Zephyr® Project
Developer Summit

Graphics Support In Zephyr RTOS

Daniel DeGrasse, *NXP*

ABSTRACT

- Intro to 2D Graphics
- Current Graphics Support in Zephyr
- Overview of Graphics Hardware
- Zephyr’s Use Cases
- Existing APIs
- Zephyr API Proposal
- Future Work
- Open Discussion

VECTOR VS RASTER GRAPHICS

Vector graphics

- Mathematical description of image
- Best for logos, engravings, fonts, UI elements

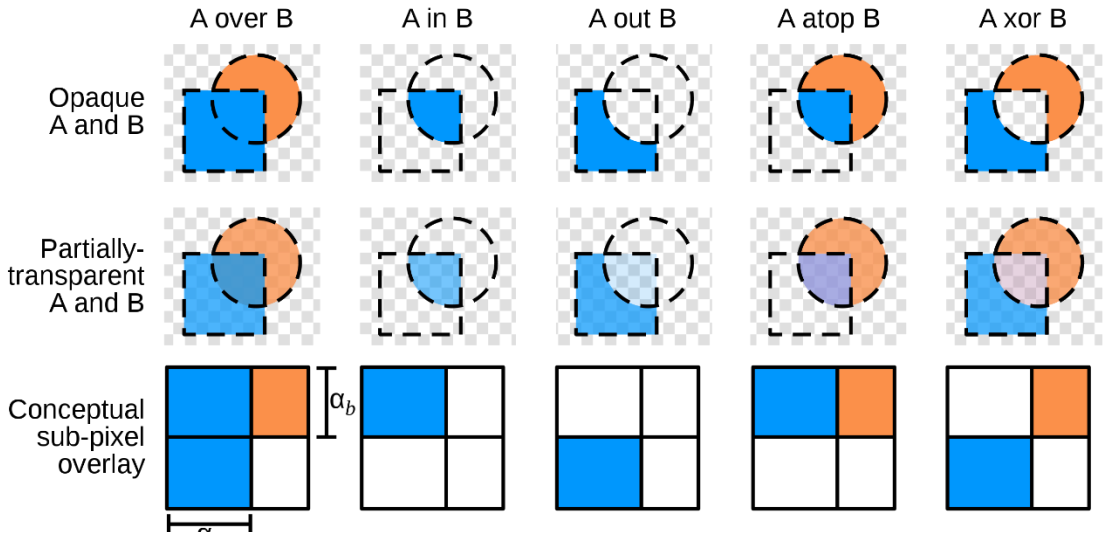
Zephyr

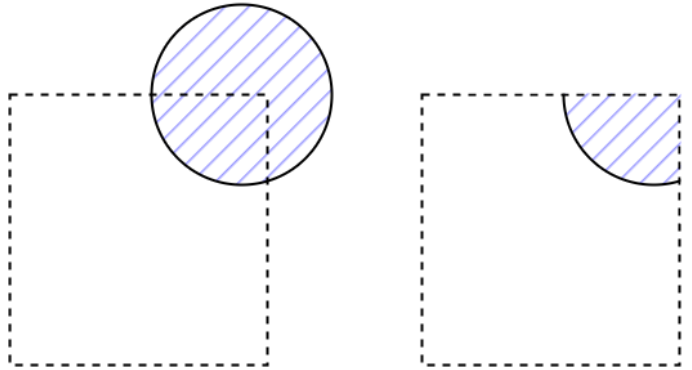
Raster Graphics

- 2D pixel array
- Best for photographs or other images

