ESWIN

ENNP Developer Manual

EIC7x Series AI Digital SoC

Engineering Draft / Rev0.94

Jan. 01, 2025

Notice

INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND DOES NOT REPRESENT A COMMITMENT ON THE PART OF ESWIN. UNLESS OTHERWISE SPECIFIED, ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS DOCUMENT ARE PROVIDED “AS IS” WITHOUTH WARRANTIES, GUARANTEES, OR REPRESENTATIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED.

If applicable, Open-Source Software and/or free software licensing notices are available in the product installation.

Change History

Version	Date	Descriptions
Rev 0.6	Feb. 29, 2024	Initial version
Rev 0.92	Oct. 23, 2024	Added composite model-related interfaces to NPU runtime
Rev 0.93	Jan. 21, 2025	Add load model from memory interface to NPU runtime
Rev 0.94	2025/1/24	Chapter 3: AcceleratorKit - Modified the ES_AK_DSP_WarpAffine interface parameters and documentation.

Content

Content of Tables

1. DSP Runtime

1.1 Overview

The ENNP (ESWIN Neural Network Processing) platform is an intelligent heterogeneous acceleration platform for ESWIN media processing, which includes development tools and interfaces for NPU, DSP, HAE, and GPU. DSP (Digital Signal Processor) is a programmable hardware acceleration module under the ENNP platform. Users can develop intelligent solutions based on DSP to accelerate intelligent analytics and lower CPU usage.

1.2 Function Description

1.2.1 Important Conception

- Operator Handle: When loading DSP operators, the system assigns a handle for each operator as a unique ID.
- Task Description: Essential information that users need to fill in when submitting operator tasks, referring to the definition of ES_DSP_TASK_S.
- Task Handle: When invoking DSP to process operator tasks, the system assigns a handle to each task to distinguish different tasks.
- Task Query: After users submit asynchronous tasks using Type 1 interfaces (refer to section 1.3.1.1), they can use ES_DSP_QueryTask to check whether the related task has been completed based on the handle returned by the system.
- Process Report: After user submit asynchronous tasks using Type 2 interfaces (refer to section 1.3.1.2), one need to call ES_DSP_LL_ProcessReport to handle the asynchronous tasks. The interface will invoke the callback function for completed task.

1.2.2 Module Parameters

- fw_timeout: Timeout period for DSP tasks, measured in seconds. Users can configure a reasonable maximum timeout period based on their usage scenarios. Use the following command to config after DSP driver loading:

```
echo 10 > /sys/module/eic7700_dsp/parameters/fw_timeout
```

1.3 API Reference

1.3.1 API Types

DSP provides two different types of APIs. Type 1 interface partially hides the underlying interfaces, making it simpler to use. Type 2 interface exposes more low-level functionalities, suitable for advanced users. Users should not mix up the usage of these two types of interfaces to prevent potential errors during computation, which may lead to internal logic issue within the process. Details are as follows:

1.3.1.1 Type I

- ES_DSP_GetVersion：Get version of the DSP Runtime interface as an U64 number.
- ES_DSP_GetVersionString： Get the string version of the DSP Runtime interface.
- ES_DSP_GetCapability： Get DSP capability information.

- ES_DSP_SetLogLevel：Set the DSP log level.
- ES_DSP_Open：Open the DSP device.
- ES_DSP_Close：Close the DSP device.
- ES_DSP_LoadOperator：Load a DSP operator.
- ES_DSP_UnloadOperator：Unload a DSP operator.
- ES_DSP_SubmitTask：Execute a DSP operator task synchronously.
- ES_DSP_SubmitTaskAsync：Execute a DSP operator task asynchronously.
- ES_DSP_QueryTask：Query whether a DSP operator task is completed.

1.3.1.2 Type II

- ES_DSP_LL_GetVersion：Get version of the DSP Runtime interface as an U64 number.
- ES_DSP_LL_GetVersionString：Get version of the DSP Runtime interface as a string.
- ES_DSP_LL_GetCapability：Get DSP capability information.
- ES_DSP_LL_SetLogLevel：Set the DSP log level.
- ES_DSP_LL_Open：Open the DSP device.
- ES_DSP_LL_Close：Close the DSP device.
- ES_DSP_LL_LoadOperator：Load a DSP operator.
- ES_DSP_LL_UnloadOperator：Unload a DSP operator.
- ES_DSP_LL_PrepareDMABuffer：Allocates a DMA buffer based on operator data.
- ES_DSP_LL_UnprepareDMABuffer：Releases the allocated DMA buffer.
- ES_DSP_LL_SubmitTask：Executes operator tasks synchronously.
- ES_DSP_LL_SubmitTaskAsync：Executes operator tasks asynchronously.
- ES_DSP_LL_ProcessReport：Invokes the callback function registered when submitting asynchronous operator tasks that have been completed.

1.3.2 Runtime API

1.3.2.1 ES_DSP_GetVersion

[Declaration]

ES_S32 ES_DSP_GetVersion(ES_U64 *version)

[Description]

Get the version of the DSP Runtime interface as an U64 number.

[Parameters]

Parameter	Description	Input/Output
Version	DSP version number	Output, see

Parameter	Description	Input/Output
		ES_SDK_VERSION_U

[Return Value]

Return Value	Description
Zero	Success
Non-zero	Failure, see section 1.5 错误!未找到引用源。 error code.

1.3.2.2 ES_DSP_GetVersionString

[Declaration]

ES_S32 ES_DSP_GetVersionString(

ES_CHAR *version,
ES_U32 maxSize)

[Description]

Get DSP string version information.

[Parameters]

Parameter	Description	Input/Output
Version	DSP version number	Output
maxSize	maxSize Maximum length of DSP version string	Input

[Return Value]

Return Value	Description
Zero	Success
Non-zero	Failure, see 1.5 错误!未找到引用源。 error code.

1.3.2.3 ES_DSP_GetCapability

[Declaration]

ES_S32 ES_DSP_GetCapability(ES_DSP_Capability_S *capability)

[Description]

Get DSP capability information.

[Parameters]

Parameter	Description	Input/Output
dspId	DSP capability information	Output, see ES_DSP_Capability_S

[Return Value]

Return Value	Description

Zero	Success
Non-zero	Failure, see 1.5 错误!未找到引用源。 error code.

1.3.2.4 ES_DSP_SetLogLevel

[Declaration]

ES_S32 ES_DSP_SetLogLevel(ES_U32 level)

[Description]

Set the logging level of DSP at runtime.

[Parameters]

Parameter	Description	Input/Output
Level	DSP log level	Input

[Return Value]

Return Value	Description
Zero	Success
Non-zero	Failure, see 1.5 错误!未找到引用源。 error code.

1.3.2.5 ES_DSP_Open

[Declaration]

ES_S32 ES_DSP_Open(

ES_DSP_ID_E dspId,
ES_DSP_LOAD_POLICY_E opLoadPolicy)

[Description]

Open DSP device according to DSP ID number.

[Parameters]

Parameter	Description	Input/Output
dspId	DSP ID number	Input, see ES_DSP_ID_E
opLoadPolicy	DSP operator loading policy	Input, see ES_DSP_LOAD_POLICY_E

[Return Value]

Return Value	Description
Zero	Success
Non-zero	Failure, see 1.5 错误!未找到引用源。 error code.

1.3.2.6 ES_DSP_Close

[Declaration]

ES_S32 ES_DSP_Close(ES_DSP_ID_E dspId)

[Description]

Close DSP device according to DSP ID number.

[Parameters]

Parameter	Description	Input/Output
dspId	DSP ID number	Input, see ES_DSP_ID_E

[Return Value]

Return Value	Description
Zero	Success
Non-zero	Failure, see 1.5 错误!未找到引用源。 error code.

1.3.2.7 ES_DSP_LoadOperator

[Declaration]

```
ES_S32 ES_DSP_LoadOperator(
        ES_DSP_ID_E dspId,
        const ES_CHAR* operatorName,
        const ES_CHAR* operatorLibDir,
        ES_DSP_HANDLE* operatorHandle)
```

[Description]

Load operator with specified name and library path to the specified DSP device, and return the operator handle.

[Parameters]

Parameter	Description	Input/Output
dspId	DSP ID number	Input, see ES_DSP_ID_E
operatorName	DSP operator name	Input
operatorLibDir	DSP operator library directory	Input
operatorHandle	DSP operator handle	Output

[Return Value]

Return Value	Description
Zero	Success
Non-zero	Failure, see 1.5 错误!未找到引用源。 error code.

1.3.2.8 ES_DSP_UnloadOperator

[Declaration]

```
ES_S32 ES_DSP_UnloadOperator(
                    ES_DSP_ID_E dspId,
                    const ES_DSP_HANDLE operatorHandle)
```

[Description]

Unload the operator from the specified DSP device using the loaded operator handle.

[Parameters]

Parameter	Description	Input/Output
dspId	DSP ID number	Input
operatorHandle	DSP operator handle	Input

[Return Value]

Return Value	Description
Zero	Success
Non-zero	Failure, see 1.5 错误!未找到引用源。 error code.

1.3.2.9 ES_DSP_SubmitTask

[Declaration]

```
ES_S32 ES_DSP_SubmitTask(
                    ES_DSP_ID_E dspId,
                    ES_DSP_TASK_S* operatorTask)
```

[Description]

Synchronously submit operator task to the specified DSP device, the interface waits for DSP to complete the computation, and returning means the task is completed.

[Parameters]

Parameter	Description	Input/Output
dspId	DSP ID number	Input
operatorTask	DSP operator task message body	Input, see ES_DSP_TASK_S

[Return Value]

Return Value	Description
Zero	Success
Non-zero	Failure, see 1.5 错误!未找到引用源。 error code.

1.3.2.10 ES_DSP_SubmitTaskAsync

[Declaration]

```
ES_S32 ES_DSP_SubmitTaskAsync(
                    ES_DSP_ID_E dspId,
                    ES_DSP_TASK_S* operatorTask,
                    ES_DSP_HANDLE* taskHandle)
```

[Description]

Asynchronously submit operator task to the specified DSP device, and return the operator task handle. ES_DSP_QueryTask needs to be used to query whether the task is completed.

[Parameters]

Parameter	Description	Input/Output
dspId	DSP ID number	Input
operatorTask	DSP operator task message body	Input, see ES_DSP_TASK_S
taskHandle	DSP operator task handle	Output

[Return Value]

Return Value	Description
Zero	Success
Non-zero	Failure, see 1.5 错误!未找到引用源。 error code.

1.3.2.11 ES_DSP_QueryTask

[Declaration]

```
ES_S32 ES_DSP_QueryTask(
                    ES_DSP_ID_E dspId,
                    ES_DSP_HANDLE taskHandle,
                    ES_BOOL block,
                    ES_BOOL* taskFinish)
```

[Description]

Query whether the operator on the specified DSP device is completed based on the operator task handle.

[Parameters]

Parameter	Description	Input/Output
dspId	DSP ID number	Input
taskHandle	DSP operator task handle	Input
block	Whether to block the query	Input
taskFinish	DSP operator task completion status pointer	Output

[Return Value]

Return Value	Description
Zero	Success
Non-zero	Failure, see 1.5 错误!未找到引用源。 error code.

1.3.3 Runtime Low-level API

1.3.3.1 ES_DSP_LL_GetVersion

[Declaration]

ES_S32 ES_DSP_LL_GetVersion(ES_U64 *version)

[Description]

Get DSP digital version information.

[Parameters]

Parameter	Description	Input/Output
version	DSP version number	Output, see ES_SDK_VERSION_U

[Return Value]

Return Value	Description
Zero	Success
Non-zero	Failure, see 1.5 错误!未找到引用源。 error code.

1.3.3.2 ES_DSP_LL_GetVersionString

[Declaration]

```
ES_S32 ES_DSP_LL_GetVersionString(
                ES_CHAR* version,
                ES_U32 maxSize)
```

[Description]

Get DSP string version information.

[Parameters]

Parameter	Description	Input/Output
version	DSP version number	Output
maxSize	Maximum length of DSP version string	Input

[Return Value]

Return Value	Description
Zero	Success
Non-zero	Failure, see 1.5 错误!未找到引用源。 error code.

