Powering the Future of Embedded Vision

Embedded World
March 2023

Laurent Pinchart
Founder & CEO, Ideas on Board
Kamaros Working Group Member
Laurent Pinchart is the founder and CEO of Ideas on Board, a company specialized in delivering camera and display solutions for Linux across all markets.

Making cameras work

With 20 years of experience as a Linux kernel developer and maintainer, Laurent has driven the design of the Linux kernel camera API and has participated in multiple industry working groups to standardize camera protocols. Most recently, he has started the libcamera® project to give Linux a full camera stack in collaboration with silicon vendors and OEMs.
Topics

• Introduction to Khronos and its family of open standard APIs
• Overview of Khronos APIs for Parallel Computation
• Khronos and Safety Critical APIs
• Introduction to the new Kamaros Embedded Camera API in development
• How to get involved!
Khronos Connects Software to Silicon

Open, royalty-free interoperability standards to harness the power of GPU, XR and multiprocessor hardware

3D graphics, augmented and virtual reality, parallel programming, inferencing and vision acceleration

Non-profit, member-driven standards organization, open to any company

Proven multi-company governance and Intellectual Property Framework

Founded in 2000
~ 200 Members with ~ 40% US, 30% Europe, 30% Asia
Khronos Active Standards

- **3D Graphics**
  - Desktop, Mobile and Web
  - Vulkan
  - ANARI
  - OpenGL
  - OpenGL ES
  - WebGL

- **3D Assets**
  - Authoring and Delivery
  - glTF
  - COLLADA
  - 3DCommerce

- **Portable XR**
  - Augmented and Virtual Reality
  - OpenXR
  - OpenVX
  - NNEF
  - SYCL
  - SPIR

- **Parallel Computation**
  - Vision, Camera, Inferencing, Machine Learning
  - OpenCL
  - Kamaros
  - OpenVX
  - NNEF

- **Safety Critical APIs**
  - OpenGL SC
  - Vulkan SC
Need for Vision Acceleration Standards

**Increasing Sensor Compute Load**
Diverse camera and sensor arrays feed sophisticated processing - including inferencing

**Advanced User Interfaces**
High quality 3D graphics, augmented reality, diverse display systems

Cost and time to integrate and utilize sensors, GPUs and processors in diverse markets has become a major constraint on innovation and efficiency

Open Standard APIs in Embedded Markets

Enable cross-platform software reusability
Decouple software and hardware for easier development and integration of new components
Provide cross-generation reusability
Facilitate field upgradability
Khronos Compute Acceleration Standards

Increasing industry interest in parallel compute acceleration to combat the ‘End of Moore’s Law’
Growing Need for APIs for Functional Safety

Demand for advanced GPU-accelerated graphics and compute is growing in an increasing number of industries where safety is paramount, such as automotive, autonomy, avionics, medical, industrial, and energy.

Safety-critical APIs are designed to reduce system-level safety-critical certification effort and costs:
1) Streamlined to reduce documentation and testing surface area
2) Deterministic behavior to simplify system design and testing
3) Unambiguous and comprehensive fault handling

Industry safety-critical standards include:
- RTCA DO-178C (avionics)
- ISO 26262 (automotive)
- IEC 61508 (industrial)
- IEC 62304 (medical)
Khronos Safety Critical Standards Evolution

OpenGL ES 1.0 - 2003
Fixed function graphics

OpenGL ES 2.0 - 2007
Programmable Shaders

OpenGL SC 1.0 - 2005
Fixed function graphics
Safety-critical subset

OpenGL SC 2.0 - 2016
Programmable Shaders
Safety-critical subset

Vulkan 1.2 - 2020
Explicit Graphics and Compute
and Display

Vulkan SC 1.0 - 2022
Explicit Graphics, Compute and
Display safety-critical subset

OpenVX SC Extension – 2017
Graph-based vision and inferencing

OpenVX 1.3 - 2019
SC Extension Integrated into
core OpenVX specification

SYCL 2020
C++-based heterogeneous
parallel programming

SYCL SC Working Group announced to
develop C++-based heterogeneous
parallel compute programming
framework for safety-critical systems

March 2022

Khronos has 20 years experience in
standards for safety-critical markets
Leveraging proven mainstream standards with
shipping implementations and developer
tooling and familiarity
A choice of abstraction levels to suit different
markets and developer needs
The Need for a Camera System API Standard

Increasing Sensor Diversity
Including camera arrays and depth sensors such as Lidar

Multiple Sensors Per System
Synchronization and coordination become essential

The cost and time to integrate and utilize sensors in embedded systems has become a major constraint on innovation and efficiency in the embedded vision market