BLENDING AND CLIPPING

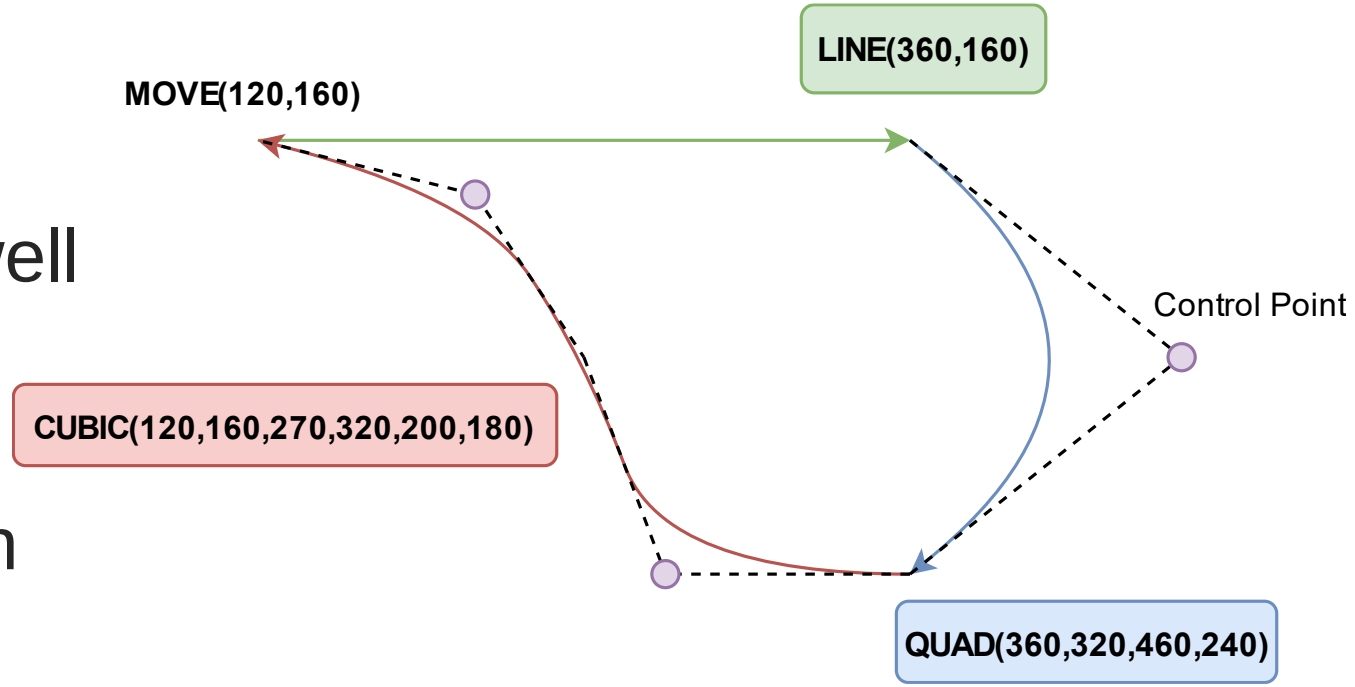
- Blending
  - Porter-duff blending modes
  - Primarily applies when one shape has alpha component
  - Can also be used for more complicated clipping operations
- Clipping
  - Any output shape can be clipped
  - Boundaries are typically just a rectangle, versus blending two shapes together





PATHS

- Path commands allow drawing complex shapes
- Used in formats like SVG, as well as 2D vector graphics engines
- Rendering can be done within path, or only on pixels within an odd number of paths