1.3.3.3 ES_DSP_LL_GetCapability

[Declaration]

ES_S32 ES_DSP_LL_GetCapability(ES_DSP_Capability_S *capability)

[Description]

Get DSP capability information.

[Parameters]

Parameter	Description	Input/Output
dspId	DSP capability information	Output, see ES_DSP_Capability_S

[Return Value]

Return Value	Description
Zero	Success
Non-zero	Failure, see 1.5 错误!未找到引用源。 error code.

1.3.3.4 ES_DSP_LL_SetLogLevel

[Declaration]

ES_S32 ES_DSP_LL_SetLogLevel(ES_U32 level)

[Description]

Set the logging level of DSP at runtime.

[Parameters]

Parameter	Description	Input/Output
level	DSP log level	Input

[Return Value]

Return Value	Description
Zero	Success
Non-zero	Failure, see 1.5 错误!未找到引用源。 error code.

1.3.3.5 ES_DSP_LL_Open

[Declaration]

```
ES_S32 ES_DSP_LL_Open(
            ES_DSP_ID_E dspId,
            ES_S32 *dspFd)
```

[Description]

Open DSP device according to DSP ID number.

[Parameters]

Parameter	Description	Input/Output
dspId	DSP ID number	Input, see ES_DSP_ID_E
dspFd	DSP device descriptor	Output

[Return Value]

Return Value	Description
Zero	Success
Non-zero	Failure, see 1.5 错误!未找到引用源。 error code.

1.3.3.6 ES_DSP_LL_Close

[Declaration]

ES_S32 ES_DSP_LL_Close(ES_S32 dspFd)

[Description]

Close DSP device according to DSP device descriptor.

[Parameters]

Parameter	Description	Input/Output
dspFd	DSP device descriptor	Input

[Return Value]

Return Value	Description
Zero	Success
Non-zero	Failure, see 1.5 错误!未找到引用源。 error code.

1.3.3.7 ES_DSP_LL_LoadOperator

[Declaration]

```
ES_S32 ES_DSP_LL_LoadOperator(
            ES_S32 dspFd,
            const ES_CHAR *operatorName,
            const ES_CHAR *operatorLibDir,
            ES_DSP_HANDLE *operatorHandle)
```

[Description]

Load operator with specified name and library path to the specified DSP device, and return the operator handle.

[Parameters]

Parameter	Description	Input/Output
dspFd	DSP device descriptor	Input

Parameter	Description	Input/Output
operatorName	DSP operator name	Input
operatorLibDir	DSP operator library directory	Input
operatorHandle	DSP operator handle	Output

[Return Value]

Return Value	Description
Zero	Success
Non-zero	Failure, see 1.5 错误!未找到引用源。 error code.

1.3.3.8 ES_DSP_LL_UnloadOperator

[Declaration]

```
ES_S32 ES_DSP_LL_UnloadOperator(
            ES_S32 dspFd,
            const ES_DSP_HANDLE operatorHandle)
```

[Description]

Unload the operator from the specified DSP device using the loaded operator handle.

[Parameters]

Parameter	Description	Input/Output
dspFd	DSP device descriptor	Input
operatorHandle	DSP operator handle	Input

[Return Value]

Return Value	Description
Zero	Success
Non-zero	Failure, see 1.5 错误!未找到引用源。 error code.

1.3.3.9 ES_DSP_LL_PrepareDMABuffer

[Declaration]

```
ES_S32 ES_DSP_LL_PrepareDMABuffer(
            ES_S32 dspFd,
            ES_DEV_BUF_S buffer)
```

[Description]

Prepares the SMMU mapping on the specified DSP device in advance for the DMA Buffer required by the operator task. This can accelerate the subsequent data execution process.

[Parameters]

Parameter	Description	Input/Output
dspFd	DSP device descriptor	Input
buffer	DMA Buffer	Input

[Return Value]

Return Value	Description
Zero	Success
Non-zero	Failure, see 1.5 错误!未找到引用源。 error code.

1.3.3.10 ES_DSP_LL_UnprepareDMABuffer

[Declaration]

```
ES_S32 ES_DSP_LL_UnprepareDMABuffer(
                ES_S32 dspFd,
                ES_U64 fd)
```

[Description]

Releases the device SMMU mapping on the DSP corresponding to the DMA Buffer allocated by the fd. This function should be used in pairs with ES_DSP_LL_PrepareDMABuffer and called when the buffer is no longer needed to execute tasks on the DSP device.

[Parameters]

Parameter	Description	Input/Output
dspFd	DSP device descriptor	Input
fd	DMA Buffer Description	Input

[Return Value]

Return Value	Description
Zero	Success
Non-zero	Failure, see 1.5 错误!未找到引用源。 error code.

1.3.3.11 ES_DSP_LL_SubmitTask

[Declaration]

```
ES_S32 ES_DSP_LL_SubmitTask(
                ES_S32 dspFd,
                ES_DSP_TASK_S *operatorTask)
```

[Description]

Submits a synchronous operator task to the specified DSP device.

[Parameters]

Parameter	Description	Input/Output
dspFd	DSP device descriptor	Input
operatorTask	DSP operator task message body	Input, see ES_DSP_TASK_S

[Return Value]

Return Value	Description
Zero	Success
Non-zero	Failure, see 1.5 错误!未找到引用源。 error code.

1.3.3.12 ES_DSP_LL_SubmitTaskAsync

[Declaration]

```
ES_S32 ES_DSP_LL_SubmitTaskAsync(
            ES_S32 dspFd,
            ES_DSP_TASK_S *operatorTask)
```

[Description]

Submits an asynchronous operator task to the specified DSP device. Asynchronous tasks need to be queried and waited for completion using ES_DSP_LL_ProcessReport, and the callback function should be invoked upon completion.

[Parameters]

Parameter	Description	Input/Output
dspFd	DSP device descriptor	Input
operatorTask	DSP operator task message body	Input, see ES_DSP_TASK_S

[Return Value]

Return Value	Description
Zero	Success
Non-zero	Failure, see 1.5 错误!未找到引用源。 error code.

1.3.3.13 ES_DSP_LL_ProcessReport

[Declaration]

```
ES_S32 ES_DSP_LL_ProcessReport(
            ES_S32 dspFd,
            ES_S32 timeout)
```

[Description]

Executes the callback function registered during asynchronous operator task submission.

[Parameters]

Parameter	Description	Input/Output
dspFd	DSP device descriptor	Input
timeout	Unit：ms Maximum time to wait for task completion when no tasks are completed, -1 blocks until a task is completed	Input

[Return Value]

Return Value	Description
Zero	Success
Non-zero	Failure, see 1.5 **错误!未找到引用源。** error code.

1.4 Data Types and Data Structures

DSP-related data types and structures are defined as follows:

- ES_DSP_Capability_S：This structure defines the DSP capability information. It contains fields such as reserved bits for future use.
- ES_DSP_ID_E： An enumeration that defines DSP IDs. Values range from ES_DSP_ID_0 to ES_DSP_ID_7, with ES_DSP_ID_BUTT indicating the maximum DSP ID value.
- ES_DSP_PRI_E：An enumeration that defines the priority levels for DSP tasks. Priorities range from ES_DSP_PRI_0 (highest priority) to ES_DSP_PRI_3 (lowest priority).
- ES_DSP_HANDLE：A data type that defines handles for DSP operators and tasks. It is used as a unique identifier for each operator or task.
- ES_DSP_TASK_S： A structure that defines the message of DSP operator tasks. It includes fields such as operatorHandle, dspBuffers, bufferCntCfg, bufferCntInput, bufferCntOutput, priority, syncCache, pollMode, taskHandle, callback, and cbArg.
- ES_DSP_TASK_CALLBACK：A callback function type used for DSP operator tasks. It is invoked upon task completion, providing a mechanism for post-processing based on return value. The callback receives context and state information, allowing for appropriate response to the task.

1.4.1 ES_DSP_Capability_S

[Explanation]

Defines DSP Capability Information.

[Definition]

```
typedef struct DSP_Capability_S {
    ES_U64 reserved;
} ES_DSP_Capability_S
```

[Members]

Parameter	Description
reserved	Reserved field

1.4.2 ES_DSP_ID_E

[Explanation]

Defines DSP ID.

[Definition]

```
typedef enum DSP_ID_E {
    ES_DSP_ID_0 = 0x0,
    ES_DSP_ID_1 = 0x1,
    ES_DSP_ID_2 = 0x2,
    ES_DSP_ID_3 = 0x3,
    ES_DSP_ID_4 = 0x4,
    ES_DSP_ID_5 = 0x5,
    ES_DSP_ID_6 = 0x6
    ES_DSP_ID_7 = 0x7,
    ES_DSP_ID_BUTT
} ES_DSP_ID_E
```

[Members]

Parameter	Description
ES_DSP_ID_0	DSP ID 0
ES_DSP_ID_1	DSP ID 1
ES_DSP_ID_2	DSP ID 2
ES_DSP_ID_3	DSP ID 3
ES_DSP_ID_4	DSP ID 4
ES_DSP_ID_5	DSP ID 5
ES_DSP_ID_6	DSP ID 6
ES_DSP_ID_7	DSP ID 7
ES_DSP_ID_BUTT	The number of dsp id enumerations.

1.4.3 ES_DSP_PRI_E

[Explanation]

Defines DSP Task Priority. Tasks queued on the same DSP device are scheduled according to priority.

[Definition]

```
typedef enum DSP_PRI_E {
    ES_DSP_PRI_0 = 0x0,
    ES_DSP_PRI_1 = 0x1,
    ES_DSP_PRI_2 = 0x2,
    ES_DSP_PRI_3 = 0x3,
    ES_DSP_PRI_BUTT
} ES_DSP_PRI_E
```

[Members]

Parameter	Description
ES_DSP_PRI_0	Priority 0, highest
ES_DSP_PRI_1	Priority 1
ES_DSP_PRI_2	Priority 2
ES_DSP_PRI_3	Priority 3
ES_DSP_PRI_BUTT	The number of priority enumerations.

1.4.4 ES_DSP_HANDLE

[Explanation]

Defiens DSP task handle.

[Definition]

```
typedef ES_U64 ES_DSP_HANDLE
```

1.4.5 ES_DSP_TASK_CALLBACK

[Explanation]

Defines task callback function for DSP.

[Definition]

```
typedef void (*ES_DSP_TASK_CALLBACK)(void *arg， ES_S32 state)
```

1.4.6 ES_DSP_TASK_S

[Explanation]

Defines DSP Operator Task Description.

[Definition]

```
typedef struct DSP_TASK_S {
    ES_DSP_HANDLE operatorHandle;
    ES_DEV_BUF_S dspBuffers[BUFFER_CNT_MAXSIZE];
    ES_U32 bufferCntCfg;
    ES_U32 bufferCntInput;
    ES_U32 bufferCntOutput;
    ES_DSP_PRI_E priority;
    ES_BOOL syncCache;
    ES_BOOL pollMode;
    ES_DSP_HANDLE taskHandle;
    ES_DSP_TASK_CALLBACK callback;
    ES_VOID *cbArg;
} ES_DSP_TASK_S
```

[Members]

Parameter	Description
operatorHandle	Handle of the loaded DSP operator
dspBuffers	Device buffers to be used by the DSP operator
bufferCntCfg	Total number of buffers for operator configuration information
bufferCntInput	Total number of buffers for operator input information
bufferCntOutput	Total number of buffers for operator output information
priority	Priority of the operator task
syncCache	Driver is responsible for synchronizing host-side cache
pollMode	DSP operates in polling mode
taskHandle	Handle returned after submitting the operator task
callback	Callback function that needs to be called upon task completion
cbArg	Arguments for the callback function

1.5 Error Codes

[Explanation]

Defines DSP API error codes.

