Khronos Safety Critical Advisory Forum

AutoSens Brussels 2019

Stephane Strahm (Kalray)
September 2019
Why do we need Safety Critical Open Standards?

Standards Grow Markets...
By reducing consumer confusion and increasing usability

... Reduce Costs ...
By sharing development between many companies and driving volume

... Accelerate Time to Market
With well-proven testing and interoperability
Automotive Safety Standards
Current and emerging

Automotive Safety Standards - current and emerging

- IEEE P700x Ethical Life Cycle
- PAS/ISO 21448 SOTIF
- ISO 26262: 2010
- ISO 26262: 2018
- PAS/ISO 21434 Automotive Cyber Security
- SAEJ 3061 Cyber Security
- UL 4600 Standard for Safety Cases
- Regulations
- Environmental standards
Market Needs

- MAIN FUNCTIONS
  - Awareness of environment
  - Decision making
  - Compute capability
  - Algorithms acceleration
  - Sensor raw data analysis
  - Image analysis
  - Path planning

- ESTIMATED PERFORMANCE
  - L2: 10-20 TOPS
  - L3: 20-100 TOPS
  - L4-5: 100-300+ TOPS

- KEY PRODUCTION CRITERIA
  - Performance per Watts
  - Scalability

MANDATORY ELEMENTS
- Safety
- Security
System Overview and Scope
Automotive compute stack

Typical Automotive ADAS / autonomous application(s)

- ADAS
- Sensor fusion
- DNN
- HUD

One software API

Abstraction layer

#1 HW  #2 HW  #3 HW  #n HW

GPU DSP FPGA NSP CPU
other acceleration HW

Automotive heterogeneous platforms

AI CNN stack

Gfx / compute
Automotive compute stack

Typical Automotive ADAS / autonomous application(s)

What is in the middle?

Abstraction layer

One software API

ADAS Sensor fusion DNN HUD

#1 HW #2 HW #3 HW #n HW

GPU DSP FPGA NSP CPU
other acceleration HW

Automotive heterogeneous platforms
Automotive compute stack

Illustration of the stacks based on Khronos Open Standard

Open Standard API

Abstraction layer

#1 HW  #2 HW  #3 HW  #n HW

AI CNN stack

Gfx / compute

Automotive heterogeneous platforms

GPU DSP FPGA NSP CPU other acceleration HW
What is An Open Standard?

**Bad Standards**
- Overprescribes implementation details
- Forces everyone to implement a lowest common denominator
- Stifles innovation
- -> Commoditization

**Good Standards**
- Prescribes only interoperability
- Enables implementation diversity
- Encourages innovation
- -> Differentiation

A truly OPEN standard
- Is not controlled by a single company - but by the whole industry
- Is freely available to use by any company without royalty payments
What is Khronos?

Today

• A non profit member funded consortium focused on creating royalty free open standards for parallel compute, graphics and dynamic media

• 150+ members

• Khronos members are able to contribute to the development of Khronos API specifications
Khronos Mission

Khronos is an open, member-driven industry consortium developing royalty-free standards, and vibrant ecosystems, to harness the power of silicon acceleration for demanding graphics rendering and computationally intensive applications.
Active Khronos Standards

- **High Performance 3D Graphics**
  - Vulkan
  - OpenGL ES
  - OpenGL
  - OpenGL SC
  - WebGL

- **3D Asset Authoring and Delivery**
  - glTF
  - COLLADA

- **Portable XR – Virtual and Augmented Reality**
  - OpenXR

- **Parallel Computation, Vision, Machine Learning and Inferencing**
  - OpenCL
  - SYCL
  - SPIR
  - OpenVX
  - NNEF

**Safety Critical Forum**: Creating API design guidelines for markets requiring safety certification
Active Khronos Standards