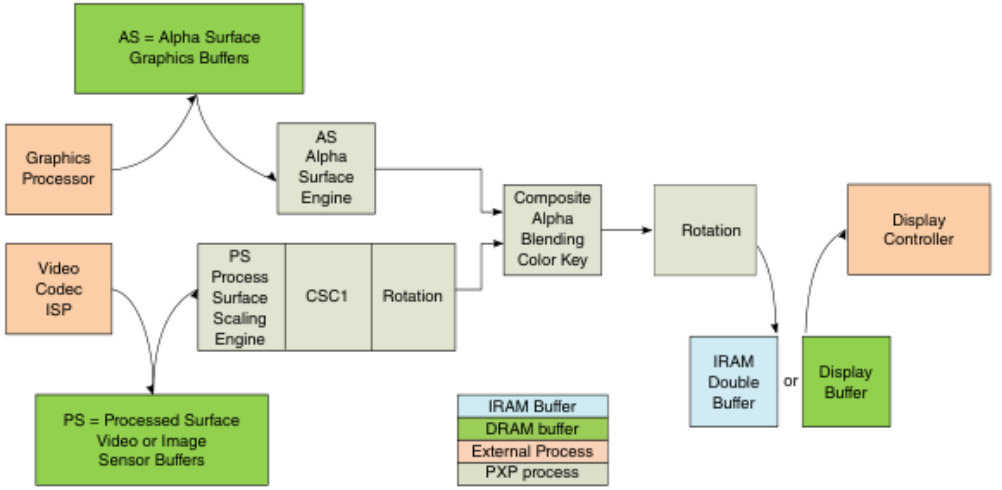
CURRENT STATE OF ZEPHYR GRAPHICS SUPPORT

- CFB subsystem
  - Fonts can be rasterized, and rendered onto a monochrome display
  - Basic shape drawing (rectangles and points) also supported
  - Monochrome only

- LVGL
  - Upstream has support for several graphical acceleration engines, but Zephyr does not enable this
  - Rich set of APIs for graphics accelerators to implement
  - All drawing is done in software when using Zephyr

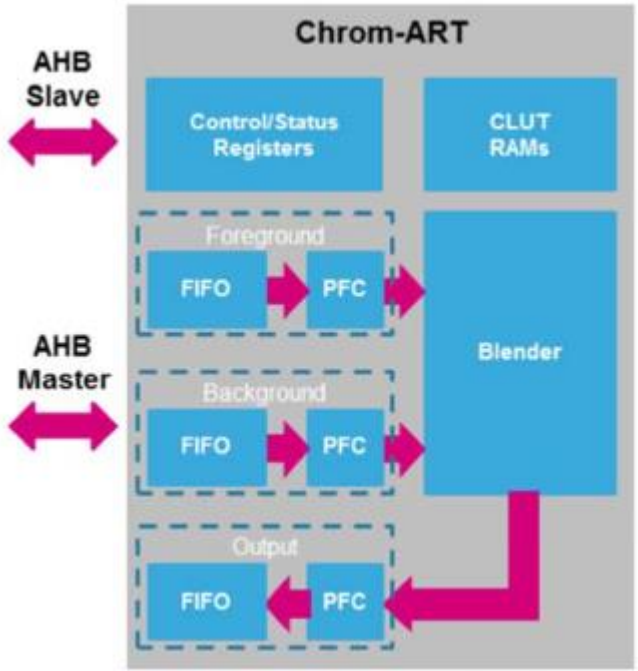
PXP AND SMARTDMA

- NXP PXP
  - Multiple blending modes
  - Blends alpha surface onto background (process surface)
  - 90/180/270-degree rotation
  - Color space conversion
  - Process surface scaling
  - Can output directly to the LCD controller without an intermediate buffer
- NXP SmartDMA
  - Pixel format conversion
  - Direct output to display via MIPI-DSI or FlexIO attached display
  - 90/180/270-degree rotation



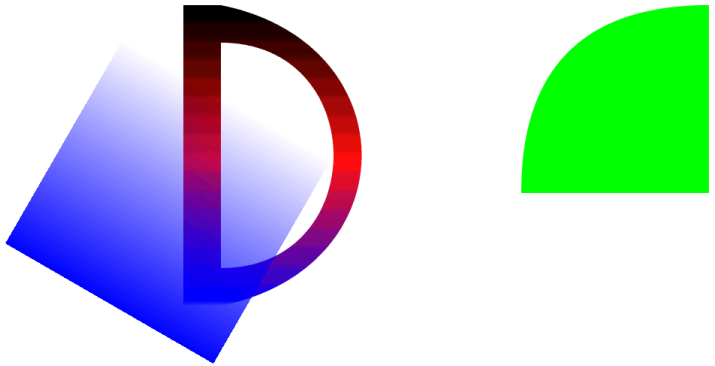
DMA2D

- Fast clear- set buffer to solid color
- Blends foreground surface onto background
- Color space conversions
- Color lookup table to limit input framebuffer size