[Definition]

```
typedef enum RET_DSP_E {
        ES_DSP_SUCCESS = 0x0,
        ES_DSP_ERROR_INVALID_DEVID = 0x2030001,
        ...
        ES_DSP_ERROR_READ_FILE
} ES_S32
```

[Members]

Parameter	Value	Description
ES_DSP_SUCCESS	0x0	Operation successful
ES_DSP_ERROR_INVALID_DEVID	0x2030001	Device ID is out of valid range
ES_DSP_ERROR_INVALID_CHNID	0x2030002	Channel number error or invalid region handle
ES_DSP_ERROR_ILLEGAL_PARAM	0x2030003	Parameter is out of valid range
ES_DSP_ERROR_EXIST	0x2030004	Attempt to recreate an existing device, channel, or resource
ES_DSP_ERROR_UNEXIST	0x2030005	Attempt to use or destroy a non-existing device, channel, or resource
ES_DSP_ERROR_NULL_PTR	0x2030006	Null pointer found in function parameters
ES_DSP_ERROR_NOT_CONFIG	0x2030007	Module is not configured
ES_DSP_ERROR_NOT_SUPPORT	0x2030008	Unsupported parameter or feature
ES_DSP_ERROR_NOT_PERM	0x2030009	Operation not permitted, such as attempting to modify static configuration parameters
ES_DSP_ERROR_NOMEM	0x203000C	Memory allocation failed, such as insufficient system memory
ES_DSP_ERROR_NOBUF	0x203000D	Buffer allocation failed, such as when the requested image buffer size is too large
ES_DSP_ERROR_BUF_EMPTY	0x203000E	No image in the buffer
ES_DSP_ERROR_BUF_FULL	0x203000F	Buffer is full of images
ES_DSP_ERROR_NOTREADY	0x2030010	System not initialized or corresponding module not loaded
ES_DSP_ERROR_BADADDR	0x2030011	Illegal address
ES_DSP_ERROR_BUSY	0x2030012	System busy

Parameter	Value	Description
ES_DSP_ERROR_SYS_TIMEOUT	0x2030040	System timeout
ES_DSP_ERROR_QUERY_TIMEOUT	0x2030041	Query timeout
ES_DSP_ERROR_OPEN_FILE	0x2030042	Failed to open file
ES_DSP_ERROR_READ_FILE	0x2030043	Failed to read file

1.6 Debugging Information

1.6.1 Proc Debug Information

Debugging information utilizes the Linux proc file system, which can reflect the current system's running state in real time.

This recorded information can be used for problem localization and analysis. The directory for this information is `/proc/esdsp/info`.

To view this information, you can use the `cat` command in the console, for example, `cat /proc/esdsp/info`.

Other common file operation commands, such as `cp /proc/esdsp/info ./`, can also be used to copy the file to the current directory.

In applications, this file can be treated as a regular read-only file for reading operations, such as `fopen`, `fread`, etc.

1.6.2 Proc Information Explanation

- Description

```
# cat /proc/esdsp/info
-----------------------------------DSP PARAM INFO------------------------------
 DieId   CoreId   Enable   CmdTOut(s)
 0         0        Yes      1000000
 0         1        NO          0
 0         2        NO        1000000
 1         0        NO          0
 1         1        NO          0
 1         2        NO          0

--------------------------------DSP RUNTIME INFO-------------------------------
 DieId   CoreId   TotalIntCnt   FinishedTaskCnt   FailedTaskCnt   PendingTaskCnt   LTaskRunTm
 0          0          32              31               0                0       67780000
 0          1          0               0                0                0       0
 0          2          0               0                0                0       0
 1          0          0               0                0                0       0
 1          1          0               0                0                0       0

------------------------DSP HW PERF INFO---------------------------
DieId CoreId TaskName StartTm   PrepSTm   PrepETm   EvalSTm   EvalETm   IPCSTm   EndTm
 0        0    argmax_f16f32 9081856 9081876 9081956 9081976 9091856 9092856 9181856
 0        1      NULL          0         0         0         0         0         0         0
 0        2      NULL          0         0         0         0         0         0         0
 0        3      NULL          0         0         0         0         0         0         0
 1        0      NULL          0         0         0         0         0         0         0
 1        1      NULL          0         0         0         0         0         0         0
 1        2      NULL          0         0         0         0         0         0         0
 1        3      NULL          0         0         0         0         0         0         0

--------------------------------DSP INVOKE INFO--------------------------------
 DieId   CoreId   Pri   TaskName   TaskHnd   TaskStat   TaskRunTm
 0          0      0     softmax    7fe069c0    eval       78900000
 0          1      0     NULL       0           NULL       0
 0          2      0     NULL       0           NULL       0
 1          0      0     NULL       0           NULL       0
 1          1      0     NULL       0           NULL       0
 1          2      0     NULL       0           NULL       0
```

- Debugging Information Analysis

Records current DSP working state and resource information, including DSP queue state information, task state information, runtime state information, and invocation information.

- Parameter Explanation

Module	Parameter	Description
DSP PARAM INFO	DieId	Die ID
	CoreId	DSP device ID
	Enable	DSP enable status: 0 not enabled, 1 enabled
	TaskTmOut	Task execution timeout period in seconds
DSP RUNTIME INFO	DieId	Die ID

Module	Parameter	Description
	CoreId	DSP device ID
	TotalIntCnt	Total number of interrupts generated by DSP
	LTaskRunTm	Execution time of the last task, in microseconds
	FinishedTaskCnt	Total number of successful tasks
	FailedTaskCnt	Total number of failed tasks
	PendingTaskCnt	List of pending tasks
DSP HW PERF INFO	DieId	Die ID
	CoreId	DSP device ID
	Pri	Task priority
	TaskName	Name of the currently running task
	StartTm	Operator start time, in nanoseconds
	PrepSTm	Task prepare start time, in nanoseconds
	PrepETm	Task prepare end time, in nanoseconds
	EvalSTm	Task eval start time, in nanoseconds
	EvalETm	Task eval end time, in nanoseconds
	IPCSTm	DSP notification to driver time, in nanoseconds
	EndTm	Operator end time, in nanoseconds
DSP INVOKE INFO	DieId	Die ID
	CoreId	DSP device ID
	Pri	Task priority
	TaskName	Name of the currently running task
	TaskHnd	Handle of the currently running task
	TaskStat	Status of the currently running task, either prepare or eval
	TaskRunTm	Task run time, in microseconds

1.7 Precautions for Use

For the following operations, it is necessary to ensure they appear in pairs:

- **ES_DSP_Open / ES_DSP_Close:** This pair of functions is used to open and close the DSP device, respectively. It's crucial to close any opened DSP devices to release resources properly.
- **ES_DSP_LoadOperator / ES_DSP_UnloadOperator:** These functions load and unload DSP operators, respectively. After an operator is loaded and used, it should be unloaded to free the allocated resources.
- **ES_DSP_LL_PrepareDMABuffer / ES_DSP_LL_UnprepareDMABuffer:** These functions prepare and release DMA Buffers, respectively. DMA Buffers must be released after their use is complete to prevent memory leaks and ensure efficient resource management.

Failure to use these function pairs appropriately can lead to unexpected errors and resource leaks, impacting system performance and stability.

2. NPU Runtime

2.1 Overview

The ENNP (ESWIN Neural NetWork Processing) platform is an intelligent computing heterogeneous acceleration platform for ESWIN media processing chips. It includes development tools and interfaces for NPU, DSP, HAE and GPU. NPU Runtime is a set of runtime systems and interfaces provided by ENNP for loading NN models and performing inference tasks. It can directly load offline models based on quantification and compilation of ENNP tools to implement functions such as object recognition and image classification. Users develop intelligent analysis solutions based on NPU Runtime. The compiler automatically analyzes and optimizes the execution process, and automatically generates optimized offline models to maximize the reuse of NPU, DSP, HAE, GPU and other hardware, improve hardware utilization and optimize system power consumption.

The NPU runtime software stack is shown in the figure below:

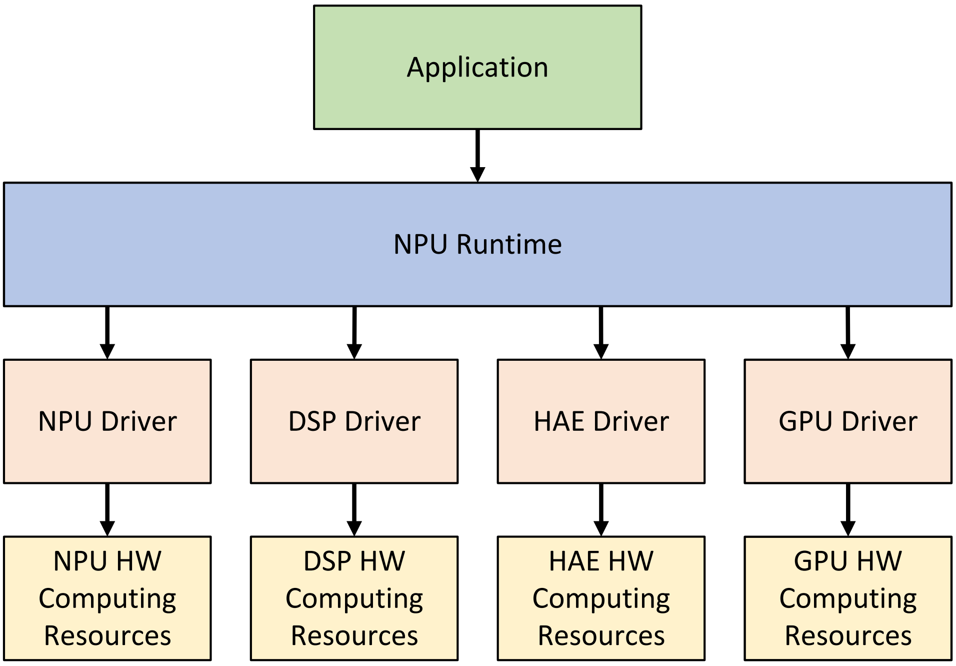
Application
NPU Runtime
NPU Driver
DSP Driver
HAE Driver
GPU Driver
NPU HW Computing Resources
DSP HW Computing Resources
HAE HW Computing Resources
GPU HW Computing Resources

Figure 2-1 NPU Runtime software stack

2.2 Function Description

2.2.1 Function Description

EIC7700 hardware components includes NPU, DSP, HAE, and GPU. According to the user's configuration and model implementation, the composite model will call the corresponding hardware modules as needed during runtime to perform corresponding calculations. The scheduling between different operators is determined by Runtime based on the dependencies described by the model. NPU runtime supports multi-task and multi-process submission.

2.2.2 Basic Conception

NPU Runtime provides C APIs including device management, context management, stream management, memory management, model loading and execution, etc.