HIGH PERFORMANCE 3D GRAPHICS

3D ASSET AUTHORING AND DELIVERY

PORTABLE XR – VIRTUAL AND AUGMENTED REALITY

PARALLEL COMPUTATION, VISION, MACHINE LEARNING AND INFERENCEING

SAFETY CRITICAL FORUM: Creating API design guidelines for markets requiring safety certification
The compute stack - fragmentation

Open Standards can give a clear path through the stack

Level of abstraction

Device implementation

Higher-level language enabler

C-level programming

C++-level programming

Graph programming

*Not exhaustive, for demonstration only
The compute stack - fragmentation

Open Standards can give a clear path through the stack

Device implementation

Higher-level language enabler
- NVIDIA PTX
- HSA
- OpenCL SPIR
- SPIR-V

C-level programming
- OpenCL C
- DSP C
- MCAPI/MTAPI

C++-level programming
- CUDA
- HCC
- C++ AMP
- SYCL

Graph programming
- Halide
- OpenCV
- OpenVX
- VisionCpp
- TensorFlow
- Caffe

*Not exhaustive, for demonstration only
The compute stack - fragmentation

Open Standards can give a clear path through the stack

Level of abstraction

Open Standards

Device implementation

Higher-level language enabler
- NVIDIA PTX
- HSA
- OpenCL SPIR
- SPIR-V

C-level programming
- OpenCL C
- DSP C
- MCAPI/MTAPI

C++-level programming
- CUDA
- HCC
- C++ AMP
- SYCL

Graph programming
- Halide
- OpenCV
- OpenVX
- VisionCpp
- TensorFlow
- Caffe

Support
- Machine Intelligence

*Not exhaustive, for demonstration only
Safety Critical Open Standards

Khronos Safety Critical Advisory Forum

(KSCAF)
Khronos Safety Critical Advisory Forum

The SC Forum

Deliverables

Document

SC API features: Guidelines

Document

SC API features: Requirements
Khronos Safety Critical Advisory Forum

- SC API features: Guidelines
- SC API features: Requirements
- Internal + External Experts
- Change management
  - Git repository
- Open source
- Document
  - PDF
  - HTML

- Create new items
- Amend items

© The Khronos® Group Inc. 2019 - Page 21
Khronos Safety Critical Advisory Forum

Khronos experts from:
- Semiconductor
- Image processing
- Parallel data compute processing
- ....

Khronos invited experts from:
- Medical
- Automotive
- Avionics
- ....

Open Standard Safety Critical APIs

Encourages familiarity, innovation and SC ratified

Can lead to low entry cost, efficiency, flexibility, reduce time
**Illustration: SC Guidelines**

<table>
<thead>
<tr>
<th>Guideline</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>List of SC RTOS and their unique features</td>
</tr>
<tr>
<td>2</td>
<td>Memory management</td>
</tr>
<tr>
<td>3</td>
<td>Conformance testing</td>
</tr>
<tr>
<td>4</td>
<td>The SC API</td>
</tr>
<tr>
<td>5</td>
<td>Explicit timeout values</td>
</tr>
<tr>
<td>6</td>
<td>MISRA compliance</td>
</tr>
<tr>
<td>7</td>
<td>Real Time</td>
</tr>
<tr>
<td>8</td>
<td>Scheduling</td>
</tr>
<tr>
<td>9</td>
<td>Debug Functionality</td>
</tr>
</tbody>
</table>
### 3.2.1. List of SC RTOS and their unique features

This section lists some of the unique features some operating systems support and how those features can result in unintended behavior. This list is not exhaustive or complete, it is meant as a guide and place to share issues which were caused by the unique features of operating systems.