GC355/GCNANOLITE-V

- Based on Vivante GPU IP
- Supported by VGLite API
- Blending
- Vector based paths
- Gradients and solid fill
- Can clip output image within a path
- Transformation matrix for arbitrary rotation, scaling, and translation
- Path clipping



ZEPHYR USE CASES

- Basic vector graphics
  - Rectangles, ellipses, arcs
- Arbitrary vector paths
  - Includes arbitrary scaling and transformation
- Basic raster graphics
  - Scaling, 90/180/270-degree rotation
  - Blending
  - Clipping
- Advanced raster graphics
  - Clipping raster within arbitrary path
  - Arbitrary scaling and transform
- Direct output to display
  - Graphics engine transforms buffer and writes output directly to display

VGLITE API

- Vector Graphics
  - Path drawing based on draw commands (similar to SVG format)
  - Gradient and solid color fill
  - Rotate or scale path using transformation matrix
- Raster Graphics
  - Clip image within path based on draw commands
  - Rotate or scale image using transformation matrix
- Subset of Porter-duff blending modes
- Designed by Verisilicon for use with 2D GPUs like those on NXP RT1170 and RT595
- API lacks “simple” graphics functions
  - Not realizable on hardware like NXP PXP or ST DMA 2D
- No support for writing directly to a display output

LVGL BACKEND

- Vector Graphics
  - Rectangle, arc, letter, line, and polygon draw functions
  - No support for paths like those present in VGLite
- Raster Graphics
  - Draw image with output coordinates, angle, and zoom (arbitrary transformation)
- Subset of porter-duff blend modes supported
- Direct output to display could be implemented within display flush callback
- No support for arbitrary paths like those in VGLite
- Can be supported by all listed hardware engines

ARM2D

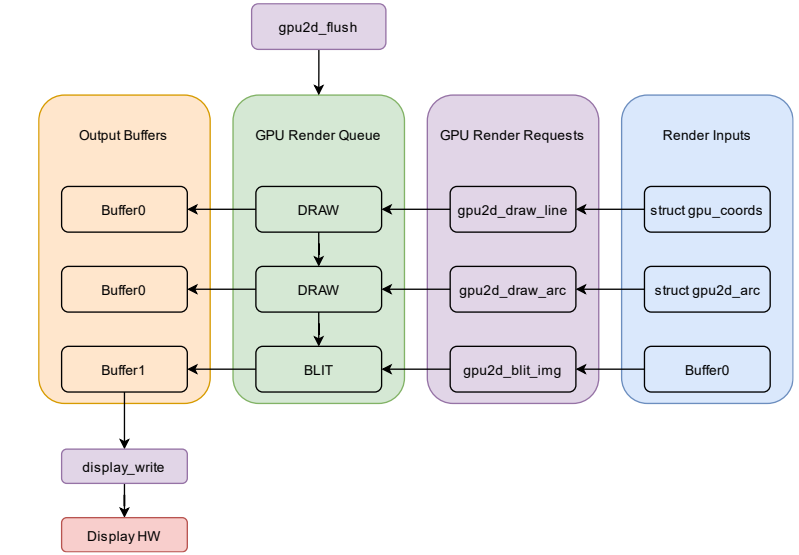
- Optimized software backend for Cortex-M MCUs
- Supports adding custom GPU backends
- Draw operations
  - Draw rectangle
- Raster operations
  - Copy image into output buffer, with alpha blending
  - Transform image (rotation and scaling)
  - Color space conversion
- Drawing APIs are limited, no direct output to display
- API can be implemented as a software backend