Parameters	Descriptions
Synchronous/ Asynchronous	Synchronization and asynchronousness are from the perspective of the caller and the executor. In the current scenario, if the board environment calls the interface without waiting for the device execution to complete before returning, it means that the scheduling of the board environment is asynchronous; If you need to wait for the device execution to complete before returning after calling the interface, it means that the scheduling is synchronous.
process /thread	The processes and threads mentioned in this article, unless otherwise noted, refer to the processes and threads on the board environment.
Device	NPU (Neural-Network Processing Unit) hardware unit.
Context	Context describes a configuration context for NPU running. At the same time, it serves as a container and manages the life cycle of objects created with the corresponding Context, which is used to manage asynchronously submitted

Parameters	Descriptions
	tasks. A Context can be associated with multiple threads of a process, and submitted tasks will be scheduled in order. Streams of different Contexts are completely isolated. Call the ES_NPU_CreateContext interface in a process or thread to create a Context
Stream	A queue waiting for the user to retrieve it after the asynchronous operation task is completed. Call the ES_NPU_CreateStream interface in a process or thread to create a Stream

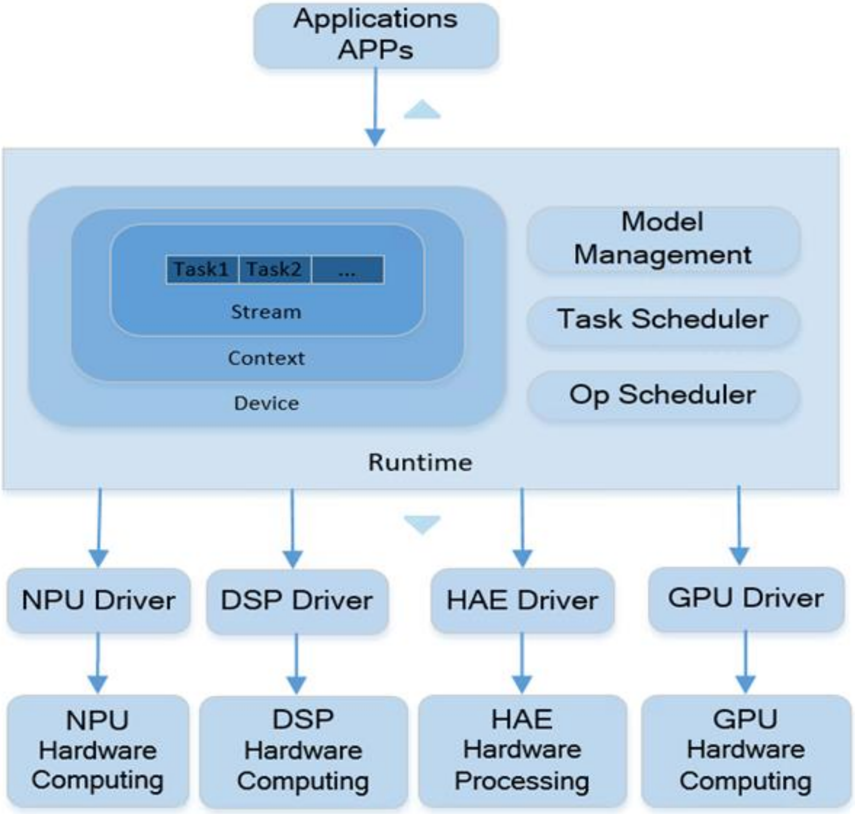

Figure 2-2 Device, Context, Stream, Task relationship

- Users use the ES_NPU_SetDevice interface to specify the Device used for calculations. When ES_NPU_ReleaseDevice is called, the corresponding Device is released. After the Device is released, its corresponding resources are unavailable in this process.
- When using the NPU asynchronous task submission interface, Context is used to specify the waiting queue used by the task. An NPU device can contain multiple Contexts, but a Context can only belong to a specific device. The Context life cycle begins with ES_NPU_CreateContext and ends with ES_NPU_DestroyContext. Context is bound to user threads. One user thread corresponds to one Context. You can call the ES_NPU_SetCurrentContext interface to switch or bind Context. A Context can be bound to multiple threads.
- When using the NPU asynchronous task submission interface, Stream is used to specify the completion waiting queue after the task is completed.
  The Stream life cycle begins with ES_NPU_CreateStream and ends with ES_NPU_DestroyStream. A Context can correspond to multiple Streams.
- If you use the NPU synchronous task submission interface, you do not need to apply to create Context and Stream.
- Task is the real execution body on Device. When submitting a task, associate it

to a specific Stream.

2.3 API Reference

2.3.1 ES_NPU_GetVersion

【Declaration】

ES_S32 ES_NPU_GetVersion (ES_U64 * version)

【Description】

Gets the version information of NPU runtime software stack.

【Parameters】

Parameter Name	Descriptions	Input/Output
version	Runtime version number	Output, see ES_SDK_VERSION_U

【Return】

Return value	Descriptions
0	Success
non-0	Failure, see 2.5Error Code

2.3.2 ES_NPU_GetVersionString

【Declaration】

```
ES_S32 ES_NPU_GetVersionString (
                ES_CHAR *version,
                ES_U32 maxSize)
```

【Description】

Gets NPU runtime software stack version information in character format.

【Parameters】

Parameter Name	Descriptions	Input/Output
version	The string of runtime version number.	output
maxSize	The maximum size of runtime version number string.	Input

【Return】

Return value	Descriptions
0	Success

non-0	Failure, see 2.5Error Code

2.3.3 ES_NPU_GetCapability

【Declaration】

ES_S32 ES_NPU_GetCapability (ES_NPU_Capability_S * capability)

【Description】

Get NPU capability information.

【Parameters】

Parameter Name	Descriptions	Input/Output
capability	Structure to store NPU capability information	output

【Return】

Return value	Descriptions
0	Success
non-0	Failure, see 2.5Error Code

2.3.4 ES_NPU_SetLogLevel

【Declaration】

ES_S32 ES_ NPU _SetLogLevel (ES_U32 level)

【Description】

Set the printing log level when NPU is running.

【Parameters】

Parameter Name	Descriptions	Input/Output
level	NPU print log level	Input

【Return】

Return value	Descriptions
0	Success
non-0	Failure, see 2.5Error Code

2.3.5 ES_NPU_GetDeviceProperties

【Declaration】

ES_S32 ES_NPU_GetDeviceProperties (

ES_S32 deviceId,
NPU_DEVICE_PROP_S * prop)

【Description】

Get the properties of the device identified by the specified deviceId.

【Parameters】

Parameter Name	Descriptions	Input/Output
deviceId	The deviceId of the device.	Input
prop	The properties of device.	output

【Return】

Return value	Descriptions
0	Success
non-0	Failure, see 2.5Error Code

2.3.6 ES_NPU_GetNumDevices

【Declaration】

ES_S32 ES_NPU_GetNumDevices(ES_U16 *deviceNum)

【Description】

Gets the number of available devices in the current environment.

【Parameters】

Parameter Name	Descriptions	Input/Output
deviceNum	The number of devices obtained	output

【Return】

Return value	Descriptions
0	Success
non-0	Failure, see 2.5Error Code

2.3.7 ES_NPU_SetDevice

【Declaration】

ES_S32 ES_NPU_SetDevice(ES_U16 deviceId)

【Description】

Sets device to be used for NPU executions.

【Parameters】

Parameter Name	Descriptions	Input/Output
deviceId	Current device Id	Input

【Return】

Return value	Descriptions
0	Success
non-0	Failure, see 2.5Error Code

【Notice】

The deviceId should be smaller than deviceNum obtained through ES_NPU_GetNumDevices.

2.3.8 ES_NPU_ReleaseDevice

【Declaration】

ES_S32 ES_NPU_ReleaseDevice(ES_U16 deviceId)

【Description】

Release the device.

【Parameters】

Parameter Name	Descriptions	Input/Output
deviceId	Current device Id	Input

【Return】

Return value	Descriptions
0	Success
non-0	Failure, see 2.5Error Code

【Notice】

The incoming parameter should be the deviceId set by the ES_NPU_SetDevice interface.

2.3.9 ES_NPU_GetDevice

【Declaration】

ES_S32 ES_NPU_GetDevice(ES_U16 *deviceId)

【Description】

Get the deviceId currently in use.

【Parameters】

Parameter Name	Descriptions	Input/Output
deviceId	Current device Id	output

【Return】

Return value	Descriptions
0	Success
non-0	Failure, see 2.5Error Code

2.3.10 ES_NPU_CreateContext

【Declaration】

```
ES_S32 ES_NPU_CreateContext (
            npu_context *    context,
            ES_U16 deviceId)
```

【Description】

Create a runtime context.

【Parameters】

Parameter Name	Descriptions	Input/Output
context	The returned context pointer	output
deviceId	The deviceId to be used in the current context	Input

【Return】

Return value	Descriptions
0	Success
non-0	Failure, see 2.5Error Code

2.3.11 ES_NPU_SetCurrentContext

【Declaration】

```
ES_S32 ES_NPU_SetCurrentContext (npu_context context)
```

【Description】

Set the context to the current thread.

【Parameters】

Parameter Name	Descriptions	Input/Output
context	context to set	Input

【Return】

Return value	Descriptions
0	Success
non-0	Failure, see 2.5Error Code

2.3.12 ES_NPU_GetCurrentContext

【Declaration】

ES_S32 ES_NPU_GetCurrentContext (npu_context *context)

【Description】

Get the context of the current thread.

【Parameters】

Parameter Name	Descriptions	Input/Output
context	Context of current thread	Output

【Return】

Return value	Descriptions
0	Success
non-0	Failure, see 2.5Error Code

2.3.13 ES_NPU_DestroyContext

【Declaration】

ES_S32 ES_NPU_DestroyContext(npu_context context)

【Description】

Destroy context.

【Parameters】

Parameter Name	Descriptions	Input/Output
context	The context to be destroyed	Input

【Return】

Return value	Descriptions

0	Success
non-0	Failure, see 2.5Error Code

2.3.14 ES_NPU_CreateStream

【Declaration】

ES_S32 ES_NPU_CreateStream(npu_stream *stream)

【Description】

Create a stream in the Context corresponding to the current thread.

【Parameters】

Parameter Name	Descriptions	Input/Output
stream	The newly created stream.	output

【Return】

Return value	Descriptions
0	Success
non-0	Failure, see 2.5Error Code

2.3.15 ES_NPU_DestroyStream

【Declaration】

ES_S32 ES_NPU_DestroyStream (npu_stream stream)

【Description】

Destroys stream in the current thread.

【Parameters】

Parameter Name	Descriptions	Input/Output
stream	The stream to be destroyed.	Input

【Return】

Return value	Descriptions
0	Success
non-0	Failure, see 2.5Error Code

2.3.16 ES_NPU_SynchronizeStream

【Declaration】

ES_S32 ES_NPU_SynchronizeStream (npu_stream stream)

【Description】

Blocks the thread until all tasks in the specified stream are completed.

【Parameters】

Parameter Name	Descriptions	Input/Output
stream	The stream to be synchronized	Input

【Return】

Return value	Descriptions
0	Success
non-0	Failure, see 2.5Error Code

2.3.17 ES_NPU_AbortStream

【Declaration】

ES_S32 ES_NPU_AbortStream (npu_stream stream)

【Description】

Aborts all tasks in this stream.

【Parameters】

Parameter Name	Descriptions	Input/Output
stream	The stream to be aborted	Input

【Return】

Return value	Descriptions
0	Success
non-0	Failure, see 2.5Error Code

2.3.18 ES_NPU_LoadModelFromFile

【Declaration】

```
ES_S32 ES_NPU_LoadModelFromFile (
            ES_U32 * modelId,
            const ES_CHAR * modelPath)
```

【Description】

Load offline model from file.

【Parameters】

Parameter Name	Descriptions	Input/Output
modelId	Returned offline model ID	output
modelPath	Offline model file path	Input

【Return】

Return value	Descriptions
0	Success
non-0	Failure, see 2.5Error Code

2.3.19 ES_NPU_LoadModelFromMemory

【Declaration】

```
ES_S32 ES_NPU_LoadModelFromMemory(
                ES_U32 * modelId,
                ES_CHAR * pBuffer,
                ES_U32 nBufLen)
```

【Description】

Load the offline model from memory.

【Parameters】

Parameter Name	Descriptions	Input/Output
modelId	Returned offline model ID	output
pBuffer	The buffer used to save model	input
nBufLen	The length of buffer	input

【Return】

Return value	Descriptions
0	Success
non-0	Failure, see 2.5Error Code

2.3.20 ES_NPU_UnloadModel

【Declaration】

```
ES_S32 ES_NPU_UnloadModel (ES_U32 modelId)
```

【Description】

Unloads offline models.

【Parameters】

Parameter Name	Descriptions	Input/Output
modelId	Offline model ID to be uninstalled	Input

【Return】

Return value	Descriptions
0	Success
non-0	Failure, see 2.5Error Code

2.3.21 ES_NPU_LoadCompositeModel

【Declaration】

```
ES_S32 ES_NPU_LoadCompositeModel (
                ES_U32 * modelId,
                const ES_CHAR *modelPath )
```

【Description】

Load the composite model, which is a set of models made to maximize the use of NPU/DSP hardware.

【Parameters】

Parameter Name	Descriptions	Input/Output
modelId	Returned offline model ID	Output
modelPath	Model directory path	enter

【Return】

Return value	Descriptions
0	success
Non-zero	Failed, see 2.5Error Code

【Notice】

The composite model loading interface is only used for composite model loading and cannot be used with ordinary offline models.

2.3.22 ES_NPU_GetCompositeModelInfo

【Declaration】

```
ES_S32 ES_NPU_GetCompositeModelInfo (
                ES_U32 modelId,
```

NPU_COMPOSITE_MODEL_INFO_S *compositeModelInfo)

【Description】

Gets composite model information, including detailed information of all models in the model set.

