Tina Linux NPU VIPLite API 说明

版本号	日期	制/修订人	内容描述
1.0	2022.09.14	AWA1911	初版，npu 函数 API 说明

vip_init()	**描述:** 初始化 VIP 硬件和 VIPLite 软件环境。详细地说，该 API 重置和初始化 VIP 硬件处于就绪状态以接受命令。它还初始化软件环境，例如 如视频内存堆、电源管理和 MMU 表。 在应用程序调用任何其他 VIPLite API 以使用 VIP 硬件之前调用此 API。之后应用程序完成后，调用 vip_destroy（）。 您可以多次调用 vip_init（）API。但是，vip_destroy（）调用的数量应该匹配 vip_init（）调用的数量。只有第一个 vip_init（）调用和最后一个 vip-destroy（）执行调用。之间的其他 vip_init（）和 vip_destroy（）调用不会触发初始化或销毁应用。 **Syntax:** vip_status_e **vip_init**(void); **Parameters:** None **Returns:** vip_status_e
vip_destroy()	**描述:** 终止 VIPLite 驱动程序，释放 vip_init（）请求的资源，并关闭 vip hardware 在应用程序完成后调用此 API。执行此 API 后，在任何其他 API 之前调用 vip_init（） VIPLite API。 您可以多次调用 vip_init（）API。但是，vip_destroy（）调用的数量应该匹配 vip_init（）调用的数量。只有第一个 vip_init（）调用和最后一个 vip-destroy（） 执行调用。之间的其他 vip_init（）和 vip_destroy（）调用不会触发初始化或销毁应用。 **Syntax:**

	vip_status_e **vip_destroy**(void); **Parameters:** None **Returns:** vip_status_e
vip_create_network()	**描述:** 从给定的二进制文件创建网络。二进制是由生成的二进制大对象（BLOB）数据图是一个二进制生成器。VIPLite 驱动程序可以解释它以创建网络对象。 **Syntax:** vip_status_e **vip_create_network**(void *data, vip_uint32_t size_of_data, vip_enum type, vip_network *network); **Parameters:** IN *data The pointer to the graph binary. IN size_of_data The size in bytes of the graph binary. IN type The network type. The supported types are defined in the vip_create_network_type_e enumeration. OUT *network The pointer to receive the created network object if the network is created successfully. If the network creation fails, VIP_NULL is returned. **Returns:** vip_status_e
vip_destroy_network()	**描述:** 销毁网络。该 API 释放分配给网络的所有相关资源。为指定网络执行此 API 后，网络的命令缓冲区将从指定的网络也被释放。然而，复制网络的系数为保留。 **Syntax:**

	vip_status_e **vip_destroy_network**(vip_network network); **Parameters:** IN network An opaque handle to the network to be destroyed. **Returns:** vip_status_e
vip_query_network()	**描述:** 查询网络的属性。 **Syntax:** vip_status_e **vip_query_network**(vip_network network, vip_enum property, void *value); **Parameters:** IN network An opaque handle to the network to be queried. IN property The network property to be queried. The following properties are available for query: VIP_NETWORK_PROP_LAYER_COUNT VIP_NETWORK_PROP_INPUT_COUNT VIP_NETWORK_PROP_OUTPUT_COUNT VIP_NETWORK_PROP_NETWORK_NAME VIP_NETWORK_PROP_READ_REG_IRQ VIP_NETWORK_PROP_ADDRESS_INFO VIP_NETWORK_PROP_MEMORY_POOL_SIZE VIP_NETWORK_PROP_PROFILING For details, see Section *2.2.5, vip_network_property_e*. OUT *value A pointer in memory to store the returned property value. The data type of the value varies according to the property queried. **Returns:**
vip_set_network()	**描述:**

	配置网络。在运行网络之前，需要通过调用 vip_prepare_network(). **Syntax:** vip_status_e **vip_set_network**(vip_network network, vip_enum property, void *value); **Parameters:** IN network An opaque handle to the network to be configured. IN property The network property to be configured. The supported properties are: VIP_NETWORK_PROP_CHANGE_PPU_PARAM VIP_NETWORK_PROP_SET_MEMORY_POOL VIP_NETWORK_PROP_SET_DEVICE_ID VIP_NETWORK_PROP_SET_PRIORITY For details, see Section *2.2.5, vip_network_property_e*. IN *value A pointer in memory to the property value. **Returns:** vip_status_e
vip_prepare_network ()	**描述:** 验证网络的配置。该 API 将内部存储器资源分配给网络，将所有操作的资源部署到内部内存池，分配并修补命令内部内存池中资源的缓冲区。成功执行此 API 后，网络被认为准备在 VIP 硬件上运行。在此 API 之前，使用 vip_set_network（）API 配置网络。如果调用此 API，则更多在网络配置保持不变的情况下，驱动程序多次默认忽略 API 调用，除了第一个请求。 **Syntax:** vip_status_e **vip_prepare_network**(vip_network network); **Parameters:** IN network An opaque handle to the network to be prepared. **Returns:**