Increasing Sensor Processing Demands
Including inferencing. Sensor outputs need to be flexibly and efficiently generated and streamed into acceleration processors

Proprietary APIs Hinder Innovation
Vendor-specific APIs to control cameras, sensors and close-to-sensor ISPs prevent rapid integration of new technologies
Benefits of Embedded Camera API Standard

An effective open, cross-vendor open standard for camera, sensor and ISP control could provide multiple benefits

- Cross-vendor portability of camera/sensor code for easier system integration of new sensors
- Preservation of application code across multiple generations of cameras and sensors
- Sophisticated control over sensor stream generation increases effectiveness of downstream accelerated processing

Development of Camera and sensor APIs may also generate new requirements for downstream vision and inferencing acceleration APIs
Throughout 2021 EMVA and Khronos hosted an open cross-industry Exploratory Group to consider the needs for a camera API and how they should be addressed.

- **33** Weekly Calls
- **33** Presentations
- **73** Companies
- **107** Unique Attendees
- **23** Use Cases
- **60** Requirements
- **1** Glossary
- **1** Statement of Work

Scope of Work document captures generated consensus on terminology, scope, requirements, and design methodology.

Camera System API Working Group
Scope of Work
December 2021
Approved by Exploratory Group vote on 8th December 2021

This document has been produced and agreed by the Embedded Camera Exploratory Group jointly hosted by the EMVA and The Khronos Group which met over the period of March to December 2021.

Created under the Khronos New Initiative Process, this document identifies the industry need, and outlines the scope, requirements, and design methodology for a new open standard Camera System API. Deliverables and proposed timeline for the standard's design and deployment by a working group are discussed.

SOW is downloadable [here](#)
What is Kamaros?

EMBEDDED CAMERA SYSTEM API

Jointly promoted by Khronos and the European Machine Vision Association (EMVA), the Kamaros API Working Group is now developing an open, royalty-free standard for controlling camera system runtimes in embedded, mobile, industrial, XR, automotive, and scientific markets.
The Timeline So Far

**Camera API Exploratory Group**

In response to industry requests the EMVA and Khronos create a group to explore industry interest for an open camera API standards, and consensus on use cases and requirements.

Over 70 companies join and contribute to the discussions and approve the proposed scope of work.

**Scope of Work**

Exploratory group publishes the scope, requirements and design methodology for a new open standard Camera System API.

Kamaros

Kamaros name adopted for Working Group and API.

**Camera Working Group**

Working Group formed under the Khronos membership and IP framework. Work starts on the detailed specification of the API, guided by the scope of work.

**Draft Spec**

Expected release of draft 1.0 specification for community feedback.
Typical Kamaros Software Stack

Frameworks & Middleware
- GStreamer
- OpenVX
- Libraries

Kamaros API
- Kamaros System Runtime

Transport
- CSI-2
- USB
- Ethernet

Physical Devices
- Sensors
- Lenses
- Lights
- Processors

Names of transport layers, framework and operating systems are illustrative examples.
Primary Design Influences and Inputs

Kamaros will fill an ecosystem gap for a cross-vendor embedded camera system API

May complement existing APIs
e.g., be used in their implementation, or be implemented over them

Android Camera2 Framework API

Hosted by EMVA including GenTL, SFNC (Standard Features Naming Convention) and PFNC (pixel format naming convention)

Open-source camera stack and framework for Linux, Android, and ChromeOS

libArgus API for acquiring images and associated metadata from cameras
Design Principles

- Physical Devices = queryable and controllable via a Device ID:
- Logical Device = set of Devices queried and controlled via a single Device ID
- Frame = Image + Metadata accessed via Frame ID
- Stream = sequence of Frames
- Camera = a Logical Device that exports one or more Streams from the Camera System

OS-agnostic
  - Including support for minimal embedded OS

Multiple Language Bindings
  - Object-oriented API enabling efficient bindings to C, C++, Python, and other programming languages

Loadable Layers
  - Command dispatch system to enable developers to use installable layers for validation, profiling, and debugging, etc.
Leveraging Vulkan?

Applications, Frameworks & Middleware

The Kamaros WG is exploring the option of additionally providing a lower-level API to ease integration of sensors and ISPs in a Kamaros System Runtime, by leveraging applicable concepts from the Vulkan API.

An open-source reference system runtime could be provided by the Kamaros WG or by a third-party project.

A layered approach enables a wider range of use cases.

A componentized architecture allows each vendor to focus on their own components.

Easy integration with Vulkan and other Khronos APIs.

