  - “store engine” stores the blended image into the frame buffer. (4)
Data flow in IBO mode: Blend with alpha mask

- Blend with alpha mask
  - “SRC” fetches the “moon” image. (1)
  - “MASK” fetches alpha mask data and passes it to “SRC”’s input. (2)
  - “SRC” applies input alpha to the fetched image and output it. (3)
  - Other operations are same as “Blend” operation.

![Diagram of image data flow]
BLIT data flow (LBO mode)
Data flow in LBO mode

- This section explains about rendering a slice of the frame buffer.
- Assume we have the “moon” image and alpha mask data in VRAM.
- Rendering the frame buffer is on the way.
- Rendering continues 8 lines by 8 lines until the frame buffer is completed. (8 lines = 1 slice)
Data flow in LBO mode: Fill

- Fill
  - BLIT in 2D GFX core works
  - "Line control block" generates constant color stream.
  - The color data will be stored in the internal line buffer.

Note: Image saved in the internal line buffer is called a "Slice". One slice is equal to eight lines.
Data flow in LBO mode: Copy

- **Copy**
  - **SRC0** - A fetch engine in BLIT works
  - It copies a slice of the moon image into the internal line buffer RGBA.
Data flow in LBO mode: Blend

- **Blend**
  - SRC1 - A fetch engine in BLIT- works
  - It reads a slice of the moon image, then reads the slice date from the internal line buffer RGBA, and blends them.
  - It writes back the blended data to the destination in internal line buffer.

Note: Previous “copy” operation doesn't necessarily finish. SRCs in BLIT can work in parallel.
Data flow in LBO mode: Blend with alpha mask

- Blend with alpha mask (Let us assume the previous operations are all completed for simplification of the description)
  - SRC2 copies the alpha mask data into the internal line buffer mask
  - SRC0 reads a slice of the moon image.
  - Applies the alpha mask to the image
  - Blends it with the destination image in the internal line buffer RGBA
  - Writes it back into the internal line buffer.
Data flow in LBO mode: Store

- **Store**
  - “Store engine” in the BLIT stores the contents of the internal line buffer, RGBA, into the frame buffer as a slice.
  - Continue fill, copy, blend and, blend with alpha mask operations until the frame buffer completes.
Data flow (OTF mode)
Data flow in OTF mode

- Let us assume we have the "moon" image and alpha mask data in VRAM
- The way to render slices is the same as in the LBO mode (1)
- Video I/O does not wait for frame rendering to complete. It starts fetching after the last slice stored in the E-Line buffer. (2)
Session 2: Outputting display image
Related hardware
Related hardware: Overview

› Hardware described in this section
  - VRAM
  - Video I/O
  - FPD-Link0

› Here, we will learn the manner in which the frame image is displayed.

Review TRM chapter 35.3.2, 36 for additional details
Related hardware: Video I/O

- **Components of video I/O**
  - Composition engine
    - Input
      - Pixels from AXI bus matrix (5 fetch engines)
      - Pixels from capture engine
      - Pixels from frame dump
    - Output
      - Pixels to AXI bus matrix (1 store engine)
      - Secondary stream to display engine
      - Primary stream to display engine
    - LBH for line base operation
  - Display engine 0
    - Input
      - Secondary stream from composition engine
      - Primary stream from composition engine
    - Output
      - Video stream with display timing information to FPD-Link
      - Pixels to frame dump

[Diagram of Video I/O system with labels for components and connections]
Related hardware: Video I/O composition engine (1/2)

› Composition engine
  - Generates a sequence of frames which are overlaid into the active area of the display stream.
› Five background layers
  - Four constant color layers
  - One capture input layer
› Five foreground layers
  - Two fractional layers
  - Two decoding layers
  - One warping layer

Diagram: Video I/O composition engine flowchart.
Related hardware: Video I/O composition engine (2/2)

- Components of composition engine (unused IPs are not listed)
  - Fetch decode 0
    - Input
      - Pixels from AXI bus matrix
    - Output
      - Pixels to secondary stream of layer blend input
  - Const4
    - Output
      - Constant color pixels to primary stream of layer blend input
  - Layer Blend 1~5 (only Layer Blend 1 used)
    - Input
      - Background (primary) pixel stream
      - Foreground (secondary) pixel stream
    - Output
      - Blended layer stream to external destination interface
  - extdest 4
    - Input
      - Stream from Layer Blend
    - Output
      - Primary stream to Display Engine 0
Related hardware: Video I/O display engine (1/2)

- Display engine 0
  - Generates video output to the Fpdlink device from two inputs: primary and secondary stream
  - Generates vertical blanking intervals and synchronization pulses
Related hardware: Video I/O display engine (2/2)

- Components of display engine
  - Frame generator
    - Input
      - Primary stream from composition engine
      - Secondary stream from composition engine
    - Output
      - Pixels on continuous pixel protocol with display timing information
  - Gamma cor (Pass)
  - dither (Pass)
  - TCON (timing controller)
    - Input
      - Video stream from dither
    - Output
      - Freely programmable control and data signals to Fpdlid

Review TRM chapter 35.4 for additional details.
Review AN231948 (Graphic applications with TRAVEO T2G family) 2.4 for additional details.
Related hardware: Video out

- **Video out**
  - Only TCON 0 has two channels.
  - Only channel 0 of TCON 0 is used for single display output.
  - FPD-Link 0 receives pixel data from TCON0 and outputs LVDS signals.

Review TRM chapter 35.4 for additional details.
Review AN231948 (Graphic applications with TRAVEO T2G family) 2.3 - 2.4 for additional details.
Related hardware: Video out FPD-Link

- **TCON**
  - The TCON defines color data and status/timing information map as shown in this image.
  - Dual channel mode supports the following schemes:
    - Interleaved
    - Split

Review TRM chapter 35.4 for additional details.

