# VkAttachmentDescription for VkRenderPass

```
typedef struct VkAttachmentDescription {
    VkAttachmentDescriptionFlags flags;// usually 0
    VkFormat                     format;
    VkSampleCountFlagBits        samples;
    VkAttachmentLoadOp           loadOp;
    VkAttachmentStoreOp          storeOp;
    VkAttachmentLoadOp           stencilLoadOp;
    VkAttachmentStoreOp          stencilStoreOp;
    VkImageLayout                initialLayout;
    VkImageLayout                finalLayout;
} VkAttachmentDescription;
```

```
typedef enum VkFormat {
    ...
    VK_FORMAT_R32_UINT = 98,
    VK_FORMAT_R32_SINT = 99,
    VK_FORMAT_R32_SFLOAT = 100,
    VK_FORMAT_R32G32_UINT = 101,
    VK_FORMAT_R32G32_SINT = 102,
    VK_FORMAT_R32G32_SFLOAT = 103,
    VK_FORMAT_R32G32B32_UINT = 104,
    VK_FORMAT_R32G32B32_SINT = 105,
    VK_FORMAT_R32G32B32_SFLOAT = 106,
    VK_FORMAT_R32G32B32A32_UINT = 107,
    VK_FORMAT_R32G32B32A32_SINT = 108,
    VK_FORMAT_R32G32B32A32_SFLOAT = 109,
    ...
} VkFormat;
```

```
typedef enum VkSampleCountFlagBits {
    VK_SAMPLE_COUNT_1_BIT = 0x00000001,
    VK_SAMPLE_COUNT_2_BIT = 0x00000002,
    VK_SAMPLE_COUNT_4_BIT = 0x00000004,
    VK_SAMPLE_COUNT_8_BIT = 0x00000008,
    VK_SAMPLE_COUNT_16_BIT = 0x00000010,
    VK_SAMPLE_COUNT_32_BIT = 0x00000020,
    VK_SAMPLE_COUNT_64_BIT = 0x00000040,
} VkSampleCountFlagBits;
```

```
typedef enum VkAttachmentLoadOp {
    VK_ATTACHMENT_LOAD_OP_LOAD = 0,
    VK_ATTACHMENT_LOAD_OP_CLEAR = 1,
    VK_ATTACHMENT_LOAD_OP_DONT_CARE = 2,
  // Provided by VK_EXT_load_store_op_none
    VK_ATTACHMENT_LOAD_OP_NONE_EXT = 1000400000,
} VkAttachmentLoadOp;
```

```
typedef enum VkAttachmentStoreOp {
    VK_ATTACHMENT_STORE_OP_STORE = 0,
    VK_ATTACHMENT_STORE_OP_DONT_CARE = 1,
  // Provided by VK_VERSION_1_3
    VK_ATTACHMENT_STORE_OP_NONE = 1000301000,
  // Provided by VK_KHR_dynamic_rendering
    VK_ATTACHMENT_STORE_OP_NONE_KHR = VK_ATTACHMENT_STORE_OP_NONE,
  // Provided by VK_QCOM_render_pass_store_ops
    VK_ATTACHMENT_STORE_OP_NONE_QCOM = VK_ATTACHMENT_STORE_OP_NONE,
  // Provided by VK_EXT_load_store_op_none
    VK_ATTACHMENT_STORE_OP_NONE_EXT = VK_ATTACHMENT_STORE_OP_NONE,
} VkAttachmentStoreOp;
```

```
typedef enum VkImageLayout {
    VK_IMAGE_LAYOUT_UNDEFINED = 0,
    VK_IMAGE_LAYOUT_GENERAL = 1,
    VK_IMAGE_LAYOUT_COLOR_ATTACHMENT_OPTIMAL = 2,
    VK_IMAGE_LAYOUT_DEPTH_STENCIL_ATTACHMENT_OPTIMAL = 3,
    VK_IMAGE_LAYOUT_DEPTH_STENCIL_READ_ONLY_OPTIMAL = 4,
    VK_IMAGE_LAYOUT_SHADER_READ_ONLY_OPTIMAL = 5,
    VK_IMAGE_LAYOUT_TRANSFER_SRC_OPTIMAL = 6,
    VK_IMAGE_LAYOUT_TRANSFER_DST_OPTIMAL = 7,
    VK_IMAGE_LAYOUT_PREINITIALIZED = 8,
    ...
} VkImageLayout;
```

# VkRenderPass

```
VkRenderPassCreateInfo
sType = VK_STRUCTURE_TYPE_RENDER_PASS_CREATE_INFO;
pNext = nullptr;
flags = 0;
attachmentCount;
pAttachments;
subpassCount;
pSubpasses;
dependencyCount;
pDependencies;
```

VkAttachmentDescription for VkRenderPass

VkSubpassDescription for VkRenderPass

VkSubpassDependency for VkRenderPass

```
VkResult vkCreateRenderPass(
    VkDevice                    device,
    const VkRenderPassCreateInfo* pCreateInfo,
    const VkAllocationCallbacks*  pAllocator,
    VkRenderPass*               pRenderPass
);
```

VkDevice

VkRenderPass

```
void vkDestroyRenderPass(
    VkDevice                   device,
    VkRenderPass               renderPass,
    const VkAllocationCallbacks* pAllocator);
```

VkDevice

VkRenderPass