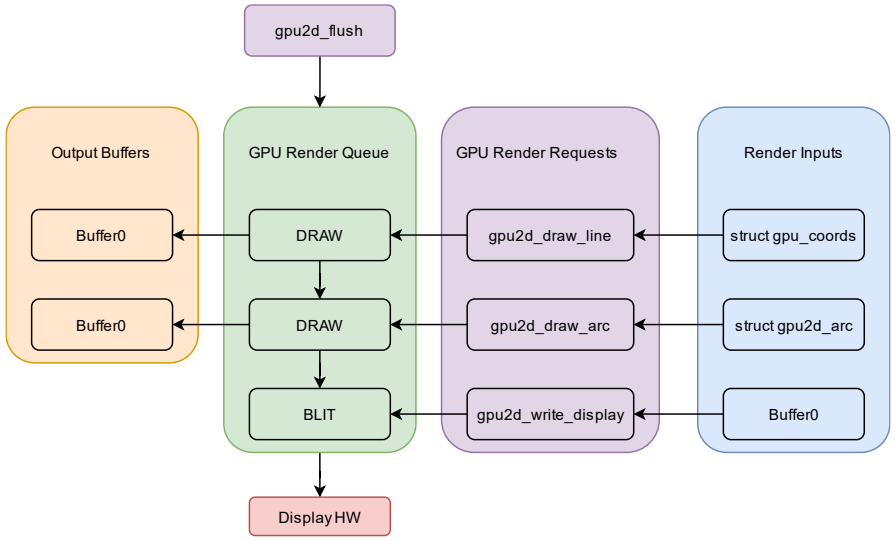
API PROPOSAL

API	Description	Hardware Engines
gpu2d_draw_rect	Draw basic rectangle	PXP, DMA2D, GC355, GCNanoLiteV
gpu2d_draw_line	Draw line	GC355, GCNanoLiteV
gpu2d_draw_arc	Draw arc, or full circle	GC355, GCNanoLiteV
gpu2d_draw_path	Draw path using lines, cubic and quadratic curves	GC355, GCNanoLiteV
gpu2d_blit_path	Copy source buffer onto path described by lines, cubic and quadratic curves	GC355, GCNanoLiteV
gpu2d_blit_img	Copy source buffer onto output buffer with optional 90/180/270-degree rotation	PXP, DMA2D, GC355, GCNanoLiteV
gpu2d_flush	Run all queued GPU operations	All
gpu2d_write_display	Write buffer directly to the display, with optional 90/180/270-degree rotation	SMARTDMA, PXP

API CONCEPTS

- Render operations queue
  - Allows for multiple operations to be queued
  - Flush must be called before next frame can be rendered
- Direct to display rendering
  - Same operations queue, but no need to write final buffer to display