Review AN231948 (Graphic applications with TRAVEO T2G family) 2.3 - 2.4 for additional details.
Related hardware: Video out FPD-Link

FPD-Link
- **Input**
  - Programable pixel protocol with display timing information.
  - Data: 7bit x 4line
  - Clock: 7bit (constant) x 1line
- **Output**
  - Serialized display image data for LVDS interface

Review TRM chapter 36 for additional details.
Review AN231948 (Graphic applications with TRAVEO T2G family) 2.3 for additional details.
Shadow registers
Shadow register load

- Almost all VideoSS registers are shadowed.
- Shadowed registers allow to set up the next operation while the previous one is still in progress.
- In this example…
  - VIDEO I/O settings
    - Core: CPU
    - GFX HW: Video I/O components
    - Interrupt: frame generator programable interrupt 0 (frame complete)
  - BLIT operation
    - Core: CMDSEQ
    - GFX HW: BLIT
    - Interrupt: BLIT shdLoad (BLIT store complete)
Outputting display image sequence (IBO/LBO mode)
Processing flow

› Initialize GFX system
› Set video interface
› Open display
› Create window
› BLIT
› Commit window

Note: In the image on the right,
- Frame buffer for IBO/LBO mode
- External Line buffer for OTF mode
Processing flow: Initialize GFX system

› This process affects the entire graphics subsystem
  - Enable graphics subsystem
  - Enable GFX2D
  - Enable display 0 (Not applicable for rev A0)

› Root clock will be provided to the graphics subsystem
  - Video subsystem root clock: CLK_HF10
  - Source PLL: PLL400#3
  - Maximum frequency: 250MHz
  - Set the PLL so it outputs maximum frequency clock
  - Route the PLL output to CLK_HF10
Processing flow: Set video interface

- Video output interface is set in this procedure
- Clock with frequency which realizes display resolution and FPS will be provided to FPD-link and display engine.

Start

1. Initialize GFX system

2. Set video interface

3. Open display

4. Create window

5. BLIT

6. Commit window

End

Graphics subsystem

- Video RAM
- Frame buffer (or E-line buffer)
- AXI bus matrix
- 2D GFX core
- Video I/O
- Composition engine
- Display engine 0
- Fpdlink 0
- LVDS

CLK_HF10
CLK_HF11
Processing flow: Set video interface - Clock settings

- Check minimum and maximum input clock frequency of FPD-link (min = 110 [MHz], max = 220 [MHz] in this example)
- Considered formulas (for single display)
  - \( FREQ_{\text{CLK DOT0}} = FREQ_{\text{CLK PIX0}} = 29,760,000(922 \times 500 \times 60 \text{ Frame Per Second}) \)
  - \( FREQ_{\text{CLK DSP0}} / 2 = FREQ_{\text{CLK DOT0}} \)
  - \( 110[\text{MHz}] \leq FREQ_{\text{CLK HF11}} \leq 220[\text{MHz}] \)
- Frequency of clocks are as follows:
  - \( FREQ_{\text{CLK DOT0}}: 29,760,000\text{Hz} \)
  - \( FREQ_{\text{CLK DSP0}}: 59,520,000\text{Hz} \)
  - \( FREQ_{\text{CLK PIX0}}: 29,760,000\text{Hz} \)
  - \( FREQ_{\text{CLK HF DSP0}} = FREQ_{\text{CLK HF11}} \text{(from PLL400\#4 = 119,040,000Hz} \)
- Divider values
  - Video I/O divider \#0: dividing by 2
  - GFX SS Divider \#0: dividing by 1
  - Output Divider \#0: dividing by 1

Review AN226071 (Clock Configuration Setup in Traveo II Family CYT4D Series) for additional details.

Review AN231948 (Graphic applications with TRAVEO T2G family) 2.2 for additional details.
Processing flow: Set video interface - TCON and FPD-Link settings

› Initialize TCON0
  - Set color data and display timing map according to the display used.

› Initialize FPDLINK0
  - Make sure output divider #0 of FPDLINK divides by 1
  - Bit order: MSB first

Hint Bar
Review TRM chapter 35.4, 36 for additional details
Review AN231948 (Graphic applications with TRAVEO T2G family) 2.3 - 2.4 for additional details.
Processing flow: Open display

- Composition engine starts outputting constant color data
- Frame generator in the display engine starts processing input data
- After this procedure, black constant display images are shown in the LVDS display.
Processing flow: Open display - overall

› Set shadow registers of
  – const 4 (1)
    – output all “0”
  – extdest 4 (2)
    – input from const 4
  – Frame generator (3)
    – display resolution, interrupt timing, etc
› Trigger loading shadow registers
  – EXTDEST4_REQUEST (in extdest 4)
  – FGSLR (in Frame Generator)
› Note: From here on, image frames are provided to the frame generator and it outputs interrupt signals to CPU at the interval of frames. (Technically, at the timing of (1, 1) pixel in the frame. The interrupt is programable frame generator int0)
Processing flow: Open display - Frame generator

› Frame generator settings
  - Hactive = 800
  - Htotal = 992
  - Hbp = 96
  - Hsync = 72
  - Vactive = 480
  - Vtotal = 500
  - Vbp = 10
  - Vsync = 7
  - Programable interrupt 0
    - When the frame generator processes pixel of (1,1).
      Note: This interrupt will guarantee the CPU that previously shadowed registers are loaded.
Processing flow: Create window

- Fetch engine (fetch decode 0) and layer blend 1 in Composition Engine will be initialized in this procedure.
- Layer blend 1 is not in Blend mode yet. It received only const 4 output as a primary (background) image. Output from fetch decode 0 is ignored.
### Processing flow: Create window - overall

- Wait frame generator programable interrupt 0
- Set fetch decode 0 and layer blend 1 shadow registers
  - Fetch decode 0 (1)
    - set widow size, and etc.
  - Layer blend 1 (2)
    - Does not blend primary and secondary stream now. Only primary stream will be used.
    - extdest 4 (3)
      - input from layer blend 1
- Trigger loading shadow registers
  - EXTDST4_REQUEST (in extdest 4)
Processing flow: BLIT (IBO/LBO mode)

› The moon image and alpha mask data will be copied from external flash to VRAM. (This document doesn’t care the way.)
› BLIT in the 2D GFX core makes frame buffer data. (fill, copy, and blend operation)
› Detailed procedure is described in section “BLIT Sequence”
Processing flow: Commit window (IBO/LBO mode)

- Fetch engine (fetch decode 0) in the composition engine starts fetching the frame buffer
- The frame buffer image output to the display at last
Processing flow: Commit window

› Wait frame generator programable interrupt 0
› Set fetch decode 0 and layer blend 1 shadow registers
   – Fetch decode 0 (1)
     ‒ Set source image AXI Bus address, etc.
   – Layer blend 1 (2)
     ‒ Blend mode: blends primary and secondary stream
› Trigger loading shadow registers
   – EXTDST4_REQUEST (in extdest 4)
Outputting display image sequence
(OTF mode)
Processing flow (OTF mode)

- Initialize GFX system
- Set video interface
- Open display
- Create window

- All are same as IBO mode and thus skipped
Processing flow: BLIT (OTF mode)

- The moon image and alpha mask data will be copied from external flash to VRAM. (This document is not concerned about the method used)
- In OTF mode, instructions for the command sequencer are prepared in BLIT procedure.
- The command sequencer will not be triggered, and hence, no data is transmitted.
Processing flow: Commit window (OTF mode)

- 2D GFX core starts BLIT to the external line buffer
- After completion of storing the last slice of the E-line buffer, the composition engine starts fetching from the external line buffer
Session 3: Detailed description of making display image
Related hardware
Related hardware: Overview