	vip_status_e
vip_query_input()	**描述:** 查询网络输入的属性。 **Syntax:** vip_status_e **vip_query_input**(vip_network network, vip_uint32_t index, vip_enum property, void *value); **Parameters:** IN network An opaque handle to the network to be queried. IN index The index of the network input to be queried. IN property The input buffer property to be queried. The following properties are available for query: VIP_BUFFER_PROP_QUANT_FORMAT VIP_BUFFER_PROP_NUM_OF_DIMENSION VIP_BUFFER_PROP_SIZES_OF_DIMENSION VIP_BUFFER_PROP_DATA_FORMAT VIP_BUFFER_PROP_FIXED_POINT_POS VIP_BUFFER_PROP_TF_SCALE VIP_BUFFER_PROP_TF_ZERO_POINT VIP_BUFFER_PROP_NAME VIP_BUFFER_PROP_DATA_TYPE For details, see Section *2.2.6, vip_buffer_property_e*. OUT *value A pointer in memory to store the returned property value. **Returns:** vip_status_e
vip_query_output()	**描述:** 查询网络输出的属性. **Syntax:** vip_status_e **vip_query_output**(

	vip_network network, vip_uint32_t index, vip_enum property, void *value); **Parameters:** IN network An opaque handle to the network to be queried. IN index The index of the network output to be queried. IN property The output buffer property to be queried. The following properties are available for query: VIP_BUFFER_PROP_QUANT_FORMAT VIP_BUFFER_PROP_NUM_OF_DIMENSION VIP_BUFFER_PROP_SIZES_OF_DIMENSION VIP_BUFFER_PROP_DATA_FORMAT VIP_BUFFER_PROP_FIXED_POINT_POS VIP_BUFFER_PROP_TF_SCALE VIP_BUFFER_PROP_TF_ZERO_POINT VIP_BUFFER_PROP_NAME VIP_BUFFER_PROP_DATA_TYPE For details, see Section *2.2.6, vip_buffer_property_e*. OUT *value A pointer in memory to store the returned property value. **Returns:** vip_status_e
vip_set_input()	**描述:** 将输入缓冲区连接到网络。将输入缓冲区连接到网络时，VIPLite 驱动程序修补网络命令缓冲区以填充输入缓冲区。您还可以调用此 API 来更新输入缓冲区。更新将从下一个网络生效处决在将输入缓冲区连接到网络之前，请确保使用 vip_prepare_network（）API。在使用 vip_run_network（）运行网络之前，请确保每个有效的网络输入附加有缓冲器。否则，返回 VIP_ERROR_MISSING_INPUT_OUTPUT 调用 vip_run_network（）API。 **Syntax:** vip_status_e **vip_set_input**(

	vip_network network, vip_uint32_t index, vip_buffer input); **Parameters:** IN network An opaque handle to the network to be configured. IN index The index of the network input to be configured. IN input An opaque handle to the buffer to be attached to the network input. **Returns:** vip_status_e
vip_set_output()	**描述:** 将输出缓冲区连接到网络。将输出缓冲区连接到网络时，VIPLite 驱动程序修补网络命令缓冲区以填充输出缓冲区。您还可以调用此 API 来更新输出缓冲区。更新将从下一个网络生效执行在将输出缓冲区连接到网络之前，请确保使用 vip_prepare_network（）API。在使用 vip_run_network（）运行网络之前，请确保已连接网络输出具有缓冲器。否则，返回 VIP_ERROR_MISSING_INPUT_OUTPUT 调用 vip_run_network（）API。 **Syntax:** vip_status_e **vip_set_output**(vip_network network, vip_uint32_t index, vip_buffer output); **Parameters:** IN network An opaque handle to the network to be configured. IN index The index of the network output to be configured. IN