【Parameters】

Parameter Name	Descriptions	Input/Output
modelId	Composite Model Id	enter
compositeModelInfo	Model information for composite models	Output

【Return】

Return value	Descriptions
0	success
Non-zero	Failed, see 2.5Error Code

【Notice】

This interface is only valid for the modelId of the composite model.

2.3.23 ES_NPU_UnloadCompositeModel

【Declaration】

ES_S32 ES_NPU_UnloadCompositeModel (ES_U32 modelId)

【Declaration】

Gets composite model information, including detailed information of all models in the model set.

【Parameters】

Parameter Name	Descriptions	Input/Output
modelId	Composite Model Id	enter

【Return】

Return value	Descriptions
0	success
Non-zero	Failed, see 2.5Error Code

【Notice】

This interface is only valid for the modelId of the composite model.

2.3.24 ES_NPU_GetNumInputTensors

【Declaration】

```
ES_S32 ES_NPU_GetNumInputTensors (
                ES_U32 modelId,
                ES_S32 * inputTensors)
```

【Description】

Get the number of tensors for the input of the specified model.

【Parameters】

Parameter Name	Descriptions	Input/Output
modelId	Offline model ID	Input
inputTensors	The number of input tensors returned	output

【Return】

Return value	Descriptions
0	Success
non-0	Failure, see 2.5Error Code

2.3.25 ES_NPU_GetNumOutputTensors

【Declaration】

```
ES_S32 ES_NPU_GetNumOutputTensors (
                ES_U32 modelId,
                ES_S32 *outputTensors)
```

【Description】

Get the number of output tensors of the specified model.

【Parameters】

Parameter Name	Descriptions	Input/Output
modelId	Offline model ID	Input
out putTensors	The number of output tensors returned	output

【Return】

Return value	Descriptions
0	Success
non-0	Failure, see 2.5Error Code

2.3.26 ES_NPU_GetInputTensorDesc

【Declaration】

```
ES_S32 ES_NPU_GetInputTensorDesc (
            ES_U32 modelId,
            ES_S32 tensorId,
            NPU_TENSOR_S *tensor)
```

【Description】

Gets input tensor description of specified model.

【Parameters】

Parameter Name	Descriptions	Input/Output
modelId	Offline model ID	Input
tensorId	The tensorId starts from 0 and goes up to the maximum value obtained by subtracting 1 from the number of inputTensors retrieved via the ES_NPU_GetNumInputTensors interface.	Input
tensor	Return tensor description information	output

【Return】

Return value	Descriptions
0	Success
non-0	Failure, see 2.5Error Code

2.3.27 ES_NPU_GetOutputTensorDesc

【Declaration】

```
ES_S32 ES_NPU_GetOutputTensorDesc (
            ES_U32 modelId,
            ES_S32 tensorId,
            NPU_TENSOR_S *tensor)
```

【Description】

Get the output tensor description specified by the specified model.

【Parameters】

Parameter Name	Descriptions	Input/Output
modelId	Offline model ID.	Input
tensorId	The tensorId starts from 0 and is less than the number of output tensors obtained via	Input

Parameter Name	Descriptions	Input/Output
	the ES_NPU_GetNumOutputTensors interface.	
tensor	Return tensor description information.	output

【Return】

Return value	Descriptions
0	Success
non-0	Failure, see 2.5Error Code

2.3.28 ES_NPU_Prepare

【Declaration】

```
ES_S32 ES_NPU_Prepare (
                ES_DEV_BUF_S *devBuf)
```

【Description】

Prepares the SMMU mapping on NPU device for the DMA buffer required during task execution. Before calling this interface, it is necessary to first call ES_NPU_SetDevice.

【Parameters】

Parameter Name	Descriptions	Input/Output
devBuf	DMA buffer pointer	Input

【Return】

Return value	Descriptions
0	Success
non-0	Failure, see 2.5Error Code

2.3.29 ES_NPU_Unprepare

【Declaration】

```
ES_S32 ES_NPU_Unprepare (
                ES_DEV_BUF_S *devBuf)
```

【Description】

Release the SMMU mapping of the DMA Buffer to NPU device and call it in pairs with ES_NPU_Prepare.

【Parameters】

Parameter Name	Descriptions	Input/Output
devBuf	DMA buffer pointer	Input

【Return】

Return value	Descriptions
0	Success
non-0	Failure, see 2.5Error Code

2.3.30 ES_NPU_Submit

【Declaration】

```
ES_S32 ES_NPU_Submit (
                NPU_TASK_S *tasks,
                ES_U32 numTasks)
```

【Description】

Submits tasks (synchronization interface). This interface submits NPU tasks and waits for all tasks to be executed.

【Parameters】

Parameter Name	Descriptions	Input/Output
tasks	Submitted tasks array pointer	Input
numTasks	The number of submitted tasks	Input

【Return】

Return value	Descriptions
0	Success
non-0	Failure, see 2.5Error Code

2.3.31 ES_NPU_SubmitAsync

【Declaration】

```
ES_S32 ES_NPU_SubmitAsync(
                NPU_TASK_S *tasks,
                ES_U32 numTasks,
                npu_stream stream)
```

【Description】

Submits tasks (asynchronization interface). This interface returns directly after submitting the NPU task. The user needs to query the completion status of the task on the corresponding stream through

ES_NPU_ProcessReport.

【Parameters】

Parameter Name	Descriptions	Input/Output
tasks	Tasks to be submitted	Input
numTasks	The number of submitted tasks	Input
stream	Pointer to submit the task to the specified stream	Input

【Return】

Return value	Descriptions
0	Success
non-0	Failure, see 2.5Error Code

2.3.32 ES_NPU_AllocTaskMemory

【Declaration】

```
ES_S32 ES_NPU_AllocTaskMemory(
                ES_U32 modelId,
                ES_U32 nums,
                NPU_TASK_MEM_S *taskMem);
```

【Description】

Allocate memory required for the task.

【Parameters】

Parameter Name	Descriptions	Input/Output
modelId	The model id corresponding to the task to allocate memory	enter
nums	The number of tasks that need to allocate memory	enter
taskMem	Save allocated memory information	Output

【Return】

Return Value	Descriptions
0	success
Non-zero	Failed, see 2.5Error Code

2.3.33 ES_NPU_ReleaseTaskMemory

【Declaration】

```
ES_S32 ES_NPU_ReleaseTaskMemory(
            ES_U32 modelId,
            ES_U32 nums,
            NPU_TASK_MEM_S *taskMem);
```

【Description】

Release task memory.

【Parameters】

Parameter Name	Descriptions	Input/Output
modelId	The model id corresponding to the memory to be released	enter
nums	The amount of memory to be released	enter
taskMem	Memory information that needs to be released	enter

【Return】

Return Value	Descriptions
0	success
Non-zero	Failed, see 2.5Error Code

2.3.34 ES_NPU_SetFlexibleTaskAttr

【Declaration】

```
ES_S32 ES_NPU_SetFlexibleTaskAttr (
            ES_U32 modelId,
            NPU_FLEXIBLE_TASK_ATTR_S *attr )
```

【Description】

Set the required properties of the flexible task.

【Parameters】

Parameter Name	Descriptions	Input/Output
modelId	Composite Model Id	enter
attr	Flexible task attributes	enter

【Return】

Return Value	Descriptions
0	success
Non-zero	Failed, see 2.5Error Code

【Notice】

This interface is only valid for the modelId of the composite model.

2.3.35 ES_NPU_SubmitFlexibleTask

【Declaration】

```
ES_S32 ES_NPU_SubmitFlexibleTask(
            NPU_TASK_S *tasks,
            ES_U32 numTasks,
            npu_stream stream);
```

【Description】

Submit composite model inference tasks asynchronously. Compared with ordinary offline model tasks, users do not need to assemble corresponding model tasks according to batch size. They can directly construct corresponding tasks according to single frame requirements to reduce the complexity of task construction. For multi-frame tasks, the number of tasks can be specified. At the same time, the npu runtime will automatically adjust the batch size for the scheduling of composite model tasks to achieve better performance.

【Parameters】

Parameter Name	Descriptions	Input/Output
tasks	Pointer to the task that needs to be submitted	enter
numTask	Number of tasks submitted	enter
stream	The stream corresponding to the submitted task	enter

【Return】

Return Value	Descriptions
0	success
Non-zero	Failed, see 2.5Error Code

【Notice】

Tasks submitted using this interface need to use the ES_NPU_AllocTaskMemory interface to allocate memory and can only be used to submit tasks for composite models.

2.3.36 ES_NPU_ProcessReport

【Declaration】

```
ES_S32 ES_NPU_ProcessReport (
        npu_stream stream,
        ES_S 32 timeoutMs)
```

【Description】

Set the waiting timeout, query which tasks have been completed, and then call the task's callback function. This function will also return if no tasks have completed when the timeout expires.

【Parameters】

Parameter Name	Descriptions	Input/Output
stream	The stream pointer to be processed	Input
timeoutMs	The unit is ms. If timeoutMs is - 1, this interface will block and wait until a task is completed. If timeoutMs >=0 , wait for the specified time to query the task completion	Input

【Return】

Return value	Descriptions
0	Success
non-0	Failure, see 2.5Error Code

2.4 Data Types and Data Structures

NPU Runtime related data types and data structures are defined as follows:

- NPU_DIMS4_S: Defines tensor dimension parameters
- NPU_TENSOR_S: Defines tensor structure parameters
- NPU_TASK_S: Defines npu task structure description
- NPU_TaskCallback: Defines the callback function that needs to be called after the task is executed.
- npu_context: Defines the context pointer of npu.
- npu_stream: Defines the stream pointer of npu.

2.4.1 NPU_DIMS4_S

【Explanation】

Define tensor dimension parameters.

【Definition】

```
typedef struct {
    ES_S32 n;
    ES_S32 c;
    ES_S32 h;
    ES_S32 w;
} NPU_DIMS4_S
```

【Member】

Parameters	Descriptions
n	Batch size, indicating the number of batches
c	Number of channels in an image
h	The vertical dimension of image, in number of pixels
w	The horizontal dimension of image, in number of pixels

2.4.2 NPU_TENSOR_S

【Explanation】

Define tensor structure parameters.

【Definition】

```
typedef struct {
    ES_CHAR name [ES_NPU_TENSOR_DESC_NAME_MAX_LEN + 1];
    ES_U64 bufferSize;
    NPU_DIMS4_S dims;
    ES_U8 dataFormat;
    ES_U8 dataType;
    ES_U8 dataCategory;
    ES_U8 pixelFormat;
    ES_U8 pixelMapping;
    ES_U32 stride[ES_NPU_TENSOR_DESC_NUM_STRIDES];
} NPU_TENSOR_S
```

【Member】

Parameters	Descriptions
name	Tensor 's name
bufferSize	Tsensor buffer size

dims	Tensor dimension parameters
dataFormat	Tensor data layout format (NCHW/NHWC)
dataType	Tensor's data type (int8 , int 16 , fp 16 , fp 32 , etc.)
dataCategory	Tensor (image, weight, bias, etc.)
pixelFormat	Pixel format
pixelMapping	Pixel Mapping
stride	Stride information

2.4.3 NPU_TASK_S

【Explanation】

Define task structure parameters.