› Hardware used in this example
  - 2D GFX Core (BLIT)
  - 2D GFX Core (CMDSEQ)
  - VRAM
  - Video I/O (COMP)

› Here, we learn the manner in which the CPU (e.g. CM7) and GFX masters interact with each other in BLIT operation.
Related hardware: 2D GFX core - Command sequencer (1/4)

- Command sequencer
  - Unloads CPU’s task by controlling “BLIT Engine” and “Draw Engine”.
  - Write/copy “2D GFX Core” MMIO registers
  - Outputs “Scheduled line” to control LBH of “BLIT Engine”
  - Autonomous programming of GFX2D
    - Fetching command list
    - Writing register configuration
    - Triggering BLIT or Draw Engine
    - Synchronizing the execution to software and hardware
Related hardware: 2D GFX core - Command sequencer (2/4)

› Task buffer: Buffer where a batch of instructions is stored (1)
  - Fetched by the command sequencer
  - Supports up to 8 task buffers
  - Executed in time-multiplex
  - Stored in VRAM (usually)

› General purpose register (GPR) (2)
  - Required for synchronization purpose
  - Accessible only by SW and by the command sequencer itself

› Special purpose register (SPR) (3)
  - Provide value of scheduled line to LBH in BLIT Engine
Related hardware: 2D GFX core - Command sequencer (3/4)

- **Scheduler (1)**
  - Determines which GFX2D operations have to be processed in which order within a single TB and to schedule the execution between TBs

- **Programmer (2)**
  - Programs GFX2D operations
  - Direct write/copy access to GFX2D registers (no AHB bus transfers required)

- **Parallel execution**
  - The scheduler and the programmer work in parallel.
  - The scheduler transfers command list information to the programmer by EXEC command. If the programmer is busy the information will be buffered (3). The scheduler then executes the next command.
Instruction overview

- **EXEC**
  - Execute sub-command lists depending on the value of GPR. Address & size of the command list will be transferred to the programmer. It will be buffered if the programmer is busy.

- **JUMP**
  - Jump to specific address depending on the value of GPR

- **CALC**
  - Initialization/incrementation of specific register with/by the specific value

- **INTR**
  - Trigger an interrupt for the synchronization purpose

- **Programming instructions**
  - **WRITE**
    - Write data to configuration registers
  - **COPY**
    - Copy data between configuration registers

- **Scheduling and programming instructions**
  - **SYNC**
    - Synchronize to GFX2D HW events and states
  - **NOP**
    - Performs a waiting for the specified number of clock cycles.
Detailed sequence BLIT (IBO mode)
Sequence BLIT (IBO mode)

- This section describes the method to make a frame buffer for BLIT.
- Only “fill” and “copy” are described, because the sequence of “blend” and “blend with alpha mask” are similar to “copy”.

Sync signals used in the sequence

- BLIT to CMDSEQ (CmdSeq SYNC)
  - BLIT ShdLoad
    - Notifies loading BLIT shadow registers has been done
  - SyncIdComplete
    - Notifies previous BLIT operation has been done
  - ConstActive
    - Always high. This gives other tasks with higher priority a chance to interrupt this task.

- BLIT to CPU (Interrupt)
  - SyncIdComplete (with BLITENG_CFG_SYNC_ID_REQUEST value changed)
    - Notifies the CPU that BLIT operation has been done.
Sequence BLIT (IBO mode) (1/9)

1. **Fill**
   - Make program
   - Set stop address

2. **Copy**
   - Make program
   - Set stop address

3. **Sync**
   - Make program
   - Set stop address
   - Wait till BLIT finishes

4. **CPU start**
   - Stop address

5. **BLIT start**

6. **Hardware Diagram**
   - VRAM
   - Frame buffer
   - Task buffer
   - Command Sequence (CMDSEQ)
   - CPU
   - Programmer

7. **Scheduler**

8. **Copyright © Infineon Technologies AG 2021. All rights reserved.**

9. **BLIT start**

10. **BLIT end**

---

*Note: The sequence diagram outlines the steps involved in the BLIT process, including making programs, setting stop addresses, and managing the hardware and software components.*
Sequence BLIT (IBO mode) (2/9)

- Fill - Make program
- Fill - Set stop address
- Copy - Make program
- Copy - Set stop address
- Sync - Make program
- Sync - Set stop address
- Sync - Wait till BLIT finishes
- Setup BLIT shadowed registers
- Trigger BLIT
- Sync to BLIT shdLoad
- Clear BLIT shdLoad flag
- Execute "Fill"
- Sync to "ConstActive"
- Execute Fill
- Sync to ConstActive

HW diagram

VRAM
Frame buffer
Task buffer
BLIT
CMDSEQ
Programmer
Scheduler
Buffer
CPU

Copyright © Infineon Technologies AG 2021. All rights reserved.
Sequence BLIT (IBO mode) (3/9)

CPU start
 Fill - Make program
 Fill - Set stop address
 Copy - Make program
 Sync - Set stop address
 Sync - Wait till BLIT finishes
 CPU end

CMDSEQ scheduler start
 Execute “Fill”
 Sync to “ConstActive”

BLIT start
 Setup BLIT shadowed registers
 Trigger BLIT
 Sync to BLIT shdLoad
 Clear BLIT shdLoad flag
 Sync to ConstActive

HW diagram
 VRAM
 Frame buffer
 CMDSEQ
 Programer
 Scheduler
 Buffer
 CPU
 Task buffer
Sequence BLIT (IBO mode) (4/9)

Note: The CPU doesn’t wait HW.
Sequence BLIT (IBO mode) (5/9)

**CPU start**
- Fill
  - Make program
- Fill
  - Set stop address
- Copy
  - Make program
- Copy
  - Set stop address
- Sync
  - Make program
- Sync
  - Set stop address
- Sync
  - Wait till BLIT finishes

**CMDSEQ scheduler start**
- Execute “Fill”
- Sync to “ConstActive”
- Execute “Copy”
- Sync to “ConstActive”

**Programmer**
- Setup BLIT shadowed registers
- Trigger BLIT
- Sync to BLIT shdLoad
- Clear BLIT shdLoad flag
- Setup BLIT shadowed registers
- Trigger BLIT
- Sync to BLIT shdLoad
- Clear BLIT shdLoad flag

**BLIT start**
- Fill start
  (ShdLoad = 1)
- Fill on going
- Fill finished

**HW diagram**
- VRAM
- Frame buffer
- BLIT
  - Programmer
  - Buffer
  - Scheduler
  - CPU
- Task buffer

(1) Note: The CPU doesn’t wait HW.