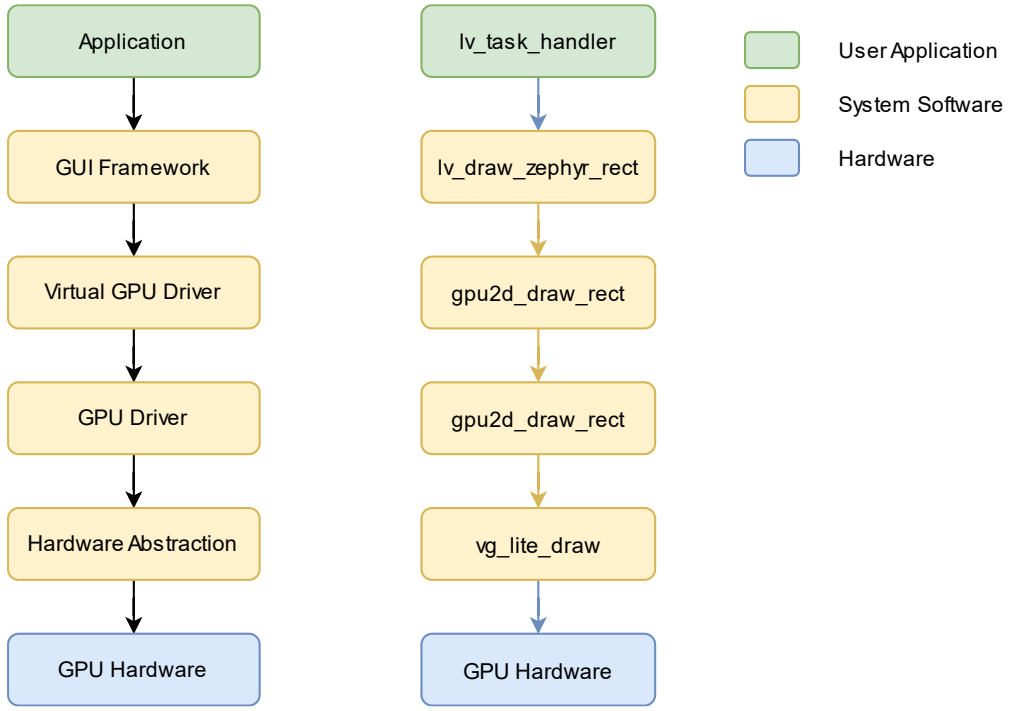


API CONCEPTS

- Virtual GPU driver can wrap multiple GPU devices

```
gpu-shared: gpu-node {
    blit-engine = <&gc355>;
    vector-engine = <&gc355>;
    output-engine = <&pxp>; /* Used with gpu2d_write_display */
};
```

- Operations queued by graphics framework will be flushed before display render occurs



ALTERNATIVES

- LVGL
  - Already implemented for many hardware blocks, just needs to be ported to Zephyr
  - Proven abstraction layer
  - Locks Zephyr into LVGL, other frameworks like QT cannot be implemented
  - Does not expose all drawing capabilities of more complex 2D GPUs like GC355/GCNanoLiteV
- VGLite
  - Well defined API, already has conformance tests and implementations
  - Managed by Verisilicon, who controls definition
  - No APIs for engines like DMA2D or PXP

FUTURE WORK

- LVGL Module Updates
- RFC for Graphics API
- API Implementations

Zephyr® Project

Developer Summit

Backup Slides

SECURE CONNECTIONS
FOR A SMARTER WORLD

PATH APIS

Opcode	Purpose
MOVE	Move cursor to coordinate
LINE	Draw line from cursor to coordinate, update cursor
QUAD	Draw quadratic curve from cursor to coordinate with one control point, update cursor.
CUBIC	Draw cubic curve from cursor to coordinate with two control points, update cursor.
ARC_CLOCKWISE	Draw clockwise arc from cursor to coordinate, with a given sweep angle and horizontal/vertical radius
ARC_COUNTER_CLOCKWISE	Draw counter clockwise arc from cursor to coordinate, with a given sweep angle and horizontal/vertical radius

API STRUCTURES

- gpu2d_buf
  - buffer data
  - height, width, stride
  - pixel format
- gpu2d_src_data
  - color or gradient definition
- gpu2d_matrix
  - 3x2 transformation matrix
- gpu2d_arc
  - center point, inner/outer radius, start/end angle
- gpu2d_rect
  - rectangle dimensions

DIRECT 2D

- Microsoft 2D API, utilizes GPU acceleration where possible
- Vector Graphics
  - Rectangle drawing, as well as rounded rectangles
  - Drawing ellipses
  - Text drawing
  - Arbitrary paths using draw commands like VGLite
  - Rotation and scaling via transformation matrix
- Raster Graphics
  - Majority of porter-duff blend modes
  - Can scale a bitmap to fit within a rectangle then copy to output buffer

G2D

- Designed for engines like the PXP
- Raster Graphics
  - Copy image into output buffer, with set location
  - 90/180/270-degree rotation
  - Horizontal and vertical flip
  - Subset of porter-duff blending modes
  - Upscale/downscale

LINUX DRM

- Abstracts command queue for all GPUs
- Userspace is responsible for submitting requests to the graphics execution manager
- Mesa uses graphics execution manager requests to implement APIs like OpenGL and Vulkan
- DRM uses opcodes to send commands to GPU, vendors can add custom opcodes
- Would make applications like GPGPU easier with Zephyr, but we would likely use dedicated hardware for those functions on MCUs
- Potentially more abstraction than we want on smaller MCUs