	output An opaque handle to the buffer to be attached to the network output. **Returns:** vip_status_e
vip_run_network()	**描述:** 为网络提交执行任务。VIP 硬件执行最高优先级的任务在已提交的任务中。您可以多次调用此 API。 VIP 硬件完成执行后返回 API 执行状态。如果你需要要在不等待执行完成的情况下立即返回状态，请使用 vip_trigger_network（）API。要设置网络优先级，请使用 vip_set_network（）API。在运行网络之前，请确保使用 vip_prepare_network（）准备网络。在里面此外，通过使用确保每个网络输入和网络输出都带有缓冲区 set_ input（）和 vip_。否则，VIP_ERROR_MISSING_INPUT_OUTPUT 为调用 vip_run_network（）后返回。 要在一个组中运行多个网络，请使用 vip_run_group（）或 vip_ trigger_group（）。 **Syntax:** vip_status_e **vip_run_network**(vip_network network); **Parameters:** IN network An opaque handle to the network to be run. **Returns:** vip_status_e
vip_trigger_network()	**描述:** 为网络提交执行任务。VIP 硬件执行最高优先级的任务在已提交的任务中。您可以多次调用此 API。 API 执行状态立即返回，无需等待硬件完成处决要获取状态，请调用 vip_wait_network（）进行同步。如果您需要状态要在 VIP 硬件完成执行后返回，请使用 VIP_run_network（）API。要设置网络优先级，请使用 vip_set_network（）API。在运行网络之前，请确保使用 vip_prepare_network（）准备网络。在里面此外，通过使用确保每个网络输入和网络输出都带有缓冲区 set_ input（）和 vip_。否则，VIP_ERROR_MISSING_INPUT_OUTPUT 为调用 vip_trigger_network（）后返回。 要在一个组中运行多个网络，请使用 vip_run_group（）或 vip_ trigger_group（）。 **Syntax:** vip_status_e **vip_trigger_network**(vip_network network); **Parameters:** IN network

	An opaque handle to the network to be executed. **Returns:** vip_status_e
vip_wait_network()	**描述:** 等待 VIP 硬件完成指定网络的推断。 调用 vip_trigger_network（）后调用此 API。 **Syntax:** vip_status_e **vip_wait_network**(vip_network network); **Parameters:** IN network An opaque handle to the network. **Returns:** vip_status_e
vip_finish_network()	**描述:** 释放准备好的网络的资源。调用此 API 后，所有内部内存资源分配给网络的数据在网络未被破坏的情况下被释放。如果不再需要网络，使用 vip_destroy_network（）API 销毁它。仅当网络不再使用时，调用 vip_finish_network（）API 来完成准备好的网络或者剩余的系统资源对于其他网络是有限的。如果仍然需要网络，请不要调用此 API，因为准备网络非常耗时。在为准备好的网络成功执行 vip_finish_network（）调用后，重复调用包括静默忽略，直到使用 vip_prepare_network（）API 重新准备网络。为了未准备好的网络，vip_finish_network（）调用将被静默忽略。重要提示：不要为正在运行的网络调用 vip_finish_network（）API。 **Syntax:** vip_status_e **vip_finish_network**(vip_network network); **Parameters:** IN network An opaque handle to the network to be finished. **Returns:** vip_status_e
vip_create_buffer()	**描述:** 创建指定大小的 VIP 缓冲区，行、片或批之间没有填充。 **Syntax:** vip_status_e **vip_create_buffer**(vip_buffer_create_params_t *create_param,

	vip_uint32_t size_of_param, vip_buffer *buffer); **Parameters:** IN *create_param The pointer to a vip_buffer_create_params_t structure. IN size_of_param The size of the data structure created by *create_param in bytes. OUT *buffer The pointer to receive the created buffer object if the VIP buffer is created successfully. If the VIP buffer creation fails, VIP_NULL is returned. **Returns:** vip_status_e If VIP_SUCCESS is returned, a VIP buffer is created successfully. If VIP_ERROR_<*error_type*> is returned, no buffer is created.
vip_create_buffer_from_handle()	**描述:** 从句柄创建 VIP 缓冲区，并将句柄关联的物理地址映射到缓冲区。 使用此 API 之前，请启用 VIP MMU。否则，API 返回 VIP_ERROR_FAILURE。 **Syntax:** vip_status_e **vip_create_buffer_from_handle**(vip_buffer_create_params_t *create_param, vip_ptr handle_logical, vip_uint32_t handle_size, vip_buffer *buffer) **Parameters:** IN *create_param A pointer to a vip_buffer_create_params_t structure. IN handle_logical The address of the handle from which the new VIP buffer is to be created. For a non-real-time Linux operating system, specify a logical