(2)
Sequence BLIT (IBO mode) (6/9)

CPU start
- Fill - Make program
- Fill - Set stop address
- Copy - Make program
- Copy - Set stop address
- Sync - Make program
- Sync - Set stop address
- Sync - Wait till BLIT finishes
- CPU end

CMDSEQ scheduler start
- Execute "Fill"
- Sync to "ConstActive"
- Execute "Copy"
- Sync to "ConstActive"
- Execute "Sync"
- Sync to "SyncComplete"
- Sync to "ConstActive"

Programmer
- Setup BLIT shadowed registers
- Trigger BLIT
- Sync to BLIT shdLoad
- Clear BLIT shdLoad flag
- Setup BLIT shadowed registers
- Trigger BLIT
- Sync to BLIT shdLoad
- Clear BLIT shdLoad flag
- Prepare sync ID (Set SYNCIDREQUEST)
- Trigger interrupt (Set SYNCIDTRIGGER)

BLIT start
- Fill start (ShdLoad = 1)
- Fill on going
- Fill finished
- Copy start (ShdLoad = 1)
- Copy on going
- Copy finished
- Sync to ConstActive
- Sync to ConstActive
- Sync to ConstActive
- Sync to ConstActive

HW diagram
- VRAM
- Frame buffer
- Buffer
- Scheduler
- CPU
- Task buffer
Sequence BLIT (IBO mode) (6/9)

**Sequence Diagram:**
- **CPU start**
  - Fill: Make program
  - Copy: Make program
  - Sync: Make program
  - Sync: Wait till BLIT finishes
  - Sync: Set stop address
- **CMDSEQ scheduler start**
  - Execute “Fill”
  - Sync to “ConstActive”
- **Programmer**
  - Setup BLIT shadowed registers
  - Trigger BLIT
  - Sync to BLIT shdLoad
  - Clear BLIT shdLoad flag
- **BLIT start**
  - Fill start (ShdLoad = 1)
  - Fill on going
- **Sync**
  - Set stop address
  - Sync to “ConstActive”
  - Sync to “SyncIdComplete”
  - Sync to “ConstActive”
- **Sync**
  - Prepare sync ID (Set SYNCIDREQUEST)
  - Trigger interrupt (Set SYNCIDTRIGGER)
  - Copy start (ShdLoad = 1)
  - Copy on going
  - Copy finished
  - Sync finished

**HW Diagram:**
- VRAM
- Task buffer
- Frame buffer
- BLIT
- CMDSEQ
- Scheduler
- Programmer
- CPU

**Flowchart:**
1. Fill: Make program
2. Copy: Make program
3. Sync: Set stop address
4. CPU start
5. CMDSEQ scheduler start
6. Execute “Fill”
7. Sync to “ConstActive”
8. Setup BLIT shadowed registers
9. Trigger BLIT
10. Sync to BLIT shdLoad
11. Clear BLIT shdLoad flag
12. Fill start (ShdLoad = 1)
13. Fill on going
14. Sync finished
15. Sync to “ConstActive”
16. Sync to “SyncIdComplete”
17. Sync to “ConstActive”
18. Sync to “ConstActive”
19. Prepare sync ID (Set SYNCIDREQUEST)
20. Triggerinterrupt (Set SYNCIDTRIGGER)
21. Copy start (ShdLoad = 1)
22. Copy on going
23. Copy finished
Sequence BLIT (IBO mode) (7/9)

**CPU start**
- Fill: Make program
  - Fill: Set stop address
- Copy: Make program
  - Copy: Set stop address
- Sync: Make program
  - Sync: Set stop address
- Sync: Wait till BLIT finishes

**CMDSEQ scheduler start**
- Execute “Fill”
  - Sync to “ConstActive”
- Execute “Copy”
  - Sync to “ConstActive”
- Execute “Sync”
  - Sync to “SyncIdComplete”
  - Sync to “ConstActive”

**Programmer**
- Setup BLIT shadowed registers
- Trigger BLIT
- Sync to BLIT shdLoad
- Clear BLIT shdLoad flag
- Setup BLIT shadowed registers
- Trigger BLIT
- Sync to BLIT shdLoad
- Clear BLIT shdLoad flag
- Prepare sync ID (Set SYNCIDREQUEST)
- Trigger interrupt (Set SYNCIDTRIGGER)

**BLIT start**
- Fill start (ShdLoad = 1)
  - Fill on going
  - Copy on going
  - Sync to ConstActive

**HW diagram**
- VRAM
- Frame buffer
- Task buffer
- CMDSEQ
- Programmer
- Scheduler
- CPU

**Copy finished**
- Copy finished
  - Copy on going
  - Sync to ConstActive

**Sync to ConstActive**
- Sync to ConstActive

**Setup BLIT shadowed registers**
- Setup BLIT shadowed registers

**Trigger BLIT**
- Trigger BLIT

**Sync to BLIT shdLoad**
- Sync to BLIT shdLoad

**Clear BLIT shdLoad flag**
- Clear BLIT shdLoad flag

**Prepare sync ID (Set SYNCIDREQUEST)**
- Prepare sync ID (Set SYNCIDREQUEST)

**Trigger interrupt (Set SYNCIDTRIGGER)**
- Trigger interrupt (Set SYNCIDTRIGGER)

**Pulses SyncIdComplete**
- Pulses SyncIdComplete

**Sync to SyncIdComplete**
- Sync to SyncIdComplete

**BLIT end**
- BLIT end

**Sync to ConstActive**
- Sync to ConstActive

**Sync to ConstActive**
- Sync to ConstActive

**Sync to ConstActive**
- Sync to ConstActive

**Sync to ConstActive**
- Sync to ConstActive

**Sync to ConstActive**
- Sync to ConstActive

**Sync to ConstActive**
- Sync to ConstActive

**Sync to ConstActive**
- Sync to ConstActive
Sequence BLIT (IBO mode) (9/9)

**CPU start**
- Fill
  - Make program
  - Set stop address
- Copy
  - Make program
  - Set stop address
- Sync
  - Make program
  - Set stop address
- Sync
  - Wait till BLIT finishes

**CMDSEQ scheduler start**
- Execute "Fill"
- Sync to "ConstActive"
- Execute "Copy"
- Sync to "ConstActive"
- Execute "Sync"
- Sync to "SyncdComplete"
- Sync to "ConstActive"

**Programmer**
- Setup BLIT shadowed registers
- Trigger BLIT
- Sync to BLIT shdLoad
- Clear BLIT shdLoad flag
- Setup BLIT shadowed registers
- Trigger BLIT
- Sync to BLIT shdLoad
- Clear BLIT shdLoad flag
- Prepare sync ID (Set SYNCIDREQUEST)
- Trigger interrupt (Set SYNCIDTRIGGER)
- Pulses SyncdComplete

**BLIT start**
- Fill start (ShdLoad = 1)
- Fill on going
- Fill finished
- Copy start (ShdLoad = 1)
- Copy on going
- Copy finished
- BLIT end

**HW diagram**
- VRAM
- Frame buffer
- CMDSEQ
  - Programmer
  - Buffer
  - Scheduler
- Task buffer
- CPU

002-27785 ‘B, 2021-9-20
Copyright © Infineon Technologies AG 2021. All rights reserved.
Detailed sequence BLIT (LBO mode)
Sequence BLIT (LBO mode)

- This section describes the method to make a frame buffer for BLIT in LBO mode.
- Contents are the same as that of IBO mode.