【Definition】

```
typedef struct {
    ES_U32 taskId;
    ES_U32 modelId;
    ES_U8 inputFdNum;
    ES_U8 outputFdNum;
    ES_DEV_BUF_S inputFd[ESTASK_MAX_FD_CNT];
    ES_DEV_BUF_S outputFd[ESTASK_MAX_FD_CNT];
    NPU_TaskCallback callback;
    ES_VOID *callbackArg;
    NPU_TASK_STATE_E state;
    ES_U8 sdkPrivate[ES_TASK_SDK_PRIVATE_LEN];
} NPU_TASK_S
```

【Member】

Parameters	Descriptions
TaskId	ID of the task , assigned by the system after the task is submitted
modelId	ID corresponding to the task
inputFdNum	Number of memory blocks as input
outputFdNum	Number of memory blocks as output
inputFd	Store the mem Fd value of the input memory block
outputFd	Store the memFd value of the output memory block
callback	The callback function that needs to be called after the task is executed
callbackArg	callback function parameters

state	task state

2.4.4 NPU_TASK_MEM_S

【Definition】

```
typedef struct {
ES_U8 inputFdNum;
ES_U8 outputFdNum;
ES_DEV_BUF_S inputFd[ES_TASK_MAX_FD_CNT];
ES_DEV_BUF_S outputFd[ES_TASK_MAX_FD_CNT];
} NPU_TASK_MEM_S;
```

【Member】

Parameters	Descriptions
inputFdNum	The number of memory blocks to use as input
outputFdNum	The number of memory blocks to output
inputFd	Stores information about the input memory block
outputFd	Stores information about output memory blocks

2.4.5 NPU_MODEL_INFO_S

【Definition】

```
typedef struct {
ES_CHAR modelName[MAX_MODEL_FILE_NAME];
NPU_MODEL_TYPE_E modelType;
ES_U32 modelBatch;
ES_FLOAT modelCost;
ES_U32 modelId;
ES_U32 modelDevices;
} NPU_MODEL_INFO_S;
```

【Member】

Parameters	Descriptions
modelName	Model file path
modelType	Model Type
modelBatch	Model batch size
modelCost	Model cost, theoretical time consumed (ms)
modelId	ModelId

modelDevices	The amount of hardware the model requires

2.4.6 NPU_COMPOSITE_MODEL_INFO_S

【Definition】

```
typedef struct {
ES_U32 modelNums;
NPU_MODEL_INFO_S modelsInfo[MAX_MODELS_OF_SET];
} NPU_COMPOSITE_MODEL_INFO_S;
```

【Member】

Parameters	Descriptions
modelNums	Number of models
modelsInfo	Model information array

2.4.7 NPU_FLEXIBLE_TASK_ATTR_S

【Definition】

```
typedef struct {
ES_S32 timeOut;
} NPU_FLEXIBLE_TASK_ATTR_S;
```

【Member】

Parameters	Descriptions
timeOut	Task timeout (after timeout, schedule the task as soon as possible in ms)

2.4.8 NPU_TaskCallback

【Explanation】

The callback function that needs to be called after the task is executed.

【Definition】

```
typedef ES_S32 (*NPU_TaskCallback) (void *arg)
```

2.4.9 npu_context

【Explanation】

Define the context pointer of npu.

【Definition】

```
typedef void *npu_context
```

2.4.10 npu_stream

【Explanation】

Define npu's stream pointer.

【Definition】

typedef void * npu_ stream

2.5 Error Code

Parameters	value	Descriptions
ES_NPU_ERROR_BAD_PARAM	0xA00F6003	Illegal parameters
ES_NPU_ERROR_NULL_PTR	0xA00F6006	null pointer
ES_NPU_ERROR_NO_MEMORY	0xA00F600C	Not enough storage
ES_NPU_ERROR_INVALID_ADDR	0xA00F6011	Invalid address
ES_NPU_ERROR_BUSY	0xA00F6012	Device is busy
ES_NPU_ERROR_NOT_INIT	0xA00F6040	Device not initialized
ES_NPU_ERROR_TIME_OUT	0xA00F6041	Operation timeout
ES_NPU_ERROR_INVALID_STATE	0xA00F6042	Invalid status
ES_NPU_ERROR_INVALID_SIZE	0xA00F6043	Invalid size
ES_NPU_ERROR_BAD_VALUE	0xA00F6044	wrong value
ES_NPU_ERROR_DEV_NOT_FOUND	0xA00F6045	Device not found
ES_NPU_ERROR_DEV_MULTI_SET	0xA00F6046	Multiple device settings
ES_NPU_ERROR_DEV_OPEN_FAILED	0xA00F6047	Device opening failed
ES_NPU_ERROR_MODEL_NOT_PRESENT	0xA00F6048	Model not loaded
ES_NPU_ERROR_MODEL_NOT_FOUND	0xA00F6049	Model not found
ES_NPU_ERROR_CONTEXT_NOT_FOUND	0xA00F604A	Context not found
ES_NPU_ERROR_CONTEXT_INVALID	0xA00F604B	Invalid context
ES_NPU_ERROR_STREAM_NOT_FOUND	0xA00F604C	Stream not found
ES_NPU_ERROR_STREAM_INVALID	0xA00F604D	Invalid stream
ES_NPU_ERROR_TASK_NOT_FOUND	0xA00F604E	Task not found

Parameters	value	Descriptions
ES_NPU_ERROR_FILE_WRITE_FAIL	0xA00F604F	File write failed
ES_NPU_ERROR_FILE_READ_FAIL	0xA00F6050	File read failed
ES_NPU_ERROR_FILE_OPREATION_FAIL	0xA00F6051	File operation failed
ES_NPU_ERROR_END_OF_FILE	0xA00F6052	end of file
ES_NPU_ERROR_DIR_OPREATION_FAIL	0xA00F6053	Directory operation failed
ES_NPU_ERROR_END_OF_DIR_LIST	0xA00F6054	End of directory listing
ES_NPU_ERROR_IOCTL_FAIL	0xA00F6055	IOCTL operation failed
ES_NPU_ERROR_CREAT_EVENTFD_FAIL	0xA00F6056	Failed to create EventFD
ES_NPU_ERROR_LOAD_OP_FAILED	0xA00F6058	Load operation failed
ES_NPU_ERROR_SUBMIT_TASK_FAILED	0xA00F6059	Failed to submit task
ES_NPU_ERROR_LOAD_SRAM_FAIL	0xA00F605A	Loading SRAM failed
ES_NPU_ERROR_BAD_SRAM_SIZE	0xA00F605B	Wrong SRAM size
ES_NPU_ERROR_DSP_QUERY_FAILED	0xA00F605C	DSP query failed

2.6 Debugging Information

Log level	value	Output information
ES_LOG_ERR	3	Print error log only
ES_LOG_WARN	4	Print error and warning logs
ES_LOG_NOTICE	5	Print error and warning and prompt level logs
ES_LOG_INFO	6	Information level, for detailed information during debugging and development phases
ES_LOG_DBG	7	Debug level, provides more detailed information, usually used for in-depth debugging

2.7 Precautions for Use

During the construction and destruction of objects, it is a good practice to follow the construction sequence of device ->context->stream and the destruction sequence of stream->context->device to ensure the correct initialization of the object and the correct resources freed.

Specifically, when constructing an object, first construct the outermost Device object, then construct the Context object it depends on, and then construct the Stream object that depends on the Context. This ensures that the object's dependencies are correctly satisfied, and that each object at construction time can depend on the outer object that has been correctly constructed.

When destructing an object, first destruct the Stream object, then destruct the Context object that depends on the Stream, and finally destruct the Device object that depends on the Context. This order ensures that object dependencies are released correctly, and each object can depend on the inner object that has been correctly destroyed when it is destroyed.

3. AcceleratorKit

3.1 Overview

The AcceleratorKit encapsulates calls to hardware such as NPU, DSP, HAE, GPU, etc. and exports them as corresponding computing acceleration function interfaces. It includes some complex preprocessing, post-processing operators, image processing, machine vision and other functions. The AcceleratorKit library provides operators and functions that can be executed on different types of devices to maximize the computing power of different types of devices.

3.2 Function Description

The AcceleratorKit offers users a range of commonly used computing acceleration operators, as detailed in the following table：

Operators Type	API Interface Function	Device
cosDistance	Calculate the cosine distance between two sets of data.	DSP
argmax	Perform Argmax calculation to sort the input data along the specified dimension, identifying the maximum value.	DSP
detectionOut	Perform post-processing on the output feature map generated by detection network.	DSP
softmax	Perform Softmax calculation	DSP
perspectiveAffine	Perform perspective transformation on the image	DSP
warpAffine	Perform affine transformation on the image	DSP
similarityTransform	Calculate the coefficient matrix for similarity transformation.	CPU

3.3 API Reference

3.3.1 ES_AK_GetVersion

【Declaration】

ES_S32 ES_AK_GetVersion(ES_U64 *version)

【Description】

Get Accelerator digital version information.

【Parameters】

Parameter Name	Descriptions	Input/Output
Version	Accelerator version number.	output，refer ES_SDK_VERSION_U

【Return】

Return value	Descriptions
0	Success
others	Failure, see error code 3.5.

3.3.2 ES_AK_GetVersionString

【Declaration】

```
ES_S32 ES_ AK _GetVersionString(
                ES_CHAR *version,
                ES_U32 maxSize)
```

【Description】

Get the Accelerator string version information.

【Parameters】

Parameter Name	Descriptions	Input/Output
Version	Accelerator version number.	output
maxSize	AcceleratorMaximumMaximum length of the version number string.	input

【Return】

Return value	Descriptions
0	Success
others	Failure, see error code 3.5.

3.3.3 ES_AK_GetCapability

【Declaration】

```
ES_S32 ES_AK_GetCapability(ES_AK_Capability_S *capability)
```

【Description】

Get the Accelerator capability information.

【Parameters】

Parameter Name	Descriptions	Input/Output
capability	Get the Accelerator capability information.	output，refer ES_AK_Capability_S

【Return】

Return value	Descriptions
0	Success
others	Failure, see error code 3.5.

3.3.4 ES_AK_SetLogLevel

【Declaration】

ES_S32 ES_AK_SetLogLevel(ES_U32 level)

【Description】

Set the logging level for the Accelerator runtime.

【Parameters】

Parameter Name	Descriptions	Input/Output
Level	Accelerator logging level	input

【Return】

Return value	Descriptions
0	Success
others	Failure, see error code 3.5.

3.3.5 ES_AK_Init

【Declaration】

ES_S32 ES_AK_Init()

【Description】

Initialize device resources, including opening hardware devices such as DSP.

【Parameters】

None

【Return】

Return value	Descriptions
0	Success
others	Failure, see error code 3.5.

3.3.6 ES_AK_Deinit

【Declaration】

ES_S32 ES_AK_Deinit()

【Description】

Release device resources and close hardware devices.

【Parameters】

None

【Return】

Return value	Descriptions
0	Success
others	Failure, see error code 3.5.

3.3.7 ES_AK_SetDevice

【Declaration】

```
ES_S32 ES_AK_SetDevice (
            const ES_AK_DEVICE_E*      devices,
            ES_U32                       devNum)
```

【Description】

Configure devices for computation, including configuring specified DSP and GPU for computation.

【Parameters】

Parameter Name	Descriptions	Input/Output
devices	Activate the necessary hardware devices for computation.	input
devNum	Number of hardware device configurations.	input

【Return】

Return value	Descriptions
0	Sucess

others	Failure, see error code 3.5.

3.3.8 ES_AK_GetDevice

【Declaration】

```
ES_S32 ES_AK_GetDevice (
                ES_AK_DEVICE_E*         devices,
                ES_U32                    devNum)
```

【Description】

Get the current running device.

【Parameters】

Parameter Name	Descriptions	Input/Output
devices	The current computing hardware device.	output
devNum	The number of hardware devices.	input

【Return】

Return value	Descriptions
0	Success
others	Failure, see error code 3.5.

3.3.9 ES_AK_DSP_CosDistance

【Declaration】

```
ES_S32 ES_AK_DSP_CosDistance(
                ES_AK_DEVICE_E deviceId,
                ES_TENSOR_S    inputBase,
                ES_TENSOR_S    inputQuery,
                ES_TENSOR_S    output)
```

【Description】

Calculate the cosine distance between two sets of data.

【Supported Types】

Input float16, output float16.

Input float16, output float32.

Input float32, output float32.

【Parameters】

Parameter Name	Descriptions	Constraint
deviceId	Device ID value	For detailed information, refer to section 3.4.4 of the ES_AK_DEVICE_E structure
inputBase	Input data information for calculating cosine distance.	The format is inputBase[featureNum1, dim], only supports 2D operation, where featureNum1 represents the number of feature vectors in inputBase, and dim represents the length of feature vectors. featureNum1 range: [1, 512] dim range: [1, 8192]
inputQuery	Input data information for calculating cosine distance.	The format is inputQuery[featureNum2, dim], only supports 2D operation where featureNum2 represents the number of feature vectors in inputQuery, and dim represents the length of feature vectors, same as inputBase. featureNum2 range: [1, 8192]
output	Output the cosine distance information between InputBase and inputQuery.	The format is output[featureNum1, featureNum2]

【Return】

Return value	Descriptions
0	Success
others	Failure, see error code 3.5.