	address. The address is allocated by the Linux malloc() function. For a real-time operating system, specify a physical address. **Note:** Address alignment to 64 bytes is recommended. IN handle_size The size of the memory to which the handle points. **Note:** Size alignment to 64 bytes is recommended. OUT *buffer The pointer to receive the created buffer object if the VIP buffer is created successfully. If the VIP buffer creation fails, VIP_NULL is returned. **Returns:** vip_status_e If VIP_SUCCESS is returned, a VIP buffer is created successfully. If VIP_ERROR_<*error_type*> is returned, no buffer is created.
vip_destroy_buffer()	**描述:** 销毁 VIP 缓冲区并释放缓冲区使用的内存。 **Syntax:** vip_status_e **vip_destroy_buffer**(vip_buffer buffer); **Parameters:** IN buffer The opaque handle of the buffer to be destroyed. **Returns:** vip_status_e
vip_map_buffer()	**描述:** 创建指向指定 VIP 缓冲区的指针。应用程序可以使用指针访问缓冲区。 **Syntax:** void ***vip_map_buffer**(vip_buffer buffer); **Parameters:** IN buffer The opaque handle of the buffer for which a pointer is to be created. **Returns:** A pointer to the buffer that applications can use to read or write the buffer data

vip_unmap_buffer()	**描述:** 释放应用程序用于访问 VIP 缓冲区的指针。 **Syntax:** vip_status_e ***vip_unmap_buffer**(vip_buffer buffer); **Parameters:** IN buffer The opaque handle of the buffer whose pointer is to be released. **Returns:** vip_status_e
vip_get_buffer_size()	**描述:** 检索缓冲区的大小（以字节为单位）。 **Syntax:** vip_uint32_t **vip_get_buffer_size**(vip_buffer buffer); **Parameters:** IN buffer The opaque handle of the buffer whose size is requested. **Returns:** vip_uint32_t The buffer size in bytes.
vip_flush_buffer()	**描述:** 刷新或使从 VIP_create_buffer（）创建的 VIP 缓冲区缓存无效，或 vip_create_buffer_from_handle（）API。 在以下情况下调用此 API：如果正在使用的 VIP 缓冲区包含 CPU 缓存，则在调用 vip_run_network（）。 从 vip_wait_network（）或 vip_ run_networks（）返回后，使用此 API 使缓冲区缓存。 **Syntax:** vip_status_e **vip_flush_buffer**(vip_buffer buffer, vip_buffer_operation_type_e type); **Parameters:** IN buffer The opaque handle of the buffer whose cache is to be flushed or

	invalidated. IN type The buffer cache operation to be executed on the buffer. The buffer catch operations are defined in the vip_buffer_operation_type_e enumeration. **Returns:** vip_status_e
Running a Single Network	运行网络的过程详细如下: 1. 调用 vip_init() 初始化 VIPLite 引擎，包括软件和硬件。 该 API 重置硬件以使其准备好使用，并初始化软件资源。它设置了视频. 内存和其他硬件资源，如中断和寄存器内存，供 VIPLite 使用。 2.从文件或存储器读取网络二进制图（NBG）数据。 3.使用 vip_Create_network（）API 创建网络。 该 API 对 NBG 数据执行健全性检查。因此，建议您检查返回的用于验证网络是否已成功生成的错误代码。 4.使用 vip_Query_input（）和 vip_Query_output（）查询输入和输出属性。 建议执行此步骤以避免输入或输出属性不匹配导致的错误。 5. 使用以下 API 创建输入和输出缓冲区：vip_Create_buffer（），vip_create_buffer_from_handle（）、vip_create_buffer_from_physical（）和 vip_create_buffer_from_fd（）。 6.（可选）使用 vip_set_network（）配置网络属性。 7.使用 vip_Prepare_network（）API 准备网络命令缓冲区。 建议您检查此 API 返回的错误代码。这是因为 API 可能会失败，因为由于资源限制，例如内存不足。 8.借助 vip_map_buffer（）API 从输入和输出缓冲区加载数据。 9.使用 vip_set_input（）和 vip_ set_output（）将输入和输出缓冲区连接到网络。 10.使用 vip_Run_network（）或 vip_trigger_networks（）API 运行网络。 如果使用 vip_run_network（），则在 vip 硬件完成处决如果使用 vip_trigger_network（），它将立即返回，而无需等待 vip 硬件完成执行。如果 CPU 工作负载很重，这将优化 CPU 使用。在这种情况下，当 CPU 需要 API 结果时，调用 vip_wait_network（）进行同步。 注意：可以创建多个网络。但是，一次只能运行一个网络。应用程序运行 根据使用 vip_set_network（）API 配置的网络优先级，一个接一个地访问网络。 11.使用 vip_Flush_buffer（）将网络执行结果从 CPU 缓存刷新到输出缓冲区。 12.在 vip_map_buffer（）的帮助下，检查输出缓冲区的网络执行结果。 13.（可选）重复步骤 6 至 12 以多次运行网络。 14.调用 vip_finish_network（）以释放分配给网络的内部内存。 15.调用 vip_destroy_network（）释放分配给网络的所有其他资源。 16.调用 vip_destroy_buffer（）释放分配给输入和输出缓冲区的内存。 17.调用 vip_destroy（）释放 VIPLite 资源并退出。