Scheduled line (SL in following slides)
- This indicates the last rendering completed line number in the frame buffer, after the current BLIT storing is completed.
Sequence BLIT (LBO mode) (1/10)

1. CPU start
   - Initialize
     - Make program
   - Fill
     - Make program
   - Copy
     - Make program
   - Store
     - Make program
   - Sync
     - Make program
     - Set stop address
     - Wait till BLIT finishes

2. Stop address

3. BLIT start
   - Make program
   - BLIT
     - I-line Buffer
   - CMDSEQ
   - CPU
   - Task buffer

4. HW diagram
   - VRAM
   - Frame buffer

5. CPU end

6. Iteration
   - Make program
Sequence BLIT (LBO mode) (2/10)

**Initialization**
- Make program

**Fill**
- Make program

**Copy**
- Make program

**Store**
- Make program

**Iteration**
- Make program

**Sync**
- Make program
  - Set stop address
  - Wait till BLIT finishes

**CPU start**

**BLIT start**

**HW diagram**

- **VRAM**
- **Frame buffer**
- **Task buffer**

**CPU**

**CMDSEQ**

**Iteration**
- Make program

**Sync**
- Make program
  - Set stop address

**BLIT start**

- **BLIT**
  - **1-line Buffer**

**CMDSEQ scheduler start**

**Programmer**

- **SL = 0**
  - Execute "init" (task id, dimension)

- **SL += 8 (slice height)**
  - Execute "Fill" (set, trigger, sync)

  - Execute "Copy" if SL is in image range (set, trigger, sync)

  - Execute "Store" (set, trigger, sync)

  - Execute "Sync" (request, trigger)

- **Sync to ConstActive**

- **Jump if SL < Frame Height**

- **Sync to ConstActive**

- **Sync to SyncIdComplete**

- **Sync to ConstActive**

**CMDSEQ scheduler end**

**CPU start**

**CPU end**

Copyright © Infineon Technologies AG 2021. All rights reserved.
Sequence BLIT (LBO mode) (3/10)

- **CPU start**
  - Initialize (Make program)
  - Fill (Make program)
  - Copy (Make program)
  - Store (Make program)
  - Iteration (Make program)
  - Sync (Make program) - Set stop address - Wait till BLIT finishes

- **CMDSEQ scheduler start**
  - SL = 0
  - Execute "init"
  - Execute "Fill"
  - Execute "Copy" if SL is in image range
  - Execute "Store"
  - SL += 8 (slice height)
  - Sync to ConstActive
  - Jump if SL < Frame Height
  - Execute "Sync"
  - Sync to "SyncIdComplete"
  - Sync to "ConstActive"

- **Programmer**
  - Init (task id, dimension)
  - Fill (set, trigger, sync)
  - Copy (set, trigger, sync)
  - Store (set, trigger, sync)
  - Sync (request, trigger)

- **BLIT start**
  - Fill

- **VRAM**
  - VRAM
  - Frame buffer

- **HW diagram**
  - BLIT
  - I-line Buffer
  - (1)
  - Stop address

- **CMDSEQ**
  - CPU
  - Iteration (Make program)

002-27785 ‘B, 2021-9-20

Copyright © Infineon Technologies AG 2021. All rights reserved.
Sequence BLIT (LBO mode) (4/10)

**HW diagram**

- **VRAM**
- **Frame buffer**
- **Task buffer**
- **CMDSEQ**
- **CPU**

**Copy** skipped

**Initialize**
- Make program

**Fill**
- Make program

**Copy**
- Make program

**Store**
- Make program

**Iteration**
- Make program

**Sync**
- Make program
- Set stop address
- Wait till BLIT finishes

**CPU start**

**CMDSEQ scheduler start**
- SL = 0
- Execute "init"
- Execute "Fill"
- Execute "Copy" if SL is in image range
- Execute "Store"
- SL += 8 (slice height)
- Sync to ConstActive
- Jump if SL < Frame Height
- Execute "Sync"
- Sync to "SyncIdComplete"
- Sync to "ConstActive"

**Programmer**
- Init (task id, dimension)
- Fill (set, trigger, sync)
- Copy (set, trigger, sync)
- Store (set, trigger, sync)
- Sync (request, trigger)

**BLIT start**
- Fill
- Store

**BLIT**
- I-line Buffer

**I-line Buffer**
- BLIT start

**CPU end**

Copyright © Infineon Technologies AG 2021. All rights reserved.
Sequence BLIT (LBO mode) (5/10)

- Initialize: Make program
- Fill: Make program
- Copy: Make program
- Store: Make program
- Iteration: Make program
- Sync: Make program - Set stop address - Wait till BLIT finishes

**CPU start**
- Initialize: Make program
- Fill: Make program
- Copy: Make program
- Store: Make program
- Iteration: Make program
- Sync: Make program - Set stop address - Wait till BLIT finishes

**CPU end**

**CMDSEQ**
- scheduler start
- scheduler end

**Programmer**
- Execute "init"
- Execute "Fill"
- Execute "Copy" if SL is in image range
- Execute "Store"
- SL += 8 (slice height)
- Sync to ConstActive
- Jump if SL < Frame Height
- Execute "Sync"
- Sync to ConstActive
- Sync to SyncIdComplete