3.3.10 ES_AK_DSP_PerspectiveAffine

【Declaration】

```
ES_S32 ES_AK_DSP_PerspectiveAffine(
            ES_AK_DEVICE_E          deviceId,
            ES_TENSOR_S             input,
            ES_TENSOR_S             output,
            ES_FLOAT                 M[AFFINE_MATRIX_SIZE],
            ES_INTER_FLAG_E         flag
            ES_BORDER_TYPES_E      borderMode,
            ES_U8                     borderValue)
```

【Description】

Perform perspective transformation on the image

【Supported Types】

Input uint8, output uint8.

【Parameters】

Parameter Name	Descriptions	Constraint
deviceId	Device ID value	For detailed information, refer to section 3.4.4 of the ES_AK_DEVICE_E structure
input	Input image data	Only support four-dimensional data operation, in the form of input[batch, height, width, channel]. batch range: [1, 512]; channel can only be 3, i.e., only support 3-channel image data; height, width range: [1, 8192].
output	The user specifies the output image dimensions, and the calculated perspective-transformed image data is calculated and output accordingly.	Only support four-dimensional data operation, in the form of output[batch, height, width, channel]. batch range: [1, 512]; channel can only be 3; height, width range: [1, 8192]
M	Perspective transformation matrix.	Format is M[m11, m12, m13, m21, m22, m23, m31, m32, m33]
flag	Interpolation calculation method.	The current version only supports nearest-neighbor interpolation.
borderMode	Describe the border padding mode.	The current version only support "BORDER_CONSTANT" mode
borderValue	The value used for filling when using BORDER_CONSTANT padding.	

【Return】

Return value	Descriptions
0	Success
others	Failure, see error code 3.5.

3.3.11 ES_AK_DSP_WarpAffine

【Declaration】

```
ES_S32 ES_AK_DSP_WarpAffine(
            ES_AK_DEVICE_E          deviceId,
            ES_TENSOR_S             input,
            ES_TENSOR_S             output,
            ES_S32                  channel_first,
            ES_FLOAT                M[AFFINE_MATRIX_SIZE],
            ES_INTER_FLAG_E        flag
            ES_BORDER_TYPES_E      borderMode,
            ES_U8                    borderValue)
```

【Description】

Perform affine transformation on an image.

【Supported Types】

Input uint8, output uint8.

Input uint16, output uint16.

Input/output image aspect ratio [1/5, 5].

【Parameters】

Parameter Name	Descriptions	Constraint
deviceId	Device ID value	For detailed information, refer to section 3.4.4 of the ES_AK_DEVICE_E structure
input	Input image data.	Only supports four-dimensional data: Batch value range: [1] Channel={1,3}; Height, width range: see note for details

output	The user specifies the output image dimension and calculates the output image data after affine transformation	Only supports four-dimensional data:
		Batch value range: [1];
		Channel={1,3};
		Height, width range: see note for details

channel_first	Meaning of data dimensions	The optional configuration options are: 0, 1. The meaning is as follows: 0: Dimension meaning [batch, height, width, channel]; 1: Dimension means [batch, channel, height, width]
M	Affine transformation matrix.	Format is M[m11, m12, m13, m21, m22, m23] Scaling factor part>1/10
flag	Interpolation calculation method.	Supports bilinear interpolation: ES_INTER_LINEAR; Support nearest neighbor interpolation: ES_INTER_NEAREST

Parameter Name	Descriptions	Constraint
borderMode	Border padding Mode.	The current version only support “BORDER_CONSTANT“ mode
borderValue	The value used for filling when using BORDER_CONSTANT padding.	Currently only supports filling in 0

【Return】

Return value	Descriptions
0	Success
others	Failure, see error code 3.5.

【Note】

Due to the constraints of hardware space, the size of input and output images is limited to a certain extent. Due to the complex constraints, it is difficult to provide a strictly executable size range. The image size is mainly limited by the following aspects:

- Image bit count (8bit/16bit);
- Number of image channels (single channel/triple channel);
- Image channel format (NCHW/NHWC), different formats have some additional space consumption;
- Output image size;
- Input/output image aspect ratio; Output image size;
- Affine transformation matrix, mainly consisting of scaling and rotation components;

In addition, there is no constraint on the original size of the input image. What is constrained is the size of the effective area of the output image in the input image under the action of the affine transformation matrix Minv (the inverse matrix of M). Here are some examples under classical conditions, and for other cases, please pay attention to the return value of the operator for its feasibility.

The classic example is as follows:

1.Examples of uint8 and single channel images:

Single channel images do not distinguish between front and back channel formats, with consistent size constraints.

input_shape: "[1,2200,2200,1]"
output_shape: "[1,2200,2200,1]"

M:Rotation angle 45° scaling factor 1.0

2.Examples of uint8、channel_first=1、three channel image:

input_shape: "[1,3,2200,2200]"
output_shape: "[1,3,2200,2200]"

M:Rotation angle 45° scaling factor 1.0

3.Examples of uint8、channel_first=0、three channel image:

input_shape: "[1,2200,2200,3]"
output_shape: "[1,1100,1100,3]"

M:Rotation angle 45° scaling factor 1.0

4.Example of uint16 image:

Reduce the size of the above uint8 image to within half

5.Other situations:

Rotate 45 ° and 135 °, and the area of the affine transformation corresponding to the output image in the input image is the largest. The area corresponding to other angles is smaller than this type of situation, so larger output image sizes are acceptable;

When the output image is reduced compared to the input image (M matrix scaling component<1), the area of the corresponding affine transformation region in the input image will be larger, resulting in a smaller size of the received output image.

3.3.12 ES_AK_DSP_DetectionOut

【Declaration】

```
ES_S32 ES_AK_DSP_DetectionOut(
                ES_AK_DEVICE_E              deviceId,
                ES_TENSOR_S *            inFeaturesPtr,
                ES_S32                    inFeaturesNum,
                ES_TENSOR_S              outBoxInfo,
                ES_TENSOR_S              outBoxNum,
                ES_DET_NETWORK_E detectNet,
                const ES_DETECTION_OUT_CFG *cfg
                )
```

【Description】

Perform post-processing on the output feature map of the detection network, outputting the target class_id, score, and coordinates of the detection box that meet the conditions. This operator is a fusion operator of box_decode and nms.

【Supported Types】

Input float16, outBoxInfo output float16, outBoxNum output int32.

Input float16, outBoxInfo output float32, outBoxNum output int32.

Input int16, outBoxInfo output float16, outBoxNum output int32.

Input int16, outBoxInfo output float32, outBoxNum output int32.

【Parameters】

Parameter Name	Descriptions	Constraint
deviceId	Device ID value	For detailed information, refer to section 3.4.4 of the ES_AK_DEVICE_E structure
inFeaturesPtr	The starting address of the tensor describing the input features, where this address stores inFeaturesNum features.	
inFeaturesNum	The number of input features.	range：[1,9]
outBoxInfo	The output box information.	Only supports 2D operation data, in the format of outBoxInfo [dim0, dim1]. Where dim0 is always 7,

Parameter Name	Descriptions	Constraint
		indicating that each box stores data in the format [batchId, classId, score, coord1, coord2, coord3, coord4]. Based on these 7 values, users can find the corresponding image, object category, score, and the associated box coordinates. The last 4 values represent the box coordinates and support various formats such as xyxy, xywh, etc., which can be configured by users through params. There is no restriction on the value of dim1, which is 'input batch * maxBoxesPerBatch'."
outBoxNum	Number of output boxes for each image.	For one-dimensional data, the width range is [1,512].
detectNet	Name of the detection model	For detailed information, please refer to 3.4.4 enumeration
cfg	Pointer to the configuration structure for detection output parameters	For detailed information, refer to Section 3.4.4 of the ES_DETECTION_OUT_CFG structure.

【Return】

Return value	Descriptions
0	Success
others	Failure, see error code 3.5.

3.3.13 ES_AK_DSP_Argmax

【Declaration】

```
ES_S32 ES_AK_DSP_Argmax(
                ES_AK_DEVICE_E deviceId,
                ES_TENSOR_S     input,
                ES_TENSOR_S     output,
                ES_TENSOR_S     outputIdx,
                ES_S32           k,
                ES_S32           axis)
```

【Description】

Call DSP device for argmax calculation.

【Supported Types】

Input float16, output float16, outputIdx uint16.

Input float16, output float32, outputIdx uint16.

Input float32, output float32, outputIdx uint16.

【Parameters】

Parameter Name	Descriptions	Constraint
deviceId	Device ID value	For detailed information, refer to section 3.4.4 of the ES_AK_DEVICE_E structure
input	Description of the input data, including shape, type, and address, where the address contains the values to be sorted.	Only support four-dimensional data operation, in the form of input[batch, channel, height, width], with values ranging from [1, 8192].
output	Description of the output data, including shape, type, and address, where the address contains the (maximum) sorted values.	Only support four-dimensional data operation, in the form of output[batch, channel, height, width], with values ranging from [1, 8192].
outputIdx	Description of the output data, including shape, type, and address, where the address contains the original indices of the (maximally) sorted values.	
k	The number of data points in the sorted dimension of the output.	range：[1,32]
axis	The specified sorting dimension.	Currently, sorting by the channel dimension is supported.

【Return】

Return value	Descriptions
0	Success
others	Failure, see error code 3.5.

3.3.14 ES_AK_DSP_Softmax

【Declaration】

```
ES_S32 ES_AK_DSP_Softmax(
                ES_AK_DEVICE_E deviceId,
                ES_TENSOR_S       input,
                ES_TENSOR_S       output,
                ES_FLOAT           scaleIn)
```

【Description】

Call DSP devices to perform Softmax calculations.

【Supported Types】

Input int8, output float16.

Input int8, output float32.

Input float16, output float16.

Input int16, output float16.

【Parameters】

Parameter Name	Descriptions	Constraint
deviceId	Device ID value	For detailed information, refer to section 3.4.4 of the ES_AK_DEVICE_E structure
input	Description of the input data, including shape, type, and address, where the address contains the data to be processed.	Only support four-dimensional data operations, in the form of input[batch, channel, height, width], with values ranging from [1, 8192].
output	Description of the output data, including shape, type, and address, where the address contains the output data of softmax.	Only support four-dimensional data operations, in the form of input[batch, channel, height, width], with values ranging from [1, 8192].
scaleIn	Input data quantization factor.	Range: [0.00001, 1.0]. Set to 1.0 when the input is a float.

【Return】

Return value	Descriptions
0	Success
others	Failure, see error code 3.5.

3.3.15 ES_AK_CPU_SimilarityTransform

【Declaration】

```
ES_S32 ES_AK_CPU_SimilarityTransform(
            ES_AK_DEVICE_E deviceId,
            ES_TENSOR_S      srcPoint,
            ES_TENSOR_S      dstPoint,
            ES_TENSOR_S      transformMat)
```

【Description】

Calculate Similarity Transformation Matrix.

【Supported Types】

Input float32, output float32.

【Parameters】

Parameter Name	Descriptions	Constraint
deviceId	Device ID value	For detailed information, refer to section 3.4.4 of the ES_AK_DEVICE_E structure
srcPoints	The source points of the similarity transformation matrix.	Supports only 2D operation data, in the form of srcPoints[dim0, dim1], where dim0 indicates the number of source points, currently limited to 5 points, hence dim0 must be set to 5. dim1 represents the dimension of source point coordinates, which can only be 2, meaning each point is a 2D coordinate like (x, y).
dstPoints	The target points of the similarity transformation matrix.	Same as srcPoints
transformMat	Similarity transformation matrix calculation result.	A matrix of shape [3,2].

【Return】

Return value	Descriptions
0	Success
others	Failure, see error code 3.5.

