Safety Critical Extension
v1.1

Jesse Villarreal | Texas Instruments
Introduction

• **Objective:**
  • *Facilitating* implementation of OpenVX into Safety Critical Applications
    • Does not guarantee implementation is safe.

• Software written for safety critical applications require rigorous demands on the deployment software and development processes (e.g. **ISO26262**)

• V-Model for safety critical development processes
Requirements

- Annotated OpenVX spec with Functional Requirement tag numbers
  - Each requirement is identified with a [R#####] tag

```c
vxCreateRemap()

vx_remap VX_API_CALL vxCreateRemap (

  vx_context context,
  vx_uint32 src_width,
  vx_uint32 src_height,
  vx_uint32 dst_width,
  vx_uint32 dst_height )
```

Creates a remap table object [R01166].

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>context</code></td>
<td>The reference to the overall context [R01167].</td>
</tr>
<tr>
<td><code>src_width</code></td>
<td>Width of the source image in pixels [R01168].</td>
</tr>
<tr>
<td><code>src_height</code></td>
<td>Height of the source image in pixels [R01169].</td>
</tr>
<tr>
<td><code>dst_width</code></td>
<td>Width of the destination image in pixels [R01170].</td>
</tr>
<tr>
<td><code>dst_height</code></td>
<td>Height of the destination image in pixels [R01171].</td>
</tr>
</tbody>
</table>

- Aids in traceability from requirements, to design, to implementation, to testing
- Each requirement must be tested
Implementation

• Updated OpenVX headers to be MISRA-C compliant
  • Change enums to #defines
  • Changes to vx_pixel_value_t
    • This is a union, with no variant tag. It is not portable across architectures, and requires a variant tag to specify the type of the contents.

• Revision of some API’s
  • Those that pass pointers to objects with no defined size
Verification

- Safety-critical software development has a much higher testing and verification cost.

- Software tools that are used for development have less stringent requirements than deployed software in a safety critical system
  - Development environment tool: Functional correctness
  - Safety critical environment SW: Functional correctness, real-time performance, fault tolerance, error condition recovery, deterministic etc.
  - e.g. A tool can crash without killing people

- SC spec describes a distinction between *development* and *deployment* feature sets (2.19), and utilizes the *import/export framework* (2.18) to bridge the two.
  - Not all of OpenVX needs to be used in the safety critical environment software
  - Reducing features to only what is needed in the safety critical environment may reduce costs and development time.
Development & Deployment Feature Sets

Development Environment

Create and verify OpenVX graph

Export all the objects that needs access during deployment

Release all objects

Deployment Environment

Create context

Import objects from binary

Graph execution

Release all objects

Vendor / 3rd Party Tools

OpenVX Binary in Vendor Specific Format
Development & Deployment Feature Sets

• **Needed for deployment:**
  - Graph import
  - Creating and accessing graph input/output data objects
  - Processing graphs

• **Not needed for deployment (Development only):**
  - vxu immediate kernels
  - Graph construction
    - All node creation APIs for each of the vision kernels
  - Graph verification
    - System mapping and optimization decisions which typically occur during graph verification may be decided and fixed in the development feature set
  - Logging & Performance features
  - Graph export functions
Development & Deployment Feature Sets

- Spec identifies which APIs are in deployment feature set
- Example (sample from spec, which contains full list):

<table>
<thead>
<tr>
<th>API Group</th>
<th>API Function</th>
<th>Is deployment feature?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>vxSetParameterByReference</td>
<td></td>
</tr>
<tr>
<td></td>
<td>vxQueryParameter</td>
<td></td>
</tr>
<tr>
<td>SCALAR</td>
<td>vxCreateScalar</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>vxReleaseScalar</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>vxQueryScalar</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>vxCopyScalar</td>
<td>Yes</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>vxQueryReference</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>vxReleaseReference</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>vxRetainReference</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>vxSetReferenceName</td>
<td>Yes</td>
</tr>
<tr>
<td>DELAY</td>
<td>vxQueryDelay</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>vxReleaseDelay</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>vxCreateDelay</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>vxGetReferenceFromDelay</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>vxAgeDelay</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>vxRegisterAutoAging</td>
<td></td>
</tr>
<tr>
<td>LOGGING</td>
<td>vxAddLogEntry</td>
<td></td>
</tr>
<tr>
<td></td>
<td>vxRegisterLogCallback</td>
<td></td>
</tr>
</tbody>
</table>
Summary

• Annotated OpenVX spec with Functional Requirement tag numbers

• Made header files MISRA-C compliant

• API definition revisions

• Defined Development vs Deployment Feature sets

• Added Import Export Extension as part of SC Extension
Next Steps

• Considering merging OpenVX SC into main OpenVX 1.3 spec
  • No longer maintain a separate spec