**BLIT start**
- Fill
- Store

**BLIT**
- I-line Buffer

**VRAM**
- Frame buffer
- Task buffer

**CMDSEQ**
- CPU

**Stop address**

**Copyright © Infineon Technologies AG 2021. All rights reserved.**
Sequence BLIT (LBO mode) (6/10)

- **CPU start**
  - Initialize
    - Make program
  - Fill
    - Make program
  - Copy
    - Make program
  - Store
    - Make program
  - Iteration
    - Make program
  - Sync
    - Make program
    - Set stop address
    - Wait till BLIT finishes
- **CMDSEQ scheduler start**
  - SL = 0
  - Execute "init"
  - Execute "Fill"
  - Execute "Copy" if SL is in image range
  - Execute "Store"
  - SL += 8 (slice height)
  - Sync to ConstActive
  - Jump if SL < Frame Height
  - Execute "Sync"
  - Sync to ConstActive
  - Sync to SyncIdComplete
- **Programmer**
  - Init (task id, dimension)
  - Fill (set, trigger, sync)
  - Copy (set, trigger, sync)
  - Store (set, trigger, sync)
  - Sync (request, trigger)
- **BLIT start**
  - Fill
  - Store
- **BLIT start**
  - Fill
  - Copy
  - CMDSEQ
  - CPU
  - VRAM
  - Frame buffer
  - Task buffer
- **CPU start**
  - BLIT start
  - Copy
  - CMDSEQ
  - CPU
  - VRAM
  - Frame buffer
  - Task buffer
- **VRAM**
  - Copy
  - CMDSEQ
  - CPU
  - VRAM
  - Frame buffer
  - Task buffer
Sequence BLIT (LBO mode) (7/10)

**CPU start**
- Initialize
  - Make program
- Fill
  - Make program
- Copy
  - Make program
- Store
  - Make program
- Iteration
  - Make program
- Sync
  - Make program
  - Set stop address
  - Wait till BLIT finishes

**BLIT start**
- Fill
- Store
- Copy
- Fill

**Programmer**
- Init (task id, dimension)
- Copy (set, trigger, sync)
- Store (set, trigger, sync)
- Sync (request, trigger)
- Execute
- Sync to ConstActive
- SL += 8 (slice height)
- Sync to ConstActive
- Jump if SL < Frame Height
- Execute “Sync”
- Sync to "SyncIdComplete"
- Sync to “ConstActive”

**CMDSEQ**
- scheduler start
- scheduler end
- Sync to ConstActive
- SL += 8 (slice height)
- Programmer
- Sync to SyncIdComplete
- Fill (set, trigger, sync)

**VRAM**
- Frame buffer
- Task buffer

**HW diagram**
- BLIT
  - I-line Buffer
- (1)
- (2)
- (1)
- (2)
- Stop address
Sequence BLIT (LBO mode) (8/10)

**CPU start**
- Initialize
  - Make program
- Fill
  - Make program
- Copy
  - Make program
- Store
  - Make program
- Iteration
  - Make program

**Sync**
- Make program
- Set stop address
- Wait till BLIT finishes

**CPU end**

**BLIT start**
- Fill
- Store
- Copy
- Store

**Sync**
- Make program
- Set stop address
- Wait till BLIT finishes

**Store**
- Make program

**Fill**
- Make program

**Iteration**
- Make program

**CMDSEQ**
- scheduler start
  - Init
    - task id, dimension
  - Fill
    - set, trigger, sync
  - Copy
    - set, trigger, sync
  - Store
    - set, trigger, sync
  - Sync to ConstActive
  - SL += 8 (slice height)
  - Sync to ConstActive
  - SL = 0
  - Execute "Sync"
  - Sync to "SyncIdComplete"
  - Sync to "ConstActive"
  - Jump if SL < Frame Height

**CMDSEQ scheduler end**

**VRAM**

**Frame buffer**

**Task buffer**

**CPU**

**HW diagram**

- VRAM
- Frame buffer
- Task buffer
- CMDSEQ
- CPU

**Stop address**

---

Copyright © Infineon Technologies AG 2021. All rights reserved.
Sequence BLIT (LBO mode) (9/10)

**CPU start**
- Initialize
  - Make program
- Fill
  - Make program
- Copy
  - Make program
- Store
  - Make program
- Iteration
  - Make program
- Sync
  - Make program
  - Set stop address
  - Wait till BLIT finishes

**CPU end**

**CMDSEQ scheduler start**
- SL = 0
  - Execute "init"
  - Execute "Fill"
  - Execute "Copy" if SL is in image range
  - Execute "Store"
  - SL += 8 (slice height)
  - Sync to ConstActive
  - Jump if SL < Frame Height
  - Execute "Sync"
  - Sync to "SyncIdComplete"
  - Sync to "ConstActive"
- CMDSEQ scheduler snd

**Programmer**
- Init (task id, dimension)
- Fill (set, trigger, sync)
- Copy (set, trigger, sync)
- Store (set, trigger, sync)
- Sync (request, trigger)

**BLIT start**
- Fill
- Store
- Copy
- Store

**Sync to ConstActive**

**BLIT end**

**SyncIdComplete** signal after the operation finished

**Trigger interrupt so the BLIT pulses**

**VRAM**

**Frame buffer**

**(1)** (2)

**Pulses**

**(1)**

Copyright © Infineon Technologies AG 2021. All rights reserved.
Sequence BLIT (LBO mode) (10/10)

**CPU start**
- Initialize
  - Make program

**Fill**
- Make program

**Copy**
- Make program

**Store**
- Make program

**Iteration**
- Make program

**Sync**
- Make program
- Set stop address
- Wait till BLIT finishes

**CPU end**

**CMDSEQ scheduler start**
- SL = 0
  - Execute "init"
  - Execute "Fill"
  - Execute "Copy"
    if SL is in image range
  - Execute "Store"
  - Sync to ConstActive
  - Jump if
    SL < Frame Height

**Programmer**
- Execute "sync"
- Sync to ConstActive
- Sync to SyncIdComplete

**BLIT start**
- Fill
  - Store
  - Copy

**VRAM**
- Frame buffer
- Task buffer

**HW diagram**
- BLIT
  - I-line Buffer
  - (1)
- CPU
  - CMDSEQ
  - (2)
- Stop address
  - Pulses
  - SyncIdComplete
  - BLIT end

Copyright © Infineon Technologies AG 2021. All rights reserved.
Detailed sequence BLIT (OTF mode)
External line buffer and LBH control
External line buffer in VRAM

- Line buffer in VRAM is called E (external)-line buffer in this document.
- In the OTF mode, we use the E-line buffer instead of frame buffer.
- Usually, the E-Line buffer size is smaller than the frame buffer. E-Line buffer does not need to have whole frame image area; instead it contains some lines of the frame image. (e.g. E-line buffer size is (128 lines * width * bpp) byte while frame buffer size is (height * width * bpp) byte.)
External line buffer in VRAM

- E-line buffer is actually a ring buffer where FetchLine and StoreLine are chasing.

- FetchLine: A line being fetched by one of fetch engines in the composition engine

- StoreLine: A line being stored by the BLIT
External line buffer in VRAM

› Same as ordinal ring buffer, StoreLine should not overtake FetchLine (overflow). FetchLine should not overtake StoreLine (underflow) as well.

› Line buffer handshake controller (LBH Control) works so that overflow and underflow don’t happen.

› To avoid the overflow and the underflow, “LBH Control” is introduced.
External line buffer in VRAM: LBH control

› The LBH control
  – Intermediates among store engine in the BLIT engine, fetch engines (only fetch decode0 in this example) in composition engine and command sequencer.