3.4 Data Types and Data Structures

The definitions of AcceleratorKit related data types and data structures are as follows：

- ES_AK_Capability_S： define the data structure of Accelerator capability information.
- ES_AK_DEVICE_E：define the hardware devices supported by the Accelerator.
- ES_DATA_BUFFER：define the buffer address information for storing data.
- ES_DEV_BUF_S：define the buffer address information on the device side.
- ES_TENSOR_S：Describe the shape, buffer address, and other information of the tensor.
- ES_DET_NETWORK_E：Define the types of detection networks supported by the ES_DSP_DetectionOut operator.
- ES_IOU_METHOD_E：Define the method of calculating Intersection over Union (IoU) supported by the ES_DSP_DetectionOut operator.
- ES_BOX_TYPE_E： Define the output box format supported by the ES_DSP_DetectionOut operator.
- ES_INTER_FLAG_E：Define the interpolation calculation method information for

interpolation-type operators.

- ES_BORDER_TYPES_E：Define enumeration types for boder filling methods.

3.4.1 ES_AK_Capability_S

【Description】

The definition of AcceleratorKit related data types and data structures is as follows.

【Definition】

```
typedef struct AK_Capability_S {
    ES_U64 reserved;
} ES_AK_Capability_S
```

【成员】

Parameter Name	Descriptions
Reserved	Reserved fields.

3.4.2 ES_AK_DEVICE_E

【Description】

Define the hardware devices supported by the Accelerator.

【Definition】

```
typedef enum AK_DEVICE_E {
        ES_AK_DEV_DSP_0 = 0x0,
        ES_AK_DEV_DSP_1 = 0x1,
        ES_AK_DEV_DSP_2 = 0x2,
        ES_AK_DEV_DSP_3 = 0x3,
        ES_AK_DEV_DSP_4 = 0x4,
        ES_AK_DEV_DSP_5 = 0x5,
        ES_AK_DEV_DSP_6 = 0x6,
        ES_AK_DEV_DSP_7 = 0x7,
        ES_AK_DEV_HAE_0 = 0x8,
        ES_AK_DEV_HAE_1 = 0x9,
        ES_AK_DEV_GPU_0 = 0xa,
        ES_AK_DEV_GPU_1 = 0xb,
        ES_AK_DEV_DIE0_CPU = 0xc,
        ES_AK_DEV_DIE1_CPU = 0xd,
        ES_AK_DEV_BUTT
  } ES_AK_DEVICE_E
```

【成员】

Parameter Name	Descriptions
ES_AK_DEV_DSP_0	DSP0 device
ES_AK_DEV_DSP_1	DSP1 device
ES_AK_DEV_DSP_2	DSP2 device
ES_AK_DEV_DSP_3	DSP3 device
ES_AK_DEV_DSP_4	DSP4 device
ES_AK_DEV_DSP_5	DSP5 device
ES_AK_DEV_DSP_6	DSP6 device
ES_AK_DEV_DSP_7	DSP7 device
ES_AK_DEV_HAE_0	HAE0 device
ES_AK_DEV_HAE_1	HAE1 device
ES_AK_DEV_GPU_0	GPU0 devicev
ES_AK_DEV_GPU_1	GPU1 device
ES_AK_DEV_DIE0_CPU	DIE0 CPU
ES_AK_DEV_DIE1_CPU	DIE1 CPU
ES_AK_DEV_BUTT	Maximum number of devices

3.4.3 ES_TENSOR_S

【Description】

Tensor structure parameters。

【Definition】

```
typedef struct {
    union {
            ES_DEV_BUF_S pData;
            void *hostBuf;
        };
    ES_DATA_PRECISION_E dataType;
    ES_U32 shapeDim;
    ES_U32 shape[MAX_DIM_CNT];
} ES_TENSOR_S
```

【成员】

Parameter Name	Descriptions

dataType	The data type of Tensor , Enumeration type of ES_DATA_PRECISION_E：
	ES_PRECISION_INT8 = 1,
	ES_PRECISION_UINT8 = 2,
	ES_PRECISION_INT16 = 3,
	ES_PRECISION_UINT16 = 4,
	ES_PRECISION_INT32 = 5,
	ES_PRECISION_UINT32 = 6,
	ES_PRECISION_INT64 = 7,
	ES_PRECISION_UINT64 = 8,
	ES_PRECISION_FP16 = 9,
	ES_PRECISION_FP32 = 10

pData	Device side buffer start address.
hostBuf	CPU side buffer start address.
shapeDim	Describe the true dimensions of a tensor.
Shape	An array with a length of MAX-DIM-CNT, storing the width information of each dimension of the tensor in order. MAX-DIM-CNT value is 6.

3.4.4 ES_DET_NETWORK_E

【Description】

Detection network types supported by detectionOut operator

【Definition】

```
typedef enum {
    yolov3_u = 0,
    yolov3_d = 1,
    yolov4 = 2,
    yolov5 = 3,
    yolov7 = 4,
    yolov8 = 5
} ES_DET_NETWORK_E
```

[note]

yolov3_d refers to the darknet version, the original yolo v3 version; yolov3_u refers to the ultralytics version, which is currently the most widely used version. Users can choose the corresponding version based on the source of their own model. If they are unsure about the source of the model, it is recommended to first try configuring yolov3_u for testing.

3.4.5 ES_NMS_METHOD_E

【Description】

NMS method supported by detectionOut operator.

【Definition】

```
typedef enum {
    HARD_NMS,
    SOFT_NMS_GAUSSIAN,
    SOFT_NMS_LINEAR
} ES_NMS_METHOD_E
```

3.4.6 ES_IOU_METHOD_E

【Description】

IouIoU method supported by detectionOut operators.

【Definition】

```
typedef enum {
    IOU,
    GIOU,
    DIOU
} ES_IOU_METHOD_E
```

3.4.7 ES_BOX_TYPE_E

【Description】

Box format supported by detectionOut operators.

【Definition】

```
typedef enum {
    XminYminXmaxYmax,
    XminYminWH,
    YminXminYmaxXmax,
    XmidYmidWH
} ES_BOX_TYPE_E
```

3.4.8 ES_INTER_FLAG_E

【Description】

Enumeration types that describe interpolation methods.

【Definition】

```
typedef enum {
    ES_INTER_NEAREST = 0,
    ES_INTER_LINEAR = 1,
    ES_INTER_AREA = 2,
    ES_INTER_CUBIC = 3,
    ES_INTER_LANCZOS4 = 4,
    ES_INTER_NEAREST_EXACT = 6,
    ES_INTER_STRETCH = 20,
    ES_INTER_FILTER = 21
} ES_INTER_FLAG_E
```

3.4.9 ES_BORDER_TYPES_E

【Description】

An enumeration type that describes the border filling method.

【Definition】

```
typedef enum {
    BORDER_CONSTANT = 0,
    BORDER_REPLICATE = 1,
    BORDER_REFLECT = 2,
    BORDER_WRAP = 3,
    BORDER_REFLECT_101 = 4,
    BORDER_TRANSPARENT = 5,
    BORDER_ISOLATED = 16
} ES_BORDER_TYPES_E
```

3.4.10 ES_DETECTION_OUT_CFG

【Description】

Definition of the configuration structure for detection output parameters.

【Definition】

```
typedef struct {
    ES_S32 anchorsNum;
    ES_FLOAT anchorScale[MAX_TOTAL_ANCHORS_NUM * 2];
    ES_S32 imgH;
    ES_S32 imgW;
    ES_S32 clsNum;
    ES_FLOAT inputScale[MAX_IN_TENSOR_NUM];
    ES_NMS_METHOD_E nmsMethod;
    ES_IOU_METHOD_E iouMethod;
    ES_BOX_TYPE_E outBoxType;
    ES_BOOL coordNorm;
    ES_S32 maxBoxesPerClass;
    ES_S32 maxBoxesPerBatch;
    ES_FLOAT scoreThreshold;
    ES_FLOAT iouThreshold;
    ES_FLOAT softNmsSigma;
    ES_FLOAT effecImgOffsetX;
    ES_FLOAT effecImgOffsetY
} ES_DETECTION_OUT_CFG
```

【成员】

参数	描述
anchorsNum	Number of anchor boxes used in the detection network.
anchorScale	Array of anchor scales, defined as pairs of width and height ratios.
imgH	Height of the input image (in pixels).
imgW	Width of the input image (in pixels).
clsNum	Number of object classes for detection.
inputScale	Array of scales for normalizing the input tensor.

参数	描述
nmsMethod	Non-maximum suppression (NMS) method used to filter overlapping detections.
iouMethod	Intersection over Union (IOU) method used in NMS calculations.
outBoxType	Type of the output bounding box format (e.g., corner coordinates or center coordinates).
coordNorm	Boolean flag indicating whether the bounding box coordinates are normalized to [0, 1].
maxBoxesPerClass	Maximum number of bounding boxes retained per class after NMS.
maxBoxesPerBatch	Maximum number of bounding boxes retained for the entire batch after NMS.
scoreThreshold	Minimum score threshold to retain a detection.
iouThreshold	Intersection over Union (IOU) threshold used in NMS for filtering overlapping boxes.
softNmsSigma	Sigma value for Soft NMS used to adjust the scores of overlapping boxes.
effecImgOffsetX	Effective offset in the X direction for adjusting bounding box coordinates.
effecImgOffsetY	Effective offset in the Y direction for adjusting bounding box coordinates.

3.5 Error Code

Parameter Name	Value	Descriptions
ES_AK_ERROR_INVALID_DEVID	0xA0146029	The device ID exceeds the legal range.
ES_AK_ERROR_ILLEGAL_PARAM	0xA014602A	Illegal parameter.
ES_AK_ERROR_LOAD_OP_FAILED	0xA014602B	Loading operator failed.
ES_AK_ERROR_SUBMIT_TASK_FAILED	0xA014602C	Task submission failed.
ES_AK_ERROR_QUERY_FAILED	0xA014602D	Query timeout.
ES_AK_ERROR_NOT_SUPPORT	0xA014602E	Unsupported parameters or functions.
ES_AK_ERROR_OUTBUF	0xA014602F	Too many buffers applied.
ES_AK_ERROR_GET_OP_FUNC_FAILED	0xA0146030	Failed to obtain operator execution function.
ES_AK_ERROR_OP_EVAL_FAILED	0xA0146031	Operator execution failed.
ES_AK_ERROR_OPEN_FILE_FAILED	0xA0146032	File open failed.

Parameter Name	Value	Descriptions
ES_AK_ERROR_READ_FILE_FAILED	0xA0146033	File read failure.
ES_AK_ERROR_WRITE_FILE_FAILED	0xA0146034	File write failure.
ES_AK_ERROR_INVALID_VER	0xA0146035	File write failure.

3.6 Debugging Tips

Users can monitor the current execution status of the software through the following log information, which helps to debug issue.

- invalid vec size： The shape field is empty or cannot be parsed properly.
- The operator registration list is empty, check that compiler.a or manager.so is linked correctly： The operator list is empty and not correctly linked to compiler. a and manager. so. It is necessary to link to these two library files in cmake or other forms, which contain registered operators.
- No candidate operators for xxx： The paramters of operator are invalid, or no matching operator been found.
- Failed to emit operator description： Launch operator failed, possibly due to parameter validation failure; If it is a custom operator, it may be because the operator description pointer is empty (not set correctly) or the buffer of the param exceeds the upper limit (32).
- Failed to initialize buffers： The fd configuration of Input, Output, or Param is invalid. Please check the value of buffers for input, output, and param.
- Failed to load operator： Failed to load the specified operator. Please check if the DSP operator library is properly compiled; Or check if the operator name emitted by Emit meets the requirements.
- Failed to submit task for operator： Execution failed, check the print information of driver and framework.

3.7 Notification

3.7.1 Parameter Config

Before calling the Accelerator kit interface, it is necessary to configure the respective config structure of the operator as an input parameter. For example, before invoking the argmax operator, it is necessary to configure the ES-DSP-Argmax structure.

In addition, the data shape and type of inputs and outputs also need to be configured in advance, and the parameters must meet the interface requirements, otherwise, it failed on validation.