Low-Level API

Sensor > Raw VkImage > ISP > RGB/YUV VkImage

Kamaros™ Low-Level Runtime

Applications, Frameworks & Middleware

Kamaros™ System Runtime

Camera Command Queue

ISP Command Queue

3A Algorithms

Application API

Low-Level API

Sensor

ISP

Camera

ISP

Command
Queue

Command
Queue

Sensor

ISP

Raw VkImage

RGB/YUV VkImage

Easy integration with Vulkan and other Khronos APIs.

© The Khronos® Group Inc. 2023 - Page 18

This work is licensed under a Creative Commons Attribution 4.0 International License
Kamaros Deliverables

The API shall be made openly available to the industry under royalty-free licensing terms as defined by the Khronos Intellectual Property (IP) Framework

Working Group Deliverables

- Camera System API specification for use by implementers and developers
- Central extension namespace registry for Working Group and vendor extensions
- Open-source conformance test suite, including a precise definition of conformance
- Adopters Program to enable implementations to become officially conformant
- Trademark and logo for promotion and use on conformant implementations
- A conformant portable open-source sample implementation of the API
- Open-source samples and documentation
- Open-source SDK, tools and Libraries
Kamaros Working Group Organization

Any company is welcome to join Khronos for access to any Khronos Working Groups.

EMVA Members - reach out to EMVA for details on joining the Kamaros Advisory Panel.
Get Involved!

Any company is welcome to join Khronos to influence standards development
https://www.khronos.org/members/ or email memberservices@khronosgroup.org

More information on any Khronos APIs
https://www.khronos.org/

Khronos members can participate in the Kamaros Camera Working Group
EMVA Members can join the Kamaros Advisory panel
https://www.khronos.org/kamaros

Khronos is developing a growing family of open, royalty-free API standards
relevant to embedded and safety-critical markets
Compute and Safety Critical Background
Vulkan New Generation 3D Graphics and Compute

Vulkan is the only open standard modern, cross-platform GPU API

Advantages:
- Simpler drivers
- Direct GPU Control
- Multiple graphics, command and DMA queues
- Multiple offline language compilers

A Graphics API

A GPU API

OpenGL

OpenGLES

Vulkan

High-level Driver Abstraction
- Layered GPU Control
- Context management
- Memory allocation
- Full GLSL compiler
- Error detection

Thin Driver
- Explicit GPU Control

Loadable debug and validation layers

Multiple Front-end Compilers
- GLSL, HLSL etc.

Multiple offline language compilers

Single thread per context

Memory allocation
- Thread management
- Explicit Synchronization
- Multi-threaded generation of command buffers
OpenCL 3.0 Adoption and Evolution

Programming and Runtime Framework for Application Acceleration

- Offload compute-intensive kernels onto parallel heterogeneous processors
- CPUs, GPUs, DSPs, FPGAs, Tensor Processors
- OpenCL C and C++ kernel languages
- OpenCL 3.0 is cleanly extensible baseline
- C++ for OpenCL compiler for C++17 kernels
- Vulkan/OpenCL Interop (provisional)
- Asynchronous DMA for embedded platforms

Example future OpenCL 3.0 extensions under consideration at Khronos

- Command Buffer Record/Replay
- Unified Shared Memory
- Floating Point Atomics
- Image Tiling Controls
- YUV Multi-planar Images
- Cross-workgroup Barriers
- External Memory Export
- Cooperative Matrices
- Timeline Semaphores
- Generalized Image from buffer
- 32 and 64-length vectors
- Indirect Dispatch

OpenCL 3.0 Adopters

Product Conformance Status

- arm
- Google
- Intel
- NVIDIA
- QUALCOMM
- codeplay
- Imagination
- Microsoft
- QNX
- VeriSilicon

OpenCL 3.0 Adopters Shipping Conformant Implementations
SYCL Single Source C++ Parallel Programming

C++ Kernel Fusion can give better performance on complex apps and libs than hand-coding.

Accelerated code passed into device OpenCL compilers.

Complex ML frameworks can be directly compiled and accelerated.

C++ templates and lambda functions separate host & accelerated device code.

SYCL accelerates C++-based engines and applications with performance portability.