› Communication between the LBH control and the “fetch decode0” (1)
  – FetchStopLine
    – Indicates line at which the “fetch decode0” should stop fetching to avoid buffer underflow
  – FetchLine
    – Indicates line the “fetch decode0” is now fetching from. This is used to derive StoreStopLine.
External line buffer in VRAM: LBH control

- Communication between the LBH control and the “store” (1)
  - StoreStopLine
    - Indicates line at which the “store” should be stop to storing to avoid buffer overflow
  - StoreLine
    - Indicates line the “store” is now storing into. This is use to derive FetchStopLine.

- Communication between the LBH control and command sequencer (2)
  - LBH ready
    - Indicates there are enough lines in the E-Line buffer to save a slice of the image. Command sequencer waits for this signal to get high before blitting the next slice in order to not to stall BlitEng.
  - Scheduled line
    - Indicates the value which the StoreLine will be after the current store operation is finished. This is used to derive LBH ready signal. This is a special purpose register value managed by the command sequencer.
LBH proxy

- LBH link is the connection and the synchronization between one store and one fetch units.
- There are (currently) 5 LBH-capable fetches in the composition engine (CompEng) and therefore 5 LBH links, but only one store unit exists in BlitEng.
- "StoreStopLine" is multiplexed to the store unit.
- A task in the command sequencer has to set LBH so that the LBH provides the "store" "StoreStopLine" of the target fetch engine at the first place.
Sequence BLIT (OTF mode)

- This section describes the method to make a frame buffer for BLIT in OTF mode.
- Contents are the same as that of IBO/LBO mode.

Scheduled line (SL in the following slides)
- This indicates the last rendering completed line number in the frame buffer, after the current BLIT storing was finished.
- Command sequencer uses a special purpose register to manage this value, because LBH Control will derive the LBHReady signal from this register value.
Sequence BLIT (OTF mode) (1/9)

**HW diagram**

- **VRAM**
  - E-Line buffer
  - Task buffer
- **CPU**
- **CMDSEQ**
- **BLIT**
  - I-line Buffer
- **COMP**

**Programmer**

- **CPU start**
  - Initialize and Sync
    - Make program
  - Fill
    - Make program
  - Copy
    - Make program
  - Sync CPU
    - Make program
  - Store
    - Make program
  - Iteration
    - Make program
  - Trigger CmdSeq
    - Set stop address
  - Wait
    - E-Line Buffer Full
  - Trigger Composition Engine
- **BLIT start**
  - stop address
  - CMDSEQ scheduler start
  - **CPU end**

**Notes**

- Copyright © Infineon Technologies AG 2021. All rights reserved.
- Sequence BLIT (OTF mode) (1/9)
Sequence BLIT (OTF mode) (2/9)

- Initialize and Sync - Make program
- Copy - Make program
- Sync CPU - Make program
- Store - Make program
- Iteration - Make program
- Trigger CmdSeq - Set stop address
- Wait E-line buffer full
- Trigger Composition engine

CPU start

1. Initial steps:
   - SL = 0
   - Sync to LBHReady4
   - Execute "Init"
   - Fill (set, trigger, sync)
   - Execute "Copy" if SL is in image range
   - Execute "Sync CPU" if SL = E-Line buffer Height - 8
   - Execute "Store"

2. Conditional jumps:
   - SL += 8 (slice height)
   - Sync to ConstActive
   - Jump if SL < Frame Height
   - Jump

BLIT start

HW diagram

- COMP
- BLIT
- VRAM
- E-line buffer
- Task buffer
- CPU

Copyright © Infineon Technologies AG 2021. All rights reserved.
Sequence BLIT (OTF mode) (3/9)

**CPU start**
- Initialize and Sync (Make program)
- Fill (Make program)
- Copy (Make program)
- Sync CPU (Make program)
- Store (Make program)
- Iteration (Make program)
- Trigger CmdSeq (Set stop address)
- Wait (E-line buffer full)
- Trigger Composition engine

**CMDSEQ scheduler start**
- \( SL = 0 \)
- Sync to LBHReady4
- Execute “Init”
  - (task id, dimension)
- Execute “Fill”
- Execute “Copy” if \( SL \) is in image range
- Execute “Sync CPU” if \( SL = \) E-Line buffer Height - 8
- Execute “Store”
- \( SL += 8 \) (slice height)
- Sync to ConstActive
- Jump if \( SL < \) Frame Height
- Jump

**Programmer**

**BLIT start**

**HW diagram**
- \( \text{COM} \)
- \( \text{LBH fetch decode0} \)
- \( \text{BLIT} \)
- \( \text{I-line Buffer} \)
- \( \text{CMDSEQ} \)
- \( \text{CPU} \)
- \( \text{VRAM} \)
- E-line buffer
- Task buffer

Copyright © Infineon Technologies AG 2021. All rights reserved.
Sequence BLIT (OTF mode) (6/9)

**CPU start**
- Initialize and Sync - Make program
- Fill - Make program
- Copy - Make program
- Sync CPU - Make program
- Store - Make program
- Iteration - Make program
- Trigger CmdSeq - Set stop address
- Wait - E-line buffer full
- Trigger - Composition engine

**CMDSEQ scheduler start**
- SL = 0
- Sync to LBHReady4
- Execute “Init” (task id, dimension)
- Execute “Fill”
- Execute “Copy” (set, trigger, sync)
- Execute “Sync CPU” (request, trigger)
- Execute “Store” (set, trigger, sync)
- SL += 8 (slice height)
- Sync to ConstActive
- Jump if SL < Frame Height
- Jump

**BLIT start**
- Fill
- Store

**HW diagram**
- COMP
- BLIT
- I-line buffer
- E-line buffer
- Task buffer
- VRAM

**Stop address**

Copyright © Infineon Technologies AG 2021. All rights reserved.
Sequence BLIT (OTF mode) (6/9)

- **CPU start**
  - Initialize and Sync
  - Make program
  - Fill
  - Make program
  - Copy
  - Make program
  - Sync CPU
  - Make program
  - Store
  - Make program
  - Iteration
  - Make program
  - Trigger CmdSeq
  - Set stop address
  - Wait
  - E-line buffer full
  - Trigger Composition engine

- **CMDSEQ scheduler start**
  - SL = 0
  - Sync to LBHReady4
  - Execute "Init"
  - Execute "Fill"
  - Execute "Copy" if SL is in image range
  - Execute "Sync CPU" if SL = E-Line buffer Height - 8
  - Execute "Store"
  - SL += 8 (slice height)
  - Sync to ConstActive
  - Jump if SL < Frame Height