**Table 3.2.1: List of RTOS**

<table>
<thead>
<tr>
<th>RTOS Name</th>
<th>Vendor</th>
<th>Unique Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeOS</td>
<td>DDC-I</td>
<td>Time Partitioning</td>
</tr>
<tr>
<td>Integrity</td>
<td>Green Hills Software</td>
<td>Time Partitioning</td>
</tr>
<tr>
<td>Integrity-178</td>
<td>Green Hills Software</td>
<td>ARINC 653 Time Partitioning</td>
</tr>
<tr>
<td>Lynx</td>
<td>LynxOS</td>
<td></td>
</tr>
<tr>
<td>Lynx</td>
<td>LynxOS 178</td>
<td>ARINC 653 Time Partitioning</td>
</tr>
<tr>
<td>Nucleaus</td>
<td>Mentor Graphics</td>
<td></td>
</tr>
<tr>
<td>QNX</td>
<td>QNX</td>
<td></td>
</tr>
<tr>
<td>ThreadX</td>
<td>Express Logic</td>
<td></td>
</tr>
<tr>
<td>VxWorks</td>
<td>Wind River</td>
<td></td>
</tr>
<tr>
<td>VxWorks 653</td>
<td>Wind River</td>
<td>ARINC 653 Time Partitioning</td>
</tr>
</tbody>
</table>
## Illustration: SC Requirements

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Memory management</td>
</tr>
<tr>
<td>2</td>
<td>Deterministic behavior</td>
</tr>
<tr>
<td>3</td>
<td>Asynchronous Calls</td>
</tr>
<tr>
<td>4</td>
<td>Notification of Change of State</td>
</tr>
<tr>
<td>5</td>
<td>Garbage Collection Methods</td>
</tr>
<tr>
<td>6</td>
<td>Fully Testable</td>
</tr>
<tr>
<td>7</td>
<td>Undefined Behavior</td>
</tr>
</tbody>
</table>
3.2.3. Asynchronous Calls

Asynchronous calls are those which are initiated by the program but may not execute or use their parameter data until a later time. Safety Critical APIs shall be clearly define when any parameter data is used, especially data which is passed via reference or pointer. When pointers are used for output parameters to Asynchronous functions the API shall clearly define when the data is copied to the output parameter. If the output data is populated after an asynchronous event the API shall define a means by which the program can check to validate the data has been written to the output. If a pointer is used for an input to the function the API shall define when the data is used and when that buffer may be reused by the application.

The API shall also define when the parameters can be changed and reused by the program. When API internal data structures are used by a program the life of the variable shall be defined; including when the internal data structure can be reused, if the data must be in the data structure prior to a specific call, and if the data is changed in a way that the application can view the change.

All asynchronous functions shall provide a means to allow the application to determine completion of the function.
Khronos Safety Critical Advisory Forum benefits

- Wider range of development experiences to draw from and capture
- Non Safety Critical (SC) developers are advised on SC issues they may be not aware of
- More robust and compatible SC API
- SC APIs are compatible with a range of functional safety standards
- SC guidelines and requirements ‘knowledge base’ always available
- Internal experience complemented with real use cases
Your benefits

• Help your safety case

• More confidence the API or implementation is compatible with your functional safety demands - less gaps

• Help accelerate the availability of the SC API to implementers which in turn makes it available to you

• May enable higher safety critical level, less decomposition required

Cost

• Experienced practitioners are invited to apply for Advisory Forum membership, at no cost

• Medium to long term investment

• A few hours a week from a safety functional engineer, senior SC software engineer or cybersecurity expert.
Current Khronos SC APIs in 'development':
- OpenCL Safety Critical
- Vulkan Safety Critical

In the future we are likely to see
- Thin and predictable graphics and compute for SC systems

What is in it for us?
Some of the past participants in the forum representing SC domains:

- ISO 26262 – Automotive
- DO-178C - Software for avionics
- DO-254 - Hardware for avionics

17 participated in the forum representing:

- Safety functional managers
- Automotive related software development using graphics APIs
- Semiconductor for automotive and CoS
- Safety critical software for vision processing, machine learning / AI, 3D, compute applications
- Software developers
Khronos Safety Critical Advisory Forum

- You can be involved in this forum on invitation, for Free!

- https://www.khronos.org/advisors/kscaf
THANK YOU!