SYCL 2020 Launched February 2021
Closer alignment with C++17
Smaller code size, faster performance

New Features
Unified Shared Memory | Parallel Reductions | Subgroup Operations | Class template Argument Deduction
OpenVX Cross-Vendor Vision and Inferencing

High-level graph-based abstraction for portable, efficient vision processing

- Optimized OpenVX drivers created, optimized and shipped by processor vendors
- Implementable on almost any hardware or processor with performance portability
- Graph can contain vision processing and NN nodes for global optimization
- Run-time graph execution need very little host CPU interaction

OpenVX Graph

Vendors optimize and ship drivers for their platform
Full list of conformant OpenVX implementations here:
https://www.khronos.org/conformance/adapters/conformant-products/openvx

ONNX
TensorFlow
TensorFlowLite
Caffe
Caffe2
Open-Source Convertors

NNEF Import converts a trained Neural Network into OpenVX Graph
Layers are represented as OpenVX nodes

Open-Source Projects

This work is licensed under a Creative Commons Attribution 4.0 International License
Open Source OpenVX & Samples

Open Source OpenVX 1.3
Fully Conformant on Raspberry Pi
Raspberry Pi 3 and 4 Model B with Raspbian OS
Memory access optimization via tiling/chaining
Highly optimized kernels on multimedia instruction set
Automatic parallelization for multicore CPUs and GPUs
Automatic merging of common kernel sequences
Supports NNEF Import Feature Set

“Raspberry Pi is excited to bring the Khronos OpenVX 1.3 API to our line of single-board computers. Many of the most exciting commercial and hobbyist applications of our products involve computer vision, and we hope that the availability of OpenVX will help lower barriers to entry for newcomers to the field.”

Eben Upton
Chief Executive Raspberry Pi Trading

Open Source OpenVX Tutorial and Code Samples
https://github.com/rgiduthuri/openvx_tutorial
https://github.com/KhronosGroup/openvx-samples
https://github.com/kiritigowda/OpenVX/tree/master/digitClassification#digit-classification
Vulkan SC 1.0 Design Philosophy

Vulkan 1.2 is a compelling starting point
- Widely adopted, royalty-free open standard
- Explicit control of device scheduling, synchronization and resource management
- Smaller surface area than OpenGL
- Not burdened by runtime debug functionality
- Very little internal state
- Well-defined thread behavior
- Ingests SPIR-V IR - no runtime front-end compiler

Streamlined
- Remove non-essential runtime functionality
  - Sparse memory
  - Descriptor update templates
  - Certain types of object deleters

Deterministic
- Predictable execution times and results
  - Offline compilation of pipelines
  - Static memory allocation

Robust
- Removing Ambiguity
  - No ignored parameters or undefined behaviors
  - Enhanced fault handling and reporting functionality

Testable
- Open-source Conformance Test Suite
  - Freely available under Apache 2.0 open-source license
  - Leverages 1 million+ test Vulkan test suite with added SC-specific tests
  - Confirms and documents Vulkan SC implementation compatibility

Vulkan SC reduces cost and effort for to produce evidence packages for system certification
Vulkan SC can be invaluable for real-time embedded applications, even if not formally safety-certified

This work is licensed under a Creative Commons Attribution 4.0 International License
Vulkan SC Offline Compiled Pipelines

A Vulkan Pipeline defines how the GPU processes data

- SPIR-V
- JSON Pipeline Description

Lists all SPIR-V modules used with related state

Implementation-Specific Pipeline Cache Compiler (PCC)

Extracts information from pipeline cache files to analyze dataflow and the amount of memory used by the processing in the pipeline

Pipeline Cache Utility

Memory for pipelines is reserved at device at device creation time as fixed size pools. Similarly sized pipelines can be assigned to the same pool to minimize memory size and fragmentation

Application

Vulkan SC Precompiled Pipelines enable run-time determinism
Eliminates need for runtime memory allocation

Offline
Runtime
OpenVX SC Profile

Minimizes Run-time Surface Area and Implementation Size
Eases system-level safety certification
Separated Development and Deployment environments

Robust Specification
Annotated specification with Functional Requirement tag numbers
MISRA-C compliant headers

The OpenVX SC profile combined with ingestion of trained Neural Networks enables OpenVX as a cross-platform inferencing engine for safety critical markets
SYCL Safety-Critical Exploratory Forum

Exploring real-world industry requirements for open and royalty-free high-level compute APIs suitable for safety-critical markets

**Khronos SYCL Safety-Critical Exploratory Forum**

Online discussion forum and weekly Zoom calls

No detailed design activity to protect participants IP

Explore if consensus can be built around an agreed Scope of Work document

Discuss what standardization activities can best execute actions in the Scope of Work

Any company is welcome to join

No cost or IP Licensing obligations

Project NDA to cover Exploratory Forum Discussions

More information and signup instructions
https://www.khronos.org/syclsc

Initiation of Khronos Working Group to execute the SOW

Agreed SOW document released from NDA and made public

Proven Khronos Exploratory Process to ensure industry requirements are fully understood before starting standardization initiatives