- **BLIT start**
  - Fill
  - Store

- **HW diagram**

  - **VRAM**
  - **E-line buffer**
  - **BLIT I-line buffer**
  - **CMDSEQ**
  - **CPU**
  - **Composition engine**
  - **Task buffer**

- **CMDSEQ scheduler end**

**Notes**
- **Sync CPU** (request, trigger)
- **Copy** (set, trigger, sync)
- **Sync to LBHReady4**
- **Sync CPU** (set, trigger, sync)
- **Sync to ConstActive**
- **Jump**
- **Jump if SL < Frame Height**
- **Jump**

**Copyright © Infineon Technologies AG 2021. All rights reserved.**
Sequence BLIT (OTF mode) (8/9)

- CPU start
  - Initialize and Sync
  - Make program
  - Fill
  - Make program
  - Copy
  - Make program
  - Sync CPU
  - Make program
  - Store
  - Make program
  - Iteration
  - Make program
  - Trigger CmdSeq
  - Set stop address
  - Wait
  - E-line buffer full
  - Trigger
  - Composition engine
  - CPU end

- CMDSEQ scheduler start
  - SL = 0

- Programmer
  - Sync to LBHReady4
  - Execute "Fill"
  - Copy
  - Sync CPU
  - if SL is in image range
  - Execute "Copy"
  - Store
  - Sync to ConstActive
  - Jump if
  - SL < Frame Height
  - Jump

- BLIT start
  - Fill
  - Store

- HW diagram
  - COMP
  - LBH fetch decode0
  - E-line buffer
  - Task buffer
  - VRAM

- Stop address

002-27785 *B, 2021-9-20
Copyright © Infineon Technologies AG 2021. All rights reserved.
Sequence BLIT (OTF mode) (9/9)

**CPU start**
- Initialize and Sync
  - Make program
- Fill
  - Make program
- Copy
  - Make program
- Sync CPU
  - Make program
- Store
  - Make program
- Iteration
  - Make program
- Trigger CmdSeq
  - Set stop address

**CMDSEQ scheduler start**
- SL = 0
- Sync to LBHReady4
- Execute "Fill"
- Execute "Copy" if SL is in image range
- Execute "Sync CPU" if SL = E-Line buffer Height - 8
- Execute "Store"
- SL += 8 (slice height)
- Sync to ConstActive
- Jump if SL < Frame Height
- Jump

**Programmer**

**BLIT start**
- Fill
- Store
- Fill
- Copy
- Store
- Fill
- Copy
- Store
- Pulses SyncIdComplete

**HW diagram**

**VRAM**

**BLIT**
- I-line buffer
- E-line buffer

**CMDSEQ**

**CPU**

**Stop address**

Copyright © Infineon Technologies AG 2021. All rights reserved.
Appendix
Related hardware: VRAM (2/3)

Memory structure (an example)
- Eight memory banks, each with 64-bit (8 byte) data bus
- 4-Mbyte address space

Note: These numbers (8 logical banks, 4-Mbyte address space) may depend on the TRAVEO™ T2G device. See the specific device datasheet to check the numbers.
Related hardware: VRAM (3/3)

- Bank interleaving
  - Data to sequential addresses will be written to the 8 logical SRAM banks in sequential order.
  - Each bank stores a 64-bit word, thus the bank selection is done based on the address bits [5:3].
## Basic BLIT example: BLIT operations and supporting feature

<table>
<thead>
<tr>
<th>Feature</th>
<th>IBO mode</th>
<th>LBO mode</th>
<th>OTF mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>YUV422 format source</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ROP2/3</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Filter</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Arbitrary warping</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Perspective transformation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Color lookup table</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Drawing engine content</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Rotation</td>
<td>✓</td>
<td>✓ (90° steps)</td>
<td>✓ (90° steps)</td>
</tr>
<tr>
<td>Scaling</td>
<td>✓</td>
<td>✓ (limited quality)</td>
<td>✓ (limited quality)</td>
</tr>
<tr>
<td>Scaling of compressed image</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Movement</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Blending</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Basic BLIT example: BLIT operations and supporting features

› Example of ROP
  – Logical calculation

\[
\begin{array}{cccc}
1 & 1 & 1 & 2 \\
1 & 1 & 1 & 2 \\
3 & 3 & 2 & 2 \\
3 & 3 & 2 & 2 \\
\end{array}
\]

\[
\begin{array}{cccc}
1 & 2 & 2 & 3 \\
2 & 2 & 3 & 3 \\
4 & 3 & 4 & 3 \\
5 & 4 & 3 & 2 \\
\end{array}
\]

\[
\begin{array}{cccc}
2 & 3 & 3 & 5 \\
3 & 3 & 4 & 5 \\
7 & 6 & 6 & 5 \\
8 & 7 & 5 & 4 \\
\end{array}
\]

\[a + b \rightarrow c\]

› Example of filter
  – e.g. Blurring

![Example of filter](image-url)
Basic BLIT example: BLIT operations and supporting features

› Example of arbitrary warping
  – Move pixel color at any position to any position

› Example of perspective transformation
  – Effect like tilting 2D image
Basic BLIT example: BLIT operations and supporting features

› Example of color lookup table
  – Express image with at least 256 colors

› Example of drawing engine content
  – Make shapes, fonts.

[Image of lavender flowers]

[Diagram showing a rectangle, an arrow pointing downwards, and an ellipse with the word 'FONT']
Basic BLIT example: BLIT operations and supporting features

› Example of rotation

› Example of scale
Basic BLIT example: OTF mode restrictions

- Frame buffer (line buffer) width has to be equal to window width
- Frame buffer (Line buffer) height is minimum two slices (16 lines) and maximum (2 * window height - 1)
- Frame buffer (Line buffer) for OTF window must have a power of 2-line count
- Sync driver functions are not available for OTF windows
- Width maximum 1600, height maximum 800
- OTF window must be inside the display dimension.
- OTF window cannot have CAPTURE attribute
- KEEPLINES of OTF window must be set for warping feature
- Frame buffer (line buffer) properties cannot be changed in the OTF window
- Set matrix is not supported by OTF window
- Only one OTF window on multi-layer fetch is available
Part of your life. Part of tomorrow.
<table>
<thead>
<tr>
<th>Revision</th>
<th>ECN</th>
<th>Submission Date</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>**</td>
<td>6638974</td>
<td>2019-07-29</td>
<td>Initial release.</td>
</tr>
<tr>
<td>*A</td>
<td>7020960</td>
<td>2020-10-20</td>
<td>Updated slides 2, 5.</td>
</tr>
<tr>
<td>*B</td>
<td>7305314</td>
<td>2021-09-20</td>
<td>Updated slides 3 ~ 140.</td>
</tr>
</tbody>
</table>